

การวางแผนการเปลี่ยนการจับวัดถูรูปหลายเหลี่ยม วัดถูทรงหลายหน้า และวัดถูที่ถูกกำหนดด้วยจุดสัมผัส



นายชนะธร พ่อคำ

ศูนย์วิทยพัทยากร  
จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรดุษฎีบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2553

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

REGRASP PLANNING FOR POLYGONAL, POLYHEDRAL AND DISCRETE OBJECTS



Mr.Thanathorn Phoka

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

A Dissertation Submitted in Partial Fulfillment of the Requirements  
for the Degree of Doctor of Philosophy Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

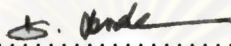
Academic Year 2010

Copyright of Chulalongkorn University

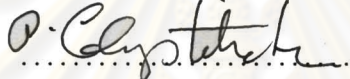
Thesis Title           REGRASP PLANNING FOR POLYGONAL, POLYHEDRAL  
AND DISCRETE OBJECTS  
By                        Mr.Thanathorn Phoka  
Field of Study        Computer Engineering  
Thesis Advisor       Assistant Professor Attawith Sudsang, Ph.D.


---

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


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


THESIS COMMITTEE

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

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

  
..... Examiner  
(Professor Boonserm Kijirikul, Ph.D.)

  
..... Examiner  
(Associate Professor Ratchatin Chanchareon, Ph.D.)

  
..... External Examiner  
(Assistant Professor Worasait Suwannik, Ph.D.)

ธนธรร พอค้ำ: การวางแผนการเปลี่ยนการจับวัตถุรูปหลายเหลี่ยม วัตถุทรงหลายหน้า และวัตถุที่ถูกกำหนดด้วยจุดสัมผัส. (REGRASP PLANNING FOR POLYGONAL, POLYHEDRAL AND DISCRETE OBJECTS) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผู้ช่วยศาสตราจารย์ ดร.อรรถวิทย์ สุดแสง, 119 หน้า.

วิทยานิพนธ์ชิ้นนี้พิจารณาปัญหาการวางแผนการเปลี่ยนการจับเพื่อจัดวัตถุรูปหลายเหลี่ยม วัตถุรูปหลายเหลี่ยมที่ประกอบด้วยด้านจำนวนหลายด้าน วัตถุทรงหลายหน้า และวัตถุที่แยกเป็นส่วนซัด วิทยานิพนธ์นี้ได้นำเสนอวิธีในการคำนวณหาลำดับการเปลี่ยนตำแหน่งของนิ้วเพื่อเปลี่ยนจากการจับหนึ่งไปยังการจับอื่นโดยยังรักษาคุณสมบัติปิดของแรงตลอดกระบวนการ วิธีที่นำเสนอขึ้นอยู่กับพื้นฐานของการค้นหาในโครงสร้างที่เรียกว่ากราฟการสับเปลี่ยนการจับ โดยที่ในแต่ละจุดในกราฟเก็บเซตของการจับที่ถูกคำนวณไว้แล้วในกรณีของวัตถุรูปหลายเหลี่ยม และวัตถุทรงหลายหน้า

สำหรับวัตถุที่แยกเป็นส่วนซัด เนื่องจากข้อมูลของจุดจับมีจำนวนมาก การคำนวณโดยสมบูรณ์ไม่สามารถนำมาใช้แก้ปัญหาได้ วิทยานิพนธ์นี้จึงนำเสนอวิธีการจัดกลุ่มของจุดจับตามความสามารถในการออกแรงและแรงบิด จุดจับตัวแทนได้ถูกเลือกมาจากแต่ละกลุ่มเพื่อนำมาสร้างกราฟครอบคลุม เนื่องจากการจับทั้งหมดไม่ได้อยู่ในกราฟครอบคลุมวิธีในการเชื่อมโยงการจับใดๆ ไปยังกราฟนี้จึงถูกนำเสนอไว้ด้วย

ความเชื่อมต่อกันในโครงสร้างกราฟแสดงถึงความสามารถในการเปลี่ยนจากการจับหนึ่งไปการจับอื่น ซึ่งทำให้ปัญหาการวางแผนการเปลี่ยนการจับสามารถแก้ได้ด้วยการค้นหาในกราฟ และโปรแกรมเพื่อจำลองวิธีที่นำเสนอนี้ได้ถูกพัฒนาขึ้นเพื่อแสดงผลการทดลอง

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

ภาควิชา ..... วิศวกรรมคอมพิวเตอร์ ..... ลายมือชื่อนิสิต ..... พงศ์ .....  
สาขาวิชา ..... วิศวกรรมคอมพิวเตอร์ ..... ลายมือชื่ออ.ที่ปรึกษาวิทยานิพนธ์หลัก .....  
ปีการศึกษา ..... 2553 .....

## 4771814021: MAJOR COMPUTER ENGINEERING

KEYWORDS: FORCE CLOSURE / DISCRETE CONTACT POINTS / REGRASP PLANNING / 2D GRASPING / 3D GRASPING / GRASP HEURISTIC / DEXTEROUS MANIPULATION


THANATHORN PHOKA : REGRASP PLANNING FOR POLYGONAL, POLYHEDRAL AND DISCRETE OBJECTS. THESIS ADVISOR : Assistant Professor Attawith Sudsang, Ph.D., 119 pp.

This dissertation addresses the problems of planning a set of regrasp sequences for manipulating a polygon, a polygon with a large number of edges, a polyhedron and a discrete object. Assuming frictional point contact, we propose approaches for computing sequences of finger repositioning that allow the hand to switch from one grasping configuration to another while maintaining force closure during the entire process. The proposed approaches are based on exploring a structure called switching graph. For a polygon or a polyhedron, a vertex in a switching graph explicitly contains a set of force closure grasps.

In contrast, for a discrete object, the input with a large number of discrete contact points is considered. In this setting, traditional methods of complete solution is not available. Based on wrench space information of the input, our proposed algorithm clusters the input into groups and chooses a representative from each group. A global graph structure for regrasp planning is then constructed using all force closure grasps that can be formed only by representatives. Also described are approaches for finding a regrasping sequence from an arbitrary grasp to a grasp in the global structure.

The connectivity of a graph captures ability to switch from one grasp to another and allows regrasp planning to be formulated as a graph search. The proposed approaches have been implemented and computational results are presented.

Department: ... Computer Engineering ...

Student's Signature 

Field of Study: .. Computer Engineering ..

Advisor's Signature 

Academic Year: .....2010.....

## Acknowledgements

I would like to express my gratitude to my advisor, Dr. Attawith Sudsang who has patiently tutored me from the very start of my study. Dr. Sudsang not only shapes my work but also inspires many of my thought. I also express my thankfulness to my dissertation committee: Dr. Prabhas Chongstitvatana, Dr. Boonserm Kijirikul, Dr. Ratchatin Chanchareon and Dr. Worasait Suwannik.

I also would like to thank all people, past and present, in the department of computer engineering, the Intelligent System Laboratory 2 (ISL2) at Chulalongkorn University. I appreciate the financial support from the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program under Grant No. Ph.D. 1.O.CU/49/D.1 and the 90<sup>th</sup> Anniversary of Chulalongkorn University Fund through the Ratchadapiseksomphot Fund.

I would like to thank my family for their unconditional understanding, extensive support and unceasing love which nourishes me when I need it most.



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

# Contents

	Page
<b>Abstract (Thai)</b> . . . . .	iv
<b>Abstract (English)</b> . . . . .	v
<b>Acknowledgements</b> . . . . .	vi
<b>Contents</b> . . . . .	vii
<b>List of Tables</b> . . . . .	xi
<b>List of Figures</b> . . . . .	xii
<b>Chapter</b>	
<b>I Introduction</b> . . . . .	<b>1</b>
1.1 Related Works . . . . .	3
1.1.1 Robot Hands . . . . .	4
1.1.2 Contact Kinematics, Dynamic and Control of Manipulation . . . . .	4
1.1.3 Grasp Definition . . . . .	6
1.1.4 Force Closure . . . . .	7
1.1.5 Regrasp Planning . . . . .	9
1.1.6 Dexterous Manipulation Planning . . . . .	12
1.2 Problem Statement . . . . .	14
1.2.1 Contribution . . . . .	14
1.3 Dissertation Outline . . . . .	14
<b>II Grasping and Regrasping Preliminaries</b> . . . . .	<b>15</b>
2.1 Nomenclatures . . . . .	15
2.2 Contact Model . . . . .	15
2.3 Grasp and Wrenches . . . . .	16
2.3.1 Primitive Contact Wrenches . . . . .	17
2.3.2 Grasp Wrench Set . . . . .	18
2.4 Force Closure . . . . .	18
2.5 Condition of Force Closure . . . . .	19
2.6 Regrasping . . . . .	20
2.6.1 Finger Switching and Finger Sliding . . . . .	21
<b>III Regrasp Planning for a Polygonal Object</b> . . . . .	<b>22</b>
3.1 Introduction . . . . .	22
3.2 Force-closure conditions in 2D . . . . .	23

Chapter	Page
3.3 Switching Graph for a Polygonal Object . . . . .	25
3.3.1 Representing Force-closure Grasps . . . . .	25
3.3.1.1 Representing Concurrent Grasps . . . . .	26
3.3.1.2 Representing 2-finger Grasps . . . . .	28
3.3.1.3 Representing Parallel Grasps . . . . .	28
3.3.2 Finger Switching . . . . .	32
3.3.2.1 Finger Switching among Concurrent Grasps . . . . .	33
3.3.2.2 Finger Switching between 2-finger Grasps and Concurrent Grasps . . . . .	33
3.3.2.3 Finger Switching among Parallel Grasps . . . . .	34
3.3.2.4 Finger Switching between Parallel Grasps and 2-finger Grasps . . . . .	35
3.3.3 Finger Aligning . . . . .	36
3.3.4 Computing Switching Graph . . . . .	38
3.3.4.1 Computing Vertices of Concurrent Grasps . . . . .	38
3.3.4.2 Computing Vertices of 2-finger Grasps . . . . .	40
3.3.4.3 Computing Vertices of Parallel Grasps . . . . .	40
3.3.4.3.1 . . . . .	41
3.3.4.4 Computing Edges . . . . .	42
3.4 Using Switching Graph . . . . .	42
3.5 Implementation and Results . . . . .	44
3.6 Summary . . . . .	46
<b>IV Regrasp Planning for a Polygon with a Large Number of Edges . . . . .</b>	<b>51</b>
4.1 Introduction . . . . .	51
4.2 Representing force closure grasps . . . . .	51
4.2.1 Computing $G_{i,j}$ . . . . .	53
4.2.2 Extending Configuration Space . . . . .	54
4.2.3 Constructing $G$ . . . . .	56
4.3 Finger Switching . . . . .	59
4.4 Finger Aligning . . . . .	60
4.5 Constructing Switching Graph . . . . .	61



Chapter	Page
4.6 Using Switching Graph . . . . .	62
4.6.1 Unconstrained Regrasp Sequence . . . . .	62
4.6.2 Optimal Regrasp Sequence . . . . .	63
4.7 Experimental Results . . . . .	66
4.8 Summary . . . . .	68
<b>V Regrasp Planning for a Polyhedral Object . . . . .</b>	<b>70</b>
5.1 Introduction . . . . .	70
5.2 Force-closure conditions in 3D . . . . .	70
5.3 Switching Graph for a Polyhedral Object . . . . .	71
5.3.1 Representing Concurrent Grasps . . . . .	72
5.3.2 Finger Switching . . . . .	74
5.3.3 Finger Aligning . . . . .	74
5.3.4 Computing a Switching Graph . . . . .	75
5.3.4.1 Direct Geometric Computation . . . . .	79
5.3.4.2 Random Sampling . . . . .	79
5.4 Implementation and Results . . . . .	80
5.5 Summary . . . . .	81
<b>VI Regrasp Planning for a Triangular-Mesh Object . . . . .</b>	<b>88</b>
6.1 Introduction . . . . .	88
6.2 Regrasp Planning on Discrete Point Set . . . . .	89
6.2.1 Overview . . . . .	90
6.2.2 Spectral Clustering for Contact Point Set . . . . .	90
6.2.2.1 Affinity Matrix . . . . .	91
6.2.2.2 Spectral Clustering Algorithm . . . . .	93
6.2.3 Constructing Representative-Level Roadmap . . . . .	94
6.2.4 Planning Regrasp Sequence . . . . .	94
6.3 Experiments and Results . . . . .	97
6.4 Summary . . . . .	99
<b>VII Conclusion . . . . .</b>	<b>106</b>
7.1 Dissertation Summary . . . . .	106
7.2 Further Improvement and Extension . . . . .	107

Chapter	Page
7.3 Discussion . . . . .	109
<b>Biography . . . . .</b>	<b>119</b>



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

## List of Tables

Table	Page
3.1 Results of the algorithm for all grasp types . . . . .	46
3.2 Combined results . . . . .	46
4.1 Switching graph construction of 10° half-angle . . . . .	67
4.2 Switching graph construction of 15° half-angle . . . . .	68
4.3 Switching graph construction of 20° half-angle . . . . .	68
5.1 Results from direct intersection approach . . . . .	82
5.2 Results from random sampling approach for each test object with 1,000 sampling points . . . . .	82
5.3 Results from random sampling approach for each test object with 5,000 sampling points . . . . .	82
5.4 Results from random sampling approach for each test object with 10,000 sampling points . . . . .	83
5.5 Results from random sampling approach for each test object with 20,000 sampling points . . . . .	83
6.1 Result of 500 mesh objects clustered into 30 clusters . . . . .	100
6.2 Result of 500 mesh objects clustered into 50 clusters . . . . .	100
6.3 Result of 500 mesh objects clustered into 70 clusters . . . . .	101
6.4 Result of 1000 mesh objects clustered into 30 clusters . . . . .	101
6.5 Result of 1000 mesh objects clustered into 50 clusters . . . . .	101
6.6 Result of 1000 mesh objects clustered into 70 clusters . . . . .	102
6.7 Result of local planning of 500 mesh objects clustered into 30 clusters . . . . .	102
6.8 Result of local planning of 500 mesh objects clustered into 50 clusters . . . . .	102
6.9 Result of local planning of 500 mesh objects clustered into 70 clusters . . . . .	103
6.10 Result of local planning of 1000 mesh objects clustered into 30 clusters . . . . .	103
6.11 Result of local planning of 1000 mesh objects clustered into 50 clusters . . . . .	103
6.12 Result of local planning of 1000 mesh objects clustered into 70 clusters . . . . .	104

## List of Figures

Figure	Page
2.1 Coulomb friction . . . . .	16
2.2 Regrasping overview . . . . .	21
3.1 Switching diagram . . . . .	25
3.2 Construction of a focus cell . . . . .	27
3.3 2-finger force-closure focus cell construction . . . . .	28
3.4 Construction of a common cone . . . . .	29
3.5 Three contact points forming a parallel grasp . . . . .	30
3.6 Representing a common cone . . . . .	31
3.7 Finger switching between concurrent grasps . . . . .	33
3.8 Finger switching between parallel grasps . . . . .	35
3.9 Finger switching between 2-finger grasps and parallel grasps . . . . .	36
3.10 Finger aligning . . . . .	37
3.11 Generating candidate triples of concurrent grasps . . . . .	40
3.12 Generating pairs of 2-finger grasps and triples of parallel grasps . . . . .	40
3.13 Computing a vertex of parallel grasps . . . . .	41
3.14 Using switching graph . . . . .	44
3.15 Test polygons . . . . .	45
3.16 A regrasp sequence for concurrent grasps . . . . .	47
3.17 A regrasp sequence for parallel grasps . . . . .	48
3.18 A regrasp sequence for concurrent grasps and 2-finger grasps . . . . .	48
3.19 A regrasp sequence for parallel grasps and 2-finger grasps . . . . .	49
3.20 A regrasp sequence for all grasp types . . . . .	50
4.1 Computing $G_{i,j}$ . . . . .	54
4.2 Extreme points of $G_{i,j}$ . . . . .	55
4.3 Independent contact region mapping . . . . .	55
4.4 Extending configuration space . . . . .	57
4.5 Adjacencies of vertices . . . . .	58
4.6 Finger switching . . . . .	60
4.7 Finger aligning . . . . .	60
4.8 Using switching graph . . . . .	63
4.9 The largest square on Voronoi edges . . . . .	66
4.10 Determining a local optimum . . . . .	67
4.11 Test polygons . . . . .	67

Figure	Page
4.12 A regrasp sequence . . . . .	69
5.1 Inverted friction cone . . . . .	72
5.2 Union volume . . . . .	73
5.3 Moving between non-overlapping cells . . . . .	75
5.4 Moving within a focus cell . . . . .	76
5.5 Parameterization of a unit normal vector . . . . .	77
5.6 Mapping from the spherical to cartesian coordinates . . . . .	78
5.7 Two bounding boxes . . . . .	79
5.8 Test objects . . . . .	81
5.9 Shaded test objects . . . . .	84
5.10 A regrasp sequence of the object in Fig. 5.8(b) . . . . .	85
5.11 A regrasp sequence of the object in Fig. 5.8(f) . . . . .	86
5.12 A regrasp sequence of the object in Fig. 5.8(g) . . . . .	87
6.1 Transformation distance . . . . .	92
6.2 Test objects . . . . .	98
6.3 A regrasping sequence for the object in Fig. 6.2(b) with 500 triangles clustered into 50 groups . . . . .	104
6.4 A regrasping sequence for the object in Fig. 6.2(d) with 1000 triangles clustered into 30 groups . . . . .	105

# CHAPTER I

## INTRODUCTION

The ability to manipulate objects is one of the fundamental tasks which we need a robot to interact with its surroundings. Grasping and regrasping are operations which a human performs to change the environment by grabbing an object, lifting and placing it to another position or another posture. This seems to be a natural and simple ability for a human but not for a robot. The robot does not have its own instinct to perform any task. It has to be controlled restrictively on a task, the geometry of an object, the constraints of itself and the environment. This challenges robotic researchers to analyze and transfer these manipulation skills to a robot in the recent decades.

The central idea of manipulation is to move an object to a desired configuration while the object being restrained in stable states. One approach of object manipulation is grasping an object in a fixed stable grasping configuration and then moving the object from place to place. However, the pick and place operation is performed by a motion sequence of an arm which requires a large workspace to change the grasping configuration even for manipulating a small object. Further, it requires a stable placement for the object when the robot changes a grasping configuration. To avoid these limitations, an in-hand manipulation is taken place to bring the object through several actions by changing grasping configurations without releasing the grasped object. The operation of changing a grasping configuration is usually called *regrasping*. To achieve a manipulation task, the fingers have to be moved several times to reach the target posture. This arises the *in-hand manipulation planning problem*, given an initial grasp and a target grasp, the goal is to compute the sequence of the fingers' movements which changes the grasp to the target position while still maintaining stability. For manipulation involving grasping, to verify stability of a grasp, the *force-closure property* is usually considered in several literatures dealing with the grasp synthesis problem.

In a manipulation, we have to consider all constraints arising from a task and a robot hand. Despite planning a manipulation sequence considering all these constraints could accomplish a task, there are some kinds of arising restrictions and drawbacks. All complexities are gathered in the manipulation planning that might use non-reasonable time to compute a simple manipulation sequence. Further, combining mechanical constraints of a robot hand in planning confines the result manipulation sequences to the specific robot

hand platform. Therefore, the manipulation sequences may be not feasible for another hand platform.

The problem is that, currently, we have so many tasks, so many objects and so many robot hands. There is no algorithm that works best on all settings. This is mainly because tasks, objects and robot hands are varying. The problem of task/object/hand dependency is the inspiration of this work. We propose to decompose the manipulation problem into layers. We split the problem into three main levels, each of which considers task constraints, object constraints and hand constraints separately. It allows a hardware practitioner to concentrate on creating a hand. At the same time, we can imagine any use of robot in manipulation. The advantage of the decomposition is that robot visionaries can abstract their manipulation algorithms away from constraints and limitations of the recent robot hand developments.

We aim to derive a framework that shall be applicable to any task, any object or any robot hand. The key concept is simple; a task and a hand impose on manipulation planning many constraints, which we simply decide to neglect them. Without any assumption on a task or a hand, solutions from the algorithm will be dependent only on the object being manipulated, not with any predefined hand or task. After solutions are identified, when knowledge of a hand and a task is provided, we then find solutions that satisfy the arising constraints.

Given an object to be manipulated, a condition that can be verified without considering constraints of a task and a robot hand is the force-closure condition because we can consider only contact positions of the end effectors which are applicable only for force-closure verification. This reduces our consideration into the problem of planning a sequence of the end effectors from an initial grasp to a target grasp while all grasps in the sequence maintain force-closure. We call a sequence of changing the end effectors' positions *regrasp sequence*. The problem of determining such sequence is referred to as *regrasp planning problem*.

This work studies the problem of regrasp planning which computes a sequence of finger repositioning from initial grasping configuration to a desired configuration for polyhedral-modelled object and a set of discrete points based on the following assumptions. The polyhedral model is chosen because most objects in the real world can be represented by linear surfaces. An object is described by linear segments in 2D and flat surfaces in 3D. Further, since the representation of an object is linear, we can efficiently

solve the regrasp planning problem using existing linear algebra and computational geometry algorithms. For a complex object, we describe it by a set of discrete points on its surface. Approaches to handle this problem setting in both 2D and 3D workspace are also proposed.

The hands are assumed to be equipped with three or four in 2D workspace and five fingers in 3D workspace. 2- and 3-finger grasp are sufficient to grasp a 2D object and 4-finger grasp is sufficient to grasp a 3D object. The other one finger is used to switch grasping position. Our planner aims to construct general solution satisfying grasping constraints regardless task constraints, kinematic constraints, dynamic constraints, etc. The most advantage of general solution is independency. It is applicable to any task or hand in the real world. A finger is therefore assumed to be a free-flying point contact. To maintain stability, grasping constraint considered in this work is associated with force closure property. Every grasping configuration in the obtained sequence of finger repositioning has to satisfy force closure property to ensure stability during the entire repositioning process.

## 1.1 Related Works

Regrasp problem consists of various problems in many subfields on robotics. Firstly, we have to define what we want the robot to do. This is according to task constraints. Based on the classification of grasp by Cutkosky (1989), two main grasping types are concerned, fingertip grasp and power grasp. Fingertip grasps achieve dexterity by holding the objects by the tips of the fingers. Power grasps are distinguished by large areas of contact between the object and the fingers and palm which do not allow the motion of the grasped object. The grasps perform with low dexterity. For regrasping, the fingertip grasps are preferred since the problem required dexterity of grasps.

Regrasp planning is the main theme of this work. The method reports a sequence of fingers' position from initial grasp to desired grasp given by task planner. The obtained grasping positions not only associate with task requirement but also satisfy stability constraint. The force closure property is applied to satisfy the stability constraint. This means that every grasping position in a sequence calculated by the method has to achieve force closure grasp.

In practice, a robot finger is not a point. A grasp has to satisfy kinematic constraints and dynamic constraints as well. Motions of fingers when a regrasp process is performed, also introduce to a manipulation planning problem which mainly mentions accessibility



and collision avoidance of a path from an initial configuration to a desired one.

### 1.1.1 Robot Hands

Dexterous manipulation or regrasp problem require a manipulator which is able to change a grasped object's configuration with respect to the hand without releasing it. The robot hand is one suitable manipulator for this task. It may be designed to be an approximation of the human hand or specified for particular tasks. A well-known 3-finger robot hand is Barret Hand(Townsend, 2000) commercially made by Barrett Technology Inc. Two fingers can be spread synchronously by  $180^\circ$  around the palm. The Utah/MIT hand (Jacobsen et al., 1986) is the first anthropomorphic hand with four fingers. Each finger has four degrees of freedom. The whole hand system is very large including the out-hand actuators. The Robonaut hand (Lovchik and Diftler, 1999) designed for space based operations has five fingers. The hand combined with wrist and forearm has fourteen degrees of freedom. Another anthropomorphic hand is the DLR-Hand (Butterfaß et al., 2001). The hand consists of four fingers with the actuators embedded inside.

### 1.1.2 Contact Kinematics, Dynamic and Control of Manipulation

When the object has been grasped, the hand is possible to perform in-hand manipulation. To gain more dexterity, the hand is not required to maintain a rigid grasp. It may therefore roll, slide or release and place fingers to change the grasp configuration. The accurate control of the force applied to the object, which associates with the contact constraints is required to achieve the operations.

One approach that the dexterous hand manipulates an object, is exploiting a rolling contact. Rolling is the operation that the fingertip rolls without slipping on the object's surface. It is defined by the constraints that the fingertip and object velocities are equal at contact point. The kinematic constraints and transformations between task-space and local coordinates are presented in (Kerr and Roth, 1986) and (Montana, 1988). The rolling constraints are formulated in different ways. Kerr and Rott (1986) derived the force analysis for the systems using a set of differential equations to describe the motion of the object with pure rolling contact. Montana (1988) proposed a method for relating relative rigid body motion to the rates of change of contact coordinates using a matrix formulation of the motion of a point of contact over the rolling surfaces. Sarkar *et al.* (1997) introduced local contact coordinates which allow them to formulate the dynamics and control of manipulation via rolling contacts in explicit equations relating the velocities and ac-

celerations of the contact points. The formulation admits motion of the contacts during the manipulation process. Li *et al.* (2000) developed a unified formulation describing the relationship between the object motion and the joint motion.

Dexterous manipulation sometimes exploits slippage between the fingers and the object to change grasping configuration. Sliding a finger along the surface of an object requires a good model of the contact friction which is mostly assumed Coulomb friction model. A finger exerts a force inward to the object's surface when it slides along the surface. According to Coulomb friction model, when the finger is sliding, the contact force must lie on the edge of the friction cone. Brock (1988) derived a kinematic relation between the object motion, the motion constraints and the grasp forces. Cole *et al.* (1989) presented a coordinated control law for sliding contacts between an object and fingertip including a problem of choosing contact positions for collision avoidance. In (Cole *et al.*, 1992), the sliding motion of the fingertips along the object's surface is dynamically controlled simultaneously with controlling the position and orientation of the held object. Zheng *et al.* (2000) formulated a dynamic control of a 3-finger robot hand manipulating an object in 3D. One finger is allowed to slide on the object's surface. Motion equations of the whole system are derived. They also proposed a dynamic control law for linearizing the system dynamics and realizing the desired object motion, the desired finger sliding and desired grasping force.

Combinations of rolling and sliding are in consideration as well. Cai and Roth (1987), (1988) studied spatial motions combining rolling and sliding between rigid bodies for point contact and line contact, respectively. Chong (1993) proposed an algorithm generating finite motion of object by considering sliding contacts as well as rolling contacts between the fingertips and the object. The minimum contact forces and minimum joint velocities are solved for the relative velocity at the contact point.

Forces applied to the object by the fingers are controlled for the desired manipulation. Kerr and Roth (1986) developed a hand Jacobian which calculates the joint torques from the desired contact forces. Yoshikawa and Nagai (1988) decomposed forces into two components. Manipulating or external forces produce a net force and torque on the object. The other forces are grasping or internal forces which produce no net force nor torque on the object. These forces are used to maintain a secure grasp. The same authors gave a physically reasonable definition of manipulating force and grasping force for 2-, 3- and 4-finger hands in (1991). They also presented an algorithm for decomposing a given fingertip force into manipulating and grasping forces. Using the concept of the manip-

ulating and grasping forces, they proposed a dynamic manipulation/grasping controller of multifingered robot hands based on the dynamic control and the hybrid position/force control. The controller consists of a compensator which linearizes the whole grasping system and a servo controller for the linearized system. Nakamura *et al.* (1989) discussed the dynamical coordination of a multifingered robot hand. The coordination problem is solved in two phases. Firstly, determine the resultant force used for maintaining dynamic equilibrium and for generating the restoring force. Secondly, determine the internal force used to satisfy the static frictional constraints and is related to contact stability. Li *et al.* (1998) studied a formulation of dynamic stability of grasping using Lapunov stability theory for measurement purpose.

The systems discussed above are formed by complex constraints. A system that a manipulation is achieved by low velocity motions is called quasi-static. Quasi-static analysis results are therefore much simpler and practical. Fearing (1986) considered slip from a quasi-static viewpoint to achieve grasp stability. Yoshikawa *et al.* (1993) used controlled slip in quasi-static system to modify the grasp and increase manipulation range for a 3-finger robot hand.

### 1.1.3 Grasp Definition

Secure holding an object in a robot hand is required in grasping. The concept of a firm grasp is formalized in various ways. Equilibrium, force closure and form closure property are usually applied to ensure the stability of a grasp. Equilibrium grasp is a grasp that the resultant of forces and torques exerted to the grasped object are zero. According to the definition, an equilibrium grasp cannot resist any disturbance. This property is therefore not sufficient to ensure the stability of a grasp. Force closure grasp is a grasp that can exert a resisting force and torque balancing any external disturbance on the object. A closely related property to force closure is form closure firstly investigated by Reuleaux (1963). The distinction between form closure and force closure is that form closure considers the immobility of an object in presence of fixed contact points whereas force closure considers how contact points can exert force and torque on an object. Another difference between form and force closure is the presence of friction. Friction effect is considered in force closure while it is neglected in form closure analysis. Markenscoff *et al.* (1990) provided an upper bound to the number of contacts necessary to achieve form closure grasps. They showed that four contact points are sufficient for the form-closure grasp of any planar object and seven contact points are sufficient in spatial case. Bicchi (1995) considered form closure as a purely geometric property of a set of contact

constraints. Rimon and Burdick (1996) gave precise definitions for first and second order form closure for frictionless grasps based on mobility theory. They also showed that a frictionless grasp is force closure if and only if it is form closure for both first order and second order.

#### 1.1.4 Force Closure

To ensure that the object is grasped securely, the classical force closure condition is employed. A grasp of an object achieves force closure when it can resist any external wrench exerted on the grasped object. The well-known qualitative test for a force closure grasp is to check whether the contact wrenches of the grasp positively span the whole wrench space (Salisbury, 1982). This is equivalent to checking whether the convex hull of the primitive contact wrenches contains the origin (Mishra et al., 1987b). Various approaches for testing whether the origin is inside the convex hull are proposed. Yun-Hui Liu (1998) proposed a recursive reduction technique which allows the problem of testing convex hull containing the origin in high dimensions to be solved in the lowest dimension. The same authors transformed this problem to ray-shooting which can be solved by linear programming (Liu, 1999). Zhu and Wang (2003a) developed the force closure test based on the concept of  $Q$  distance which uses a convex hull containing the origin as a metric to test whether the origin lies in the interior of the convex hull of the primitive wrenches. Recently, Zhu *et al.* (2004) discussed that the problem can be transformed into the problem of calculation of distance between convex objects. They proposed the use of pseudodistance function to solve the problem.

Other approaches of qualitative test for a force closure grasp by considering the workspace, not the wrench space, were also investigated. Nguyen (1988b) proposed a geometric method for testing 2-finger force closure grasps on polygonal objects. The synthesis of stable grasps was proven by constructing virtual springs at the contact points, such that a desired stiffness matrix about its stable equilibrium can be acquired. Ponce *et al.* proposed the concept of non-marginal equilibrium which implies the force closure property. Based on this concept, the qualitative tests of 3-finger grasps for polygonal objects (Ponce and Faverjon, 1995a) and 4-finger grasps for polyhedral objects (Ponce et al., 1997) were proposed.

For regrasping, a set of force closure grasps has to be calculated. In (Ponce and Faverjon, 1995a) and (Ponce et al., 1997), a grasp is represented by parameters related to positions on the grasped faces. To calculate all possible grasps, two(three) additional

parameters are required to construct linear constraints for 2D(3D) case. The additional parameters have to be eliminated to acquire a set of force closure grasping positions on given grasped faces. Sudsang and Ponce (1995) proposed another representation of grasps avoiding the use of additional parameters. A point in workspace is used to represent a set of force closure grasps.

Quantitative tests of force closure grasps are also considered to define the quality of grasps. Kirkpatrick *et al.* (1990) considered the most general stability measurement which does not know a priori knowledge of disturbance. An external wrench is assumed to be uniformly distributed in every direction. The minimum magnitude of a particular external wrench that breaks force closure property is measured. This is equivalent to the radius of the maximal ball that can fit inside the convex hull of primitive contact wrenches. Ferrari and Canny (1992) applied this criterion to plan the optimal grasp. The radius of maximal ball is used in many works, such as (Mirtich and Canny, 1993; Borst *et al.*, 2003; Jia, 1995).

Recently, the best performance in resisting external wrenches as the optimality criterion is still studied. Yun-Hui Lui (1999) addressed the problem of minimizing the  $L_1$  norm of the grasp forces in balancing an external wrench, which can be transformed to ray-shooting problem. Zhu and Wang (2003a) addressed the problem of planning optimal grasps that minimize the  $Q$  distance and expresses the best performance in firmly holding an object while resisting external wrench loads. Zhu *et al.* (2004) solved the same problem by optimizing the pseudodistance function.

Methods mentioned above are used to determine grasps that require precision of fingertip on the objects. To allow some positioning errors, the notion of *independent contact regions* was introduced by Nguyen (1988b). In short, an independent contact region is a parallel-axis rectangular region in fingers' configuration space which represents areas on object's boundary where fingers can be placed independently to compose a force closure grasp. In (Nguyen, 1988b), Nguyen also showed how to geometrically determine independent contact regions for 2-finger grasps of a polygon. Tung and Kak (1996) attacked the completeness of the previous work and proposed an algorithm which is correct and complete. Recently, Cornella and Suarez investigated an algorithm of determining independent grasp regions on 2D discrete objects (Cornella and Suarez, 2005a). A four frictionless grasp is considered. The algorithm determines the independent regions of two fingers when the locations of the other two fingers are given.

In order to find the *best* independent contact region, one needs to define what *best* means. There have been many different definitions of the best independent contact region due to different purposes and constraints of grasping devices. The two popular criteria are: (1) the largest n-cube, and (2) the largest rectangular region (product of lengths on every axis). Using the first criterion, the optimization can be done by linear programming as discussed in (Ponce and Faverjon, 1995a) and (Ponce et al., 1997). Faverjon and Ponce (1991) tackled the problem of 2-finger grasping on curved objects using the second criterion. In their work, a numerical optimization algorithm was presented, but they could not guarantee the algorithm's completeness. Cornella and Suarez (2005b) presented an approach to determine independent contact regions on polygonal objects considering arbitrary number of friction or frictionless contacts on given edges. Their approach subdivides configuration space so that the graspable region in each subdivision is convex, then computes the independent contact region in each subdivision.

### 1.1.5 Regrasp Planning

Regrasp or dexterous manipulation is required when a grasp is not appropriate for a specific task. A planner calculating a sequence of feasible configuration of robot hand and object transforming to the desired one is applied to solve the problem. The obtained results from a planner have to satisfy constraints considered in the system. The distinction between various planners are constraints discussed above, kinematics, dynamic, stability constraints, etc. In this work, force closure constraint is satisfied only for more general results. Some different planners are discussed here.

Hong *et al.* (1990) proved the existence of two and three finger grasps for 2D and 3D objects assuming isolated hard point contacts with friction. The manipulated objects are assumed to be smooth. This paper also proposed a fine motion of an object by repositioning the grasping fingers while maintaining a grasp during entire process. A subclass of fine motion problem focused in this paper is gait problem. Finger gaits with three and four fingers on the plane are proven for the existence. For the prove of three finger gait, a two finger force closure condition is taken into consideration. In the case of four finger gait, two different gaits can behave which are using two pairs separately or using a three finger grasp and replacing one finger with the remaining finger to form a new grasp.

Regrasp planning for reorientation of a prism was addressed by Omata and Nagata (1994). The 4-finger hand and frictional contact point are assumed. The planner plans

a sequence of repositioning of fingers for horizontal rotation of an object for a desired angle. The calculation of finger repositioning are classified into three problems. Problem  $A(c)$  tests whether the finger  $c$  can be removed from the initial grasp. This problem can be solved by linear programming method. Problem  $B(c, n)$  is solved for calculating feasible region of finger  $c$  to form equilibrium grasp without finger  $n$ . The last one is problem  $C(c, n, d)$  which calculates the feasible region of finger  $c$  when finger  $c$  and  $n$  form a grasp without finger  $d$ . These two problem can be solved by non-linear programming. Problem  $C$  is harder and takes more calculation time than  $B$ . Sequences of finger repositioning are attained by a search tree. Each node represents a removed finger. The search algorithm begins with solving problem  $A(c)$  then solves  $B(c, n)$  to remove finger  $n$  and bring finger  $c$  to form a grasp. Problem  $C$  will be solved when the problem  $B$  cannot produce feasible solution. Child nodes are expanded according to a heuristic function. The function is based on a angle which a grasp can rotate the object, the depth of a node and the penalty when problem  $C$  has to be solved.

Omata and Farooqi (1996) studied object reorientation by using regrasp primitive. Two primitives are carried out for reorientation task. The *rotation* presented in (Omata and Nagata, 1994) is a primitive that the fingers grasp on the side faces of the object and rotate it. The *pivoting* primitive uses the two fingertips to form an axis of pivoting and the third finger exerts the force on the side facts to rotate the object about the axis. The algorithm of this primitive is explained in this paper. Based on the following assumptions, four fingered hand and a prism object, sequential executions of these primitives can achieve reorientation. The search tree is applied to solve the problem. Each branch represents a primitive and each node contains the current orientation. The search procedure uses quaternion concept to solve resultant rotation about a unique axis.

An approach to solve the problem of dexterous manipulation using geometrical reasoning techniques was proposed by Munoz *et al.* (1995). Kinematic constraints are respected by checking non-penetration between the fingertips and the object. Some accessibility limitations due to the kinematic constraints of the hand are also considered. Three manipulation modes, which are fixed-point, rolling and sliding, are applied in the planning algorithm. A combination of manipulations in these three modes can form a nominal trajectory of a task that the object is being grasped by a dexterous hand. A manipulation task is represented by a homogeneous transformation that brings the object from its initial configuration to its final configuration. The planner decomposes the transformation into a sequence of infinitesimal motions by exploring the space of potential solutions for the

problem of changing the orientation of the grasped object. Each infinitesimal motion is solved for every manipulation mode. The equilibrium constraints are considered in this procedure. A solution is represented in the form of joint motion. The minimum joint motion is selected by the planner for the particular infinitesimal motion.

Leveroni (1997) addressed finger gait problem for a planar convex object. One method to determine whether local motions will suffice to reorient the object is the grasp map, a graphical representation of all stable grasps. Workspace map is constructed to determine workspaces of three fingers. A sequence of finger gaits can be extracted from the combination of the grasp map and the workspace map. In planning, a new grasp cannot always be found if the object is moved locally until a finger reaches a workspace limit; often a grasp gait must occur before the limit is reached.

In (Cherif and Gupta, 1997), The system of Cherif and Gupta assumed that the manipulation system processes at low velocities. Planning feasible quasi-static trajectories for the fingertips to move object to a desired configuration is available. Two motions which are rolling and sliding the fingertips on the surface of the object are considered. The planner is a 2-level planning scheme. The global planning level applies an  $A^*$  search algorithm to find connectivity between sub-goals in the configuration space of the object. The nominal path generated by this planner ignores any manipulation constraints. The second level is the local manipulation planner. The local planner is based on solving an *inverse finger motion problem* to plan for feasible quasi-static motions of the hand-object system between sub-goals. The instantaneous solution satisfies collision-free, reachability, friction and equilibrium constraints.

Han and Trinkle (1998b) proposed a Framework for dextrous manipulation by rolling fingers on the surface of an object and finger gaiting. Three taxonomies of manipulation tasks for multifingered hand systems are stated: *Object Manipulation*, *Grasp Adjustment* and *Dextrous Manipulation*. The contribution of this paper is to propose a general methodology to implement large-scale object manipulation tasks when the capability of the fingers are limited by their workspace. Two strategies, *finger rewind* and *finger substitution*, are applied to accomplish a task. Dextrous manipulation of a sphere is exemplified. The condition of two soft-finger and three hard-finger force closure grasp are derived for spherical object. The trajectory of the finger on the object is restricted to be a great circle which simplifies contact constraint.

Regrasp planning for discrete contact points using independent regions is proposed



in (Roa and Suarez, 2009). The regrasp operation that is allowed in the work is only motion of a finger without contact breaking. The main restriction of applying only regrasp operation is that the approach fails to find a path between two grasps in distinct connected grasp sets.

### 1.1.6 Dexterous Manipulation Planning

Since an object cannot move by itself. The robot hand has to grasp and move it from one stable position to another. The objective of the planner is to calculate a path of robot hand and object's configuration from an initial configuration to a desired configuration while avoiding collision with obstacles, other objects and self-collision.

Modelling the problem with as fully dynamic and using control-based planning is costly expensive. Thus, Alami *et al.* (1989) developed another approach using two distinct paths which are transfer paths and transit paths. The former are defined as motions of the system while the robot hand grasps the object. Transit paths are defined as motions of the robot when it moves alone while the object is in a stable position. Regrasping operation is also calculated by the planner. Based on this concept, Koga and Latombe (1994) solved the manipulation problem for robots with many degrees of freedom. The planner compute a series of transfer and transit paths for the robot that make the robot grasp and move the object from an initial configuration to a goal configuration. Recently, probabilistic algorithms are applied for manipulation planner under continuous grasps and placements in (Siméon *et al.*, 2002), (Sahbani *et al.*, 2002).

Nielsen and Kavraki (2000) developed a manipulation planner which extends the probabilistic roadmap (PRM) frameworks. The planner consists of two levels. The first level builds a manipulation graph. Nodes represent stable placements of the object. Edges represent transfer and transit actions. The actual motion planning for the transfer and transit paths is done by PRM planners at the second level. The fuzzy roadmap was introduced to apply in both levels. The computations is efficient by verifying that the edges are collision-free only if they are part of the final path. Instead, the local planner assigns a probability to the edge that expresses its belief that the edge is collision-free.

Sahbani *et al.* (2002) proposed a probabilistic algorithm for manipulation planning under continuous grasps and continuous object placements. Instead of classifying the regrasping operation as another subproblem, their approach transforms a regrasping operation into a finite sequence of transfer and transit paths. Therefore, a particular planner

for the regrasping operation is not needed.

Saut *et al.* (2007) attacked in-hand manipulation planning problem by using PRM. Two fundamental paths are applied which are transfer path and regrasp path. The object is immobile and some fingers move to change the grasp during a regrasp path. Based on PRM, a manipulation graph is constructed to plan a path between initial and goal configurations. Instead of sampling a hand's configurations in configuration space, grasping configurations are sampled over grasp subspaces and then verified chain closures at contact positions by considering the kinematics of the robot hand.

Xu and Li (2008) solved finger gait problem for a smooth surface object by evolution of hybrid automaton. The finger gaiting is analyzed into discrete and continuous characteristics. The discrete variables describe two actions of all fingers in either manipulation mode or substitution mode. The continuous variables represent the controls of the fingertips in continuous time. In (Xu *et al.*, 2007), the hybrid automaton is used for finger gait planning by improving the RRT approach such that the discrete metric and continuous metric are defined on the state space.

Huber and Grunpen (2002) presented finger gaits as finite state control strategies in a discrete event dynamic system framework. A small set of control laws are used as basis controllers to solve a manipulation task in a bottom-up fashion. However, actual contact locations and object motions are computed based on local contact information. Therefore, this framework suits for local manipulation planning. Platt *et al.* (2004) presented a control basis capable to generate a variety of force-based interaction focusing on the grasp and contact artificial potentials. Finger gaits are formulated into states and actions modeled in a Markov Decision Process (MDP) which is defined over the space of wrench closure conditions. However, this space is not explicitly computed. A state in the MDP is not a geometrical assertion but a report about the membership of grasps in the state.

Finger repositioning can be casted into a stratified system. Goodwine and Burdick (2002) proposed a nonlinear motion planning algorithm in a stratified configuration space. The configuration space of finger reposition consists of several smooth strata corresponding to the conditions of fingers used in manipulation. Harmati *et al.* (2002) developed a fitted stratified manipulation planning algorithm which works on a space that a fingertip position is described more directly to its representation in the real physical system. A semi-stratified was also proposed by assuming that a finger can be moved freely in the space to provide a greater degree of freedom for finger repositions in manipulation plan-

ning and to allow more constraints taken into account. However, most works about the stratified system study relations between a manipulated object and joint configurations or fingertip positions while force-closure condition is mostly assumed. Therefore, result trajectories obtained from these frameworks have to be verified for the force-closure condition for a practical use. Trajectories that do not achieve force-closure are not applicable in a real manipulation.

## 1.2 Problem Statement

Given an object (a polygon or a polyhedron or a set of contact points), an initial grasp and a goal grasp, we wish to identify a regrasp sequence from the initial grasp to the goal grasp.

### 1.2.1 Contribution

The contribution of this work is to propose a framework for regrasp planning problem. Our planner reports a general set of feasible finger repositioning satisfying force closure property for task and constraint independence. An approach using a structure called *Switching Graph* has been introduced. Connectivity in a graph presents ability to change a grasping configuration to another. This allows the regrasp planning to be transformed to graph search. A node in switching grasp represents a connected set of force closure grasps for given surfaces. Any grasps of which representations are in the same node can be transformed to one another using finger sliding along the continuous surfaces. An edge connecting two nodes indicates the ability of switching one finger to another different surface. Based on this structure, the obtained results are not a single solution, they are a set of feasible solutions. An advantage of a set of solutions is that it allows any planner to find a sequence of grasping positions which optimized according to some considered criteria or to add more constraints for practical uses.

## 1.3 Dissertation Outline

In the next chapter, we provide a theoretical preliminaries on grasping which is used subsequently in the remaining of the dissertation. The remaining chapters describe algorithms to solve the problem in each setting which are regrasp planning for a polygon, a polygon with a large number of edges, a polyhedron and a discrete contact point set.

Finally, Chapter 7 concludes our work and describes future extension of our work.

# CHAPTER II

## GRASPING AND REGRASPING PRELIMINARIES

In this chapter, we describe necessary definitions and propositions which will be applied in the discussion on our grasp planning problems.

### 2.1 Nomenclatures

Following the definitions in (Boyd and Vandenberghe, 2004), we denote by  $\text{INT}(\cdot)$ ,  $\text{RI}(\cdot)$  and  $\text{CO}(\cdot)$  the interior, the relative interior<sup>1</sup> and the convex hull of a set. For an arbitrary vector  $\mathbf{v}$ , let us denote by  $P_v$  the plane containing the origin and orthogonal to  $\mathbf{v}$ , i.e.,  $P_v = \{\mathbf{x} | \mathbf{x} \cdot \mathbf{v} = 0, \mathbf{x} \in \mathbb{R}^3\}$ . A point at  $\mathbf{x}$  is said to lie in the positive side of, negative side of, or exactly on  $P_v$  when  $\mathbf{x} \cdot \mathbf{v} > 0$ ,  $\mathbf{x} \cdot \mathbf{v} < 0$  or  $\mathbf{x} \cdot \mathbf{v} = 0$ , respectively. A closed half space  $\mathcal{H}(\mathbf{v})$  is the set of all points that lie exactly on  $P_v$  or in the positive side of  $P_v$ . An open half space  $\mathcal{H}^+(\mathbf{v})$  is simply  $\mathcal{H}(\mathbf{v}) - P_v$ . We define  $\mathcal{H}^{z+}$  to be  $\mathcal{H}^+((0, 0, 1))$  and  $\mathcal{H}^{z-}$  to be  $\mathcal{H}^+((0, 0, -1))$ .

### 2.2 Contact Model

In grasping, the most commonly used contact model are hard contact without friction, hard contact with friction and soft finger contact. Soft contact grasp is different from hard contact grasp with ability that soft finger can exert torque about the surface normal while hard finger can exert force at contact point only. For analysis of hard contact, the point contact without friction can only exert a unidirectional force normal to the surface. Tangential forces can be produced by a finger up to the friction coefficient when friction is considered.

Coulomb friction (Stewart, 2000) is usually applied for friction model. Coulomb's law of friction states that for a contact point exerting a force  $f_N$  along the contact normal, the friction force (the tangential contact force) is less than or equal to  $f_t = \mu f_N$  where  $\mu$  is the frictional coefficient. This equation indicates that when the contact is maintained without slip, the contact can exert any force in a cone  $C$  of which the half angle is equal to  $\tan^{-1}(\mu)$ . The cone is emanated from the contact point and the axis coincides with the contact normal  $n$ . This cone is commonly called a friction cone. Cone in 2D case can

---

<sup>1</sup>A relative interior of a set is the interior relative to the affine hull of the set. Intuitively speaking, a relative interior are all points not on the relative edge of the set, e.g., A relative interior of a line segment is the segment minus its endpoints, regardless of the dimension where the line is situated.

be expressed by two vectors as shown in Figure 2.1(a). In 3D case, a cone is described by quadratic function. Cone introduces complexity of nonlinearity to the problem. To simplify the problem, a cone can be replaced with an  $m$ -sided pyramid (Figure 2.1(b)). A pyramid has planar facets which avoid nonlinearity from the problem but at a price of lesser precision.

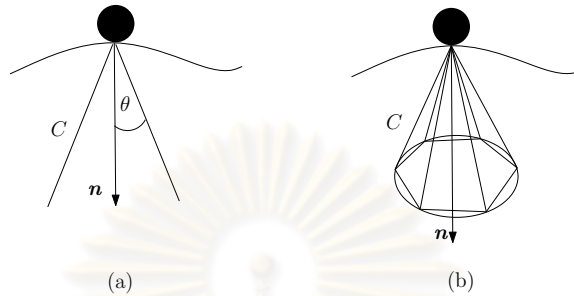


Figure 2.1: Coulomb friction: (a) is the friction cone for 2D grasps and (b) is the friction cone for 3D grasps and its approximating pyramid cone.

### 2.3 Grasp and Wrenches

Force closure is a property of a grasp which is defined by a set of contacts. Each contact can be defined by its position and inward normal direction. In this work, it is assumed that every contact of the same object is represented by the same contact model.

**Definition 2.1 (Grasp)** A grasp  $G$  is defined by a set of ordered pairs  $\{(\mathbf{p}_1, \mathbf{n}_1), \dots, (\mathbf{p}_n, \mathbf{n}_n)\}$  where  $\mathbf{p}_i$  and  $\mathbf{n}_i$  are the position vector and the inward normal vector of  $i^{\text{th}}$  contact.

A grasp achieves force closure when the grasp is able to counterbalance any external disturbance to the object being grasped. The external disturbance and the effect of contact points are represented as a force  $\mathbf{f}$  and a torque  $\boldsymbol{\tau}$ . In 2D, it is conventional to combine a force  $\mathbf{f} = (f_x, f_y)$  and a torque  $\boldsymbol{\tau}$  into an entity called a *wrench*  $\mathbf{w} = (f_x, f_y, \boldsymbol{\tau})$ . A wrench is a vector of force concatenated with a vector of torque. In 2D space, force can be described by a 2D vector while torque is described by a 1D vector, hence, a wrench in 2D space is a 3D vector. Likewise, a wrench in 3D space is 6D vector formed by a 3D force vector concatenated with a 3D torque vector. Formally, a wrench  $\mathbf{w}$  is denoted by  $(\mathbf{f}, \mathbf{t})$  where  $\mathbf{f}$  is a force vector and  $\mathbf{t}$  is a torque vector.

Combining force and torque into wrench makes it simpler to consider the force closure property. An effect of a contact point or external disturbance can be easily described

as a wrench. For example, let us consider an equilibrium in terms of wrenches. An object is said to be under equilibrium when the summation of all force and torque acting on the object is zero. Using wrench notation, an object achieves equilibrium when the summation of acting wrenches is the zero vectors.

Analysis on force closure concerns wrenches that can be exerted by a grasp. A contact is associated with a set of wrenches that it can exert. The set of wrenches that can be exerted by a contact and by a grasp are referred to as a *contact wrench set* and a *grasp wrench set*, respectively. In force closure analysis, a contact wrench is allowed to take arbitrarily large magnitude<sup>2</sup>. Since wrenches can be added up linearly, the set of wrenches exorable by the grasp is the positive combination of wrenches of its contacts. Let us refer to a positive combination of a set of vectors as a *linear positive span*, or positive span for short. Exorable wrenches of a grasp is a positive span of a contact wrench set of each contact.

**Definition 2.2 (Positive Span)** *Let  $W$  be a set of vectors. A positive span of  $W$ , denoted by  $\text{SPAN}^+(W)$ , is a set  $\{\alpha_i \mathbf{w}_i \mid \alpha_i \geq 0, \mathbf{w}_i \in W\}$ .*

### 2.3.1 Primitive Contact Wrenches

A contact wrench set can also be conveniently represented using positive span notation. A frictionless contact can only exert force in one direction and its contact wrench set is a ray in its respective wrench space. The ray can be represented as a positive span of a single wrench with arbitrary length lying in the same direction. For a frictional contact, a friction cone of which can be represented by positive span of its boundary force vectors. These vectors correspond to boundary wrenches and the whole contact wrench set can be represented by a positive span of these boundary wrenches, using one single arbitrary length for each direction.

We refer to unit length boundary wrenches as *primitive contact wrenches*. A contact wrench set is a positive span of primitive contact wrenches. Similarly, a grasp wrench set is a positive span of its contact wrench sets which is also equal to the positive span of all primitive contact wrenches (from all contact points). Let  $\mathbf{w}_1, \dots, \mathbf{w}_n$  be primitive contact wrenches of a grasp. The grasp wrench set of a grasp whose primitive contact wrenches are  $\mathbf{w}_1, \dots, \mathbf{w}_n$  can be represented as follows.

---

<sup>2</sup>In practice, a magnitude of a wrench is limited by the realization of the contact, e.g., the actuator of finger, the size of motor, etc. This detail is unrelated to the contact position and hence is neglected.

$$\{\sum_{i=1}^n \alpha_i \mathbf{w}_i \mid \alpha_i \geq 0\} \quad (2.1)$$

### 2.3.2 Grasp Wrench Set

Primitive contact wrenches and positive span represent a grasp wrench set in a compact form. It is necessary to understand the properties of a grasp wrench set when it is represented as a positive span of the primitive contact wrenches. A key feature of a positive span is its convexity. Convexity of a grasp wrench set is an important property exploited by most grasping works.

Other than convexity, a grasp wrench set also has other interesting properties. In the 3D frictional contact case, a friction cone is bounded by a quadratic surface, not a finite number of wrenches. A prominent difference is that a 3D friction cone, though it still maintains convexity, is no longer a linear structure. This implies that the corresponding grasp wrench set itself is nonlinear as well. In many works, a circular friction cone is simplified by an  $m$ -sided pyramid. Each boundary force vector of the pyramid yields one primitive contact wrench. Since  $m$  is finite, the number of primitive contact wrenches is also finite and thus the grasp wrench set can now be represented by linear surfaces allowing several tools in linear algebra to be applicable for analysis.

## 2.4 Force Closure

A grasp achieves force closure when its grasp wrench set covers the entire wrench space. A property called *positively spanning* is defined to describe that the positive span of a vector set covers the entire space.

**Definition 2.3 (Positively Span)** *We say that a set  $V$  of  $n$ -dimensional vector positively spans  $\mathbb{R}^n$  when  $\text{SPAN}^+(V) = \mathbb{R}^n$*

The force closure property can be formally defined using the notion of positively spanning, namely, a grasp achieves force closure when its associated wrenches, i.e., the polyhedral convex cone generated from the primitive contact wrenches, positively span their respective wrench space (3D wrench space in case of planar grasp and 6D wrench space in case of 3D grasp).

**Definition 2.4 (Force Closure)** *A grasp, whose primitive contact wrenches form the set  $W$  in  $\mathbb{R}^n$ , is said to achieve force closure when  $\text{SPAN}^+(W)$  positively span  $\mathbb{R}^n$ .*

Since the force closure property is defined over a set of vector (wrenches) associated with a grasp, it is more convenient to say that a set of vector achieves force closure, even though a set of vector cannot literally achieve force closure. Hereafter, saying that a set of wrenches achieves force closure is a short hand of saying that a grasp whose associated set of wrenches positively span  $\mathbb{R}^n$ .

## 2.5 Condition of Force Closure

The force closure property is defined using the notion of positively spanning. However, it is still indefinite to assert whether a set of vectors positively span a space. In this section we recite some of the well known conditions that assert on positively spanning of a set of vectors.

Mishra et al. related positively spanning of a set of vectors with a convex hull of the vectors. It is shown in (Mishra et al., 1987b) that a set of vectors  $W$  positively span a space when the origin of the space lies strictly inside the convex hull of  $W$ .

**Proposition 2.5** *A set of wrenches  $W$  in  $\mathbb{R}^n$  achieve force closure when the origin lies in the interior of the convex hull of  $\text{INT}(\text{CO}(W))$ .*

Proposition 2.5 transforms the force closure testing problem into a well defined computational geometry problem. A straightforward approach to solve the problem is to compute the convex hull of the primitive contact wrenches and directly whether the origin lies inside the interior. From this approach, it comes directly that if we can identify a half space through the origin that contains all primitive contact wrenches, the primitive contact wrenches cannot positively span the space.

**Proposition 2.6** *A set of wrenches  $W$  do not positively span  $\mathbb{R}^3$  if there exists a vector  $v$  such that the closed half space  $\mathcal{H}(v)$  contains every wrench in  $W$ .*

A closely related property of force closure is equilibrium. Equilibrium indicates that the net resultant wrench of the system is a zero vector. A grasp is said to achieve



equilibrium when it is possible for some contacts of the grasp to exert wrenches such that the net resultant wrench is zero vector. Formally, a grasp is an equilibrium grasp when Equation (2.2) has a non-trivial solution.

$$\sum_{i=1}^n \alpha_i \mathbf{w}_i = \mathbf{0} \quad (2.2)$$

Apparently, a grasp that achieves force closure also is an equilibrium grasp. However, the inverse is not necessary true. In the case of frictional contact, there exists a special class of equilibrium grasp called *non-marginal equilibrium*. A grasp achieves non-marginal equilibrium when the wrenches achieving equilibrium are not the wrenches associated with the boundary of a force cone. In practice, it means that any equilibrium grasp is also a force closure grasp under any arbitrarily greater frictional coefficient.

Nguyen (1988a) shows that a 2D 2-finger non-marginal equilibrium grasp is also a force closure grasp. Ponce and Faverjon (1995a) show the same implication in the case of 2D 3-finger grasp and also in the case of 3D 4-finger grasp (Ponce et al., 1997). Care should be taken not to take this implication into general. Though it might seem that non-marginal equilibrium implies force closure, this is not always true for any number of fingers. For example a 3D two finger non-marginal equilibrium grasp *does not* achieve force closure.

**Proposition 2.7** *A sufficient condition for 2- and 3-finger force closure in 2D and 4-finger force closure in 3D is non-marginal equilibrium*

## 2.6 Regrasping

Regrasping is a process of repositioning contact points of robot fingers. Two primitive forms of repositioning are *finger switching* and *finger sliding*. To determine an appropriate sequence of these two processes, we introduce a structure called a switching graph. A node in a switching graph represents a connected set of force closure grasps on three(four) particular polygonal edges(faces) in 2D(3D). An edge connecting two nodes indicates that there exist a grasp associated with one node that can be switched to a grasp associated with the other by finger switching. By using a switching graph, the regrasp problem can be formulated into a graph search problem. A path from the graph search

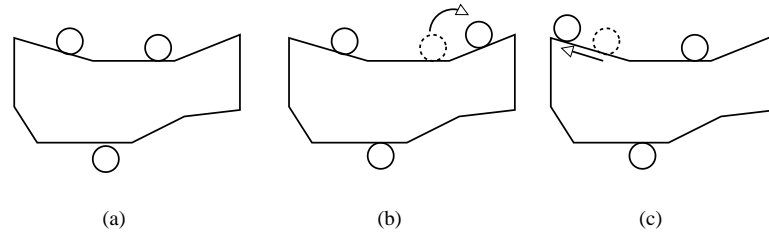


Figure 2.2: Regrasping overview: (a) Initial grasping configuration (b) A result of finger Switching. (c) a result of a finger sliding

determines a sequence of actions – switching and sliding to be executed in order to traverse from the initial to the final grasp. The following sections will describe the finger switching and sliding primitives and the switching graph in detail.

### 2.6.1 Finger Switching and Finger Sliding

Regrasping process which changes grasping configuration by placing an additional finger on desired contact point and then releasing one finger of the initial grasp is called finger switching. For example, let us assume that a starting grasp holds a polygonal object on points  $p_a$ ,  $p_b$  and  $p_c$  and we want to switch to a grasp holding points  $p_b$ ,  $p_c$  and  $p_d$ . A finger switching process starts by placing an additional finger on  $p_d$  and then releasing the finger at  $p_a$ . If both grasps satisfy the force closure property, the entire process still holds the force closure property. For the case of 4(5)-finger hand grasping a polygonal(polyhedral) object, finger switching requires that two(three) grasping configurations must have two contact points in common and both of them achieve force closure.

Finger sliding is a process for repositioning fingers by sliding them along edges(faces) of a polygon(polyhedron) while maintaining a force closure grasp during the sliding process. Using this process, we can change grasping configuration with in the same set of force closure grasps. This means the relation between finger sliding and a node of switching graph. However, finger sliding may be hard to implement mechanically since it is required that fingers must always touch the edge during sliding. Finger switching can imitate finger sliding by switching fingers from the initial to the final position of the sliding. Examples of finger switching and sliding are shown in Figure 2.2.

## CHAPTER III

# REGRASP PLANNING FOR A POLYGONAL OBJECT

### 3.1 Introduction

A framework to deal with the problem of regrasp planning for a polygonal object will be discussed in this chapter. Since we separate the regrasp planning from task and mechanical constraints, general sets of force-closure grasps for an object can be computed beforehand. We propose an efficient structure called *Switching Graph* to store sets of force-closure grasps which will be further used to solve the regrasp planning problem. Sets of force-closure grasps are computed by considering combinatorial sets of polygonal edges. Each set is assigned to a vertex of the switching graph. Our planner exploits the connectedness of a grasp set to compute a regrasp sequence between two grasps in the same set. This type of sequence can be performed by continuous movements of end effectors while all grasps in this sequence are guaranteed to satisfy the force-closure condition. An edge joining two vertices indicates that a grasp in one vertex can change to a grasp in the other vertex using a finger switching. The connectivity of this structure captures ability to switch from one grasp set to another grasp set and allows regrasp planning to be formulated as a graph search. Since the structure contains sets of grasps, the regrasp planner is permitted to extract the information of the graph and compute a set of regrasp sequences that ensures force-closure for every grasping configuration in the sequences without re-verifying all grasps in the sequences. An important advantage of our framework is generality of solutions which does not specifically depend on a task or a robot hand. By applying the switching graph, we can consider the regrasp planner as a middle-level planner which acquires an initial and a goal grasp from a task planner then computes a set of regrasp sequences and then transfers the solution set to mechanical-controlled level. Another advantage is globalization of the switching graph. Since the switching graph contains sets force-closure grasps considering all combinations of polygonal edges of an object therefore it allows a planner to globally search for regrasp sequences.

The organization of this chapter is as follows. Force-closure conditions in 2D are presented in Section 3.2. The switching graph for a polygon is discussed in Section 3.3. The description of grasp representations are appeared in Section 3.3.1. In this section, we will describe simplification of a force-closure grasp set into a linear structure which

is easy to extract its information based on the assumption of a polygonal object. The relation between regrasp operations and sets of grasps is presented in Section 3.3.2 and 3.3.3. Construction of the switching graph containing the sets of grasps is described in Section 3.3.4. We provide a guideline of using the switching graph in Section 3.4. The implementation of our approach and experimental results are shown in Section 3.5.

### 3.2 Force-closure conditions in 2D

In Chapter 2, we have described that 2- and 3-finger non-marginal equilibrium is sufficient to satisfy force-closure in 2D. Due to (Nguyen, 1988b), the following proposition characterizes 2-finger equilibrium.

**Proposition 3.1** *A necessary and sufficient condition for two points to form an equilibrium grasp with non-zero contact forces is that the line joining both points lies completely in the two double-sided friction cones at the points.*

The following two propositions completely characterize 3-finger grasps achieving equilibrium with non-zero contact forces.

**Proposition 3.2** *A necessary and sufficient condition for three points to form an equilibrium grasp with three non-zero contact forces, not all of them being parallel, is that (Pa) there exist three lines in the corresponding double-sided friction cones that intersect in a single point and (Pb) the vectors parallel to these lines and lying in the internal friction cones at the contact points positively span<sup>1</sup>  $\mathbb{R}^2$ .*

For a polygonal object, given three edges, the set of equilibrium grasps satisfying the conditions of Proposition 3.2 is described by non-linear relation of three contact positions and directions of forces in three friction cones. Instead of using Proposition 3.2, the construction of the switching graph relies on a stricter condition given below in Proposition 3.4. The following definition is needed to write the proposition.

**Definition 3.3** *Let  $C_i (i = 1, 2, 3)$  be the cones centered on  $\omega_i$  with half angle  $\theta$ . We say that the three vectors  $\omega_i (i = 1, 2, 3)$   $\theta$ -positively span  $\mathbb{R}^2$  when any triple of vectors  $v_i \in C_i (i = 1, 2, 3)$  positively span  $\mathbb{R}^2$ .*

<sup>1</sup>A set of vectors positively spans  $\mathbb{R}^n$  if any vector in  $\mathbb{R}^n$  can be written as a positive linear combination of the set.

It is easy to see that when three vectors  $\theta$ -positively span the plane, the three vectors positively span the plane and every pairwise angle is smaller than  $\pi - 2\theta$ . In the following proposition and the remainder of the paper, we will denote by  $\theta$  the half angle of every friction cone.

**Proposition 3.4** *A sufficient condition for three points to form an equilibrium grasp with non-zero contact forces is that: (Pa) there exist three lines in the corresponding double-sided friction cones that intersect in a single point and (Pc) the internal normals at the three contact points  $\theta$ -positively span  $\mathbb{R}^2$ .*

A proof of the above proposition can be found in (Ponce and Faverjon, 1995a). Note that replacing condition Pb in Proposition 3.2 with condition Pc yields a stricter condition; certain grasps satisfying Proposition 3.2 will not satisfy Proposition 3.4. Some of them, however, form 2-finger force-closure grasps or parallel grasps satisfying Proposition 3.5. The underlying reason for the use of a more restrictive condition will become clear as we explain the representation of concurrent grasps in Section 5.3.1.

**Proposition 3.5** *A necessary and sufficient condition for three points to form an equilibrium grasp with three parallel and non-zero contact forces is that there exist three parallel lines in the corresponding double-sided friction cones and for three vectors parallel to these lines and lying in the internal friction cones at the contact points, the vector parallel to the middle line are in the opposite direction from the other two.*

*Proof:* Obviously, three parallel non-zero contact forces achieve a force equilibrium only when exactly one of them lies in the opposite direction of the other two. If the opposing force does not lie between the other two, the moment with respect to any points along the other vectors will not be zero. To achieve force closure, that force must be in the middle. In that case, it is obvious that a moment equilibrium can also be achieved. ■

This type of force-closure grasp is taken into account in order to cover some grasps missing by applying Proposition 3.4. A formulation of the conditions in Proposition 3.4 into linear constraints will be described later in Section 3.3.1.3.

### 3.3 Switching Graph for a Polygonal Object

This work assumes finger switching performed by changing one contact at a time. At least one free finger is needed when switching from one force-closure grasp to another. In 2D workspace, 2-finger and 3-finger force-closure grasps are sufficient for grasp stability. Therefore, a robot hand used in this work is assumed to be equipped with four fingers. For 3-finger force-closure grasps, our approach consider (1) *parallel grasps*: force-closure grasps satisfying Proposition 3.5, and (2) *concurrent grasps*: force-closure grasps satisfying Proposition 3.4<sup>2</sup>.

However, a parallel grasp and a concurrent grasp cannot switch to each other directly. For a parallel grasp satisfying Proposition 3.5, the three double-sided friction cones of the three grasped edges, when being drawn at the same point, must intersect in a nonempty region (i.e., so that three parallel lines in the cones exist). This prevents any finger switching for a parallel grasp to result in a concurrent grasp because there is still a pair of edges whose internal normals forbid the three internal normals from  $\theta$ -positively spanning the plane no matter which edge is chosen to participate in the finger switching. It is, however, possible for a finger switching to change into a 2-finger force-closure grasp. This information allows us to draw the diagram in Fig. 3.1 showing the overall structure of a switching graph characterizing types of grasps a finger switching can transform a certain type of grasps into.

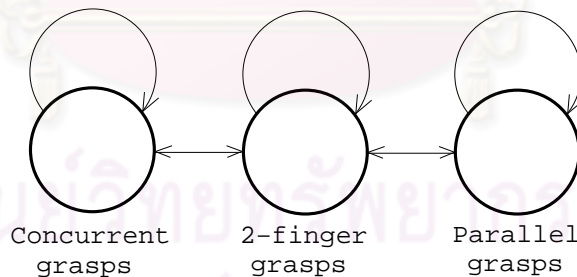


Figure 3.1: Switching diagram

#### 3.3.1 Representing Force-closure Grasps

Generally, a set of force-closure grasps can be described in the configuration space of contact points. In 2D, a contact point on the object's surface can be identified by

<sup>2</sup>by not using Proposition 3.2, some grasps may be missing as mentioned in Section 3.2 but this will allow simple characterization of independent contact regions which is an important foundation of the switching graph

one parameter. Hence, a 2- and 3-finger grasp is represented by a point in 2D and 3D parameter spaces, respectively. However, it is not straight forward to compute and store a grasp set in a data structure. In this section, we will describe representations of force-closure grasp sets for each type. We transform the representation of grasps satisfying Proposition 3.1 and 3.4 from parameter space into workspace. Since our representations of 2-finger and concurrent grasps are in the same  $\mathbb{R}^2$  space, therefore we can efficiently compute a set of grasps by using computational geometry algorithms in 2D. In contrast, since the lines of parallel forces intersect at infinity, a parallel grasp is represented by a point in 3D parameter space but planning regrasp sequences can be reduced into a problem in 2D.

### 3.3.1.1 Representing Concurrent Grasps

As mentioned earlier, a grasp is geometrically defined by the positions of the fingers on the object's boundary. Assuming that an object is a polygon, a contact point on a polygonal edge can be defined by distance from an endpoint of the edge. This amounts to using three parameters to uniquely define a 3-finger grasp (with the three grasped edges already chosen). However, using Proposition 3.4, we can define a set of concurrent grasps with only two parameters. In the following, we explain how this can be done.

Let us consider Fig. 3.2(a) where  $E_i, i = a, b, c$  ( $a \neq b \neq c$ ) are the three shown edges whose internal normals  $\theta$ -positively span the plane. Consider also a point  $\mathbf{x}_0$  such that each of the three inverted friction cones<sup>3</sup> at  $\mathbf{x}_0$  intersects the corresponding edge in a non-empty segment. Let us denote the intersection segment on edge  $E_i$  by  $E'_i$  and consider a grasp defined by  $\mathbf{x}_i \in E'_i, i = a, b, c$  (Fig. 3.2(b)). Obviously from the construction, the three double-sided friction cones at  $\mathbf{x}_i, i = a, b, c$  intersect in a region containing  $\mathbf{x}_0$  (regardless of where  $\mathbf{x}_i$  is chosen in  $E'_i$ ) and in turn, according to Proposition 3.4, the three contact points  $\mathbf{x}_i, i = a, b, c$  form a concurrent grasp (Fig. 3.2(b)). Therefore,  $\mathbf{x}_0$  can be used for defining a set of concurrent grasps formed by all possible triples  $\mathbf{x}_i \in E'_i, i = a, b, c$ . Equivalently, we obtain the following proposition (a 3D version of this proposition can be found in (Sudsang and Ponce, 1995)).

---

<sup>3</sup>an inverted friction cone w.r.t an edge is a friction cone projecting toward the edge with its axis parallel to the normal of the edge

**Proposition 3.6** *A sufficient condition for three fingers to form a concurrent grasp is that the internal normals of the three grasped edges  $\theta$ -positively span the plane and there exists a point  $x_0$  such that the inverted friction cones at this point intersect the three grasped edges.*

Note that each point  $x_0$  satisfying Proposition 3.6 yields three *independent contact regions* where fingers can be placed independently while achieving concurrent grasp: these regions are simply the intersection of the inverted cones in  $x_0$  with the contact edges (Fig. 3.2(b)).

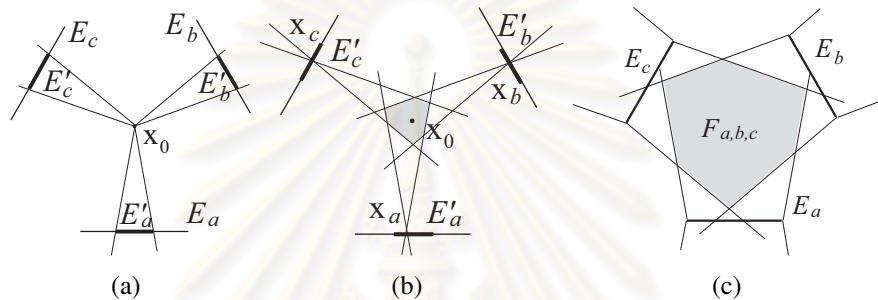


Figure 3.2: Construction of a focus cell: (a) inverted friction cones, (b) independent contact regions, (c) focus cell from the intersection of the union of cones

We are now ready to discuss how a vertex in the switching graph represents a set of grasps. A vertex of the switching graph represents a set of concurrent grasps by having an association with a set of all points  $x_0$  satisfying Proposition 3.6 for a given triple of edges. Since an inverted friction cone at  $x_0$  intersect the corresponding edge when  $x_0$  lies in the polygon defined by the union of all double-sided friction cones at every point on the edge (Fig. 3.2(c)), the set of all  $x_0$  satisfying Proposition 3.6 can be obtained from the intersection of the three polygons each of which is the union of all double-sided friction cones on each edge. In the following definition, we give a name for the intersection polygon for future references.

**Definition 3.7** *The polygon defining the set of all points  $x_0$  satisfying Proposition 3.6 for a given set of three edges  $E_i$ ,  $E_j$  and  $E_k$  where  $i \neq j \neq k$  will be called the focus cell for the edges and will be denoted by  $F_{i,j,k}$*

With the above definition, we can say that a vertex in the switching graph represents



a set of concurrent grasps on edge  $E_i$ ,  $E_j$  and  $E_k$  by having an association with  $F_{i,j,k}$ , the focus cell for the triple of edges.

### 3.3.1.2 Representing 2-finger Grasps

A point in the plane can be also used to represent a set of 2-finger grasps. This representation is applied for a simplicity of checking a finger switching between a 2-finger grasp and a concurrent grasp as described in Section 3.3.2.2. To understand the process, consider Fig. 3.3(a) showing a grasp at  $x_a$  on  $E_a$  and  $x_b$  on  $E_b$ . To achieve force-closure, according to Proposition 3.1, the line segment  $L$  joining  $x_a$  and  $x_b$  must lie within the friction cones at the contact points ( $C_a$  and  $C_b$ ). An equivalent condition is that the arrangement of the segment  $L$  must be within the double-sided cone  $C_{a,b}^\cap$  where  $C_{a,b}^\cap$  is obtained from the intersection of double-sided friction cones  $C_a$  and  $C_b$  drawn at the same point (Fig. 3.3(b)). In other words, the double-sided friction cones intersects when the angle between two associated normals of the contact edges is in  $(\pi - 2\theta, \pi + 2\theta)$ . Following (Nguyen, 1988b), this allows independent contact regions to be found as the intersection between the double-sided cone  $C_{a,b}^\cap$  at a point  $x_0$  and the two grasped edges (Fig. 3.3(c)). The corresponding focus cell is, in turn the set all points  $x_0$  with non-empty independent contact regions. Like the concurrent case, the focus cell can be constructed from the intersection of the two polygon each of which is the union of the cone  $C_{a,b}^\cap$  at all points on each edge (Fig. 3.3(d)).

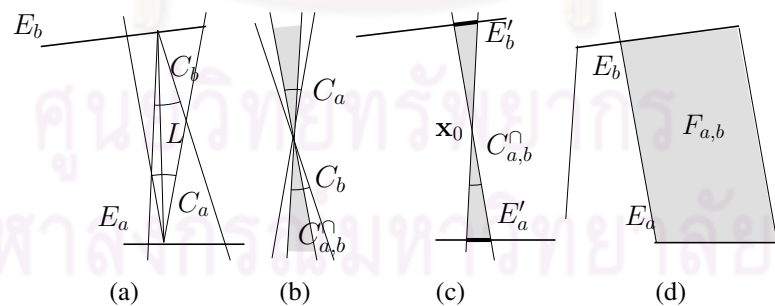


Figure 3.3: 2-finger force-closure focus cell construction. (see text)

### 3.3.1.3 Representing Parallel Grasps

Parallel grasp is another type of 3-finger grasps. It provides additional force-closure grasps that cannot satisfy the conditions of Proposition 3.4. Since the lines of forces

forming a parallel grasp do not intersect at a point, a set of parallel grasps cannot be represented by any elements in the plane. We use three parameters to indicate positions on three grasped edges instead.

However, we do not apply the condition in Proposition 3.5 to construct a set of parallel grasps directly because the conditions are formulated into non-linear constraints. We have presented a new condition for three contact points to form a parallel grasp. Let us consider Proposition 3.5. There exists three parallel forces from three contact points  $\mathbf{x}_a$ ,  $\mathbf{x}_b$  and  $\mathbf{x}_c$  whose normals respectively are  $\mathbf{n}_a$ ,  $\mathbf{n}_b$  and  $\mathbf{n}_c$  (Fig. 3.4(a)) when there exists  $i, j, k \in \{a, b, c\}$  and  $i \neq j \neq k$  such that the intersection of cones  $C_i, C_j$  and  $-C_k$ , all of them originated at the same point, is not empty (Fig. 3.4(b)). This condition is equivalent to the condition that the angle between  $\mathbf{n}_i, \mathbf{n}_j$  and  $-\mathbf{n}_k$  are pairwise less than  $2\theta$ . If we limit  $\theta$  to be less than  $\pi/4$  (i.e., friction coefficient  $< 1$ ), only one triple of  $(i, j, k)$  will satisfy the previous condition. We call the contact point that has opposite direction force as a *center point*. We define a structure called a *common cone* that aids in existence checking of a parallel grasp as follows. A common cone exists only when three contact points  $\mathbf{x}_a$ ,  $\mathbf{x}_b$  and  $\mathbf{x}_c$  have three parallel forces, one of them lies in the opposite direction from the other two. From Fig. 3.4(a),  $\mathbf{x}_a$  is the center point, a common cone  $C_{a,b,c}^\cap$  is the double-sided cone of the intersection of three cones  $-C_a, C_b$  and  $C_c$  (Fig. 3.4(c)). If we draw a common cone on the center point, part of the plane that is not occupied by the cone will be divided into two regions. These regions are called the *outer regions*. The next proposition from (Phoka et al., 2005) uses the notion of outer regions to define a necessary and sufficient condition for an existence of a parallel grasp when  $\theta$  is less than  $\pi/4$ .

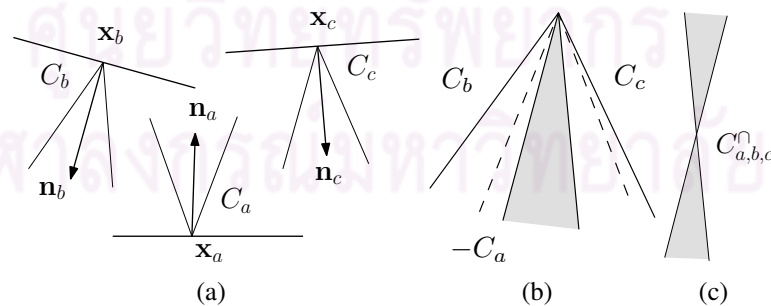


Figure 3.4: Construction of a common cone: (a) A parallel grasp. (b) Three friction cones of (a) drawn at the same point. The dashed cone is inverted. (c) A common cone.

**Proposition 3.8** A necessary and sufficient condition for three contact points  $\mathbf{x}_a$ ,  $\mathbf{x}_b$  and

$\mathbf{x}_c$ , whose normals respectively are  $\mathbf{n}_a, \mathbf{n}_b$  and  $\mathbf{n}_c$ , to form a parallel grasp is that two following conditions hold. (Pd) a common cone  $C_{\mathbf{n}_a, \mathbf{n}_b, \mathbf{n}_c}^\cap$  is not empty. (Pe) Let us assume that the center point of these three points be  $\mathbf{x}_a$ . The points  $\mathbf{x}_b$  and  $\mathbf{x}_c$  do not lie in the same outer region separated by the common cone  $C_{a,b,c}^\cap$  originated at  $\mathbf{x}_a$  (Fig. 3.5).

*Proof:* For the sufficient side, let us draw a segment connecting  $p_b$  and  $p_c$ . If both of them do not lie in the same side of the common cone, the segment  $\overline{p_b p_c}$  will definitely intersect the common cone. Let  $p_x$  be any point in the intersection of  $\overline{p_b p_c}$  and the common cone, we can draw a line from  $p_a$  to  $p_x$ . That line definitely lies in the friction cone of  $p_a$  (see Fig. 3.5). A line parallel to  $\overline{p_a p_x}$  that passes  $p_b$  also lies in the friction cone of  $p_b$ , and so is the case of  $p_c$ . From a construction of a common cone, we can find three forces parallel along these lines that form a parallel grasp.

For the necessary side, if there exists a parallel grasp, a common cone will also exist. Now, if  $p_b$  and  $p_c$  lie in the same side,  $\overline{p_b p_c}$  does not intersect the common cone. A line lying in the middle of  $p_b$  and  $p_c$ , which is necessary for a parallel grasp, must intersect with the segment  $\overline{p_b p_c}$ . However, since  $p_b$  and  $p_c$  lie completely in one outer region, every point in  $\overline{p_b p_c}$  also lies in that outer region. It follows that if we pick some points on  $\overline{p_b p_c}$  and use it to define a middle line, the other two lines passing through  $p_b$  and  $p_c$  that are parallel to the first line will also lie outside their respective common cone. Thus, at least one of them must lie outside its friction cone. This completes the proof as a contrapositive. ■

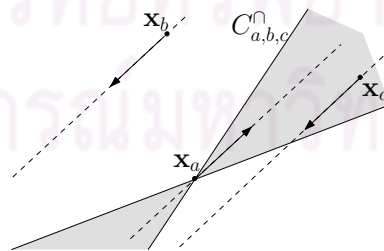


Figure 3.5: 3 contact points forming a parallel grasp: When  $\overline{p_b p_c}$  intersects with the common cone, we can find three parallel lines and vectors that satisfy 3.8

Since we are dealing with a polygonal object, we need a representation of a contact point on a polygonal edge. Let  $E_a$  be an edge with an end point  $\mathbf{a}_0$  and a unit direction  $\mathbf{t}_a$ . The length of  $E_a$  is  $l_a$ . A point  $\mathbf{x}_a$  on an edge  $E_a$  can be represented by  $\mathbf{x}_a = \mathbf{a}_0 + u_a \mathbf{t}_a$

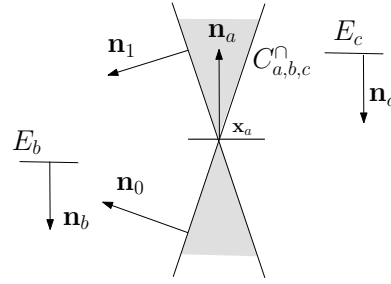


Figure 3.6: Representing a common cone: A common cone on point  $\mathbf{x}_a$  and vectors  $\mathbf{n}_0$  and  $\mathbf{n}_1$ .

where  $u_a \in [0, l_a]$ . By using this representation, we can represent a set of all grasping configurations  $G_{a,b,c}$  by a polytope in 3D, each dimension represents a value of  $u_a$ ,  $u_b$  and  $u_c$ , respectively.

The polytope  $P$  is defined by a set of linear constraints. For a polytope of  $E_a$ ,  $E_b$  and  $E_c$ , contact points  $\mathbf{x}_a$ ,  $\mathbf{x}_b$ ,  $\mathbf{x}_c$  are constrained to be on the polygonal edges. We define length constraints

$$0 \leq u_i \leq l_i \text{ for } i = a, b, c \quad (3.1)$$

Next, a set of constraints that bounds contact points to satisfy Proposition 3.8 is presented. Let us assume that the center edge is  $E_a$  and the others are  $E_b$  and  $E_c$ . Intuitively, if one point  $\mathbf{x}_b$  on  $E_b$  lies in an outer region (separated by a common cone of some point  $\mathbf{x}_a$  on  $E_a$ ), the second point  $\mathbf{x}_c$  on  $E_c$  must be in a common cone of the third point or in the other outer region. However, the feasible area may not be convex so we construct it from a union of six convex polytopes. We denote  $L(\mathbf{n}, \mathbf{x}) \equiv \mathbf{n} \cdot \overrightarrow{\mathbf{x}_a \mathbf{x}} \geq 0$  and  $R(\mathbf{n}, \mathbf{x}) \equiv \mathbf{n} \cdot \overrightarrow{\mathbf{x}_a \mathbf{x}} \leq 0$  to describe the constraints of a point  $\mathbf{x}$  on the left and the right side of the half space described by the normal vector  $\mathbf{n}$  and a line passing through  $\mathbf{x}_a$ . We define constraints for each of them as follows.

$$K_0 \equiv L(\mathbf{n}_0, \mathbf{x}_b) \cap L(\mathbf{n}_1, \mathbf{x}_b) \cap R(\mathbf{n}_0, \mathbf{x}_c) \cap R(\mathbf{n}_1, \mathbf{x}_c) \quad (3.2)$$

$$K_1 \equiv R(\mathbf{n}_0, \mathbf{x}_b) \cap R(\mathbf{n}_1, \mathbf{x}_b) \cap L(\mathbf{n}_0, \mathbf{x}_c) \cap L(\mathbf{n}_1, \mathbf{x}_c) \quad (3.3)$$

$$K_2 \equiv L(\mathbf{n}_0, \mathbf{x}_b) \cap R(\mathbf{n}_1, \mathbf{x}_b) \quad (3.4)$$

$$K_3 \equiv L(\mathbf{n}_0, \mathbf{x}_c) \cap R(\mathbf{n}_1, \mathbf{x}_c) \quad (3.5)$$

$$K_4 \equiv R(\mathbf{n}_0, \mathbf{x}_b) \cap L(\mathbf{n}_1, \mathbf{x}_b) \quad (3.6)$$

$$K_5 \equiv R(\mathbf{n}_0, \mathbf{x}_c) \cap L(\mathbf{n}_1, \mathbf{x}_c) \quad (3.7)$$

Where  $\mathbf{n}_0$  and  $\mathbf{n}_1$  are the normal vector of left margin and right margin of the common cone respectively (see Fig. 3.6). The first two constraints,  $K_0$  and  $K_1$ , are cases that  $u_b$  and  $u_c$  are on two distinct outer regions separated by a common cone at  $u_a$  while the others are for the cases when  $u_b$  or  $u_c$  are in the common cone. Each sub-polytope  $P'_i$  are defined as a convex hull constrained by Eqs. (3.1) and  $K_i$ . These polytopes represent a connected set of all parallel grasps on the edges  $E_a$ ,  $E_b$  and  $E_c$  and involve with a vertex in the switching graph.

### 3.3.2 Finger Switching

Regrasp process which changes grasping configuration by placing an additional finger on desired contact point and then releasing one finger of the initial grasp is called finger switching. Intuitively, considering grasps on two different edge sets, a finger switching can be performed when contact points on the common grasped edges are restrained. In parameter spaces, the common contact points are computed in subspaces of the common edges. It requires projections of two grasp sets onto the subspaces. The projections are then checked for the intersection which indicates a set of common contact points. This method is applied for parallel grasps as described in Section 3.3.2.3. On contrary, our algorithm computes finger switching of 2-finger and concurrent grasps in 2D workspace.

This operation involves with an edge in the switching graph. Considering two grasp sets associated with two vertices, existence of finger switching between these sets indicates an edge linking the related vertices. Switchings among concurrent grasps or between concurrent grasps and 2-finger grasps can be described by a set of common points in the plane representing grasps on distinct grasped edges. Finger switching of parallel grasp is computed in parameter subspace of two common edges. Representation of a 2-finger grasp set is transformed into parameter space when finger switching into a parallel grasp is needed.

### 3.3.2.1 Finger Switching among Concurrent Grasps

For the sake of representing a set of concurrent grasps by a focus cell, finger switching is related to the intersection of two focus cells that their associated grasps have two common grasped edges. Let us consider two focus cells  $F_{a,b,c}$  and  $F_{a,b,d}$  such that  $F_{a,b,c} \cap F_{a,b,d} \neq \emptyset$  (Fig. 3.7) where  $q$  be a point in  $F_{a,b,c} \cap F_{a,b,d}$ . Clearly,  $q$  defines two sets of concurrent grasps: one for triple of edges  $E_a, E_b, E_c$  and the other for triple of edges  $E_a, E_b, E_d$ . Let us suppose that the fingers 1,2 and 3 are respectively on edges  $E_a, E_b$  and  $E_c$  and forming one of the concurrent grasps defined by  $q$ . It is easy to see that the hand can switch to another concurrent grasp on edges  $E_a, E_b$  and  $E_d$  by placing finger 4 on any point in the intersection between edge  $E_d$  and its inverted friction cone at  $q$  (Fig. 3.7(c)). Once finger 4 is on  $E_d$ , finger 3 can leave edge  $E_c$  resulting in a switching from a concurrent grasp on  $E_a, E_b, E_c$  by fingers 1,2,3 to another concurrent grasp on  $E_a, E_b, E_d$  by fingers 1,2,4. This finger repositioning sequence enables us to plan finger switching by identifying intersection between two focus cells for which their triples of grasped edges are different from each other by only one edge.

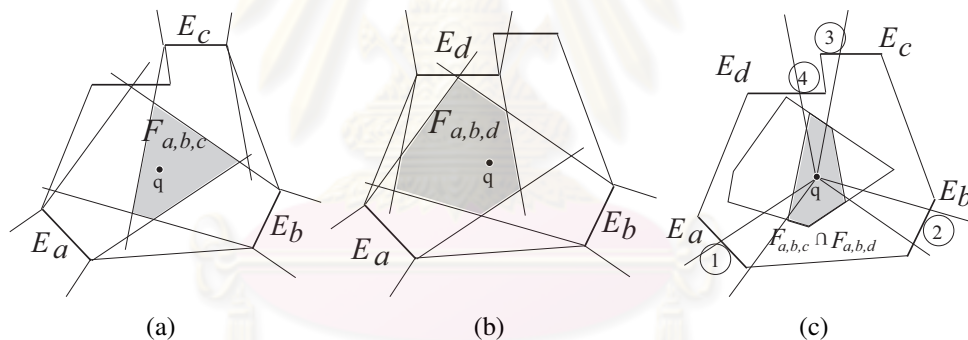


Figure 3.7: Finger switching between concurrent grasps: (a)  $F_{a,b,c}$ , (b)  $F_{a,b,d}$ , (c) their intersection

### 3.3.2.2 Finger Switching between 2-finger Grasps and Concurrent Grasps

According to the assumption of a 4-finger hand used in this work, any couple of 2-finger grasps can always switch to each other. Therefore, an edge linking any two vertices of two 2-fingers grasps always exists.

Let us consider 2-finger grasps on  $E_a, E_c$  and concurrent grasps on  $E_b, E_c, E_d$ . Since a set of 2-finger grasps and a set of concurrent grasps are represented by points in the plane, finger switching between these grasp types can be described by the intersec-

tion between two focus cells  $F_{a,c}$  and  $F_{b,c,d}$  where  $E_c$  is the common edge. Clearly, a point  $\mathbf{q}$  lying in  $F_{a,c} \cap F_{b,c,d}$  represents two sets of grasps. The independent contact regions of  $E_b, E_c, E_d$  are formed by the projections of the inverted cones at  $\mathbf{q}$  onto them, namely, the intersections are denoted by  $E'_b, E'_c, E'_d$ . The independent contact regions of 2-finger grasps on  $E_a, E_c$  are determined by the intersections between  $E_a, E_c$  and the cone  $C_{a,c}^\cap$  emanated from  $\mathbf{q}$ . Let these independent contact regions denoted by  $E''_a$  and  $E''_c$ . From the construction of  $C_{a,c}^\cap$ , it is clear that  $C_{a,c}^\cap$  is a subset of  $C_c$  when their origin is at the same point. Therefore,  $E''_c$  is also a subset of  $E'_c$ . To perform a finger switching, three fingers forming a concurrent grasp have to be placed on  $E'_b, E''_c, E'_d$  and the other finger has to be positioned on  $E''_a$  to form a 2-finger grasp with the finger on  $E''_c$ .

### 3.3.2.3 Finger Switching among Parallel Grasps

Parallel grasps can switch among them or to 2-finger grasps. In the former case, finger switching requires that two non-switching contact points must remain the same during the process. Formally, there will be an edge connecting a vertex  $v_{a,b,c}$  and a vertex  $v_{b,c,d}$  when there exists a triple of points  $(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c) \in G_{a,b,c}$  and a triple  $(\mathbf{x}'_b, \mathbf{x}'_c, \mathbf{x}'_d) \in G_{b,c,d}$  such that  $\mathbf{x}_b = \mathbf{x}'_b$  and  $\mathbf{x}'_c = \mathbf{x}_c$ .

To check whether there exist grasps from two grasp sets that can switch to each other, we consider two polytopes representing these grasp sets. Let  $P_1$  be the polytope for edges  $E_a, E_b, E_c$  and  $P_2$  be the polytope for edges  $E_b, E_c, E_d$ . The space of  $P_1$  and  $P_2$  have two components (axes) in common, namely the axes of  $u_b$  and  $u_c$ . These components correspond to the non-switching edges, i.e., the common edges of both grasps. The projection of  $P_1$  on the space of these two components represents the set of points on edges  $E_b$  and  $E_c$  that a parallel grasp on  $E_a, E_b$  and  $E_c$  is possible. Similarly, the projection of  $P_2$  represents the subspace of parallel grasps on  $E_b, E_c$  and  $E_d$ . If the intersection between these two projections is not empty, then there exists points on  $E_b$  and  $E_c$  that form a parallel grasp on both  $E_a, E_b, E_c$  and  $E_b, E_c, E_d$ . The reverse is also definitely true. Fig. 3.8 depicts the projection process.

The process is completed by picking six sub-polytopes  $P'_0 \dots P'_5$  associated with a parallel grasp vertex. We find their projections on a non-switching plane by examining their extreme points. For each sub-polytope, we project every extreme point of it on the non-switching plane and construct a convex hull from these points. The union of all

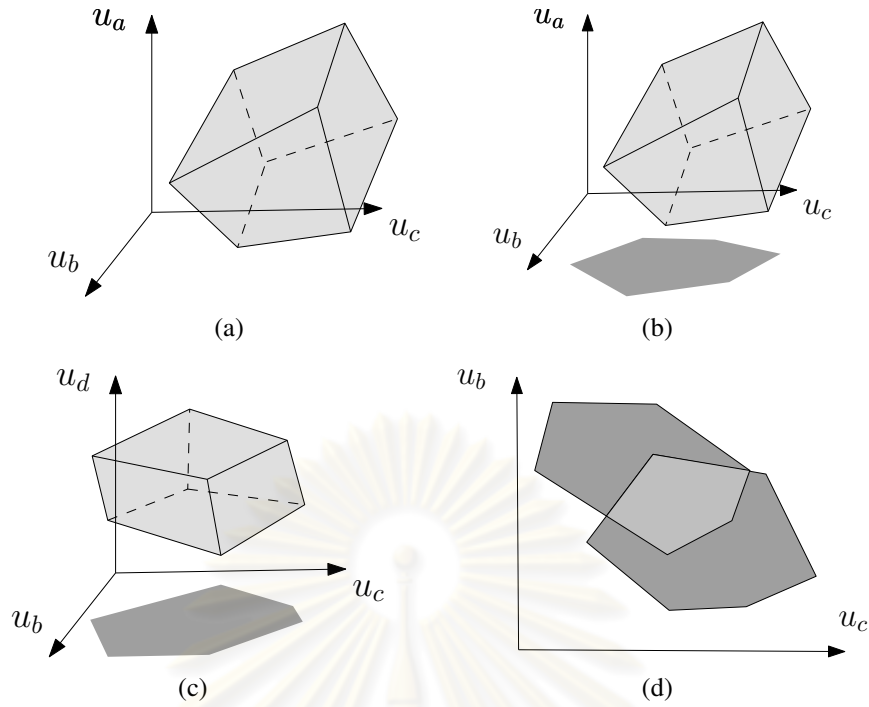


Figure 3.8: Finger switching between parallel grasps: (a) A polytope representing possible contact points (in term of  $u_a$ ,  $u_b$  and  $u_c$  (b), (c) two polytopes and their projections. (d) Intersection of the projected polygon representing a set of common points for a finger switching.

projected convex hulls is a projection of the entire polytopes.

### 3.3.2.4 Finger Switching between Parallel Grasps and 2-finger Grasps

Planning finger switching between parallel grasps and 2-finger grasps requires more operations because the representations of two grasp types are different. A set of 2-finger grasps has to be transformed into positions on the grasped edges. Let a parallel grasp is on edges  $E_a, E_b, E_c$  and a two finger grasp is on edges  $E_a, E_d$  where  $E_a$  is the common edge. To transform the representation of 2-finger grasps, we compute the projections of cones  $C_{a,d}^\cap$  emanated from all points in the focus cell  $F_{a,d}$  to  $E_a, E_d$ . the double-sided cone  $C_{a,d}^\cap$  is drawn at every point in the focus cell  $F_{a,d}$  to intersect with the grasped edges. The intersections are regions that are feasible for 2-finger grasps (Fig. 3.9(a)) and denoted by  $E_a'', E_d''$ . For the parallel grasp, the polytopes of the parallel grasps are projected onto  $u_a$  axis (Fig. 3.9(b)). The union of projections  $I_a$  of the polytopes is transformed into a region on  $E_a$ . We denote this region by  $E_a'$  which is a feasible region that can form parallel grasps with some points on  $E_b, E_c$ . Clearly, a finger switching can be performed



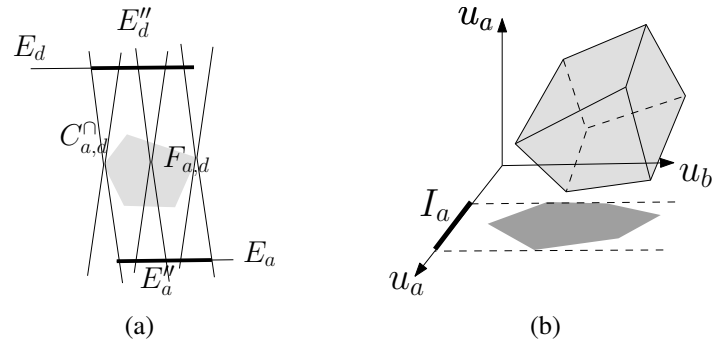


Figure 3.9: Finger switching between 2-finger grasps and parallel grasps: (a) A focus cell is transformed into graspable regions on the grasped edges. (b) A polytope is projected in the axis of the common edge.

when the finger on  $E_a$  is placed in  $E'_a \cap E''_a$  and forms a 2-finger grasp with a point on  $E_d$  and a parallel grasp with two points each of which is on  $E_b, E_c$  concurrently. Non-empty intersection region  $E'_a \cap E''_a$  indicates finger switching. As a result, there exists an edge joining the vertices associated with these two grasp sets. The number of common fingers for switching between 2-finger grasp and parallel grasp is up to two fingers. If two edges are in common, one non-empty intersection region on a common grasped edge is sufficient to perform finger switching.

### 3.3.3 Finger Aligning

Finger aligning is a process for repositioning fingers by rolling or sliding them along edges of a polygon while maintaining a force-closure grasp during the repositioning process. By applying this operation, we can change grasping configuration within the same connected set of grasps. This expresses the direct relation between finger aligning and a vertex of switching graph explained in section 3.3.1.

Finger aligning is necessary as exemplified in the following instance. Let us consider Fig. 3.10(a). Obviously, because  $F_{a,b,c} \cap F_{b,d,e} = \emptyset$ , it is not possible to switch directly from a grasp on edges  $E_a, E_b, E_c$  to another grasp on edges  $E_b, E_d, E_e$  using finger switching. However, suppose the current grasp on  $E_a, E_b, E_c$  is defined by  $\mathbf{q}_1$ , a finger switching can be performed to switch to another grasp on edge  $E_a, E_b, E_d$  (i.e.,  $\mathbf{q}_1$  is in both  $F_{a,b,c}$  and  $F_{a,b,d}$ ) and somehow if the hand can adjust the finger to change from the grasp defined by  $\mathbf{q}_1$  to a grasp defined by  $\mathbf{q}_2$  (which could be any point in  $F_{a,b,d} \cap F_{b,d,e}$ ), another finger switching at  $\mathbf{q}_2$  can be applied to switch to a grasp on edge  $E_b, E_d, E_e$  as

desired.

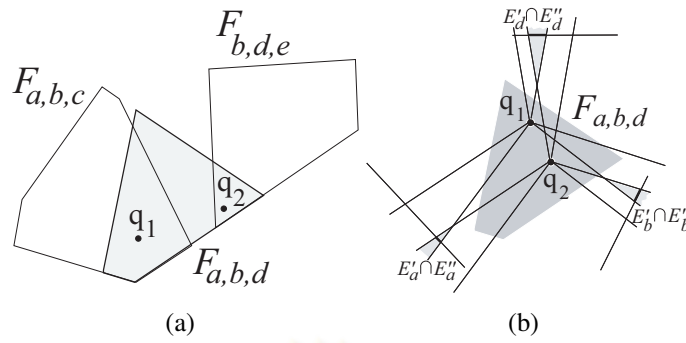


Figure 3.10: Finger aligning: (a) moving between non-overlapping focus cells, (b) moving locally within a focus cell

In fact, for 2-finger and concurrent grasps, changing grasping configuration within the same focus cell is the process we referred to as finger aligning. This process can be accomplished by taking advantage of the idea that force-closure can be maintained during finger sliding, finger rolling, or finger switching within an independent contact region. We illustrate in the case of concurrent grasps. Let us consider Fig. 3.10(b) showing configuration points  $q_1$  and  $q_2$  in the same focus cell  $F_{a,b,d}$ . The inverted friction cones at  $q_1$  intersect the three grasped edges in the three independent contact regions  $E'_a$ ,  $E'_b$  and  $E'_c$  and likewise the inverted friction cones at  $q_2$  intersect the three grasped edges in  $E''_a$ ,  $E''_b$  and  $E''_d$ . Suppose that the three fingers are at  $x_a \in E'_a$ ,  $x_b \in E'_b$  and  $x_c \in E'_c$ . This can be represented by  $q_1$ . To move from  $q_1$  to  $q_2$ , we move the three fingers from  $x_i$  to  $x'_i \in E'_i \cap E''_i$  ( $i = a, b, c$ ). It is sufficient to ensure force-closure during the fingers' motion by maintaining that the fingers are in the independent contact regions of  $q_1$  during the entire process. This can be done by rolling or sliding the fingers along the grasped edges from  $x_i$  to  $x'_i$  ( $i = a, b, c$ ). Instead of rolling or sliding, it is also possible to apply finger switching within each independent contact region by placing a free finger at  $x'_i$  and lifting off the finger at  $x_i$ . Because there is only one free finger during a concurrent grasp, this kind of finger switching can be performed in one independent region at a time.

By continuity, for any point in a focus cell of 2-finger or concurrent grasps, there exists a neighborhood for which the independent contact regions of the point intersect the independent contact regions of every point in the neighborhood. That is, there always exists a finger repositioning sequence to move between any pair of configuration points in the same focus cell.

Finger aligning for parallel grasps is trivial from the construction of a grasp set

that is formed by a union of connected convex hulls. Each vertex in the switching graph corresponds to exactly one grasp set. Every grasp in each vertex can be repositioned to another grasp of the same vertex by finger aligning because of continuity in a set of parallel grasps for each triple of polygon's edges.

### 3.3.4 Computing Switching Graph

To construct a switching graph, all of its vertices and edges have to be found. The constructions of switching graphs for all grasp types will be explained in this section. In the proposed algorithms, required information about an edge is maintained in a structure *EdgeStruct*. An instance of *EdgeStruct* for an edge contains two fields which are (1) *id*: the number uniquely identifying the edge, and (2) *normalAngle*: the angle between the internal normal of the edge and the x-axis written in radian in the range from 0 to  $2\pi$ . The input of the algorithm is an array *allEdge*[1..*n*] containing *EdgeStruct* instances for all edges of the polygon. The algorithm begins by sorting *allEdge* in an increasing order of the field *normalAngle* then constructs an array *upper*[1..*m*<sub>1</sub>] containing all *EdgeStruct* instances such that the field *normalAngle* is in the range  $[0, \pi)$  and an array *lower*[1..*m*<sub>2</sub>] containing all *EdgeStruct* instances in array *allEdge* that are not in array *upper*. The algorithm sorts *upper* in the increasing order of *normalAngle* and sorts *lower* in the decreasing order of *normalAngle* (this takes  $O(n)$  time since *upper* and *lower* are constructed from *allEdge* which is already sorted).

#### 3.3.4.1 Computing Vertices of Concurrent Grasps

We compute all focus cells to identify vertices of all concurrent grasp sets. Computing all focus cells requires identifying all triple of edges having concurrent grasps satisfying Proposition 3.6. Instead of enumeratively checking all triples, the number of candidate triples can be significantly reduced by considering only those triples whose internal normals  $\theta$ -positively span the plane. Let us present an algorithm for generating these candidate triples and then discuss how it works. The algorithm proceeds as described in the following pseudocode.

```

1: for  $i = 1$  to  $m_1$  do
2:    $\alpha = \text{upper}[i].\text{normalAngle}$ 
3:    $j = 1$ 
4:   while  $j \leq m_2$  and  $\text{lower}[j].\text{normalAngle} \geq \alpha + \pi + 2\theta$  do
5:      $\beta = \text{lower}[j].\text{normalAngle}$ 
6:     for each  $k$  such that
7:        $\text{allEdge}[k].\text{normalAngle} \geq \beta - \pi + 2\theta$  and
8:        $\text{allEdge}[k].\text{normalAngle} \leq \alpha + \pi - 2\theta$  do
9:         generate candidate triple of edges:
10:         $\{\text{upper}[i].\text{id}, \text{lower}[j].\text{id}, \text{allEdge}[k].\text{id}\}$ 
11:        $j = j + 1$ 

```

This algorithm is based on the idea that selecting one normal restricts how the next one can be selected. The algorithm selects the first normal from the upper half of the unit circle (line 1) and the second normal from the lower one (line 4). This is due to the fact that three vectors cannot be in the same half of the unit circle when they  $\theta$ -positively span the plane. According to Definition 3.3, once the first normal is selected, it is needed that the angle between the first and the second normals is smaller than  $\pi - 2\theta$ . This amounts to choosing the second normal in the lower circle and outside the cone with half angle  $2\theta$  and centered on the vector opposite to the first normal (Fig. 3.11(b)). This results in two regions where the second normal may be chosen (regions A and B in Fig. 3.11(b)). However, the region starting at smaller angle (region B) need not be considered because selecting the second normal from this region would lead to generating triples that were already generated in previous iterations (i.e., generating the third normal that was already considered as the first or second normals in previous iterations). Once the first and second normals are determined, Definition 3.3 is used again to specify the range of angles where the third normal can be selected (line 6 and region C in Fig. 3.11(c)). Note that although the upper bound running time of this algorithm is  $O(n^3)$ , it is in practice output sensitive and efficient. This claim is supported by experimental results in Section 3.5 that the number of the candidate triples generated from the presented algorithm varies closely with the number of focus cells found for polygons with varying number of edges.

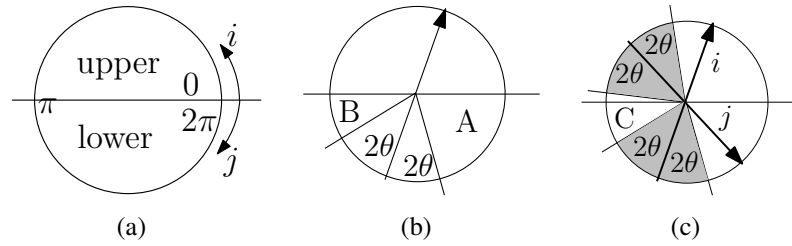


Figure 3.11: Generating candidate triples of concurrent grasps (see text)

### 3.3.4.2 Computing Vertices of 2-finger Grasps

Before computing focus cells for 2-finger grasps, the angle between the normals of two grasped edges is necessary to be in range  $(\pi - 2\theta, \pi + 2\theta)$ . The algorithm starts with selecting one edge from *allEdge* in range  $[0, \pi + 2\theta)$  because an edge out of this range induces redundant pairs of edges (Fig. 3.12(b)). Let *normalAngle* of this edge be  $\alpha$ , the other edge is restricted by its normal being in range  $(\alpha + \pi - 2\theta, \alpha + \pi + 2\theta)$  (Fig. 3.12(a)) and not exceeding  $2\pi$  to avoid redundancy. The generated candidates are then computed for focus cells. A non-empty focus cell is associated with a vertex in the switching graph.

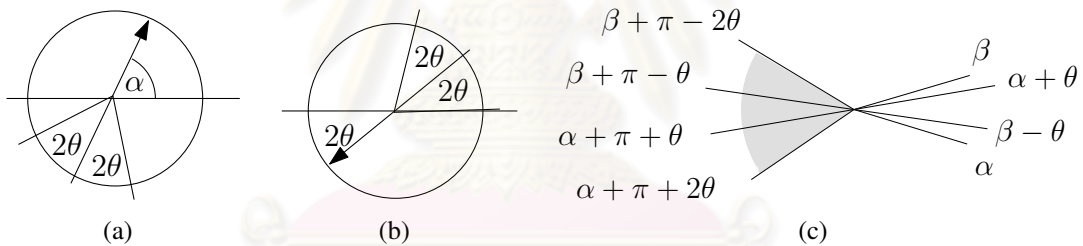


Figure 3.12: Generating pairs of 2-finger grasps and triples of parallel grasps

### 3.3.4.3 Computing Vertices of Parallel Grasps

To participate a switching graph, we start by building all vertices of parallel grasps. Since solving linear constraints of three grasped edges that induce null polytopes is worth nothing, we need a condition that can check the existence of a parallel grasp on three polygonal edges. Proposition 3.8 can be extended to cover an existence of a parallel grasp on three polygonal edges. We define a union volume  $U_{a,b,c}^a$  of polygonal edges  $a, b$  and  $c$  on edge  $a$  as the union of all common cones  $C_{a,b,c}^\cap$  originated on every points of edge  $a$ . A union volume also divides the plane into two outer regions. We can uniquely identify the edge that contains a center point in the same way as the case of point. This edge will be called the center edge. Fig. 3.13 illustrates a union volume.

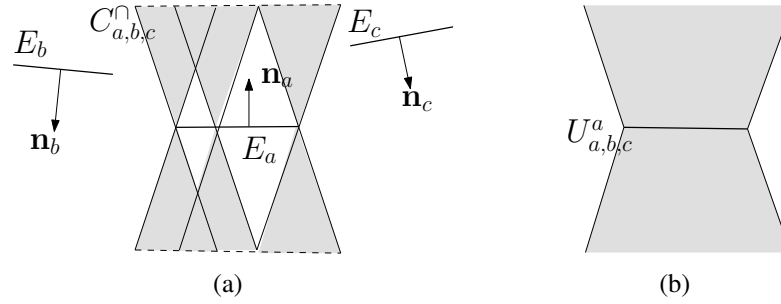


Figure 3.13: Computing a vertex of parallel grasps: (a) Three edges and a common cone drawn on some points on center edge. (b) A union volume.

**Proposition 3.9** *A necessary and sufficient condition for the existence of a parallel grasp on three polygonal edges  $E_a$ ,  $E_b$  and  $E_c$ , whose normals are  $\mathbf{n}_a$ ,  $\mathbf{n}_b$  and  $\mathbf{n}_c$ , is that the two following conditions hold. (Pd) a common cone  $C_{a,b,c}^{\cap}$  is not empty. (Pf) Let us assume that the edges that contain a center point is  $E_a$ . The edges  $E_b$  and  $E_c$  do not entirely lie in the same outer region separated by the union volume  $U_{a,b,c}^a$ .*

*Proof:* Let  $p_b$  be a point on  $e_b$  and  $p_c$  be a point on  $e_c$ . If two edges  $e_b$  and  $e_c$  do not entirely lie in the same outer region, then there exists  $p_b$  and  $p_c$  such that the line  $\overline{p_b p_c}$  intersects with the union volume. According to the definition of the union volume, the point on the intersection of  $\overline{p_b p_c}$  and the union volume must lie in a common cone of some point on  $e_a$ . Let us assume that the origin of that common cone is  $p_a$ . Three points  $p_a$ ,  $p_b$  and  $p_c$  must form a parallel grasp according to Proposition 3.8. This complete the proof for the sufficient condition.

For the necessary side, since  $e_b$  and  $e_c$  entirely lie on the same outer region, every pair of point  $p_b$  on  $e_b$  and  $p_c$  on  $e_c$  also lies outside the union volume. From the definition of the union volume, we know that for every point  $p_a$  on  $e_a$ , any pair of  $p_b$  and  $p_c$  will lie outside the common cone originated at  $p_a$ . From Proposition 3.8, we know that there can not be a parallel grasp. Thus, the proof is completed. ■

**3.3.4.3.1** We compute all vertices by using a condition Pd in Proposition 3.9 for pruning. The pruning starts by iteratively selecting the first edge. It limits that the angle of a normal vector between itself and a second edge must be less than  $2\theta$ . Let  $\alpha$  be the

value of *normalAngle* of a first edge and  $\beta$  be the value of *normalAngle* of the second edge. Clearly,  $\beta$  has to be selected in range  $(\alpha + \theta, \alpha - \theta)$ . The value of *normalAngle* of a third edge is restricted in the range  $(\beta + \pi - 2\theta, \alpha + \pi + 2\theta)$  (Fig. 3.12(c)). All generated triples satisfying Pd are then checked against Pf. If a triple passes the verification, it constitutes a vertex in the switching graph.

#### 3.3.4.4 Computing Edges

Once all vertices are computed, every pair of concurrent grasp vertices having two common edges are checked for intersection of the associated focus cells. Also, the focus cells of every pair of concurrent grasp vertex and 2-finger grasp vertex having one common edge are checked for intersection. If the intersection is not empty, an edge is created in the graph for linking the two vertices that represent the two focus cells. According to the number of fingers, any two vertices representing sets of 2-finger grasps are always adjacent.

For parallel grasps, every pair of parallel grasp vertices having two common edges are checked for intersection of the projections of the polytopes associated with the vertices as described in Section 3.3.2.3. To save time, when a vertex is computed in the first step, we do a preprocessing of computing a projection of its polytope on all three pairs of planes (plane  $(u_a, u_b)$ , plane  $(u_b, u_c)$  and plane  $(u_a, u_c)$ ). If the intersection on subspace is not empty, an edge linking these vertices is created. Vertices of 2-finger grasps are also considered for switching to parallel grasps. A parallel grasp vertex and a 2-finger grasp vertex having at least one common edge are checked for intersection of their graspable regions on the common edge.

### 3.4 Using Switching Graph

A switching graph provides a tool for planning a regrasp sequence. A graph search is performed to find a path joining the two vertices representing the two grasp sets. A path connecting the vertex containing the initial grasping position and the vertex containing the required grasping position indicates a sequence of edges that a finger switching should be performed. However, a path in a switching graph does not directly determine which contact points on grasping edges are to be used in each step. For a pair of vertices having an edge connecting them, a switching graph tells us that we can switch between two grasps on these edges but it does not tell which contact points that we can do a finger switching. This section describe a method of transforming a path in a switching graph to

an actual regrasp sequence.

First, let us consider a finger switching. Finger switching takes place when we move from one vertex to another vertex in a graph. An edge in the graph tells us that a finger switching is viable. We have to find two grasps on each vertex that have non-switching contact points on common edges. For finger switching among concurrent grasps or between concurrent grasps and 2-finger grasps, the intersection of the two associated focus cells is used to compute switchable regions on the grasped edges. A point in the intersection indicates regions on the common edges which fingers can be positioned on them and form two different grasps with the two switched edges. The calculation of the switchable regions has been described in Section 3.3.2.1 and 3.3.2.2.

In the case of switching between parallel grasps, we pick a point from the intersection of the projections of polytopes described in Section 3.3.2.3. That point indicates two actual points on non-switching edges. The next step is to find a point forming a grasp of the first vertex and a point forming a grasp of the second vertex. Let us consider a polytope defined in Section 3.3.1.3. Once a value of  $(u_b, u_c)$  in the intersection of projected  $P_1$  and  $P_2$  is chosen, we can construct a set of feasible contact points for the other two fingers by solving the linear system in (1)-(7) with the fixed values  $u_b$  and  $u_c$ . To switch between a parallel grasp and a 2-finger grasp, a common contact point is picked from the intersection of graspable regions on the common edge. Contact points on the remaining switched edges of parallel grasps are obtained by solving the same linear system with the fixed value of parameter on the common edge. The graspable region on the switched edge for 2-finger grasps is obtained by projecting the double-sided cone originated from the common contact point to the switched edge. The intersection of the cone and the edge indicates a region that a finger can be placed to form a 2-finger grasp with the common contact point.

Next, let us consider a finger aligning. We exemplify in 3-finger grasp situation. Finger aligning may be required in-between two finger switching, i.e., when we just traversed from vertex  $v_{a,b,c}$  to vertex  $v_{b,c,d}$  and about to move to the next vertex  $v_{b,c,f}$ . Let us assume that the first finger switching is just performed and we currently are in a grasp represented in  $v_{b,c,d}$ . In order to perform the next finger switching, i.e., to move to the vertex  $v_{b,c,f}$ , the grasping position must have two contact points in common with the final grasp. An appropriate grasping configuration is computed as described earlier in this section when we have to change from the finishing grasp of the first switching to the a next switching. Since these two grasps lie on same polygon's edges, we can change the



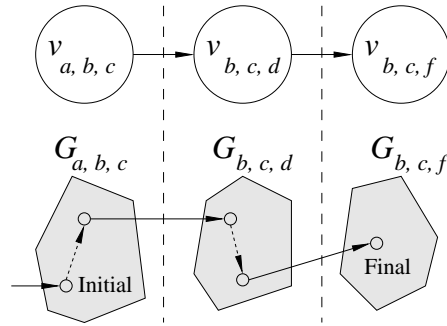


Figure 3.14: A corresponding between vertices and edges in a switching graph and a finger switching and a finger aligning. A dashed line in the bottom figure represents a finger aligning while a solid line represents a finger switching

current grasp to an appropriate grasp for the next switching by a finger aligning. Fig. 3.14 shows the corresponding between a switching graph and the actual action performed on a regrasp.

Since the switching graph contains sets of grasps, additional constraint, such as finger kinematics, may be incorporated as a search policy to find a sequence that meets additional requirement. Once a path is found, for each pair of consecutive vertices in the path, a grasp in the associated grasp set is chosen such that a finger switching can occur. Again, the grasp can be selected such that the resulting grasps optimize some criteria. After these grasps are computed, finger aligning can be planned to complete the sequence. An advantage of this approach is that a path in the graph represents a set of regrasp sequences, not just one. This allows selecting sequences based on additional constraint or any fine tuning on the sequences to be performed more efficiently than an approach that returns one sequence at a time.

### 3.5 Implementation and Results

We have implemented the regrasp planning for concurrent grasps, parallel grasps and 2-finger grasps based on the switching graph concept. The program is written in C++ using LEDA library (Mehlhorn and Naher, 2000). To achieve accuracy, rational numbers supported by LEDA are used in geometric computation. All run times are measured on a PC with a 2.8 GHz CPU.

Some test polygons with varying number of edges are shown in Figure 3.15. The results classified for each grasp type are in Table 3.1 showing the numbers of candidate sets, vertices and connected components of switching graphs. Table 3.2 shows the overall results when all grasps are taken into account and combined in one switching graph.

The results show that concurrent grasps are quite complete in themselves since the combination with 2-finger grasps and parallel grasps decreases the number of connected components only for the object in Fig. 3.15(b). The numbers of parallel grasp vertices are small comparing with concurrent grasps but they provide more 3-finger grasps that do not satisfy the strong condition in Proposition 3.4. 2-finger grasps play a role as transitions between concurrent grasps and parallel grasps. Moreover, they can decrease the number of connected components as we can see in the cases of parallel grasps for all objects and concurrent grasps for the object in Fig. 3.15(b).

Some regrasp sequences are presented in Fig. 3.16 - 3.20. Fig. 3.16 shows a regrasp sequence for only concurrent grasps. A regrasp sequence for parallel grasps is shown in Fig. 3.17. The cones in this figure are common cones for parallel grasps. Fig. 3.18 presents an example when 2-finger grasps are taken into account with concurrent grasps. Dashed line segments represent line joining two contact points which form 2-finger grasp. An example of parallel grasps and 2-finger grasps is appeared in Fig. 3.19. The cones in Fig. 3.19(b) and Fig. 3.19(i) of the parallel grasps are the common cones. Fig. 3.20 shows a regrasp sequence from a concurrent grasp to a parallel grasp. The cones in Fig 3.20(a)-(d) are double-sided friction cones whereas the cones in Fig 3.20(f) are common cones for the parallel grasp.

In conclusion, the most important grasp type is concurrent grasp which covers the most number of vertices and captures the connectivity of the switching graph. However, more grasps found induces more running time from the computation of edges linking switchable vertices. Also, 2-finger grasps cannot be neglected because they are transition between concurrent grasps and parallel grasps which cannot directly switch to the other different type. In practice, 2-finger grasps can shorten a path from two vertices representing the same grasp type.

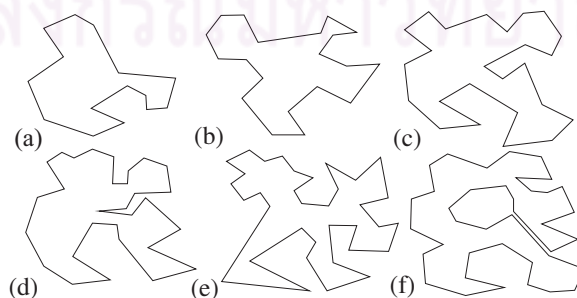


Figure 3.15: Test polygons

Table 3.1: Results of the algorithm for all grasp types

Fig.	#edge	2-finger grasps		concurrent grasps			parallel grasps		
		#vertex	#can	#vertex	#can	#CC	#vertex	#can	#CC
3.15(a)	15	10	13	43	61	1	11	11	4
3.15(b)	20	12	26	77	121	6	31	40	4
3.15(c)	25	21	40	185	250	2	81	101	3
3.15(d)	30	22	42	407	577	1	62	78	3
3.15(e)	35	26	64	550	853	2	122	162	4
3.15(f)	40	37	97	736	1074	1	249	358	1

Table 3.2: Combined results

Fig.	#vertex	#CC	time(s)
3.15(a)	64	1	1.05
3.15(b)	120	3	1.89
3.15(c)	267	2	6.34
3.15(d)	491	1	12.13
3.15(e)	698	2	20.98
3.15(f)	1022	1	38.27

### 3.6 Summary

We have proposed a method for solving the regrasp planning problem for a polygon. A hand using in this work is assumed four free-flying fingers. Our method provides general solutions represented by a graph which allows us to plan a regrasp sequence by using a graph search. Since an obtained result is a set of general solutions satisfying force-closure thus other constraints can be taken into account to determine an appropriate regrasp sequence for a given hand platform. The experimental results show the efficiency of our algorithm which can cover many sets of grasps. The switching graphs have a few number of connected components which means that any two vertices in a graph are mostly connected.

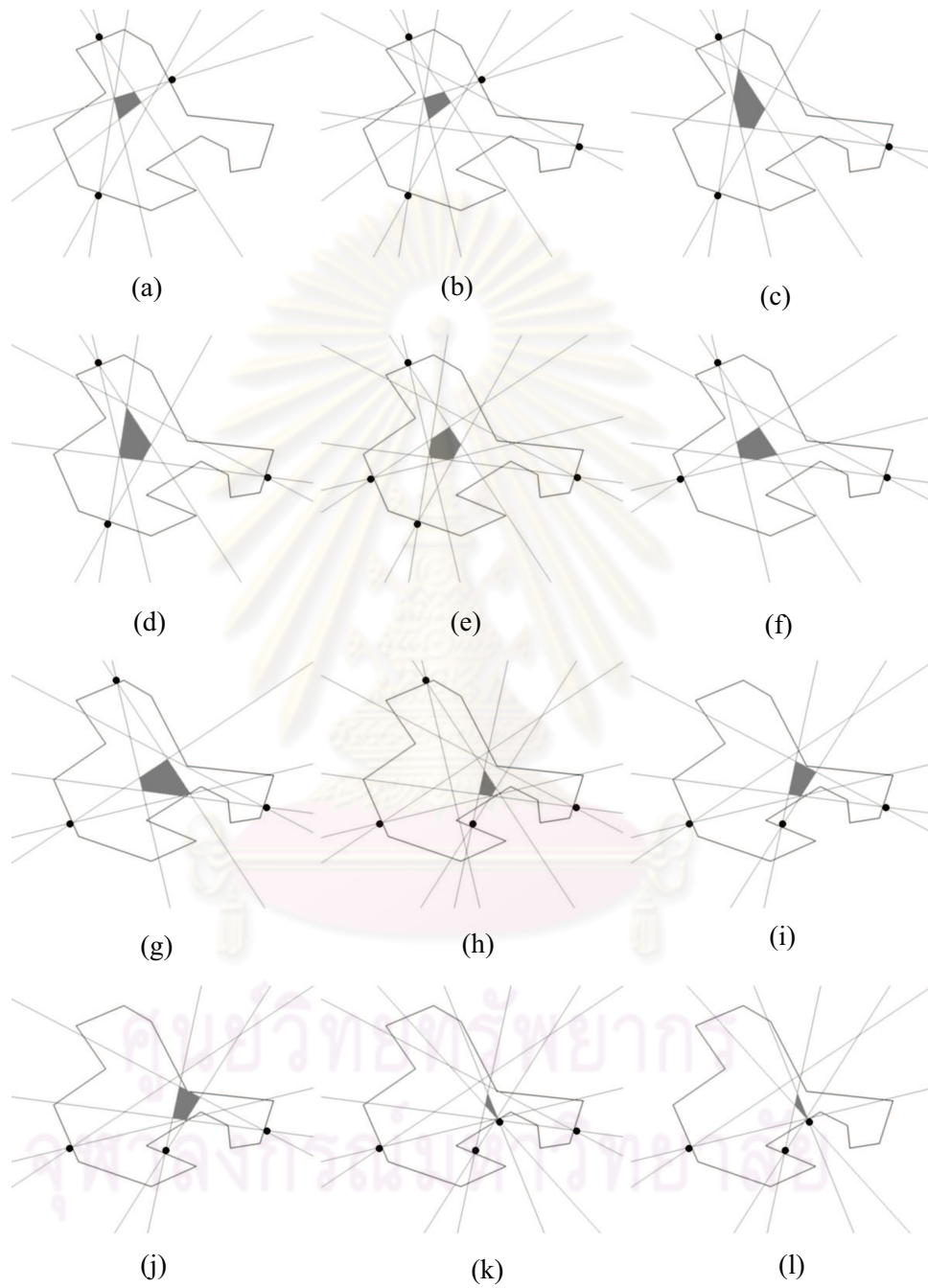


Figure 3.16: A regrasp sequence for concurrent grasps

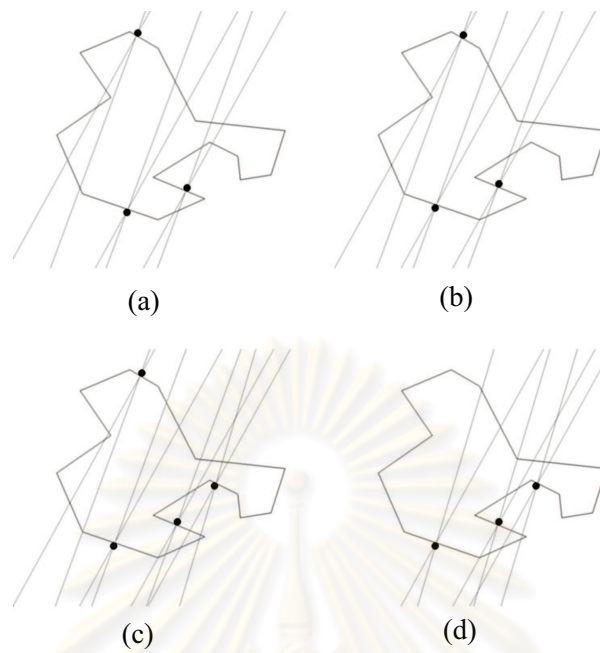


Figure 3.17: A regrasp sequence for parallel grasps

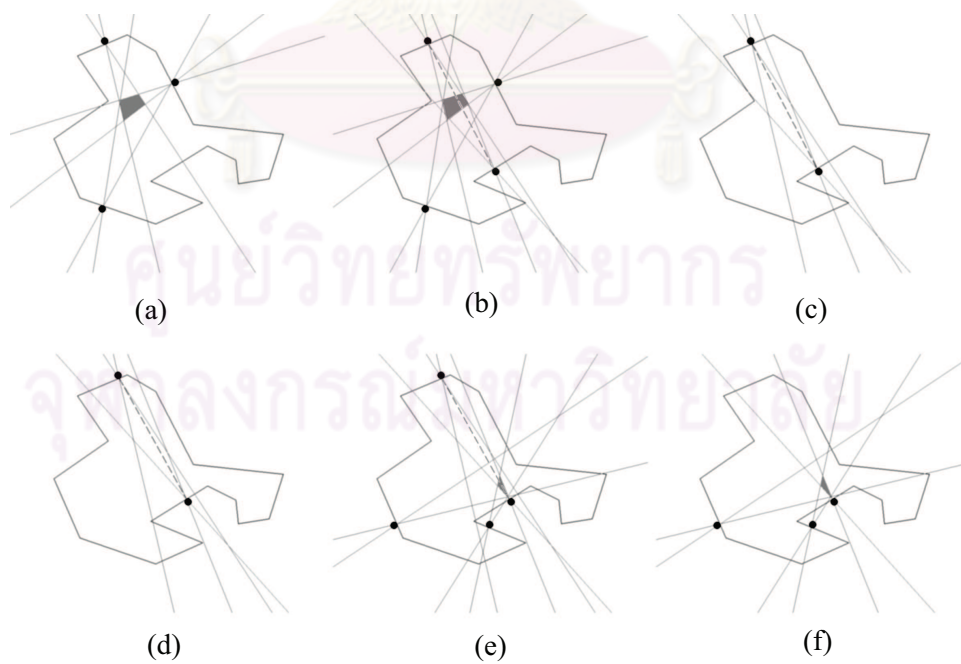


Figure 3.18: A regrasp sequence for concurrent grasps and 2-finger grasps

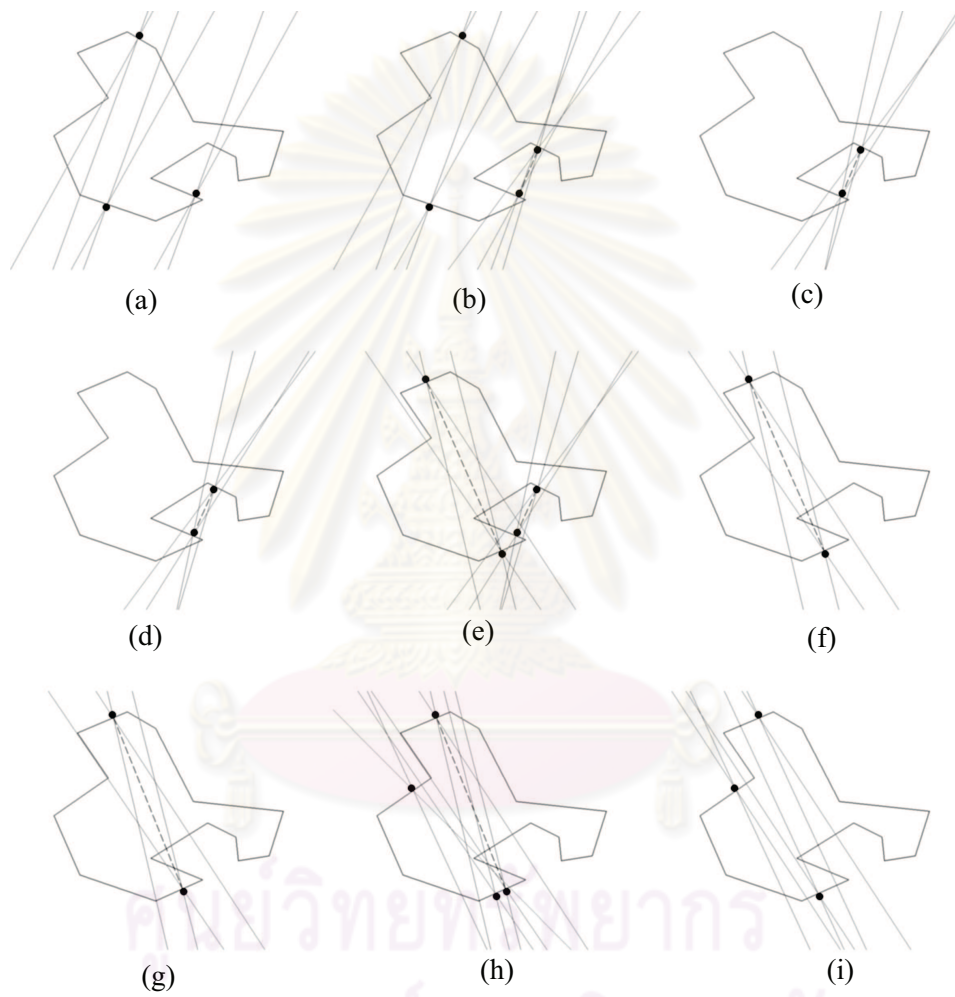


Figure 3.19: A regrasp sequence for parallel grasps and 2-finger grasps

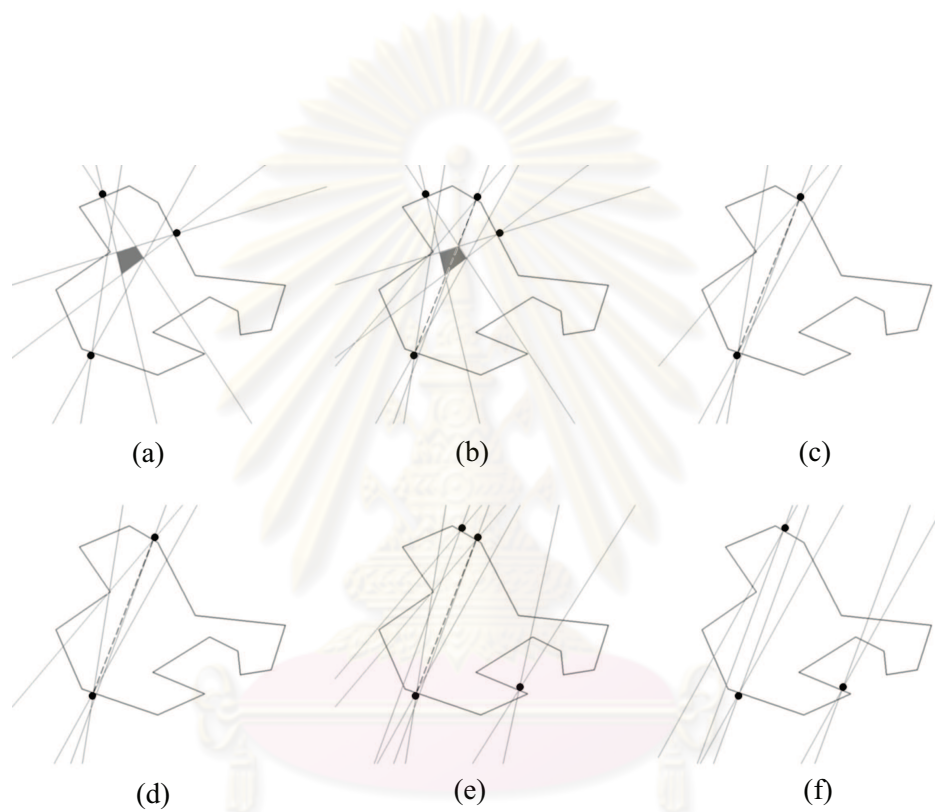


Figure 3.20: A regrasp sequence for all grasp types

ศูนย์วิจัยทรัพย์สินทางปัญญา  
จุฬาลงกรณ์มหาวิทยาลัย

## CHAPTER IV

# REGRASP PLANNING FOR A POLYGON WITH A LARGE NUMBER OF EDGES

### 4.1 Introduction

This chapter addresses the problem of regrasp planning for a polygon with a large number of edges. We consider the case that uses minimum number of fingers to grasp and regrasp, i.e., 2-finger grasps is taken into account and one additional finger is used for a finger switching. Therefore, a hand applied in this work is assumed to equip with three fingers. To obtain the complete structure for regrasp planning of an object, all force closure grasps are computed in grasp space. Instead of naively constructing a switching graph from all uncombined sets of grasps computed from all combination of polygonal edges, we propose to merge sets of grasps that connect to one another into a connected set. In grasp space, a connected set allows continuous changing among any grasping configurations in the set, i.e., it allows continuous movements of fingers across polygonal edges. The obtained connected sets are then used to construct nodes of a switching graph. We propose an output sensitive algorithm which efficiently computes all edges of a switching graph. Regrasp planning then can be formulated as a graph search problem where nodes of the graph represent connected sets of force closure grasps while an edge connects two sets that can be changed between each other by finger switching. A method for finding the optimal solution of a finger switching is also presented.

### 4.2 Representing force closure grasps

In this section, we describe how to represent and construct the configuration space that characterizes all force closure grasps. The polygonal boundary of the rigid object to be grasped must not be self-intersecting but could be broken into many simple polygons. It is sufficient to consider only the case that the boundary is composed of at most two simple polygons: if there are more than two simple polygons that define the boundary, we can pick two at a time and run the same algorithm over all possible pairs.

The configuration space of the problem is clearly the contact space. A configuration



consists of two parameters, each of which defines where each finger is placed along the boundary of the grasped object. This section describes how to represent and construct the configuration space that characterizes all force closure grasps.

The entities of a polygon needed in our discussion are defined as follows. A simple polygon  $P$  is described by  $n$  distinct vertices  $\mathbf{v}_i \in \mathbb{R}^2$  where  $i \in \mathbf{Z}_n$ <sup>1</sup>. It is assumed that  $\mathbf{v}_i$  are arranged counterclockwise if they represent the outer boundary of the object, or arranged clockwise if they represent the hole inside the object. Edges  $E_i$  are line segments with endpoints  $\mathbf{v}_i$  and  $\mathbf{v}_{i+1}$ . Every point  $\mathbf{p}$  on the boundary of  $P$  can be mapped to the length of curve measured counterclockwise from  $\mathbf{v}_0$  to  $\mathbf{p}$  along the boundary. We will write  $length(\mathbf{p})$  to represent such length. Lengths of  $E_i$  can be computed by the equation  $l_i = \|\mathbf{v}_{i+1} - \mathbf{v}_i\|$ . It is obvious that  $L_i = length(\mathbf{v}_i) = \sum_{j \in \mathbf{Z}_i} l_j$ . We denote by  $L$  the total length of the boundary of  $P$ , which can be computed by  $L = \sum_{i \in \mathbf{Z}_n} l_i$ .

Next, let us define the tangents of  $E_i$  as  $\mathbf{t}_i = (\mathbf{v}_{i+1} - \mathbf{v}_i)/l_i$ . The normal vectors  $\mathbf{n}_i$  of  $E_i$  are unit vectors that are perpendicular to  $\mathbf{t}_i$  and point inward. Note that  $\mathbf{n}_i$  can be obtained by rotating  $\mathbf{t}_i$  counterclockwise by  $\pi/2$  radian. The cone of forces  $C_i$  that can be exerted on the edge  $E_i$  is defined by two vectors  $\mathbf{n}_i + (\tan \alpha)\mathbf{t}_i$  and  $\mathbf{n}_i - (\tan \alpha)\mathbf{t}_i$  where  $\alpha \in [0, \pi/2)$  is the half-angle of the friction cone.

Since we deal with two fingers that might not reside on the same polygon, two sets of entities for different polygons are needed. Let all entities defined above correspond to the polygon  $P$  which is in contact with the first finger, and let  $n', \mathbf{v}'_i, E'_i, length', l'_i, L'_i, L', \mathbf{t}'_i, \mathbf{n}'_i$  and  $C'_i$  be defined similarly for the polygon  $P'$  in contact with the second finger. If the two fingers are on the same polygon, then  $n = n', length = length', L = L'$  and  $X_i = X'_i$  where  $X = \mathbf{v}, E, l, L, \mathbf{t}, \mathbf{n}$  or  $C$ .

The configuration space  $\mathbf{C}$  of the two fingers is  $[0, L) \times [0, L')$ . Given a configuration  $(u, u') \in \mathbf{C}$ , we say that  $(u, u')$  composes a 2-finger grasp if and only if the two contact points  $length^{-1}(u)$  and  $(length')^{-1}(u')$  achieve force closure. (Recall that  $length$  is a function that maps a vertex into a number, so  $length^{-1}$  gives a vertex.) Let the *graspable set*  $G \subseteq \mathbf{C}$  be the set of all configurations that compose 2-finger grasps (Obviously,  $G$  is the set of all configurations contained in all force closure regions mentioned in Section ??). Also let the *graspable subsets*  $G_{i,j}$  be graspable regions on edges  $E_i$  and  $E'_j$  defined by

$$G_{i,j} = G \cap ([L_i, L_{i+1}] \times [L'_j, L'_{j+1}]).$$

<sup>1</sup> $\mathbf{Z}_n$  is a group of non-negative integers less than  $n$ . Addition and subtraction are computed modulo  $n$ .

### 4.2.1 Computing $G_{i,j}$

Clearly each  $G_{i,j}$  corresponds to configurations whose one finger is on  $E_i$  and the other is on  $E'_j$ . This problem of finding all force closure grasps on a pair of edges has been well studied. Using Proposition 3.1, it has been shown in (Faverjon and Ponce, 1991) that  $G_{i,j}$  can be defined by eight linear inequalities in the parameters  $u$  and  $u'$ . Here, let us present an easier method to define  $G_{i,j}$ . We define the inverted force cone  $-C'_j$  as  $\{-\mathbf{x} \mid \mathbf{x} \in C'_j\}$ . It was shown in (Nguyen, 1988b) that emptiness of  $C_{i,j} = C_i \cap (-C'_j)$  implies emptiness of  $G_{i,j}$ . If  $C_{i,j}$  is not empty, we claim that  $G_{i,j}$  can be defined by no more than six points on the bounding rectangle. The claim is justified as follows.

Since a 2-finger grasp can be either *compressive*(squeezing grasp) or *expansive*(stretching grasp), we define for simplicity  $DC_{i,j} = C_{i,j} \cup (-C_{i,j})$  as the double-sided cone of  $C_{i,j}$  so that both the stretching and squeezing cases can be dealt with together. Now we prove the above claim by examining  $DC_{i,j}$  centered on  $E_i$ . Let us first extend both sides of the edges  $E_i$  and  $E'_j$  to infinity, choose an arbitrary real number  $u$ , find  $\mathbf{p}(u) = \text{length}^{-1}(u)$  on  $E_i$ , then let  $DC_{i,j}(u)$  be the cone  $DC_{i,j}$  centered at  $\mathbf{p}(u)$ . The intersection  $I(u)$  of  $E'_j$  and the cone  $DC_{i,j}(u)$  is a line segment on  $E'_j$  which represents the region that the second finger can be placed to achieve force closure with the first finger at  $\mathbf{p}(u)$ . This means for a given position of the first finger  $u$ ,  $\text{length}'(I(u))$  is the corresponding graspable interval in the second finger's configuration space (Fig. 4.1(a, b)).

It is easy to see that if  $u$  moves by  $\Delta u$ ,  $\mathbf{p}(u)$  will move in the direction of  $\mathbf{t}_i$  by the same distance  $\Delta u$ , the endpoints of  $I(u)$  will move in the direction of  $-\mathbf{t}'_j$  by the distance proportional to  $\Delta u$ , and the endpoints of  $\text{length}'(I(u))$  will move in the  $-\Delta u$  direction by the distance proportional to  $\Delta u$  (Fig. 4.1(c, d)). These linear relationships imply that the graspable region is bounded by two straight lines. It is now obvious that cutting  $E_i$  and  $E'_j$  to their original lengths is equivalent to imposing four rectangular constraints  $u \geq L_i$ ,  $u \leq L_{i+1}$ ,  $u' \geq L'_j$  and  $u' \leq L'_{j+1}$  in the  $(u, u')$ -space (Fig. 4.1(e)). Therefore,  $G_{i,j}$  can be defined with no more than six points on the bounding rectangle. In the real implementation, all defining points of  $G_{i,j}$  can be found by computing endpoints of four intersections:  $DC_{i,j}(\mathbf{v}_i) \cap E'_j$ ,  $DC_{i,j}(\mathbf{v}_{i+1}) \cap E'_j$ ,  $DC_{i,j}(\mathbf{v}'_j) \cap E_i$ , and  $DC_{i,j}(\mathbf{v}'_{j+1}) \cap E_i$  (Fig. 4.2).

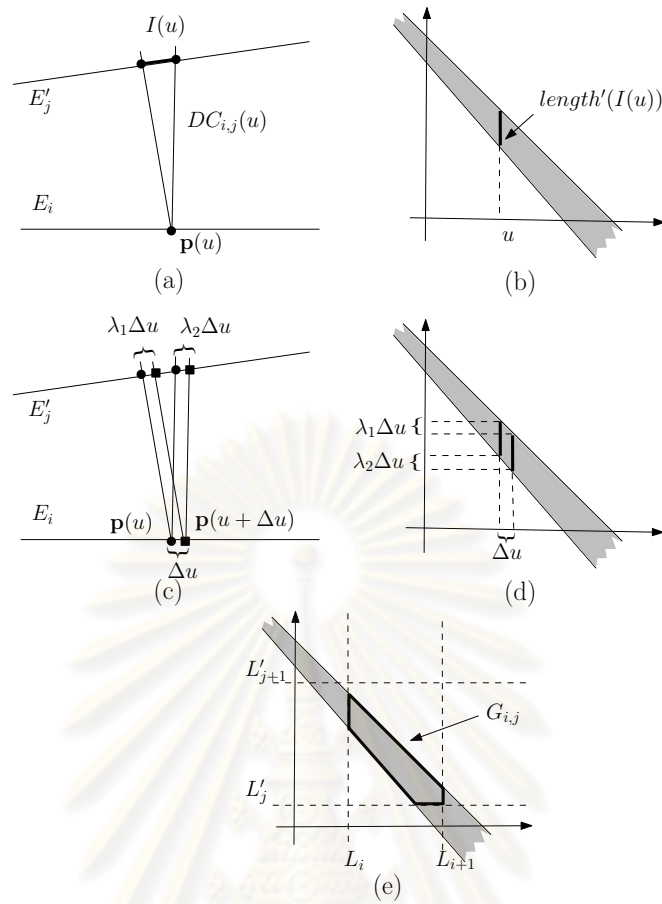


Figure 4.1: Computing  $G_{i,j}$ : (a)  $I(u)$  is the graspable region of the second finger when the first finger is at  $p$ . (b)  $length'(I(u))$  is an interval in  $u'$  configuration space. (c) Endpoints of  $I(u)$  move by the distances proportional to  $\Delta u$ . (d) Endpoints of  $length'(I(u))$  move by the same distances as endpoints of  $I(u)$ , giving two straight lines bounding the graspable region. (e)  $G_{i,j}$  is the result of cutting the infinite area by the rectangle.

#### 4.2.2 Extending Configuration Space

Since each finger can be positioned anywhere in its ICR in order to form a force closure grasp, and its position can be represented by one parameter, it is clear that the ICRs can be depicted in the configuration space as rectangles whose sides are parallel to  $u$  and  $u'$  axes. The mapping results in one rectangle if the ICRs do not contain  $v_0$  or  $v'_0$ , two rectangles if the ICRs contain  $v_0$  or  $v'_0$  but not both, or four rectangles if the ICRs contain both  $v_0$  and  $v'_0$  (Fig. 4.3). To eliminate the need to find ICRs with multiple rectangles, we extend the domain of  $length^{-1}$  and  $(length')^{-1}$  to the whole real line so they both become functions with periods  $L$  and  $L'$  respectively. ( $length$  and  $length'$  are no longer one-to-one.) The new  $G$  in the expanded configuration space  $\mathbb{R}^2$  can be defined from the old  $G$  with these periodic relations:

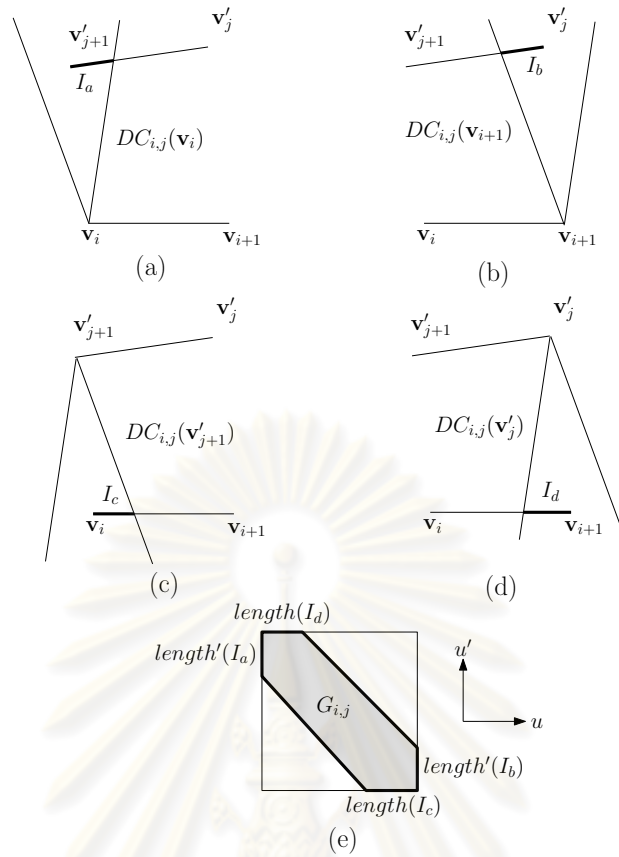


Figure 4.2: Extreme points of  $G_{i,j}$ : Four intersections  $I_a$ ,  $I_b$ ,  $I_c$  and  $I_d$  are shown in (a), (b), (c) and (d).  $G_{i,j}$  can be immediately defined by these intersections as shown in (e).

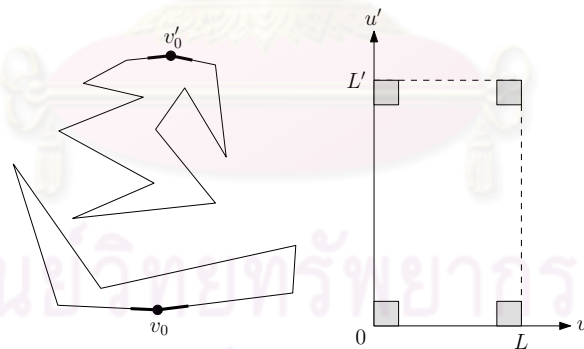


Figure 4.3: The independent contact region shown in thick lines (left) maps to four rectangles in the configuration space (right).

- $(u, u') \in G \Leftrightarrow (u + L, u') \in G$ .
- $(u, u') \in G \Leftrightarrow (u, u' + L') \in G$ .

We claim that despite infiniteness of  $G$ , the ICRs have one corresponding rectangle within  $G \cap [0, 2L] \times [0, 2L']$ . This is easily proved by the following argument.

- Suppose a position of one finger is given, there must be some positions of the second finger that do not form force closure with the first finger.
- It follows that all  $u$ -constant line segments in  $G$  are shorter than  $L'$  and all  $u'$ -constant line segments in  $G$  are shorter than  $L$ .
- Since the ICRs can be mapped into some rectangles in  $G$  whose sides are axis-parallel segments in  $G$ , one of these rectangles must lie inside  $[0, 2L] \times [0, 2L']$ .

The special case where the two fingers touch the same polygon can be handled with a smaller configuration space.  $G$  will be symmetric about the axis  $u = u'$ , which means we can cut out one half of  $G$  that lies above (or below) the line  $u = u'$  (Fig. 4.4(a)). The remaining part of  $G$  above  $u' > L$  (or to the right of  $u > L$ ) can also be eliminated because to every rectangle crossing the line  $u' = L$  (or the line  $u = L$ ), there corresponds a rectangle in  $[0, 2L] \times [0, L]$  (or  $[0, L] \times [0, 2L]$ ) that represents the same configurations (Fig. 4.4(b)). Finally, the region to the right of the line  $u = u' - L$  (or above the line  $u = u' + L$ ) is redundant because no point on this line is in  $G$  (Fig. 4.4(c)). Therefore, the region of consideration is the shaded portion as shown in Fig. 4.4(d).

### 4.2.3 Constructing $G$

Now we know that each  $G_{i,j}$  contains at most six defining vertices, so all  $G_{i,j}$  can be constructed within  $O(nn')$  time. In the final algorithm, we will need the polygonal representation of  $G$ , so adjacent  $G_{i,j}$  must be merged together into big pieces. Many simple polygons may be needed to define  $G$  because  $G$  does not have to be simple nor connected.

A vertex of some  $G_{i,j}$  is a *defining vertex of  $G$* , or *defines  $G$* , if it is a vertex on a boundary of  $G$ . It can be observed that a vertex  $v$  of some  $G_{i,j}$  defines  $G$  if and only if one of the following is true:

- $v$  is not at a corner of the bounding rectangle.
- $v$  is a corner of four bounding rectangles (one contains  $G_{i,j}$  and the other three are adjacent), but is not contained in some  $G_{k,l}$  bounded by these rectangles.

Note that if  $G_{i,j}$  is neither empty nor full (“full” means  $G_{i,j} = [L_i, L_{i+1}] \times [L'_j, L'_{j+1}]$ ), at least one vertex of  $G_{i,j}$  must be a defining vertex of  $G$ .

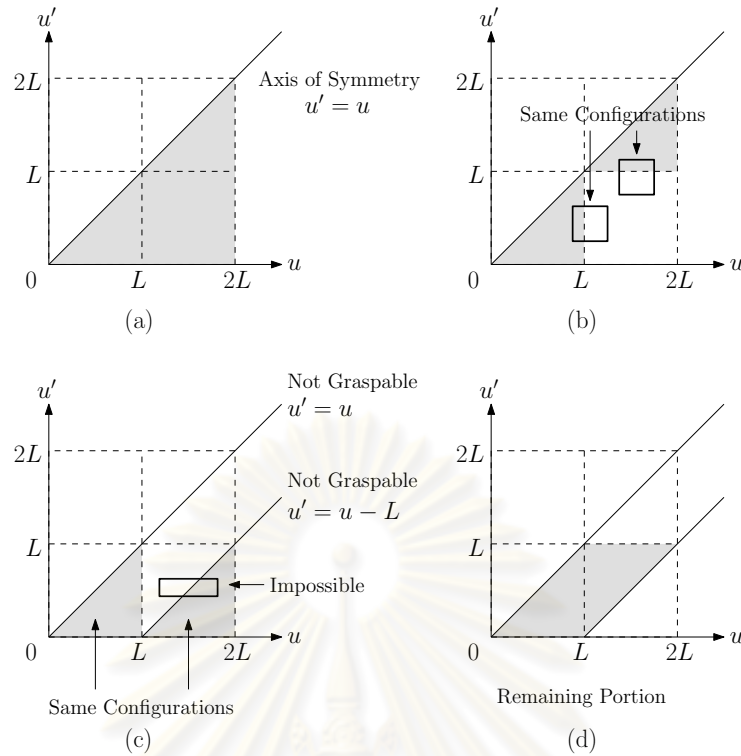


Figure 4.4: (a) The axis of symmetry is  $u' = u$ . (b) For each rectangle that crosses the line  $u' = L$ , there corresponds another rectangle in  $[0, 2L] \times [0, L]$  that crosses the line  $u = L$ . (c) The line  $u' = u$  and  $u' = u - L$  never intersect  $G$ , and the part of  $G$  to the right of  $u' = u - L$  represents the same configurations as the remaining portion in  $[0, L] \times [0, L]$ . (d) The remaining portion to consider is shown in the shaded area.

The algorithm to find all simple polygons that define boundaries of  $G$  is described as follows. Let us first attach a state “used/unused” to all vertices of all  $G_{i,j}$ . All vertices are initialized as “unused”. We scan through all values of  $i$  and  $j$ , and do the following:

- While  $G_{i,j}$  has an “unused” vertex  $v$  that defines  $G$ ,
  - It is clear that  $v$  is on a boundary of  $G$ , so we can *trace* the boundary of  $G$  from  $v$  until we get back at  $v$ .
  - All vertices traced along the way defines a simple polygon which is a boundary of  $G$ . Mark these vertices as “used”.

Note that *tracing* the boundary of  $G$  from  $G_{i,j}$  may involve many  $G_{k,l}$ .

The tracing process can be simplified by first defining adjacencies of vertices that define  $G$ . Situations when two vertices  $v_1$  and  $v_2$  that define  $G$  are adjacent in  $G$  are listed below:

- If  $v_1$  and  $v_2$  are adjacent in the polygonal representation of  $G_{i,j}$  and they lie on different sides of the bounding rectangle of  $G_{i,j}$ , they are adjacent in  $G$  (Fig. 4.5(a)).
- If  $v_1$  and  $v_2$  are adjacent in the polygonal representation of  $G_{i,j}$  and they lie on the same side of the bounding rectangle of  $G_{i,j}$ , we assume without loss of generality that  $v_1, v_2 \in \{L_i\} \times [L'_j, L'_{j+1}]$ .  $v_1$  and  $v_2$  will be adjacent in  $G$  if  $G_{i-1,j} \cap \overline{v_1 v_2} = \emptyset$  (Fig. 4.5(b)).
- If  $v_1$  and  $v_2$  belong to different pieces, i.e.  $G_{i,j}$  and  $G_{k,l}$ , we assume without loss of generality that  $v_1 \in G_{i,j}$ ,  $v_2 \in G_{i-1,j}$ .  $v_1$  and  $v_2$  can be adjacent in  $G$  if and only if they lie on the same segment  $\{L_i\} \times [L'_j, L'_{j+1}]$  and
  - $\overline{v_1 v_2} \subseteq G_{i-1,j}$  and  $\overline{v_1 v_2} \cap G_{i,j} = \{v_1\}$ , or
  - $\overline{v_1 v_2} \subseteq G_{i,j}$  and  $\overline{v_1 v_2} \cap G_{i-1,j} = \{v_2\}$  (Fig. 4.5(c)).

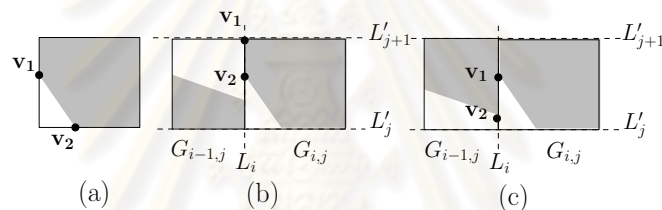


Figure 4.5: Adjacencies of vertices

A vertex on a boundary of  $G$  defines either an outer boundary or a hole of  $G$ . It is necessary to distinguish that the vertex defines an outer boundary or a hole. A sequence of vertices representing a polygon starting at  $v$  is obtained when a tracing gets back at  $v$  but we do not exactly know the orientation of the sequence. We firstly assume that it represents a simple closed polygon and then rearrange it into a counterclockwise order. To rearrange it, we start with the vertex that is extreme in  $u'$  direction denoted by  $v_i$ . Let its adjacent vertices be  $v_{i-1}$  and  $v_{i+1}$ . The cross product of  $\overline{v_{i-1}v_i}$  and  $\overline{v_i v_{i+1}}$  is calculated. Since the internal angle at the extreme vertex is convex and strictly less than  $\pi$ , they orient counterclockwise when the cross product is positive. If the cross product is negative, the sequence of the vertices is reversed.

The rearranged sequence defines an outer boundary when two adjacent vertices  $v_i$  and  $v_{i+1}$  of the sequence form the segment  $\overline{v_i v_{i+1}}$  such that the graspable region is on the left side of  $\overline{v_i v_{i+1}}$ , whereas the sequence represents a hole when the graspable region is on the right side of  $\overline{v_i v_{i+1}}$ . All vertices in the sequence are identified that they define an outer boundary or a hole for further use.

The number of vertices defining  $G$  can be decreased by eliminating collinear vertices. Let a connected boundary of  $G$  consist of  $n$  vertices  $v_i \in \mathbb{R}^2$  where  $i \in \mathbf{Z}_n$ . A vertex  $v_i$  can be eliminated when the slope of  $\overline{v_{i-1}v_i}$  is equal to the slope of  $\overline{v_i v_{i+1}}$ .

### 4.3 Finger Switching

Regrasp process which changes grasping configuration by placing an additional finger on desired contact point and then releasing one finger of the initial grasp is called finger switching. Intuitively, considering grasps on two different grasp sets, a finger switching can be performed when there exists a common contact point on the grasped object. In grasp space, the common contact points are computed in subspaces of one parameter. It requires projections of two grasp sets onto the subspaces. The projections are then checked for the intersection which indicates a set of common contact points. This operation involves with an edge in the switching graph. Considering two grasp sets associated with two nodes, existence of finger switching between these sets indicates an edge linking the two nodes.

Finger switching requires that one non-switching contact points must remain the same during the process. Formally, there will be an edge connecting a node  $v_a$  and a node  $v_b$  when there exists a couple of points  $(length^{-1}(u_a), length^{-1}(u'_a))$  where  $(u_a, u'_a) \in P_a$  and a couple  $(length^{-1}(u_b), length^{-1}(u'_b))$  where  $(u_b, u'_b) \in P_b$  such that  $u_a = u_b$  or  $u'_a = u'_b$ .

To check whether there exist grasps from two grasp sets that can switch to each other, we consider two polygons representing these grasp sets. The projection  $\pi_u(P_a)$  of  $P_a$  on the axis of parameter  $u_a$  (Fig. 4.6(a)) represents the set of points on the object that are possible to form 2-finger grasps with some points corresponding to the projection  $\pi_{u'}(P_a)$  of  $P_a$  on the axis of parameter  $u'_a$ . Similarly, the projections of  $P_b$  (Fig. 4.6(b)) represents the subspaces of 2-finger grasps. Note that  $\pi_u(P_a)$  and  $\pi_u(P_b)$  are in the same subspace, if the intersection between these two projections is not empty (Fig. 4.6(c)), then there exists points  $length^{-1}(u_a)$  on the object that form 2-finger grasps with  $length^{-1}(u'_a)$  and  $length^{-1}(u'_b)$  concurrently when  $u_a = u_b$  is satisfied. We apply the same process to check the existence of finger switching on the space of parameters  $u'_a$  and  $u'_b$  by considering  $\pi_{u'}(P_a)$  and  $\pi_{u'}(P_b)$ .



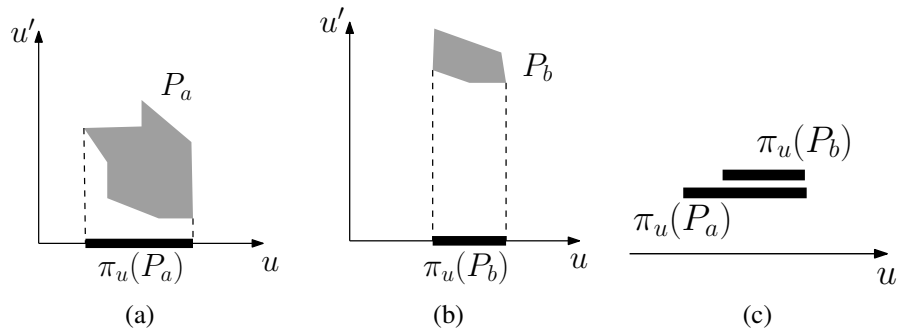


Figure 4.6: Finger switching: The projection of (a)  $P_a$ , (b)  $P_b$  on  $u$  parameter space. (c) Overlapping projections

#### 4.4 Finger Aligning

Finger aligning is a process for repositioning fingers by rolling or sliding them along edges of a polygon while maintaining a force closure grasp during the repositioning process. By applying this operation, we can change grasping configuration within the same connected set of grasps. This expresses the direct relation between finger aligning and a node of the switching graph.

Finger aligning is necessary as exemplified in the following instance. Let us consider Fig. 4.7(a). Obviously, because the current position of finger 1 cannot form a 2-finger grasp with the upper part, it is not possible to switch directly from the current grasp to a grasp that one finger is placed on the upper part using finger switching. However, somehow if the hand can continuously adjust finger 1 and 2 to change from the current grasp to a new grasp in Fig. 4.7(b), a finger switching can be performed to switch to another grasp by placing finger 3 on the upper part to form a 2-finger grasp with finger 1 (Fig. 4.7(c)) and then releasing the finger 2 (Fig. 4.7(d)).

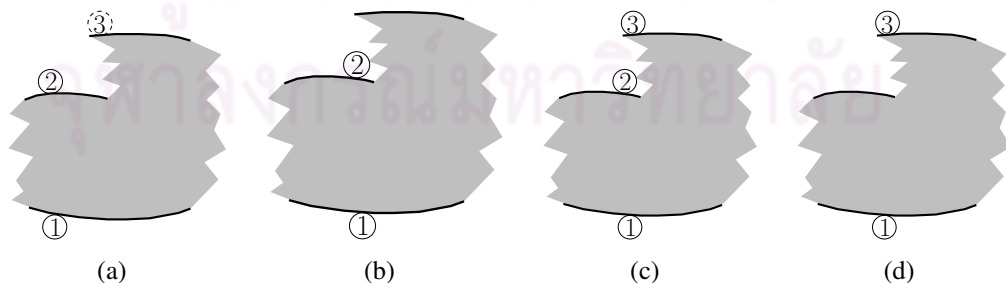


Figure 4.7: Finger aligning

Finger aligning is trivial from the construction of connected grasp sets  $P_1, \dots, P_m$ . Each node in the switching graph corresponds to exactly one grasp set. Every grasp in each node can be repositioned to another grasp of the same node by finger aligning

because of continuity in a connected set of 2-finger grasps.

#### 4.5 Constructing Switching Graph

To construct a switching graph, all of its vertices and edges have to be found. We compute all  $G_{i,j}$  and construct  $G$  to identify all connected polygons  $P_1, \dots, P_m$ . Each connected polygon is associated with a node of the switching graph.

To construct all edges, all polygons have to be checked for finger switching among them. Instead of exhaustively testing all pair of polygons, we apply a sweep algorithm to find overlaps among the projections of the polygons in a parameter subspace. Let the projection  $\pi_u(P_a)$  of a polygon  $P_a$  on the axis of parameter  $u_a$  be represented by an interval  $(l_a : h_a)$  where  $l_a$  is the lower endpoint and  $h_a$  is the higher endpoint. The lower endpoint and the higher endpoint are obtained from the leftmost vertex and the rightmost vertex of  $P_a$  respectively. The intervals of all polygons are used in our algorithm. We firstly sort all endpoints in increasing order and store the sorted endpoints into an event queue  $E$ . A priority queue  $Q$  is used to store intervals and identify overlaps among intervals. The priority of a interval is based on the value of its higher endpoint, less value has more priority. We are now ready to start the sweeping process from the first element in  $E$ . An endpoint is dequeue from  $E$ . If it is a lower endpoint, its associated interval is added into  $Q$ . Otherwise, if it is a higher endpoint, its associated interval is dequeued from  $Q$ . Clearly, the interval has the highest priority, we can use *ExtractMin* operation of the priority queue to remove it from  $Q$ . Let the dequeued interval be  $(l : h)$ . All remaining intervals in  $Q$  have the lower endpoints that less than  $h$  and the higher endpoints that higher than  $h$  therefore they overlap the interval  $(l : h)$ . As a consequence, the associated node of the interval  $(l : h)$  has edges linking nodes associated with the remaining intervals in  $Q$ . The process is repeated from dequeuing  $E$  and so on until  $E$  is empty. We also have to check overlaps in the subspace  $u'$  using the same algorithm. The pseudocode of edge construction is as follows.

- 1:  $E = \text{Sort}(l_1, h_1, l_2, h_2, \dots, l_m, h_m)$
- 2: **while**  $E$  is not empty **do**
- 3:    $e = E.\text{Dequeue}()$
- 4:   **if**  $e$  is lower endpoint **then**
- 5:      $Q.\text{Insert}(\text{interval of } e)$
- 6:   **else**
- 7:      $Q.\text{ExtractMin}()$
- 8:      $L = \text{all remaining elements of } Q$

```

9:      $v =$  the associated node of  $e$ 
10:    for all  $i \in L$  do
11:         $n =$  the associated node of  $i$ 
12:         $link(v, n)$ 
13:    end for
14: end if
15: end while

```

The construct of the event queue  $E$  takes  $O(m \log m)$  running time. A priority queue using a heap give performance  $O(1)$  to insert an element into  $Q$ . *Extract-Min* operation takes  $O(\log m)$ . For all endpoints, our output sensitive algorithm takes  $O(m \log m + mk)$  where  $k$  is the average number of overlapping intervals of one interval.

## 4.6 Using Switching Graph

### 4.6.1 Unconstrained Regrasp Sequence

A switching graph provides a tool for planning a regrasp sequence. A path connecting the node containing the initial grasping position and the node containing the required grasping position indicates a sequence of edges that a finger switching should be performed. However, a path in a switching graph does not directly indicate which contact points on grasping edges are to be used in each step. For a pair of nodes having an edge connecting them, a switching graph tells us that we can switch between two grasps from two grasp sets but it does not tell which grasping points that we can perform a finger switching. This section describe a method of transforming a path in a switching graph to an actual regrasp sequence.

First, let us consider a finger switching. Finger switching takes place when we move from one node to another node in a graph. An edge in the graph tells us that a finger switching is viable. We have to find two grasps on each node that have one non-switching contact point in common. We pick a point from the intersection of the projections described in section 4.3. That point indicates one actual non-switching point. The next step is to find a point forming a grasp of the first node and a point forming a grasp of the second node. Let us consider a polygon defined in section 4.3. Once a value of  $u_a$  or  $u'_a$  in the intersection of the projections of  $P_a$  and  $P_b$  is chosen, we can construct a set of feasible contact points for the other finger by intersecting  $P_a$  with the line passing  $u_a$  and parallel with the axis of  $u'_a$  or the line passing  $u'_a$  and parallel with the axis of  $u_a$

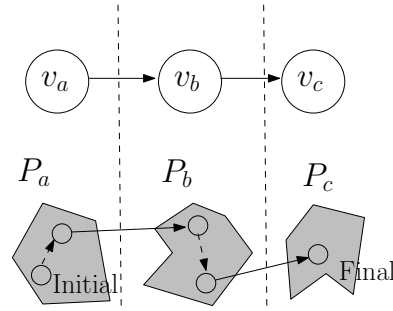


Figure 4.8: A corresponding between nodes and edges in a switching graph and a finger switching and a finger aligning. A dashed line in the bottom figure represents a finger aligning while a solid line represents a finger switching

Next, let us consider a finger aligning. Finger aligning may be required in-between two finger switchings, i.e., when we just traversed from node  $v_a$  to node  $v_b$  and about to move to the next node  $v_c$ . Let us assume that the first finger switching is just performed and we currently are in a grasp represented in  $v_b$ . In order to perform the next finger switching, i.e., to move to the node  $v_c$ , the grasping position must have one contact point in common with the final grasp. However, it might not be the current grasp. When an appropriate grasping configuration is computed as described earlier in this section, we have to change from the finishing grasp of the first switching to the a next switching. Since these two grasps are from the same connected set, we can change the current grasp to an appropriate grasp for the next switching by a finger aligning. Fig. 4.8 shows the corresponding between a switching graph and the actual action performed on a regrasp.

#### 4.6.2 Optimal Regrasp Sequence

In this section, we will plan for a regrasp sequence that independent contact regions (ICR) are locally optimized for each finger switching using the principle of  $L_\infty$  Voronoi diagram (Papadopoulou and Lee, 2001). We propose to accomplish this task by computing the  $L_\infty$  Voronoi diagram of the edges of  $G$ . Considering only part of the resulting  $L_\infty$  Voronoi diagram that lies inside  $G$ , we claim that the largest square in  $G$  must be centered at a vertex of the diagram. Before proving this claim, let us briefly review the concept of  $L_\infty$  Voronoi diagram.

The  $L_\infty$  Voronoi diagram is conceptually defined in the same manner as the ordinary Voronoi diagram except that  $L_\infty$  distance metric is used instead of the more familiar  $L_2$ . Recall that the  $L_\infty$  distance  $d(\mathbf{p}, \mathbf{q})$  between two points  $\mathbf{p} = (p_x, p_y)$  and  $\mathbf{q} = (q_x, q_y)$  is given by  $d(\mathbf{p}, \mathbf{q}) = \max(|p_x - q_x|, |p_y - q_y|)$ , i.e., the maximum of the differences in each coordinate, and the  $L_\infty$  distance  $d(\mathbf{p}, e)$  between point  $\mathbf{p}$  and segment  $e$  is defined to

be  $\min_{\mathbf{q} \in e} d(\mathbf{p}, \mathbf{q})$ , i.e., the shortest  $L_\infty$  distance between  $\mathbf{p}$  and any point on  $e$ . Let  $S$  be a set of segments in the plane. A point lies on the  $L_\infty$  Voronoi diagram of  $S$  if, among all segments in  $S$ , at least two of them are equally of the lowest  $L_\infty$  distance to the point (in other words, they are the  $L_\infty$  nearest segments of the point). Equivalently speaking, considering the definition of the  $L_\infty$  distance, a point is on the  $L_\infty$  Voronoi diagram of  $S$  if it is the center of a square that touches at least two of the segments in  $S$  without having its interior intersect with any. An  $L_\infty$  Voronoi edge is defined to be the set of all points on the  $L_\infty$  Voronoi diagram that are on the same straight line segment. A Voronoi vertex is defined to be the point at which at least two Voronoi edges meet (implying that the point is equally  $L_\infty$  far from at least three segments). Since every point on the  $L_\infty$  Voronoi diagram lies at least on one Voronoi edge, the  $L_\infty$  Voronoi diagram is essentially a network of straight line segments. Specifically, the  $L_\infty$  Voronoi diagram of  $S$  divides the plane into polygonal regions called Voronoi regions. Each Voronoi region is associated with a unique segment in  $S$  such that the region entirely contains the segment, and any point in the interior of the region is  $L_\infty$  closer to this segment than to any other segment in  $S$ .

ICR are defined by a rectangle in  $G$  whose shorter side length is maximum. We extend to the problem of optimizing ICR for a finger switching which involves two grasps concurrently. The measure of goodness of two rectangles  $R_a$  with side lengths  $a_1$  and  $a_2$  and  $R_b$  with side lengths  $b_1$  and  $b_2$  is given by  $f(R_a, R_b) = \min\{a_1, a_2, b_1, b_2\}$ . Our goal is to maximize  $f(R_a, R_b)$  such that the grasps represented by the centers of  $R_a$  and  $R_b$  can switch to each other.

To optimize the criterion, we use another representation of a square to describe ICR. Let  $\mathbf{v}$  be a point in  $G$  and let  $square(\mathbf{v})$  denote the largest square centered at  $\mathbf{v}$  that is fully contained in  $G$ . The size of  $square(\mathbf{v})$  is determined by  $size(\mathbf{v}) = \min_{\mathbf{p} \in \partial G} (d(\mathbf{v}, \mathbf{p}))$  where  $d(\mathbf{v}, \mathbf{p}) = \max(|u_{\mathbf{v}} - u_{\mathbf{p}}|, |u'_{\mathbf{v}} - u'_{\mathbf{p}}|)$ . Therefore, any largest square is described by its center  $\mathbf{v}$  and  $size(\mathbf{v})$ . Let  $\mathbf{v}_a \in P_a$  and  $\mathbf{v}_b \in P_b$ , it is clear that maximizing the criterion is equivalent to maximizing  $\min\{size(\mathbf{v}_a), size(\mathbf{v}_b)\}$ . We denote by  $\pi_u(\mathbf{v})$  and  $\pi_{u'}(\mathbf{v})$  the projections of a point  $\mathbf{v}$  on the axis of  $u$  and  $u'$  parameters respectively. The problem is transformed to locating the centers  $\mathbf{v}_a \in P_a$  and  $\mathbf{v}_b \in P_b$  of two squares such that  $\pi_u(\mathbf{v}_a) = \pi_u(\mathbf{v}_b)$  or  $\pi_{u'}(\mathbf{v}_a) = \pi_{u'}(\mathbf{v}_b)$  and  $\min\{size(\mathbf{v}_a), size(\mathbf{v}_b)\}$  is maximal.

Our algorithm exploits an important characterization of Voronoi edges which allows us to search squares centered on them to optimize the criterion. We claim that the largest  $square(\mathbf{v})$  must be centered on a Voronoi edge when one parameter of the point  $\mathbf{v} \in P$

is restrained. This is justified by the following argument:

We describe in the case that parameter  $u$  is restrained as shown by the dotted line in Fig. 4.9(a).

- If  $v$  is inside a Voronoi region whose upper and lower boundaries are a Voronoi edge and an edge of  $P$  ( $v_4$  in Fig. 4.9(a)),  $square(v)$  must have one corner on that polygonal edge. Moving  $v$  away from that polygonal edge will increase  $size(v)$ . We can move  $v$  in such direction until it reaches a Voronoi edge while  $square(v)$  is growing.
- If  $v$  is on a Voronoi region whose upper and lower boundaries are both Voronoi edges ( $v_2$  in Fig. 4.9(a)), moving  $v$  in one direction,  $size(v)$  is increasing or decreasing along the way, and  $size(v)$  is decreasing or increasing along the opposite way until it reaches a Voronoi edge ( $v_1$  and  $v_3$  in Fig. 4.9(a)).

This argument allows us to search two points  $v_a \in VE_a$  and  $v_b \in VE_b$  such that  $\pi_u(v_a) = \pi_u(v_b)$  or  $\pi_{u'}(v_a) = \pi_{u'}(v_b)$  for optimizing  $\min\{size(v_a), size(v_b)\}$  where  $VE_a$  and  $VE_b$  are sets of Voronoi edges of  $P_a$  and  $P_b$ .

Searching procedure begins with identifying an interval for finger switching by the intersection between  $\pi_u(P_a)$  and  $\pi_u(P_b)$  or  $\pi_{u'}(P_a)$  and  $\pi_{u'}(P_b)$ . We again describe in the case of an interval in the space of parameter  $u$ . Let  $\pi_u(P_a) \cap \pi_u(P_b) \neq \emptyset$  be denoted by an interval  $(l : h)$ . Voronoi edges in  $VE_a$  and  $VE_b$  that intersect this interval are considered. It is possible that we obtain many branches of Voronoi edges. All combinations of pairs of Voronoi edge branches are investigated, a pair consists of one Voronoi edge branch from  $VE_a$  and another branch from  $VE_b$ . For each pair, we divide the two branches using subintervals defined by all Voronoi vertices in the branches as shown by the dotted lines in Fig. 4.9(b). Voronoi vertices are used to determine subdivisions of two Voronoi edge branches because sizes of largest squares centered at them are critical. Each pair of subsets of two Voronoi edges in a subinterval is then searched for local optimization of the criterion. Let the two subsets be represented by two segments whose endpoints are  $s_a, t_a$  and  $s_b, t_b$ . Since the size of a square centered on a Voronoi edge linearly increases or decreases and a Voronoi edge is also linear, therefore the size of a square can be parameterized by a parameter  $\alpha$ . We define new size functions as

$$size_a(\alpha) = size(s_a) + \frac{\alpha}{r}(size(t_a) - size(s_a))$$

$$size_b(\alpha) = size(\mathbf{s}_b) + \frac{\alpha}{r}(size(\mathbf{t}_b) - size(\mathbf{s}_b))$$

where  $r$  is the length of the associated subinterval and  $\alpha \in [0, r]$ . Clearly,  $\alpha$  can be linearly inverted for positions on the segments. Let  $\uparrow, \downarrow$  be increasing and decreasing. The locally optimum is obtained as follows (Fig. 4.10).

- If  $\alpha \uparrow \Rightarrow size_a(\alpha) \downarrow$  and  $size_b(\alpha) \downarrow$ ,  $\alpha = 0$  induces a local optimum.
- If  $\alpha \uparrow \Rightarrow size_a(\alpha) \uparrow$  and  $size_b(\alpha) \uparrow$ ,  $\alpha = r$  induces a local optimum.
- If  $\alpha \uparrow \Rightarrow size_a(\alpha) \downarrow$  and  $size_b(\alpha) \uparrow$  and  $size_a(0) \leq size_b(0)$ ,  $\alpha = 0$  induces a local optimum.
- If  $\alpha \uparrow \Rightarrow size_a(\alpha) \downarrow$  and  $size_b(\alpha) \uparrow$  and  $size_a(r) \geq size_b(r)$ ,  $\alpha = r$  induces a local optimum.
- If  $\alpha \uparrow \Rightarrow size_a(\alpha) \downarrow$  and  $size_b(\alpha) \uparrow$  and  $size_a(0) > size_b(0)$  and  $size_a(r) < size_b(r)$ ,  $\alpha$  causing  $size_a(\alpha) = size_b(\alpha)$  induces a local optimum.
- The remaining cases are replica of the last three cases.

All pairs of segments from all subintervals are queried for local optima. Our approach is done when all combinations of pairs of Voronoi edge branches have been explored. The best local optimum is the optimal solution for a finger switching.

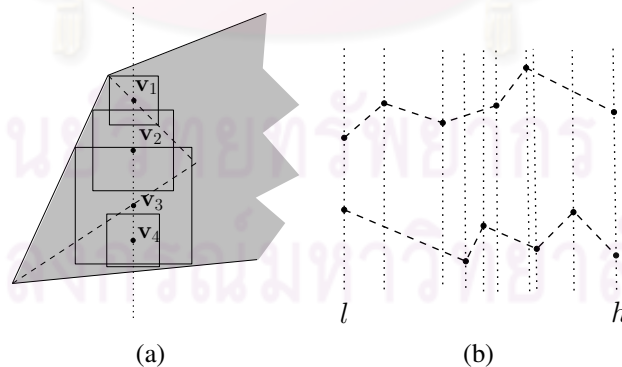


Figure 4.9: The largest square on Voronoi edges

## 4.7 Experimental Results

We have implemented the regrasp planning for a polygon with a large number of edges based on the switching graph concept. The program is written in C++ programming language. All run times are measured on a PC with a 2.4 GHz CPU.

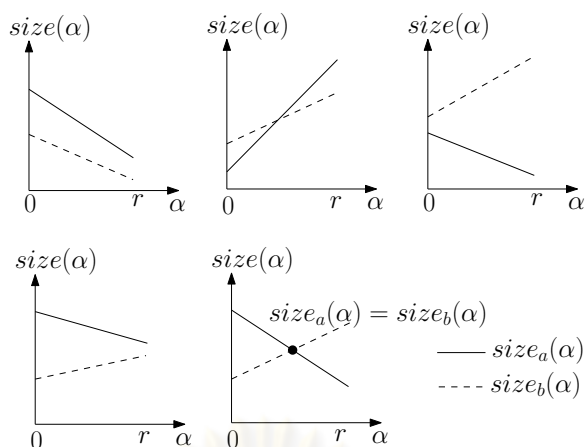


Figure 4.10: Determining a local optimum

Some test polygons with varying number of edges are shown in Fig. 4.11. We also vary the half-angle of the friction cone by  $10^\circ$ ,  $15^\circ$  and  $20^\circ$ .

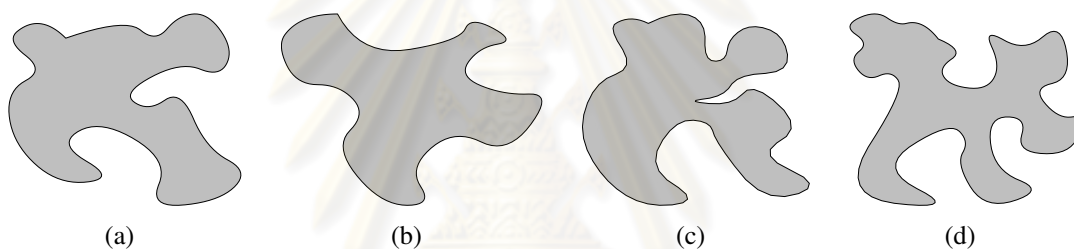


Figure 4.11: Test polygons with number of edges (a) 128 (b) 200 (c) 256 (d) 300

Table 4.1: Switching graph construction of  $10^\circ$  half-angle

Fig.	#node	#edge	#CC	time for nodes	time for edges
(a)	70	162	15	0.11	0.03
(b)	35	79	6	0.20	0.03
(c)	84	222	11	0.25	0.05
(d)	134	400	11	0.30	0.09

The results of switching graph constructions are shown in Table 4.1-4.3. They present for all test objects the number of connected polygons or nodes of switching graphs, edges of switching graphs, the number of connected components of the switching graphs, time spent in node and edge construction in second. The number of nodes of a switching graph depends on the object's shape. An object with more complexity produces more connected polygons. The number of connected components indicates probability to have a path joining any two nodes in the switching graph. The half-angle of the friction cone heavily effects the results. It's clear that larger friction cone induces larger feasible



Table 4.2: Switching graph construction of  $15^\circ$  half-angle

Fig.	#node	#edge	#CC	time for nodes	time for edges
(a)	65	194	5	0.16	0.03
(b)	41	121	4	0.22	0.02
(c)	85	323	3	0.31	0.06
(d)	150	630	4	0.39	0.19

Table 4.3: Switching graph construction of  $20^\circ$  half-angle

Fig.	#node	#edge	#CC	time for nodes	time for edges
(a)	58	230	3	0.17	0.03
(b)	41	156	2	0.27	0.03
(c)	84	424	2	0.38	0.06
(d)	143	782	2	0.45	0.08

grasp sets. This causes sets of force closure grasps to be merged more and connected polygons to be larger. As a result, the number of nodes decreases while the number of edges increases so that the number of connected components of a switching graph decreases. Run times of node constructions depend on areas of merged connected polygons which are inherited from the objects' shapes and the values of the half-angles. For an edge construction, a run time relates to the number of connected polygons.

An example of a regrasp sequence is presented in Fig. 4.12. The sequence is computed using the algorithm described in Section 4.6.1. The dashed lines are lines connecting two contact points which entirely lie in the two associated friction cones.

#### 4.8 Summary

We have proposed a method for solving the regrasp planning problem for a polygon with a large number of edges. A hand using in this chapter is assumed three free-flying fingers. Our method provides complete solutions represented by a graph which allows us to plan a regrasp sequence by using a graph search. The experimental results show the efficiency of our algorithm which merges grasp sets that are adjacent to one another into one connected set. The obtained connected sets are used to construct a switching graph in realtime.

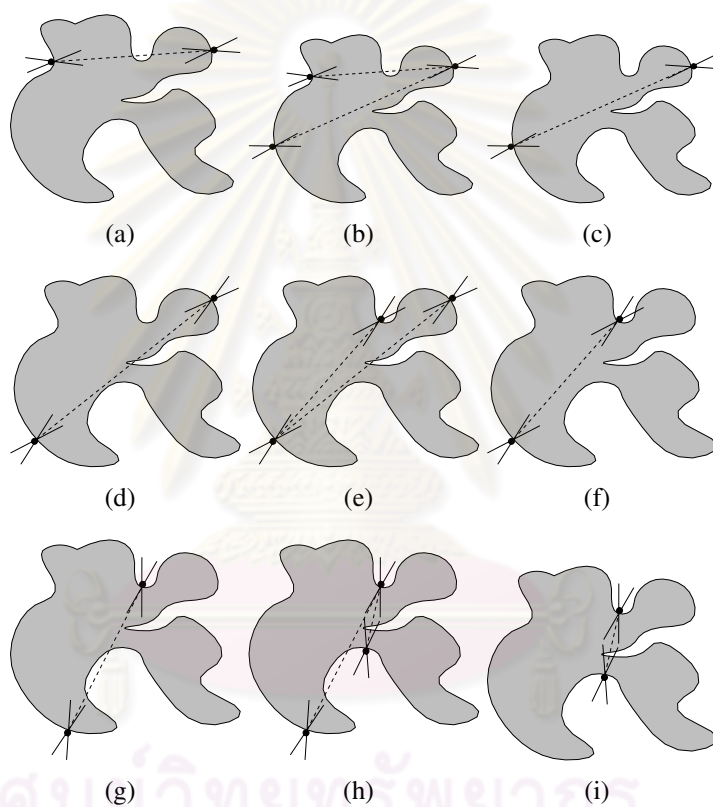


Figure 4.12: A regrasp sequence for the object in Fig. 3.15(c) when the half-angle is  $15^\circ$ .

# CHAPTER V

## REGRASP PLANNING FOR A POLYHEDRAL OBJECT

### 5.1 Introduction

This chapter addresses the regrasp planning problem of a 5-finger hand manipulating a polyhedron. We propose a technique for computing a sequence of finger repositioning that transforms an initial grasp into a desired one while keeping the manipulated object in a force-closure grasp during the entire process.

The proposed technique is based on the idea that a set of force-closure grasps can be represented geometrically as a compact set of points in 3D space. Based on this representation, overlapping volumes corresponding to different sets of grasps can be represented as a *switching graph*. The switching graph captures ability to switch from one set of grasps to another and, as a result, allows the regrasp planning to be thought of as a graph search problem. We demonstrate that the switching graph can be efficiently constructed for test objects with over 40 faces using a randomized technique. Note that although finger kinematics and other relevant constraints are not initially taken into account, different search strategies and policies may be later incorporated to generate regrasping sequences that meet additional requirements.

### 5.2 Force-closure conditions in 3D

For 3D grasp, we also exploit the condition of non-marginal equilibrium to imply force-closure for a grasp. A zero-pitch wrench  $w = (\mathbf{f}, \mathbf{t})$  for the force  $\mathbf{f}$  can be thought of as the line of action of this force and can be written in Plücker coordinates. Equilibrium therefore implies that the lines (represented as Plücker vectors) associated with the contact forces are linearly dependent. As mentioned in (Ponce et al., 1997), Grassman geometry (Dandurand, 1984), which characterizes the varieties of various dimensions formed by sets of dependent lines, can be applied to yield a necessary and sufficient condition for non-planar equilibrium, namely, the contact forces must *positively span*<sup>1</sup>  $\mathbb{R}^3$  and their

---

<sup>1</sup>A set of vectors positively spans some space when any vector in the space can be written as a linear combination of the vectors in the set with positive coefficients

lines of action all intersect in a point (concurrent grasps), lie in two flat pencils having a line in common (pencil grasps), or form a regulus (regulus grasps). Instead of using this condition directly for grasp computation, (Ponce et al., 1997) proposes a sufficient condition that does not depend on the actual contact forces. This condition provides an underlying idea for constructing the switching graph. It is given here as Proposition 5.2 which requires the following definition.

**Definition 5.1** *Let  $V_i, i = 1, 2, 3, 4$  be the four cones of half-angle  $\theta$  centered on vector  $v_i$ . We say that the four vectors  $v_i, i = 1, 2, 3, 4$   $\theta$ -positively span  $\mathbb{R}^3$  if any combination of vectors  $v'_i \in V_i, i = 1, 2, 3, 4$  positively span  $\mathbb{R}^3$ .*

To tell whether four given vectors  $\theta$ -positively span  $\mathbb{R}^3$ , we may verify that for any triple  $v_1, v_2, v_3$  of these vectors, the cones  $V_1, V_2, V_3$  of half-angle  $\theta$  centered on  $v_1, v_2$  and  $v_3$  lie in the interior of the same half-space and the cone  $-V_4$  of half-angle  $\theta$  centered on the direction opposite to the fourth vector  $v_4$  lies in the interior of the intersection of the trihedra formed by all triples of vectors belonging to  $V_1, V_2$  and  $V_3$ . Geometrically, it can be shown that the intersection of the trihedra is essentially the trihedron bounded by three planes, each of which passes through the origin and touches two of the three cones  $V_1, V_2, V_3$  while separating them into different half-space from the remaining cone.

In the following proposition and the remainder of the paper, we will denote by  $\theta$  the half angle of every friction cone.

**Proposition 5.2** *A sufficient condition for four non-coplanar points to form a force-closure grasp is that: (P1) there exist four lines in the corresponding double-sided friction cones that either intersect in a single point, form two flat pencils having a line in common but lying in different planes, or form a regulus, and (P2) the internal normals at the four contact points  $\theta$ -positively span  $\mathbb{R}^3$ .*

### 5.3 Switching Graph for a Polyhedral Object

The switching graph concept is based on the idea that a set of concurrent grasps can be represented by a point in 3D space. This representation will be explained in detail in Section 5.3.1. We will also show how contiguous points representing concurrent grasps can be grouped together to form a cell. A vertex of a switching graph represents a set of

grasps by establishing an association with a cell. The way we form a cell allows us to compute (1) a finger aligning between two grasps within the same cell and (2) a finger switching between a grasp in one cell and another grasp in another cell (associated with a neighboring vertex). This computation will be discussed in Section 5.3.4.

### 5.3.1 Representing Concurrent Grasps

As mentioned earlier, a grasp is geometrically defined by the positions of the fingers on the object's faces. Assuming polygonal object model, the position of a contact point can be defined by specifying an ordered pair representing coordinates of the point on the corresponding grasped face. With four grasping fingers, this amounts to using eight parameters to uniquely define a grasp (with the four grasped faces already chosen). However, using Proposition 5.3 from (Sudsang and Ponce, 1995), we can define a set of concurrent grasps with only three parameters. This proposition follows directly from Proposition 5.2.

**Proposition 5.3** *A sufficient condition for four fingers to form a force-closure grasp is that the four internal normals at the four contact points  $\theta$ -positively span  $\mathbb{R}^3$  and there exists a point  $\mathbf{x}_0$  such that the inverted friction cones at this point (Fig. 5.1) intersect the four contact faces.*

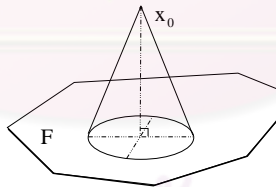


Figure 5.1: Inverted friction cone of face  $F$  at  $\mathbf{x}_0$

Note that each point  $\mathbf{x}_0$  satisfying Proposition 5.3 yields four *independent contact regions* where fingers can be placed independently while achieving concurrent grasp: these regions are simply the intersection of the inverted friction cones in  $\mathbf{x}_0$  with the contact faces. As we will discuss in Section 5.3.3, locally adjusting contact points within independent contact regions is a means for finger aligning to move from one grasping configuration to another one belonging to the same vertex in the switching graph.

We are now ready to discuss how a vertex in a switching graph represents a set of grasps. A vertex of a switching graph represents a set of concurrent grasps by having an association with a set of all points  $\mathbf{x}_0$  satisfying Proposition 5.3 for a given combination

of four faces. Since an inverted friction cone at  $\boldsymbol{x}_0$  intersect the corresponding face when  $\boldsymbol{x}_0$  lies in the volume defined by the union of all double-sided friction cones at every point on the face (Fig. 5.2(a)), the set of all  $\boldsymbol{x}_0$  satisfying Proposition 5.3 can be obtained from the intersection of the four volumes each of which is the union of all double-sided friction cones on each face. In the following definition, we name the union and the intersection for future references.

**Definition 5.4** *The union of all double-sided friction cones at every point on face  $F_a$  will be called the union volume for the face and will be denoted by  $U_a$ .*

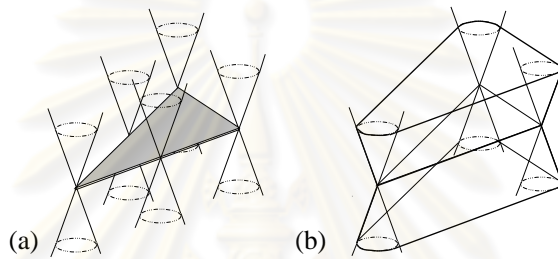


Figure 5.2: Union volume: (a) construction, and (b) its shape (see text)

**Definition 5.5** *The volume containing all points  $\boldsymbol{x}_0$  satisfying Proposition 5.3 for a given combination of four faces  $F_i, F_j, F_k$  and  $F_l$  where  $i \neq j \neq k \neq l$  will be called the focus cell for the faces and will be denoted by  $C_{i,j,k,l}$ .*

Before proceeding to the next section, it is helpful to discuss briefly about the shape of the union volume and the focus cell. Let us begin by considering an example of a triangular face with its union volume. As shown in Fig. 5.2(b), the union volume is composed of two symmetric parts (in mirror-like fashion): one above the face, and the other one below. Note that the union volume is an unbounded body. This is because double-sided friction cones are symmetric and unbounded. Clearly from the construction, the boundary of the union volume consists of unbounded rectangular and conic patches (at rounded corners). With conic parts involved, quadric surfaces are needed to exactly describe the union volume's boundary. This requirement implies that to construct a focus cell by intersecting four union volumes, univariate polynomial equations of degree upto 8 are to be solved (e.g., to obtain curved edges from intersecting two conic patches and to obtain a vertex from intersecting three conic patches). A typical technique to avoid this complexity is to give up some exactness by approximating conic parts of the boundaries of

union volumes with multi-facet pyramids. This approximation will allow a union volume to be described as a polyhedron and, in turn, a focus cell can readily be obtained using an algorithm for intersecting polyhedra (Hoffman, 1989). This approximation scheme should be used with caution because when the number of facets of the approximating pyramids is too large, the resulting polyhedron will have so many faces that intersecting polyhedra might be slower than using algorithms for computing intersection of quadric surfaces (Hoffman, 1989). This issue on construction of focus cells will become important as we discuss how to build a switching graph in Section 5.3.4. Before then, let us explain how focus cells are related to finger switching and finger aligning operations.

### 5.3.2 Finger Switching

Let us consider two focus cells  $C_{a,b,c,d}$  and  $C_{a,b,c,e}$  such that  $C_{a,b,c,d} \cap C_{a,b,c,e} \neq \emptyset$ . Let  $\mathbf{q}$  be a point in  $C_{a,b,c,d} \cap C_{a,b,c,e}$ . Clearly,  $\mathbf{q}$  defines two sets of concurrent grasps: one for the combination of faces  $F_a, F_b, F_c, F_d$  and the other for the combination of faces  $F_a, F_b, F_c, F_e$ . Let us suppose that the fingers 1,2,3 and 4 are respectively on faces  $F_a, F_b, F_c$  and  $F_d$  and forming one of the concurrent grasps defined by  $\mathbf{q}$ . It is easy to see that the hand can switch to another concurrent grasp (represented by  $\mathbf{q}$ ) on faces  $F_a, F_b, F_c$  and  $F_e$  by placing finger 5 on any point in the intersection between face  $F_e$  and its inverted friction cone at  $\mathbf{q}$  (because  $\mathbf{q} \in C_{a,b,c,d} \cap C_{a,b,c,e}$ ). Once finger 5 is on  $F_e$ , finger 4 can leave face  $F_d$  resulting in a switching from a concurrent grasp on  $F_a, F_b, F_c, F_d$  by fingers 1,2,3,4 to another concurrent grasp on  $F_a, F_b, F_c, F_e$  by fingers 1,2,3,5. This finger repositioning sequence enables us to plan finger switching by identifying intersection between two focus cells having one different grasped face.

### 5.3.3 Finger Aligning

Clearly, a finger switching cannot occur between two grasps whose corresponding focus cells do not overlap. For example, let us consider focus cells in Fig. 5.3. Obviously, because  $C_{a,b,c,d} \cap C_{a,b,c,f} = \emptyset$ , it is not possible to switch directly from a grasp on faces  $F_a, F_b, F_c, F_d$  to another grasp on faces  $F_a, F_b, F_c, F_f$  using a finger switching discussed in the previous section. However, suppose the current grasp on faces  $F_a, F_b, F_c, F_d$  is defined by  $\mathbf{q}_1$ , a finger switching can be performed to switch to another grasp on faces  $F_a, F_b, F_c, F_e$  ( $\mathbf{q}_1$  is in both  $C_{a,b,c,d}$  and  $C_{a,b,c,e}$ ) and somehow if the hand can adjust the fingers to change from the grasp defined by  $\mathbf{q}_1$  to a grasp defined by  $\mathbf{q}_2$  (which could be any point in  $C_{a,b,c,d} \cap C_{a,b,c,e}$ ), another finger switching at  $\mathbf{q}_2$  can be applied to switch to a grasp on faces  $F_a, F_b, F_c, F_f$  as desired.

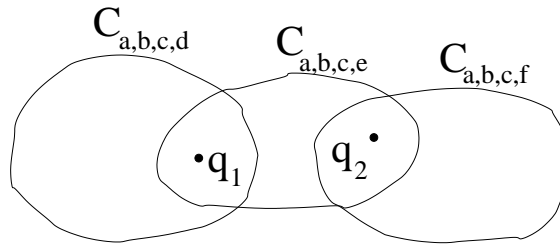


Figure 5.3: Moving between non-overlapping cells

In fact, changing grasping configuration within the same focus cell is the process we referred to as finger aligning. This process can be accomplished by taking advantage of the idea that force closure can be maintained during finger sliding, finger rolling (see (Han and Trinkle, 1998b; Bicchi and Marigo, 2000) on how to apply rolling in dexterous manipulation), or finger switching within an independent contact region. To illustrate, let us consider Fig. 5.4 showing configuration points  $q$  and  $q'$  in the same focus cell  $C_{a,b,c,d}$ . The inverted friction cones of the four grasped faces at  $q$  intersect the faces in the four independent contact regions  $R_a, R_b, R_c$  and  $R_d$  and likewise the inverted friction cones at  $q'$  intersect the four grasped faces in  $R'_a, R'_b, R'_c$  and  $R'_d$ . Suppose that the four fingers are at  $x_a \in R_a, x_b \in R_b, x_c \in R_c$  and  $x_d \in R_d$ . This can be represented by  $q$ . To move from  $q$  to  $q'$ , we move the four fingers from  $x_i$  to  $x'_i \in R_i \cap R'_i (i = a, b, c, d)$ . It is sufficient to ensure force closure during the fingers' motion by maintaining that the fingers are in the independent contact regions of  $q$  during the entire process. This can be done by rolling or sliding the fingers on the grasped faces from  $x_i$  to  $x'_i (i = a, b, c, d)$ . Instead of rolling or sliding, it is also possible to apply finger switching within each independent contact region by placing a free finger at  $x'_i$  and lifting off the finger at  $x_i$ . Because there is only one free finger during a concurrent grasp, this kind of finger switching can be performed in one independent region at a time.

By continuity, for any point in a focus cell, there exists a neighborhood for which the four independent contact regions of the point intersect the four independent contact regions of every point in the neighborhood. That is, there always exists a finger repositioning sequence to move between any pair of configuration points in the same focus cell.

#### 5.3.4 Computing a Switching Graph

To construct a switching graph, all of its vertices and edges need to be found. To identify all vertices of a switching graph, we compute all focus cells and to identify all



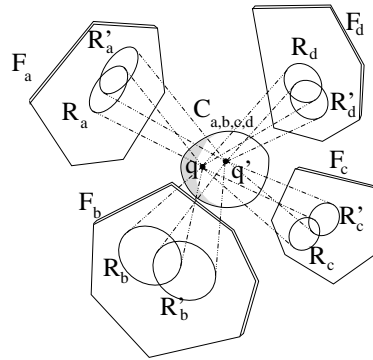


Figure 5.4: Moving within a focus cell

edges, we compute all pairs of overlapping focus cells with three common grasped faces.

Computing all focus cells requires identifying all combinations of four faces with concurrent grasps satisfying Proposition 5.3. Instead of enumeratively checking all combinations, the number of candidates can be significantly reduced by considering only those combinations whose internal normals positively span the plane. Our technique for generating such combinations is based on the fact that when three normals are given, the fourth one must lie strictly inside the trihedron formed by the inverses of the three given normals in order that the four normals positively span  $\mathbb{R}^3$  (otherwise, they would be in the same half space).

It is also important that every combination of four normals is listed without any repetition. This is essentially the problem of generating all  $k$ -subsets (i.e., subsets with  $k$  members) of a given set with  $n$  members. A simple solution for this problem is to assign a totally ordered relation to all members of the set and list every  $k$ -subset in the form of a  $k$ -tuple for which each element (except the last one) precedes the next one according to the assigned order. Applying this method to our problem, each unit normal is reparameterized using an ordered pair of two angles  $(\alpha, \beta)$  where  $\alpha \in [0, 2\pi]$  is the angle between the  $x$ -axis and the projection of the normal on the  $x$ - $y$  plane, and  $\beta \in [0, \pi]$  is the angle between the  $z$ -axis and the normal. With this parameterization, a sorted order can be imposed by defining that a normal  $\mathbf{n}_a = (\alpha_a, \beta_a)$  precedes a normal  $\mathbf{n}_b = (\alpha_b, \beta_b)$  when  $\alpha_a < \alpha_b$ , or when  $\beta_a < \beta_b$  in the case that  $\alpha_a = \alpha_b$ . For clarity, let us present pseudocode of the resulting algorithm. In the pseudocode, the  $n$  sorted unit normals are stored in the array  $normal[1..n]$  with corresponding indices to faces in the array  $faceId[1..n]$  and variable  $upwardIndex$  containing the index to the last normal in the array with angle  $\beta$  smaller than  $\pi/2$  (i.e., all normal vectors in  $normal[1..upwardIndex]$  points in the upward direction).

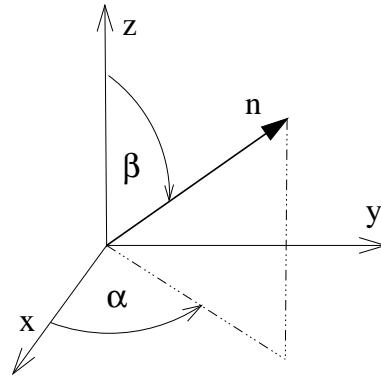


Figure 5.5: Parameterization of a unit normal vector

```

1: for  $i = 1$  to  $upperIndex$  do
2:    $n1 = normal[i]$ ;  $f1 = faceId[i]$ 
3:   for  $j = i + 1$  to  $n - 2$  do
4:      $n2 = normal[j]$ ;  $f2 = faceId[j]$ 
5:     for  $k = j + 1$  to  $n - 1$  do
6:        $n3 = normal[k]$ ;  $f3 = faceId[k]$ 
7:        $m = \max(k + 1, upperIndex + 1)$ 
8:       Compute all normal vectors in  $normal[m..n]$ 
       that is contained in the trihedron formed by
        $-n1, -n2$  and  $-n3$ 

```

From line 1 of the above pseudocode, we can see that every first normal is chosen such that it has to point upward (with  $\beta < \pi/2$ ). This is because choosing a first normal with angle  $\beta \geq \pi/2$  would result in having all four normals with  $\beta \geq \pi/2$  which means that they are all in the same lower half-space and therefore cannot positively span  $\mathbb{R}^3$ . The same reason is applied in line 7 to allow a fourth normal to point downward only (with  $\beta > \pi/2$ ), otherwise all four normals would be pointing upward and lie in the same upper half-space. Line 7 also incorporates the fact that, to generate different combinations without repetition, a fourth normal must be after the third normal according to the sorted order (i.e., with greater  $\beta$  than that of the third normal). The following paragraphs describe how line 8 can be implemented.

Because a unit normal can be thought of as a point on the unit sphere, and a trihedron formed by three unit vectors intersects the unit sphere in a triangular region (bounded by three sections of great circles), all normal vectors contained in the trihedron are therefore

those vectors corresponding to the points lying inside this triangular region. If we can somehow map the surface of the sphere onto the plane, a range searching algorithm can be applied to find the desired normals.

In fact, we have already mentioned such mapping. Recall that we parameterize every unit normal using an ordered pair of angles  $(\alpha, \beta)$ . This allows each normal vector to be mapped to a point in the  $\alpha$ - $\beta$  plane. The triangular region on the sphere mentioned above will be mapped to a planar region bounded by three vertices and three curved edges (Fig. 5.6). Since a curve of constant  $\alpha$  (resp.  $\beta$ ) on the sphere maps to a straight line parallel to the  $\beta$ -axis (resp.  $\alpha$ -axis) on the  $\alpha$ - $\beta$  plane, it is intuitive that the smallest isothetic box<sup>2</sup> covering the planar region can be drawn by considering only the range of the coordinates of the three vertices. With this bounding box, we can then apply an orthogonal range searching algorithm (de Berg et al., 1997) to find all the points contained in the box (note that before applying the range searching, the bounding box may need to be clipped to ensure that the angle  $\beta$  of a fourth normal is greater than that of the third normal). For each point obtained, its corresponding normal is checked with the three previously chosen normals to tell whether they can positively span  $\mathbb{R}^3$ . By using range trees (de Berg et al., 1997) to perform orthogonal range searching, the overall algorithm runs in  $O(n^3 \log^2 n)$ .

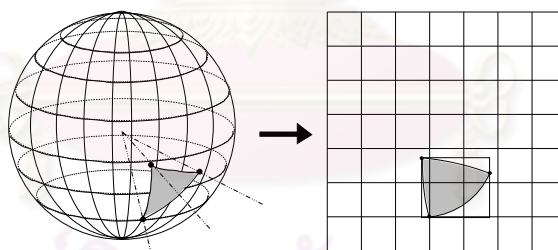


Figure 5.6: Mapping from the spherical to cartesian coordinates

In constructing the bounding box described above, it is important to take into account nature of the mapping from the spherical to the cartesian coordinates. In particular, when the triangular region on the sphere intersects the arc defined by  $\alpha = 0$  (Fig. 5.7), two bounding boxes are to be constructed to reflect that the arcs  $\alpha = 0$  and  $\alpha = 2\pi$  coincide.

Another case is when the triangular region covers the “south pole” (bottommost point) of the sphere. When this occurs, the normals corresponding to the three vertices of

<sup>2</sup>a rectangle with its sides parallel to the axes

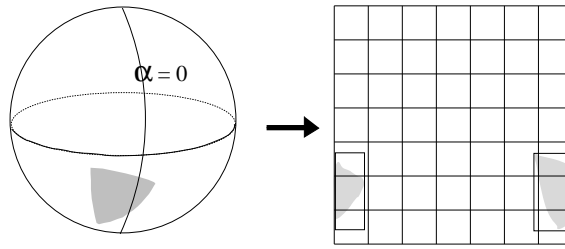


Figure 5.7: Two bounding boxes are needed when the triangular region cross over the arc  $\alpha = 0$

the triangular region have their normal projection on the  $x$ - $y$  plane positively spanning the plane. This should be handled by constructing a bounding box covering the entire range of  $\alpha$  (from 0 to  $2\pi$ ).

Every combination of faces found by the algorithm outlined above is also tested whether the corresponding four normal vectors  $\theta$ -positively span  $\mathbb{R}^3$ . This can be done in constant time by following geometric description given after Definition 5.1. Now that we know all combinations of faces whose normal vectors  $\theta$ -positively span  $\mathbb{R}^3$ , the next step is then to find which ones of these combinations yield focus cells, and which pairs of these focus cells overlap. In this paper, we investigate two different approaches for this task: direct geometric computation, and random sampling.

#### 5.3.4.1 Direct Geometric Computation

To test whether a combination of four faces  $F_a, F_b, F_c, F_d$  (with normals  $\theta$ -positively spanning  $\mathbb{R}^3$ ) forms a focus cell, intersection of the union volumes  $U_a, U_b, U_c, U_d$  is computed. The intersection, if not empty, is the resulting focus cell  $C_{a,b,c,d}$ . To find overlapping focus cells corresponding to an edge in the switching graph, all pairs of resulting focus cells with one different face are again checked for intersection.

#### 5.3.4.2 Random Sampling

The underlying idea is that all the points contained in a focus cell are contained in all the union volumes of the faces that form the cell. This implies that if we have found such points, we have an evidence showing that the corresponding focus cell exists. Likewise, we can conclude that two focus cells overlap if we can find some points that are contained in both cells. Following this simple idea, instead of directly computing intersection to explicitly obtain focus cells, a number of points in 3D are randomly selected, each of which is then tested to list all faces whose union volumes can contain the point.

The resulting list of faces is then scanned for matching with combinations of four faces whose normal vectors  $\theta$ -positively span  $\mathbb{R}^3$  (obtained from the algorithm previously described). A matching indicates a focus cell found, and any pair of matching with the two corresponding combinations having one different face indicates that the corresponding focus cells overlap and an edge in the switching graph linking the two cells exists.

It is clear that the completeness of the switching graph generated using this approach depends heavily on the number of sampled points and the region in  $\mathbb{R}^3$  where the sampling takes place. To define the sampling region that is guaranteed to cover all focus cells without actually computing them is still an open problem. Our implementation shown in the next section relies on an *ad hoc* alternative by defining the sampling region to be the cube obtained from enlarging the smallest isothetic cube that can contain the object four times about its center. Although a complete switching graph cannot be guaranteed, experimental results show large number of vertices and edges are found within a fraction of the time used by the direct geometric approach.

#### 5.4 Implementation and Results

We have implemented the regrasp planning based on the switching graph concept described in the paper. The program is written in C++ using ACIS library (Corney and Lim, 2002) for geometric computation. All run times are measured on a PC with a 2.4 GHz CPU.

Some test polyhedra are shown in Fig. 5.8 and 5.9. Test results in Table 5.1 show the number of focus cells found, the number of links found, the number of connected component of switching graphs and the run time for each object in Fig. 5.8 when using the direct intersection approach to build the switching graph. Test results from random sampling approach with 1,000, 5,000, 10,000, and 20,000 sampling points are shown in Tables 5.2-5.5 correspondingly (these are numbers of one run for each test object to show tendency of the random approach). Without guaranteeing a complete switching graph, the random sampling approach appears to generate a large portion of the graph when spending only small amount of time compared with the direct intersection approach. In particular, for most objects, the random sampling approach is much faster and also producing the nearly complete switching graph. It is of course difficult to give a general statement from only few examples, however we feel that the random sampling approach is very promising especially in its ability to quickly produce a sketch of the switching graph. Fig. 5.10, 5.11 and 5.12 show snapshots of a short sequence of finger repositioning generated from the

program to transform the initial grasps into the target grasps. With a switching graph already computed, the program takes less than 0.1 second to generate the sequence.

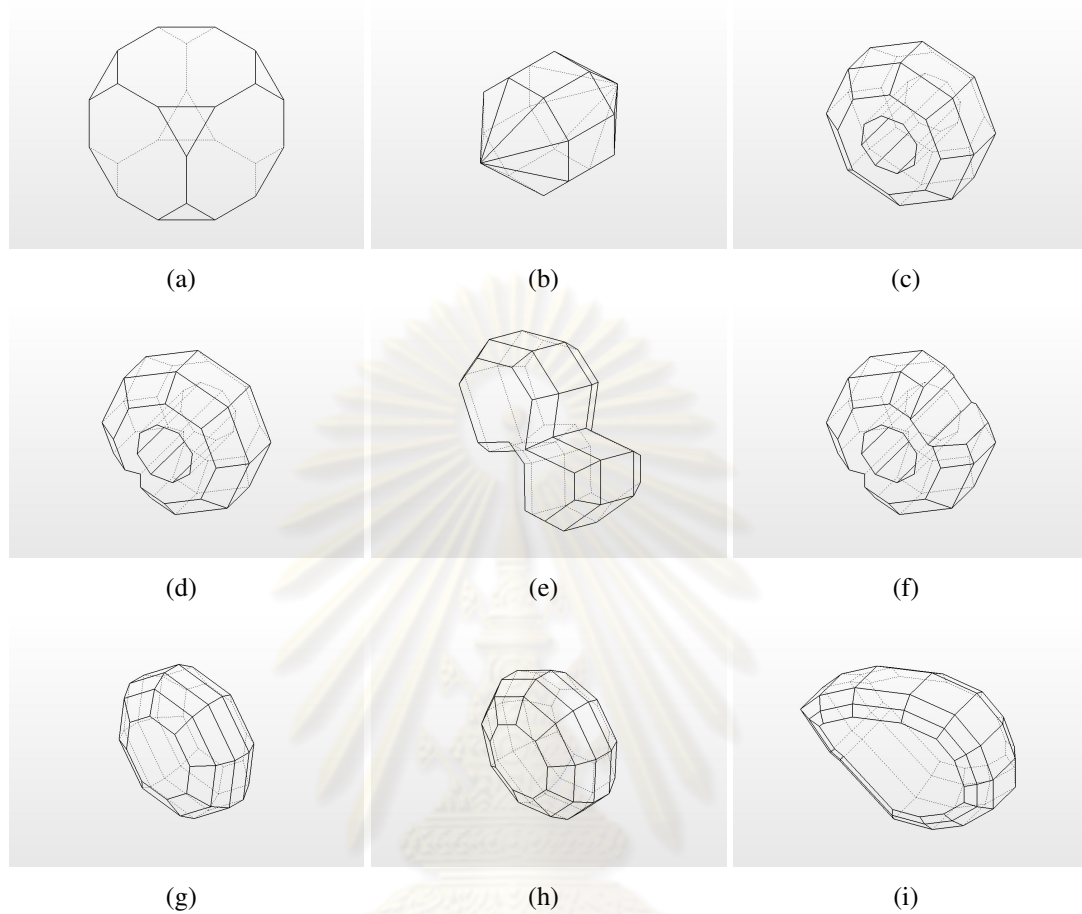


Figure 5.8: Test objects with the number of faces = (a) 14, (b) 24, (c) 34, (d) 36, (e) 38, (f) 40, (g) 42, (h) 47 and (i) 67

## 5.5 Summary

We have presented a method for regrasp planning of a polyhedron by a 5-finger hand based on the concept of the switching graph. A set of force-closure grasps is represented geometrically as a compact set of points in 3D space which allows us to solve the regrasp planning problem by computational geometry algorithms in 3D. Based on this representation, overlapping volumes corresponding to different sets of grasps can be computed by finding intersections between polyhedrons representing the grasp volumes. The experimental results demonstrate an efficient implementation of the proposed approach. The direct computation provides a complete switching graph while the randomized approach is more efficient in the aspect of computational time when an input object consists of a large number of faces.

Table 5.1: Results from direct intersection approach for each test object in Fig. 5.8

Fig.	# focus cells	# links	# cc	time (s)
5.8(a)	22	24	3	1.61
5.8(b)	177	384	1	19.03
5.8(c)	503	1408	1	49.99
5.8(d)	585	1664	5	60.97
5.8(e)	509	1451	12	53.59
5.8(f)	527	1430	8	46.99
5.8(g)	830	2434	17	52.89
5.8(h)	2319	13331	20	461.42
5.8(i)	621	2498	3	136.92

Table 5.2: Results from random sampling approach for each test object in Fig. 5.8 with 1,000 sampling points

Fig.	# focus cells	# links	# cc	time (s)
5.8(a)	14	14	2	0.06
5.8(b)	111	217	1	0.2
5.8(c)	268	520	5	0.63
5.8(d)	226	462	10	0.79
5.8(e)	99	130	15	0.94
5.8(f)	92	157	5	1.03
5.8(g)	137	286	1	1.38
5.8(h)	716	3237	8	2.16
5.8(i)	50	129	4	8.56

Table 5.3: Results from random sampling approach for each test object in Fig. 5.8 with 5,000 sampling points

Fig.	# focus cells	# links	# cc	time (s)
5.8(a)	22	24	3	0.36
5.8(b)	177	384	1	0.56
5.8(c)	311	612	5	1.22
5.8(d)	338	688	4	1.45
5.8(e)	158	250	11	1.56
5.8(f)	413	917	12	1.69
5.8(g)	235	520	11	2.03
5.8(h)	1150	5013	2	2.98
5.8(i)	236	817	3	9.58

Table 5.4: Results from random sampling approach for each test object in Fig. 5.8 with 10,000 sampling points

Fig.	# focus cells	# links	# cc	time (s)
5.8(a)	22	24	3	0.67
5.8(b)	177	384	1	1.06
5.8(c)	495	1307	1	2.3
5.8(d)	493	1146	9	2.39
5.8(e)	196	303	14	2.55
5.8(f)	429	948	15	2.52
5.8(g)	482	1166	16	3.25
5.8(h)	1826	9276	4	4.97
5.8(i)	360	1436	4	11.28

Table 5.5: Results from random sampling approach for each test object in Fig. 5.8 with 20,000 sampling points

Fig.	# focus cells	# links	# cc	time (s)
5.8(a)	22	24	3	1.28
5.8(b)	177	384	1	1.95
5.8(c)	503	1359	1	3.9
5.8(d)	548	1378	1	4.17
5.8(e)	274	466	15	4.16
5.8(f)	505	1225	10	4.14
5.8(g)	545	1333	17	5.11
5.8(h)	2065	11279	11	8.66
5.8(i)	370	1467	3	14.16



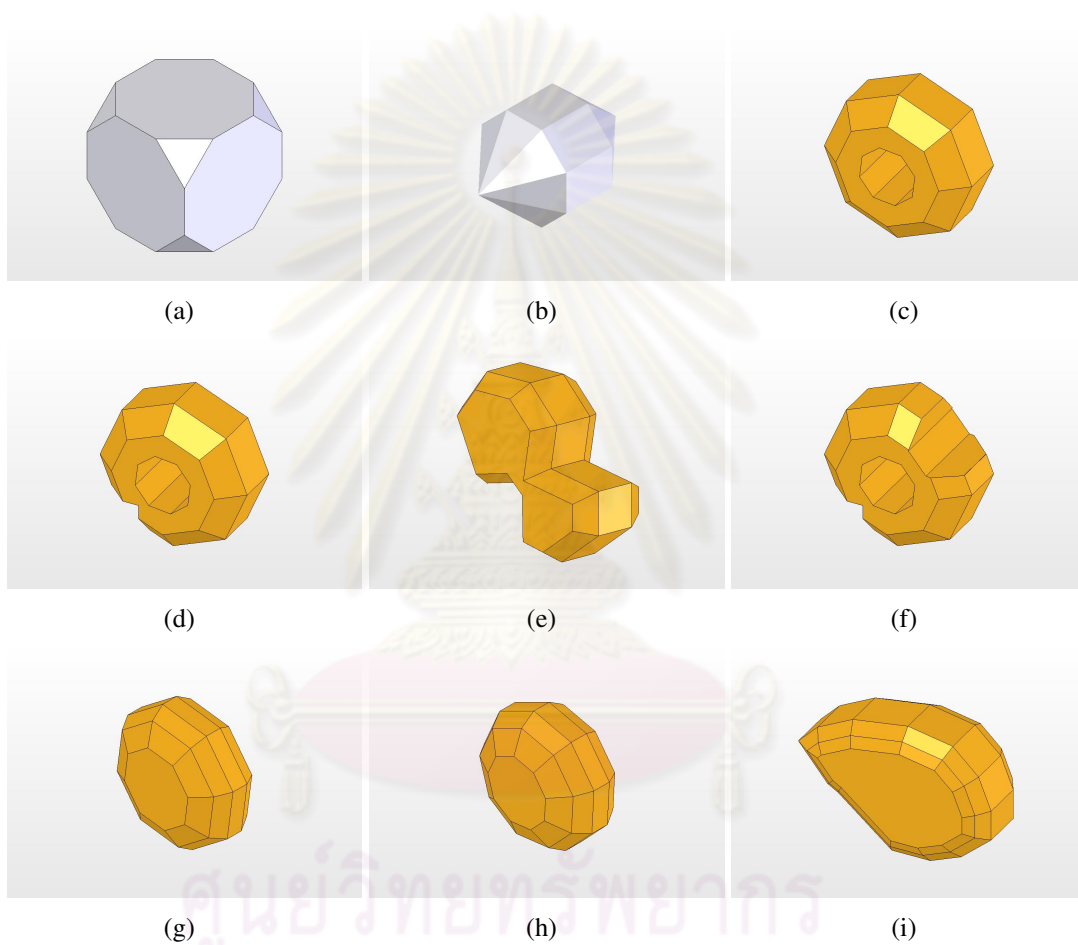


Figure 5.9: Shaded test objects with the number of faces = (a) 14, (b) 24, (c) 34, (d) 36, (e) 38, (f) 40, (g) 42, (h) 47 and (i) 67

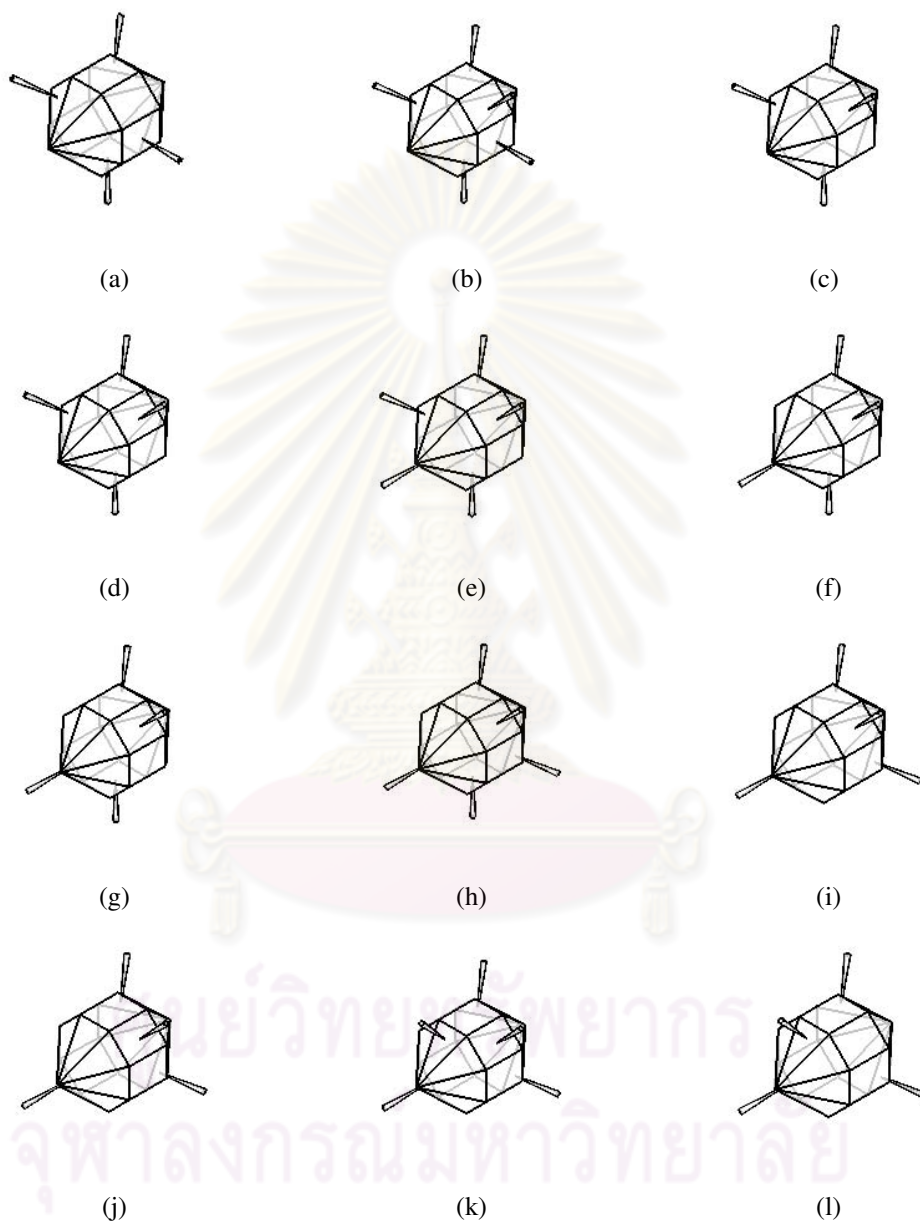


Figure 5.10: A regrasp sequence generated from a switching graph of the object in Fig. 5.8(b)

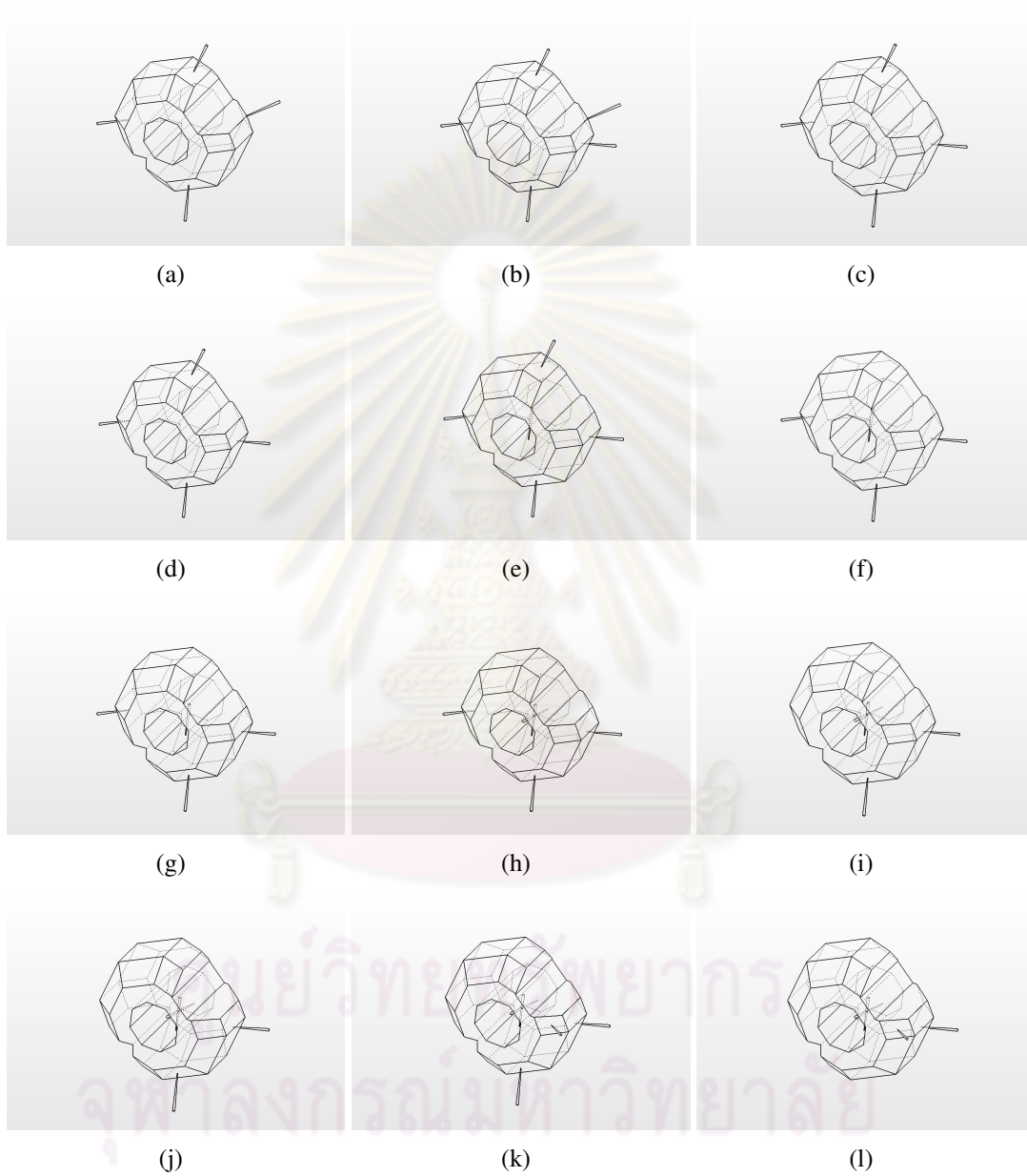


Figure 5.11: A regrasp sequence generated from a switching graph of the object in Fig. 5.8(f)

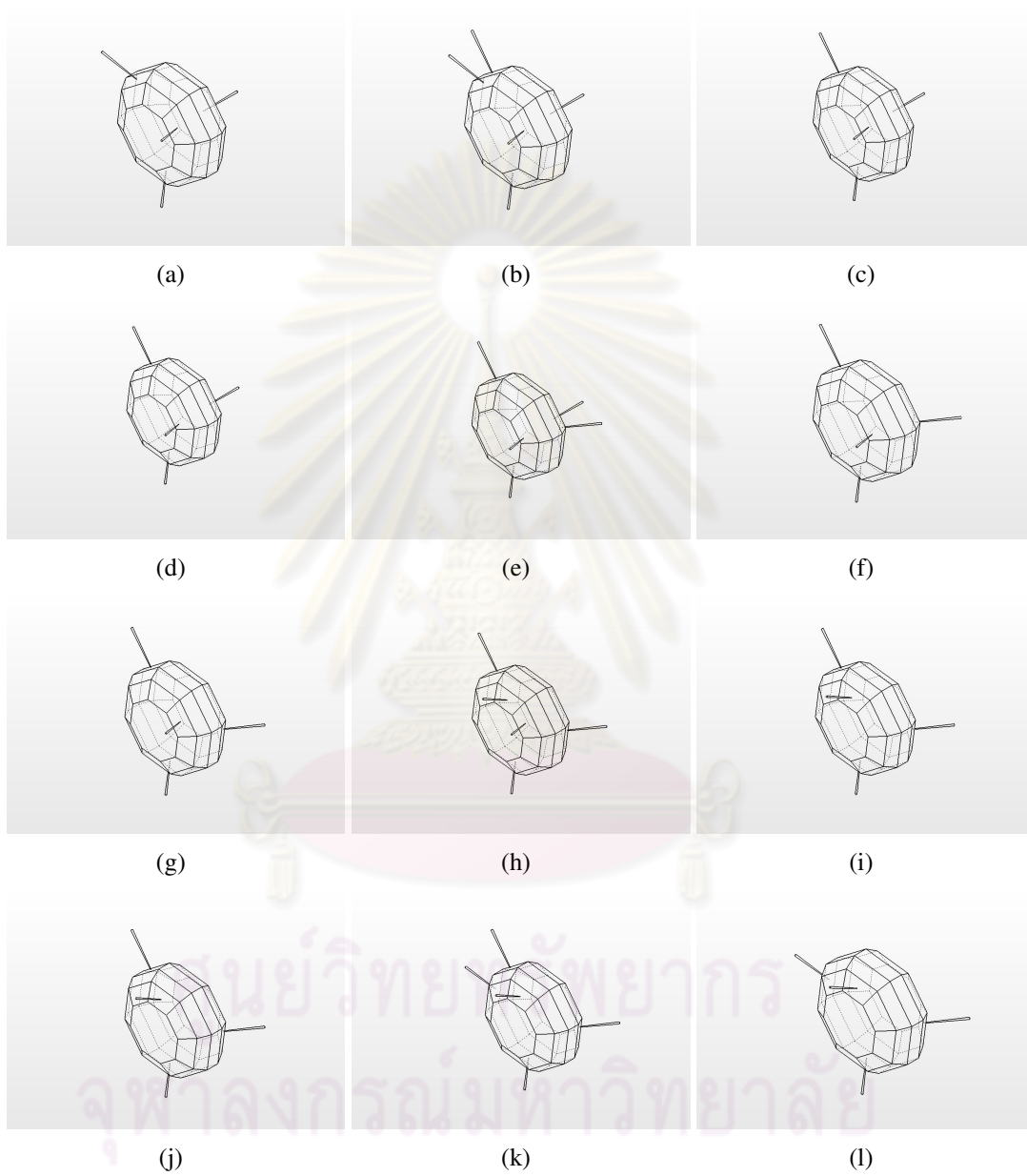


Figure 5.12: A regrasp sequence generated from a switching graph of the object in Fig. 5.8(g)

# CHAPTER VI

## REGRASP PLANNING FOR A TRIANGULAR-MESH OBJECT

### 6.1 Introduction

As we have described in the previous chapters, regrasp planning can be formulated as a graph search problem where vertices of the graph represent force closure grasps while an edge connects two grasps that can be changed between each other by simple movement. To construct such graph, it is required to compute all force closure grasps and calculate all possible simple movements. This approach works well when the object is of low complexity, e.g. polyhedron with small number of facets. However, this is not the case for most real world objects, especially when the information of the object is acquired by sensor rather than being nicely modeled by human.

To achieve complete automation, it is vital for the robot to be able to perform sensing by itself. Acquisition of object spatial structure is usually done via range sensing devices such as a laser range finder or stereoscopic cameras. The data obtained through this method usually involve thousands of surface points. This poses a very challenging issue in grasp planning because of the combinatorial explosion of the search space. Given a thousand contact points, the number of all possible 4-finger grasps reach  $10^{12}$ , beyond the address space of an ordinary computer. Considering all possible grasps would take too much time. To provide a trade-off between accuracy and computational simplicity, Goldfeder *et al.* (2007) applied superquadric fitting to parameterize object shapes from the point-cloud input. A grasp planning is performed in the hierarchy of superquadrics from the coarsest to finer approximations. In (Huebner et al., 2008), the minimum volume bounding box approach is used to fit input data by primitive box shapes. The result bounding boxes and data points are iteratively split to yield better box approximations. A grasp planner exploits the approximations as clues to synthesize grasps on arbitrary objects.

While the grasp planning problem for discrete contact points just recently began to receive more attention, the regrasp planning problem in the same setting remains mostly

unexplored. Most existing works in grasp planning for point data approximate the model of the object using only spatial information from the input points. Although this approach may be sufficient for computing a single grasp, it offers no clues on how a regrasp sequence can be obtained. The main contribution of this paper is to present a novel method for organizing the input contact points to facilitate the regrasp planning process. Our underlying idea is to partition the input points based on their wrenches. Contact points that can exert similar wrenches are grouped together. For each group, a representative is selected. Every set of representatives that can form a force closure grasp is then computed and referred to as a representative grasp. Our idea is motivated by a typical regrasp scenario when a supporting finger need to be placed on the object before some finger in a force closure grasp can be lifted off. Obviously, if the supporting finger can exert wrenches similar to those by the finger to be lifted off, it is likely that the object will still remain in force closure after the finger swap. The most important product of the proposed clustering strategy is the roadmap structure that leads to significant reduction of the search space. This structure is a graph that captures how representative grasps can switch among one another via finger swapping. Since each representative grasp roughly describes a different way wrenches can be aligned to form a force closure grasp, the roadmap somewhat approximates the global relationship that describes how force closure grasps can be switched among one another. Of course, an arbitrary force closure grasp may not be a member of the roadmap. Therefore, to utilize the roadmap for regrasp planning from an arbitrary initial grasp to a target grasp, we need regrasp sequences that bring the initial and the target grasps to some grasps in the roadmap. We will present methods for computing such sequences based on the clustering information. Preliminary experimental results show that most regrasp planning problems can be solved within a few seconds or a few minutes by our approach whereas exhaustive search takes about a day or longer.

## 6.2 Regrasp Planning on Discrete Point Set

In this chapter, we assume that the model of the input object is described by triangular meshes. This will result in a polyhedron with a large number of triangular facets. Each facet is usually very small relative to the entire object. We use the centroid of each facet as a possible contact point. This approach is also adopted in (Roa and Suarez, 2008). Since we consider discrete contact points, finger rolling and finger sliding which require continuous motion at a contact are not permitted. Only finger switching is considered in the planning.

### 6.2.1 Overview

The first step of regrasp planning is to compute a roadmap of all possible grasping configurations. Since we consider 4-finger grasps, a grasping configuration consists of four contact points. A grasping configuration that satisfies force closure induces a vertex in the roadmap and the configuration is stored in this vertex. An edge joining two vertices exists when two associated grasping configurations can switch to each other. This work assumes five fingers for regrasping. Four fingers are used to securely grasp the object. The remaining finger is used for finger switching. This means that two grasping configurations that have one distinct contact point can perform finger switching which implies that there exists an edge joining the two vertices corresponding to these two grasping configurations.

There are some drawbacks associated with this traditional approach. Suppose there are  $N$  contact points to consider. The possible grasping configurations are as many as  $N^4$ . For each grasping configuration, there can be as many as  $4N$  other grasping configurations that can be reached directly by finger switching. Consequently, size of a roadmap constructed from a large number of contact points could easily become larger than the memory space of an ordinary computer.

To overcome these limitations, we propose to cluster the input contact points into groups. Instead of using all contact points, one contact point is picked from each group to be a representative for constructing the representative-level roadmap. The number of groups is a user-defined variable which indicates trade off between completeness and resource used in the computation. Of course, the representative-level roadmap does not cover grasping configurations consisting of some contact points that are not representatives. A local planner is required to compute a path from such grasping configurations to a grasp in the representative-level roadmap.

### 6.2.2 Spectral Clustering for Contact Point Set

Before constructing a representative-level roadmap, we apply spectral clustering algorithm to partition the input contact points into meaningful clusters. In spectral clustering, users are free to define how a cluster is meaningful according to their task at hand. However, the definition of meaningful clusters from existing works, mostly in the field of computer graphics, are not related to our regrasp planning problem. Our proposed idea is to define meaningful clusters from similarity of wrenches by means of grasping. Recall the finger switching operation: the remaining finger is placed on the object then one

finger is lifted to change grasping configuration. To maintain force closure, a reasonable heuristic is to ensure that the chosen contact point for the remaining finger and the contact point of the finger to be lifted can produce similar wrenches.

Spectral clustering takes a contact point set as input to compute an affinity matrix which embeds similarities of every pair of contact points. The matrix is solved for eigenvectors corresponding to the  $k$  largest eigenvalues. The eigenvectors are then used to determine the clustering of contact points.

### 6.2.2.1 Affinity Matrix

Affinity matrix contains information that reflects how contact points are grouped according to their applicable forces and torques. Each pair of contact points is measured for pairwise distance. The pairwise distances of all pairs form a matrix that encodes similarity between contact points. An affinity matrix is symmetric and denoted by  $A \in \mathbb{R}^{N \times N}$ , where  $0 \leq a_{ij} \leq 1$  for all contact points  $i$  and  $j$ . Element  $a_{ij}$  encodes the likelihood that contact points  $i$  and  $j$  can be clustered into the same group. Let  $\mathbf{f}_i, \mathbf{r}_i$  be a unit force perpendicular to the object's surface and the position of  $i$ th contact point w.r.t. a reference point  $\mathbf{o}$ . The associated torque is  $\mathbf{t}_i = (\mathbf{r}_i - \mathbf{o}) \times \mathbf{f}_i$ . The wrench generated by  $\mathbf{f}_i$  at this contact point  $i$  is therefore  $\mathbf{w}_i = (\mathbf{f}_i, \mathbf{t}_i)$ . When friction is assumed, the friction cone at contact point  $i$  is approximated using an  $m$ -sided pyramid bounded by  $\mathbf{f}_{i1}, \dots, \mathbf{f}_{im}$ . The associated wrench cone is defined by  $\mathbf{w}_{i1}, \dots, \mathbf{w}_{im}$  and called primitive wrenches.

Since a force closure test in the wrench space considers only the directions of wrenches, the distance function is formulated based on measuring the angle between two wrenches from two distinct contact points, which can be calculated from their inner product. However, the torque component of a wrench depends on the choice of the origin assumed in the torque calculation. We therefore use the centroid of the object as the reference origin. Since any vector  $(\mathbf{r}_i - \mathbf{o})$  is a constant vector, the torque component of the associated wrench is not affected by any rigid transformation applied to the object, i.e., independent from the object's pose.

The proposed pairwise distance between two contact points  $i$  and  $j$  considers the difference between wrenches that the two contact points can exert. Roughly speaking, we compare the geometries of the two wrench cones. Instead of integrating all differences between all pairs of wrenches from the two cones, an approximation is taken by



comparing only the boundaries of the linearized wrench cones, i.e., angles between the primitive wrenches from the two wrench cones are measured. Each primitive wrench of  $i$  is compared with a primitive wrench of  $j$ . Obviously, there are many ways to match pairs of primitive wrenches for comparison. We have to select one correspondence that is appropriate for the distance function. Since the linearization of a friction cone is in either a clockwise or counterclockwise order, the result primitive wrenches are arranged in the same order and a reasonable correspondence of the primitive wrenches has to preserve this order. The starting index for a wrench cone in the comparison is however not necessary the first index obtained from the linearization. The indices of the primitive wrenches of a contact point can all be shifted by an integer  $x$  while preserving the order. Without loss of generality, we apply the shifting  $x$  for the primitive wrenches of the contact point  $j$ . The angles between  $\mathbf{w}_{i_1}, \dots, \mathbf{w}_{i_m}$  and  $\mathbf{w}_{j(1+x \bmod m)}, \dots, \mathbf{w}_{j(m+x \bmod m)}$  are measured pair by pair in order (Fig. 6.1). The summation of these angles defines our geometrical difference between these two wrench cones. However, this value is varied by changing  $x$ . We imitate the principal of the shortest distance between two bodies in the Euclidean space: by varying  $x$ , the minimal summation of angles is used as the pairwise distance.

The difference between two wrench cones is measured as follows

$$M(i, j) = \min \sum_{k=1}^m \text{angle}(\mathbf{w}_{ik}, \mathbf{w}_{j(k+x)})$$

where  $0 \leq x \leq m - 1$  and  $k + x \in \mathbf{Z}_m$ . Since value of  $M(i, j)$  depends on the number of pyramid's sides, the distance function is normalized as  $d(i, j) = M(i, j)/m$ .

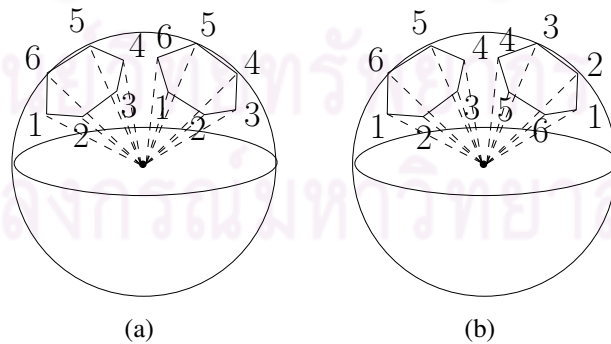


Figure 6.1: Transformation distance: An example of cones in 3D where the numbers show the order of comparison (a)  $x = 0$  (b)  $x = 2$

The Gaussian similarity function is applied to encode pairwise measures into the affinity matrix. It is formed by an exponential function as

$$a(i, j) = e^{-d(i,j)/2\sigma^2}.$$

Clearly,  $0 \leq a_{ij} \leq 1$ , and contact points of which their pairwise distance is smaller have larger affinities between them. The issue of choosing  $\sigma$  is neglected. We simply choose  $\sigma$  as the average of all measures, i.e.,  $\sigma = \frac{1}{N^2} \sum_{1 \leq i, j \leq N} d(i, j)$ .

### 6.2.2.2 Spectral Clustering Algorithm

The spectral clustering algorithm in (Ng et al., 2001) is applied. The eigenvectors of the affinity matrix are used in clustering of contact points. The spectral clustering algorithm from (Ng et al., 2001) is as follows.

- i. Compute the affinity matrix  $A$ .
- ii. Define  $D$  to be the diagonal matrix whose  $(i, j)$ -element is the sum of  $A$ 's  $i$ -th row, and construct the matrix  $L = D^{-1/2}AD^{-1/2}$
- iii. Compute the eigenvectors  $\mathbf{v}_1, \dots, \mathbf{v}_k$  of  $L$  associated with the  $k$  largest eigenvalues.
- iv. Construct the matrix  $V = [\mathbf{v}_1 \mathbf{v}_2 \dots \mathbf{v}_k] \in \mathbb{R}^{N \times k}$  by stacking the eigenvectors in columns.
- v. Form the matrix  $U$  from  $V$  by normalizing the row sums to have unit length, that is  $u_{ij} = v_{ij} / (\sum_k v_{ij}^2)^{1/2}$
- vi. Extracting each row of  $U$  as a point in  $\mathbb{R}^k$ , cluster them into clusters  $K_1, \dots, K_k$  by performing  $k$ -means algorithm
- vii. Assign the original contact point  $p_i$  to cluster  $K_j$  if and only if row  $i$  of the matrix  $U$  is assigned to cluster  $K_j$ .

After running the clustering algorithm, we need to construct some structures for further uses in regrasp planning. Let the input contact points be stored in a table denoted by  $T_O$ . From the above clustering algorithm, Euclidean distances in the affinity matrix are used to partition the input contact points. We then apply the same distance measurement for the successor procedures instead of the pairwise distance measurement proposed in Section 6.2.2.1. The contact point associated with the row vector of  $U$  that is closest to the center of  $K_j$  is chosen to be the representative of the cluster  $K_j$ . We construct a

matrix  $E$  to store Euclidean distances between all row vectors of  $U$  and the centers of all clusters. A matrix  $F$  contains Euclidean distances among all row vectors. A table  $T_R$  is constructed to store all representatives.

### 6.2.3 Constructing Representative-Level Roadmap

We are now ready to compute a roadmap for the representatives. All vertices are constructed by checking force closure grasp for every four representatives in  $T_R$ . We call such grasping configurations *representative grasping configurations*. Each representative grasping configuration satisfying force closure is assigned to a vertex. The number of vertices in a switching graph is equal to the number of grasps found in force closure checking. In this chapter, we apply the algorithm of (Niparnan and Sudsang, 2007) for fast filtering grasps that do not satisfy the necessary condition. Grasps that pass the filter are then verified for force closure by applying the algorithm of (Zhu and Wang, 2003a). An edge joining two vertices of which corresponding representative grasping configurations can switch to each other can be easily computed by checking the number of common contact points between the two corresponding configurations. Since 4-finger force closure grasps are considered, two representative grasping configurations can switch to each other when they share three common contact points. The computed representative-level roadmap is denoted by  $R$ .

### 6.2.4 Planning Regrasp Sequence

The regrasp sequence planning process is splitted into two level search: in representative-level roadmap and in local roadmap. In the previous section, the construction of representative-level roadmap has been described. A regrasp sequence acquired from the representative-level roadmap contains only configurations consisting of contact points in  $T_R$  which is a subset of the original contact point set  $T_O$ . This means that traversal in the representative-level roadmap does not cover grasping configurations that consist of some contact points in  $T_O \setminus T_R$ . Therefore, for arbitrary initial and goal grasps, the regrasp planner has to find regrasp sequences that link both grasps to some grasps in the representative-level roadmap.

An intuitive way is to find a path from the initial grasp or the goal grasp to their representative grasp, i.e., the grasp that consists of the representatives of all contact points forming the initial grasp or the goal grasp. The planner performs Algorithm 1 which exploits a heuristic of similarity in a cluster to determine a vertex in the representative-level

roadmap that the initial grasp or the goal grasp should be switched to. Let  $g$  be the initial grasping configuration or the goal grasping configuration that consists of contact points  $p_a, p_b, p_c$  and  $p_d$ . Let us denote by  $p_{a'}, p_{b'}, p_{c'}$  and  $p_{d'}$  the representatives of the clusters that respectively contain  $p_a, p_b, p_c$  and  $p_d$ . Also denote by  $g'$  the representative grasping configuration consisting of  $p_{a'}, p_{b'}, p_{c'}$  and  $p_{d'}$ . If  $g'$  does not achieve force closure, i.e. the vertex defining  $g'$  is not in the representative-level roadmap  $R$ , the planner then performs Algorithm 2. Otherwise, a regrasp sequence from  $g$  to  $g'$  is planned as follows.

For each contact point  $p_{i'}$  of  $g'$  ( $i = a, \dots, d$ ), all contact points in  $T_O$  that are in the same cluster of  $p_{i'}$  are considered. We exploit nearness among  $p_i, p_{i'}$  and contact points in the cluster to limit search space in local planning. The distance  $F_{ii'}$  between  $p_i$  and  $p_{i'}$  is queried from  $F$ . A contact point  $j$  in the cluster that induces  $F_{ij} \leq F_{ii'}$  and  $F_{i'j} \leq F_{ii'}$  is copied to a set  $S_i$ . We then compute all possible 4-finger force closure grasps such that the first, second, third and fourth contact points are picked from  $S_{a'}, S_{b'}, S_{c'}$  and  $S_{d'}$ , respectively. A local roadmap is then constructed such that each of its vertices represents each force closure grasp mentioned above, and each of its edges joins two vertices representing two grasping configurations with three common contact points (finger switching is possible). With a local roadmap, any graph search can be used to retrieve a path from the vertex representing  $g'$  to the vertex representing  $g$ . If no such path can be found, the planner invokes Algorithm 2.

---

### Algorithm 1

---

```

1: Determine  $p_{a'}, p_{b'}, p_{c'}, p_{d'}$  and  $g'$ 
2: if  $g' \notin R$  then
3:   Algorithm 2
4: else
5:   Determine  $S_{a'}, S_{b'}, S_{c'}, S_{d'}$ 
6:    $L = \text{ConstructRoadmap}(S_{a'}, S_{b'}, S_{c'}, S_{d'})$ 
7:   if path = FindPath( $L, g, g'$ ) then
8:     return path
9:   else
10:    Algorithm 2
11:   end if
12: end if

```

---

Algorithm 2 again exploits the information of clusters. It is invoked when  $g'$  does not achieve force closure or Algorithm 1 fails to find a regrasp sequence from  $g$  to  $g'$ . The underlying idea of this algorithm is to find another appropriate representative grasping configuration, which the given grasping configuration will change to, that satisfies force closure and has one different contact point from  $g'$ . Since the new representative grasping

configuration does not consist of all representatives of the given grasping configuration, to apply the same local planning strategy, we have to relocate one contact point of the given grasping configuration from its cluster to the cluster of the different representative. A local planning, is then performed among one changed cluster and three unchanged clusters, which guarantees that  $g$  and the new representative grasping configuration are force closure. However, this method perturbs the local properties of the changed clusters. To minimize the effect of this transfer and to maximize the use of local property, this algorithm aims to transfer one contact point in  $g$  to its second nearest cluster. The procedure *FindNearestCluster* begins with finding vertices in  $R$  of which the representative grasping configuration  $f'$  has one distinct representative from  $g'$ . Let  $p_{x'}$  and  $p_{y'}$  be the distinct representative of  $g'$  and  $f'$ . The associated contact point of  $p_{x'}$  is denoted by  $p_x$ . We query the matrix  $E$  for the distance between  $p_x$  and the center of cluster  $K_{y'}$  of  $p_{y'}$ . All representative grasping configurations in  $R$  having one distinct representative from  $g'$  are used to query the distances. The pair that induces the shortest distance between them is selected for locality reason. Now we redefine some notations to understand the pseudocode. Let the selected contact point be  $p_x$ , its cluster be  $K_{x'}$ , its second nearest cluster be  $K_{y'}$  and the representative grasping configuration possessing the cluster  $K_{y'}$  be  $f'$ . This procedure returns the variable  $X$  which consists of  $p_x$ ,  $K_{x'}$ ,  $K_{y'}$  and  $f'$ . If  $X$  is determined, we then temporarily add  $p_x$  into  $K_{y'}$ . Let  $f'$  consist of  $p_q, p_r, p_s, p_t$ . We perform a local roadmap construction as applied in Algorithm 1 and find a path between  $g$  and  $f'$  which both are members of the local roadmap.

---

**Algorithm 2**


---

```

1: Determine  $p_{a'}, p_{b'}, p_{c'}, p_{d'}$ 
2:  $X = \text{FindNearestCluster}(p_{a'}, p_{b'}, p_{c'}, p_{d'})$ 
3: if  $X$  is not determined then
4:   Algorithm 3
5: else
6:   Add  $p_x$  into  $K_{y'}$ 
7:   Determine  $S_q, S_r, S_s, S_t$ 
8:    $L = \text{ConstructRoadmap}(S_q, S_r, S_s, S_t)$ 
9:   if path =  $\text{FindPath}(L, g, f')$  then
10:    Remove  $p_x$  from  $K_{y'}$ 
11:    return path
12:  else
13:    Remove  $p_x$  from  $K_{y'}$ 
14:    Algorithm 3
15:  end if
16: end if

```

---

If finding a path from  $g$  to  $f'$  still does not achieve, Algorithm 3 is applied by updat-

ing the representative-level roadmap. The contact points  $p_a, p_b, p_c, p_d$  are now considered as representatives. Let  $T_G$  be a set of the contact points  $\{p_a, p_b, p_c, p_d\}$ . All grasping configurations partially consisting of some contact points in  $T_G$  are verified for force closure condition and reported as new vertices in  $R$ . New edges are computed among the new vertices and between the new vertices and the recent vertices of  $R$  to complete updating  $R$ . Clearly,  $g$  is associated with a vertex in  $R$ . Therefore, a path from  $g$  to any grasp in  $R$  can be computed by a graph search.

---

**Algorithm 3**


---

- 1:  $T_G = \{p_a, p_b, p_c, p_d\}$
  - 2:  $T_A = T_R \cup T_G$
  - 3:  $L = \text{ConstructRoadmap}(T_A, T_A, T_A, T_G)$
  - 4: Update  $R$  by adding  $L$  and linking  $L$  to  $R$
- 

In conclusion, the overall algorithm firstly clusters the input contact points. Then a representative-level roadmap  $R$  is constructed from the representatives. To query a regrasp sequence from an initial grasping configuration  $g$  to a goal grasping configuration  $h$ , they have to be linked with  $R$  by using Algorithm 1 and 2. These algorithms find a path between  $g$  to a vertex in  $R$  and so on for  $h$ . If the algorithms fail to report a path, Algorithm 3 is performed by adding  $g$  or  $h$  into  $R$ . Finally, a graph search is then applied to find a path connecting two grasps in  $R$ .

### 6.3 Experiments and Results

The test objects are shown in Fig. 6.2. They are simplified and modeled with about 500 and 1000 triangular meshes. The half angle of friction cones is  $10^\circ$ . All objects are clustered into 30, 50 and 70 groups for regrasp planning. All run times are measured on a PC with a 2.4 GHz CPU. Examples of regrasping sequences are presented in Fig. 6.3 and 6.4.

Table 6.1-6.6 show the result from the construction of representative-level roadmaps. The result presents for each test object the number of force closure grasp vertices of the resulting representative-level roadmap, time spent in the construction and the number of connected components of the roadmap. The numbers of vertices of roadmaps depend heavily on the numbers of groups. Another factor is the object's shape. The objects in Fig. 6.2(a) and (f) result in more vertices than the others due to their sphere-like shapes which generally yield more force closure grasps. Note that the number of vertices is much smaller than the total number of force closure grasps that can be formed by the input contact points (up to millions for each test object). Small number of connected components

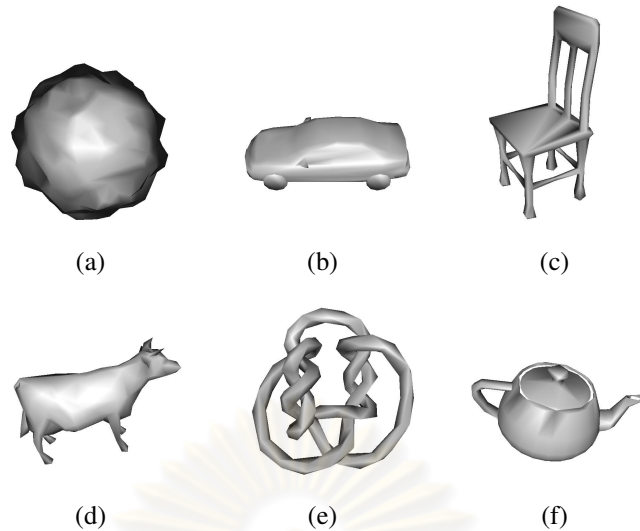


Figure 6.2: Test objects

indicates high probability to have a path joining any two vertices in the roadmap.

The tables also present the result from querying regrasping sequences. We randomly pick 30 pairs of force closure grasps to serve as the initial and target grasps for each query. Although the number of connected components of the representative-level roadmaps are quite large for some objects, successful regrasping sequences can be found for most pairs of grasps. This implies that the majority of grasps lie in the same connected component. The success rates tend to increase when the numbers of groups are increased as we can see from the situations when the numbers of groups are changed from 30 to 50. Minimum, maximum and average querying times are also shown in the tables. The numbers of vertices and the numbers of meshes affect computational times used to find a path connecting two grasps. The former effects when Algorithm 2, 3 and representative-level planner are executed whereas the later effects when all local planners are executed. From our experiments, minimum times are spent when both initial and target grasps can connect to representative-level roadmaps by Algorithm 1. Maximum times mostly occur when Algorithm 3 is provoked. However, when the number of groups is quite low such as 30 groups of 1,000 triangles, the average number of members in a group is greater than the number of groups so that computing a local roadmap by Algorithm 1 and 2 takes more computational time than provoking Algorithm 3. As discussed above, our approach provides trade off between completeness and resource used in the computation. From the results, the numbers of vertices and the success rates depend on the numbers of groups. Less number of groups provides faster computational times but variations of grasps and success rates decrease. This is reasonable for an ordinary personal computer to be used

in the regrasp planning problem.

For more insight of the local planners, the results of local planning are shown in Table 6.7 - 6.12. For each setting, we sample 10,000 force closure grasps. For Algorithm 1, each sampled grasp is verified whether the associated representatives form a force closure grasp. The numbers of force closure grasps found are shown in percent. For Algorithm 2, the procedure *FindNearestCluster* is executed for each sampled grasp. A grasp passes the verification if the variable  $X$  is determined. Then 30 grasps are randomly picked from the sampled grasps that pass the verification for each algorithm. Each grasp is planned for a regrasping sequence that joins it to the representative-level roadmap. The results list the numbers of grasps (out of 30) for which such sequence are found. To measure the efficiency of Algorithm 3, 30 sampled grasps that do not pass the verifications of Algorithm 1 and 2 are randomly picked. We then perform Algorithm 3 for each of them and verify whether it connects with part of the representative-level roadmap that exists before the update by Algorithm 3.

The results show the low passing rate of the verification by Algorithm 1 for some objects. Most grasps that pass the verification however can be connected to the representative-level roadmap. Algorithm 2 appears to be more effective than Algorithm 1. The passing rates for the verification of Algorithm 2 are significantly higher than those of Algorithm 1, (with slightly fewer regrasping sequences found on average). However, Algorithm 1 is still needed because of its considerably higher verification speed. By varying the numbers of groups, in most cases, the results show that the execution rates of Algorithm 1 and 2 increase and the numbers of achievements of path connection also tend to increase when the number of group is increased. Although applying Algorithm 3 can connect most grasps to the representative-level roadmap, it takes much longer to update the representative-level roadmap than to run both Algorithms 1 and 2 (the computational times are not shown here). However, computational time of Algorithm 3 will decrease when the number of groups is decreased because the number of vertices of the representative-level roadmap decreases and updating the representative-level roadmap takes less computational time.

#### 6.4 Summary

We have proposed a new approach to solve the regrasp planning problem. Existing methods typically solve the problem only for a complete solution. However, using such approach on real world data such as highly complex contact points is next to impossible.



Table 6.1: Result of 500 mesh objects clustered into 30 clusters

Fig.	#Vert	#CC	time(s)	#Con	Search time(s)		
					min	max	avg
(a)	2156	1	3.04	30	0.21	3.46	0.94
(b)	1101	1	1.60	30	0.14	3.27	1.28
(c)	386	7	0.66	28	0.16	2.06	0.94
(d)	341	13	0.61	26	0.12	2.98	1.01
(e)	374	14	0.65	27	0.15	1.82	0.76
(f)	2071	1	2.84	30	0.31	7.16	1.80

Table 6.2: Result of 500 mesh objects clustered into 50 clusters

Fig.	#Vert	#CC	time(s)	#Con	Search time(s)		
					min	max	avg
(a)	15928	1	22.67	30	1.62	19.92	2.47
(b)	7752	2	10.89	30	0.79	17.32	3.12
(c)	2970	12	4.98	30	0.32	7.42	1.82
(d)	2281	12	4.25	29	0.23	6.16	1.92
(e)	2479	31	4.39	28	0.24	7.77	2.69
(f)	10993	2	15.54	30	1.13	15.85	3.25

This is due to combinatorial explosion of the search space. Our approach provides trade off between completeness and the resource used in the computation. It clusters the input using spectral clustering. The representatives from all clusters are used to constrain possible search space. The underlying idea is similar to that of the classical motion planning problem. We construct a partial solution, called a representative-level roadmap. This allows the original problem to be divided into three parts: planning regrasp sequence from the starting grasp to the roadmap, planning regrasp sequence in the roadmap and, finally, planning regrasp sequence from the roadmap to the target grasp. Since the set of representatives contains much fewer contact points, solving the problem on the roadmap is much less complex. Nevertheless, this is achieved at the cost of completeness.

Table 6.3: Result of 500 mesh objects clustered into 70 clusters

Fig.	#Vert	#CC	time(s)	#Con	Search time(s)		
					min	max	avg
(a)	69898	1	100.55	30	10.88	76.42	13.42
(b)	30434	1	44.58	30	4.02	23.48	4.84
(c)	8562	9	16.28	30	0.67	10.06	4.07
(d)	10322	10	19.03	30	0.73	22.41	3.79
(e)	7683	23	14.87	30	0.45	15.29	4.20
(f)	51675	3	73.17	30	8.49	160.31	14.02

Table 6.4: Result of 1000 mesh objects clustered into 30 clusters

Fig.	#Vert	#CC	time(s)	#Con	Search time(s)		
					min	max	avg
(a)	2813	1	3.91	30	1.17	19.50	7.26
(b)	754	1	1.13	29	0.89	25.13	7.34
(c)	846	2	1.29	30	0.16	11.45	3.33
(d)	321	14	0.59	24	0.42	6.49	2.68
(e)	278	12	0.51	28	0.20	7.45	1.78
(f)	1137	3	1.66	30	1.15	29.11	7.67

Table 6.5: Result of 1000 mesh objects clustered into 50 clusters

Fig.	#Vert	#CC	time(s)	#Con	Search time(s)		
					min	max	avg
(a)	17056	1	24.25	30	2.19	42.75	10.56
(b)	6249	1	9.02	30	0.85	11.95	3.28
(c)	3747	7	6.02	30	0.40	6.01	2.36
(d)	1969	11	3.75	29	0.20	7.40	2.29
(e)	3770	12	6.09	30	0.43	9.67	2.79
(f)	14580	2	20.37	30	2.04	21.25	5.65

Table 6.6: Result of 1000 mesh objects clustered into 70 clusters

Fig.	#Vert	#CC	time(s)	#Con	Search time(s)		
					min	max	avg
(a)	79444	1	110.41	30	15.82	18.16	16.83
(b)	25064	2	36.04	30	3.62	28.40	7.37
(c)	9288	19	17.08	30	0.89	12.99	4.19
(d)	11352	11	20.19	30	1.01	23.53	5.29
(e)	8258	23	15.33	29	0.76	10.12	4.28
(f)	46386	3	64.47	30	7.88	164.11	22.22

Table 6.7: Result of local planning of 500 mesh objects clustered into 30 clusters

Fig.	Algorithm 1		Algorithm 2		Algorithm 3
	%run	#connect	%run	#connect	#connect
(a)	24.58	30	87.75	29	30
(b)	25.94	29	81.23	30	30
(c)	8.90	22	72.47	24	29
(d)	7.57	26	70.49	26	29
(e)	6.06	25	63.01	23	28
(f)	37.88	30	84.77	30	30

Table 6.8: Result of local planning of 500 mesh objects clustered into 50 clusters

Fig.	Algorithm 1		Algorithm 2		Algorithm 3
	%run	#connect	%run	#connect	#connect
(a)	30.21	30	93.24	30	30
(b)	34.88	28	87.45	29	30
(c)	15.24	27	84.69	24	29
(d)	9.62	28	81.14	24	30
(e)	9.62	27	79.12	21	29
(f)	32.32	30	90.33	30	30

Table 6.9: Result of local planning of 500 mesh objects clustered into 70 clusters

Fig.	Algorithm 1		Algorithm 2		Algorithm 3
	%run	#connect	%run	#connect	#connect
(a)	40.59	30	95.08	28	30
(b)	47.41	29	92.49	24	30
(c)	14.65	27	89.13	20	30
(d)	14.90	26	91.72	23	30
(e)	10.25	27	87.35	19	30
(f)	47.54	30	93.64	30	30

Table 6.10: Result of local planning of 1000 mesh objects clustered into 30 clusters

Fig.	Algorithm 1		Algorithm 2		Algorithm 3
	%run	#connect	%run	#connect	#connect
(a)	30.28	30	86.91	30	30
(b)	24.49	30	80.42	28	30
(c)	15.92	26	79.84	24	29
(d)	6.46	28	68.67	28	30
(e)	4.01	25	60.50	27	29
(f)	25.27	30	83.20	30	30

Table 6.11: Result of local planning of 1000 mesh objects clustered into 50 clusters

Fig.	Algorithm 1		Algorithm 2		Algorithm 3
	%run	#connect	%run	#connect	#connect
(a)	30.01	30	92.79	30	30
(b)	38.01	30	89.28	30	30
(c)	17.18	25	84.08	22	30
(d)	9.05	28	78.80	30	30
(e)	10.59	27	85.08	27	30
(f)	43.17	30	92.03	29	30

Table 6.12: Result of local planning of 1000 mesh objects clustered into 70 clusters

Fig.	Algorithm 1		Algorithm 2		Algorithm 3
	%run	#connect	%run	#connect	#connect
(a)	38.39	30	95.07	30	30
(b)	40.72	29	91.80	28	30
(c)	12.89	28	89.44	23	30
(d)	16.06	28	90.74	27	30
(e)	9.23	27	86.34	28	30
(f)	49.28	29	93.77	30	30

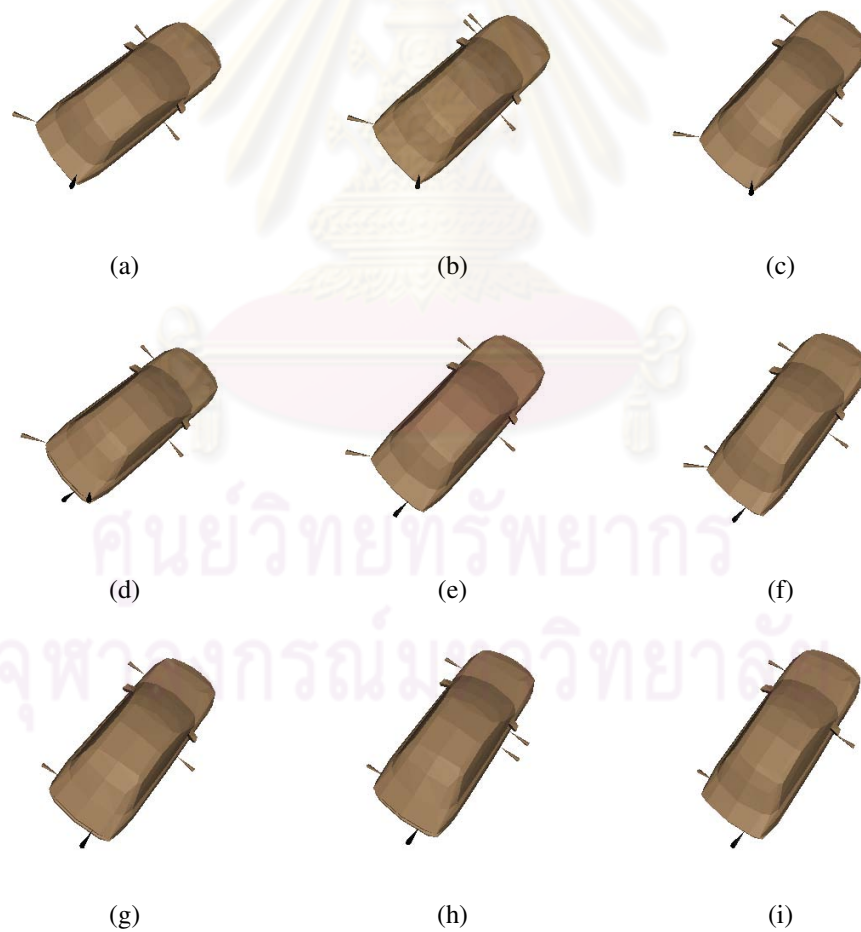


Figure 6.3: A regrasping sequence for the object in Fig. 6.2(b) with 500 triangles clustered into 50 groups

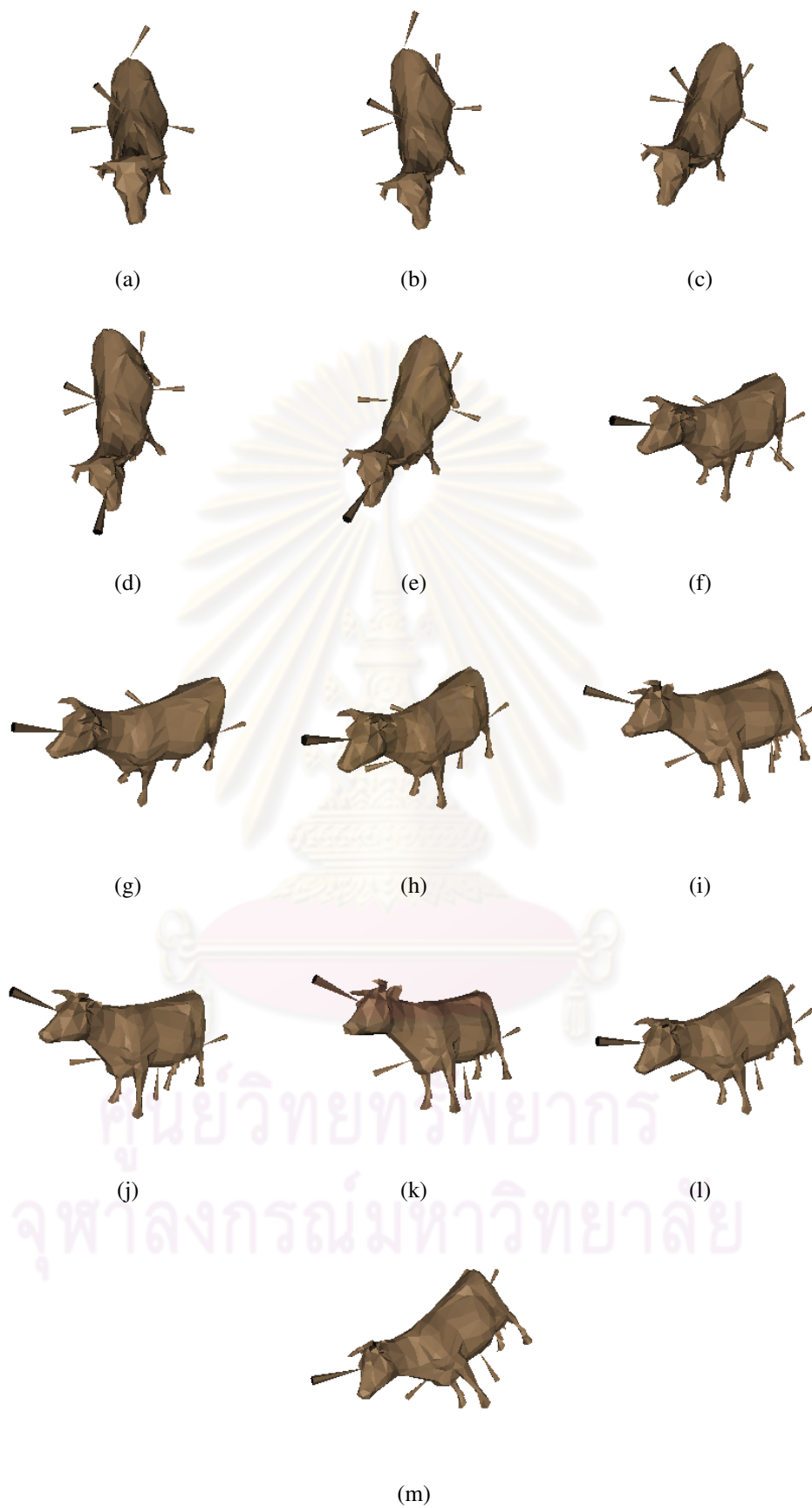


Figure 6.4: A regrasping sequence for the object in Fig. 6.2(d) with 1000 triangles clustered into 30 groups

# CHAPTER VII

## CONCLUSION

### 7.1 Dissertation Summary

In this work, we consider the problem of regrasp planning on an object that is described by a polygon, polyhedron or a set of discrete contact points. Given an object, our algorithm constructs a graph structure that stores sets of force-closure grasps and captures abilities of changing grasping configurations among these grasp sets using basic operations which are finger switching and finger aligning.

For a polygon, our algorithm computes a switching graph to store sets of force-closure grasps. A node in the graph contains a set of 2-finger grasps, concurrent grasps or parallel grasps for couple or triple of polygonal edges. In the same node, a finger aligning can be performed without losing contacts on the grasped edges. A finger switching is performed to change grasping configuration between two different sets of grasped edges. This operation directly involves with an edge of the switching graph. We apply a necessary and sufficient condition for 2-finger force-closure grasp to compute a 2-finger grasp node. However, in the case of 3-finger grasps, a necessary and sufficient condition is non-linear. We simplify the condition into two grasping types : concurrent grasps and parallel grasps of which constraints are linear. Further, representations of all grasping types are in 2D and 3D spaces, we can apply linear algebra and computational geometry theory to compute the complete switching graph.

For a polygon consisting of a large number of edges, we propose an algorithm to solve the problem of finger switching by three fingers. A node in a switching graph contain a connected set of 2-finger grasps. This allows finger aligning across edges of a polygon and also reduces computational cost of a switching graph construction. By applying the principal of  $L_\infty$  Voronoi diagram, we can locally optimize finger switching based on independent contact region criterion.

For the 3D case, for each four faces of a polyhedron, we consider only concurrent grasps which can be represented by a set of points in spatial. Our algorithm applies the same principal of the polygon case. A node of the switching graph contains a set of

concurrent grasps for four grasped faces. An edge linking two nodes when there exist two grasps from the distinct nodes that change to each other using a finger switching. However, computing a complete switching graph may take much running time. We propose a random approach to compute partial solution of a switching graph. The obtained switching graph is constructed in lazy fashion, i.e., actual force-closure grasp sets are not computed when the switching graph is being constructed. The results show that the random approach drastically decreases running time but it can almost capture the topology of the complete switching graph.

For the discrete contact point set case, we propose a heuristic approach to solve the regrasp planning problem. Since the input is discrete, only the finger switching is considered. Our algorithm exploits similarities among contact points to group them into disjoint clusters. This principal is induced from the finger switching. If we want to change one contact point in a grasping configuration, contact points that produce similar wrenches are appropriate for switching. Therefore, contact points producing similar wrenches are grouped into the same cluster. A representative contact point is chosen for each cluster. A graph structure called representative-level roadmap is constructed by considering 4-finger force-closure grasps whose configurations consist of representative contact points to decrease search space for regrasp planning. Clearly, this roadmap does not cover all contact points. Given arbitrary initial and target grasping configurations, a local planner is also proposed to find paths from these grasping configurations to the representative-level roadmap. In our experiments, the results show that our approach can mostly find a regrasp sequence between two arbitrary grasping configurations in a few seconds whereas constructing the complete roadmap and planning a regrasp sequence over it seem to be not possible using an ordinary PC.

## 7.2 Further Improvement and Extension

In this section we list some future improvement and extension that could be done to this work.

1. **Condition improvement:** All force-closure conditions applied for regrasp planning of a polygon and a polyhedron are sufficient conditions. To plan in the complete search space, necessary and sufficient conditions could be taken place. A new planner is needed to handle with non-linear constraints. Appropriate grasping representations are also required to describe a set of force-closure grasps. However, it is quite complex to compute the exact geometry of a grasp set. A node of a switch-



ing graph may contain a set of constraints instead. An edge may contain a set of constraints for finger switchings. We can apply non-linear optimization to prove the existence of a solution of non-linear constraints. we are also interested in addition of the other two types of 4-finger force closure grasps (i.e., pencil and regulus grasps) to our regrasp planning.

2. **Optimized regrasp sequence:** A regrasp sequence obtained from our algorithm is guaranteed for force-closure but it is not considered for its quality or qualities of grasps in the sequence. Quality measure metrics for a grasp can be exploited such as independent contact region, Q-distance, etc. There are many ways to determine a quality of a regrasp sequence such as integrating qualities of all grasps in the sequence or optimizing bound of qualities of all grasps in the sequence.
3. **Random approach improvement and application:** The random approach in this work uses the ordinary random function in C++ programming language. There are many probabilistic approaches in motion planning such as PRM and RRT that could be improve convergence of the switching graph construction. We can also apply the random approach to plan a regrasp sequence over pre-computed sets of force-closure grasps and finger switching in a switching graph. For example, in the case of a polyhedron, sample points in a focus cell for finger aligning and sample points in the intersection of two focus cell that have one distinct grasped face for finger switching operations.
4. **Local planning improvement:** For the problem of regrasp planning for discrete contact points, we can speed up the local planner by exploiting the properties of the spectral clustering algorithm. In the clustering procedure, the measurement of a contact point is converted into Euclidean space and  $L - 2$  distance is used to cluster contact points. Since  $L - 2$  distance is metric distance function, we can apply the existence nearest neighbor search algorithms to improve the determination of sets  $S_i$ .
5. **Including hand constraints:** With pre-computed sets of force-closure grasp at hand, we can plan a regrasp sequence in these sets including kinematic and dynamic constraints of a hand without verifying for force-closure anymore. Recently, probabilistic approaches are reasonable when these constraints are included.

### 7.3 Discussion

An autonomous robot that accomplishes a required task with minimum supervision is a goal yearned by most researchers. A similar goal is also set for the dexterous manipulation problem. It is the uttermost goal of this dissertation to, at least, provide a stepping stone to that problem.

Recently, robot hands are widely applied to many tasks instead of human because they work with more decision and endurance. As we can see in many industries, robot hands are programmed to assembly cars or tiny circuit boards. However, the ability of the robot hands are much less comparing with a human hand. To control a robot hand, a planner has to consider many constraints : task, grasping stability, kinematic constraints and dynamic of the hand. Kinematic focuses on a robot hands' geometry, an object's geometry, configurations and limitations that allow us to derive relations among joints and fingertips' positions and also relations between the hand and the object. Many works in kinematics often assume that a robot hand is perfectly controlled neglecting hand dynamic. Actually, forces at fingertips are exerted via movements or rotations of joints in a robot hand which requires analysis of hand dynamic. The remaining procedure is to control forces and torques of joints for precision of fingertip positioning and forces exerted at the fingertips. Although hand kinematic and dynamic are necessary for analysis of the dexterous manipulation, it also requires higher level planner that provides a manipulation sequence satisfying grasping constraints. Recently, there are a few works that attack the problem of planning a sequence of finger repositioning which our work focuses on. Our planner provides a sequence of finger repositioning that all grasps in the sequence satisfy force-closure. For a given object, our approach constructs a graph structure that contains a set of force-closure grasps in a node and an edge is associated with the finger switching operation. We apply this graph structure as a framework for the regrasp planning problem. Since the graph contains sets of force-closure grasps, a planner is allowed to include other constraints such as kinematic or dynamic of an arbitrary hand for a regrasp sequence that all grasps satisfying these additional constraints also maintain force-closure.

The main advantage of the switching graph is that it *explicitly* contains sets of force-closure and sets of finger switchings. Note that the traditional necessary and sufficient conditions for computing a set of force-closure grasp on given edges are non-linear. Therefore, the set of force-closure grasps is *implicitly* represented by non-linear constraints which are complex to transform them into geometries. The advantage of our approach is strongly based on the simplifications of force-closure conditions. In 2D, we

classify grasps into three types which are the 2-finger grasp, the concurrent grasp and the parallel grasp. For given grasped edges, the conditions of all grasping types can be formulated into linear constraints. A set of 2-finger grasps and a set of concurrent grasps are represented by a set of points in the plane. In contrast, a set of parallel grasps consists of polytopes in the 3D parameter space. Finger switchings between grasps in distinct two sets are computed using existing boolean operation of polygons in the plane. Although our approach simplifies the force-closure conditions into sufficient conditions, but we can solve the problem efficiently using linear algebra and computational geometry in 2D. The results evidences that the proposed approach covers a large number of force-closure grasp sets which adequate for the regrasp planning of a polygon.

For a polyhedron, we focus on concurrent grasps which are natural for 3D grasping, i.e., exerted forces intersect at a point. A set of points in spatial is used to represent a concurrent grasp set. Although the condition of concurrent grasps is just a sufficient condition, but it reduces the dimension of the representation from 8D (2 parameters for a contact point) into 3D. This condition allows us to apply the existing geometric computation library which is ACIS library to our implementation. Moreover, based on the representation of concurrent grasps in 3D, the regrasp planning problem can be efficiently solved using a probabilistic approach in low dimensions.

Our last problem is the regrasp planning for discrete contact points. Discrete contact points suit more to the data acquisition sensors, such as a laser range scanner or a stereoscopic camera which are widely available. Discrete contact point model also calls forth the need to handle input of a large number of contacts. Though it is possible to approximate the scanned data with one polynomial, this approach suffers from the high cost of curve fitting and the accuracy problem from Runge phenomenal. Spline fitting, arguably, reduces the effect of both problems but the result is still a large number of polynomials. In fact, when the resolution of the scan is large enough, spline fitting results in similar representation of discrete contact points. We realize that the use of the discrete contact point model is necessary for complete automation. Evidently, this problem is included as one objective of this dissertation.

Using discrete contact points result in an enormous number of contact points of which all force closure grasps and finger switchings must be computed. The number of the solutions can be as high as  $O(N^4)$  and  $O(N^5)$ . It is precisely this problem that our work tries to cope with. Instead of planning in the whole search space, we apply a two-level scheme approach to plan in much smaller search space. We show that, at least,

the pre-computed representative-level roadmap contains partial solutions; it is possible to solve a regrasp planning problem using our proposed local planner in much lesser time than complete approach and there are many possible improvement that could be done.

Another advantage of our framework is generality of the structure which does not specifically depend on a task or a robot hand. The switching graph can be considered as a middle level in manipulation planning. Given an initial grasp and a goal grasp from a task planner, the switching graph provides sets of force-closure grasps and finger switchings that involve changing of grasps between the two grasps. These sets and their relations are then transferred to lower levels to compute feasible trajectories of a hand constrained on the grasp sets and the finger switching sets. More practically, the switching graph can be applied to other planner. Computed grasp sets serve explicit wrench closure sets for the approach in (Jr et al., 2004). Also, finger switching sets can be applied to search for transitions of contacts. For the recent approach in (Saut et al., 2007), the switching graph provides the explicit grasp subspaces from which a PRM planner can sample grasp sequences without verifying all generated grasps for the force-closure condition. Another advantage is globalization of the switching graph. Once we compute a switching graph that contains all force-closure grasps for an object, the switching graph then allows a planner to globally search for sets of force-closure grasps and finger switchings.

The author strongly believe that, with the proposed algorithms and the proposed approaches, we could see many interesting, or the ultimate, solution to the dexterous manipulation problem in the near future.

## References

- Alami R., Siméon T. and Laumond J.P. A geometrical approach to planning manipulation tasks. In International Symposium on Robotics Research. 1989.
- Bicchi A. On the closure properties of robotic grasping. International Journal of Robotics Research, 14(4) (August 1995): 319–334.
- Bicchi A. and Marigo A. Rolling contacts and dexterous manipulation. In IEEE Int. Conf. on Robotics and Automation. 2000.
- Borst C., Fischer M. and Hirzinger G. Grasping the dice by dicing the grasp. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 2003.
- Boyd S. and Vandenberghe L. Convex Optimization. New York: Cambridge University Press, 2004.
- Brock D.L. Enhancing the dexterity of robot hands using controlled slip. In IEEE Int. Conf. on Robotics and Automation. 1988.
- Butterfaß J., Grebenstein M., Liu H. and Hirzinger G. DLR-hand II: Next generation of a dextrous robot hand. pp. 109–114, In IEEE Int. Conf. on Robotics and Automation. 2001.
- Cai C. and Roth B. On the spatial motion of a rigid body with point contact. pp. 686–695, In IEEE Int. Conf. on Robotics and Automation. 1987.
- Cai C. and Roth B. On the spatial motion of a rigid body with line contact. pp. 1036–1041, In IEEE Int. Conf. on Robotics and Automation. 1988.
- Cherif M. and Gupta K.K. Planning quasi-static motions for re-configuring objects with a multi-fingered robotic hand. pp. 986–991, In IEEE Int. Conf. on Robotics and Automation. 1997.
- Chong N.Y., Choi D. and Suh I.H. A finite motion planning strategy for multifingered robotic hands considering sliding and rolling contacts. pp. 180–187, In IEEE Int. Conf. on Robotics and Automation. 1993.
- Cole A.A., Hsu P. and Sastry S. Dynamic regrasping by coordinated control of sliding for a multifingered hand. pp. 781–786, In IEEE Int. Conf. on Robotics and Automation. 1989.

- Cole A.A., Hsu P. and Sastry S. Dynamic control of sliding by robot hands for regrasping. IEEE Transactions on Robotics and Automation, 8(1) (February 1992): 42–52.
- Cornella J. and Suarez R. Determining independent grasp regions on 2d discrete objects. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 2005a.
- Cornella J. and Suarez R. Fast and flexible determination of force-closure independent regions to grasp polygonal objects. In IEEE Int. Conf. on Robotics and Automation. 2005b.
- Corney J. and Lim T. 3D Modeling with ACIS. Kippen Stirling: Saxe-Coburg Publications, 2002.
- Cutkosky M.R. On grasp choice, grasp models, and the design of hands for manufacturing tasks. IEEE Transactions on Robotics and Automation, 5(3) (1989): 269–279.
- Dandurand A. The rigidity of compound spatial grid. Structural Topology, 10.
- de Berg M., van Kreveld M., Overmars M. and Schwarzkopf O. Computational Geometry: Algorithms and Applications. Berlin Heidelberg: Springer, 1997.
- Faverjon B. and Ponce J. On computing two-finger force-closure grasps of curved 2D objects. pp. 424–429, In IEEE Int. Conf. on Robotics and Automation. Sacramento, CA. 1991.
- Fearing R. Implementing a force strategy for object re-orientation. In IEEE Int. Conf. on Robotics and Automation. 1986.
- Ferrari C. and Canny J. Planning optimal grasps. pp. 2290–2295, In IEEE Int. Conf. on Robotics and Automation. Nice, France. 1992.
- Goldfeder C., Allen P.K., Lackner C. and Pelosof R. Grasp planning via decomposition trees. In IEEE Int. Conf. on Robotics and Automation. 2007.
- Goodwine B. and Burdick J. Motion planning for kinematic stratified systems with application to quasi-static legged locomotion and finger gaiting. IEEE Transactions on Automatic Control, 18(2) (2002): 209–222.
- Han L. and Trinkle J. Dextrous manipulation by rolling and finger gaiting. pp. 730–735, In IEEE Int. Conf. on Robotics and Automation. 1998a.
- Han L. and Trinkle J. Dextrous manipulation by rolling and finger gaiting. In IEEE Int. Conf. on Robotics and Automation. 1998b.

- Harmati I., Lantos B. and Payandeh S. On fitted stratified and semistratified geometric manipulation planning with fingertip relocations. International Journal of Robotics Research, 21(5) (2002): 489–510.
- Hoffman C.M. Geometric and Solid Modeling. Morgan Kaufmann, San Mateo, California, 1989.
- Hong J., Lafferiere G., Mishra B. and Tan X. Fine manipulation with multifinger hands. pp. 1568–1573, In IEEE Int. Conf. on Robotics and Automation. Cincinnati, OH. 1990.
- Huber M. and Grupen R.A. Robots finger gaits from closed-loop controllers. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 2002.
- Huebner K., Ruthotto S. and Kragic D. Minimum Volume Bounding Box Decomposition for Shape Approximation in Robot Grasping. In IEEE Int. Conf. on Robotics and Automation. 2008.
- Jacobsen S., Iversen E., Knutti D., Johnson R. and Bigger K. Design of the utah/mit dextrous hand. pp. 96–102, In IEEE Int. Conf. on Robotics and Automation. 1986.
- Jia Y.B. On computing optimal planar grasps. In IEEE Int. Conf. on Robotics and Automation. 1995.
- Jr R.P., Fagg A.H. and Grupen R.A. Manipulation gaits: Sequences of grasp control tasks. In IEEE Int. Conf. on Robotics and Automation. 2004.
- Kerr J. and Roth B. Analysis of multifingered hands. International Journal of Robotics Research, 4(4) (1986): 3–17.
- Kirkpatrick D., Mishra B. and Yap C. Quantitative Steinitz's theorems with applications to multifingered grasping. pp. 341–351, In 20<sup>th</sup> ACM Symp. on Theory of Computing. Baltimore, MD. 1990.
- Koga Y. and Latombe J. On multi-arm manipulation planning. In IEEE Int. Conf. on Robotics and Automation. 1994.
- Leveroni S.R. Grasp Gaits for Planar Object Manipulation. Ph.D. thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering (1997).

- Li J., Zhang Y. and Zhang Q. Kinematic algorithm of multifingered manipulation with rolling contact. pp. 338–343, In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 2000.
- Li Y.F., Tso S.K. and Meng Q. Grasping force measurement for dynamic grasp stability assessemnt. IEEE Transactions on Instrumentation and Measurement, 47(5) (1998): 1294–1299.
- Liu Y.H. Computing n-finger force-closure grasps on polygonal objects. In IEEE Int. Conf. on Robotics and Automation. 1998.
- Liu Y.H. Qualitative test and force optimization of 3-d frictional form-closure grasps using linear programming. IEEE Transactions on Robotics and Automation, 15(1) (1999): 163–173.
- Lovchik C.S. and Diftler M.A. The robonaut hand: A dexterous robot hand for space. pp. 907–912, In IEEE Int. Conf. on Robotics and Automation. 1999.
- Markenscoff X., Ni L. and Papadimitriou C.H. The geometry of grasping. International Journal of Robotics Research, 9(1) (February 1990): 61–74.
- Mehlhorn K. and Naher S. Leda: A Platform for Combinatorial and Geometric Computing. New York: Cambridge University Press, 2000.
- Mirtich B. and Canny J. Optimum force-closure grasps. Technical Report ESRC 93-11/RAMP 93-5, Robotics, Automation, and Manufacturing Program, University of California at Berkeley (July 1993).
- Mishra B., Schwartz J. and Sharir M. On the existence and synthesis of multifinger positive grips. Algorithmica, Special Issue: Robotics, 2(4) (November 1987a): 541–558.
- Mishra B., Schwartz J. and Sharir M. On the existence and synthesis of multifinger positive grips. Algorithmica, Special Issue: Robotics, 2(4) (November 1987b): 541–558.
- Montana D.J. Kinematics of contact and grasp. International Journal of Robotics Research, 7(3) (1988): 17–32.
- Munoz L.A., Bard C. and Najera J. Dexterous manipulation: A geometrical reasoning point of view. pp. 458–463, In IEEE Int. Conf. on Robotics and Automation. 1995.



- Nakamura Y., Nagai K. and Yoshikawa T. Dynamics and stability in coordination of multiple robotic mechanisms. International Journal of Robotics Research, 8(2) (1989): 44–61.
- Ng A.Y., Jordan M.I. and Weiss Y. On spectral clustering: Analysis and an algorithm. pp. 849–856, In Advances in Neural Information Processing Systems 14. MIT Press. 2001.
- Nguyen V. Labeling polyhedral scenes. pp. 1160–1165, In Proc. DARPA Image Understanding Workshop. 1988a.
- Nguyen V.D. Constructing force-closure grasps. International Journal of Robotics Research, 7(3) (June 1988b): 3–16.
- Nielsen C.L. and Kavraki L.E. A two level fuzzy prm for manipulation planning. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 2000.
- Niparnan N. and Sudsang A. Positive span of force and torque components of four-fingered three-dimensional force-closure grasps. pp. 4701–4706, In IEEE Int. Conf. on Robotics and Automation. Roma, Italy. 2007.
- Omata T. and Farooqi M.A. Regrasps by a multifingered hand based on primitives. pp. 2774–2780, In IEEE Int. Conf. on Robotics and Automation. 1996.
- Omata T. and Nagata K. Planning reorientation of an object with a multifingered hand. pp. 3104–3110, In IEEE Int. Conf. on Robotics and Automation. 1994.
- Papadopoulou E. and Lee D.T. The  $l_\infty$ -voronoi diagram of segments and vlsi applications. Int. J. Comput. Geometry Appl., 11(5) (2001): 503–528.
- Phoka T., Pipattanasomporn P., Niparnan N. and Sudsang A. Regrasp planning of four-fingered hand for parallel grasp of a polygonal object. In IEEE Int. Conf. on Robotics and Automation. 2005.
- Ponce J. and Faverjon B. On computing three-finger force-closure grasps of polygonal objects. IEEE Transactions on Robotics and Automation, 11(6) (December 1995a): 868–881.
- Ponce J. and Faverjon B. On computing three-finger force-closure grasps of polygonal objects. IEEE Transactions on Robotics and Automation, 11(6) (December 1995b): 868–881.

- Ponce J., Sullivan S., Sudsang A., Boissonnat J.D. and Merlet J.P. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. International Journal of Robotics Research, 16(1) (February 1997): 11–35.
- Reuleaux F. The Kinematics of Machinery. Macmillan 1876, republished by Dover, NY, 1963.
- Rimon E. and Burdick J. On force and form closure for multiple finger grasps. pp. 1795–1800, In IEEE Int. Conf. on Robotics and Automation, volume 2. 1996.
- Roa M. and Suarez R. Independent contact regions for frictional grasps on 3d objects. In IEEE Int. Conf. on Robotics and Automation. 2008.
- Roa M. and Suarez R. Regrasp planning in the grasp space using independent regions. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 2009.
- Sahbani A., Cortés J. and Siméon T. A probabilistic algorithm for manipulation planning under continuous grasps and placements. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 2002.
- Salisbury J. Kinematic and force analysis of articulated hands. Ph.D. thesis, Stanford University, Stanford, CA (1982).
- Sarkar N., Yun X. and Kumar V. Dynamic control of 3-d rolling contacts in two-arm manipulation. IEEE Transactions on Robotics and Automation, 13(3) (June 1997): 364–376.
- Saut J., Sahbani A., El-Khoury S. and Perdereau V. Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 2007.
- Siméon T., Cortés J., Sahbani A. and Laumond J.P. A manipulation planner for pick and place operations under continuous grasps and placements. In IEEE Int. Conf. on Robotics and Automation. 2002.
- Stewart D.E. Rigid-body dynamics with friction and impact. IEEE Transactions on Robotics and Automation, 42(1) (March 2000): 3–39.
- Sudsang A. and Ponce J. New techniques for computing four-finger force-closure grasps of polyhedral objects. In IEEE Int. Conf. on Robotics and Automation. 1995.
- Townsend W.T. The BarrettHand grasper—programmably flexible part handling and assembly. Industrial Robot: An International Journal, 27(3) (2000): 181–188.

- Tung C.P. and Kak A.C. Fast construction of force-closure grasps. IEEE Transactions on Robotics and Automation, 12(4) (1996): 615–626.
- Xu J., Koo T.J. and Li Z. Finger gaits planning for multifingered manipulation. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 2007.
- Xu J. and Li Z. A kinematic model of finger gaits by multifingered hand as hybrid automaton. IEEE Transactions on Automation Science and Engineering, 5(3) (2008): 515–522.
- Yoshikawa T. and Nagai K. Evaluation and determination of grasping forces for multifingered hands. In IEEE Int. Conf. on Robotics and Automation. 1988.
- Yoshikawa T. and Nagai K. Manipulating and grasping forces in manipulation by multifingered robot hands. IEEE Transactions on Robotics and Automation, 7(1) (1991): 67–77.
- Yoshikawa T., Yokokohji Y. and Nagayama A. Object handling by three-fingered hands using slip motion. pp. 99–105, In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. 1993.
- Zheng X.Z., Nakashima R. and Yoshikawa T. On dynamic control of finger sliding and object motion in manipulation with multifingered hands. IEEE Transactions on Robotics and Automation, 18(5) (October 2000): 469–481.
- Zhu X., Ding H. and Tso S.K. A pseudodistance function and its applications. IEEE Transactions on Robotics and Automation, 20(2) (2004): 344–352.
- Zhu X. and Wang J. Synthesis of force-closure grasps on 3-d objects based on the  $q$  distance. IEEE Transactions on Robotics and Automation, 19(4) (2003a): 669–679.
- Zhu X. and Wang J. Synthesis of force-closure grasps on 3-D objects based on the  $q$  distance. IEEE Trans. Robot. Autom., 19(4) (August 2003b): 669–679.

## Biography

Thanathorn Phoka was born in Bangkok, Thailand, on December, 1980. He received B.Eng. and M.Eng., both in computer engineering, from Chulalongkorn University, Thailand, in 2002 and 2004, respectively. In his master degree and the first few years in his doctorate, he also serves as a teaching assistant at the Department of Computer Engineering, Chulalongkorn University. His bachelor degree and his master degree have been supervised by Dr. Attawith Sudsang. His doctorate has been also under the supervision of Dr. Attawith Sudsang. In summer 2005, he was granted a visiting scholarship by Toshiba International Foundation to participate in training at Humancentric Laboratory, Research and Development Center, Toshiba Corporation, Kawasaki. Since 2007, he has received a grant from the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program under Grant No. Ph.D. 1.O.CU/49/D.1. In 2007, he additionally received a grant from the 90<sup>th</sup> Anniversary of Chulalongkorn University Fund through the Ratchadapiseksomphot Fund. His field of interest includes various topics in Robotics with emphasis on regrasp planning, grasp planning, grasp analysis and dexterous manipulation. In September 2009 - January 2010, he researched at Computer Science Department Department of Electrical Engineering, Stanford University.



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย