การคำนวณบริเวณสัมผัสอิสระที่ดีที่สุดบนวัตถุหลายเหลี่ยมในสองและสามมิติด้วยนิ้วจับสองนิ้ว

นายชัยชนะ นิลวัชรารัง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย
บทคัดย่อและแฟ้มข้อมูลฉบับเต็มของวิทยานิพนธ์ตั้งแต่ปีการศึกษา 2554 ที่ให้บริการในคลังปัญญาจุฬาฯ (CUIR)
เป็นแฟ้มข้อมูลของนิสิตเจ้าของวิทยานิพนธ์ที่ส่งผ่านทางบัณฑิตวิทยาลัย
The abstract and full text of theses from the academic year 2011 in Chulalongkorn University Intellectual Repository(CUIR)
are the thesis authors' files submitted through the Graduate School.

OPTIMAL INDEPENDENT CONTACT REGION FOR 2D AND 3D POLYGONAL OBJECT GRASPING BY TWO SOFT FINGERS

Mr. Chaichana Nilwatchararang

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2011

Copyright of Chulalongkorn University

หัวข้อวิทยานิพนธ์	การคำนวณบริเวณสัมผัสอิสระที่ดีที่สุดบนวัตถุหลายเหลี่ยมในสองและสามมิติด้วยนิ้วจับสองนิ้ว
โดย	นาย ชัยชนะ นิลวัชรารัง
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.อรรถวิทย์ สุดแสง

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

…… ประธานกรรมการ
(ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา)

…….. อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร.อรรถวิทย์ สุดแสง)

…… กรรมการ
(อาจารย์ ดร.นัทที นิภานันท์)

…… กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.จิตร์ทัศน์ ฝักเจริญผล)

ชัยชนะ นิลวัชรารัง : การคำนวณบริเวณสัมผัสอิสระที่ดีที่สุดบนวัตถุหลายเหลี่ยมในสองและสามมิติด้วยนิ้วจับสองนิ้ว (OPTIMAL INDEPENDENT CONTACT REGION FOR 2D AND 3D POLYGONAL OBJECT GRASPING BY TWO SOFT FINGERS). อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร.อรรถวิทย์ สุดแสง, 83 หน้า.

วิทยานิพนธ์นี้นำเสนอวิธีคำนวณหาบริเวณสัมผัสอิสระที่ดีที่สุดบนวัตถุหลายเหลี่ยมในสองและสามมิติด้วยนิ้วจับสองนิ้ว ในกรณีสองมิติ จะใช้วิธีเปลี่ยนปัญหาไปบนมิติของบริเวณที่จับติดและทำการคำนวณหาสี่เหลี่ยมจตุรัสที่ใหญ่ที่สุดที่อยู่บนมิติดังกล่าวโดยใช้ L_∞ voronoi diagram เป็นตัวหาคำตอบ ในกรณีสามมิติจะใช้การเปลี่ยนปัญหาให้อยู่ในรูปของกราฟความสัมพันธ์ของความใกล้กันของแต่ละหน้าของวัตถุสามมิติและคุณสมบัติฟอร์สโคลสเซอร์ของแต่ละคู่หน้าและใช้การประมวลผลแบบขนาดโดยการ์ดแสดงผลเป็นตัวคำนวณหาบริเวณสัมผัสอิสระที่ดีที่สุด

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา 2554

5070684121: MAJOR COMPUTER ENGINEERING
KEYWORDS: GRASPING / INDEPENDENT CONTACT REGION / GPU / CUDA

CHAICHANA NILWATCHARARANG: OPTIMAL INDEPENDENT CONTACT REGION FOR 2D AND 3D POLYGONAL OBJECT GRASPING BY TWO SOFT FINGERS. ADVISOR: ATTAWITH SUDSANG, Ph.D., 83 pp.

This thesis presents you two methods for calculating optimal independent contact region for 2D and 3D case. In 2D case, we'll construct graspable region on configuration space and then use L_∞ voronoi diagram to find largest square which represents optimal contact region of that grasp. In 3D case, we'll transform the problem into graph system which vertex represent polyhedral face and 2 type of edge which represent adjacency property and force closure grasp property between every pair of edges. Then use GPU programming to calculate optimal independent contact region in parallel programming manners.

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

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างสูงจาก ผศ. ดร. อรรถวิทย์ สุดแสง อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้ประสิทธิ์ประสาทความรู้ คำแนะนำ และข้อคิดเห็นต่าง ๆ ในการวิจัย รวมถึงการดำเนินชีวิต ผู้วิจัยรู้สึกซาบซึ้งและปลาบปลื้มในความโอบอ้อมอารีเป็นอย่างมาก และขอขอบคุณคณะกรรมการวิทยานิพนธ์ ศ.ดร.ประภาส จงสถิตย์วัฒนา, ผศ.ดร.จิตร์ทัศน์ ฝักเจริญผล และ ดร.นัทที นิภานันท์ ที่กรุณาเสียสละเวลาให้คำแนะนำตรวจและแก้ไขวิทยานิพนธ์ฉบับนี้

ขอบคุณสมาชิกชาวแลป ISL2 ทุกคนสำหรับความคิดสร้างสรรค์และความรู้รอบตัวที่ได้แบ่งปันกันมาในหลายปีที่ผู้วิจัยใช้ชีวิตอยู่ที่แลป ISL2 แห่งนี้

ขอบคุณพี่ฟู, ดร.นัททีและนุชเป็นพิเศษที่เป็นปัจจัยสำคัญในการทำให้การเขียนวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี

สุดท้ายนี้ ผู้วิจัยใคร่กราบขอบพระคุณและขอขมาบิดา มารดา ที่อดทนและให้โอกาสผู้วิจัยเสมอ ขอกราบขอบพระคุณและขอขมา ผศ.ดร.อรรถวิทย์ที่เปรียบเสมือนบิดาคนที่สองของผู้วิจัยที่คอยเปิดโลกทัศน์และให้ความรู้กับผู้วิจัยเป็นอย่างมาก รวมถึงต้องอดทนลุ้นกับผู้วิจัยในทุกเรื่อง (จะจบหรือไม่, งานจะส่งทันหรือไม่, ฯลฯ) ตั้งแต่สมัยเรียนปริญญาตรีจนถึงปัจจุบัน

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1

บทนำ

เป้าหมายสำคัญประการหนึ่งของการใช้งานหุ่นยนต์คือการจัดวัตถุ (object manipulation) ในบริเวณทำงานของหุ่นยนต์ (robot workspace) วิธีพื้นฐานซึ่งได้รับความสนใจในทางวิทยาการหุ่นยนต์คือการจับ (grasping) ด้วยมือหุ่นยนต์ คอนฟิกุเรชัน (configuration) ของการจับประกอบไปด้วยตำแหน่งของนิ้วหุ่นยนต์บนวัตถุ สิ่งสำคัญอย่างหนึ่งที่ต้องคำนึงถึงในการจับคือ เสถียรภาพ (stability) ซึ่งการจับที่มีเสถียรภาพนั้นถูกมองได้สองมุมมองหลักๆ คือ ในเชิงเรขาคณิต และในเชิงกลศาสตร์ แม้ว่านิยามของเสรียรภาพในการจับจะถูกมองได้หลายมุม แต่ในทางทฤษฎีแล้ว เราสามารถสนใจแค่ในมุมมองใดมุมมองหนึ่งก็เพียงพอสำหรับการอธิบายความมีเสถียรภาพของการจับ โดยสิ่งที่ต้องการสำหรับเสถียรภาพในการจับคือ เมื่อวัตถุถูกจับอย่างมีเสถียรภาพนั้น คอนฟิกุเรชันของการจับจะต้องไม่เปลี่ยนแปลงตลอดการทำงานไม่ว่าจะถูกรบกวนจากแรงภายนอกใด ๆ ยกตัวอย่างเช่น การจับค้อนเพื่อใช้ในงานตอกตะปู ในขณะที่ค้อนนั้นถูกใช้งาน ค้อนจะต้องไม่หลุดออกจากมือเนื่องจากแรงที่กระทำกับตะปู รวมถึงแรงที่เกิดจากแรงโน้มถ่วง ซึ่งการจับที่มีเสถียรภาพนี้จะถูกเรียกว่า การจับแบบปิดของแรง (force closure grasp) และเสถียรภาพนี้จะถูกเรียกว่า คุณสมบัติแบบปิดของแรง (force closure)

การจะคำนวณหาการจับที่มีเสถียรภาพนั้นไม่ใช่เรื่องที่ยากเกินไปนัก ระเบียบวิธีในการสังเคราะห์การจับแบบง่ายๆ ด้วยเวลาเชิงเส้นได้ถูกเสนอไว้ตั้งแต่ปี 1987 (Mishra et al. 1987) อย่างไรก็ตาม การจับแต่ละการจับนั้นมีคุณภาพที่ต่างๆ กัน เมื่ออยู่ในสภาวะที่ถูกจำลองขึ้น เงื่อนไขบางเงื่อนไขนั้นได้ถูกละเลย การจับทุกการจับจึงสามารถถูกมองให้เทียบเท่ากันได้ แต่ในการทำงานในโลกจริงเงื่อนไขรวมถึงสภาวะต่างๆ ต้องถูกนำมาพิจารณาด้วย ส่งผลให้เกิดการเลือกการจับที่เหมาะสมกับเงื่อนไขและสภาวะต่างๆ ที่เกิดขึ้นในการทำงานจริง

อย่างไรก็ตามการคำนึงถึงทุกเงื่อนไขเพื่อคำนวณหาการจับนั้นมีความซับซ้อนสูง ดังนั้นงานวิจัยที่เกี่ยวข้องกับการวางแผนการจับจึงมักจะแยกส่วนกลศาสตร์ของแขนกลออกจากการวางแผนหาตำแหน่งจับ ซึ่งแขนกลทุกๆ ชิ้นมีความคลาดเคลื่อนไม่มากก็น้อยแตกต่างกัน วิทยานิพนธ์นี้จึงได้มุ่งเน้นถึงปัญหาที่สภาวะของการควบคุมตำแหน่งจับที่ปลายนิ้วหุ่นยนต์มีความผิดพลาด เนื่องจากกลศาสตร์ของแขนกลได้ถูกละเลย การวางแผนการจับจึงต้องหาวิธีรับมือกับปัญหาความคลาดเคลื่อนในการจับที่เกิดขึ้น โดยการจับนั้นจะไม่เจาะจงกับแขนกลในแขนกลหนึ่ง

การคำนวณหาบริเวณสัมผัสอิสระ (independent contact region) จึงได้ถูกพัฒนาขึ้นเพื่อรับมือกับปัญหาดังกล่าว โดยคำนวณหาบริเวณสัมผัสอิสระของแต่ละปลายนิ้วจับแต่ละนิ้ว โดยที่แต่ละนิ้วนั้นสามารถวาง ณ ตำแหน่งใดในบริเวณสัมผัสอิสระและยังทำให้การจับมีคุณสมบัติแบบปิดของแรงเสมอ ดังนั้นสิ่งที่เราต้องคำนึงถึงในการหาการจับที่เหมาะสมกับสภาวะนี้คือ การจับที่ยอมให้เกิดความผิดพลาดในตำแหน่งการจับมากที่สุด นั่นก็คือการจับที่มีบริเวณสัมผัสอิสระที่กว้างที่สุดนั่นเอง

วิทยานิพนธ์นี้ศึกษาปัญหาการคำนวณบริเวณสัมผัสอิสระรวมถึงปัญหาการวางวางแผนการจับที่มีบริเวณสัมผัสอิสระที่กว้างที่สุดสำหรับวัตถุรูปหลายเหลี่ยมใน 2 มิติ และวัตถุทรงหลายหน้าใน 3 มิติด้วยนิ้ว 2 นิ้ว โดยนิ้วที่ใช้ในการจับใน 2 มิติถูกกำหนดให้เป็นนิ้วแข็ง ในขณะที่นิ้วที่ใช้ในการจับใน 3 มิติถูกกำหนดให้เป็นนิ้วอ่อน ความเสียดทานที่จุดสัมผัสระหว่างนิ้วกับวัตถุเป็นไปตามกฎของคูลอมบ์ (Coulomb friction) วิทยานิพนธ์นี้นำเสนอระเบียบวิธีที่จะคำนวณหาบริเวณสัมผัสอิสระและการวางแผนการจับที่มีบริเวณสัมผัสอิสระที่กว้างที่สุด

แนวคิดหลักของวิธีที่นำเสนอนี้คือ ในกรณี 2 มิติ ตำแหน่งของแต่ละนิ้วจะถูกอธิบายด้วยระยะบนเส้นขอบของรูปหลายเหลี่ยมจากจุดอ้างอิงจุดหนึ่งบนเส้นขอบนั้น นิ้วหนึ่งนิ้วจึงสามารถถูกอธิบายด้วยตัวแปร 1 มิติ ดังนั้นปัญหาการวางแผนการจับด้วย 2 นิ้วจึงถูกแก้ในปริภูมิ 2 มิติของตัวแปรระยะทาง เมื่อเราพิจารณาถึงปริภูมินี้จะเห็นว่าบริเวณสัมผัสอิสระสามารถถูกอธิบายด้วยรูปสี่เหลี่ยมที่มีด้านขนานกับแกนของปริภูมิ ปัญหาการคำนวณบริเวณสัมผัสอิสระก็คือการคำนวณหารูปสี่เหลี่ยมที่ทุกจุดในพื้นที่แทนการจับแบบปิดของแรงทั้งหมด และปัญหาการวางแผนการจับที่มีบริเวณสัมผัสอิสระที่กว้างที่สุดก็คือการคำนวณหารูปสี่เหลี่ยมใหญ่สุดที่ทุกจุดในพื้นที่แทนการจับแบบปิดของแรงทั้งหมดนั่นเอง

สำหรับในกรณี 3 มิติการแก้ปัญหาจะอยู่ในปริภูมิของวัตถุ ผิวหน้าของวัตถุจะถูกอธิบายด้วยโครงข่ายของรูปหลายเหลี่ยม (polygonal mesh) โดยสมมติว่าโครงข่ายนี้มีความละเอียดพอที่จะอธิบายตัววัตถุและรูปหลายเหลี่ยมที่อธิบายผิวหน้าของวัตถุมีพื้นที่ใกล้เคียงกันมาก การคำนวณบริเวณสัมผัสอิสระจะเริ่มจากคำนวณการจับด้วย 2 นิ้วบนรูปหลายเหลี่ยม 2 รูป หลังจากนั้นจะทำการหาการจับ 2 นิ้วจากรูปหลายเหลี่ยมรอบข้างแพร่ขยายไปเรื่อยๆ โดยรูปหลายเหลี่ยมที่เคยอยู่ในบริเวณสัมผัสอิสระจะต้องคงอยู่ในบริเวณนี้เสมอ ดังนั้นปัญหาการวางแผนการจับที่มีบริเวณสัมผัสอิสระที่กว้างที่สุดก็คือการคำนวณหาการจับที่สามารถแพร่บริเวณไปยังรูปหลายเหลี่ยมรอบข้างแล้วกินพื้นที่มากที่สุด

เนื่องจากวิทยาพนธ์นี้ให้ความสำคัญกับวัตถุที่มีความละเอียดสูงนั่นคือรูปหลายเหลี่ยมที่ประกอบด้วยด้านเล็กๆ จำนวนมาก และทรงหลายเหลี่ยมที่ประกอบด้วยรูปหลายเหลี่ยมเล็กๆ จำนวนมาก แต่ระเบียบวิธีในการแก้ปัญหาในปัจจุบันสามารถแก้ปัญหาการวางแผนการจับที่มีบริเวณสัมผัสอิสระที่กว้างที่สุดได้เพียงบนด้านหรือรูปหลายเหลี่ยมเดียว ไม่สามารถที่จะหาคำตอบที่ดีที่สุดข้ามด้านหรือรูปหลายเหลี่ยมรอบข้างได้ ผลที่ได้จึงมีคุณภาพห่างจากความเป็นจริง วิทยานิพนธ์ฉบับนี้จึงเสนอวิธีสำหรับคำนวณหาบริเวณสัมผัสอิสระและวางแผนการจับที่มีบริเวณสัมผัสอิสระที่กว้างที่สุดของการจับวัตถุรูปหลายเหลี่ยมหรือทรงหลายหน้าด้วยนิ้วจับจำนวน 2 นิ้วที่ให้คำตอบข้ามด้านของรูปหลายเหลี่ยมหรือข้ามรูปหลายเหลี่ยมที่อธิบายทรงหลายหน้าได้

เนื้อหาของวิทยานิพนธ์ฉบับนี้ถูกแบ่งออกเป็น 4 บทดังนี้ คือ บทที่ 1 เป็นบทนำบทที่ 2 จะเป็นงานวิจัยและทฤษฎีที่เกี่ยวข้อง บทที่ 3 จะเป็นการทดลองและผลการทดลองโดยแบ่งออกเป็น 2 ส่วนคือการหาบริเวณสัมผัสอิสระบน 2 มิติ และบน 3 มิติ และ บทที่ 4 จะเป็นเรื่องการสรุปงานวิจัย ข้อเสนอแนะ และงานวิจัยในอนาคต

บทที่ 2

งานวิจัยและทฤษฎีที่เกี่ยวข้อง

2.1 งานวิจัยที่เกี่ยวข้อง

งานวิจัยเกี่ยวกับงานจับเป็นงานวิจัยที่มีความสำคัญควบคู่มากับงานวิจัยทางด้านวิทยาการหุ่นบนต์ องค์ประกอบที่สำคัญที่สุดคือมือหุ่นยนต์ สืบเนื่องจากความต้องการในการทำงานที่ต่างกัน ทำให้มือหุ่นยนต์ถูกออกแบบมาในลักษณะที่ต่างๆ กัน มือหุ่นยนต์ที่เรียบง่ายที่สุดคือมือที่มี 2 นิ้วอย่างเช่นแขนคาตานะที่ถูกนำมาใช้งานกับหุ่นยนต์เคลื่อนที่สแตร์ [1] มือหุ่นยนต์ที่ถูกนำมาใช้ในงานวิจัยอย่างแพร่หลายคือมือที่ประกอบด้วย 3 นิ้วที่ชื่อว่ามือแบร์เรตต์ [2] ประกอบด้วยที่คล้ายนิ้วโป้ง และอีกสองนิ้วที่เหลือที่สามารถแยกออกจากกันได้ถึง 180 องศารอบอุ้งมือ มือหุ่นยนต์ที่มีความซับซ้อนและถูกมาทำงานโดยเฉพาะเจาะจงก็มีการพัฒนามาควบคู่ด้วยเช่นกัน มือยูทาห์และเอ็มไอที [3] เป็นมือที่มีสี่นิ้ว แต่มีขนาดใหญ่เนื่องจากตำแหน่งของตัวกำเนิดแรงของนิ้ว (actuator) อยู่ภายนอกมือหุ่นยนต์ มือโรโบนอท [4] เป็นมือที่มีห้านิ้ว ถูกพัฒนาขึ้นเพื่อการใช้งานในอวกาศ ตำแหน่งของตัวกำเนิดแรงของนิ้วอยู่ภายนอกเช่นเดียวกับมือยูทาห์และเอ็มไอทีแต่มีขนาดเล็กกว่า มือดีแอลอาร์สอง [5] เป็นมือรุ่นที่สองของมือดีแอลอาร์ มีนิ้วจำนวนสี่นิ้ว ใช้มอเตอร์ซึ่งอยู่ภายในมือเป็นตัวกำเนิดแรงให้กับนิ้ว ทำให้ขนาดของมือหุ่นยนต์นั้นไม่ใหญ่มาก แต่น้ำหนักตรงส่วนมือสูง และโครงสร้างภายในนิ้วมีความซับซ้อน

ในการวางแผนการจับนั้น คุณสมบัติแบบปิดของแรงมักจะเป็นเงื่อนไขที่จำเป็นในการคำนวณหาการจับ (อย่างน้อยการจับใดๆต้องเป็นไปตามเงื่อนไขนี้) สำหรับวัตถุที่จะทำการจับชิ้นใดๆ ก็มักจะมีการจับที่เป็นไปได้หลายคำตอบ เป้าหมายของการวางแผนการจับก็คือการหาการจับที่ดีที่สุดจากการจับเหล่านั้น เงื่อนไขสำหรับ ตัดสินใจว่าการจับใดเป็นการจับที่ดีที่สุดก็มีอยู่หลายเงื่อนไขที่เป็นที่นิยมใช้กัน อย่างเช่นการทดสอบว่าการจับทนต่อแรง (force) หรือแรงบิด (torque) ที่กระทำจากภายนอกได้มากน้อยเท่าใดจึงจะทำให้การจับเสียคุณสมบัติแบบปิดของแรง แรงหรือแรงบิดที่เล็กที่สุดที่พอสำหรับทำให้การจับเสียคุณสมบัติแบบปิดของแรงนั้นจะมีขนาดเท่ากับรัศมีของทรงกลมที่ใหญ่ที่สุดที่สามารถใส่ลงไปในคอนเวกซ์ฮัลล์ที่เกิดจากการจับในมิติของแรงและแรงบิด ซึ่ง [6] เป็นผู้เสนอขึ้นมา โดยเรียกวิธีการประเมินนี้ว่า คุณภาพของการจับ (grasp efficiency) วิธีการนี้ก็ได้ถูกใช้โดย [7] ด้วย ในงานวางแผนการจับหลายๆงาน [8-10] ก็มีการใช้การวัด คุณภาพของท่าจับร่วมในงานด้วย เมื่อไม่นานมานี้ [11] ก็ได้เสนอหลักการที่คล้ายๆกัน

เรียกว่า ระยะทางคิว (Q distance) ซึ่งมีความแตกต่างจากวิธีของ Kirkpatrick คือใช้ฟังก์ชั่นที่เป็นมาตราส่วนของโพลีโทปในการวัดระยะ แทนที่จะใช้ระบบยูคลิดแบบปรกติ ทำให้สามารถเปลี่ยนมาตรวัดได้สอดคล้องกับงานที่ต้องการให้หุ่นยนต์ทำได้ การวางแผนการจับตามที่ได้กล่าวไปข้างต้นสามารถให้คำตอบเป็นท่าจับที่ดีเท่านั้น ส่วนการจับจริง เป็นหน้าที่ของแขนกลซึ่งเป็นตัวที่ต้องวางนิ้วจับลงบนจุดจับที่คำนวณได้มาให้แม่นยำเอาเอง ท่าจับที่ดี ทนแรงกระทำได้ดี จะไม่มีประโยชน์เลยถ้าในทางปฏิบัติ ถ้าการจับที่วางนิ้วจับคลาดเคลื่อนจากจุดจับที่ต้องการเพียงเล็กน้อยก็สามารถทำให้การจับครั้งนั้นล้มเหลว การศึกษาวิจัยในหัวข้อที่เกี่ยวกับความคลาดเคลื่อนจึงเป็นหัวข้อที่สำคัญเช่นกัน [12-17]

เพื่อที่จะให้คำตอบเป็นท่าจับที่ทนต่อความผิดพลาดในการวางนิ้วจับ Nguyen จึงได้เสนอหลักการ เรื่องบริเวณสัมผัสอิสระใน [18] โดยนิยามว่า บริเวณสัมผัสอิสระคือบริเวณบนผิววัตถุที่นิ้วจับแต่ละนิ้วสามารถ เคลื่อนที่ไปยังจุดใดในบริเวณนั้นก็ได้และจะยังทำให้การจับมีคุณสมบัติแบบปิดของแรงเสมอ ใน [18] Nguyen ได้แสดงวิธีทางเรขาคณิตสำหรับคำนวณหาบริเวณสัมผัสอิสระที่ใหญ่ที่สุดสำหรับการจับวัตถุรูปหลายเหลี่ยมด้วยนิ้วจับที่มีแรงเสียดทางจำนวนสองนิ้ว โดยคำตอบจะอยู่บนคู่ด้านของรูปหลายเหลี่ยม แต่ก็ยังมีข้อผิดพลาดในการ คำนวณด้านบางแบบ ซึ่งปัญหานี้ก็ได้ถูกระบุและเสนอวิธีแก้ไขใน [19] โดย Tung และ Kak เงื่อนไขที่ใช้ระบุว่าบริเวณสัมผัสอิสระใดเป็นคำตอบที่ดีนั้น จะดูจากบริเวณสัมผัสอิสระที่เล็กที่สุดของแต่ละการจับ (การจับที่บริเวณสัมผัสอิสระขนาดเล็กที่สุดมีขนาดใหญ่ที่สุด)

สำหรับการจับวัตถุสองมิติด้วยนิ้วที่มีแรงเสียดทานจำนวน 3 นิ้ว Ponce et al. ได้เสนอวิธีการ คำนวณหาบริเวณสัมผัสอิสระที่ดีที่สุดที่อยู่บนด้าน 3 ด้านของรูปหลายเหลี่ยมไว้ใน [20] โดยใช้แนวคิดของคุณสมบัติสมดุลที่ไม่รวมขอบ (non-marginal equilibrium) และเงื่อนไขของคุณสมบัติปิดของแรงเพื่อแปลงปัญหาเป็นการหาคำตอบที่ดีที่สุด จากการแก้ปัญหากำหนดการเชิงเส้น (linear programming) แต่เนื่องจากเงื่อนไขของคุณสมบัติปิดของแรงที่ใช้ในงานนี้ยังเป็นเงื่อนไขที่พอเพียง จึงทำให้การจับบางการจับไม่ถูกรวมเข้ามาในการคำนวณ ทำให้คำตอบที่ได้อาจจะแย่กว่าคำตอบที่ดีที่สุดจริงๆ ไม่นานมานี้ Cornella และ Suarez ก็ได้เสนอวิธีสำหรับคำนวณหาบริ เวณสัมผัสอิสระโดยไม่จำกัดจำนวนนิ้ว วิธีที่เสนอนั้นทำการคำนวนบนผิวของวัตถุโดยมองเป็นผิว 2 มิติ และทำการคำนวณบนนั้นเพื่อเพิ่มประสิทธิภาพ แต่วิธีนี้ก็ไม่สามารถรับประกันความว่าจะให้คำตอบที่ดีที่สุดได้ ถึงแม้ว่าปัญหาการจับวัตถุรูปหลายเหลี่ยมด้วยนิ้วจับ 2 นิ้วจะเป็นหนึ่งในปัญหาพื้นฐานของเรื่องการจับ แต่บางจุดของปัญหาก็ยังไม่ได้รับการแก้ไขและยังไม่ได้ถูกนำไปใช้ให้เกิดประโยชน์สูงสุด ตัวอย่างของการนำปัญหาการจับวัตถุ 2 มิติด้วยนิ้วจับ

2 นิ้วไปใช้จริงได้ถูกแสดงให้เห็นใน [21] ซึ่งเป็นปัญหาการใช้ ภาพจากกล้องเพื่อนำไปเป็นข้อมูลสำหรับการเคลื่อนย้ายวัตถุในโรงงานอุตสาหกรรม ที่จริงแล้วปัญหาใหม่ๆ ในเรื่องการจับก็ใช้เรื่องการคำนวณการจับวัตถุรูปหลายเหลี่ยมด้วยนิ้ว 2 นิ้วเป็นส่วนประกอบด้วย[12,21-23] เรื่องบริเวณสัมผัสอิสระได้ถูกพัฒนาขึ้นเพื่อให้การจับที่คำนวณได้นั้นมีความทนต่อความผิดพลาดได้มากขึ้น ถึงแม้ว่าเรื่องนี้จะได้ถูกเสนอมาเป็นสิบๆ ปีแล้วก้ตาม แต่ปัญหาที่น่าสนใจก็ยังคงมีอยู่ วิธีคำนวณที่มีอยู่ในปัจจุบัน[19,20] ก็ยังมีข้อจำกัดที่คำตอบของบริเวณสัมผัสอิสระไม่สามารถที่จะข้ามด้านของรูปหลายเหลี่ยมได้ ข้อจำกัดนี้อาจเป็นตัวทำให้คำตอบที่ได้ออกมานั้นมีขนาดที่เล็กกว่าความเป็นจริงไปมาก จะเห็นได้ชัดในกรณีที่ด้านของรูปหลายเหลี่ยมมีขนาดเล็ก ตัวอย่างเช่น การนำรูปหลายเหลี่ยมมาใช้ประมาณวัตถุผิวโค้ง ซึ่งจะให้ผลออกมาเป็นรูปหลายเหลี่ยมที่เป็นด้านเล็กๆ จำนวนมากในจุดที่เป็นผิวโค้ง

แนวคิดการคำนวณหาบริเวณสัมผัสอิสระที่สามารถข้ามด้วยของรูปหลายเหลี่ยมได้นั้น ก็มีปรากฏใน [22,23] แต่ในงานที่กล่าวถึงก็มิได้เป็นการหาบริเวณสัมผัสอิสระที่ดีที่สุด แต่เป็นเพียงการหาบริเวณสัมผัส อิสระโดยขยายคำตอบจากท่าจับเริ่มต้นใดๆ งานหลายชิ้นก็ได้เสนอการหาท่าจับบนวัตถุผิวโค้ง แต่บางงาน ก็เพียงแต่พุ่งเป้าไปที่การหาการจับแบบปิดของแรงเท่านั้น [24-26] แต่งานของ Ponce et al.[27] ซึ่งเป็นงานที่พยายามหาบริเวณสัมผัสอิสระโดยใช้ระเบียบวิธีเชิงตัวเลขเพื่อหาคำตอบที่ดีที่สุด แต่คำตอบที่ได้มานั้นก็ไม่สามารถรับประกันได้ว่าเป็นบริเวณสัมผัสอิสระที่ดีที่สุด

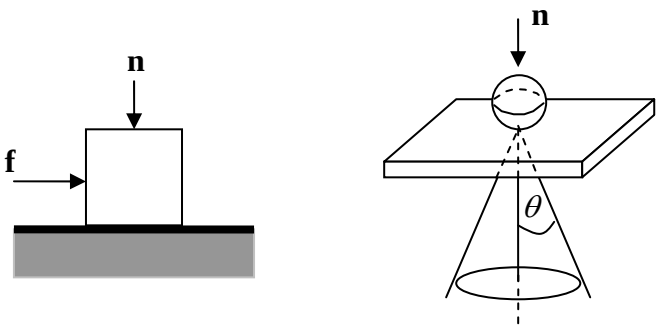
2.2 ทฤษฎีที่เกี่ยวข้อง

ความรู้พื้นฐานที่จำเป็นต้องกล่าวถึงในวิทยานิพนธ์นี้แบ่งเป็นสามส่วนคือ 2.2.1 เป็นเรื่องการจับและความเสียดทาน เนื้อหาในส่วนนี้จะกล่าวถึงความสัมพันธ์ระหว่างความเสียดทานและแรงกระทำจากนิ้ว หัวข้อ 2.2.2 กล่าวถึงสภาพสมดุลของการจับ การจับที่มีคุณสมบัติแบบปิดของแรง และความสัมพันธ์ระหว่างสภาพสมดุลของการจับและคุณสมบัติแบบปิดของแรง โดยส่วนสำคัญคือข้อสรุปที่ว่า สภาพสมดุลของการจับเป็นเงื่อนไขที่เพียงพอสำหรับคุณสมบัติแบบปิดของแรง ข้อสรุปนี้ทำให้เราสามารถหาการจับที่มีคุณสมบัติแบบปิดของแรงได้จากการคำนวณหาการจับที่ทำให้เกิดสภาพสมดุลของการจับ และหัวข้อที่ 2.2.3 จะกล่าวถึงเงื่อนไขในการจับวัตถุหลายเหลี่ยมในสภาพสมดุลด้วยสองและสามนิ้วตามลำดับ

2.2.1 การจับและความเสียดทาน

การใช้นิ้วจับนับเป็นหัวใจสำคัญของการจัดวัตถุด้วยมือหุ่นยนต์ นิ้วหุ่นยนต์แบ่งเป็นสองประเภทคือ นิ้วแข็ง (hard finger) และนิ้วอ่อน (soft finger) นิ้วอ่อนแตกต่างจากนิ้วแข็งด้วยความสามารถในการถ่ายทอดแรงบิดรอบจุดสัมผัส ในขณะที่นิ้วแข็งสามารถออกแรงกระทำที่จุดสัมผัสได้เพียงอย่างเดียวเท่านั้น การวิเคราะห์การจับด้วยนิ้วแข็งแบ่งตามลักษณะของผิวสัมผัสเป็นสองประเภทคือ แบบไม่มีแรงเสียดทานและแบบมีแรงเสียดทาน เมื่อไม่มีแรงเสียดทาน นิ้วออกแรงกระทำกับวัตถุได้ในเฉพาะทิศทางตั้งฉากกับผิวสัมผัส แต่เมื่อมีแรงเสียดทานเข้ามาเกี่ยวข้อง แรงกระทำจะถูกกำหนดด้วยธรรมชาติของแรงเสียดทาน แรงเสียดทานเป็นปรากฏการณ์ทางฟิสิกส์ที่ซับซ้อนและเกี่ยวข้องกับโครงสร้างทางวัสดุของผิวสัมผัสทั้งในระดับจุลภาค (microscale) และระดับมหภาค (macroscale) ถึงแม้ว่าพฤติกรรมโดยละเอียดของแรงเสียดทานเป็นปัญหาที่ยังต้องรอการค้นคว้าจากนักวิทยาศาสตร์ การใช้กฎของคูลอมบ์ [28] ในการอธิบายแรงเสียดทานก็เป็นที่ยอมรับว่าแม่นยำพอเพียงในทางกลศาสตร์และงานทฤษฎีเกี่ยวกับการจับ ดังที่ได้กล่าวไปแล้วในบทนำว่าการจับที่มีเสถียรภาพนั้น นิ้วจะต้องอยู่นิ่งเมื่อเทียบกับวัตถุ นั่นก็คือผิวสัมผัสระหว่างนิ้วกับวัตถุไม่เคลื่อนที่ จากกลศาสตร์เบื้องต้น [29] แรงเสียดทานที่เกี่ยวข้องในกรณีนี้คือแรงเสียดทานสถิต (static friction) แรงเสียดทานสถิตเป็นแรงต้านทานความพยายามที่จะเคลื่อนที่ของวัตถุ วัตถุที่อยู่นิ่งจะเริ่มเคลื่อนที่ได้เมื่อมีแรงกระทำในทิศทางการเคลื่อนที่ซึ่งมีขนาดไม่น้อยกว่าขนาดของแรงเสียดทานสถิตสูงสุดตามกฎของคูลอมบ์ ขนาดของแรงเสียดทานสถิตสูงสุดนี้สามารถคำนวณได้จากผลคูณระหว่างขนาดของแรงกดในแนวตั้งฉากกับผิวสัมผัสและค่าสัมประสิทธิ์แรงเสียดทานสถิต (static friction coefficient) ซึ่งเป็นค่าคงตัวขึ้นอยู่กับคุณสมบัติของผิวสัมผัสทั้งสอง ดังตัวอย่างในรูปที่ 1(a) ซึ่งแสดงวัตถุรูปลูกบาศก์ที่อยู่นิ่งบนพื้นเรียบและมีแรงกดคือ $\mathbf{n}$ เมื่อสัมประสิทธิ์แรงเสียดทานสถิตของผิวสัมผัสทั้งสองเป็น μ จะได้ว่าแรงผลัก $\mathbf{f}$ ที่มีขนาดน้อยกว่า $|\mu\mathbf{n}|$ ไม่ว่าจะมาจากทิศทางใดก็ตามก็ไม่สามารถทำให้วัตถุนี้เคลื่อนที่ไปได้ ข้อสรุปจากตัวอย่างนี้สามารถนำมาใช้ในการพิจารณาแรงที่นิ้วกระทำต่อวัตถุในขณะที่วัตถุถูกจับให้อยู่นิ่งได้ รูปที่ 1(b) แสดงนิ้วทรงกลมที่สัมผัสกับผิววัตถุ โดยเมื่อนิ้วออกแรงกด $\mathbf{n}$ ในแนวตั้งฉากกับผิวสัมผัส นิ้วก็สามารถออกแรงที่มีขนาดน้อยกว่า $|\mu\mathbf{n}|$ ในทิศทางใดก็ได้ที่ขนานกับผิวสัมผัสโดยไม่ทำให้เกิดการเลื่อนของนิ้วบนผิวสัมผัส เมื่อพิจารณาแรงในแนวตั้งฉากและแนวขนานกับผิวสัมผัส เราจึงเห็นได้ว่าแรงรวมที่นิ้วกระทำต่อวัตถุจะต้องมีแนวแรงอยู่ในกรวยซึ่งมีจุดสัมผัสเป็นจุดยอด มีแกนกลางในทิศทางที่ตั้งฉากกับผิวสัมผัส และมีครึ่งหนึ่งของมุมที่จุดยอด (half cone angle) เป็น $\theta = \tan^{-1}(\mu)$ เราเรียกกรวยดังกล่าวนี้ว่ากรวยเสียดทาน (friction cone) และสามารถเขียนได้ว่า

ทฤษฎีบท 1 เมื่อไม่มีการเคลื่อนที่ระหว่างผิวสัมผัส แรงรวมที่นิ้วกระทำต่อวัตถุวางตัวอยู่ภายในกรวยเสียดทานที่จุดสัมผัส



(a) (b)

รูปที่ 1 แรงเสียดทานตามกฎของคูลอมบ์

2.2.2 สภาพสมดุลของการจับและคุณสมบัติแบบปิดของแรง

ในทางกลศาสตร์ วัตถุอยู่นิ่งย่อมอยู่ในสภาพสมดุล วัตถุอยู่ในสภาพสมดุลก็ต่อเมื่อแรงรวมและแรงบิดรวมที่กระทำต่อวัตถุเป็นศูนย์ วัตถุอยู่ในสภาพสมดุลของการจับ (grasping equilibrium) เมื่อวัตถุอยู่ในสภาพสมดุลภายใต้แรงและแรงบิดจากนิ้ว โดยไม่มีการรบกวนจากแรงและแรงบิดอื่นๆ ดังที่ได้กล่าวไปแล้วว่า นิ้วแข็งซึ่งเราศึกษาในวิทยานิพนธ์นี้ออกแรงได้เพียงอย่างเดียว นั่นก็คือแรงบิดที่ต้องพิจารณาในการวิเคราะห์สภาพสมดุลของการจับ มีเพียงแรงบิดที่ได้จากการคำนวณโมเมนต์ (moment) ของแรงกระทำจากนิ้วรอบจุดกำเนิด การคำนวณโมเมนต์นี้คือการหาผลคูณไขว้ $\mathbf{x}\times\mathbf{f}$ โดยที่ $\mathbf{x}=(x_1, x_2, x_3)^T$ เป็นเวกเตอร์บอกตำแหน่งจุดสัมผัส และ $\mathbf{f}=(f_1, f_2, f_3)^T$ เป็นแรงกระทำจากนิ้ว ซึ่งในกรณีที่บริเวณทำงานมีสองมิติ แรง $\mathbf{f}$ และตำแหน่ง $\mathbf{x}$ ใดๆ จะอยู่บนระนาบเดียวกัน ($x_3=0$ และ $f_3=0$) ทำให้ได้โมเมนต์อยู่ในรูป $(0,0,x_1f_2-f_1x_2)^T$ ก่อนที่เราจะกล่าวถึงคุณสมบัติแบบปิดของแรง ขอนิยามสภาพสมดุลของการจับอย่างชัดเจนสำหรับการอ้างอิงถึงภายหลังดังนี้

บทนิยาม 1 สภาพสมดุลของการจับด้วยนิ้วแข็งคือ สภาพสมดุลที่เกิดจากแรงกระทำจากนิ้วเท่านั้น (ไม่มีแรงและแรงบิดอื่นๆ เกี่ยวข้อง) โดยต้องมีอย่างน้อยหนึ่งนิ้วที่ออกแรง

เมื่อพิจารณาอย่างผิวเผิน จะเห็นได้ว่าการจับที่ทำให้วัตถุอยู่ในสภาพสมดุลของการจับไม่เพียงพอ หากมีแรงหรือแรงบิดภายนอกรบกวน การจับดังกล่าวก็ไม่น่าที่จะสามารถออกแรงเพื่อ

หักล้างการรบกวนได้ในทุกกรณี ด้วยเหตุนี้จึงได้มีการนิยามคุณสมบัติที่พึงประสงค์ของการจับที่เรียกว่าคุณสมบัติแบบปิดของแรง กล่าวคือ

บทนิยาม 2 เมื่อขนาดของแรงที่นิ้วกระทำต่อวัตถุไม่ถูกจำกัด การจับที่มีคุณสมบัติแบบปิดของแรงจะสามารถออกแรงหักล้างแรงหรือแรงบิดภายนอกใดๆ ที่มารบกวนได้ ทำให้วัตถุอยู่ในสภาพสมดุล ไม่ว่าจะมีการรบกวนใดๆ ก็ตาม

บทนิยาม 2 ชี้ชัดว่าการจับที่มีคุณสมบัติแบบปิดของแรงย่อมสามารถทำให้วัตถุอยู่ในสภาพสมดุลของการจับได้ด้วย (แรงและแรงบิดรบกวนเป็นศูนย์) แต่ที่น่าสนใจกว่าคือบทกลับของข้อสรุปนี้ ดังที่ได้มีการพิสูจน์ไว้ใน [18] และ [20] ว่า

ทฤษฎีบท 2 เมื่อผิวสัมผัสมีความเสียดทาน การจับด้วยนิ้วแข็งซึ่งแรงกระทำจากแต่ละนิ้วอยู่ภายใน (interior) กรวยเสียดทานอย่างเคร่งครัด และสามารถทำให้วัตถุอยู่ในสภาพสมดุลของการจับ ย่อมเป็นการจับที่มีคุณสมบัติแบบปิดของแรงด้วยเสมอ

ทฤษฎีบท 2 นี้ทำให้เราสามารถหาการจับที่มีคุณสมบัติแบบปิดของแรงได้ด้วยการคำนวณตำแหน่งจุดสัมผัส ของการจับที่ทำให้เกิดสภาพสมดุลของการจับได้ โดยในทางปฏิบัติการคำนวณนี้อาจใช้สัมประสิทธิ์ความเสียดทานสถิตที่น้อยกว่าค่าจริง เพื่อให้แน่ใจว่าแรงกระทำจากนิ้วจะอยู่ในกรวยเสียดทานจริงตามที่กำหนดไว้ในทฤษฎี ข้อกำหนดของแรงดังกล่าวเป็นส่วนจำเป็นในการพิสูจน์ทฤษฎีใน [20] โดยในบทความนี้ ได้มีการแยกประเภทให้กับสภาพสมดุลตามทฤษฎีบท 2 โดยเรียกว่าสภาพสมดุลที่ไม่ใช่ขอบ (non-marginal equilibrium) อย่างไรก็ตาม วิทยานิพนธ์นี้ไม่มีความจำเป็นต้องแยกประเภทของสภาพสมดุล เนื่องจากแรงกระทำจากนิ้วได้รับการรับประกันจากทฤษฎีบท 1 แล้วว่าต้องอยู่ภายในกรวยเสียดทาน

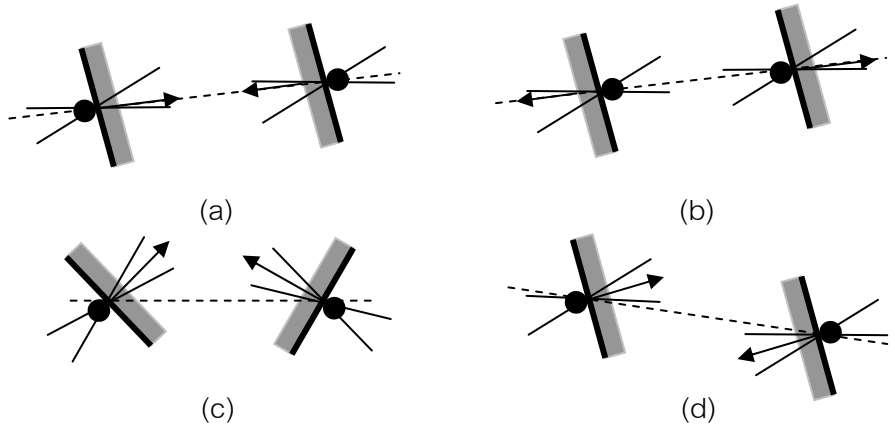
2.2.3 เงื่อนไขในการจับวัตถุหลายเหลี่ยมในสภาพสมดุล

หัวข้อนี้จะขอกล่าวถึงลักษณะของการจับด้วยสองและสามนิ้วเพื่อให้วัตถุอยู่ในสภาพสมดุลของการจับ โดยใช้เงื่อนไขของแนวแรงและตำแหน่งจุดสัมผัส ตลอดหัวข้อนี้ เรากำหนดให้ θ แทนครึ่งหนึ่งของมุมที่จุดยอดกรวยเสียดทาน นอกจากนี้จะขอใช้คำว่ากรวยเสียดทานภายใน (internal friction cone) เพื่อเรียกกรวยเสียดทานที่มีทิศพุ่งเข้าหาเนื้อวัตถุ(ตามทิศทางแรงกระทำจากนิ้ว) และเมื่อพิจารณาเส้นตรงทุกเส้นที่ขนานกับเวกเตอร์ในกรวยเสียดทานภายใน เราจะได้กรวยสองชิ้นที่วางตัวตรงข้ามกันที่จุดสัมผัส (รูปที่ 2) ขอเรียกกรวยทั้งสองชิ้นรวมกันว่ากรวยเสียดทานสองด้าน (double-sided friction cone)

ทฤษฎีบท 3 ต่อไปนี้นำมาจาก [18] ซึ่งกล่าวถึงลักษณะของการจับด้วยนิ้วจำนวน 2 นิ้ว เพื่อให้อยู่ในสภาพสมดุลของการจับ

ทฤษฎีบท 3 เงื่อนไขจำเป็นและเพียงพอสำหรับจุดจับสองจุดเพื่อให้เกิดการจับที่อยู่ในสภาพสมดุลคือ เวกเตอร์ทั้งสองที่ตั้งฉากกับด้านที่สัมผัสและมีทิศทางพุ่งเข้าหาเนื้อวัตถุทำมุมกันอยู่ในช่วง $(\pi - 2\theta, \pi + 2\theta)$ และเส้นเชื่อมระหว่างจุดจับสองจุดนั้นวางตัวอย่างเคร่งครัดอยู่ในกรวยเสียดทานสองด้านทั้งสองกรวย

ตัวอย่างของการจับด้วยนิ้วสองนิ้วซึ่งเป็นไปตามทฤษฎีบท 3 แสดงดังรูปที่ 2(a) และรูปที่ 2(b) แต่การจับดังรูปที่ 2(c) ไม่เป็นการจับที่อยู่ในสภาพสมดุลเพราะเวกเตอร์ที่ตั้งฉากกับด้านที่สัมผัสกับนิ้วทั้งสองทำมุมกันอยู่นอกช่วง $(\pi - 2\theta, \pi + 2\theta)$ และเส้นเชื่อมระหว่างจุดจับทั้งสองจุดไม่วางตัวอยู่ในกรวยเสียดทานสองด้าน และการจับดังรูปที่ 2(d) ไม่เป็นการจับที่อยู่ในสภาพสมดุลเนื่องจากเส้นเชื่อมระหว่างจุดจับทั้งสองจุดไม่วางตัวอยู่ในกรวยเสียดทานสองด้าน ถึงแม้เวกเตอร์ที่ตั้งฉากกับด้านที่สัมผัสกับนิ้วทั้งสองทำมุมกันอยู่ในช่วง $(\pi - 2\theta, \pi + 2\theta)$ ก็ตาม



รูปที่ 2 การจับด้วยนิ้วจำนวนสองนิ้ว : ลูกศรแสดงถึงแรงที่นิ้วกระทำกับวัตถุซึ่งวางตัวอยู่ในกรวยเสียดทานสองด้าน เส้นประคือเส้นเชื่อมระหว่างจุดจับทั้งสองจุด

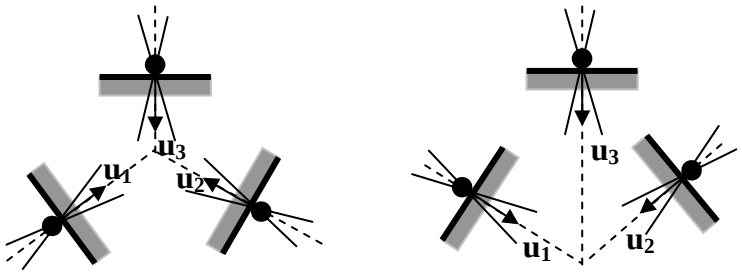
จากนี้จะขอกล่าวถึงการจับด้วยนิ้วจำนวน 3 นิ้วเพื่อให้อยู่ในสภาพสมดุลของการจับโดยอาศัยบทนิยามเพิ่มเติม ดังนี้

บทนิยาม 3 เซตของเวกเตอร์สแปนทางบวก (positively span) ใน $\Re^n$ ถ้าเวกเตอร์ใด ๆ ใน $\Re^n$ สามารถเขียนอยู่ในรูปผลบวกเชิงเส้นของเวกเตอร์ในเซตนี้ด้วยสัมประสิทธิ์ที่ไม่เป็นลบ

ทฤษฎีบทต่อไปนี้มาจาก [20]

ทฤษฎีบท 4 เงื่อนไขจำเป็นและเพียงพอสำหรับการจับด้วยนิ้วจำนวนสามนิ้วเพื่อให้อยู่ในสภาพสมดุลโดยที่แรงที่จุดสัมผัสทั้งสามไม่เป็นศูนย์ และไม่ขนานกันคือ (Pa) เมื่อมีเส้นในแต่ละกรวยเสียดทานสองด้านที่จุดสัมผัสทั้งสามตัดกัน ณ จุด ๆ หนึ่ง และ (Pb) เวกเตอร์แรงที่วางตัวอยู่ในกรวยเสียดทานภายในที่จุดสัมผัส และขนานกับเส้นทั้งสามนั้นสแปนทางบวกใน $\Re^2$

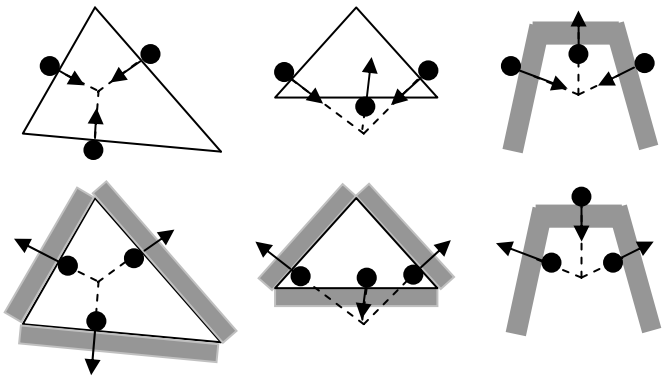
ตัวอย่างของการใช้งานทฤษฎีบท 4 แสดงในรูปที่ 3 โดยรูปที่ 3(a) มีแรง $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ ซึ่งตัดกันที่จุด ๆ หนึ่ง และสแปนทางบวกใน $\Re^2$ ดังนั้นการจับที่แสดงในรูปที่ 3(a) จึงเป็นการจับที่อยู่ในสภาพสมดุล ในขณะที่การจับที่แสดงในรูปที่ 3(b) ซึ่งแรง $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ ตัดกันที่จุด ๆ หนึ่งแต่ไม่สแปนทางบวกใน $\Re^2$ ดังนั้นจึงไม่เป็นการจับที่อยู่ในสภาพสมดุล ตัวอย่างการจับด้วย 3 นิ้วซึ่งเป็นไปตามทฤษฎีบท 4 ในแบบต่าง ๆ แสดงดังรูปที่ 4



(a) (b)

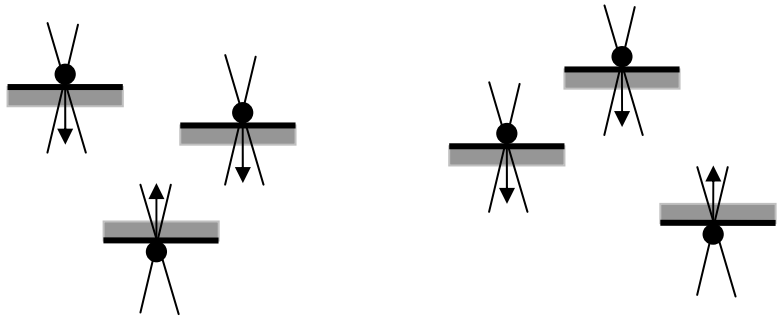
รูปที่ 3 การจับด้วยนิ้วจำนวน 3 นิ้ว

ทฤษฎีบท 5 เงื่อนไขจำเป็นและเพียงพอสำหรับการจับด้วยนิ้วจำนวนสามนิ้วเพื่อให้อยู่ในสภาพสมดุลด้วยแรงที่ขนานกันโดยที่แรงที่จุดสัมผัสทั้งสามไม่เป็นศูนย์คือ มีเส้นในกรวยเสียดทานสองด้านที่จุดสัมผัสทั้งสามขนานกัน และเวกเตอร์แรงที่ขนานกับสามเส้นนั้น (ซึ่งวางตัวอยู่ในกรวยเสียดทานภายในที่จุดสัมผัส) มีเวกเตอร์ที่ขนานกับเส้นที่อยู่กลางที่มีทิศทางตรงกันข้ามกับสองเวกเตอร์ที่เหลือ



รูปที่ 4 การจับด้วยนิ้วจำนวน 3 นิ้วในแบบต่าง ๆ ที่เป็นไปตามทฤษฎีบท 4

ตัวอย่างการจับด้วยนิ้วสามนิ้วแบบขนานซึ่งเป็นไปตามทฤษฎีบท 5 แสดงในรูปที่ 5(a) ในขณะที่การจับที่แสดงดังรูปที่ 5(b) ไม่เป็นไปตามทฤษฎีบท 5 เนื่องจากเวกเตอร์ที่มีทิศทางตรงกันข้ามกันสองเวกเตอร์ที่เหลือไม่ได้อยู่ตรงกลาง



(a) (b)

รูปที่ 5 การจับด้วยนิ้วสามนิ้วแบบขนาน

เนื่องจากการสร้างกราฟการสลับนิ้วจำเป็นต้องพิจารณาเงื่อนไขในทฤษฎีบท 6 ซึ่งเป็นเงื่อนไขที่เคร่งครัดกว่าเงื่อนไขในทฤษฎีบท 4 บทนิยามเพิ่มเติมเพื่อนำไปใช้ในทฤษฎีบท 6 เป็นดังนี้

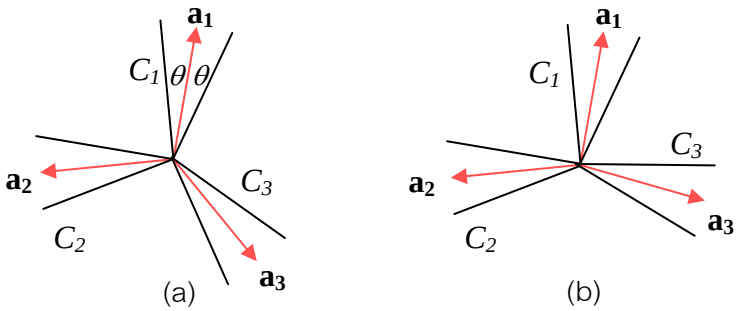
บทนิยาม 4 กำหนดให้ $C_i(i=1,2,3)$ คือกรวยซึ่งมีกึ่งกลางคือ $\mathbf{a}_i$ ด้วยครึ่งมุม θ สามารถกล่าวได้ว่าเวกเตอร์สามเวกเตอร์ $\mathbf{a}_i(i=1,2,3)$ สแปนทางบวกด้วยมุม θ (θ-positively span) ใน $\Re^2$ เมื่อสามเวกเตอร์ใด ๆ $\mathbf{b}_i \in C_i(i=1,2,3)$ ใด ๆ สแปนทางบวกใน $\Re^2$

เพื่อทำการอธิบายการสแปนทางบวกด้วยมุม θ ใน $\Re^2$ ของสามเวกเตอร์ให้พิจารณารูปที่ 6 โดยที่เวกเตอร์ $\mathbf{a}_i(i=1,2,3)$ ที่แสดงในรูปที่ 6(a) สแปนทางบวกด้วยมุม θ ในขณะที่เวกเตอร์ $\mathbf{a}_i(i=1,2,3)$ ที่แสดงในรูปที่ 6(b) ไม่สแปนทางบวกด้วยมุม θ เนื่องจากมีสามเวกเตอร์ $\mathbf{b}_i \in C_i(i=1,2,3)$ บางเวกเตอร์ที่ไม่สแปนทางบวกใน $\Re^2$ ซึ่งเห็นได้ชัดว่าเวกเตอร์สามเวกเตอร์จะสแปนทางบวกด้วยมุม θ ใน $\Re^2$ ก็ต่อเมื่อมุมระหว่างทุกเวกเตอร์มีค่าน้อยกว่า $\pi - 2\theta$

ทฤษฎีบท 6 เงื่อนไขเพียงพอสำหรับการจับด้วยนิ้วจำนวนสามนิ้วเพื่อให้อยู่ในสภาพสมดุลโดยที่แรงที่จุดสัมผัสทั้งสามไม่เป็นศูนย์ และไม่ขนานกันคือ (Pa) มีเส้นในกรวยเสียดทาน

สองด้านที่จุดสัมผัสทั้งสามตัดกัน ณ จุด ๆ หนึ่ง และ (Pc) เวกเตอร์ตั้งฉากภายใน[1] (internal normal) ที่จุดสัมผัสทั้งสามเวกเตอร์สแปนทางบวกด้วยมุม θ ใน $\Re^2$

การพิสูจน์ทฤษฎีบทนี้แสดงใน [20] ซึ่งจะสังเกตเห็นว่าการแทนเงื่อนไข Pb ในทฤษฎีบท 4 ด้วยเงื่อนไข Pc ที่เคร่งครัดกว่า ทำให้มีการจับบางคอนฟิกุเรชันเป็นไปตามทฤษฎีบท 4 แต่ไม่เป็นไปตามทฤษฎีบท 6 ถึงการจับตามทฤษฎีบท 6 จะทำให้การจับบางส่วนขาดหายไป แต่การจับส่วนที่ขาดหายไปนั้นส่วนหนึ่งคือการจับด้วยนิ้วจำนวนสองนิ้วดังแสดงใน [20] เหตุผลที่ใช้เงื่อนไขที่เคร่งครัดกว่าก็เพื่อการสร้างกราฟการสลับนิ้วอันจะกล่าวในหัวข้อที่ Error! Reference source not found.



รูปที่ 6 การสแปนทางบวกด้วยมุม θ ของเวกเตอร์สามเวกเตอร์ใน $\Re^2$

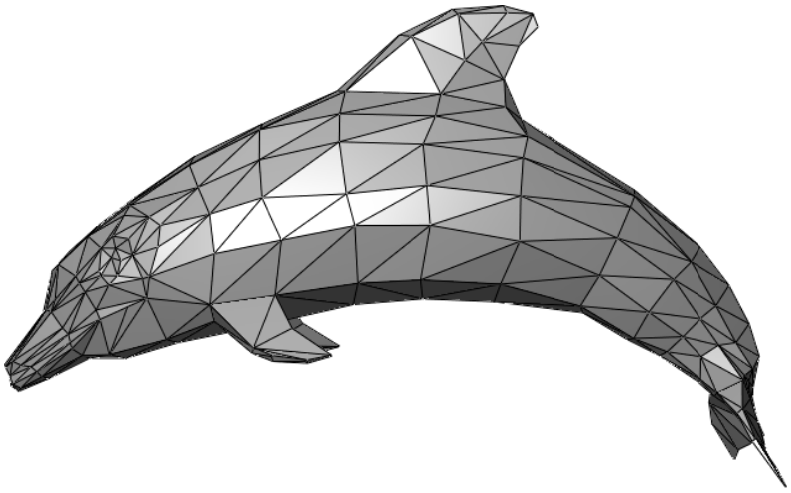
2.3 Mesh Resampling

2.3.1 Mesh

Polygon mesh คือโครงข่ายของ vertex, edge, face ที่ใช้สำหรับเป็นตัวแทนวัตถุที่เราสนใจ โดยที่ face นั้นก็คือโพลิกอนที่เกิดจากการเชื่อมต่อกัน ของ vertex และ edge ถ้า face ประกอบขึ้นจาก 3 edge เราจะเรียก face นั้นว่า triangle face, ถ้าประกอบขึ้นจาก 4 edge จะเรียกว่า quad face ส่วนข้อมูลที่เก็บอยู่บน vertex นั้น ขึ้นอยู่กับว่าต้องการจะนำไปใช้ทำอะไร ยกตัวอย่างเช่นในงานของเรานั้น ถ้าเก็บ พิกัดของจุดบนระบบ, สี และ normal ของจุดนั้นก็

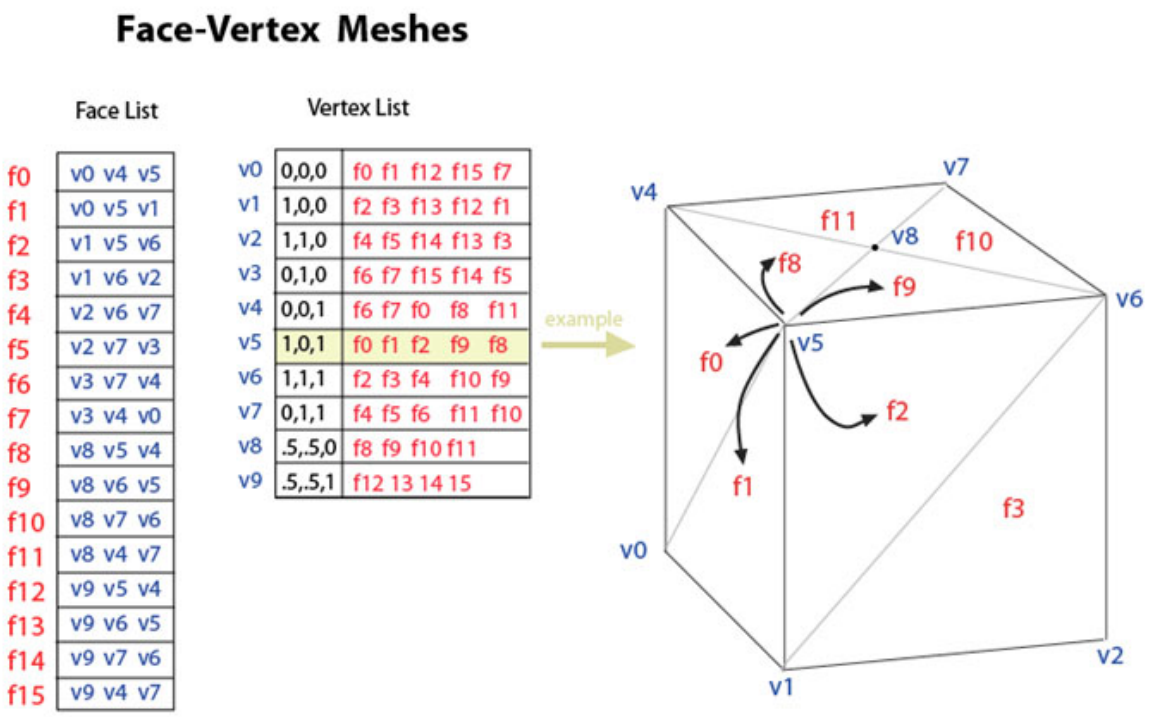
[1] เวกเตอร์ตั้งฉากที่พุ่งเข้าหาเนื้อวัตถุ

พอเพียงสำหรับการนำไปใช้งานแล้ว ส่วน edge นั้นจะเป็นเส้นเชื่อมระหว่างคู่ vertex ใดๆบนโครงข่าย



รูปที่ 7 ตัวอย่างของ Polygon Mesh

โครงสร้างการเก็บข้อมูลของ mesh สำหรับงานของเราจะใช้วิธีเก็บแบบ face-vertex คือการเก็บเซตของ vertex ที่เป็นส่วนประกอบสำหรับ face ใดๆ และสำหรับ vertex ก็จะเก็บเซตของ face ที่ vertex นั้นๆเป็นส่วนประกอบอยู่ด้วย โครงสร้างการเก็บข้อมูลลักษณะนี้สามารถทำให้การตรวจสอบว่า face ใดอยู่ติดกันหรือ vertex ใดอยู่ติดกันทำได้ง่ายและเร็ว และถ้าใช้โครงสร้างข้อมูลนี้กับ mesh ที่ประกอบขึ้นจาก triangle face หรือ quad face ประเภทใดประเภทหนึ่งเพียงประเภทเดียวก็จะสามารถประหยัดพื้นที่การเก็บข้อมูลของ mesh ได้เป็นอย่างมากอีกด้วย



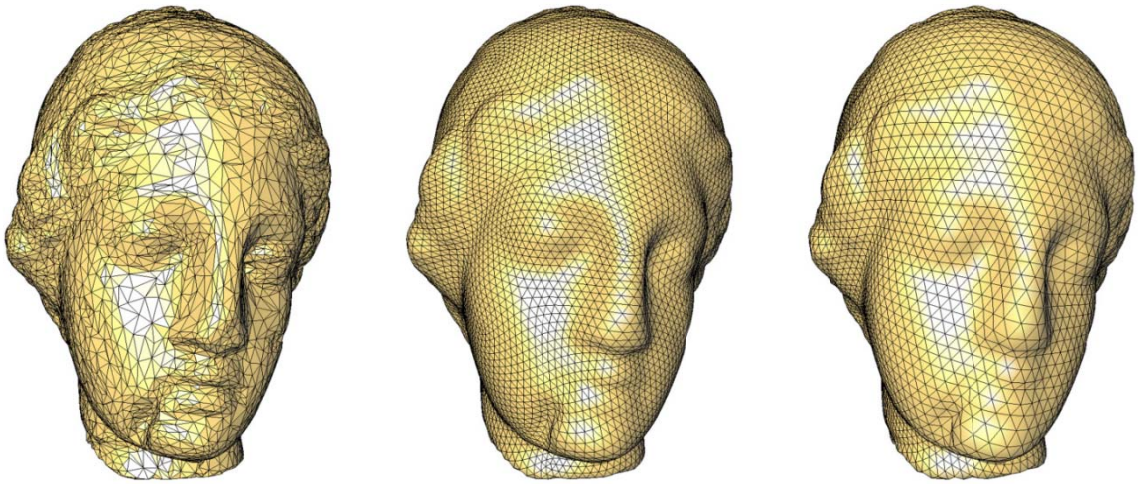
รูปที่ 8 ความสัมพันธ์ของ Face-Vertex

2.3.2 Remeshing

Remeshing คือการปรับปรุงโครงสร้างของ polygon mesh ใหม่ เพื่อให้ได้คุณสมบัติบางอย่างที่ต้องการ คุณสมบัติที่น่าสนใจหลายๆอย่างได้แก่

- รูปร่าง : เช่น ปรับให้ face เป็นสามเหลี่ยมทั้งหมดเพื่อความสะดวกในการเข้าถึงข้อมูลและประสิทธิภาพในการเก็บข้อมูล หรือปรับให้ face ส่วนมากเป็นสี่เหลี่ยมซึ่งเป็นที่นิยมใช้ในการออกแบบชิ้นส่วนหรือ reverse engineer
- ลักษณะ : เช่น ปรับให้แต่ละ face มีรูปร่างเหมือนกัน (isotropic) เช่น ถ้าเป็นสามเหลี่ยมก็จะมีลักษณะใกล้เคียงสามเหลี่ยมด้านเท่า ถ้าเป็นสี่เหลี่ยมก็จะมีลักษณะใกล้เคียงสี่เหลี่ยมจตุรัส
- ความหนาแน่น : การ remesh โดยไม่สนใจความหนาแน่นว่าในแต่ละส่วนจะต้องมีปริมาณ face พอๆกันหรือไม่จะสามารถลดปริมาณ face ที่ใช้ลงได้ โดยบริเวณที่ไม่มีความซับซ้อนเช่นเป็นพื้นราบๆก็สามารถใช้ face ชิ้นใหญ่แทนในบริเวณนั้น

- การวางตัว : บริเวณที่เป็นขอบหรือมุมของวัตถุ ถ้าทำการ remesh โดยไม่สนใจคุณสมบัตินี้ก็อาจจะทำให้ผลลัพธ์ในบริเวณขอบและมุมของวัตถุออกมาผิดพลาดได้
- ความเป็น regular : mesh จะถูกเรียกว่า completely regular ถ้า ทุกๆ vertex มี vertex ที่อยู่ติดกับมันเป็นจำนวนที่เท่ากัน เช่น mesh ที่ประกอบขึ้นจาก triangle face จะมี 6 vertex ที่อยู่ติดกับมันและ มี 4 vertex ที่อยู่ติดกับมันสำหรับ vertex ที่เป็นขอบของพื้นผิววัตถุ ส่วน quad face จะมี vertex ที่อยู่ติดกันกับ vertex ใดๆเป็นจำนวน 4 และ 3 vertex ตามลำดับ ส่วน mesh ที่ไม่สามารถรับประกันคุณสมบัตินี้ได้จะเรียกว่าเป็น Irregular

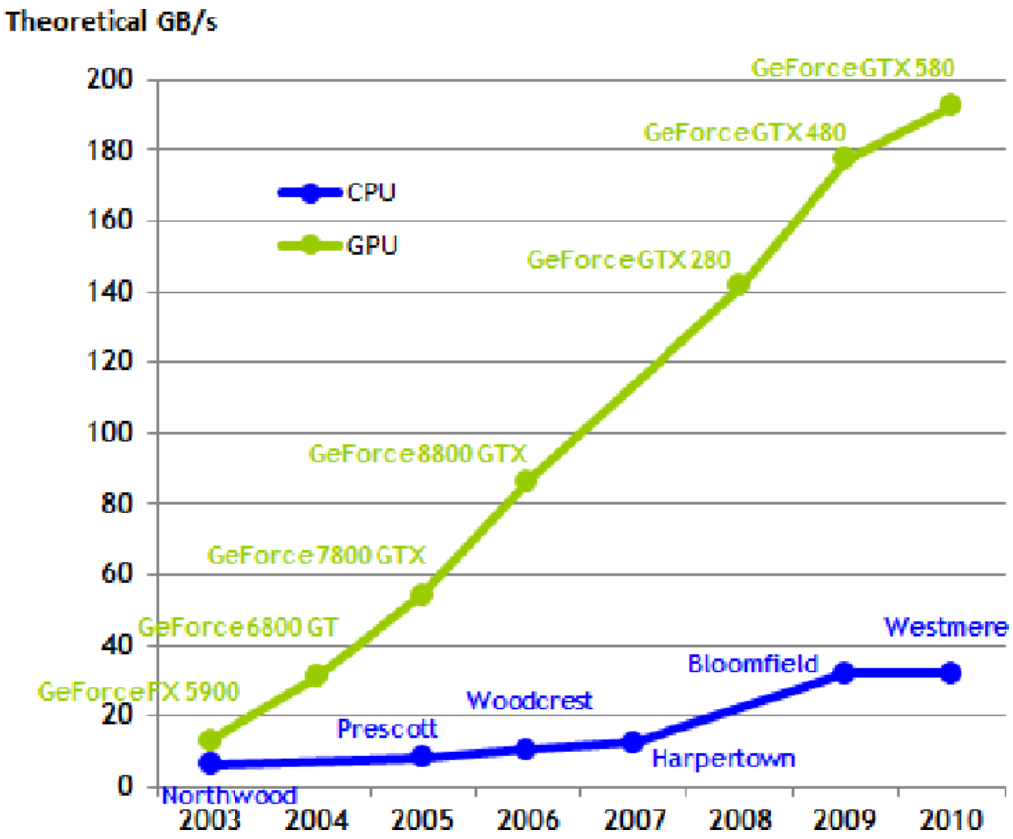


รูปที่ 9 Remeshing

2.4 Introduction to CUDA

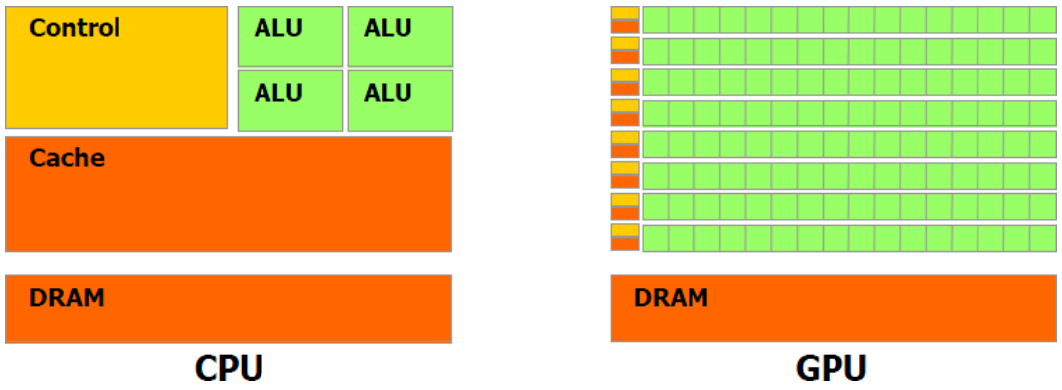
2.4.1 การใช้การ์ดแสดงผลเป็นอุปกรณ์คำนวณแบบขนาน

ในไม่กี่ปีที่ผ่านมานี้ เทคโนโลยีของการ์ดแสดงผล (graphic card) ได้ถูกพัฒนาในแง่ของพลังประมวลผลเป็นอย่างมาก (รูปที่ 10) เนื่องจากประกอบไปด้วยหน่วยประมวลผลจำนวนมากและ bandwidth สำหรับส่งข้อมูลความเร็วสูง การ์ดแสดงผลในปัจจุบันจึงเป็นอุปกรณ์ชิ้นสำคัญทั้งในการประมวลผลภาพกราฟฟิกหรือใช้ในการคำนวณประเภทอื่น



รูปที่ 10 ความเร็วของ GPU เทียบกับ CPU

เหตุผลหลักที่อยู่เบื้องหลังของการนำการ์ดแสดงผลไปใช้ในงานดังกล่าว เนื่องจาก การ์ดแสดงผลนั้นสามารถทำการคำนวณที่มีปริมาณมากๆ และการคำนวณแบบคำนวณแบบขนาน ซึ่งเป็นปัญหาสำคัญของการแสดงผลภาพบนหน้าจอ ด้วยเหตุนี้ การออกแบบของการ์ดแสดงผลจึงใช้ทรานซิสเตอร์ส่วนมากในการประมวลผลมากกว่าใช้ควบคุมการแจกจ่ายข้อมูลหรือสำรองข้อมูล



รูปที่ 11 สถาปัตยกรรมของ CPU เทียบกับ GPU

ถ้าจะกล่าวให้เฉพาะเจาะจงลงไป จุดเด่นของการ์ดแสดงผลคือสามารถที่จะจัดการกับปัญหาที่มีความขนานสูง คือ ใช้ชุดคำสั่งเดียวกัน, ทำงานพร้อมๆกัน, ในแต่ละขั้นตอนมักจะมีการคำนวณที่ซับซ้อน แต่กระทำกับข้อมูลคนละชุด และเนื่องจากเราใช้ชุดคำสั่งเดียวกัน ปัญหาการควบคุมการส่งข้อมูลจึงไม่ใช่ปัญหาใหญ่ เพราะว่าแต่ละโปรแกรมก็ใช้แต่ข้อมูลที่แตกต่างกัน ส่วนเวลาสำหรับการถ่ายโอนข้อมูลซึ่งกินเวลามากกว่าการคำนวณก็ไม่ใช่ตัวปัญหา เพราะเราสามารถทำการคำนวณได้พร้อมๆกัน ซึ่งดีกว่า เมื่อเปรียบเทียบกับการสำรองข้อมูลไว้และประมวลผลด้วยหน่วยประมวลผลตัวเดียว

การประมวลผลแบบขนานนั้น จะจับคู่ข้อมูลเข้ากับ thread ประมวลผล ในหลายลักษณะงานที่ทำการคำนวณเกี่ยวข้องกับข้อมูลปริมาณมากๆมีความเป็นไปได้ที่จะทำการคำนวณในแบบขนาดแทนวิธีปรกติ ตัวอย่างเช่น การแสดงผลภาพสามมิติ pixel และจุดจำนวนมากจะถูกจับคู่เข้ากับ thread ประมวลผลแบบขนาน เช่นเดียวกันกับ การเข้ารหัส-ถอดรหัสวีดีโอ, การย่อ-ขยายรูป , การรู้จำภาพ ซึ่งงานเหล่านี้สามารถจับคู่ส่วนย่อยๆของภาพหรือวีดีโอเข้ากับ thread ประมวลผล เพื่อให้แต่ละส่วนทำงานไปพร้อมๆกันได้ ที่จริงแล้ว งานในสาขาอื่นจำนวนไม่น้อยก็สามารถที่จะคำนวณให้เร็วขึ้นได้โดยการเปลี่ยนมาคำนวณแบบขนาน เช่นงานจำพวกการจำลองทางฟิสิกส์, การประมวลข้อมูลที่เป็นสัญญานสื่อสาร, งานทางด้านการเงินการธนาคาร รวมถึงงานเกี่ยวกับโครงสร้างทางชีววิทยาอีกด้วย

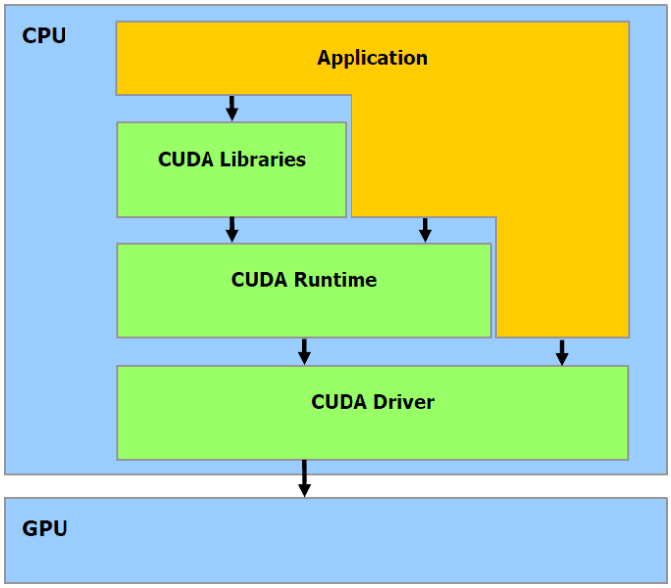
ในปัจจุบัน การจะใช้พลังการคำนวณของการ์ดแสดงผลให้เต็มประสิทธิภาพกับงานที่ไม่ใช่การแสดงผลภาพนั้นยังเป็นเรื่องที่ค่อนข้างซับซ้อนและวุ่นวายอยู่มากเนื่องจากหลายๆปัจจัย เช่น

- การ์ดแสดงผลนั้นจะต้องใช้ชุดคำสั่งพิเศษในการควบคุม ซึ่งเป็นเรื่องยากและเสียเวลาในการเรียนรู้ชุดคำสั่งใหม่ๆสำหรับผู้ที่ไม่เคยศึกษาหรือพัฒนาโปรแกรมเกี่ยวกับกราฟฟิกมาก่อน
- DRAM ของการ์ดแสดงผลนั้นมีรูปแบบการอ่านและเขียนข้อมูลที่เป็นลักษณะเฉพาะตัว เช่น โปรแกรมสามารถที่จะดึงข้อมูลจากส่วนใดๆของ DRAM ได้อย่างอิสระ แต่ไม่สามารถทำการเขียนข้อมูลลงบน DRAM แบบตรงๆได้ โปรแกรมที่ทำงานบนการ์ดประมวลผลจึงขาดความยืดหยุ่นในการพัฒนาโปรแกรม
- งานบางลักษณะที่โปรแกรมบนการ์ดแสดงผลใช้เวลาคำนวณน้อยๆอาจจะประสบปัญหาความล่าช้าที่การถ่ายโอนข้อมูลระหว่างหน่วยประมวลผลแทน

2.4.2 CUDA

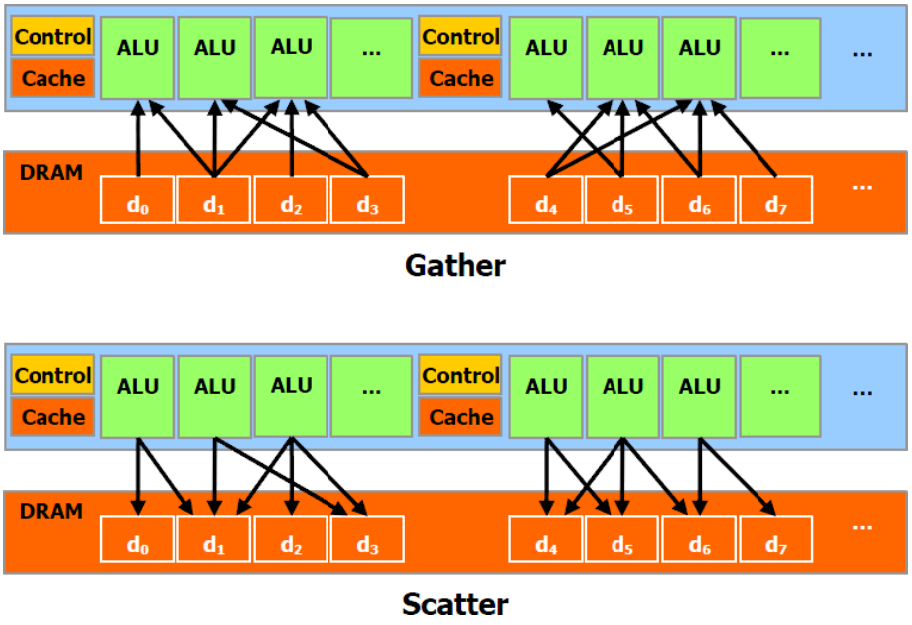
CUDA ย่อมาจาก **Compute Unified Device Architecture** เป็นสถาปัตย์กรรมทาง hardware และ software ตัวใหม่ของ NVidia ในการสั่งและควบคุมการ์ดแสดงผลให้ทำงานตามที่โปรแกรมต้องการ โดยมองการ์ดแสดงผลเป็นเสมือนอุปกรณ์คำนวณโดยที่ไวยากรณ์ไม่อิงกับชุดคำสั่งพิเศษทางกราฟฟิกของการ์ดแสดงผลอีกต่อไป และยังสามารถที่จะเรียกใช้โปรแกรมที่เขียนโดน CUDA หรือโปรแกรมทางกราฟฟิกอื่นๆได้พร้อมกันอีกด้วย โดยระบบ multitasking ของระบบปฏิบัติการของเครื่องคอมพิวเตอร์จะเป็นตัวจัดการการเข้าใช้งานการ์ดจออย่างอัตโนมัติ

ลำดับชั้นการทำงานของ CUDA ประกอบไปด้วยชั้นหลายชั้นตามที่ได้แสดงไว้ในรูป 1-3 ลำดับชั้นของ hardware driver, runtime API และ library ทางคณิตศาสตร์สำหรับไว้เรียกใช้โดยทั่วไป



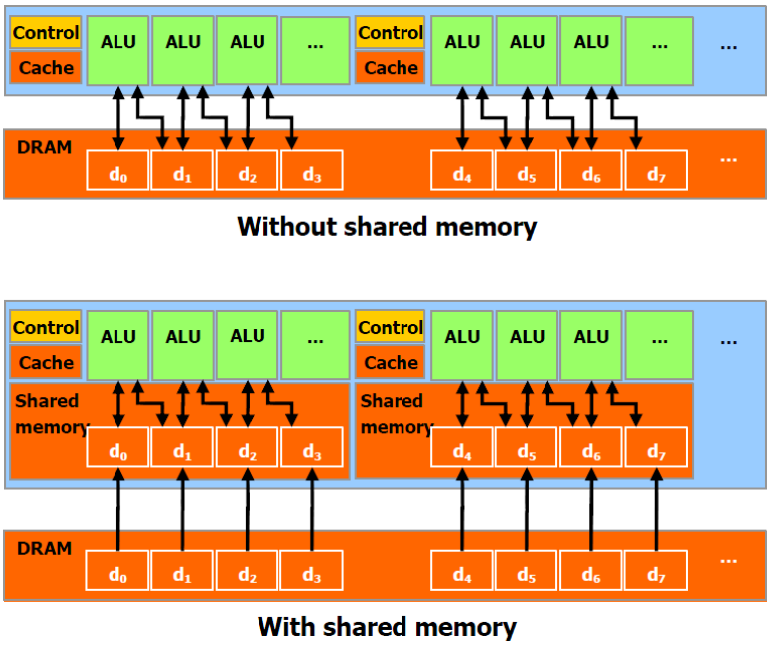
รูปที่ 12 ลำดับขั้นการทำงานของ CUDA

CUDA มีวิธีการเข้าถึงหน่วยความจำ DRAM ตามที่ได้แสดงไว้ในภาพที่ 13 เพื่อให้การอ่านและเขียนข้อมูลทำได้ง่ายขึ้นขั้นตอนการ scatter และ gather ข้อมูลจากหน่วยความจำจึงถูกเขียนคำสั่ง CUDA ครอบเพื่อทำให้ลักษณะการอ่านและเขียนทำได้เช่นเดียวกันกับ RAM ปรกติของคอมพิวเตอร์



รูปที่ 13 วิธีการเข้าถึงหน่วยความจำ DRAM

CUDA ยังมีหน่วยความจำประเภท shared memory สำหรับการเข้าถึงข้อมูลได้อย่างรวดเร็วจาก thread ที่อยู่ใน block เดียวกันอีกด้วย โปรแกรมจึงสามารถที่จะใช้หน่วยความจำประเภทนี้ในเพื่อลดการอ่านข้อมูลจาก DRAM ซึ่งกินเวลามากกว่า



รูปที่ 14 Shared Memory

2.4.3 การเขียนโปรแกรมบน CUDA

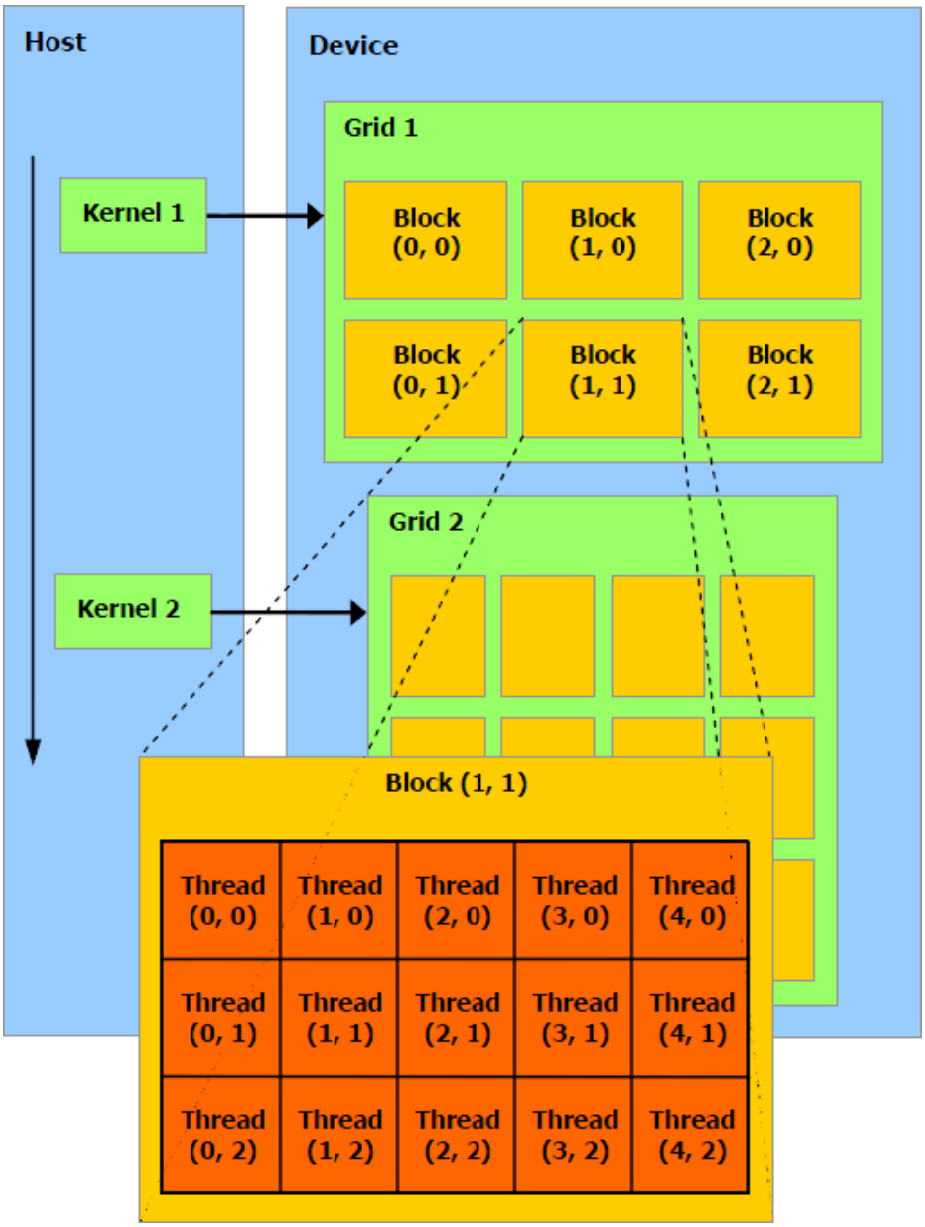
2.4.3.1 A Highly Multithreaded Coprocessor

เมื่อทำการเขียนโปรแกรมบน CUDA การ์ดแสดงผลจะถูกมองเป็นอุปกรณ์ภายในเครื่องคอมพิวเตอร์อีกชิ้นหนึ่ง(เรียกว่า device) ซึ่งสามารถประมวลผลแบบขนานได้พร้อมกันหลาย thread มากๆ โดยทำตัวเป็น coprocessor สำหรับ CPU (เรียกว่า host) อีกนัยหนึ่งคือเปรียบเสมือน CPU ยกงานที่มีขั้นตอนการมีประมวลผลเป็นแบบขนานและกินพลังคำนวณสูงๆ ไปทำที่การ์ดแสดงผล (device) ชิ้นนี้

กล่าวแบบเจาะจงลงไป คือการยกงานที่มีลักษณะเป็นรอบซ้ำๆหลายรอบ ต่างกันตรงที่แต่ละรอบกระทำกับข้อมูลคนละชุดกัน และการทำงานแต่ละรอบไม่ขึ้นอยู่กับข้อมูลของรอบก่อนๆ มาตัดเป็นฟังก์ชั่นที่สามารถทำงานไปพร้อมๆกันได้ ในทางปฏิบัติ มันจะถูก complie ให้เป็นโปรแกรมที่ทำงานบนการ์ดแสดงผลโดยที่ทำพร้อมๆกันหลาย thread และส่งโปรแกรมส่วนนี้ที่เรียกว่า kernel ขึ้นไปทำงานบนการ์ดแสดงผลในตอนที่โปรแกรมถูกใช้งาน

สำหรับหน่วยความจำนั้น ในส่วนของ CPU จะใช้ RAM ตามปรกติของเครื่อง เราจะเรียกว่า host memory ส่วนการ์ดแสดงผลก็จะใช้ DRAM ของตัวมันเองซึ่งเราจะเรียกว่า device

memory โดยการโอนข้อมูลระหว่าง host memory และ device memory นั้นจะกระทำผ่านการสั่ง API ที่ CUDA เตรียมไว้ให้เรียกใช้อยู่แล้ว



รูปที่ 15 สถาปัตยกรรม CUDA

2.4.3.2 Kernel

Kernel เป็นคำที่ใช้เรียกโปรแกรมที่ทำงานอยู่บน device โดยจะทำงานพร้อมๆกันหลายครั้งแบบขนาน โดยแต่ละชุดของการทำงานจะเรียกว่า thread และมี thread ID เป็นเลขอ้างอิงของแต่ละ thread ที่ทำงานอยู่บน device

2.4.3.3 Thread Block

Thread block คือกลุ่มของ thread ที่ถูกจัดให้รวมกลุ่มกันทำให้มีสิทธิในการเข้าถึง shared memory ซึ่งเป็น memory ความเร็วสูงที่เข้าถึงได้เฉพาะ thread ที่อยู่ใน block เดียวกันเท่านั้นรวมถึงสามารถใช้คำสั่งพิเศษเพื่อทำการ synchronize กับ thread อื่นๆใน block เดียวกันได้อีกด้วย โดย thread ที่ทำงานมาถึงจุดนี้ก่อนจะหยุดรอให้ thread อื่นๆมาถึงจุดนี้เพื่อเริ่มทำงานพร้อมๆกันต่อไป

Thread แต่ละตัวจะมีเลขประจำตัวที่เรียกว่า thread ID ซึ่งเป็นเลขเฉพาะ ใน block เดียวกันจะไม่มี thread ที่มี ID ซ้ำกัน ซึ่ง ID นี้มักจะถูกใช้เพื่ออ้างอิงตำแหน่งของข้อมูลที่แต่ละ thread ต้องการนำมาใช้หรือเขียนข้อมูลกลับลงไปในหน่วยความจำ โดย ID ตัวนี้อาจเป็นตัวแปร 1 มิติ, 2 มิติ หรือ 3 มิติก็ได้ ทั้งนี้ขึ้นอยู่กับขั้นตอนการประกาศ block size ซึ่งแล้วแต่นักพัฒนาว่างานที่ต้องใช้นั้นมีความเหมาะสมกับ thread 1 มิติ, 2 มิติ หรือ 3 มิติ โดย block size 2 มิติ *(Dx, Dy)*, จะมี thread ID อ้างอิงกับพิกัด *(x, y)* ใดๆเป็น *(x + y Dx)* และสำหรับกรณี block size 3 มิติ *(Dx, Dy, Dz)*,จะมี thread ID อ้างอิงกับพิกัดตำแหน่ง *(x, y, z)* ใดๆเป็น *(x + y Dx + z Dx Dy)*.

2.4.3.4 Grid of Thread Blocks

จำนวน thread สูงสุดที่แต่ละ block มีได้นั้นเป็นจำนวนที่จำกัด แต่ block ที่มีมิติเท่ากันและกำลังทำงานบนkernel เดียวกันนั้น สามารถที่จะถูกจัดกลุ่มรวมกันเป็น grid ได้จึงทำให้ปริมาณ thread ที่ทำงานได้พร้อมๆกันนั้นมีจำนวนมาก เพราะการทำงานพร้อมๆกัน 1 รอบนั้นสามารถทำได้ทั้ง grid ไม่ใช่เพียงทีละ block แต่การทำงานทีเดียวทั้ง grid ก็จะมีข้อเสียเพิ่มเข้ามาอีกหนึ่งอย่างซึ่งก็คือ thread จากคนละ block นั้น ไม่สามารถ synchronize และใช้ shared memory ร่วมกันได้ การออกแบบให้การ์ดจอประมวลผลเป็นทีละ grid นั้นทำขึ้นเพื่อให้การ์ดจอที่ต่างรุ่นกัน, มีความสามารถต่างกันสามารถทำงานเต็มประสิทธิภาพมากขึ้น ตัวอย่างเช่นการ์ดจอรุ่นที่ความสามารถในการทำงานแบบขนานต่ำอาจจะทำงานทีละ block ใน grid ส่วนรุ่นที่สมรรถนะสูงอาจจะทำงานพร้อมๆกันทุก block ใน grid เป็นต้น

แต่ละ block ก็จะมีเลขประจำตัวเช่นเดียวกับ thread ซึ่งเรียกว่า block ID เพื่อเป็นตัวระบุว่าตัวมันเป็น block ลำดับที่เท่าไหร่ใน grid และเช่นเดียวกับ thread, block ก็สามารถที่จะอ้างอิง ID ได้เป็น 1 หรือ 2 มิติ โดย grid ที่มีสองมิติ *(Dx, Dy)*, จะมี block ID อ้างอิงกับพิกัด *(x, y)* ใดๆ เป็น *(x + y Dx)*

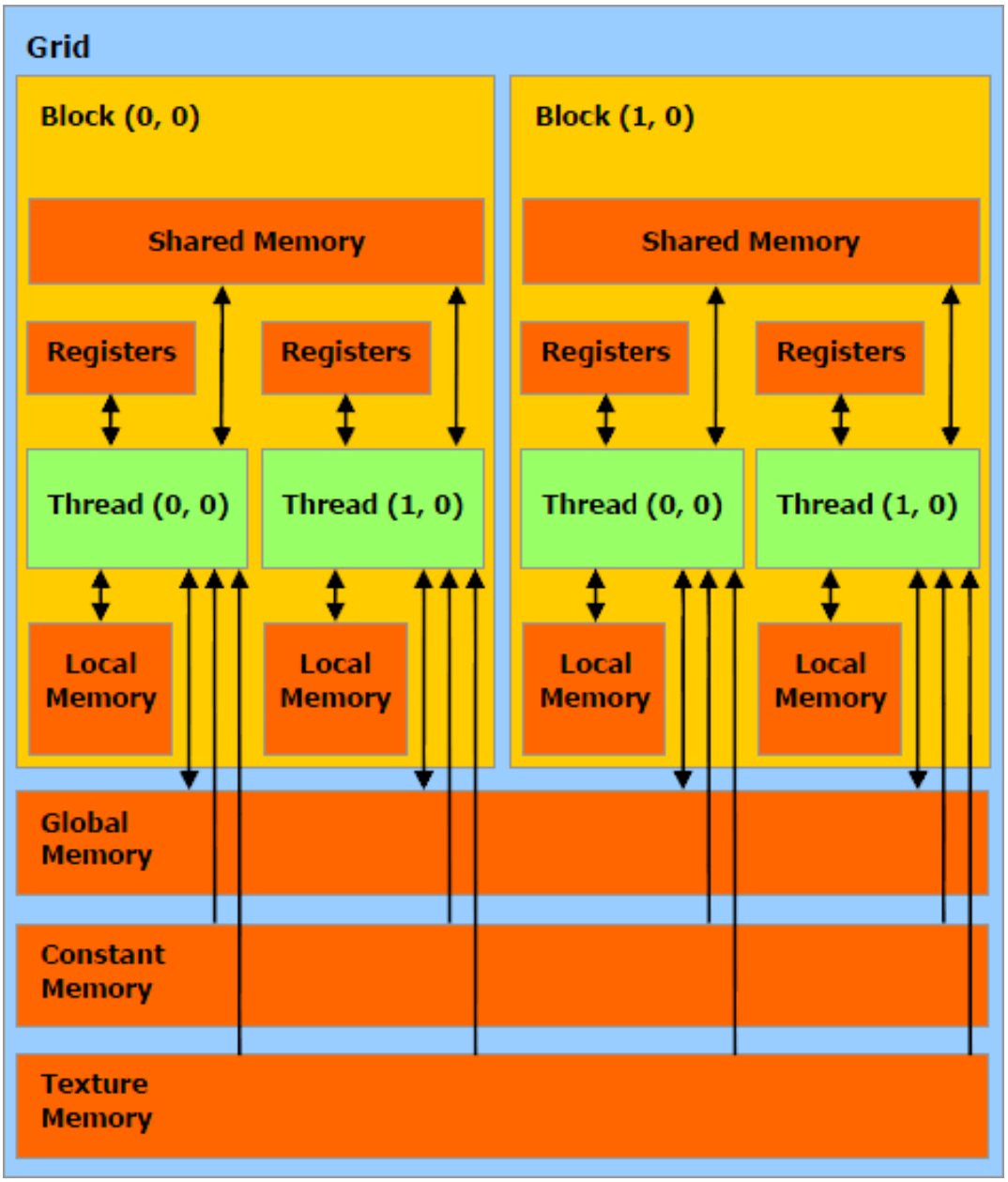
โปรแกรมจากฝั่ง host จะเป็นตัวส่งคำสั่งของ kernel ซึ่งเป็นตัวโปรแกรมสำหรับแต่ละ thread ไปยัง device และจะทำการประมวลผลโดยการรวบ thread เข้าเป็น block และ grid ตามที่ได้กล่าวไปแล้วในข้างต้น

2.4.3.5 Memory Model

Threada ที่ทำงานบน device นั้นสามารถเข้าถึง DRAM และหน่วยความจำที่อยู่บนการ์ดจอประเภทต่างๆได้ดังต่อไปนี้

- Registers : สามารถอ่านและเขียนได้, เข้าถึงได้จากระดับ thread เดียวกัน
- Local memory : สามารถอ่านและเขียนได้, เข้าถึงได้จากระดับ thread เดียวกัน
- Shared memory : สามารถอ่านและเขียนได้, เข้าถึงได้จากระดับ block เดียวกัน
- Global memory : สามารถอ่านและเขียนได้, เข้าถึงได้จากระดับ grid เดียวกัน
- Constant memory : สามารถอ่านได้อย่างเดียว, เข้าถึงได้จากระดับ grid เดียวกัน
- Texture memory : สามารถอ่านได้อย่างเดียว, เข้าถึงได้จากระดับ grid เดียวกัน

Global, constant และ texture memory นั้นสามารถอ่านหรือเขียนได้จากโปรแกรมในฝั่ง host และจะอยู่ต่อเนื่องไปจนกว่าจะทำการประมวลผล kernel ใดๆจบสิ้น global, constant และ texture memory นั้นเหมาะกับกันใช้งานในสถานะการที่แตกต่างกัน ตัวอย่างเช่น texture memory นั้นสามารถอ้างอิงต่ำแหน่งการเข้าถึงข้อมูลได้หลายแบบรวมถึงการจัดรูปแบบของค่าที่อ่านได้ด้วย



รูปที่ 16 การเข้าถึงหน่วยความจำของ CUDA

2.4.4 ระบบ Hardware ของการ์ดแสดงผล

2.4.4.1 A Set of SIMD Multiprocessors with On-Chip Shared Memory

การ์ดแสดงผลที่สามารถประมวลผลโปรแกรม CUDA ได้นั้นประกอบขึ้นจาก multiprocessor หลายๆตัวตามที่ได้แสดงในรูปที่ 17 โดยที่แต่ละ multiprocessor นั้นเป็นสถาปัตยกรรมชนิดที่เรียกว่า Single Instruction, Multiple Data architecture หรือ (SIMD): ใน

แต่ละหนึ่งรอบของนาฬิกานั้น multiprocessor แต่ละตัวจะทำงานคำสั่งเดียวกันแต่กระทำกับข้อมูลคนละชุดกัน โดย multiprocessor แต่ละตัวนั้นมีหน่วยความจำอยู่ 4 ประเภทคือ

- Registers แบบ 32-bit ต่อ หนึ่งหน่วยประมวลผล (processor)
- หน่วยความจำ shared memory ซึ่งหน่วยประมวลผลทุกตัวสามารถเข้าถึงได้
- หน่วยความจำที่อ่านได้อย่างเดียว constant cache ซึ่งหน่วยประมวลผลทุกตัวสามารถเข้าถึงได้และอ่านข้อมูลได้อย่างรวดเร็ว
- หน่วยความจำที่อ่านได้อย่างเดียว texture memory เข้าถึงได้โดยหน่วยประมวลผลทุกตัวเช่นกัน สามารถอ่านข้อมูลได้อย่างรวดเร็ว

หน่วยความจำประเภท local และ global memory นั้นจะเป็นหน่วยความจำทั่วไปบนการ์ดแสดงผล และไม่มีการ cache และแต่ละ multiprocessor จะเข้าถึงหน่วยความจำประเภท texture memory ได้ผ่านทาง texture unit ซึ่งสามารถเรียกข้อมูลได้จากหลายวิธีอ้างอิงที่แตกต่างกัน

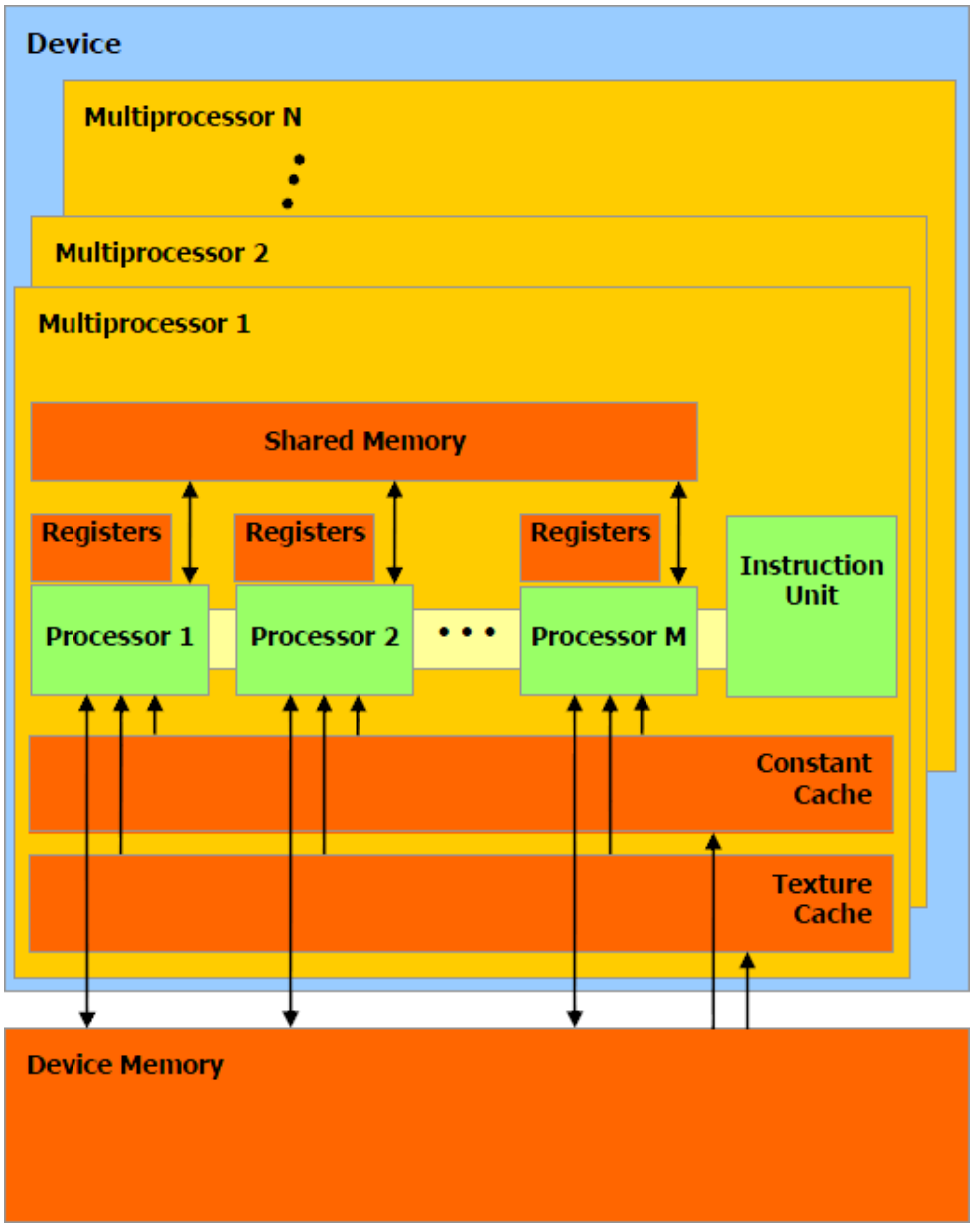
2.4.4.2 รูปแบบการประมวลผล

ในขั้นตอนการประมวลผลบนการ์ดจอนั้น โปรแกรมจะทำการประมวลทีละ block หรือมากกว่านั้นต่อหนึ่ง multiprocessor โดยแบ่งเป็นกลุ่มๆที่เรียกว่า warp โดยแต่ละ warp จะมี thread จำนวนเท่าๆกันที่เรียกว่า warp size โดยที่ thread scheduler จะสลับการทำงานของแต่ละ warp เพื่อให้ประสิทธิผลออกมาสูงสุด และครึ่งหนึ่งของ warp จะเรียกว่า half-warp ซึ่งหมายถึงครึ่งแรกหรือครึ่งหลังของ warp ก็ได้ วิธีที่ block โดนแบ่งเป็น warp นั้นเหมือนเดิมเสมอ ในแต่ละ warp จะมี thread ที่หมายเลขอ้างอิงติดกัน โดย warp แรกจะมี thread ID หมายเลข 0 อยู่เสมอ (วิธีอ้างอิงหมายเลขของ thread ดูได้จาก 1.2.2)

Block หนึ่งๆ จะถูกประมวลผลโดย multiprocessor ตัวเดียวกันเท่านั้น นี่จึงเป็นสาเหตุให้การอ่านและเขียนไปยัง shared memory สามารถเข้าทำได้อย่างรวดเร็ว ส่วน register ของ multiprocessor นั้นก็จะถูกจองและใช้งานโดยแต่ละ thread ของ block ถ้าจำนวน register ที่ถูกจองต่อๆหนึ่ง thread คูณกับจำนวน thread ต่อหนึ่ง block มากกว่าจำนวน register ที่มีต่อหนึ่ง multiprocessor แล้ว block นั้นๆก็จะไม่สามารถทำงานได้ และการประมวลผล kernel นั้นๆก็จะล้มเหลว

Multiprocessor นั้นสามารถที่จะทำการประมวลผลได้พร้อมๆกันทีละหลาย block ถ้ามี register และ shared memory เหลือพอสำหรับทำการประมวลผล ส่วนลำดับการทำงานของแต่ละ warp นั้นก็ไม่มีลำดับที่แน่นอนว่า warp ใดจะได้ทำงานก่อนหรือทำงานทีหลัง แต่ถ้าต้องการการันตีให้การทำงานของทุกๆ thread ใน block นั้นทำถึงจุดที่ต้องการจุดใดจุดหนึ่งในโปรแกรมแล้วค่อยทำงานต่อหรือเพื่อให้แน่ใจได้ว่าข้อมูลบางตัวถูกประมวลผลเรียบร้อยแล้วทั้งหมดก็สามารถทำได้โดยใส่คำสั่งลงไปในโค๊ดของโปรแกรมเพื่อทำการ synchronize thread ณ จุดที่ต้องการ แต่การ synchronize ระหว่าง block นั้นไม่สามารถกระทำได้เพราะสถาปัตยกรรมไม่ได้ออกแบบให้รองรับ

ถ้าใน warp มีขั้นตอนการทำงานใดที่เป็น non-atomic instruction และพยายามทำการเขียนข้อมูลลงไปที่ global memory หรือ shared memory พร้อมๆกันจาก thread มากกว่าหนึ่ง thread เราจะไม่สามารถบอกได้ว่า thread ใดจะได้เขียนก่อนหรือหลัง บอกได้แต่เพียงว่าจะมี thread อย่างน้อยหนึ่ง thread ที่เขียนสำเร็จ แต่สำหรับขั้นตอนที่เป็น atomic instruction ทำการอ่าน,ปรับปรุงข้อมูลหรือเขียนข้อมูล ณ ตำแหน่งเดียวกันบน global memory โดย thread มากกว่าหนึ่ง thread การทำงานของคำสั่งนั้นๆ จะรับประกันว่าจะทำงานแน่นอน แต่ thread ใดได้ทำก่อนหรือหลังนั้นไม่สามารถบอกได้



รูปที่ 17 สถาปัตยกรรมของ CUDA

2.4.4.3 Compute Capability

Compute capability นั้นเป็นเลขอ้างอิงสำหรับระบุรุ่นของ CUDA การ์ดแสดงผลใดที่มีหมายเลข compute capability เดียวกันแสดงว่าเป็นสถาปัตยกรรมรุ่นเดียวกัน ส่วนทศนิยมของหมายเลขอ้างอิงนั้นหมายความว่าได้มีการปรับปรุงจากรุ่นเดิม อาจร่วมถึงการเปลี่ยนสถาปัตยกรรมหรือรวมไปถึงการเพิ่มความสามารถใหม่ๆอีกด้วย

2.4.5 การเขียนโปรแกรมบน CUDA

ไวยากรณ์ที่เพิ่มเติมจากภาษา C แบบปรกติมีดังต่อไปนี้

- ตัวกำกับเพื่อให้ทราบว่าฟังก์ชั่นนั้นทำงานอยู่ในส่วนของ host หรือ device และถูกเรียกใช้ได้จาก host หรือ device
- วิธีประกาศว่า kernel นั้นจะมีลักษณะการทำงานอย่างไร
- ตัวแปรพิเศษสำหรับระบุถึงหายเลขอ้างอิงของ thread และ block

2.4.5.1 ตัวกำกับของฟังก์ชั่นที่เพิ่มเติมจากภาษา C แบบปรกติ

__device__ ทำงานบน device และถูกเรียกใช้ได้จาก device เท่านั้น

__global__ เป็นฟังก์ชั่นที่เป็น kernel ทำงานบน device และเรียกใช้จาก host

__host__ เป็นฟังก์ชั่นที่ทำงานบน host ซึ่งก็เหมือนกับการประกาศฟังก์ชั่นโดนไม่ใส่ตัวกำกับ แต่ที่มีตัวกำกับนี้ก็เพื่อใช้คู่กับตัวกำกับอื่นๆ ในกรณีที่ต้องการให้ฟังก์ชั่นนั้นถูก compile ไปทั้งบน host และ device

ข้อจำกัดของตัวกำกับ

- __device__ และ __global__ ไม่รองรับการเรียกซ้ำตัวเอง (recursion)
- __device__ และ __global__ ไม่สามารถประกาศตัวแปรประเภท static ไว้ภายในได้
- __device__ และ __global__ ไม่สามารถรับ input ที่มีจำนวนยืดหยุ่นได้
- __device__ และ __global__ ไม่สามารถอ้างถึง memory address ไปยัง function นั้นได้
- __host__ และ __global__ ไม่สามารถประกาศใช้กับฟังก์ชั่นเดียวกันได้
- __global__ ต้องเป็น void function เท่านั้น (ไม่มีการ return ค่า)
- การเรียกใช้ฟังก์ชั่นที่เป็น __global__ จะต้องใส่การตั้งค่าพิเศษเสมอ (จะกล่าวถึงต่อไป)
- การเรียกใช้ฟังก์ชั่นที่เป็น __global__ จะกลับมาทำงานในโปรแกรมฝั่ง host ก่อนที่จะทำงานเสร็จจริง (asynchronous)

2.4.5.2 ตัวกำกับของตัวแปรที่เพิ่มเติมจากภาษา C แบบปรกติ

__device__ ใช้กำกับเพื่อระบุว่าเป็นตัวแปรที่อยู่บน device และสามารถระบุเพิ่มเติมได้ว่าจะให้ตัวแปรนี้อยู่ที่ memory ประเภทไหน ซึ่งถ้าไม่ระบุเพิ่มเติมตัวแปรประเภท __device__ จะมีสมบัติดังนี้

- อยู่บน global memory
- อยู่ตั้งแต่เริ่มโปรแกรมจนจบโปรแกรม
- เข้าถึงได้จากทุก thread บน grid และสามารถใช้คำสั่งพิเศษในการเข้าถึงตัวแปรเหล่านี้จากฝั่ง host ด้วย

__constant__ มักใช้ร่วมกับ __device__ มีสมบัติดังนี้

- อยู่บน constant memory space
- อยู่ตั้งแต่เริ่มโปรแกรมจนจบโปรแกรม
- เข้าถึงได้จากทุก thread บน grid และสามารถใช้คำสั่งพิเศษในการเข้าถึงตัวแปรเหล่านี้จากฝั่ง host ด้วย

__shared__ มักใช้ร่วมกับ __device__ มีสมบัติดังนี้

- อยู่บน share memory space ของ block
- อยู่ตั้งแต่ block เริ่มถูกประมวลผลจน block ชิ้นนั้นจบการประมวลผล
- เข้าถึงได้จากเฉพาะ thread ที่อยู่ใน block เดียวกัน
- ข้อมูลที่ถูกเขียนลงบนตัวแปรประเภทนี้โดย thread ใด thread หนึ่งจะไม่สามารถรับประกันได้ว่าถูกทำงานพร้อมกันทุก thread เพราะ thread scheduler อาจเลือกทำงานบาง warp จนจบก่อนที่ thread ใน warp อื่นจะเริ่มทำงาน ถ้าต้องการรับประกันว่าขั้นตอนการเขียนข้อมูลกลับไปยังตัวแปรประเภทนี้จะถูกทำก่อนไปยังขั้นตอนถัดไป ก็ต้องใส่คำสั่งเพื่อทำการ synchronize ก่อน

ข้อจำกัดของตัวกำกับ

- __ shared __ และ __ constant __ ไม่สามารถใช้ร่วมกันได้
- __ constant __ ต้องถูกกำหนดค่าจากฝั่ง host เท่านั้น
- __ shared __ ไม่สามารถกำหนดค่าเริ่มต้นตั้งแต่ตอนประกาศได้

- การเรียกใช้ฟังก์ชั่นที่เป็น __global__ จะต้องใส่การตั้งค่าพิเศษเสมอ (จะกล่าวถึงต่อไป)
- การเรียกใช้ฟังก์ชั่นที่เป็น __global__ จะกลับมาทำงานในโปรแกรมฝั่ง host ก่อนที่จะทำงานเสร็จจริง (asynchronous)

ถ้ามีการประกาศตัวแปรโดยไม่ใช้ตัวกำกับใดๆเลย ตัวแปรนั้นนะใช้พื้นที่ของ register หรือในบางกรณี complier จะเลือกที่จะใช้พื้นที่บน local memory ถ้าเป็นการประกาศตัวแปรขนาดใหญ่ๆเช่น array ส่วนการใช้ตัวแปรประเภท pointer นั้นสามารถชี้ไปที่ shared memory หรือ global memory ก็ได้ ถ้า compiler สามารถจำแนกปลายทางที่จะทำการชี้ไป ถ้ามิเช่นนั้นจะบังคับให้ใช้อ้างอิงถึง global memory เพียงอย่างเดียวเท่านั้น และถ้าตั้งค่าให้ pointer ในส่วนของ host ชี้ไปยังหน่วยความจำของ device หรือตั้งค่าให้ pointer ในส่วนของ device ชี้มายังหน่วยความจำของ host ก็จะเกิดพฤติกรรมที่ไม่สามารถระบุได้ และมักจะนำไปสู่อาการ segmentation fault หรือโปรแกรมหยุดทำงานได้

2.4.5.3 **การกำหนดพารามิเตอร์ให้กับการประมวลผล** kernel

การเรียกใช้ฟังก์ชั่นใดๆที่เป็น __global__ นั้น จะต้องทำการระบุพารามิเตอร์ให้กับการประมวลผลนั้นๆด้วย โดยพารามิเตอร์ที่กำหนดให้นั้นจะประกอบไปด้วย มิติของ grid, มิติของ block และปริมาณหน่วยความจำบน shared memory ต่อ block ที่ต้องการให้โปรแกรมใช้โดยระบุพารามิเตอร์นี้ลงไประหว่างชื่อฟังก์ชั่นกับ input ของฟังก์ชั่นนั้นๆ เช่น Func **<<< Dg, Db, Ns >>>** (input) โดยที่ Dg, Db และ Ns มีความหมายดังต่อไปนี้

- Dg เป็นตัวแปรประเภท dim3 (เป็นประเภทตัวแปรเฉพาะของ CUDA หมายถึงเป็นจำนวนเต็มที่มากกว่าศูนย์ในสามมิติ แต่จะกำหนดเพียงหนึ่งหรือสองมิติก็ได้ ใช้สำหรับการอ้างอิงถึงมิติเท่านั้น) โดยที่ Dg.x * Dg.y จะเท่ากับจำนวน block ทั้งหมดที่ต้องการประมวลผล
- Db เป็นตัวแปรประเภท dim3 ซึ่งเป็นตัวระบุขนาดของ block โดยที่ Db.x * Db.y * Db.z จะเป็นจำนวน thread ต่อหนึ่ง block

- Ns เป็นจำนวน byte ของหน่วยความจำ shared memory ที่ต้องการจัดสรรให้ต่อหนึ่ง block เพื่อนำไปใช้กับตัวแปรที่ทำการประกาศเป็นประเภท external array Ns เป็นตัวแปรที่จะระบุหรือไม่ระบุก็ได้ และมีค่าตั้งต้นเป็น 0 ถ้าไม่ทำการประกาศ

2.4.5.4 Built-in Variables

- gridDim เป็นตัวแปรประเภท dim3 และจะบอกถึงขนาดมิติของ grid
- blockIdx เป็นตัวแปรประเภท unit3(เหมือนกับ dim3 ทุกอย่าง แต่ใช้กับอะไรก็ได้) และบอกถึงหมายเลขประจำตัวของ block นั้นๆ
- blockDim เป็นตัวแปรประเภท dim3 และจะบอกถึงขนาดมิติของ block
- threadIdx เป็นตัวแปรประเภท unit3 และบอกถึงหมายเลขประจำตัวของ thread นั้นๆ

ข้อจำกัดของ built-in variable คือ

- ไม่สามารถอณุญาติให้โปรแกรมอ้างอิงถึงตำแหน่งในหน่วยความจำของตัวแปรเหล่านี้ได้
- ไม่สามารถตั้งค่าใหม่ให้กับตัวแปรเหล่านี้ได้

2.4.5.5 Built-in Vector Types

- char1, uchar1, char2, uchar2, char3, uchar3, char4, uchar4, short1, ushort1, short2, ushort2, short3, ushort3, short4, ushort4, int1, uint1, int2, uint2, int3, uint3, int4, uint4, long1, ulong1, long2, ulong2, long3, ulong3, long4, ulong4, float1, float2, float3, float4
  ตัวแปรเหล่านี้ก็เหมือนกับตัวแปรของภาษา C ปรกติส่วนเลขที่ตามท้ายนั้นจะเป็นจำนวนมิติของตัวแปรโดยสามารถอ้างอิงถึงได้ด้วยตัวแปร x, y, z และ w

- dim3

  เหมือนกับ uint3 ทุกประการ แต่ค่าเริ่มต้นของมิติที่ไม่ได้ประกาศค่าไว้จะเป็น 1

2.4.6 Texture Memory

เป็นหน่วยความจำที่อยู่บน device memory และสามารถอ่านได้อย่างเดียว เมื่อมีการอ่านข้อมูล จะมีการสำรองข้อมูลของจุดใกล้เคียงเอาไว้ ถ้ามีการอ่านข้อมูลจากบริเวณที่ใกล้กับการอ่านครั้งที่แล้ว texture memory ก็สามารถที่จะส่งค่ากลับไปยังโปรแกรมได้โดยที่ไม่ต้องทำการอ่านจากหน่วยความจำจริงๆซึ่งจะช่วยประหยัดเวลาไปได้อย่างมาก โดยเฉพาะกรณีที่ thread ใน warp เดียวกันทำการอ่านข้อมูลจากตำแหน่งใกล้ๆกัน

Texture memory นั้นได้ถูกออกแบบมาโดยเน้นให้รับมือกับการอ่านข้อมูลจาก 2 มิติ กล่าวคือเมื่อทำการอ่านค่าจาก texture memory ที่พิกัดใด ข้อมูลที่อยู่ใกล้ตามระยะทางบน 2 มิติกับจุดนั้นจะถูกสำรองเอาไว้ การอ่านข้อมูลจาก texture memory แทนที่จะเลือกใช้ global memory หรือ constant memory นั้นมีข้อดีดังต่อไปนี้

- Global memory นั้นมีลักษณะการเข้าถึงข้อมูลเป็นกลุ่มๆที่เรียกว่า memory bank ถ้ามีการอ่านข้อมูลคนละตัวกันจาก memory bank เดียวกัน ก็จะต้องรอทำการอ่านมากกว่าหนึ่งรอบ ตามจำนวนข้อมูลที่แตกต่างกันนั้น แต่ texture memory จะไม่มีปัญหาประเภทนี้ จึงอ่านข้อมูลได้เร็วกว่ามาก
- การคำนวณตำแหน่งของข้อมูลที่จะทำการอ่านจะทำนอก kernel โดยอุปกรณ์เฉพาะบน multiprocessor จึงทำให้ใช้เวลาน้อยกว่า
- สามารถแปลงข้อมูลชนิด integer ให้กลายเป็นจำนวนจริงในช่วง [0.0,1.0] หรือ [-1.0,1.0] ก่อนส่งกลับได้ ซึ่งมีมีประโยชน์กับงานทางด้านกราฟฟิกอย่างมาก

การอ่านข้อมูลจาก texture memory นั้น จะต้องทำการอ่านผ่านคำสั่งเฉพาะ (texture fetches) โดยที่จะมีตัวแปรประเภท texture reference ซึ่งเป็นตัวชี้ไปยัง texture ซึ่งสามารถระบุค่าให้กับ texture reference ได้ว่าจะให้อ่านข้อมูลจาก texture ในแบบ 1 มิติ, 2 มิติหรือ 3 มิติ ตัว texture memory ที่ใช้ในวิทยานิพนธ์ฉบับนี้นั้นคือ cuda array ซึ่งเป็นหน่วยความจำประเภทที่ออกแบบ

มาให้ใช้เป็น texture memory โดยเฉพาะ สามารถประกาศได้ตั้งแต่ 1 ถึง 3 มิติ รองรับการใช้งานประเภท 8, 16, 32 bit integers หรือ 16, 32 bit float และโปรแกรมในฝั่ง device สามารถทำการอ่านข้อมูลจาก cuda array ผ่านทาง texture reference ได้เท่านั้น ส่วนการเขียนข้อมูลจะทำผ่านโปรแกรมจากฝั่ง host ผ่านคำสั่งเฉพาะ cudaMemcpyToArray ที่จะเป็นตัวกลางในการคัดลอกข้อมูลจาก array แบบปรกติบนโปรแกรมฝั่ง host ไปยัง cuda array บนฝั่ง device

2.4.7 คำสั่งอื่น ๆ ที่เกี่ยวกับการจัดการหน่วยความจำ

การจองหน่วยความจำบน device แบบมิติเดียวนั้นสามารถจะข้องเกี่ยวกับคำสั่งต่อไปนี้ ได้แก่ cudaMalloc ซึ่งทำงานเหมือนกับคำสั่ง malloc ของภาษา C ปรกติ, cudaFree สำหรับคืนหน่วยความจำ เช่นเดียวกับคำสั่ง free ของภาษา C ปรกติ และ cudaMallocPitch ซึ่งใช้สำหรับ

```
float* num;

cudaMalloc(&num,10*sizeof(float));
```

และใช้คำสั่ง cudaFree สำหรับคืนหน่วยความจำ เช่นเดียวกับคำสั่ง free ของภาษา C ปรกติ

ส่วนการจอง array ใน 2 มิตินั้น จะใช้ คำสั่ง cudaMallocArray เพื่อจอง array และ cudaFreeArray ในการคืนหน่วยความจำ คำสั่ง cudaMallocArray นั้นจะต้องใช้ร่วมกับคำสั่ง cudaCreateChannelDesc ซึ่งเอาไว้ระบุประเภทของตัวแปร, วิธีการอ่านข้อมูล และการแปลงคำตอบที่อ่านได้ก่อนที่จะคืนค่ากลับมาในขั้นตอนการอ่านข้อมูล

```
cudaChannelFormatDesc channelDesc =
cudaCreateChannelDesc<float>();
cudaArray* cuArray;
cudaMallocArray(&cuArray, &channelDesc, width, height);
```

บทที่ 3

วิธีการทดลอง

3.1 การจับวัตถุ 2 มิติด้วยนิ้ว 2 นิ้ว

3.1.1 วิธีอธิบายการจับที่มีคุณสมบัติแบบปิดของแรง

ในงานนี้เราสนใจการจับวัตถุโพลิกอนสองมิติที่ขอบวัตถุไม่ตัดกันเองและสามารถที่จะมีวัตถุจำนวนหลายๆชิ้นได้ ในกรณีที่วัตถุเป็นโพลิกอนหลายชิ้น เราจะเลือกก้อนวัตถุมาคำนวณทีละคู่ configuration space ของเราจะเป็นมิติของขอบวัตถุที่ใช้ตัวแปรสองตัวเป็นตัวระบุตำแหน่งของการวางนิ้วแต่ละนิ้วบนผิววัตถุ โดยที่บทนี้จะอธิบายการสร้าง configuration space สำหรับเก็บท่าจับที่มีคุณสมบัติแบบปิดของแรง

โพลิกอน P ที่สร้างจากจุด n จุด $v_i \in R^2; i \in Z_n$ (Z_n เป็นจำนวนเต็มที่ไม่ติดลบที่มีค่าน้อยกว่า n, การบวกหรือลบจะทำการหารหาเศษด้วย n) การไล่ลำดับของจุดจะนับในทิศทวนเข็มนาฬิกาถ้าโพลิกอนที่แทนบริเวณที่เป็นเนื้อวัตถุ และจะนับในทิศตามเข็มนาฬิกาถ้าโพลิกอนแทนที่ที่ว่างภายในวัตถุ เส้นขอบ $\mathrm{E_i}$ คือด้านที่เชื่อมระหว่างจุด v_i และ v_{i+1} ทุกๆจุด p ใดๆบนขอบของ P นั้นสามารถที่จะอ้างอิงถึงได้ด้วยระยะของเส้นทางระหว่างจุด v_0 ไปถึงจุด p โดยนับระยะทางตามขอบของ P ระยะทางดังกล่าวจะเรียกว่า $\mathrm{length(p)}$ โดยที่ความยาวของ E_i สามารถเขียนได้ในรูป $l_i = ||v_{i+1} - v_i||$ และจะเห็นว่า $L_i = length(\ v_i) = \sum_{j \in Z_i} l_j$ และความยาวของเส้นรอบรูป P คือ $L = \sum_{i \in Z_n} l_i$

ขั้นตอนถัดมาเราจะทำการนิยามเส้นสัมผัสของ E_i เป็น $t_i = (v_{i+1} - v_i)/l_i$ และ n_i เป็นเส้นตั้งฉากกับ t_i มีขนาด 1 หน่วยและชี้เข้าสู่เนื้อวัตถุ (n_i สามารถหาได้โดยการหมุน t_i ในทิศทวนเข็มนาฬิกาไป 90 องศา) กรวยแรงเสียดทาน C_i ที่สามารถกระทำบนด้าน E_i สามารถนิยามได้เป็นเวกเตอร์ $n_i + (\tan \alpha)t_i$ และ $n_i - (\tan \alpha)t_i$ เมื่อ $t_i \in [0, \pi/2)$ เป็นมุมครึ่งหนึ่งของกรวยแรงเสียดทาน

ท่าจับบางท่า นิ้วจับอาจจะไม่ได้อยู่บนโพลิกอนเดียวกัน จึงจำเป็นที่จะต้องนิยามสมบัติบางอย่างเพิ่มเติม โดยจะกำหนดให้สมบัติที่นิยามไว้ในย่อหน้าที่แล้วเป็นสมบัติของนิ้วแรก และกำหนดให้ $n', v_i', E_i', length', l_i', L_i', l', t_i', n_i'$ และ C_i' มีนิยามตามสมบัติในย่อหน้าที่แล้วแต่ใช้กับนิ้วจับที่สอง ส่วนในกรณีที่นิ้วจับทั้งสองนิ้วอยู่บนโพลิกอนเดียวกัน จะได้ว่า

$n = n', length = length', L = L'$ และ $X = X'$ เมื่อ $X = v, E, l, L, t, n$ หรือ C

Configuration space ของนิ้วจับทั้งสองจะเป็น $[0, L) \times [0, L')$ ถ้าให้ $(u, u') \in C$ จะเรียก (u, u') ว่าเป็น "การจับที่ติด" ก็ต่อเมื่อ $length'(u)$ และ $(length')^{-1}(u')$ เป็นการจับที่มีคุณสมบัติแบบปิดของแรง (ฟังก์ชั่น length เดิมจะรับข้อมูลเป็นจุด และให้คำตอบเป็นจำนวนจริงโดยที่ $length^{-1}$ จะรับข้อมูลเป็นจำนวนจริงและให้คำตอบเป็นจุดผิวของ P)

นิยามให้เซต "การจับที่ติด" $G \subseteq C$ เป็นเซตของท่าจับที่ทำให้การจับเป็นการจับที่ "ติด" (G คือเซตของท่าจับที่อยู่ในบริเวณที่มีคุณสมบัติแบบปิดของแรง) $G_{i,j}$ เป็นซับเซตของเซตที่จับติดของด้าน E_i และ E'_i จะนิยามเป็น

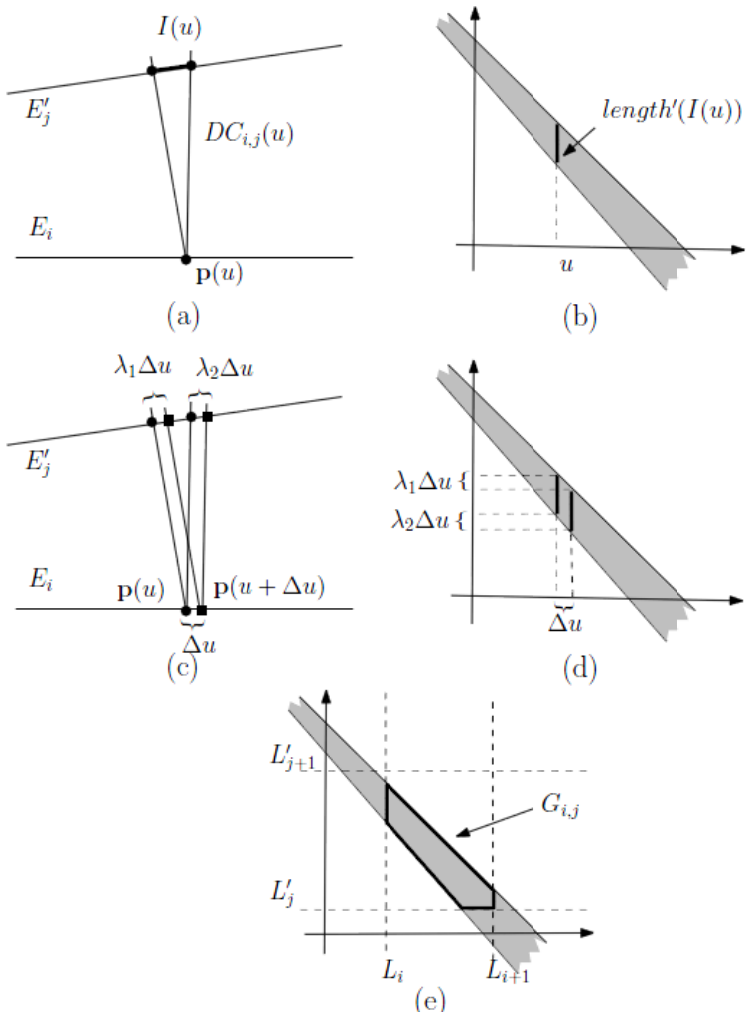
$$G_{i,j} = G \cap ([L_i, L_{i+1}] \times [L'_j, L'_{j+1}])$$

3.1.2 การคำนวณหา $G_{i,j}$

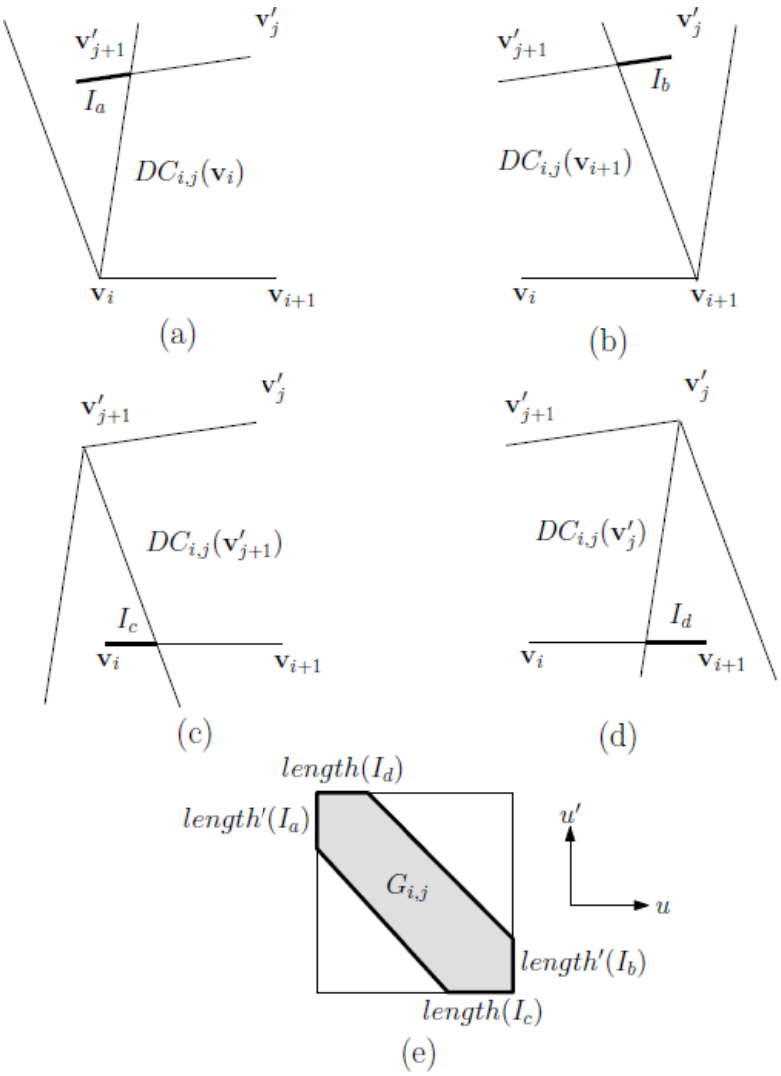
$G_{i,j}$ เป็นท่าจับของการจับที่มีนิ้วหนึ่งอยู่บนด้าน E_i และอีกนิ้วหนึ่งอยู่บนด้าน E'_j ปัญหาการจับที่มีสมบัติปิดของแรง เป็นปัญหาที่ถูกศึกษามาเป็นอย่างดีแล้ว จากประพจน์ที่ 2 ก็ได้มีผู้ทำการศึกษาไว้แล้วใน [27] ว่า $G_{i,j}$ สามารถนิยามได้โดยอสมการ 9 อสมการ บนตัวแปร u, u' เราจะสามารถหาวิธีนิยาม $G_{i,j}$ ได้จากการสร้างกรวยแรงเสียดทานกลับด้าน $-C'_j$ เป็น $\{-x | x \in C'_j\}$ตามที่ได้แสดงไว้ใน [18] ว่า $C_{i,j} = C_i \cap -C'_j$ จะเป็นส่วนที่เป็นที่ว่างของ $G_{i,j}$ ถ้า $C_{i,j}$ ไม่เป็นที่ว่าง $G_{i,j}$ ก็สามารถนิยามได้ด้วยจุดไม่เกิน 6 จุดบนสี่เหลี่ยมที่เป็นขอบเขตของ E_i และ E'_j ซึ่งสามารถอธิบายได้ดังนี้

เนื่องจากการจับด้วยนิ้วสองนิ้วมีสองแบบคือ การจับแบบบีบเข้า และการจับแบบถ่างออก เราจะนิยามกรวยแรงแบบสองด้านของ $C_{i,j}$ เป็น $DC_{i,j}$=$C_{i,j} \cup -C_{i,j}$ เพื่อช่วยในการอธิบายการจับทั้งสองกรณี เริ่มโดยสังเกตุ $DC_{i,j}$ ที่อยู่บนด้าน E_i และขยายด้าน E_i และ E'_j ให้ยาวออกไปเป็นอนันต์ เมื่อเลือกจำนวนจริง u ใดๆ, หา $p(u) = length^{-1}(u)$ บน E_i และให้ $DC_{i,j}(u)$ เป็นกรวย $DC_{i,j}$ ที่มีศูนย์กลางอยู่ที่ $p(u)$ เมื่อนำ E'_j มา intersect กับ $DC_{i,j}(u)$ เราจะได้ส่วนของเส้นตรงบน E'_j ซึ่งแทนจุดจับของนิ้วที่สองที่สามารถวางนิ้วแล้วทำให้เป็นการจับที่มีคุณสมบัติแบบปิดของแรงได้ เราจะเรียกบริเวณดังกล่าวว่า I(u) ดังนั้นถ้ากำหนดตำแหน่งของปลายนิ้วที่หนึ่งที่ตำแหน่ง u จะได้ $length^{-1}(I(u))$ เป็นช่วงคำตอบของบริเวณของนิ้วที่สองที่สามารถวางนิ้วแล้วทำให้เป็นการจับที่มีคุณสมบัติแบบปิดของแรงใน configuration space ตามรูปที่ 18(a,b)

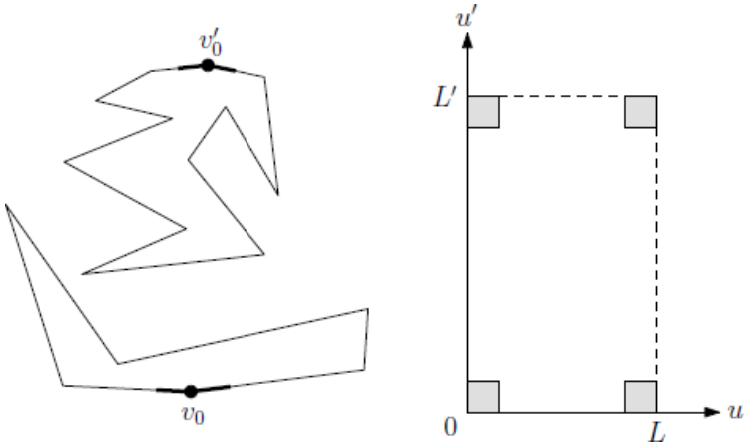
จากภาพจะเห็นได้ว่าถ้าเคลื่อน u ออกไป Δu, $p(u)$ จะเคลื่อนไปในทิศเดียวกันกับ t_i เป็นระยะทางเท่ากับ Δu เช่นกัน จุดปลายของ $\mathrm{I(u)}$ จะเคลื่อนไปในทิศ $-t_i$ เป็นระยะทางที่เป็นอัตราส่วนกับ Δu เช่นกันตามรูปที่ 18(c,d) จากความสัมพันธ์เชิงเส้นนี้ สามารถตีความได้ว่าบริเวณที่จับแล้วมีคุณสมบัติแบบปิดของแรงนั้น ถูกกำกับโดยเส้นตรงสองเส้น การที่เราตัด E_i และ E_j' ที่ทำการต่อให้ยาวออกไปเป็นอนันต์ให้เป็นขนาดเท่าที่มันเป็นอยู่ ก็เทียบได้กับการกำหนดเส้นกำกับสี่เส้นให้กับมัน อสมการของเส้นกำกับที่ว่านี้ได้แก่ $u \geq L_i, u \leq L_{i+1}, u' \geq L_j', u' \leq L_{i+1}'$ ในมิติของ (u, u') (รูปที่ 18(e)) ทำให้ $G_{i,j}$ สามารถกำหนดได้ด้วยจุดไม่เกิน 6 จุดในสี่เหลี่ยมที่กำกับมันไว้ ในทางปฏิบัติจะคำนวณได้จากการหาจุดตัดของ $DC_{i,j}(v_i) \cap E_j'$, $DC_{i,j}(v_{i+1}) \cap E_j'$, $DC_{i,j}(v_j') \cap E_i$ และ $DC_{i,j}(v_{j+1}') \cap E_i$ (รูปที่ 19)



รูปที่ 18 บริเวณที่จับติด



รูปที่ 19 การตัดกันของเส้นกำกับ



รูปที่ 20 บริเวณสัมผัสอิสระที่หลุดออกเป็น 4 ชิ้น

3.1.3 การขยาย configuration space

เนื่องจากนิ้วแต่ละนิ้วสามารถวางตัวได้อย่างอิสระใน บริเวณสัมผัสอิสระและยังทำให้การจับนั้นเป็นการจับที่มีคุณสมบัติแบบปิดของแรง ตำแหน่งของแต่ละนิ้วจับก็สามารถระบุได้ด้วยตัวแปรหนึ่งตัว ดังนั้น บริเวณสัมผัสอิสระใน configuration space ก็จะเป็นสีเหลี่ยมที่ด้านของมันขนานกับแกน u หรือ u' บริเวณสัมผัสอิสระที่ได้จะเป็นสี่เหลี่ยมชิ้นเดียวหรือมากกว่านั้น ขึ้นกับว่าสี่เหลี่ยมที่แทนบริเวณสัมผัสอิสระคร่อม u_0 หรือ u_0' หรือไม่ (รูปที่ 20) เพื่อหลีกเลี่ยงปัญหาของการหาบริเวณสัมผัสอิสระที่เป็นสี่เหลี่ยมหลายรูป เราจึงทำการขยายขอบเขตของ $length^{-1}$ และ $(length')^{-1}$ ออกไปในแนวเส้นจำนวนจริง ซึ่งจะทำให้ทั้งสองฟังก์ชั่นกลายเป็นฟังก์ชั่นคาบที่มีคาบขนาด L และ L' ตามลำดับ ($length$ และ $length'$ จะไม่เป็นฟังก์ชั่นหนึ่งต่อหนึ่งอีกต่อไป) G ที่ได้ตามนิยามใหม่ของ $length$ และ $length'$ นี้ จะขยายออกมาจนเต็ม R^2 ซึ่งสามารถนิยามได้เป็น

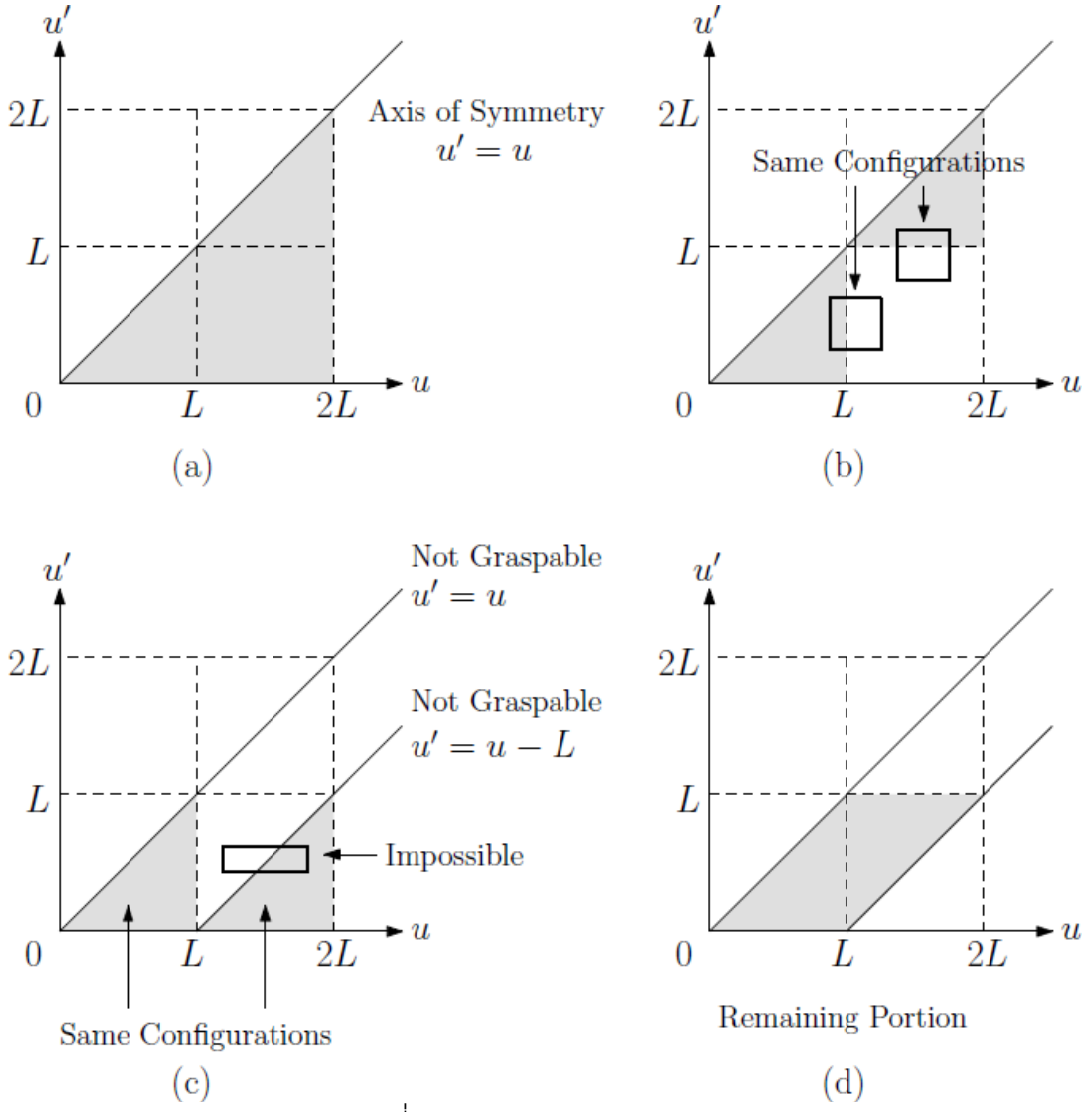
$$(u, u') \in G \Leftrightarrow (u + L, u') \in G$$
$$(u, u') \in G \Leftrightarrow (u, u' + L') \in G$$

ถึงแม้ G แบบใหม่จะมีขนาดเป็นอนันต์ แต่บริเวณสัมผัสอิสระก็จะมีตัวแทนที่เป็นสี่เหลี่ยมใน $G \cap [0,2L] \times [0,2L']$ ซึ่งสามารถพิสูจน์ได้โดย

- ถ้าระบุตำแหน่งของนิ้วที่หนึ่ง จะต้องมีบางจุดวางนิ้วของนิ้วที่สองที่ไม่มีคุณสมบัติแบบปิดของแรงกับนิ้วแรก
- ทำให้ด้าน u_i ใดๆใน G สั้นกว่า L' และ u_j' ใดๆสั้นกว่า L
- เนื่องจากบริเวณสัมผัสอิสระสามารถจับคู่กับบางสี่เหลี่ยมใน G ซึ่งมีด้านขนานกับแกน ดังนั้นจะต้องมีสี่เหลี่ยมที่ต้องการที่อยู่ใน $[0,2L] \times [0,2L']$

ในกรณีพิเศษที่นิ้วจับวัตถุก้อนเดียวกัน จะสามารถใช้ configuration space ที่มีขนาดเล็กกว่าเดิมได้ เนื่องจากในกรณีนี้ G จะสมมาตรรอบแกน $u = u'$ เพราะว่าการสลับ u กับ u' ก็เหมือนการสลับนิ้วจับที่หนึ่งกับนิ้วจับที่สอง ซึ่งก็จะทำให้ได้ท่าจับเดิม (จุดจับไม่เปลี่ยน) ดังนั้น เราสามารถตัด G ทิ้งได้ครึ่งหนึ่งซึ่งอยู่ด้านบน (หรือล่าง) เส้น $u = u'$ (รูป 21(a)) ส่วนของ G ที่อยู่เหนือ $u' > L$ (หรือทางขวาของ $u > L$ สี่เหลี่ยมใน $[0,2L] \times [0, L]$ (หรือ $[0, L] \times [0,2L]$) ซึ่งแทนที่การจับที่ซ้ำๆกันอยู่ ส่วนบริเวณทางขวาของเส้น

$u = u' - L$ (บริเวณที่อยู่สูงกว่าเส้น $u = u' + L$) ก็เป็นข้อมูลที่ซ้ำ เพราะไม่มีจุดใดบนเส้นนี้ที่อยู่ใน G (รูปที่ 21(c)) บริเวณที่ควรนำมาพิจารณาจึงเหลือแต่บริเวณที่แรเงาสีเทา (รูปที่ 21(d))



รูปที่ 21 การขยาย configuration space

3.1.4 การสร้าง G

ถึงจุดนี้ เราก็พอจะทราบแล้วว่าแต่ละ $G_{i,j}$ มีจุดปลายอย่างมาก 6 จุด ดังนั้น $G_{i,j}$ ทุกคู่สามารถสร้างได้ภายในเวลา $O(nn')$ แต่แทนที่เราจะทำการคำนวณทุกคู่ด้าน $\mathbf{i, j}$ เราสามารถใช้วิธีคำนวณตาม [30] ซึ่งเสนอวิธีการคำนวณแบบที่ความซับซ้อนของเวลาคำนวณอยู่ที่ $O(n^{\frac{4}{3}} \log^3 n + K \log K$ เมื่อ K เป็นจำนวนคู่ด้านที่เป็นคำตอบ (กรณีเลวร้ายที่สุด

สำหรับ K คือ $\mathrm{O}(\mathrm{n}^2)$) วิธีคำนวณนี้ก็ใช้ได้เฉพาะกรณีที่จุดจับต้องอยู่อยู่บนวัตถุโพลิกอนชิ้นเดียวกันเท่านั้น

วิธีคำนวณสุดท้ายที่จำเป็นต้องมีคือวิธีการรวมชิ้นส่วนของบริเวณที่จับติดในแต่ละ $G_{i,j}$ ที่อยู่ติดกันให้เป็นโพลิกอนชิ้นเดียวกัน ซึ่งอาจจะได้โพลิกอนหลายชิ้นและไม่จำเป็นต้องติดกันออกมาก็เป็นได้ บางจุดบน $G_{i,j}$ จะเป็นจุดขอบของ G เราสามารถตั้งข้อสังเกตุว่าจุดไหนจะเป็นจุดขอบของ G ได้ดังต่อไปนี้

- v ไม่ใช่มุมของ bounding rectangle
- v เป็นมุมของ bounding rectangle ทั้งสี่ชิ้น แต่ไม่ได้เป็นส่วนหนึ่งของบาง $G_{k,l}$

$G_{i,j}$ ถ้าไม่เป็นช่องว่างหรือมีคุณสมบัติแบบปิดของแรงทั้งก้อน อย่างน้อยจะมีมุมมุมหนึ่งเป็นจุดขอบ วิธีที่จะหาโพลิกอนที่นิยาม G สามารถทำได้โดยเริ่มจากการกำหนดให้ทุกจุดใน $G_{i,j}$ มีสถานะเป็น “ยังไม่โดนใช้งาน” และเป็นจุดขอบของ G

ถ้ามีจุดที่มีสถานะเป็น “ยังไม่โดนใช้งาน” และเป็นจุดขอบของ G

- แสดงว่าจุด v เป็นจุดที่อยู่บนผิวของ G ดังนั้น เราสามารถที่จะไล่หาขอบโดยเริ่มจากจุดนี้ได้
- ทุกจุดที่ไต่ออกจาก v จะเป็นจุดขอบของ G และจะถูกกำหมดสถานะให้เป็นจุดที่ “ถูกใช้งาน”

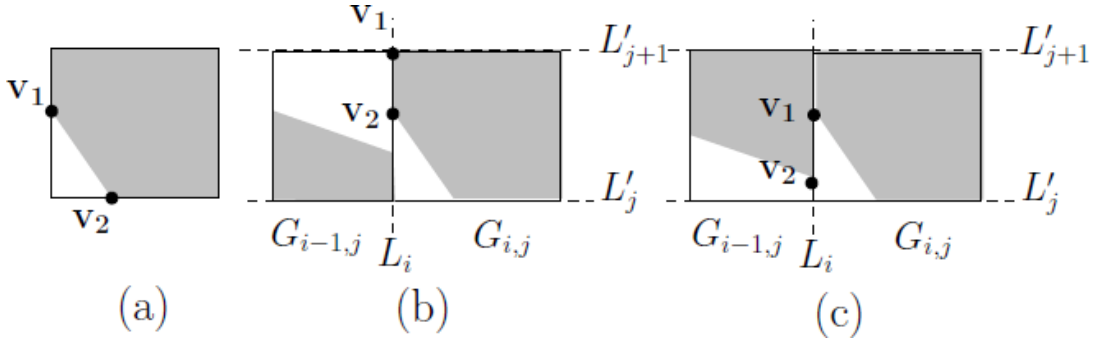
การไล่หาจุดขอบของ G นั้นอาจเกี่ยวข้องกับ $G_{i,j}$ หลายชิ้น วิธีการไล่หาขอบจึงสามารถทำให้ง่ายขึ้นโดยกำหนด “ความติดกัน” ของแต่ละจุด โดยจุด v_1 และ v_2 ที่เป็นจุดขอบของ G จะเรียกว่าอยู่ติดกันตามเงื่อนไขต่อไปนี้

- ถ้า v_1 และ v_2 อยู่ติดกันบนโพลิกอนใน $G_{i,j}$ และอยู่คนละด้านของ $G_{i,j}$ จะเรียกว่า v_1 และ v_2 อยู่ติดกัน (รูป 22(a))
- ถ้า v_1 และ v_2 อยู่ติดกันใน $G_{i,j}$ และอยู่บนด้านเดียวกัน $v_1, v_2 \in \{L_i\} \times [\mathrm{L'_j}, \mathrm{L'_{j+1}}]$ จุด v_1 และ v_2 จะอยู่ติดกันก็ต่อเมื่อ $G_{i-1,j} \cap \overline{v_1 v_2} = \varnothing$ (รูป 22(b))

- ถ้า v_1 และ v_2 อยู่บนคนละชิ้น เช่น $G_{i,j}$ และ $G_{k,l}$ โดยที่ $v_1 \in G_{i,j}$ และ $v_2 \in G_{i-1,j}$; v_1 และ v_2 จะอยู่ติดกันก็ต่อเมื่อ $\{L_i\} \times [L'_j, L'_{j+1}]$ และ
  - $\overline{v_1 v_2} \subseteq G_{i-1,j}$ และ $\overline{v_1 v_2} \cap G_{i,j} = \{v_1\}$ หรือ
  - $\overline{v_1 v_2} \subseteq G_{i,j}$ และ $\overline{v_1 v_2} \cap G_{i-1,j} = \{v_2\}$ (รูป 22(c))

จุดบนขอบของ G นั้น ถ้าไม่เป็นขอบภายนอกก็จะเป็นส่วนหนึ่งของช่องว่างภายใน G จึงจำเป็นที่จะต้องแยกแยะให้ได้ว่าแต่ละจุดนั้นเป็นจุดประเภทใด จากลำดับของจุดที่ไล่หามาได้ ซึ่งเริ่มจากจุด v และไล่ไปจนกว่าจะกลับมาครบรอบที่จุด v ตามเดิม แต่เราก็ไม่ทราบว่าลำดับที่ได้นั้นเรียงตัวอย่างไร ทราบแต่เพียงว่าเป็นโพลิกอนเท่านั้น เราจึงจะเริ่มทดสอบโดยเรียงจุดทวนเข็มนาฬิกา เริ่มจากจุดที่อยู่ไกลที่สุดในทิศ u' ให้เป็นจุด v_i จุดที่ติดกันกับ v_i ให้เป็น $v_{i+1}, v_{i+2}, \ldots$ หลังจากนั้นเราจะทำการคำนวณหา cross product ของ $\overline{v_{i-1} v_i}$ กับ $\overline{v_i v_{i+1}}$ เนื่องจากมุมภายในของจุดไกลสุดนั้นจะทำมุมน้อยกว่า π ลำดับของจุดจะเรียงทวนเข็มนาฬิกา ถ้า cross product เป็นบวก, ถ้า cross product เป็นที่ได้เป็นลบ เราก็จะทำการกลับทิศลำดับของจุดขอบเพื่อให้มันเรียงทวนเข็มนาฬิกา

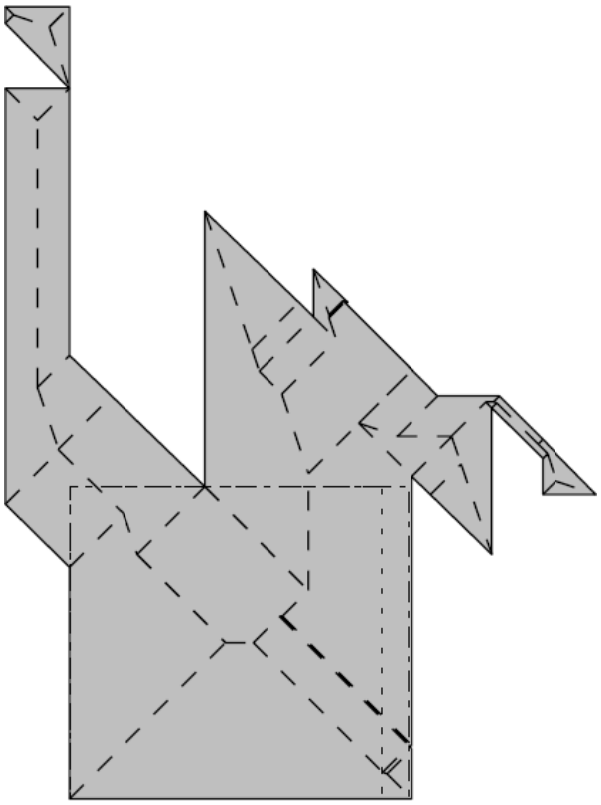
ลำดับที่จัดเรียงใหม่นี้จะเป็นขอบภายนอกของ G ถ้า $\overline{v_i v_{i+1}}$ มีบริเวณที่จับติดอยู่ทางด้านซ้ายและจะเป็นขอบภายใน(ขอบของหลุมที่ว่างกลางโพลิกอน) ถ้ามีบริเวณที่จับติดอยู่ทางด้านขวา ลำดับของจุดที่เป็นขอบของ G สามารถลดจำนวนลงได้โดยทำการกำจัดจุดที่อยู่บนเส้นตรงเดียวกันออกไป เช่น G ประกอบไปด้วยจุด n จุด จุด $v_i \in \Re^2$ เมื่อ $i \in Z_n$ จุด v_i จะถูกกำจัดออกไปได้ถ้าความชันของ $\overline{v_{i-1} v_i}$ เท่ากับ $\overline{v_i v_{i+1}}$



รูปที่ 22 ความติดกันของจุด

3.1.5 บริเวณสัมผัสอิสระที่ดีที่สุด

เมื่อทราบวิธีสร้างกลุ่มโพลิกอนที่แทนบริเวณที่จับติด G แล้ว ถัดไปจะกล่าวถึงวิธีการหาบริเวณสัมผัสอิสระที่ดีที่สุด ตามที่ได้กล่าวไปแล้วในบทที่ 1 ว่า บริเวณสัมผัสอิสระสามารถหาได้โดยคำนวณ L_∞ voronoi diagram ของ G โดยคำนวณแต่เฉพาะส่วนที่อยู่ใน G เท่านั้น โดยจุดศูนย์กลางของสี่เหลี่ยมจตุรัสที่ใหญ่ที่สุดจะอยู่ที่ vertex ของ L_∞ voronoi diagram ของ G (รูปที่ 23) ก่อนที่จะพิสูจน์สมมุติฐานดังกล่าว จะขอกล่าวถึงทฤษฎีของ L_∞ voronoi diagram ก่อน



รูปที่ 23 การขยายสี่เหลี่ยมจตุรัส

โดยหลักการแล้ว L_∞ voronoi diagram ก็เหมือนกับ voronoi diagram ปรกติ แต่ต่างกันตรงวิธีการวัดระยะ แทนที่จะวัดระยะแบบ L2 ตามปรกติ การวัดระยะแบบ L_∞ จะทำการวัดระยะห่างระหว่างสองจุด $\mathrm{p} = (\mathrm{p_x}, \mathrm{p_y})$ และ $q = (\mathrm{q_x}, \mathrm{q_y})$ ใดๆ จะนิยามระยะห่าง $d(p,q)$ เป็น $d(p,q) = max(|\mathrm{p_x} - \mathrm{q_x}|, |\mathrm{p_y} - \mathrm{q_y}|)$ อีกนัยหนึ่งก็คือผลต่างที่มากที่สุดของมิติใดๆระหว่างจุดสองจุดนั้น และนิยามระยะห่างแบบ L_∞ ระหว่างจุด p กับ

เส้นตรง e เป็น $\mathrm{d}(\mathrm{p},\mathrm{e}) = \min_{\mathrm{q}\in\mathrm{e}}\mathrm{d}(\mathrm{p},\mathrm{q})$ กล่าวอีกแบบก็คือระยะห่างที่สั้นที่สุดแบบ L_∞ ระหว่างจุด p และจุดใดๆบนเส้นตรง e

ให้ S เป็นเซตของส่วนของเส้นตรงบนระนาบ vertex ที่อยู่บน L_∞ voronoi diagram ของทุกๆส่วนของเส้นตรงที่เป็นสมาชิกของ S อย่างน้อยจะมีจุดสองจุดที่ห่างเป็นระยะทางแบบ L_∞ น้อยที่สุดกับ vertex นั้น (กล่าวอีกแบบคือเป็นส่วนของเส้นตรงที่ใกล้กับ vertex นั้นมากที่สุดแบบ L_∞ กับ vertex นั้น) เมื่อพิจารณานิยามของการวัดระยะแบบ L_∞, vertex ของ L_∞ voronoi diagram ของ S ก็ต่อเมื่อมันเป็นจุดศูนย์กลางของสี่เหลี่ยมจตุรัสที่แตะกับด้านของ S อย่างน้อยสองด้าน โดยที่สี่เหลี่ยมอันนั้นไม่ตัดกับด้านใดๆของ S

Edge ของ L_∞ voronoi diagram ตามนิยามจะเป็นจุดบน L_∞ voronoi diagram ที่อยู่บนส่วนของเส้นตรงเดียวกัน ส่วน vertex ของ L_∞ voronoi diagram ตามนิยามจะเป็นจุดที่ edge อย่างน้อยสองเส้นมาบรรจบกัน (ทำให้ตีความได้ว่า vertex คือจุดที่ห่างจากด้าน 3 ด้านของ S เป็นระยะทางแบบ L_∞ เท่ากัน) เนื่องจากทุกจุดบน L_∞ voronoi diagram จะอยู่บน edge อย่างน้อยหนึ่งเส้น L_∞ voronoi diagram จึงเป็นโครงข่ายของส่วนของเส้นตรงนั่นเอง กล่าวแบบเฉพาะเจาะจงลงไป L_∞ voronoi diagram จะแบ่ง S ออกเป็นบริเวณ voronoi ย่อยๆ โดยที่แต่ละบริเวณนั้นก็จะสัมผัสกับด้านบน S บริเวณ voronoi บางบริเวณก็จะครอบคลุมด้านทั้งด้านซึ่งแสดงว่าทุกๆจุดบน S ในบริเวณ voronoi นั้นๆใกล้กับด้านที่ถูกครอบคลุมโดยวิธีวัดระยะแบบ L_∞ มากกว่าอื่นๆบน S

เมื่อทราบความรู้พื้นฐานเกี่ยวกับ L_∞ voronoi diagram แล้ว เราก็สามารถที่จะพิสูจน์สมมุติฐานในข้างต้น โดยที่จะขอนิยามฟังก์ชั่น $square(v)$ ให้เป็นสี่เหลี่ยมจตุรัสที่ใหญ่ที่สุดที่มีจุดศูนย์กลางอยู่ที่จุด v และอยู่ภายใน G, นิยาม $size(v)$ เป็นความยาวด้านของ $square(v)$

Lemma 1 สี่เหลี่ยมจตุรัสที่ใหญ่ที่สุดใน G จะมีจุดศูนย์กลางอยู่ที่ vertex ของ L_∞ voronoi diagram

พิสูจน์

ให้ v เป็นจุดใน G ถ้า v อยู่ในบริเวณ voronoi, $square(v)$ จะมีมุมหนึ่งที่แตะกับด้านของ G, การเลื่อน v ให้ห่างออกจากด้านที่แตะอยู่นั้นจะทำให้ $square(v)$ ใหญ่ขึ้น ซึ่งสามารถเลื่อนให้ใหญ่ขึ้นได้เสมอจนกว่าจะชนกับ vertex หรือ edge ของ voronoi

ถ้าจุด v อยู่บน edge ของ voronoi, $square(v)$ จะมีมุมสองมุมอยู่บนด้านสองด้านของ G ซึ่งถ้าด้านทั้งสองขนานกัน edge ของ voronoi ก็จะขนาดกับด้านทั้งสองไปด้วย ถ้าเราเลื่อนจุด v ไปตาม edge ดังกล่าวจนถึง vertex ก็ไม่ทำให้ $square(v)$ เล็กลง

ถ้าด้านทั้งสองไม่ขนานกันเราสามารถเลือกที่จะเลื่อนจุด v ไปยัง vertex ด้านที่ทำให้จุด v ห่างจากด้านทั้งสองมากขึ้น การเลื่อนนี้ทำให้ $\mathrm{square(v)}$ ใหญ่ขึ้นได้เรื่อยๆจนไปถึง vertex ดังกล่าว

ดังนั้นเราพิจาณาเฉพาะสี่เหลี่ยมจตุรัสที่มีจุดศูนย์กลางอยู่ที่ vertex ของ voronoi ก็เพียงพอที่จะหาสี่เหลี่ยมจตุรัสที่ใหญ่ที่สุดบน G ได้ (สี่เหลี่ยม)

E.papadupoulu และ D.T. Lee ได้แสดงวิธีสำหรับคำนวณหา L_∞ voronoi diagram บนโพลิกอนที่สามารถใช้ได้กับโพลิกอนแบบที่มีรูภายในและไม่มีรูภายใน [31] โดยที่โพลิกอนที่มี m จุดจะใช้เวลาในการคำนวณหา L_∞ voronoi diagram ในเวลา $O(m \log m)$ และในขั้นตอนการสร้าง L_∞ voronoi diagram ที่ทั้งสองเสนอมานี้ ยังให้คำตอบของ $size(v)$ สำหรับ vertex ของ voronoi ได้อีกด้วย จึงทำให้เราสามารถหาคำตอบได้ในเวลา $O(m)$ หลังคำนวณหา vertex ของ voronoi ทั้งหมดได้ จากความรู้ที่ว่า $G_{i,j}$ ใดๆสามารถนิยามได้ด้วยจุดอย่างมาก 6 จุด ดังนั้นจำนวนจุดของ G จึงเป็น $O(nn')$ ทำให้ใช้เวลาในการสร้าง L_∞ voronoi diagram เป็น $O(nn' \log(nn'))$ และ $O(nn')$ ในการค้นหาคำตอบ ดังนั้นความซับซ้อนทางเวลาของปัญหาการหาบริเวณสัมผัสอิสระที่ใหญ่ที่สุดของการจับด้วยนิ้วที่มีแรงเสียดทางสองนิ้วบนโพลิกอนจึงเป็น $O(nn' \log(nn'))$

3.1.6 การขยายขอบเขตของสี่เหลี่ยมจตุรัส

เมื่อ v เป็น vertex ของ L_∞ voronoi diagram ของ G, สำหรับบางจุด v จะสามารถขยาย $\mathrm{square(v)}$ ไปในแนวตั้งหรือแนวนอน(ทิศใดทิศหนึ่ง) เพื่อให้ได้คำตอบเป็นสี่เหลี่ยมผืนผ้าที่ยังวางตัวอยู่ภายใน G (ตัวอย่างตามรูปที่ 23 ซึ่งเป็นการขยายในแนวนอน) สี่เหลี่ยมผืนผ้าที่ได้ก็จะเป็นบริเวณสัมผัสอิสระที่มีด้านที่สั้นกว่ายาวเท่ากับ $size(v)$ โดยทั่วไปแล้วก็เป็นเรื่องดีถ้าเราสามารถระบุด้านที่ใหญ่กว่าหรือเล็กกว่าของบริเวณสัมผัสอิสระ เพราะเป็นการบ่งบอกถึงความทนต่อความผิดพลาดของจุดจับนั้นๆได้จริง

จุด v ที่เป็น vertex ของ L_∞ voronoi diagram จะอยู่ห่างเป็นระยะทางที่เท่ากันจากด้านที่แตกต่างกันอย่างน้อยสามด้าน ด้านทั้งสามนี้จะสามารถหาได้ในขั้นตอนการสร้าง L_∞ voronoi diagram และด้านทั้งสามนี้จะเป็นตัวช่วยบอกทิศทางที่สามารถขยายสี่เหลี่ยมจตุรัส

ออกไปได้ซึ่งเป็นไปได้ในสี่ทิศเท่านั้นคือ บน,ล่าง,ซ้าย,ขวา โดยจะขออธิบายถึงการขยายขึ้นด้านบนเป็นตัวอย่าง เพราะการขยายในทิศอื่นนั้นสามารถกระทำได้โดยใช้วิธีการเดียวกัน

$\mathrm{square(v)}$ สามารถเขียนในรูปของ $R = [u_1, u_2] \times [u_1', u_2']$ การขยาย $\mathrm{square(v)}$ ขึ้นด้านบนนั้น ก็เปรียบได้กับการขยาย R ในทิศขึ้น $[u_1, u_2] \times \mathbb{R}$ จนมันชนกับบางจุด $(\mathrm{p}, \mathrm{p}')$ บนผิวของ G ซึ่งจุดดังกล่าวจะอยู่บนด้านของ R และเป็นจุดที่ใกล้ R ที่สุด หรือกล่าวให้ชัดเจนยิ่งขึ้น จุด $(\mathrm{p}, \mathrm{p}')$ คือจุดใน $\partial G \cap ([u_1, u_2] \times [u_1', \infty])$ โดยที่ p' มีค่าน้อยที่สุด เนื่องจาก ∂G เป็นโพลิกอนที่มี m จุด วิธีตรงไปตรงมาในการหา p' ทำได้โดยตรวจสอบทีละจุดซึ่งจะใช้เวลา $O(m)$ เพื่อจะหาคำตอบของการขยาย $[u_1, u_2] \times [u_1', p']$ ควรทราบไว้ว่าถ้าเมื่อใดที่เรามีสี่เหลี่ยมจตุรัสที่ใหญ่ที่สุดหลายๆคำตอบ เราสามารถประหยัดเวลาได้โดยทำการเตรียม interval tree จาก ∂G เอาไว้ก่อน โดยขั้นตอนการทำงานสามารถอธิบายได้ดังนั้น

เริ่มจากเก็บด้านของ ∂G ลงใน interval tree โดยใช้ u ซึ่งเป็นพิกัดแนวนอนเป็นตัวเปรียบเทียบ ขั้นตอนการเตรียม interval tree ดังกล่าวใช้เวลา $O(m \log m)$ เมื่อทำการค้นหา $[u_1, u_2] \times [u_1', u_2']$ เราจะใช้ tree ต้นนี้ในการดึงด้านที่ทับกับ $[u_1, u_2] \times \mathbb{R}$ โดยจะใช้เวลา $O(log\, m + k)$ เมื่อ k เป็นจำนวนของด้านที่ทับกันอยู่กับ $[u_1, u_2] \times \mathbb{R}$ ดังนั้น จุด $(\mathrm{p}, \mathrm{p}')$ สามารถหาได้โดยเปรียบเทียบเฉพาะกับด้านที่ได้ออกมา ซึ่งจะใช้เวลา $O(k)$ การคำนวณหาส่วนขยายจึงสามารถทำได้ในเวลา $O(log\, m + k)$

3.1.7 การวัดความทนทานของการวางนิ้วที่จุดจับ

สำหรับงานจับบางงาน เราอาจจะมีความจำเป็นที่จะต้องจับวัตถุ ณ จุดที่กำหนดมาให้ ในกรณีดังกล่าวจึงเป็นเรื่องดีถ้าทราบว่าการจับวัตถุ ณ จุดนั้นมีความทนต่อความคลาดเคลื่อนเท่าใด เพื่อที่จะสามารถทำการวางแผนการจับที่เพิ่มโอกาสในการจับติดมากยิ่งขึ้น เช่น การเคลื่อนแขนจับเข้ามายังจุดจับในบางทิศทางจะทำให้มือจับอยู่ในตำแหน่งที่จับแล้วสามารถวางตำแหน่งปลายนิ้วจับได้แม่นยำมากกว่า เป็นต้น

ตามที่ได้กล่าวถึงคุณสมบัติของ L_∞ voronoi diagram ไปก่อนหน้านี้ L_∞ voronoi diagram จะแบ่ง G ออกเป็นส่วนๆที่เรียกว่า บริเวณ voronoi ซึ่งแสดงให้เห็นว่าสำหรับท่าจับ $(u, u') \in G$ เราสามารถคำนวณหา $square(u, u')$ โดยหาบริเวณ voronoi ที่มี (u, u') อยู่ภายใน เมื่อหาบริเวณ voronoi ได้แล้ว เราสามารถหา $square(u, u')$ ได้ใน $O(1)$ โดยคำนวณระยะห่างแบบ L_∞ ระหว่าง (u, u') กับด้านที่เกี่ยวข้องกับบริเวณ

voronoi นั้นๆ ในการหาว่า (u, u') อยู่ในบริเวณ voronoi ใดนั้น เราสามารถทดสอบได้โดยทำการทดสอบว่าจุด (u, u') อยู่ในโพลิกอนของบริเวณ voronoi นั้นๆหรือไม่ ซึ่งใช้เวลา $O(m)$ ถ้ามีจุดที่จะตรวจสอบหลายๆจุด เราสามารถที่จะใช้วิธีสำหรับหาตำแหน่งจุดตาม [32] ทำให้วิธีคำนวณหาความทนต่อความคลาดเคลื่อนของจุดจับสามารถทำได้โดยเตรียม L_∞ voronoi diagram ในเวลา $O(m \log m)$ และสามารถคำตอบจาก L_∞ voronoi diagram ที่เตรียมไว้ล่วงหน้าได้ในเวลา $O(m \log m)$

3.1.8 ผลการทดลอง

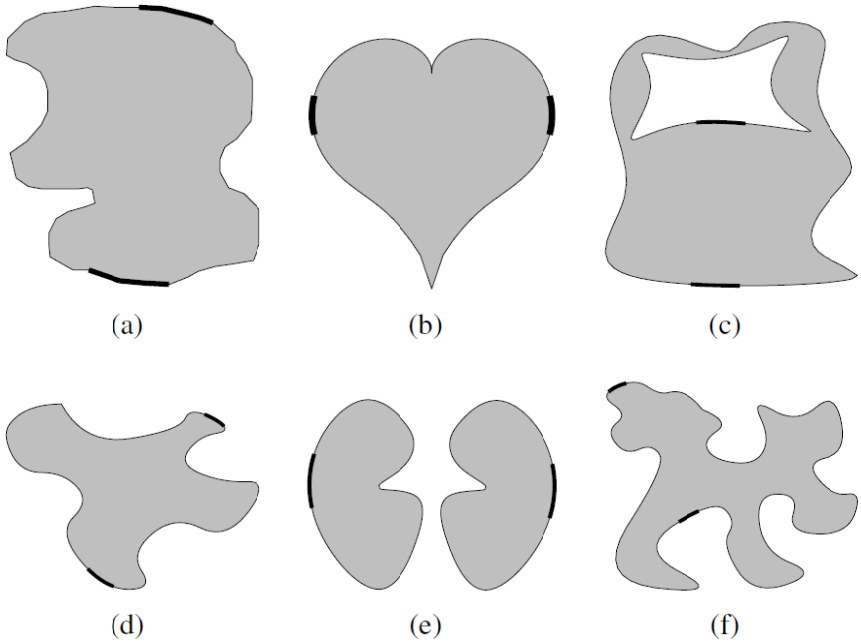
วิธีการที่ได้นำเสนอไปนี้ได้ถูกนำไปเขียนเป็นโปรแกรมภาษาจาวาและนำไปทดสอบบนเครื่องคอมพิวเตอร์ Core i7 920 2.67Ghz, 8 GB RAM เพื่อที่จะแสดงให้เห็นถึงประสิทธิภาพและผลลัพธ์ของวิธีที่นำเสนอ โดยทำการทดลองกับวัตถุโพลิกอนจำนวน 6 ชิ้น ตามที่ได้แสดงไว้ในรูปที่ 24 โดยวัตถุที่นำมาใช้จับนั้นมีจำนวนด้านตั้งแต่ 62 จนถึง 300 ด้าน และมีวัตถุโพลิกอนในรูป 24(c) เป็นตัวอย่างของวัตถุที่มีรู และวัตถุโพลิกอนในรูป 24(e) เป็นตัวอย่างของวัตถุที่มีหลายชิ้น

ผลการทดลองที่ได้ถูกแสดงไว้ในตารางที่ 1 โดยได้ทำการทดลองที่สัมประสิทธิ์แรงเสียดทานต่างกันสามค่าสำหรับแต่ละวัตถุทดสอบ (ระบุโดยครึ่งหนึ่งของมุมของกรวยแรง) ผลลัพธ์ที่ได้จะแสดงอยู่ในรูปของจำนวนด้านบน G ที่มีส่วนเกี่ยวข้องกับการสร้าง L_∞ voronoi diagram รวมถึงเวลาที่ใช้ในการคำนวณในแต่ละกรณีด้วย บริเวณสัมผัสอิสระที่ดีที่สุดที่ได้จากการคำนวณที่ ครึ่งมุมแรงเสียดทานเป็น 15 องศานั้นแสดงไว้เป็นเส้นหนาในรูปที่ 24

เวลาที่ใช้คำนวณนั้นไม่ขึ้นกับปริมาณด้านโดยตรงแต่จะขึ้นกับ G เสียมากกว่าเพราะว่าเวลาที่ใช้ในการหา L_∞ voronoi diagram นั้นเป็นส่วนที่ใช้เวลามากที่สุด กล่าวง่ายๆก็คือถ้า G ที่ได้มีจำนวนด้านเยอะก็จะใช้เวลาในการคำนวณหา L_∞ voronoi diagram นานขึ้น ความสัมพันธ์นี้จะยิ่งชัดเจนมากขึ้นเมื่อทำการเปลี่ยนค่าสัมประสิทธิ์แรงเสียดทานตามที่ได้แสดงไว้ในตารางที่ 1

สำหรับแต่ละตัวอย่างทดสอบ สัมประสิทธิ์แรงเสียดทานที่มากขึ้นก็จะทำให้ G มีหลายด้านมากยิ่งขึ้น (ยิ่งจับได้ง่าย G จะยิ่งมีขนาดใหญ่) และใช้เวลาในการคำนวณมากขึ้น เป็นเรื่องยากที่จะใช้ความสัมพันธ์ดังกล่าวในการเปรียบเทียบเวลาที่ใช้คำนวณระหว่างวัตถุตัวอย่างที่แตกต่างกัน เพราะถึงแม้ว่าจำนวนด้านใน G จะสามารถใช้เป็นตัววัดเวลาคำนวณแบบหยาบๆได้ในหลายกรณี แต่ก็ไม่เป็นจริงเสมอไป วัตถุที่แตกต่างกันสามารถทำให้เกิด G ที่มีรูปร่างที่แตกต่างกันออกไปอย่างมากๆได้ในเชิงเรขาคณิต ซึ่งเป็นปัจจัยหลักที่ทำให้การคำนวณนั้นๆจะได้ผลลัพธ์ออกมาช้าหรือเร็ว อย่างเช่นวัตถุตัวอย่างในรูปที่ 24(e) มีจำนวนด้านมากกว่าวัตถุตัวอย่างในรูปที่

24(f) ค่อนข้างมาก ทั้งนี้เป็นเพราะว่าวัตถุตัวอย่างในรูปที่ 24(e) มีด้านที่ยาวและด้านที่อยู่ติดกันมีความต่อเนื่องค่อนข้างสูง (มีการเปลี่ยนมุมของเส้นตั้งฉากน้อย) จึงทำให้เกิด บริเวณสัมผัสอิสระขนาดใหญ่ในกรณีนี้



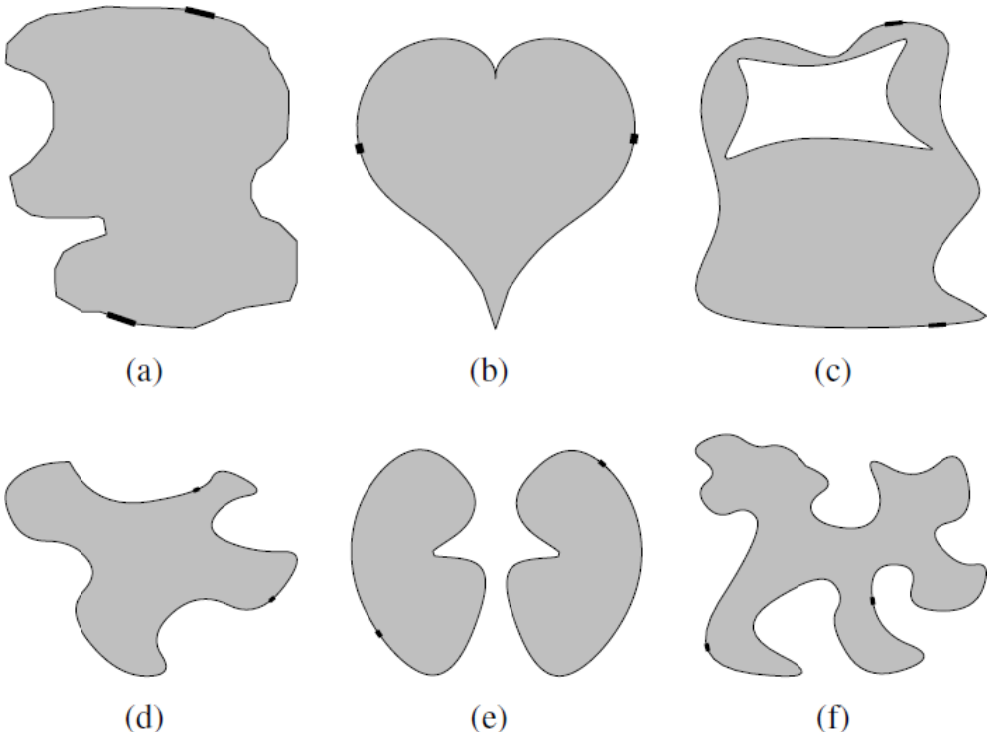
รูปที่ 24 บริเวณสัมผัสอิสระที่คำนวณได้

บริเวณสัมผัสอิสระที่ดีที่สุดที่ได้จากวิธีที่นำเสนอนี้ ซึ่งเป็นวิธีที่สามารถหาบริเวณสัมผัสอิสระที่ให้คำตอบข้ามด้านได้นั้น มักจะได้คำตอบที่ดีกว่าวิธีการหาบริเวณสัมผัสอิสระแบบเดิมๆที่มีข้อจำกัดให้บริเวณสัมผัสอิสระที่เป็นคำตอบนั้น อยู่บนด้านเท่านั้น เพื่อทำการเปรียบเทียบในเชิงปริมาณ เราจึงได้เสนออัตราส่วนระหว่างบริเวณสัมผัสอิสระที่ดีที่สุดจากวิธีของเรากับวิธีแบบเดิมๆ ซึ่งทำการทดสอบบนวัตถุ 6 ชิ้น และได้แสดงผลไว้ในตารางที่ 2 โดยจะเห็นได้ว่าคำตอบที่ได้นั้นมีขนาดใหญ่ขึ้นกว่าเดิมในทุกๆกรณี ซึ่งมีตั้งแต่ 1.51 เท่า ไปจนถึง 10.9 เท่า และเป็นที่น่าสังเกตุว่าอัตราส่วนมีแนวโน้มจะสูงขึ้นสำหรับวัตถุทดสอบที่มีด้านปริมาณมากทั้งนี้เนื่องจากบริเวณสัมผัสอิสระที่ได้จากวิธีดั้งเดิมนั้นถูกจำกัดโดยขนาดของด้านซึ่งมักจะมีขนาดเล็กลงในกรณีที่วัตถุมีดานจำนวนมาก บริเวณสัมผัสอิสระที่ดีที่สุดที่ได้จากวิธีแบบตั้งเดิมที่ครึ่งมุมแรงเสียดทานเท่ากับ 15 องศานั้นแสดงโดยเส้นทึบในรูปที่ 25

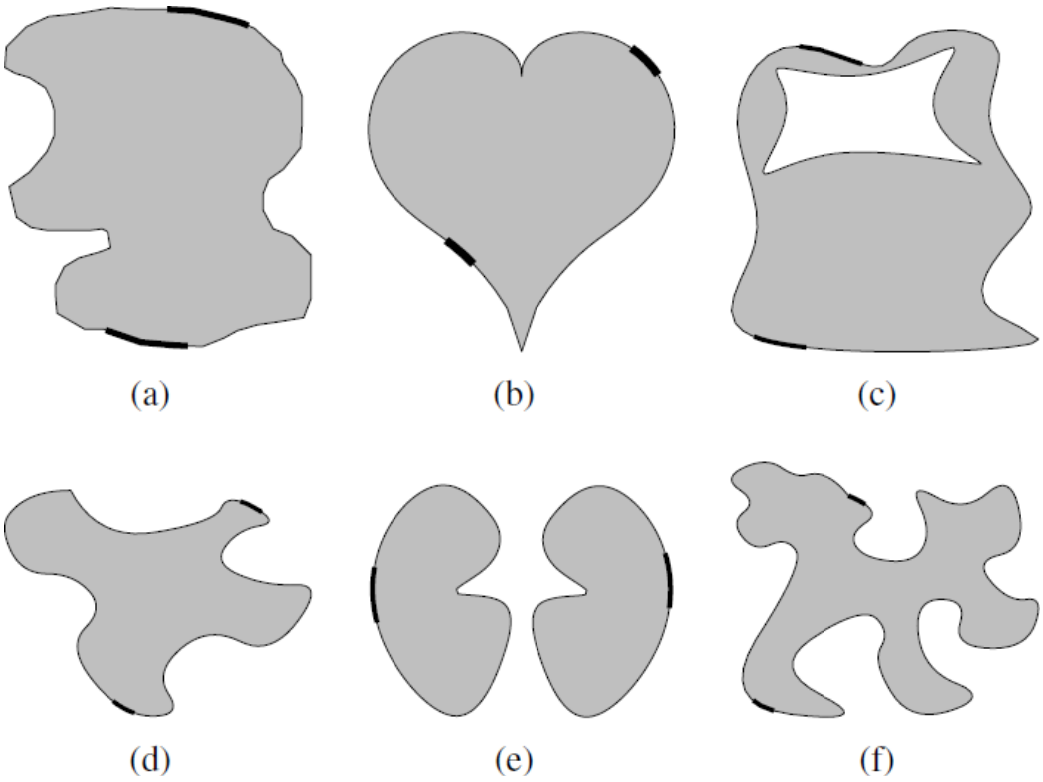
ไม่นานมานี้ ใน [23] Roa และ Saurez ได้เสนอวิธีสร้างบริเวณสัมผัสอิสระจากการจับที่มีคุณสมบัติแบบปิดของแรงที่กำหนดให้บริเวณสัมผัสอิสระสามารถทนเวรนช์ที่ใกระทำในทิศที่สนใจได้ (วิธีที่ 3 ใน paper ของทั้งสองท่าน)

เราจึงนำผลการทดลองมาเปรียบเทียบกับงานของเรา แต่เนื่องจากโมเดลของวัตถุที่ใช้ไม่ได้เป็นโพลิกอนเหมือนกับกรณีของเรา แต่เขาใช้เป็นกลุ่มจุดบนผิววัตถุโดยแต่ละจุดมีเวกเตอร์แสดงทิศที่ตั้งฉากกับผิววัตถุ ณ จุดนั้นๆด้วย บริเวณสัมผัสอิสระสำหรับกรณีการจับวัตถุบนสองมิติของเขาจึงนิยามเป็นกลุ่มจุดที่อยู่ต่อเนื่องกันแทนการใช้เส้นขอบที่อยู่ต่อเนื่องกัน โดยความถูกต้องของการคำนวณจะขึ้นอยู่กับสมมุติฐานที่ว่าการสุ่มจุดนั้นทำได้ละเอียดพอที่จะทำให้บริเวณระหว่างจุดสองจุดที่เป็นส่วนหนึ่งของคำตอบของบริเวณสัมผัสอิสระนั้น เป็นบริเวณสัมผัสอิสระไปด้วย แนวคิดดังกล่าวใช้วิธีการแบ่งมิติของเวรนช์ ด้วยระนาบที่สร้างจากด้านประกอบของ convex hull ของท่าจับเริ่มต้นที่กำหนดให้ และเริ่มทำการค้นหาจุดในระบบบนมิติของเวรนช์ที่ถูกแบ่งเพื่อขยายขนาดของบริเวณสัมผัสอิสระของแต่ละนิ้วจับ

ขนาดของบริเวณสัมผัสอิสระได้จากการวาดจุดของคำตอบที่ได้ลงบนพื้นผิววัตถุ และวัดระยะห่างระหว่างจุดเริ่มกับจุดสุดท้ายของบริเวณสัมผัสอิสระที่เป็นคำตอบ หลังจากได้ความยาวของบริเวณสัมผัสอิสระแล้ว ขนาดของบริเวณสัมผัสอิสระก็สามารถวัดได้โดยวิธีเดียวกันกับวิธีการของเรา วิธีดังกล่าวได้ถูกเขียนเป็นโปรแกรมและนำไปใช้ทดสอบกับวัตถุตัวอย่างทั้ง 6 ชิ้นของเรา โดยทำการสุ่มจุดจากพื้นผิวให้ห่างเท่าๆกันจำนวน 300 และ 500 จุด ตัวแปร alpha ตั้งค่าไว้ที่ 0.0001 เพื่อแทนเหตุการณ์ที่ไม่สนใจเวรนช์ภายนอกมากระทำ (เพื่อให้ได้คำตอบที่มีคุณสมบัติแบบปิดของแรงทั้งหมด) และเพื่อขจัดปัญหาที่เริ่มต้นที่มาจากการสุ่ม เราจึงทดสอบโดยเริ่มคำนวณจากท่าจับทั้งหมดที่เป็นไปได้ คำตอบที่ได้ถูกแสดงเอาไว้ในรูปที่ 26 สำหรับกรณีครึ่งมุมแรงเสียดทานเท่ากับ 15 องศาและสุ่มจำนวน 500 จุด ตารางที่ 2 แสดงอัตราส่วนระหว่างบริเวณสัมผัสอิสระที่ได้จากวิธีของเขาและของเรา



รูปที่ 25 บริเวณสัมผัสอิสระที่คำนวณได้จากวิธีดั้งเดิม



รูปที่ 26 บริเวณสัมผัสอิสระที่คำนวณได้จากวิธีของ Roa & Saurez

ผลการทดลองชี้ให้เห็นว่าวิธีของเขาสามารถให้คำตอบที่เป็นบริเวณสัมผัสอิสระที่มีขนาดใหญ่กว่าวิธีดั้งเดิมได้แต่ก็ยังมีขนาดเล็กกว่าผลจากวิธีที่เราได้นำเสนอในทุกๆกรณีและคำตอบที่ได้นั้นเป็นคำตอบที่ที่ดีที่สุดที่เป็นไปได้แล้วสำหรับวิธีของเขา เพราะเราได้ทำการทดลองกับท่าจับเริ่มต้นทุกๆกรณีที่เป็นไปได้ เป็นที่น่าสังเกตุอีกประเด็นหนึ่งว่า ขนาดของบริเวณสัมผัสอิสระที่ได้มีขนาดใหญ่ขึ้นเมื่อเพิ่มความละเอียดของการสุ่มจุด แต่วิธีนี้ก็ไม่สามารถที่จะนำไปสู่การหาบริเวณสัมผัสอิสระที่ดีที่สุดได้ เนื่องจากเขาได้ใช้เงื่อนไขในการขยายคำตอบของบริเวณสัมผัสอิสระแบบ sufficient เท่านั้น ไม่ใช่เงื่อนไขแบบ necessary ดังนั้น บางจุดที่ช่วยในการขยายบริเวณสัมผัสอิสระจึงไม่ถูกรวมเป็นส่วนหนึ่งของคำตอบ

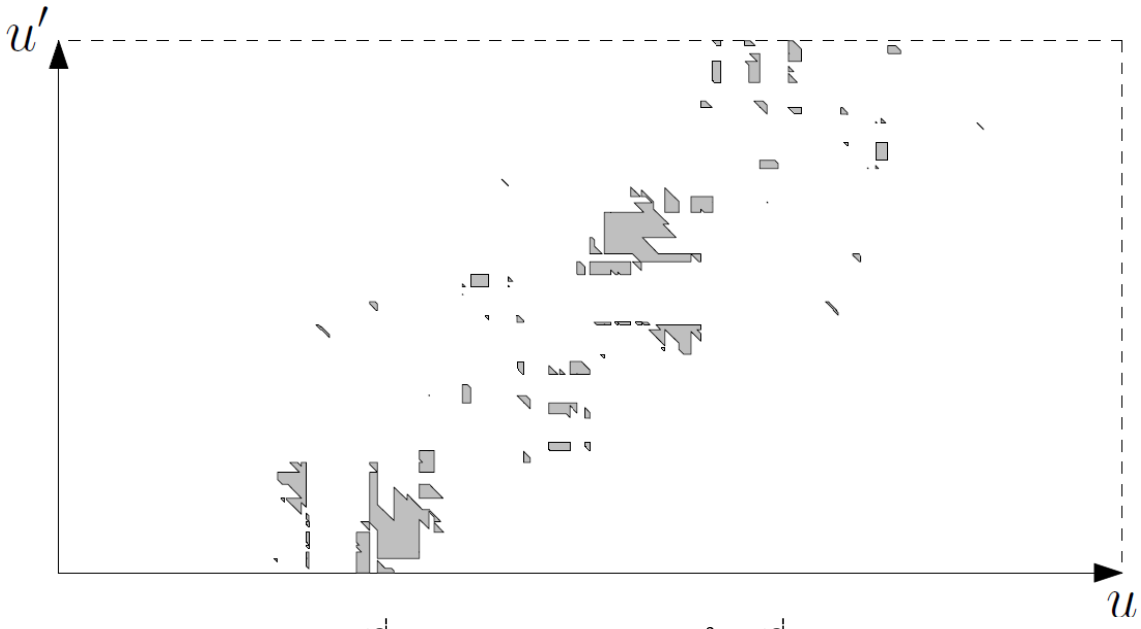
เป็นที่แน่ชัดแล้วว่า วิธีที่ยอมให้คำตอบของบริเวณสัมผัสอิสระข้ามด้านได้นั้น จะให้คำตอบที่มีขนาดใหญ่กว่ามาก ซึ่งวิธีนี้ก็มีประโยชน์อย่างมากในกรณีที่วัตถุนั้นประกอบขึ้นมาจากด้านสั้นๆซึ่งวิธีแบบดั้งเดิมจะให้คำตอบที่ไม่เป็นประโยชน์สักเท่าไหร่ (รูป 25 b,d-f) และก็ชัดเจนว่าวิธีจาก [23] ถึงจะไม่รับประกันว่าจะได้คำตอบที่ดีที่สุดแต่ก็ยังให้คำตอบที่ดีอยู่ อย่างไรก็ตาม วิธีดังกล่าวจะต้องมีท่าจับเริ่มต้นที่ดีอยู่ก่อนแล้วด้วย รวมถึงความละเอียดของการสุ่มก็ต้องเหมาะสม ซึ่งถ้าความละเอียดไม่มากพอ สมมุติฐานของวิธีนี้ก็จะไม่เป็นจริง และทำให้คำตอบออกมาผิดได้ เนื่องจากทุกวันนี้ยังไม่มีวิธีสำหรับเลือกความละเอียดในการสุ่มที่เหมาะสม ทำให้บริเวณสัมผัสอิสระที่คำนวณได้จะต้องนำไปตรวจสอบความถูกต้องอีกทีหนึ่งบนพื้นผิวแบบต่อเนื่อง

Fig.	#edge	10°		15°		20°	
		#seg	time(s)	#seg	time(s)	#seg	time(s)
24(a)	62	325	1.02	461	1.17	546	1.61
24(b)	100	312	0.95	600	2.53	660	2.78
24(c)	156	984	4.72	1372	6.91	1590	8.5
24(d)	200	759	2.7	1104	4.32	1414	5.67
24(e)	245	1544	6.81	1859	9.19	2432	12.63
24(f)	300	2000	5.5	2616	7.8	3225	10.36

ตารางที่ 1 ผลการทดลองสำหรับการจับวัตถุ 2 มิติ

Fig.	Traditional			300			500		
	10°	15°	20°	10°	15°	20°	10°	15°	20°
24(a)	1.51	2.94	3.23	1.16	1.16	1.04	1.1	1.11	1.03
24(b)	2.98	4.63	5.97	1.38	1.43	1.29	1.33	1.39	1.27
24(c)	2.04	3.17	4.35	1.22	1.17	1.15	1.15	1.13	1.11
24(d)	3.13	4.34	4.85	1.23	1.15	1.16	1.07	1.15	1.16
24(e)	5.23	8.17	10.9	1.14	1.2	1.22	1.14	1.2	1.22
24(f)	1.86	2.92	3.56	1.43	1.57	1.6	1.14	1.27	1.39

ตารางที่ 2 อัตราส่วนของบริเวณสัมผัสอิสระ



รูปที่ 27 Grasp Space ของวัตถุในรูปที่ 24 (a)

เช่นเดียวกับวิธีวัดคุณภาพของการจับวิธีอื่นๆ การหาบริเวณสัมผัสอิสระที่ดีที่สุดเพียงอย่างเดียวไม่สามารถแก้ปัญหาทุกอย่างได้ เช่น ท่าจับที่ดีที่สุดอาจเข้าไปทำการจับจริงๆไม่ได้เป็นต้น แต่วิธีที่ได้นำเสนอนี้ไม่เพียงแต่ให้คำตอบเป็นบริเวณสัมผัสอิสระที่ดีที่สุดเท่านั้น แต่ยังได้สร้างมิติของท่าจับเอาไว้อีกแล้วด้วยตามที่ได้เห็นไปในรูปที่ 27 ซึ่งเป็นมิติของท่าจับในรูปที่ 24(a) มิติของท่าจับประกอบไปด้วยบริเวณที่มีคุณสมบัติแบบปิดของแรง (L_∞ voronoi diagram ของบริเวณที่มีคุณสมบัติแบบปิดของแรงที่ใหญ่ที่สุดของรูปที่ 27 ได้ถูกแสดงไว้ในรูปที่ 23) ในเชิงคุณภาพแล้ว ท่าจับที่ต่างกันมีแนวโน้มที่จะอยู่บนบริเวณที่มีคุณสมบัติแบบปิดของแรงคนละ

บริเวณกัน ในกรณีที่ไม่สามารถจับในท่าที่มีสัมผัสอิสระที่ดีที่สุด ผู้ใช้งานสามารถพิจารณาถึงบริเวณสัมผัสอิสระอื่นที่อยู่บนชิ้นเดียวกันหรือคนละบริเวณที่มีคุณสมบัติแบบปิดของแรง ซึ่งการหาคำตอบได้ง่ายเพราะว่า vertex ของ L_∞ voronoi diagram ที่คำนวณเอาไว้นั้นมีขนาดของบริเวณสัมผัสอิสระติดมาอยู่ด้วย ซึ่งพร้อมให้ผู้ใช้งานนำไปทำการวางแผนการจับต่อไป

3.1.9 สรุป

สำหรับวิธีคำนวณหาบริเวณสัมผัสอิสระที่ดีที่สุดสำหรับการจับด้วยนิ้วที่มีแรงเสียดทานจำนวนสองนิ้วโดยวัตถุที่ทำการจับเป็นโพลิกอนสองมิติ คำตอบที่ได้ออกมาสามารถข้ามด้านของวัตถุที่ทำการจับได้ ซึ่งตรงข้ามกับวิธีแบบดั้งเดิม วิธีที่นำเสนอนี้ยังเหมาะสมกับวัตถุที่เป็นโพลิกอนที่ประกอบขึ้นจากด้านขนาดเล็กซึ่งตรงกับข้อเท็จจริงที่ว่าการจะประมาณวัตถุด้วยโพลิกอนให้แม่นยำนั้น ถ้าแทนที่ในส่วนที่เป็นผิวโค้งด้วยด้านขนาดเล็กๆ ก็จะทำให้ผลที่ได้นั้นใกล้เคียงกับวัตถุจริงมากขึ้น[33] การประมาณวัตถุด้วยโพลิกอนที่ใช้ด้านปริมาณมากๆก็เป็นวิธีประมาณวัตถุที่นิยมใช้กัน [14,34,35] โดยเฉพาะวัตถุที่สร้างขึ้นมาจากผลของเครื่องสแกน วิธีการวางแผนการจับที่สามารถใช้กับวัตถุในลักษณะนี้ได้จึงเป็นวิธีที่จำเป็น

3.2 การจับวัตถุ 3 มิติด้วยนิ้วจับ 2 นิ้ว

3.2.1 การอธิบายผิววัตถุด้วย polygon face

บริเวณสัมผัสอิสระนั้นสามารถทำการนิยามได้หลายรูปแบบซึ่งโดยทั่วไปจะมีความสัมพันธ์โดยตรงกับวิธีอธิบายผิวของวัตถุที่จะทำการจับ บางวิธีเลือกที่จะใช้การแทนที่ผิววัตถุด้วยเส้นรอบรูปในกรณีสองมิติ เช่นเดียวกับการหาบริเวณสัมผัสอิสระที่ดีที่สุดของกรณีการจับวัตถุสองมิติที่ได้นำเสนอไป และแทนที่ผิววัตถุด้วยพื้นที่และรูปร่างจริงของ face ของ polyhedral ในกรณีการจับวัตถุสามมิติ ซึ่งไม่สามารถที่จะคลี่ผิววัตถุออกมาให้เป็นระนาบสองมิติโดยที่ยังรักษาความต่อเนื่องได้ หรือถ้าใช้วิธี parameterization (เหมือนกับการมองผิววัตถุเป็นยางยืดและดึงออกจากผิวมาคลี่ให้กลายเป็นพื้นราบ) ในการแบ่งพื้นผิวก็จะทำให้พื้นที่ที่วัดได้กับพื้นที่จริงของผิววัตถุนั้นไม่เท่ากันจึงไม่มีวิธีที่ดีในการจัดการกับปัญหาโดยอธิบายพื้นผิวของวัตถุด้วยพื้นที่จริงได้

แต่ก็ยังมีวิธีการที่เป็นที่นิยมในการอธิบายผิวของวัตถุอีกวิธีหนึ่งก็คือ การใช้จุดตัวอย่างบนพื้นผิวมาอธิบาย ซึ่งมักจะทำโดยการโปรยจุดลงไปบนพื้นผิวด้วยความถี่เท่าๆกัน (uniform resampling) ที่ต้องการและเลือกจุดรวมถึง normal ณ จุดนั้นมาเป็นข้อมูลสำหรับทำการคำนวณ

หรือถ้าจะให้ผลการเลือกจุดตัวแทนออกมาได้ดีกว่านั้นก็สามารถทำได้โดยการ remeshing ตามที่ได้กล่าวไปในบทที่ 2 เมื่อได้จุดที่เป็นตัวแทนของพื้นผิววัตถุแล้วเราก็ต้องหาวิธีนิยามบริเวณสัมผัสอิสระ ซึ่งถ้ามองจากนิยามที่ว่าบริเวณสัมผัสอิสระคือบริเวณที่นิ้วจับสามารถเคลื่อนที่อย่างอิสระและยังทำให้การจับเป็นฟอร์สโคลสเซอร์ได้นั้น ปัญหาก็จะอยู่ที่ว่าระยะทางที่นิ้วจับสามารถเคลื่อนที่ได้อย่างอิสระจะวัดอย่างไรในระบบที่มองผิววัตถุเป็นจุดตัวอย่าง วิธีการหนึ่งที่สามารถทำได้ก็คือใช้วิธีวัดว่าจุดที่อยู่ห่างออกไปจุดจุดศูนย์กลางเป็นระยะทางเท่าใด ถ้าเราอธิบายบริเวณสัมผัสอิสระใดๆ เป็น $I(p_1, p_2, l)$ คือมีจุดจับสองจุด $p_1,\ p_2$ และระยะห่างที่ทุกจุดตัวอย่างที่ห่างจากจุดจับ p_1 น้อยกว่าหรือเท่ากับ l หน่วยทุกจุดจะมีคุณสมบัติแบบปิดของแรงกับ จุดตัวอย่างที่ห่างจากจุดจับ p_2 น้อยกว่าหรือเท่ากับ l หน่วย แต่การวัดระยะ l ในทางปฏิบัตินั้นทำได้ยากเพราะว่าต้องวัดระยะทางตามผิวที่โค้งของวัตถุที่จับ ดังนั้น เราจะใช้วิธีนิยามผิวให้เป็น face สามเหลี่ยมทั้งหมดที่มีขนาดและรูปร่างพอๆกัน และสามเหลี่ยมแต่ละรูปก็จะมีสามเหลี่ยมอื่นๆที่ใช้ด้านร่วมกับตัวมันอยู่สามชิ้นเสมอ ดังนั้นถ้าเราพิจารณาการจับจากทีละคู่จุดเป็นทีละคู่ face และวัดระยะทางด้วยจำนวนครั้งที่กระโดดข้ามจาก face ใดๆไปยัง face อื่นๆ แทนที่การวัดระยะทางจริงก็เป็นเรื่องที่พอจะใช้แทนกันได้ ระบบใหม่ที่เราจะใช้ในการอธิบายถึงบริเวณสัมผัสอิสระ $I(p_1, p_2, l)$ ใดๆ จะหมายถึงบริเวณสัมผัสอิสระที่มีจุดศูนย์กลางอยู่ที่ face p_1 และ p_2 และ face ที่ห่างจาก face p_1 น้อยกว่าหรือเท่ากับ l หน่วยทุก face จะมีคุณสมบัติแบบปิดของแรงกับ face ที่ห่างจาก face p_2 น้อยกว่าหรือเท่ากับ l หน่วย

3.2.2 เซตของ vertex ที่อยู่ห่างจาก vertex ที่สนใจไม่เกินระยะ n หน่วย

เมื่อพิจารณา vertex i ในระบบกราฟที่สร้างจากวัตถุโพลิกอนในข้อ 3.2.1 และ edge ที่แสดงความ เป็น face ที่อยู่ติดกัน เราจะนิยามเซต $Adj_n(i)$ เพื่อแทนเซตของ vertex ในระบบกราฟที่มีระยะจาก vertex i ไม่เกิน n หน่วย โดยระยะจะวัดจากจำนวน edge ที่น้อยที่สุดจาก vertex ใดๆไปยัง vertex n

สิ่งที่สามารถทราบได้ในขั้นตอนการอ่านข้อมูลของโพลิกอนที่ต้องการจับ จะประกอบไปด้วย จำนวน face ทั้งหมด, พิกัดของ vertex ที่เป็นมุมในแต่ละ face รวมถึงความสัมพันธ์ว่า face ใดอยู่ติดกันบ้าง จากข้อมูลเบื้องต้นที่กล่าวมา จะทำให้รู้ Adj_1 สำหรับทุกๆ face ของ polyhedral

สำหรับ $Adj_n(i)$ ก็สามารถหาได้จากการ union $Adj_{n-1}(j)$ ของทุกๆ ของทุกๆ หน้าที่อยู่ติดกับมัน ($\bigcup_{j \in Adj_1(i)} Adj_{n-1}(j)$) $Adj_n(i)$ ก็จะประกอบไปด้วย vertex ทุก vertex ที่มีระยะห่างจาก vertex i น้อยกว่าหรือเท่ากับ n หน่วยเสมอ

พิสูจน์

path จากทุก vertex ที่จะมายัง vertex i จะต้องผ่าน vertex ใด vertex หนึ่งใน $Adj_1(i)$

vertex j ที่ห่าง 2 หน่วยจะต้องผ่าน vertex $\mathrm{k} \in Adj_1(i)$ และอยู่ห่าง k เป็นระยะ 1 หน่วย ดั้งนั้น j อยู่ใน $Adj_1(k)$

vertex j ที่ห่าง 3 หน่วยจะต้องผ่าน vertex $\mathrm{k} \in Adj_1(i)$ และอยู่ห่าง k เป็นระยะ 2 หน่วย ดั้งนั้น j อยู่ใน$Adj_2(k)$

vertex j ที่ห่าง n หน่วยจะต้องผ่าน vertex $\mathrm{k} \in Adj_1(i)$ และอยู่ห่าง k เป็นระยะ $n-1$ หน่วย ดั้งนั้น j อยู่ใน $Adj_{n-1}(k)$

ดังนั้น $Adj_n(i)$ ก็สามารถสร้างได้จาก $\bigcup_{j \in Adj_1(i)} Adj_{n-1}(j)$

3.2.3 เซตของ vertex ที่มีคุณสมบัติแบบปิดของแรงกับทุก vertex ที่อยู่ใน $Adj_n(i)$

เราจะนิยาม $FC_n(i)$ ให้เป็นเซตของ vertex ที่มีคุณสมบัติแบบปิดของแรงกับทุก vertex ใน $Adj_n(i)$ edge ระหว่าง vertex ที่แสดงความมีคุณสมบัติแบบปิดของแรงระหว่างคู่ vertex ทุกๆคู่ ระหว่าง vertex ใน $Adj_n(i)$ และ vertex ใน $FC_0(i)$ สามารถสร้างได้จากการคำนวณคุณสมบัติแบบปิดของแรง ระหว่างทุกคู่ face ที่ vertex ทั้งสองเป็นตัวแทน ส่วน $FC_n(i)$ ใดๆสามารถคำนวณได้จาก การ $\bigcap_{j \in Adj_1(i)} \mathrm{FC}_{n-1}(j)$ ของทุกๆ vertex j ที่เป็นสมาชิกของ $Adj_1(i)$

พิสูจน์

vertex $j \in FC_0(i)$ และ $FC_0(k)$ เมื่อ $\mathrm{k} \in Adj_1(i)$ ดังนั้น $j \in FC_1(i)$ ด้วย

vertex $j \in FC_1(i)$ และ $FC_1(k)$ เมื่อ $\mathrm{k} \in Adj_1(i)$ ดังนั้น $j \in FC_2(i)$ ด้วย

vertex $j \in FC_{n-1}(i)$ และ $FC_{n-1}(k)$ เมื่อ $\mathrm{k} \in Adj_1(i)$ ดังนั้น $j \in FC_n(i)$ ด้วย

3.2.4 บริเวณสัมผัสอิสระ

บริเวณสัมผัสอิสระคือบริเวณที่นิ้วจะสามารถเคลื่อนที่ไปได้อย่างอิสระบนผิวหน้าโพลิกอนของวัตถุ สามมิติที่จะทำการจับโดยที่การจับยังคงคุณสมบัติแบบปิดของแรงเสมอ ในวิทยานิพนธ์ฉบับนี้จะให้คำตอบ ในรูปแบบของคู่ vertex i, j ที่เป็น vertex ศูนย์กลางของบริเวณสัมผัสอิสระขนาดความกว้าง n หน่วย หมายถึงทุกๆหน้า ของวัตถุโพลิกอนที่อยู่ห่างจากหน้าที่ i หรือ j เป็นระยะทางไม่เกิน n หน่วยจะเป็นส่วนหนึ่งของบริเวณสัมผัส อิสระนั้นๆ จากนิยามของ $Adj_n(i)$ และ$FC_n(i)$ จะสามารถอธิบายบริเวณสัมผัสอิสระได้ดังนี้

i, j จะเป็น vertex กึ่งกลางของบริเวณสัมผัสขนาด n ก็ต่อเมื่อ

$Adj_n(i) \in FC_n(j)$ หรือ $Adj_n(i) \in FC_n(i)$

อธิบาย

$\forall \mathrm{k}; \mathrm{k} \in Adj_n(i)$ และ $k \in FC_n(j)$; เนื่องจาก $k \in FC_n(j)$ ดังนั้น ทุกๆ vertex ใน $Adj_n(i)$ จะมีคุณสมบัติแบบปิดของแรง กับทุกๆ vertex ใน $Adj_n(j)$

3.2.5 ขั้นตอนการทำงานของ algorithm

เป็นที่ทราบกันอยู่แล้วว่าการเขียนโปรแกรมด้วย CUDA นั้น ถ้าต้องการให้การทำงานมีประสิทธิภาพมีข้อกำกัดอยู่หลายอย่าง เนื่องจากสถาปัตยกรรม SIMD ของการ์ดแสดงผลซึ่งทำงานทีละ instruction และถ้ามีการ branch ไปยังคนละคำสั่งกัน (ผลจาก if ออกมาต่างกันในบาง thread) ก็จะทำการหยุด thread กลุ่มหนึ่งไว้ และแยกกันทำงานคนละช่วงจนกว่าจะกลับมาการทำงานจะกลับมารวมกัน ณ จุดใดจุดหนึ่ง ซึ่งจุดนี้จึงเป็นสาเหตุที่เลือกใช้การอธิบายผิววัตถุด้วย face ที่มีด้านที่อยู่ติดกันจำนวนคงที่จะทำให้การทำงานของโปรแกรมไม่มีการ branch ไปในตัวด้วย

ข้อมูลของ polyhedral ที่ใช้เป็น input ในวิทยานิพนธ์ฉบับนี้ จะนำไปทำ mesh resampling เพื่อให้ เป็น triangluar mesh (แต่ละหน้าจะเป็น polygon 3 เหลี่ยมเสมอ) และแต่ละ

ชิ้นที่ได้มีขนาดพอๆกัน สำหรับการนิยามความเป็นหน้าที่อยู่ติดกันนั้น (adjacency) จะให้หน้าที่มีด้านร่วมกันถือว่าเป็นหน้าที่อยู่ติด กัน เพื่อให้จำนวน$Adj_0(n)$ ของทุกหน้ามีปริมาณคงที่ ซึ่งจะทำให้ขั้นตอนการหา $FC_n(i)$ และ $Adj_n(i)$ ใดๆ สามารถคำนวณได้รวดเร็ว

หลังจากอ่านข้อมูลจาก file แล้ว จะทำการจอง memory บนการ์ดจอสำหรับเก็บ Adj_n และ FC_n ใดๆ รวมถึงพื้นที่สำหรับเก็บจุดและด้านประกอบของแต่ละหน้ารวมถึง Adj_0 ซึ่งได้จากขั้นตอนการเตรียม ข้อมูลนำเข้าด้วย (สาเหตุที่จะต้องเก็บ Adj_0 แยกเอาไว้นั้น เพราะ Adj_0 ก็คือตัวอ้างถึงความเป็น adjacency และในขั้นตอนการหา Adj_n และ FC_n จะต้องใช้ index อ้างอิงถึง adjacency อยู่ทุกขั้นการคำนวณ)

kernel แรกที่จะใช้งานคือ kernel สำหรับคำนวณความเป็น force closure สำหรับทุกคู่หน้า polygon ในระบบ, การคำนวณจะใช้ทฤษฎีที่ว่าคู่จุดใดที่เป็น force closure ซึ่งกันและกัน จุดตรงข้ามจะอยู่ในกรวยแรงของอีกจุดหนึ่งเสมอ ซึ่งเราจะประยุกต์ให้เข้ากับความเป็นหน้าสามเหลี่ยม ของ polygon ในระบบโดยเพิ่มเงื่อนไขเป็น “จุดยอดของ polygon ทั้ง 3 จุดจะต้องอยู่ใน intersection ของทั้ง 3 กรวยแรงของ polygon อีกชิ้น ถึงจะเรียกว่า polygon ทั้ง 2 ชิ้นเป็น force closure ซึ่งกันและกัน” ผลที่ได้ จากการคำนวณนี จะถูกเก็บเอาไว้ใน array ชื่อ fc ขนาด n×n โดยที่ fc[i,j] ใดๆจะแทนค่าความเป็น force clousure ระหว่างจุด i และ j นั้นๆ (1=เป็น force closure) เราจะเห็นได้ว่า fc array ที่ได้จากการคำนวณนี้ เมื่อพิจารณาแยกทีละ แถว มันก็จะคือ FC_0 ของแต่ละ face นั่นเอง

ในขั้นตอนถัดไป เราจะทำการสร้าง Adj_0 ซึ่งก็จะเป็น array ขนาด nxn เช่นเดียวกันกับ fc array โดยจะกำหนดค่าให้ Adj array [i,j]=1 เมื่อ face i,j เป็นหน้าที่อยู่ติดกัน

kernel ที่ใช้สำหรับหา FC_n และ Adj_n นั้น ในวิทยานิพนธ์ฉบับนี้จะขอเรียกว่า intersectKernel และ unionKernel เนื่องมาจากพฤติกรรมของ การหา $FC_{n+1}(i)$ ที่มาจากการ $\bigcap_{j\in Adj_0(i)} FC_n(j)$ และ การหา $Adj_{n+1}(i)$ ที่มาจากการ $\bigcup_{j\in Adj_0(i)} Adj_n(j)$

kernel สุดท้ายที่จะกล่าวถึงคือ kernel สำหรับ check ว่าสำหรับ interation ที่ n ยังมี independent contact region อยู่หรือไม่ โดย checkKernel จะตรวจสอบว่าสำหรับ face i ใดๆ ยังมี $Adj_l(j) \subset FC_l(i)$ และ $Adj_l(i) \subset FC_l(j)$ อยู่หรือไม่ ถ้ายังมีเราสามารถที่จะหยุดการทดสอบคำตอบกับ face ที่เหลือได้(เนื่องจาก คู่ i,j นี้ก็เป็นคำ ตอบในระดับที่ n) ถ้า ไม่มี $Adj_n(j)$ ใดๆที่ตรงตามเงื่อนไขนี้ ก็แสดงว่าสำหรับ node i ไม่มีคำตอบของ independent contact region ที่ใหญ่กว่าหรือเท่ากับระดับ n อีกแล้ว เราจะทำการ mark face i นั้นๆเอาไว้ เพื่อละการทดสอบในรอบถัดๆไป

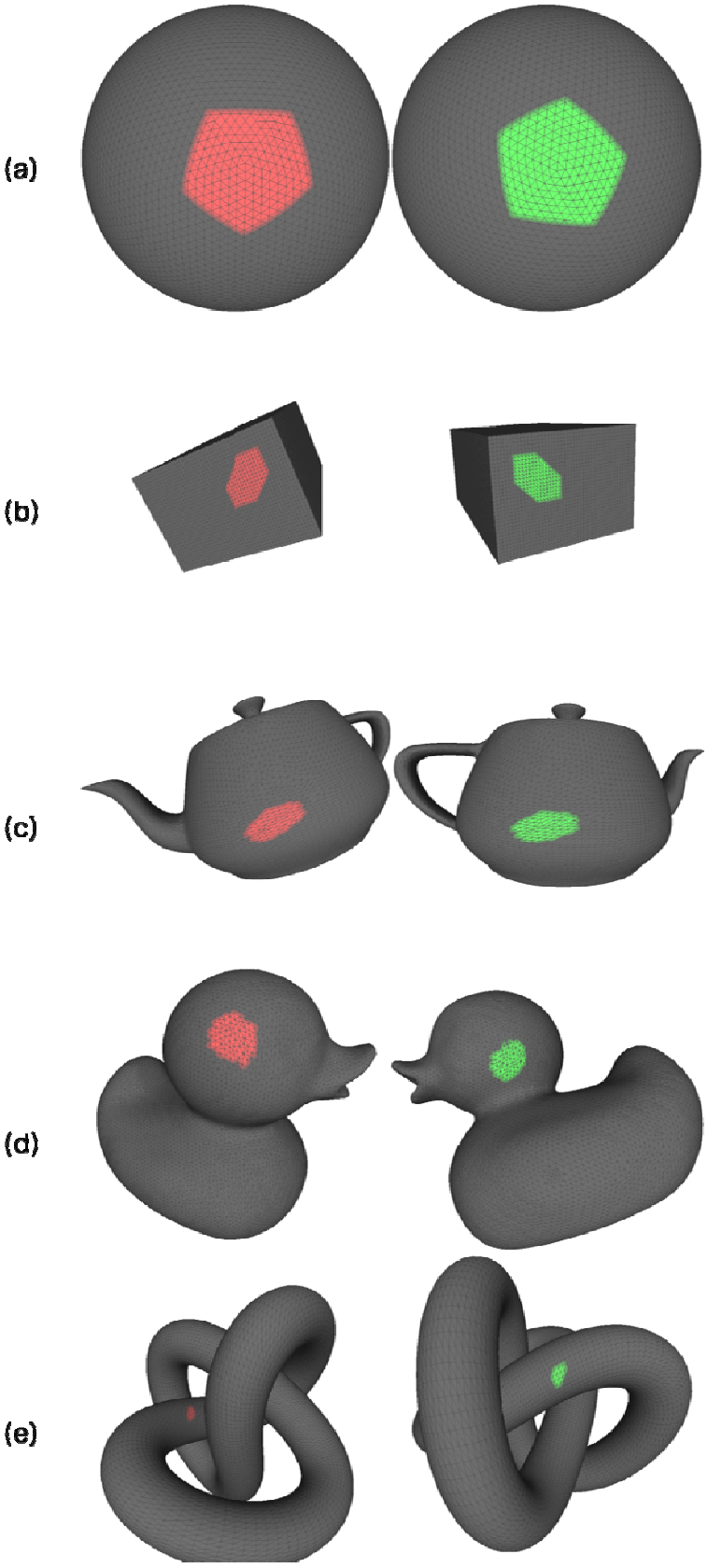
การทดสอบทั้งหมดที่กล่าวถึง สามารถเขียนเป็นขั้นตอนสั้นๆได้ดังนี้

```
fcKernel
intersectKernel
while(checkKernel==true){
 intersectKernel
 unionKernel
}
report result
```

3.2.6 ผลการทดลอง

วิธีการที่ได้นำเสนอไปนี้ได้ถูกนำไปเขียนเป็นโปรแกรมนำไปทดสอบบนเครื่องคอมพิวเตอร์ Core i7 980 3.33Ghz, 8 GB RAM เพื่อที่จะแสดงให้เห็นถึงประสิทธิภาพและผลลัพธ์ของวิธีที่นำเสนอ โดยทำการทดลองกับวัตถุโพลิกอนจำนวน 5 ชิ้น โดยแต่ละชิ้นนั้นจะทำการ remesh เพื่อให้มีจำนวนหน้าใกล้เคียงกันที่ 2 ความละเอียดคือ 15,000 face และ 30,000 face (ยกเว้นทรงกลมแบบละเอียด ซึ่งทำการสุ่มที่ 20,000 face) และทำการคำนวณเทียบกับโปรแกรมเดียวกันที่เขียนขึ้นด้วยภาษา C++ และทำงานบน CPU ตามปรกติ โดยวัตถุที่นำมาใช้จับนั้นมีความซับซ้อนของรูปร่างที่แตกต่างกันออกไป

ผลการทดลองที่ได้ถูกแสดงไว้ในตารางที่ 3 โดยได้ทำการทดลองที่สัมประสิทธิ์แรงเสียดทานต่างกันสามค่าสำหรับแต่ละวัตถุทดสอบ (ระบุโดยครึ่งหนึ่งของมุมของกรวยแรง) ผลลัพธ์ที่ได้จะแสดงอยู่ในรูปของเวลาที่ใช้ในการคำนวณและบริเวณสัมผัสอิสระที่ดีที่สุดที่ได้จากการคำนวณที่ ครึ่งมุมแรงเสียดทานเป็น 15 องศานั้นแสดงไว้เป็นบริเวณสีที่อ่อนกว่าในรูปที่ 24

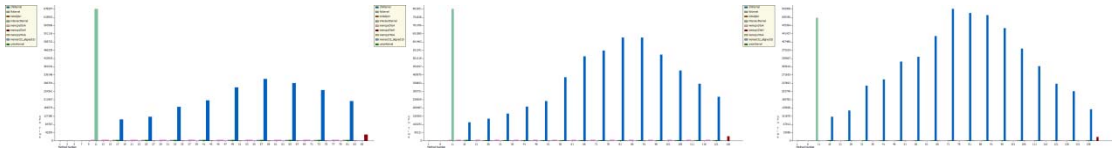


รูปที่ 28 บริเวณสัมผัสอิสระของการจับวัตถุ 3 มิติ

Fig.	CPU			GPU		
	10°	15°	20°	10°	15°	20°
sphere 15k	22	58	131	1.44	3.14	5.71
sphere 20k	47	133	302	2.85	6.24	11.73
box 15k	12	19	34	0.74	1.53	2.94
box 30k	49	96	204	3.46	7.34	------
teapot 15k	9	12	19	0.75	1.94	3.27
teapot 30k	34	54	98	4.14	------	------
duck 15k	9	11	16	0.59	1.28	2.39
duck 30k	34	47	76	2.66	5.97	11.34
knot 15k	8	9	9	0.31	0.47	0.69
knot 30k	30	32	34	1.43	1.57	1.6

ตารางที่ 3 ผลการทดลองสำหรับการจับวัตถุ 3 มิติ

เวลาที่ใช้คำนวณนั้นขึ้นอยู่กับปริมาณ face และรูปทรงของวัตถุเป็นหลัก เพราะว่าเราใช้วิธีคำนวณคุณสมบัติแบบปิดของแรงสำหรับทุกคู่ face ในตอนเริ่มต้นหาคำตอบ ปริมาณ face ที่เพิ่มขึ้นจึงส่งผลโดยตรงกับเวลาที่ใช้ในการคำนวณ ส่วนรูปทรงของวัตถุจะส่งผลต่อปริมาณ edge ความสัมพันธ์ของคุณสมบัติแบบปิดของแรงระหว่างคู่ face ตัวอย่างเช่น ทรงกลมเป็นรูปทรงที่แต่ละ face มีคู่ face ที่มีคุณสมบัติแบบปิดของแรงจำนวนมากเมื่อเทียบกับวัตถุที่มีลักษณะเป็นระยางหรือมีลักษณะเป็นกิ่งก้าน ขั้นตอนการตรวจสอบคำตอบในแต่ละรอบหลังขั้นตอนการขยายพื้นที่ของบริเวณสัมผัสอิสระจะใช้เวลามาก ถ้าด้านที่สามารถเป็นคำตอบในรอบนั้นๆอยู่แบบกระจายตัวกันเนื่องจากแต่ละ thread ใน แต่ละ block เกิดการ branch ขึ้น แตกต่างกับกรณีที่ face ใกล้ๆกันมีการ branch ที่เหมือนๆกัน เช่น ในรอบแรกของการตรวจสอบ ซึ่ง face จำนวนมากยังมีคู่ face ที่มีคุณสมบัติปิดของแรงหรือในรอบหลังๆซึ่ง face ส่วนใหญ่ไม่มีคู่ face ประเภทนี้เหลือแล้ว เมื่อสัมประสิทธิ์แรงเสียดทานที่มากขึ้นก็จะทำให้จำนวนคู่ face ที่มีคุณสมบัติปิดของแรงมีมากยิ่งขึ้น เวลาที่ใช้ในการตรวจสอบคำตอบจึงมากตามจำนวนคู่ไปด้วย



รูปที่ 29 เวลาที่ใช้ในการจับทรงกลมที่ครึ่งมุมแรงเสียดทานเป็น $10, 15, 20$ องศา

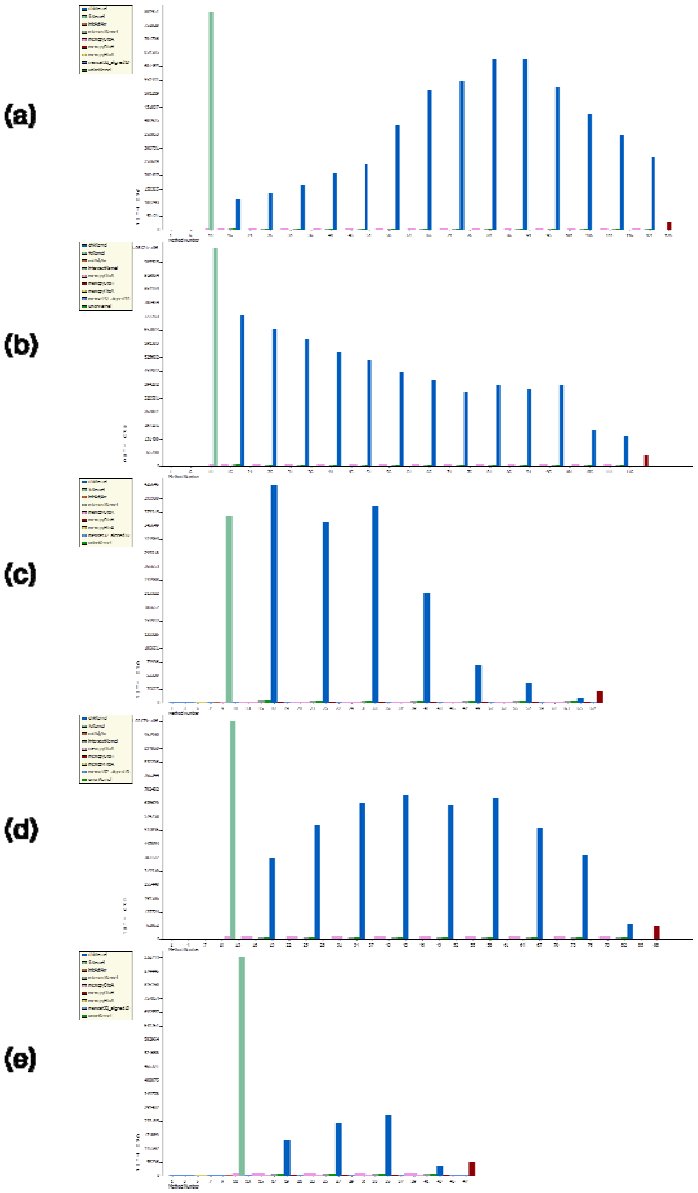
จากรูปที่ 30(a) ซึ่งเป็นเวลาที่ใช้ในแต่ละขั้นตอนของการคำนวณหาบริเวณสัมผัสอิสระของทรงกลม เนื่องจากขนาดของ face, จำนวนคู่ face ที่มีคุณสมบัติปิดของการจับ มีขนาดเท่ากันสำหรับทุก face ในแต่ละขั้นตอนการคำนวณ แต่ละ face จึงมี Adj_n และ FC_n เท่าๆกันกับ face อื่นเสมอ จึงจะขอใช้การหาบริเวณสัมผัสอิสระของทรงกลมมาอธิบายเวลาในการทำงานของวิธีที่นำเสนอ ตามกราฟที่ได้ออกมา จะเห็นว่าการทำงานเริ่มต้นจากส่งข้อมูลของ vertex และ face ขึ้นไปบน GPU และทำการสร้าง Adj_0 ในขั้นตอนถัดมาจะเป็นการคำนวณสมบัติปิดของการจับระหว่างคู่ face ทุกคู่ ซึ่งจะเห็นได้ว่าเป็นขั้นตอนที่ใช้เวลามากตามปริมาณของ face ขั้นตอนถัดมาจะเห็นการทำงานที่มีลักษณะเป็นรอบ ซึ่งประกอบไปด้วยการ union Adj set และ intersect FC set และตรวจสอบคำตอบ

ในขั้นตอนการ union และ intersect นั้น จะเห็นได้ว่าใช้เวลาน้อยมากเมื่อเทียบกับการคำนวณสมบัติปิดของการจับและการตรวจคำตอบ ที่ผลออกมาลักษณะนี้เกิดจากสถาปัตย์กรรม CUDA ซึ่งทำงานที่ไม่มีการ branch ได้ดีและถ้าข้อมูลที่จัดการมีการเรียงตัวที่ดีด้วยแล้ว เวลาที่ใช้ก็จะยิ่งลดลงอย่างมากอีกด้วย ซึ่ง unionKernel และ intersectKernel มีคุณสมบัติตรงตามที่กล่าวมา ดังนั้นถึงจะทำงานกับข้อมูลที่ใปริมาณมากๆก็ยังสามารถทำงานได้อย่างรวดเร็ว ต่างกับ fcKernel และ chkKernel ที่มีลักษณะการอ่านข้อมูลที่มีกระจายตัวสูงจึงทำให้เกิด cache missed ซึ่งการอ่านข้อมูลในกรณี cache missed จาก texture memory นั้นจะเสียเวลาอ่านมากกว่าการอ่านข้อมูลจาก global memory ตรงๆดังนั้น เมื่อเกิด cache missed มากถึงปริมาณหนึ่งก็จะทำให้การคำนวณช้าลงอย่างมากเช่นกัน

ในแต่ละรอบทำงานของ chkKernel ซึ่งเป็น kernel สำหรับตรวจสอบว่ายังมีคู่ face ที่เป็นคำตอบในรอบนั้นๆหรือไม่ มีเวลาที่ใช้ในแต่ละรอบของแต่ละ face จะขึ้นอยู่กับจำนวน Adj_n และ FC_n ถ้า Adj_n มีสมาชิก i ตัว และ FC_n มีสมาชิก j ตัว เวลาที่ใช้คำนวณจะเป็น $O(ij)$ จากกราฟจึงเห็นได้ว่าเวลาที่ chkKernel ใช้นั้น จะมีลักษณะเป็นระฆังคว่ำ เนื่องจากในรอบแรกๆ Adj_n มีจำนวนสมาชิกน้อยแต่ FC_n มีจำนวนสมาชิกมาก และในรอบหลังๆ Adj_n มีจำนวนสมาชิกมากแต่ FC_n มีจำนวนสมาชิกน้อย

เนื่องจากเราตั้งใจสร้างให้ face ใดๆมี adjacent face เท่ากับ 3 เสมอ ดังนั้นไม่ว่าวัตถุจะมีรูปร่างอย่างไรจำนวนสมาชิกของ Adj set ของแต่ละวัตถุในแต่ละรอบจึงเพิ่มขึ้นพอๆกัน แต่สมาชิกของ FC set นั้นเกิดจากการ intersect ดังนั้นจำนวนสมาชิกจะลดลงอย่างรวดเร็วถ้า face ที่ใกล้ๆกันนั้นมีแนวเส้นตั้งฉากที่ทำมุมต่างกันมาก หรือกล่าวอีกแง่หนึ่งก็คือวัตถุที่มีการเปลี่ยนรูปร่างอย่างรวดเร็วเช่น มีการบิดหักงอ หรือวัตถุที่มีลักษณะเป็นท่อหรือกิ่งเล็กๆ เนื่องจากความเล็กจึงทำให้ face ที่มีจำนวนไม่มากเมื่อเทียบกับขนาด จึงต้องบิดตัวค่อนข้างมาก มุมจึงเปลี่ยนมาก วัตถุที่มีลักษณะเป็นระยางหรือมีความบิดเบี้ยวสูงรวมถึงวัตถุที่เป็นเหลี่ยมมุมจึงใช้เวลาในการคำนวณน้อยกว่าวัตถุที่มุมระหว่างแนวเส้นตั้งฉากมีการเปลี่ยนแปลงไม่มาก

เมื่อเปรียบเทียบเวลาที่ใช้ในการคำนวณระหว่างวัตถุคนละชิ้นที่มีปริมาณ face พอๆกันก็จะพบว่า เวลาที่ใช้ของ fcKernel นั้นจะใกล้เคียงกัน จะมากหรือน้อยกว่ากรณีทั่วไปตามปริมาณคู่ที่มีสมบัติปิดของการจับ



รูปที่ 30 เวลาที่ใช้ในการคำนวณของวัตถุในรูปที่ 28

3.2.7 สรุป

วิธีคำนวณหาบริเวณสัมผัสอิสระที่ได้เสนอไปนี้ค่อนข้างมีความยืดหยุ่นสูงในการนำไปประยุกต์ใช้งาน เช่น สำหรับ polytope ที่ทำการ sampling พื้นผิวมาในลักษณะของจุดและ normal ก็ สามารถที่จะใช้จุดแทน face และใช้ระยะระหว่างจุดที่ sampling ได้มาทำการนิยามความเป็น adjacency ของแต่ละ face แทน หรือกรณีที่ทราบถึงข้อจำกัดของมือจับ เช่นมือจับไม่สามารถกางออกได้มากกว่าค่า ค่าหนึ่ง ในขั้นตอนการคำนวณความเป็น force closure ของ fcKernel ก็สามารถที่จะกำหนดว่าคู่ face ที่ห่างเกินค่านั้นๆไม่สามารถเป็น force closure ได้

(เพราะจับไม่ได้จริง) หรือในกรณีที่มีข้อมูลของชิ้น polytope ไม่ครบ, ทำให้บาง face เป็นชิ้นขอบ เราก็สามารถนิยาม dummy face ซึ่งกำหนดให้ $Adj_n(dummy)$ เป็น Ø และ $FC_n(dummy)$ เป็น Ø เพื่อทำให้ค่าความเป็นขอบ (เป็น Ø สามารถแพร่ออก ไปยัง face ถัดไปได้ถูกต้อง)

บทที่ 4

สรุปการวิจัย ข้อเสนอแนะ และงานวิจัยในอนาคต

4.1 สรุปการวิจัย

ในปัจจุบันหุ่นยนต์ได้เข้ามามีบทบาทกับชีวิตประจำวันของมนุษย์มากยิ่งขึ้นเรื่อยๆ ไม่ว่าจะเป็นงานประกอบรถยนต์ ประกอบเครื่องใช้ไฟฟ้า หรือในงานอุตสาหกรรมต่างๆ รวมไปถึงความพยายามในการให้หุ่นยนต์เข้ามาทำงานบ้านบางอย่าง เช่นการดูแลเด็กและผู้สูงอายุ ซึ่งหน้าที่เหล่านี้ต้องมีการหยิบจับวัตถุมาเกี่ยวข้องอยู่เสมอๆ แต่เมื่อเปรียบเทียบความสามารถในการหยิบจับวัตถุของหุ่นยนต์ยังแย่กว่ามนุษย์อยู่มาก การที่หุ่นยนต์จะสามารถจับสิ่งของได้นั้น หุ่นยนต์จะต้องเคลื่อนข้อต่อต่างๆ เริ่มจากส่วนแขนในแต่ละท่อนจนไปถึงมือและข้อต่อของนิ้วจับ การเคลื่อนที่เหล่านี้มีโอกาสเกิดความคลาดเคลื่อนได้เสมอ การคำนวณหาบริเวณสัมผัสอิสระจึงถูกพัฒนาขึ้นมาเพื่อหาบริเวณที่ทนต่อความผิดพลาดในการวางนิ้วจับ วิทยานิพนธ์นี้จึงได้เสนอวิธีการคำนวณหาบริเวณสัมผัสอิสระสำหรับการจับวัตถุ 2 และ 3 มิติด้วยนิ้วจับ 2 นิ้ว

ในกรณีการจับวัตถุ 2 มิติ จะใช้การสร้างมิติของท่าจับที่เป็นไปได้ทั้งหมดซึ่งแสดงเป็นระนาบสองมิติโดยที่พิกัด (x,y) ใดๆจะหมายถึงการจับที่วางนิ้วจับ ณ แหน่ง x และอีกนิ้วที่ตำแหน่ง y โดย x และ y เป็นระยะที่วัดโดยอิงกับเส้นรอบรูปของวัตถุ หรือเปรียบเทียบได้กับการคลี่ผิวของวัตถุทั้งชิ้นออกมาเป็นเส้นตรงแล้วทำการอ้างอิงตำแหน่งใดๆโดยการวัดจากจุดเริ่มต้นไปยังจุดนั้นแล้วทำการคำนวณหา L voronoi diagram ซึ่งจะทำให้ได้สี่เหลี่ยมจตุรัสที่ใหญ่ที่สุดบนมิติดังกล่าวและสามารถแปลงกลับมาเป็นบริเวณสัมผัสอิสระบนผิววัตถุได้ วิธีที่นำเสนอนี้เป็นวิธีที่มีความซับซ้อนเชิงอัลกอริทึมสูงเพื่อให้การหาคำตอบทำได้ในเวลาที่รวดเร็ว แต่เนื่องด้วยข้อจำกัดของวิธีคำนวณ L voronoi diagram บนขนาดมิติที่สูงกว่านี้ยังไม่มีผู้เสนอเอาไว้รวมถึงปัญหาของการแปลงผิววัตถุบน 3 มิติให้เป็นแผ่นราบที่มีความต่อเนื่องกันทั้งหมดโดยที่ไม่เกิดการยืดหรือบีบบริเวณใดบริเวณหนึ่งซึ่งก็ยังไม่วิธีที่สามารถนำมาประยุกต์ใช้ร่วมกันได้ วิธีที่นี้จึงไม่สามารถปรับขึ้นไปใช้กับการจับที่มีจำนวนนิ้วหรือมิติแตกต่างออกไปได้

เนื่องจากไม่สามารถปรับวิธีที่ใช้คำนวณหาบริเวณสัมผัสอิสระบน 2 มิติมาใช้กับกรณี 3 มิติได้จึงต้องหาวิธีอื่นในการแก้ปัญหา ตามที่ได้กล่าวไปในบทนำแล้วว่าวิธีที่มีผู้เสนอเอาไว้นั้นไม่

สามารถรับประกันว่าผลลัพธ์ที่ได้จะเป็นคำตอบที่ดีที่สุด ผู้วิจัยจึงได้เสนอวิธีคำนวณโดยแปลงปัญหาให้อยู่ในรูปของกราฟโดย vertex แทน face ของวัตถุ และ edge ประเภทแรกเชื่อมระหว่าง face ที่อยู่ติดกัน ส่วน edge ประเภทที่ 2 คือ edge เชื่อมระหว่าง face ที่มีสมบัติปิดของการจับและบริเวณสัมผัสอิสระที่มีขนาด l หน่วยที่มีจุดศูนย์กลางการจับอยู่ที่จุด $\mathbf{p_1}$ และ p_2 เมื่อแปลงไปอยู่บนระบบกราฟจะเทียบเท่ากับการที่มีเซต 2 เซตโดยที่เซตแรกเป็นเซตของ vertex ที่อยู่ห่าง p_1 ไม่เกิน l หน่วยเซตที่ 2 เป็นเซตของ vertex ที่อยู่ห่าง $\mathbf{p_2}$ ไม่เกิน l หน่วยโดยระยะวัดจากระยะทางการเดินที่สั้นที่สุดที่เป็นไปได้ระหว่าง vertex ผ่าน edge ประเภทแรกและเป็น complete bipartite graph ระหว่าง 2 เซตดังกล่าวโดยดูที่ edge ประเภทที่ 2 ซึ่งปัญหานี้ไม่ยากเท่าปัญหาการหา complete bipartite sub-graph จากระบบ graph จริงๆ จึงทำให้สามารถเปลี่ยนปัญหานี้ไปทำงานเป็นรอบๆโดยขยายเซตของ face ที่อยู่ติดกันและลดเซตของ face ที่มีสมบัติปิดของการจับ โดยในแต่ละรอบจะทำการทดสอบว่ายังมีคู่ face ที่มีสมบัติปิดของการจับอยู่หรือไม่ ถ้าไม่มีก็จะหยุดทำงานและรายงานผล

จากการแปลงปัญหาเป็นลักษณะดังกล่าวทำให้สามารถใช้ GPU ในการคำนวณได้ซึ่งใช้เวลาคำนวณน้อยกว่าโปรแกรมที่มีลักษณะการทำงานเหมือนกันแต่ทำงานบน CPU มากกว่า 10 เท่า

4.2 ข้อเสนอแนะ

ในกรณีของการคำนวณหาบริเวณสัมผัสอิสระบน 3 มิตินั้นวิธีคำนวณในขั้นตอนต่างๆที่เกี่ยวข้องไม่น่าจะสามารถหาวิธีที่ดีกว่าที่ใช้อยู่ในปัจจุบันได้ ณ ขณะนี้ การปรับปรุงที่ทำได้จึงน่าจะเป็นการแยกส่วนการคำนวณบางส่วนที่เป็นอิสระต่อกันออกมาทำงานแบบขนานเพื่อเพิ่มความเร็วของการคำนวณได้

ในกรณีของการคำนวณหาบริเวณสัมผัสอิสระบน 3 มิตินั้นใช้หน่วยความจำจำนวนมาก ทำให้ปริมาณ face สูงสุดที่คำนวณได้อยู่ที่ประมาณ 30,000 face เท่านั้น สาเหตุหลักอย่างหนึ่งเนื่องมาจากวิทยานิพนธ์ฉบับนี้เลือกใช้ texture memory เพื่อให้ได้ความเร็วในการอ่านข้อมูลที่สูงขึ้น แต่หน่วยความจำที่ต้องใช้มากขึ้นเป็น 2 เท่าของปรกติเนื่องจากต้องประกาศ array สำหรับเขียนค่าที่ได้จากการคำนวณแล้วค่อยทำการคัดลอกกลับไปยัง texture memory แต่ถ้าทำการศึกษาเพิ่มเติมอาจจะทำให้สามารถหาวิธีใช้ GPU ที่มีประสิทธิภาพกว่าวิธีปัจจุบัน หรือในอนาคต CUDA เวอร์ชั่นถัดไปอาจจะมีวิธีใหม่ๆในการจัดการกับปัญหานี้

4.3 งานวิจัยในอนาคต

วิธีคำนวณของกรณี 2 มิตินั้นสามารถนำไปประยุกต์ใช้กับการจับวัตถุ 2 มิติด้วยนิ้วจับ 3 นิ้วได้เนื่องจากเราสามารถคลี่เส้นรอบรูปออกมาเป็นเส้นตรงได้ในลักษณะเดียวกันกับกรณี 2 มิติ ข้อแตกต่างอย่างอยู่ 2 จุดใหญ่ๆคือการหาคำตอบจำเป็นจะต้องใช้ L voronoi diagram บน 3 มิติ ซึ่งยังไม่มีผู้เสนอวิธีคำนวณเอาไว้ อีกประเด็นคือ รูปร่างของปริมาณที่อธิบายบริเวณที่มีสมบัติปิดของการจับนั้นจะไม่เป็น polyhedral ในบางส่วนจะมีส่วนโค้งเป็นสมการกำกับรูปซึ่งจะเป็นปัญหากับการสร้าง L voronoi diagram แต่ก็อาจจะใช้วิธีการแบ่งคำนวณเป็น octree เพื่อระบุบริเวณออกเป็น 3 ประเภทคือ มีสมบัติปิดของการจับทั้งชิ้น,มีสมบัติปิดของการจับบางส่วน และเป็นบริเวณที่ไม่มีท่าจับที่เป็นไปได้อยู่ และนำชิ้นที่มีสมบัติปิดของการจับบางส่วนมาแบ่งลงไปจนกว่าจะได้ความละเอียดที่ต้องการและทำการคำนวณหา L voronoi diagram บน octree ซึ่งมีวิธีที่สามารถทำได้จริง

ส่วนการจับวัตถุ 3 มิตินั้น สามารถที่จะพัฒนาวิธีคำนวณในปัจจุบันเพื่อให้รองรับกับ input ที่มีขนาดใหญ่ขึ้นและทำงานได้เร็วขึ้นรวมถึงให้คำตอบที่ดีขึ้นได้หลายวิธี อย่างเช่นการทำ clustering ของคำตอบที่เป็นไปได้ทั้งหมดแล้วนำบริเวณที่มีคำตอบอยู่หนาแน่นมาทำการ remeshing ให้มีความละเอียดสูงขึ้นและนำเฉพาะบริเวณดังกล่าวไปทำการคำนวณใหม่เพื่อให้ได้คำตอบที่ละเอียดขึ้น หรือหาวิธีในการแบ่งงานออกเป็นส่วนย่อยๆเพื่อให้โปรแกรมสามารถแยกคำนวณที่ละส่วนได้ก็จะสามารถรองรับวัตถุที่มีจำนวน face สูงๆได้ หรือเลือกใช้วิธี remeshing ที่เหมาะสมมากกว่าที่ใช้อยู่ในปัจจุบันเพื่อให้ลำดับการเรียงของ face ที่ติดกันมีลำดับใกล้กันเพื่อลด cache missed ของการอ่านข้อมูลจาก texture memory ซึ่งเป็นปัญหาหลักที่ทำให้ใช้เวลาในการคำนวณสูง

รายการอ้างอิง

[1] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. Robotic Grasping of Novel Objects using Vision. International Journal of Robotics Research (IJRR) vol. 27, no. 2, (Feb 2008) : 157-173

[2] William T. Townsend. The BarrettHand grasper–programmably flexible part handling and

assembly. Industrial Robot: An International Journal 27,3:181–188.

[3] S. Jacobsen, E. Iversen, D. Knutti, R. Johnson, and K. Bigger. Design of the Utah/MIT Dextrous Hand. IEEE Int. Conf. on Robotics and Automation (1986) : 96-102.

[4] C.S. Lovchik and M.A. Diftler.The Robonaut hand: a dexterous robot hand for space. IEEE Int. Conf. on Robotics and Automation (1986) : 907-912,1999.

[5] J. Butterfass, M. Grebenstein, H. Liu, and G. Hirzinger. DLR-Hand II: Next Generation of a Dextrous Robot Hand. IEEE Int. Conf. on Robotics and Automation, Seoul, 2001.

[6] D. Kirkpatrick, B. Mishra, and C. Yap. Quantitative Steinitz's theorems with applications to multifingered grasping. In 20th ACM Symp. on Theory of Computing. Baltimore, MD (1990), pp.

341–351.

[7] C. Ferrari and J. Canny. Planning optimal grasps. In IEEE Int. Conf. on Robotics and Automation. Nice, France (1992), pp.2290–2295.

[8] B. Mirtich and J. Canny. Optimum force-closure grasps. Technical Report ESRC 93-11/RAMP 93-5, Robotics, Automation, and Manufacturing Program, University of California at

Berkeley (1993).

[9] C. Borst, M. Fischer, and G. Hirzinger. Grasping the dice by dicing the grasp. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2003).

[10] Y.-B. Jia. On computing optimal planar grasps. In IEEE Int. Conf. on Robotics and Automation (1995).

[11] X. Zhu and J. Wang. Synthesis of force-closure grasps on 3-d objects based on the q distance. IEEE Trans. Robot. Autom., 19(4), 669–679 (2003).

[12] R. Brost. Automatic grasp planning in the presence of uncertainty. Int. J. Robot. Res., 7(1), 3–17 (1988).

[13] Y. Zheng and W.-H. Qian. Coping with the grasping uncertainties in force-closure analysis. Int. J. Robot. Res., 24(4), 311–327 (2005).

[14] V. Christopoulos and P. Schrater. Handling shape and contact location uncertainty in grasping two-dimensional planar objects. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2007), pp. 1557–1563.

[15] A. Dollar and R. Howe. Simple, robust autonomous grasping in unstructured environments. In IEEE Int. Conf. on Robotics and Automation (2007), pp. 4693–4700.

[16] J. Felip and A. Morales. Robust sensor-based grasp primitive for a three-finger robot hand. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2009), pp. 1811–1816.

[17] D.-J. Kim, R. Lovelett, and A. Behal. Eye-in-hand stereo visual servoing of an assistive robot arm in unstructured environments. In IEEE Int. Conf. on Robotics and Automation (2009), pp.
2326–2331.

[18] V.-D. Nguyen. Constructing force-closure grasps. Int. J. Robot. Res., 7(3), 3–16 (1988).

[19] C. P. Tung and A. C. Kak. Fast construction of force-closure grasps. IEEE Trans. Robot. Autom., 12(4), 615–626 (1996).

[20] J. Ponce and B. Faverjon. On computing three-finger forceclosure grasps of polygonal objects. IEEE Trans. Robot. Autom., 11(6), 868–881 (1995).

[21] P. Sanz, A. Requena, J. I˜nesta, and A. Del Pobil. Grasping the not-so-obvious: vision-based object handling for industrial applications. IEEE Robot. Autom. Mag., 12(3), 44 – 52 (2005).

[22] J. Cornell´a and R. Su´arez. A new framework for planning threefinger grasps of 2d irregular objects. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2006).

[23] M. Roa and R. Su´arez. Computation of independent contact regions for grasping 3-d objects. IEEE Trans. Robot., 25(4), 839 –850 (2009).

[24] I.-M. Chen and J. Burdick. Finding antipodal point grasps on irregularly shaped objects. IEEE Trans. Robot. Autom., 9(4), 507 –512 (1993).

[25] A. Blake. A symmetry theory of planar grasp. Int. J. Robot. Res., 14(5), 425–444 (1995).

[26] Y.-B. Jia. Computation on parametric curves with an application in grasping. Int. J. Robot. Res., 13(7 -8), 825 –855 (2004).

[27] J. Ponce, D. Stam, and B. Faverjon. On computing two-finger force-closure grasps of curved 2D objects. Int. J. Robot. Res., 12(3), 263–273 (1993).

[28] F. P. Bowden and D. Tabor. Friction and lubrication. Oxford: Oxford University Press,1954.

[29] J. L. Meriam, L. G. Kraige, and William J., III Palm. Engineering Mechanics, Statics. New York: Wiley Text Books, 2001.

[30] J.-S. Cheong, H. J. Haverkort, and A. F. van der Stappen. Computing all immobilizing grasps of a simple polygon with few contacts. Algorithmica, 44, 117–136 (2006).

[31] E. Papadopoulou and D. T. Lee. The l1-voronoi diagram of segments and vlsi applications. Int. J. Comput. Geometry Appl., 11(5), 503–528 (2001).

[32] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Computational Geometry: Algorithms and Applications (Springer, 1997).

[33] T. K. Dey. Curve and Surface Reconstruction (Cambridge University Press, New York, 2007).

[34] J. Speth, A. Morales, and P. J. Sanz. Vision-based grasp planning of 3D objects by extending 2D contour based algorithms. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2008).

[35] C. Goldfeder, M. Ciocarlie, H. Dang, and P. Allen. The columbia grasp database. In IEEE Int. Conf. on Robotics and Automation (2009), pp. 1710 –1716.

ประวัติผู้เขียนวิทยานิพนธ์

นาย ชัยชนะ นิลวัชรารัง เกิดเมื่อวันที่ 5 มกราคม พ.ศ.2525 ที่กรุงเทพมหานคร สำเร็จการศึกษาหลักสูตรวิศวกรรมศาสตร์บัณฑิต (วศ.บ.) สาขาวิชาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2549 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตร์มหาบัณฑิต (วศ.ม.) สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2550