การสร้างภาพสามมิติกลับคืนจากภาพสองมิติของรูปทรงลูกบาศก์และรูปทรงพีระมิด

THREE-DIMENSIONAL IMAGE RECONSTRUCTION FROM TWO-DIMENSIONAL IMAGES OF CUBE AND PYRAMID SHAPES

นาย รุจน์ ปันแก้ว รหัส 53363850

นาย ภัทรพงศ์ ใจกลม รหัส 53363751

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2556

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน การสร้างภาพสามมิติกลับคืนจากภาพสองมิติของรูปทรงลูกบาศก์และรูปทรงพีระมิด

ผู้ดำเนินโครงงาน นายภัทรพงศ์ ใจกลม รหัส 53363751

นายรุจน์ ปันแก้ว รหัส 53363850

ที่ปรึกษาโครงงาน ดร.สุวิทย์ กิระวิทยา

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2556

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

Suvit Kiravittaya .....ที่ปรึกษาโครงงาน

(ดร.สุวิทย์ กิระวิทยา)

.....กรรมการ

(อาจารย์รัฐภูมิ วรานุสาสน์)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ดร. พนมขวัญ ริยะมงคล)

หัวข้อโครงงาน	การสร้างภาพสามมิติกลับคืนจากภาพสองมิติของรูปทรงลูกบาศก์และรูปทรงพีระมิด			
ผู้ดำเนินโครงงาน	นายภัทรพงศ์	ใจกลม	รหัส	53363751
	นายรุจน์	ปันแก้ว	รหัส	53363850
อาจารย์ที่ปรึกษา	ดร.สุวิทย์ กิระวิทยา			
สาขาวิชา	วิศวกรรมคอมพิวเตอร์			
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์			
ปีการศึกษา	2556			

บทคัดย่อ

โครงงานการสร้างภาพสามมิติกลับคืนจากภาพสองมิติของรูปทรงลูกบาศก์และรูปทรงพีระมิด เป็นโครงงานที่ได้นำหลักการ Free viewpoint TV มาประยุกต์กับการประมวลผลภาพเพื่อสร้างรูปสามมิติจากรูปสองมิติในแต่ละมุม โดยหาความยาวของด้านและการเคลื่อนที่ของมุมมาวิเคราะห์โดยใช้วิธีการของภาพฉาย (Projection) มาประยุกต์ใช้กับการหมุน (Rotation) เพื่อหาความยาวจริงและมุมเริ่มต้นของภาพสองมิติ

คณะผู้จัดทำโครงงานได้จัดทำโปรแกรมที่ช่วยในการสร้างภาพสามมิติกลับคืนจากภาพสองมิติ ด้วยการใช้ภาพสองมิติจากโปรแกรมแมทแลบ (MATLAB) เวอร์ชั่น 2013a แล้วนำภาพสองมิติที่ได้มาสร้างเป็นภาพสามมิติกลับคืน ซึ่งผู้ใช้โปรแกรมสามารถเลือกประเภทของรูปนำเข้าระหว่างรูปทรงลูกบาศก์และรูปทรงพิระมิดได้อย่างละ 3 รูป และหลังจากการประมวลผลจนได้รูปสามมิติแล้วผู้ใช้ยังสามารถรู้ความยาวแต่ละแกนของรูปทรงนั้น และรู้แม้กระทั่งมุมเริ่มต้นของรูปสามมิตินั้น นอกจากนี้ผู้ใช้ยังสามารถทดลองหมุนรูปได้อีกด้วย ซึ่งมันจะช่วยให้ผู้ใช้สามารถเห็นรูปได้ทุกมุม

Project Title	Three – Dimensional Image Reconstruction From Two – Dimensional Image of Cube and Pyramid shape			
Name	Mr. Pattarapong	Jaiklom	ID.	53363751
	Mr. Ruj	Pankaew	ID.	53363850
Project Advisor	Dr. Suwit Kiravittaya, Ph.D.			
Major	Computer Engineering.			
Department	Electrical and Computer Engineering.			
Academic year	2013			

Abstract

The project to create a three - dimensional reconstruction from two - dimensional image of the cube and pyramid shapes. That has adopted the Free viewpoint TV application for image processing, to create a three-dimensional, two -dimensional in each corner. By the length of the sides and angles of motion were analyzed by means of image projection applied to the rotation to determine the actual length and the start of a two-dimensional image.

The project team has prepared a program that helps to create three-dimensional images from two-dimensional image restored. Using two-dimensional image of the MATLAB program version 2013a, two - dimensional image is then used to create a three -dimensional back. The user can select any of the program is to import the cube and pyramid shapes with 3 photos and after processing until it is three - dimensional, then the user can also know the length of each axis of the shape. And know even the beginning of the third dimension. In addition, users can also try to rotate the image. It enables users to see at all.

กิตติกรรมประกาศ

โครงงานการสร้างภาพสามมิติกลับคืนจากภาพสองมิติของรูปทรงลูกบาศก์และรูปทรงพีระมิดฉบับนี้ สำเร็จลุล่วงได้ด้วยดีเนื่องจากความอนุเคราะห์ของอาจารย์ที่ปรึกษาโครงงาน คือ ดร.สุวิทย์ กิระวิทยา ที่ให้ความช่วยเหลือในด้านต่างๆ ช่วยชี้แนะแนวทางในการทำงาน พร้อมทั้งให้คำปรึกษา ตรวจสอบและแก้ไขข้อผิดพลาดต่างๆ คณะผู้จัดทำรู้สึกเป็นเกียรติอย่างมากที่ได้รับความอนุเคราะห์จากอาจารย์

ในโอกาสนี้ทางคณะผู้จัดทำโครงงานจึงขอขอบพระคุณอาจารย์ทุกท่าน บิดา มารดา ที่มีส่วนช่วยให้โครงงานนี้ประสบความสำเร็จได้ด้วยดี ทั้งนี้ขอขอบคุณเพื่อนๆ ที่คอยให้กำลังใจในการทำงานจนสำเร็จลุล่วงมาด้วยดี

คณะผู้จัดทำโครงงาน
นายภัทรพงศ์ ใจกลม
นายรุจน์ ปันแก้ว

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันภาพสามมิติ นิยมนำมาใช้กันอย่างแพร่หลาย ไม่ว่าจะเป็นด้านงานภาพยนตร์ที่ถูกพัฒนาภาพยนตร์ในระบบ 2 มิติ งานพัฒนาโครงสร้างทางวิศวกรรม และ สถาปัตยกรรม เพื่อช่วยในการจำลองโครงสร้างและมุมมอง ในรูปแบบต่างๆ หรือแม้กระทั่งทางการแพทย์ ที่ช่วยวิเคราะห์ระบบโครงสร้างอวัยวะ ต่างๆในร่างกายมนุษย์ เหล่านี้คือประโยชน์ที่ได้มาจากภาพสามมิติ

นอกเหนือจากที่กล่าวมาขั้นต้นแล้ว ยังมีเทคโนโลยีอีกมากในปัจจุบันที่ทำเกี่ยวกับ ภาพสามมิติ และการมองภาพ หนึ่งในนั้นคือ เทคโนโลยีเปลี่ยนมุมมองตามตามนุษย์ (Free Viewpoint TV) เทคโนโลยีตัวนี้เป็นเทคโนโลยีตัวหนึ่งที่ทำงานเกี่ยวกับการแปลงภาพสามมิติ โดยภาพสามารถเปลี่ยนมุมมองไปตามมุมมองของดวงตามนุษย์ ซึ่งการทำงานของเทคโนโลยีดังกล่าวนี้เอง เราจึงมีแนวคิดในการทำโครงงาน “การสร้างภาพสามมิติกลับคืนจากภาพสองมิติของรูปทรงลูกบาศก์และรูปทรงพีระมิด” เพื่อจะช่วยในการมองภาพและรูปทรงต่างๆ จากภาพสองมิติที่มองเห็นเพียงด้านใดด้านหนึ่งก็จะช่วยให้มองได้ทุกด้านซึ่งอาศัยหลักการทำงานของ เทคโนโลยีนี้นั่นเอง

การแปลงภาพ จากระบบสองมิติ เป็นสามมิตินั้น เราจำเป็นต้องทราบถึงความกว้าง ความสูง และความลึก ซึ่งหากเรามองด้วยตาเปล่าเราไม่สามารถรู้ความลึกของภาพได้ แต่เมื่อเรามองด้วยตาเปล่าสามารถรู้ว่า จุดไหนของภาพ อยู่ใกล้หรือไกลกว่ากัน และเราสามารถใช้จุดเหล่านั้น หาจุดรวมของภาพ เมื่อหาจุดรวมของภาพได้แล้ว เราจะพัฒนาเป็นภาพสามมิติได้

ในขั้นต้นนี้ เราจำเป็นต้องใช้ภาพที่มีจุดใกล้ จุดไกลชัดเจน โดยจะใช้รูปเรขาคณิต ทั้งนี้เพื่อจุดรวมภาพที่ชัดเจน ซึ่งรูปเรขาคณิตเหล่านี้ เป็นรูปพื้นฐานในการพัฒนางานแอนิเมชั่น โปรแกรมนี้จะสามารถบอกถึงรูป สองมิติที่ถูกอิมพอร์ต เข้ามาว่าเป็นรูปไหน แล้วจึงจะทำการแปลงในรูป สามมิติ ในลักษณะที่ไม่ใช่สามมิติแบบเต็มรูปแบบ (Full 3D) ซึ่งจะไม่มีแสงเข้ามาเกี่ยวข้องแต่สามารถเห็นโครงร่างรูปเรขาคณิตที่ถูกแปลงในรูปสามิติออกมาได้ค่อนข้างชัดเจน

ดังนั้น ผู้จัดทำจึงได้จัดทำโครงงาน "การสร้างภาพสามมิติกลับคืนจากภาพสองมิติของรูปทรงลูกบาศก์และรูปทรงพีระมิด" ขึ้นมาเพื่อเพิ่มทางเลือกให้กับผู้ใช้ ส่งเสริมพัฒนางานด้านศิลปะแอนิเมชั่นและตอบสนองในด้านความสะดวกสบาย และง่ายต่อการใช้งาน

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อสร้างโปรแกรม ที่ตอบสนองกับภาพในลักษณะที่แตกต่างกัน

1.2.2 เพื่อพัฒนางานด้านกราฟฟิก

1.2.3 เพื่อพัฒนาซอฟท์แวร์ในการช่วยเหลือมนุษย์ในด้านการทำงานร่วมกับแอนิเมชั่น

1.2.4 เพื่อศึกษาโครงสร้างภาพในระดับมุมมองของทัศนมิติ

1.3 ขอบเขตของโครงงาน

โครงงานนี้ศึกษาโดยใช้การทำงานของเทคโนโลยีเปลี่ยนมุมมองตามตามนุษย์ (Free Viewpoint TV) โดยใช้รูปทรงเรขาคณิตทึบแสงที่สมบูรณ์ที่สุด คือ ไม่มีรอยต่อระหว่างด้าน มีพื้นหลังสีขาว สร้างจากซอฟท์แวร์ใดๆ และเป็นเสมือนวัตถุที่วางอยู่บนโต๊ะ ซึ่งมองได้ในระดับสายตา ซึ่งไม่มีเรื่องสีและแสงภายนอกเข้ามาเกี่ยวข้อง โดยผู้ใช้จำเป็นต้องรู้ค่าของมุมเซต้า (theta) และค่ามุมแกมม่า (gamma) ซึ่งในการทดสอบจะใช้ชุดทดสอบ 2 ชุด ที่มีค่ามุมเซตา (theta) และ มุมแกมม่า (gamma) เท่ากับ 10 ดังนี้

ชุดที่ 1 ทรงลูกบาศก์ซึ่งจะประกอบไปด้วย 3 รูป ดังนี้

รูปที่ 1 เป็นรูปทดสอบต้นแบบ ที่มีมุมเริ่มต้นที่ มุมแอลฟ่า (alpha) = 40 องศา และ มุมเบต้า (beta) = 25 องศา

รูปที่ 2 เป็นรูปที่ค่ามุมแอลฟ่า + มุมเซตา และค่ามุมเบต้าคงที่

รูปที่ 3 เป็นรูปที่มีค่ามุมเบต้า + มุมแกมม่า และค่ามุมแอลฟ่าคงที่

ชุดที่ 2 ทรงพีระมิดจะประกอบไปด้วย 3 รูป ดังนี้

รูปที่ 1 เป็นรูปทดสอบต้นแบบที่มีมุมเริ่มต้นที่ มุมแอลฟ่า = 40 องศา และ มุมเบต้า = 20 องศา

รูปที่ 2 เป็นรูปที่ค่ามุมแอลฟ่า + มุมเซตา และค่ามุมเบต้า คงที่

รูปที่ 3 เป็นรูปที่มีค่ามุมเบต้า + มุมแกมม่า และค่ามุมแอลฟ่า คงที่

1.4 แผนการดำเนินงาน

ตารางที่ 1.1 ตารางขั้นตอนและแผนการดำเนินโครงงาน

กิจกรรม	ปี 2556									
	เม.ย	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค. 57
1.ศึกษาค้นคว้าข้อมูลในการทำโครงงาน	↔									
2.ศึกษาโครงสร้างภาษาแมทแลบ(MATLAB)	←	→								
3.ศึกษาโครงร่างและวิเคราะห์จุดจุดร่วมจากการสร้าง รูปเรขาคณิต แบบ 3 มิติด้วยโปรแกรมแมทแลบ (MATLAB)		←	→							
4. เขียนโปรแกรมแปลงภาพจากจุดร่วมที่ได้มา และทดลองกับภาพต้นแบบ			←	→						
5. เพิ่มจำนวนรูปเรขาคณิตแบบต่างๆ และ เขียนโปรแกรมแปลงภาพ							←	→		
6. ทำการออกแบบโปรแกรมในส่วนของ GUI และตรวจสอบพัฒนาในส่วน CODE								←	→	
7.ทดสอบการใช้งานและแก้ไขข้อบกพร่อง									↔	
8.สรุปผลการทำโครงงานและจัดทำรายงาน										↔

1.5 ผลที่คาดว่าจะได้รับ

1.5.1 โปรแกรมแปลงรูปเรขาคณิตสองมิติเป็นสามมิติได้ตามชุดทดสอบ

1.5.2 ทำให้ช่วยวิเคราะห์พื้นผิวของรูปได้ดี

1.5.3 เป็นพื้นฐานในการพัฒนางานแอนิเมชั่นได้ค่อนข้างดี

1.6 งบประมาณของโครงงาน

1.6.1 ค่าอุปกรณ์ในการดำเนินโครงงาน	250	บาท
1.6.2 ค่าเอกสารที่ใช้ในการดำเนินโครงงาน	500	บาท
1.6.3 ค่าหมึกพิมพ์	1000	บาท
1.6.4 ค่าเข้าเล่มโครงงาน	250	บาท
รวมเป็นเงินทั้งสิ้น	2000	บาท

หมายเหตุ ขออนุมัติถัวเฉลี่ยทุกรายการ

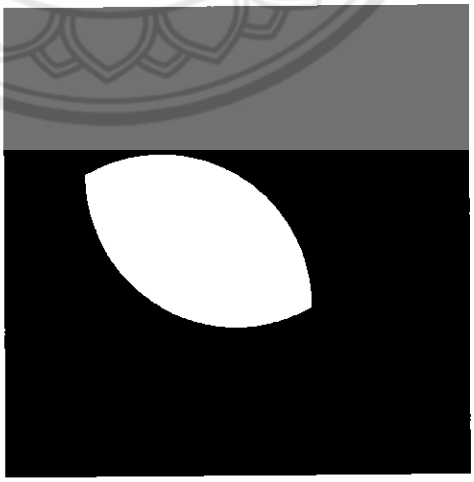
บทที่ 2

หลักการและทฤษฎี

สำหรับในบทนี้ได้อธิบายถึงในส่วนของการจัดการกับรูปภาพและการจัดการกับรูปร่าง ทั้งที่เป็นรูปทรงเรขาคณิตและรูปร่างเรขาคณิต โดยมีการอ่านภาพและกำหนดขอบเขตภาพให้อยู่ในรูปแบบตามลักษณะที่ต้องการ ซึ่งจะเก็บอยู่ในรูปของหน่วยความจำและแสดงผลผ่านโปรแกรม

2.1 ระบบสี RGB

ระบบสี RGB คือ ระบบสีที่เกิดจากการรวมตัวของแสงสีแดง เขียวและน้ำเงิน โดยจะรวมกันแบบ Additive ปกติจะใช้ในจอภาพแบบ CRT (Cathode ray tube) RGB ที่มีการนิยมใช้อย่างมากคือ RGB_{CIE} และ RGB_{NTSC} โดยในสีแต่ละสีจะถูกควบคุมด้วยจำนวนบิต 8 บิต โดยมีค่าต่ำสุดของสี คือ 0 (00 ในฐานสิบหก) และค่าสูงสุด คือ 255



รูปที่ 2.1 ระบบสีของ RGB

ที่มา : bpiinc.files.wordpress.com/2011/11/rgb.png

2.2 ภาพไบนารี่ (Binary Image)

ภาพไบนารี่ (Binary image) ในทางดิจิตอลหมายถึง ภาพหนึ่งจะมีอยู่ 2 สถานะ คือ 0 และ 1

- โดย Pixel ใด เป็น 0 จะแสดงพื้นที่นั้น เป็นสีดำ
- โดย Pixel ใด เป็น 1 จะแสดงพื้นที่นั้น เป็นขาว

การที่ภาพที่มีทั้งสีและระดับชั้นของสี ทำให้เกิดลวดลายขึ้น ซึ่งระดับความเข้มที่น้อยที่สุดเรียกกันว่าความเข้มสองระดับ (Binary) คือ ภาพขาวกับดำซึ่งเก็บรายละเอียดได้หมด คือ ภาพ 256 ระดับ หากเราต้องการจำแนกลวดลายเราจึงจำเป็นต้องทำการแปลงรูปภาพให้อยู่ในระดับของภาพไบนารี่เสียก่อน ซึ่งในแมทแลบมีคำสั่งหนึ่งในเครื่องมือ (Toolbox) ที่ชื่อว่า การประมวลผลภาพ (image processing toolbox)

รูปที่ 2.2 แสดงการเปลี่ยนแปลงของภาพปกติและภาพไบนารี่

ที่มา : www.mathworks.com/help/images/ref/bwconvhull.png

2.3 การจัดการภาพตามรูปแบบต้องการ (Manipulate images)

การจัดการรูปภาพโดยใช้เครื่องมือ (Toolbox) ในโปรแกรมแมทแลบ (MATLAB) จำเป็นต้องทราบเป้าหมายในการจัดการกับภาพก่อนเสมอเพื่อใช้คำสั่งให้ตรงกับเป้าหมายที่วางไว้ ส่วนใหญ่แล้วการจัดการกับรูปภาพ ในส่วนแรกจำเป็นที่ต้องให้รูปอยู่ในระบบภาพที่มีสีที่น้อยที่สุดก่อน เช่น สีขาวกับ สีดำ เป็นต้น เพื่อการมองเห็นโครงร่างของภาพที่ค่อนข้างชัดเจน ส่วนที่สองเป็นการกำจัดในส่วนที่ไม่ต้องการออก (โดยส่วนใหญ่ในขั้นตอนแรกก็สามารถจำกัดภาพได้ในระดับหนึ่งแล้ว) แล้วจึงเหลือส่วนที่เราต้องการไว้จัดการต่อไป ซึ่งขั้นตอนที่กล่าวมามีรายละเอียดดังนี้

2.3.1 การจัดการภาพโดยใช้เมทริกซ์ (Matrix images)

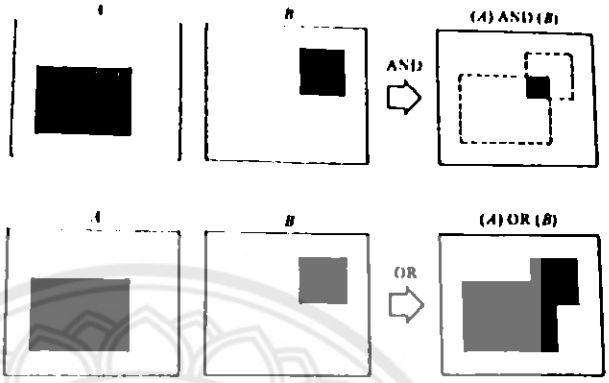
เมทริกซ์ คือ กลุ่มตัวเลขที่เขียนไว้ในวงเล็บหรือวงเล็บสี่เหลี่ยม ในทางโปรแกรมเสมือนกับการเก็บข้อมูลในลักษณะของอาร์เรย์สองมิติ กลุ่มตัวเลขดังกล่าวแต่ละตัวเรียกเป็นสมาชิกของเมทริกซ์ซึ่งกำหนดตามตำแหน่งในลักษณะของแถว (row) และหลัก (column) เมทริกซ์เป็นอาร์เรย์ขนาด 2 มิติของข้อมูลตัวเลข ซึ่งในแต่ละแถวหรือหลักประกอบด้วยตัวเลขหนึ่งหรือมากกว่าหนึ่งค่า การดำเนินการทางคณิตศาสตร์ซึ่งสามารถทำได้กับเมทริกซ์ประกอบด้วย การบวก การลบ การคูณและการหาร ขนาดของเมทริกซ์ถูกกำหนดในลักษณะของจำนวนของแถวและจำนวนหลัก โดยส่วนใหญ่แล้วเมทริกซ์ในรูปแบบของภาพไบนารี่ (Binary image) จะมีค่าอยู่สองค่า คือ 0 และ 1 ในอาเรย์สองมิติหากต้องการกลับภาพจากภาพสีดำเป็นสีขาว หรือ สีขาวเป็นสีดำ มันก็คือการกลับค่าในเมทริกซ์จาก 0 เป็น 1 ซึ่งจะใช้ความรู้เรื่อง การคำนวณและตรรกศาสาตร์ (Arithmetic/Logic Operations) เข้ามาเกี่ยวข้อง ซึ่งตัวดำเนินการทางตรรกะ (Operations) ที่สำคัญต่อการจัดการภาพที่เป็นภาพของภาพไบนารี่ (Binary image) คือ ยูเนียน (Union) , อินเตอร์เซคชัน (Intersection) และ คอมพลีเมนต์ (Complement)

ตัวอย่างการใช้งาน

Union : p AND q (also, $p \cdot q$)

Intersection : p OR q (also, $p + q$)

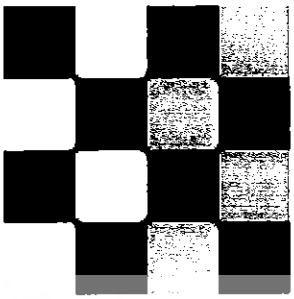
Complement : NOT q (also, ~q)



รูปที่ 2.3 ตัวอย่างของการดำเนินการของตรรกะบนภาพไบนารี่

ที่มา : หนังสือ digital image processing ในเรื่องของ Arithmetic/Logic Operation หน้า 49

2.4 การหามุมโดยใช้เทคนิค The Harris Corner Detector



รูปที่ 2.4 แสดงตัวอย่างผลลัพธ์การหามุมโดยเทคนิค Harris Corner Detector

The Harris Corner Detector เป็นเทคนิคการหาจุดที่กำลังได้รับความสนใจเป็นอย่างมาก เนื่องจากมีจุดเด่นในเรื่องของความแม่นยำของการหมุนสเกลความแปรปรวนทำให้ภาพมีจุดรบกวนซึ่งในแมทแลบ จะใช้เทคนิคนี้ในการหามุม ซึ่งหลักการของ Harris Corner Detector จะอาศัยหลักการของ Sum of Square Different (SSD) ซึ่งแทนด้วยฟังก์ชั่น S ระหว่างบริเวณที่หนึ่งในภาพ (u,v) กับอีกบริเวณที่เลื่อนไปในภาพ เป็นค่าเท่ากับ (x,y) แล้วให้น้ำหนักตามการเลื่อนฟังก์ชัน w คำนวณ

$$S(x,y) = \sum_u \sum_v w(u,v)\,(I(u+x,v+y) - I(u,v))^2 \quad \text{------ (1)}$$

โดยที่ $I(u+x,v+y)$ จะสามารถประมาณโดยการขยายตัวของเทย์เลอร์ให้ I_x และ I_y เป็นพหุนามของ I ตัวอย่างเช่น

$$I(u+x,v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y \quad \text{------ (2)}$$

ซึ่งจะประมาณค่าและเขียนให้อยู่ในรูปเมทริกซ์ดังนี้

$$S(x,y) \approx (x,y)A\begin{pmatrix} x \\ y \end{pmatrix} \quad \text{------ (3)}$$

A คือค่า เมทริกซ์ 2x2 ที่ได้จากคำนวณภาพที่รับมา

$$A = \sum_u \sum_v w(u,v)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_xI_y \rangle \\ \langle I_xI_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \quad \text{------ (4)}$$

โดยมีค่า λ_1, λ_2 เป็นค่าเจาะจง (Eigenvalues) ของ A

หาค่าไอเกนของ A ซึ่งจะพบว่าถ้าค่าไอเกนทั้งสองค่ามีขนาดใหญ่ ภาพดังกล่าวจะเป็นหัวมุม การคำนวณไอเกนจะใช้เวลาเยอะในการถอดรากจึงจำเป็นต้องใช้สมการที่ (5) แทน

$$M_c = \det M - \kappa(traceM)^2 \quad \text{------ (5)}$$

โดยที่ $\det M$ หาได้จาก $\det M = \lambda_1 \cdot \lambda_2$ ส่วนค่า $traceM$ หาจาก $traceM = \lambda_1 + \lambda_2$ และค่า κ (kappa) เป็นค่าคงที่อยู่ระหว่าง 0.04 – 0.06

2.5 ระยะทางระหว่างจุด (Distance Between a Point)

บนเส้นจำนวน ถ้าจุด P_1 แทนจำนวนจริง x_1 และจุด P_2 แทนจำนวนจริง x_2 ระยะทางระหว่างจุด P_1 และ P_2 คือค่าสัมบูรณ์ของ $X_1 - X_2$ เขียนแทน ด้วย P_1P_2 หรือ $|\overline{P_1P_2}|$ $P_1P_2 = |x_1 - x_2|$

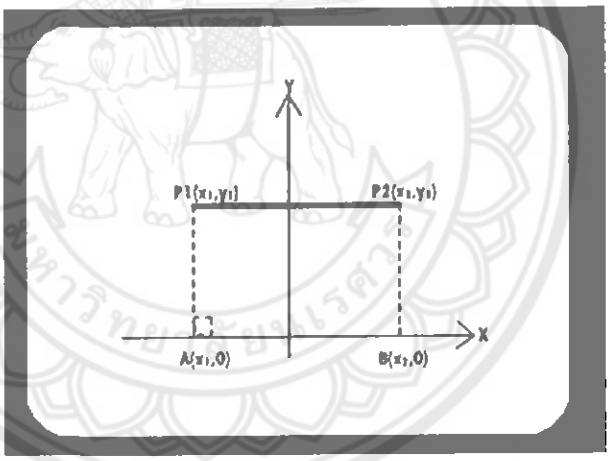
สำหรับจุด 2 จุด $P_1(x_1, x_2)$ และ $P_2(x_1, x_2)$ ใดๆ ที่ $\overline{P_1P_2}$ ขนานกับแกน x หรือขนานกับแกน y สามารถแสดงได้ว่า

1) ถ้า $\overline{P_1P_2}$ ขนานกับแกนx จะได $P_1P_2 = |x_1 - x_2|$

2) ถ้า $\overline{P_1P_2}$ ขนานกับแกน y จะได้ $P_1P_2 = |y_1 - y_2|$

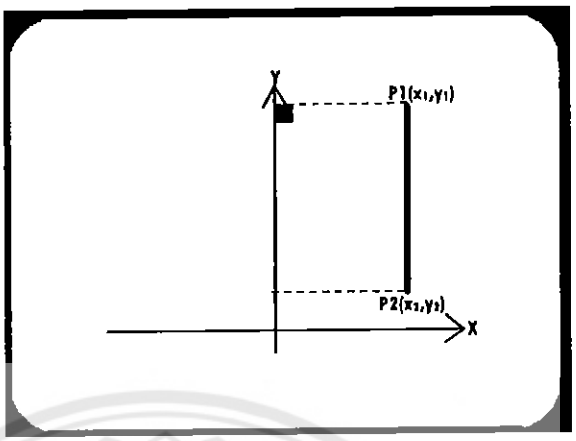
ซึ่งแสดงให้จริงได้ดังนี้

ถ้าจุด $P_1(x_1, x_2)$ และจุด $P_2(x_1, x_2)$ อยู่บนเส้นตรงซึ่งขนานกับแกน x (ดังรูป)



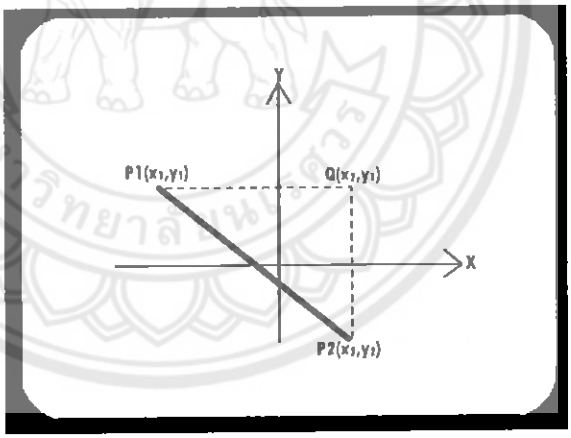
รูปที่ 2.5 แสดงจุด $P_1(x_1, x_2)$ และจุด $P_2(x_1, x_2)$ อยู่บนเส้นตรงซึ่งขนานกับแกน x

ให้ A และ B เป็นจุด $\overline{P_1A}$ ที่ และ $\overline{P_2B}$ ตั้งฉากกับแกน X ตามลำดับ จะได้จุด A มีพิกัดเป็น $(x_1, 0)$ จุด B มีพิกัดเป็น $(x_2, 0)$ ดังนั้น AB $= |x_1 - x_2|$ เส้น $\overline{P_1P_2}$ และ $\overline{AB}$ เป็นด้านตรงข้ามของสี่เหลี่ยมผืนผ้า P_1P_2 BA ดังนั้น P_1P_2 = AB นั่นคือ $P_1P_2 = |x_1 - x_2|$ ในทำนองเดียวกัน ถ้าจุด $P_1(x_1, x_2)$ และจุด $P_2(x_1, x_2)$ อยู่บนเส้นตรง ซึ่งขนานกับแกน Y (ดังรูป)



รูปที่ 2.6 แสดงจุด $P_1\ (x_1\ ,x_2)$ และจุด $P_2\ (x_1\ ,x_2)$ อยู่บนเส้นตรงซึ่งขนานกับแกน Y

จะได้ $P_1P_2\ = |y_1 - y_2|$ ในกรณีที่จุด $P_1\ (x_1\ ,x_2)$ และ $P_2\ (x_1\ ,x_2)$ อยู่บนเส้นตรงซึ่งไม่ขนานกับแกน X และไม่ขนานกับแกน Y เราจะหา P_1P_2 ได้ดังนี้



รูปที่ 2.7 แสดง การหา P_1P_2 ในกรณีจุด $P_1\ (x_1\ ,x_2)$ และ $P_2\ (x_1\ ,x_2)$ อยู่บนเส้นตรงซึ่งไม่ขนานกับแกน X และไม่ขนานกับแกน Y

ลากส่วนเส้นตรง $\overline{P_1Q}$ และ $\overline{P_2Q}$ ให้ขนานกับแกน X และแกน Y ตามลำดับ จุด Q จะมีพิกัดเป็น (x_1, x_2) และมุม P_1QP_2 เป็นมุมฉาก จากทฤษฎีบทปีทาโกรัส จะได้ว่า

$$P_1P_2 = \sqrt{P_1Q^2 + P_2Q^2}$$

$$= \sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2}$$

$$= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$ เนื่องจาก $= a^2$

ข้อสังเกต ในกรณีที่จุด P_1 และ P_2 อยู่บนเส้นตรงที่ขนานกับแกน X จะใช้สูตรนี้ก็ได้ เพราะ $Y_1 = Y_2$ จะได้ $P_1P_2 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} = |x_1 - x_2|$ ทำนองเดียวกัน ในกรณีที่จุด P_1 และ P_2 อยู่บนเส้นตรงที่ขนาน กับแกน Y จะใช้สูตรนี้ก็ได้เพราะ $x_1 = x_2$ จะได้ $P_1P_2 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} = |y_1 - y_2|$ ดังนั้นจึงสรุปเป็น ทฤษฎีบทได้ดังนี้

<u>ทฤษฎีบท</u>

ถ้า $P_1(x_1, x_2)$ และ $P_2(x_2, y_2)$ เป็นจุดในระนาบ ระยะระหว่างจุด P_1 และ P_2 เท่ากับ $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

2.6 ฟังก์ชันตรีโกณมิติ

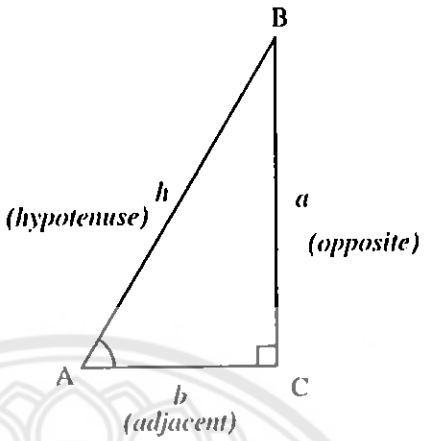
ฟังก์ชันตรีโกณมิติ คือ ฟังก์ชันของมุม ซึ่งมีความสำคัญในการศึกษารูปสามเหลี่ยมและปรากฏการณ์ในลักษณะเป็นคาบ ฟังก์ชันอาจนิยามด้วยอัตราส่วนของด้าน 2 ด้านของรูปสามเหลี่ยมมุมฉากหรืออัตราส่วนของพิกัดของจุดบนวงกลมหนึ่งหน่วย หรือนิยามในรูปทั่วไปเช่น อนุกรมอนันต์ หรือสมการเชิงอนุพันธ์

รูปสามเหลี่ยมที่นำมาใช้จะอยู่ในระนาบแบบยุคลิด ดังนั้น ผลรวมของมุมทุกมุมจึงเท่ากับ 180° เสมอ ในปัจจุบันมีฟังก์ชันตรีโกณมิติอยู่ 6 ฟังก์ชันที่นิยมใช้กันดังตารางข้างล่าง

ฟังก์ชัน	ตัวย่อ	ความสัมพันธ์
ไซน์ (Sine)	sin	$\sin\theta = \cos(\frac{\pi}{2} - \theta)$
โคไซน์ (Cosine)	cos	$\cos\theta = \sin(\frac{\pi}{2} - \theta)$
แทนเจนต์ (Tangent)	tan	$\tan\theta = \frac{1}{cot\theta} = \frac{sin\theta}{cos\theta} = \cot(\frac{\pi}{2} - \theta)$
โคแทนเจนต์ (Cotangent)	cot	$\cot\theta = \frac{1}{tan\theta} = \frac{cos\theta}{sin\theta} = \tan(\frac{\pi}{2} - \theta)$
ซีแคนต์ (Secant)	sec	$\sec\theta = \frac{1}{cos\theta} = \csc(\frac{\pi}{2} - \theta)$
โคซีแคนต์ (Cosecant)	csc	$\csc\theta = \frac{1}{sin\theta} = \sec(\frac{\pi}{2} - \theta)$

ตารางที่ 2.1 ตารางแสดงฟังก์ชันตรีโกณมิติ

2.6.1 นิยามจากรูปสามเหลี่ยมมุมฉาก



รูปที่ 2.8 สามเหลี่ยมมุมฉาก

ในการนิยามฟังก์ชันตรีโกณมิติสำหรับมุม A จะกำหนดให้มุมใดมุมหนึ่งในรูปสามเหลี่ยมมุมฉากเป็นมุม A เรียกชื่อด้านแต่ละด้านของรูปสามเหลี่ยมตามนี้

- ด้านตรงข้ามมุมฉาก (hypotenuse) คือ ด้านที่อยู่ตรงข้ามมุมฉาก หรือเป็นด้านที่ยาวที่สุดของรูปสามเหลี่ยมมุมฉาก ในที่นี้คือ h

- ด้านตรงข้าม (opposite side) คือ ด้านที่อยู่ตรงข้ามมุมที่สนใจ ในที่นี้คือ a

- ด้านประชิด (adjacent side) คือ ด้านที่อยู่ติดกับมุมที่สนใจและมุมฉาก ในที่นี้คือ b

จะได้

1). ไซน์ ของมุม คือ อัตราส่วนของความยาวด้านตรงข้าม ต่อ ความยาวด้านตรงข้ามมุมฉาก ในที่นี้คือ $\sin(A) =$ ข้าม/ฉาก $= a/h$

2). โคไซน์ ของมุม คือ อัตราส่วนของความยาวด้านประชิด ต่อความยาวด้านตรงข้ามมุมฉาก ในที่นี้คือ $\cos(A) =$ ชิด/ฉาก $= b/h$

3). แทนเจนต์ ของมุม คือ อัตราส่วนของความยาวด้านตรงข้าม ต่อความยาวด้านประชิด ในที่นี้คือ $\tan(A) =$ ข้าม/ชิด $= a/b$

4). โคซีแคนต์ $\csc(A)$ คือ ฟังก์ชันผกผันการคูณของ $\sin(A)$ นั่นคือ อัตราส่วนของความยาวด้านตรงข้ามมุมฉาก ต่อความยาวด้านตรงข้าม ในที่นี้คือ $\csc(A) =$ ฉาก/ข้าม $= h/a$

5). ซีแคนต์ sec(A) คือ ฟังก์ชันผกผันการคูณของ cos(A) นั่นคือ อัตราส่วนของความยาวด้านตรงข้ามมุมฉาก ต่อความยาวด้านประชิด ในที่นี้คือ sec(A) = ฉาก/ชิด = h/b

6). โคแทนเจนต์ cot(A) คือ ฟังก์ชันผกผันการคูณของ tan(A) นั่นคือ อัตราส่วนของความยาวด้านประชิด ต่อความยาวด้านตรงข้าม ในที่นี้คือ cot(A) = ชิด/ข้าม = b/a

2.6.2 เอกลักษณ์ตรีโกณมิติ

เอกลักษณ์ตรีโกณมิติ คือ การเท่ากันของฟังก์ชันตรีโกณมิติที่ต่างกัน และเป็นจริงสำหรับทุกๆค่าของขนาดของมุม โดยในการจัดทำโครงงานฉบับนี้จะใช้เอกลักษณ์ตรีโกณดังนี้

$$\sin A \cdot \csc A = 1$$

$$\cos A \cdot \sec A = 1$$

$$\tan A \cdot \cot A = 1$$

$$\cos A \cdot \tan A = \sin A$$

$$\sin A \cdot \cot A = \cos A$$

$$\sin^2 A + \cos^2 A = 1$$

$$\sec^2 A - \tan^2 A = 1$$

$$\sin(x + y) = \sin x \cos y + \cos x \sin y$$

$$\sin(x - y) = \sin x \cos y - \cos x \sin y$$

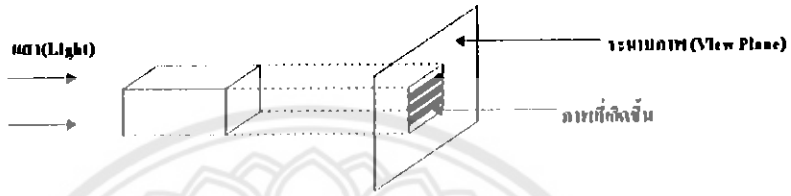
$$\cos(x + y) = \cos x \cos y - \sin x \sin y$$

$$\cos(x - y) = \cos x \cos y + \sin x \sin y$$

2.7 ภาพฉาย (Projection)

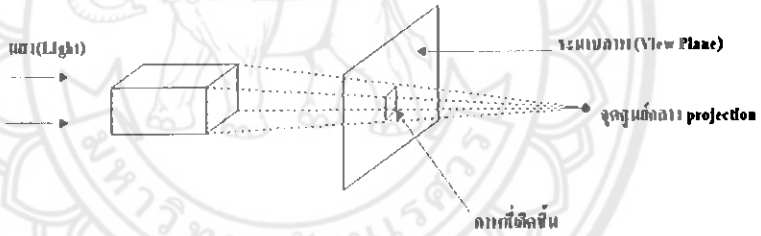
ภาพฉาย (Projection) คือ การสร้างภาพ 2 มิติ โดยการฉายเงาของวัตถุไปตกลงบนระนาบภาพ (View Plane) ซึ่งมี 2 ลักษณะ คือ

1. ภาพฉายแบบขนาน (Parallel Projection)



รูปที่ 2.9 ภาพฉายแบบขนาน (Parallel Projection)

2. ภาพฉายแบบเปอร์สเปคทีฟ (Perspective Projection)



รูปที่ 2.10 ภาพฉายแบบเปอร์สเปคทีฟ (Perspective Projection)

2.7.1 การหมุน (Rotation)

การหมุน (Rotation) เป็นการหมุนรูปหรือวัตถุรอบแกนที่กำหนดให้

- หมุนรอบแกน x

$y' = y\cos\theta - z\sin\theta$

$z' = y\sin\theta + z\cos\theta$

$x' = x$

$$[\,x'\ y'\ z'\,] = [\,x\ y\ z\,]\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{bmatrix}$$

- หมุนรอบแกน y

$x' = x\cos\theta + z\sin\theta$

$z' = -x\sin\theta + z\cos\theta$

$y' = y$

$$[\,x'\ y'\ z'\,] = [\,x\ y\ z\,]\begin{bmatrix} \cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

- หมุนรอบแกน z

$$x' = x\cos\theta + y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

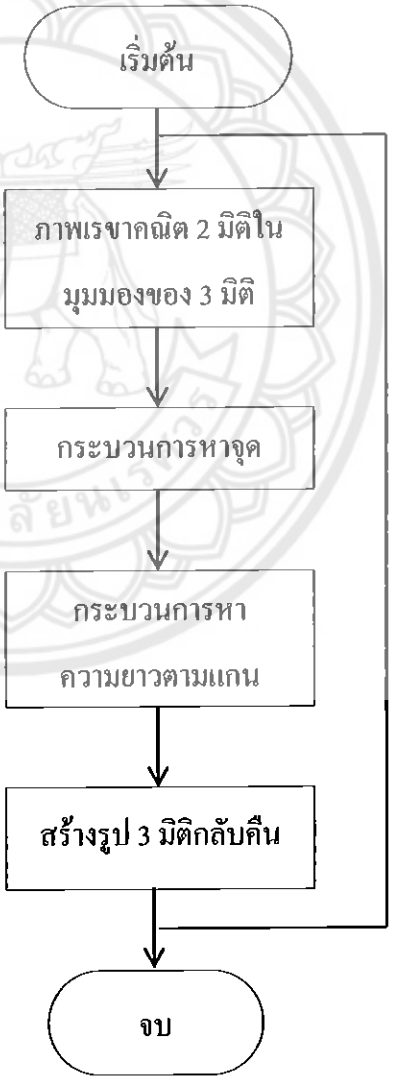
$$z' = z$$

$$[\,x'\ y'\ z'\,] = [\,x\ y\ z\,]\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

บทที่ 3
วิธีการดำเนินโครงงาน

เนื้อหาในบทนี้ได้อธิบายถึงขั้นตอนการดำเนินงาน โดยเริ่มตั้งแต่รายละเอียดการออกแบบอุปกรณ์ที่นำมาใช้ในการจัดทำโครงงานทั้งเครื่องมือที่เป็นฮาร์ดแวร์และซอฟต์แวร์ การลงมือสร้างงาน ซึ่งมีรายละเอียดดังนี้

3.1 ภาพรวมของโครงงาน



รูปที่ 3.1 แสดงวิธีการดำเนินโครงงาน

3.2 อุปกรณ์ที่ใช้

3.2.1 เครื่องคอมพิวเตอร์

- หน่วยประมวลผล Intel Core i5-450M 2.40 GHz
- หน่วยความจำ DDR3 ขนาด 2 GB
- การ์ดจอ NVIDIA GeForce GT_ 330M ขนาด 256 MB
- ระบบปฏิบัติการ Windows 7 แบบ 32-bits 512 MB

3.2.2 โปรแกรม

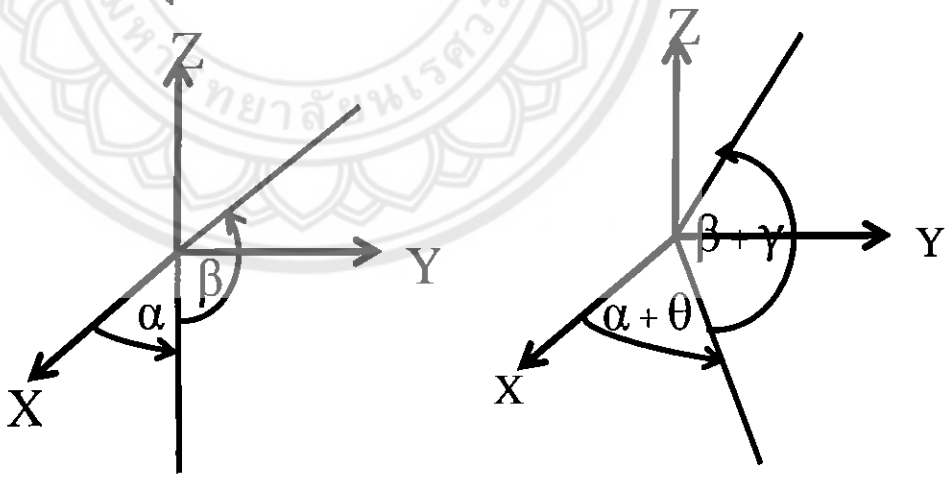
- MATLAB 2013 a
- Image Processing Toolbox

3.3 วิธีการดำเนินโครงงาน

3.3.1 การสร้างภาพเรขาคณิต 3 มิติ และ บันทึกภาพเป็นภาพเรขาคณิต 2 มิติ

การสร้างภาพเรขาคณิตรูปทรงลูกบาศก์และรูปทรงพีระมิด เพื่อที่จะใช้ในการสร้างรูป 3 มิติกลับคืนจากรูป 2 มิติ นั้น จะใช้โปรแกรมแมทแลบ (MATLAB) เวอร์ชั่น 2013a ในการสร้าง โดยจะเรียกชื่อมุมและมีวิธีการสร้าง ดังนี้

- กำหนดชื่อมุม



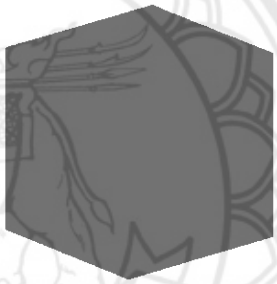
รูปที่ 3.2 แสดงชื่อมุมในการหมุนในแนวระนาบและแนวดิ่ง

จากรูปที่ 3.2 ค่ามุมแอลฟ่า (α) และ มุมเบต้า (β) คือ ค่ามุมของรูปที่หมุนไป โดย มุมแอลฟ่า (α) จะเป็นค่ามุมที่หมุนในแนวระนาบ (azimuth) ส่วนมุมเบต้า (β) จะเป็นค่ามุมเงย (หมุนในแนวดิ่ง) (elevation) และค่ามุมเซตา (θ) คือ ค่าความแตกต่าง

ระหว่างค่ามุมในแนวระนาบของรูปที่ 1 กับ รูปที่ 2 ส่วนค่ามุมแกมม่า (γ) คือ ค่าความแตกต่างระหว่างค่ามุมในแนวดิ่งของรูปที่ 1 กับ รูปที่ 3

- สร้างรูปทรงลูกบาศก์

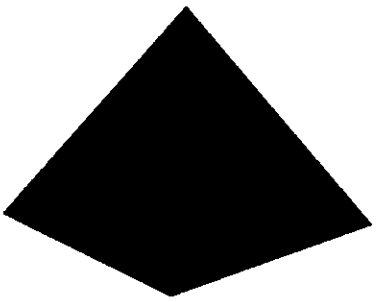
สร้างโดยใช้ฟังก์ชั่น patch ในโปรแกรมแมทแลบโดยกำหนดเป็น เมทริกซ์ (matrix) ของแต่ละด้านทั้งหมด 6 ด้าน และกำหนดมุมมองภาพเป็นภาพฉายแบบขนาน (orthographic) จากนั้นทำการกำหนดมุมแอลฟ่า และ มุมเบต้า และจะบันทึกไฟล์เป็นนามสกุล JPEG เนื่องจากรูปแบบ (format) ของภาพที่บันทึกผ่านโปรแกรมแมทแลบความละเอียดค่อนข้างต่ำ จึงใช้โปรแกรมอื่นปรับปรุงให้ภาพสมบูรณ์ขึ้น



รูปที่ 3.3 แสดงทรงลูกบาศก์จากการสร้างโดยใช้ฟังก์ชั่น patch

- สร้างรูปทรงพิระมิด

สร้างโดยใช้ฟังก์ชั่น patch ในโปรแกรมแมทแลปโดยกำหนดเป็น เมทริกซ์ ของแต่ละด้านทั้งหมด 5 ด้าน และกำหนดมุมมองภาพเป็นภาพฉายแบบขนาน จากนั้นทำการกำหนด มุมแอลฟ่า และ มุมเบต้า และจะบันทึกไฟล์เป็นนามสกุล JPEG เนื่องจากรูปแบบของภาพที่บันทึกผ่านโปรแกรมแมทแลบ ความละเอียดค่อนข้างต่ำ จึงใช้โปรแกรมอื่นปรับปรุงให้ภาพสมบูรณ์ขึ้น

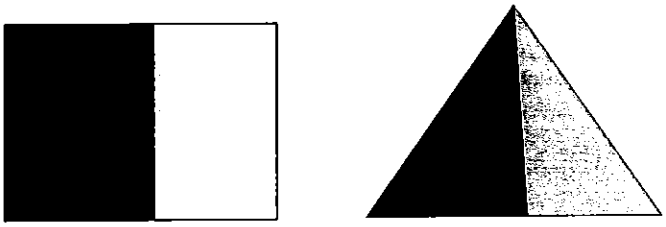


รูปที่ 3.4 แสดงทรงพีระมิดที่สร้างโดยฟังก์ชั่น patch

- สร้างลูกบาศก์และพีระมิดให้เห็นเฉพาะ 2 มิติ

ใช้วิธีการเดียวกันกับการสร้างภาพ 3 มิติ ในลูกบาศก์และพีระมิด แต่จะปรับในส่วนของค่าของมุมแอลฟ่า และ มุมเบต้า และจะบันทึกไฟล์เป็นนามสกุล JPEG

(ก)



(ข)

รูปที่ 3.5 (ก) ค่ามุมแอลฟ่า เท่ากับ 20 องศา (ข) ค่ามุมแอลฟ่า เท่ากับ 40 องศา

(ก)

(ข)

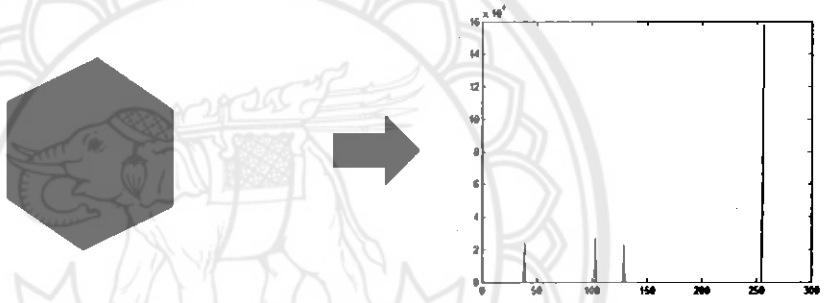
รูปที่ 3.6 (ก) ค่ามุมเบต้า เท่ากับ 20 องศา (ข) ค่ามุมเบต้า เท่ากับ 40 องศา

3.3.2 กระบวนการหาจุดมุมของภาพ (corner)

- หาจุดมุมของภาพรูปลูกบาศก์ โดยใช้ฟังก์ชั่น find_cube_corner

เป็นฟังก์ชั่นที่ใช้หามุมของภาพลูกบาศก์สองมิติเพื่อนำไปหาความยาว เพื่อใช้ในการประมวลผลแปลงรูปกลับเป็นลูกบาศก์สามมิติ

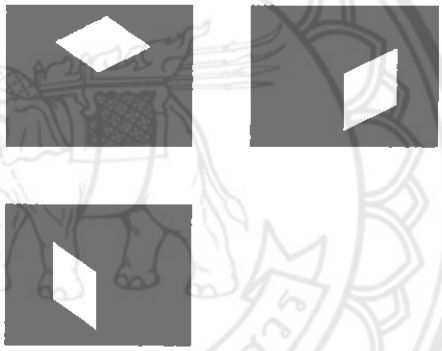
โดยขั้นแรกเราจะทำการแปลงรูปที่ได้จากภาพสี (RGB color) เป็นภาพระดับเทา (gray) และทำการหาค่าช่วงสีเทาที่น้อยที่สุดจนถึงค่าช่วงสีเทาที่มากที่สุด มาพลอต (plot) เป็นฮีสโตแกรม (Histogram) และทำการหาระดับสีเทา (gray level) ของแต่ละหน้าโดยใช้ฮีสโตแกรม



รูปที่ 3.7 แสดงกราฟฮีสโตแกรมของภาพลูกบาศก์

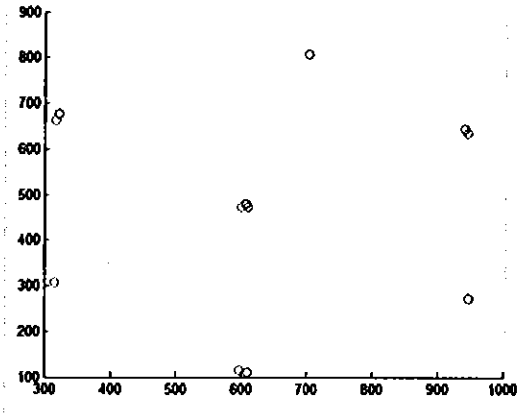
ในขั้นที่สองจำเป็นต้องทำการตรวจสอบว่าค่าระดับสีเทาที่ได้จะน้อยกว่า 3 หรือมากกว่า 10 หรือไม่ เพราะโดยทั่วไปแล้วภาพลูกบาศก์สองมิติเมื่อแปลงเป็นภาพสีเทาแล้วจะมีค่าระดับอยู่ต่ำสุดที่ 3 หน้าไม่รวมพื้นหลัง ด้วย และทำการตรวจสอบทีละด้านโดยทำการเรียงค่าระดับสีเทาที่ได้ และนำไปเก็บในเมทริกซ์ จากนั้นจึงทำการหาแต่ละด้านโดยใช้เงื่อนไขว่า หากค่าระดับสีเทา ณ ตำแหน่งสุดท้าย เท่ากับค่าสีสูงสุดคือ 256 (สีขาว) แล้วให้ทำการจับคู่โดยให้ค่าระดับสีเทา เท่ากับ ค่าระดับสีเทา ณ ตำแหน่งที่ 1 และ ตำแหน่งสุดท้ายลบออกหนึ่งตำแหน่ง เพื่อเป็นการกำจัดพื้นหลังที่เป็นสีขาวออกไป และสร้างเมตริกซ์ว่างขึ้นมา 1 ตัว โดยให้ขนาดเท่ากับค่า ระดับสีเทา ที่ได้จากฮีสโตแกรมลบด้วยหนึ่ง

ในขั้นที่สามให้ทำการสร้างเมทริกซ์ที่เป็น 0 ตามขนาดของ ค่า ระดับสีเทาลบด้วยหนึ่ง นั่นคือสามด้านนั่นเอง จากนั้นนำมาใส่ลูปโดยทำการวนตั้งแต่ 1 จนถึง ขนาดของค่าระดับสีเทาลบด้วยหนึ่ง ให้ imbw คือ ค่า ระดับสีเทา ทั้งหมดของภาพ และ v คือ ค่าระดับสีเทา ที่ได้จาก ฮีสโตแกรม และเมื่อให้ imbw > v(i) โดยหาก i = 1 แล้ว ให้เท่ากับ fil1 นั่นหมายความว่า fil1 เป็นค่าระดับสีเทาทั้งหมดของภาพมากกว่าค่า ระดับสีเทาที่ได้มาจากฮีสโตแกรมทั้งหมด จะให้ค่าเป็น 1 หลังจากนั้น ให้ fil2 = imbw <= v(i+1) โดยหาก i = 1 แล้ว นั่นหมายความว่า fil2 คือ พื้นที่ทั้งหมดที่มีค่าน้อยกว่าหรือเท่ากับค่าระดับสีเทาตัวถัดมาที่ได้จากค่า ฮีสโตแกรม สุดท้ายจึงให้หน้า (face) โดยทุกๆ i เท่ากับค่าของเมทริกซ์ fil1 และ fil2 มาคูณกันก็จะได้แต่ละส่วนของภาพออกมา



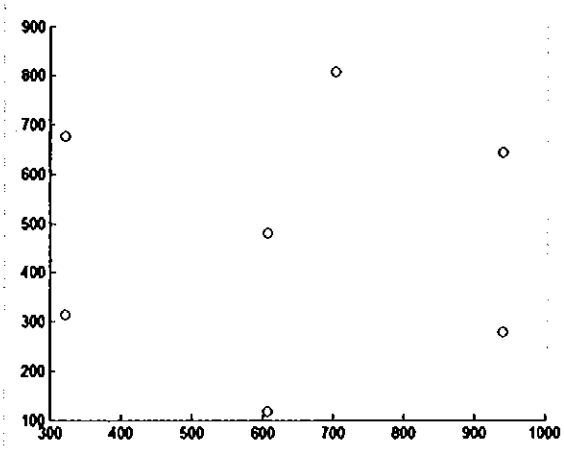
รูปที่ 3.8 แสดงค่าหน้า (face) แต่ละหน้าของภาพลูกบาศก์ที่ถูกแยกส่วนออกมา

ในขั้นที่สี่ ขั้นนี้จะเป็นการหามุมโดยใช้คำสั่ง corner เข้ามาช่วย โดยตอนแรกจะกำหนดจำนวน corner ขึ้นมาก่อนในที่นี้เป็นรูปลูกบาศก์ ซึ่งแต่ละด้านมีสี่มุม เราจึงต้องใช้จำนวน corner เท่ากับ 4 และกำหนดเมทริกซ์ใหม่เพื่อทำการพลอต corner ขึ้นไปในเมทริกซ์นั้น และทำการใส่ลูปเพื่อรันแต่ละด้านออกมาแล้วจึงทำการกัดกร่อนเส้น และทำการปรับความละเอียดใหม่ตามความเหมาะสม และทำคำสั่ง corner พลอตจุดลงไป ขณะที่เราจะทำการพลอตจุดลงกราฟ เราจะต้องปรับภาพให้สลับบนล่างก่อนเพราะการกระทำภาพกับกราฟในโปรแกรมแมทแลบ แกนไม่เท่ากันเสมอไป จากนั้นทำการพลอตจุดลงไปในกราฟ



รูปที่ 3.9 แสดงการทำงานของฟังก์ชั่น corner ในแต่ละหน้าของภาพลูกบาศก์

จากรูปที่ 3.9 จะสังเกตได้ว่าจุดทุกจุดจะเป็นไปตามมุมทุกมุมของด้านสามด้าน แต่จะนำไปใช้เพียง 7 จุดเท่านั้นจึงทำการปรับปรุงจุดโดยใช้ 4 จุดบน นั่นคือจุดสีเขียว จุดสีแดง 2 จุด และ จุดสีน้ำเงิน 1 จุด แล้วใช้ความยาวแกนกลางของจุดสีแดงเป็นความยาวหลัก โดยจะใช้วิธีการกำกับจุดของแต่ละด้าน โดยเริ่มจากสีเขียวก่อน โดยหน้าของจุดสีเขียวไม่ว่ารูปจะหมุนไปในทิศทางใด จุดสีเขียว จะมี ค่า x น้อยกว่าทุกด้านเสมอ ดังนั้นจะสามารถใช้ค่าที่น้อยที่สุดในการหาจุดสีเขียวของด้านจุดสีเขียว ซึ่งบางทีเราจะได้ค่าน้อยที่สุดออกมาสองค่า คือ จุดล่างกับจุดบน ดังนั้นเราจำเป็นต้องคำนวณหาค่าสูงสุดของ y ในจุดสีเขียวสองจุดที่มีค่า x น้อยที่สุด เพื่อนำมาเปรียบเทียบว่าจุดใดคือจุดที่ใช่ และตั้งชื่อจุดนั้นว่าจุด p1 ในส่วนของด้านจุดสีแดงจะใช้วิธีการเดียวกับหาจุดสีเขียว และตั้งชื่อจุดนั้นว่าจุด p2 จะเห็นได้ว่าจำเป็นที่ต้องใช้ความยาวระหว่างด้านจุดสีแดงที่มีค่า x น้อยที่สุด 2 ค่ามาหาความยาวและเก็บความยาวนั้นไว้ในตัวแปรความยาวนั้นที่ชื่อว่า L จากนั้นทำการหาจุด p3 โดยใช้วิธีการเดียวกับการหา p2 แต่จะเปลี่ยนโดยใช้ค่าสูงสุดของ x เป็นตัวกำหนดจุดแล้วจึงมาเปรียบเทียบค่า y ของค่าสูงสุดของ x อีกที จึงจะได้จุด p3 สำหรับจุดที่อยู่เป็นฐานลูกบาศก์ริมซ้ายสุด ให้ใช้ความยาว L ที่เราได้ เป็นระยะห่างเพื่อหาจุดฐาน โดยตั้งชื่อจุดเหล่านั้นว่า p4 , p5 , p6 โดย p4 เกิดจากการกำเนิดของ p1 ส่วน p5 เกิดจากการกำเนิดของ p2 สุดท้าย p6 กำเนิดจาก p3 ส่วนจุดที่มีความสูง y มากที่สุดในหน้าของจุดสีน้ำเงินให้ตั้งชื่อจุดนั้นว่าจุด p7 เมื่อได้ครบแล้วนำไปเก็บในอาเรย์ใหม่ที่ชื่อว่า Pointface แล้วนำไปเรียกใช้เพื่อจัดการขึ้นรูปสามมิติ

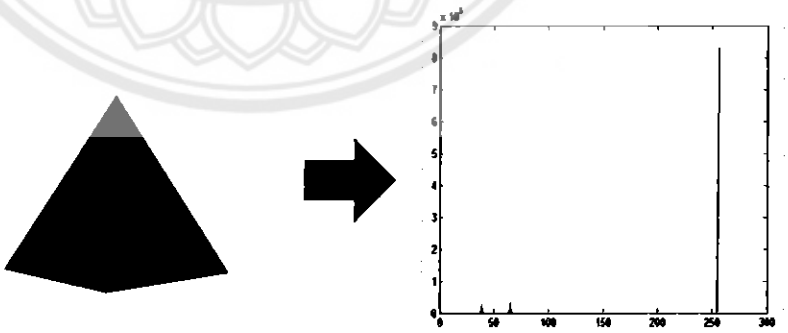


รูปที่ 3.10 แสดงกราฟที่ถูกพลอตจุดใหม่ลงไปของภาพลูกบาศก์

- หาจุดมุมของภาพรูปพีระมิด โดยใช้ฟังก์ชั่น find_cube_pyramid

เป็นฟังก์ชั่นที่ใช้หามุมของภาพพีระมิดสองมิติเพื่อนำไปหาความยาวเพื่อใช้ไปประมวลแปลงรูปกับเป็นพีระมิดสามมิติ

โดยขั้นแรกเราจะทำการแปลงรูปที่ได้จากภาพสี เป็น ภาพระดับเทาและทำการหาค่าช่วงสีเทาที่น้อยที่สุดและมากที่สุดมาพลอต เป็นฮีสโตแกรม และทำการหาระดับสีเทาของแต่ละหน้าโดยใช้ ฮีสโตแกรม



รูปที่ 3.11 แสดงกราฟฮีสโตแกรมของภาพพีระมิด

ในขั้นที่สองจำเป็นต้องทำการเช็คว่าค่าระดับสีเทาที่ได้จะน้อยกว่า 2 หรือ มากกว่า 10 หรือไม่ เพราะโดยทั่วไปแล้วภาพพีระมิดสองมิติเมื่อแปลงเป็นภาพสีเทาแล้วจะมีค่าระดับอยู่ต่ำสุดที่ 3 หน้า ไม่รวมพื้นหลังด้วย และทำการตรวจสอบทีละด้านโดยทำการเรียง

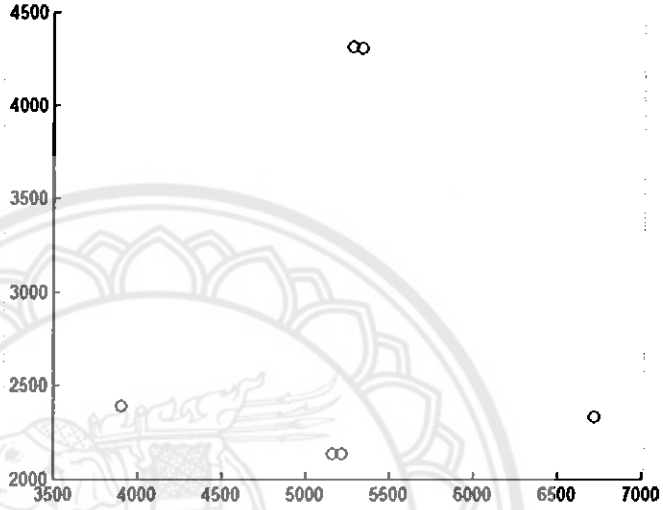
ค่าระดับสีเทาที่ได้ และทำไปเก็บในเมทริกซ์ จากนั้นจึงทำการหาแต่ละด้านโดยใช้เงื่อนไข ว่า หากระดับสีเทา ณ ตำแหน่งสุดท้าย เท่ากับค่าสีสูงสุดคือ 256 (สีขาว) แล้วทำการจับคู่ โดยให้ค่า ระดับสีเทา เท่ากับ ค่า ระดับสีเทา ณ ตำแหน่งที่ 1 และ ตำแหน่งสุดท้ายลบออก หนึ่งตำแหน่ง เพื่อเป็นการกำจัดพื้นหลังที่เป็นสีขาวออกไป และสร้างเมทริกซ์ว่างขึ้นมา 1 ตัว โดยให้ขนาดเท่ากับค่า ระดับสีเทา ที่ได้จากฮีสโตแกรมลบด้วยหนึ่ง

ในขั้นที่สามให้ทำการสร้างเมทริกซ์ที่เป็น 0 ตามขนาดของ ค่าระดับสีเทาลบด้วย หนึ่ง นั่นคือสามด้านนั่นเอง จากนั้นนำมาใส่ลูปโดยทำการวนตั้งแต่ 1 ถึง ขนาดของค่า ระดับสีเทาลบด้วยหนึ่ง ให้ imbw คือค่าระดับสีเทา ทั้งหมดของภาพ และ v คือ ค่าระดีบสีเทา ที่ได้จากฮีสโตแกรม และเมื่อให้ imbw > v(i) โดยหาก i = 1 แล้ว ให้เท่ากับ fil1 นั่นหมายความว่า fil1 เป็นค่า ระดับสีเทา ทั้งหมดของภาพมากกว่าค่าระดับสีเทา ที่ ได้มาจากฮีสโตแกรมทั้งหมด จะให้ค่าเป็น 1 หลังจากนั้น ให้ fil2 = imbw <= v(i+1) โดย หาก i = 1 แล้ว นั่นหมายความว่า fil2 คือพื้นที่ทั้งหมดที่มีค่าน้อยกว่าหรือเท่ากับค่า ระดับสีเทาตัวถัดมาที่ได้จากค่า ฮีสโตแกรม สุดท้ายจึงให้หน้า (face) โดยทุกๆ i เท่ากับค่า ของเมทริกซ์ fil1 และ fil2 มาคูณกันก็จะได้แต่ละส่วนของภาพออกมา

รูปที่ 3.12 แสดงค่าหน้า (face) แต่ละหน้าในภาพพีระมิดที่ถูกแยกส่วนออกมา

ในขั้นที่สี่ ขั้นนี้จะเป็นการหามุมโดยใช้คำสั่ง corner เข้ามาช่วย โดยตอนแรกจะ กำหนดจำนวนมุมขึ้นมาก่อนในที่นี้เป็นรูปพีระมิด ซึ่งแต่ละด้านมีสามมุม เราจึงต้องใช้ จำนวนมุม เท่ากับ 3 และกำหนดเมทริกซ์ใหม่เพื่อทำการพลอตมุม ขึ้นไปในเมทริกซ์นั้น

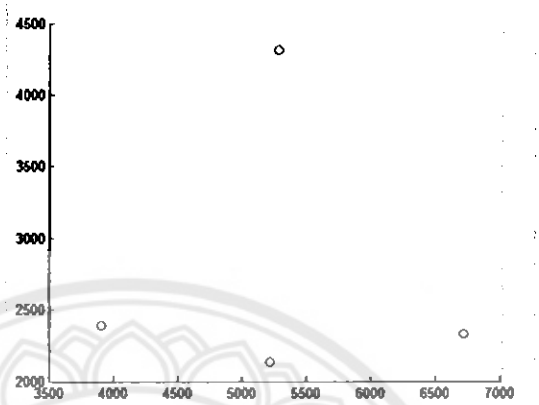
และทำการใส่ลูปเพื่อรันแต่ละด้านออกมา และทำการปรับความละเอียดใหม่ตามความเหมาะสม และทำคำสั่ง corner พลอตจุดลงไป ขณะที่เราจะทำการพลอตจุดลงกราฟ เราจะต้องปรับภาพให้สลับบนล่างก่อนเพราะการกระทำภาพกับกราฟในโปรแกรม แมทแลบแกนไม่เท่ากันเสมอไปจากนั้นทำการพลอตจุดลงไปลงกราฟ



รูปที่ 3.13 แสดงการทำงานของฟังก์ชั่น corner ในแต่ละหน้าของภาพพีระมิด

จากรูปเราจะสังเกตได้ว่าจุดทุกจุดจะเป็นไปตามมุมทุกมุมของด้านสองด้าน แต่เราจะนำไปใช้เพียง 4 จุดเท่านั้น เราจึงทำการปรับปรุงจุดโดยให้จุดทุกจุดแต่ละด้านเป็นอิสระต่อกัน จากนั้นทำการหาค่าสูงสุด และ ค่าสูงสุดของ x และนำไปเปรียบเทียบกับทุกจุดเพื่อได้จุดสองจุดของมุมซ้ายและมุมขวาของรูปออกมาและตั้งชื่อให้จุดสองจุดนั้น คือ p1 และ p3 ตามลำดับ เมื่อได้จุดสองจุดแล้ว ทำการลบสองจุดนั้นออกจากอาเรย์ของจุดทั้งหมดจะเหลือทั้งหมด 4 จุด หาจุดล่างจุดบน โดยแบ่งเป็นค่าต่ำสุดของ y ที่น้อยสุดสองค่า และค่าสูงสุดของ y ที่มากที่สุด 2 ค่า และนำไปเปรียบเทียบกับจุดทั้งหมด ก็จะได้จุด สองจุดล่างกับ บนแบ่งกันชัดเจน เพื่อให้มันอิสระต่อกัน จากนั้นนำสองจุดล่างมาคิดก่อนโดยใช้จุดค่า x ของจุดล่างสองจุดเปรียบเทียบกันว่าจุดใดใกล้กับจุด p1 ถ้าจุดไหนใกล้กว่าให้ใช้จุดนั้นโดยตั้งชื่อจุดนั้นว่า p2 จุดบนทำเช่นเดียวกับจุดล่างเมื่อได้จุดที่ใกล้ p1 ให้ใช้จุดนั้น

และตั้งชื่อว่า p4 เมื่อได้ครบแล้วนำไปเก็บในอาเรย์ใหม่ที่ชื่อว่า Pointface แล้วนำไปเรียกใช้เพื่อจัดการขึ้นรูปสามมิติ



รูปที่ 3.14 แสดงกราฟที่ถูกพลอตจุดใหม่ลงไปของภาพพิระมิด

- หาจุดมุมของภาพรูปลูกบาศก์และพีระมิดที่เห็นใน 1 มิติ

การหาจุดที่ใช้ได้กับภาพลูกบาศก์และพีระมิตในสามมิติ ในขั้นแรกเหมือนกันทุกอย่าง จนถึงขั้นตอนของการปรับปรุงจุด ซึ่งขั้นตอนนี้จะไม่ต้องใช้ เนื่องจากในมุมมอง 1 มิติจะมีระนาบอย่างน้อยแค่สองด้านดังนั้นจุดที่ได้ค่อนข้างเสถียรอยู่แล้ว จึงสามารถใช้จุดเหล่านั้นไปคำนวณหาความยาวได้

3.3.3 กระบวนการหาความยาวตามแกน

3.3.3.1 การหาค่าความยาวด้านใน 1 มิติ ของรูป 2 มิติ (Test case)

- รูปทรงลูกบาศก์

การวิเคราะห์หาสูตรที่จะใช้ในการคำนวณหาค่าความยาวด้านของรูป

รูปที่ 3.15 ภาพแสดงความยาวของด้านที่ต้องการคำนวณ

จากรูปที่ 3.15 จะเห็นว่า เมื่อหมุนรูปทรงลูกบาศก์ในแนวระนาบด้วยมุมแอลฟ่า องศา (แนวดิ่งเท่ากับ 0) จะเห็นรูปสี่เหลียมมีขนาดเล็กลง และยิ่งหมุนกล้องในแนวเดิมเพิ่มมากขึ้นเรื่อยๆ รูปสี่เหลี่ยมก็จะยิ่งเล็กลง และเมื่อใช้ทฤษฎีตรีโกณมิติ จะได้ว่า

$$L_1 = L\cos\theta_1$$

$$L_2 = L\cos\theta_2$$

จัดรูปสมการใหม่ ; $L_1 = L\cos\alpha$ ---------- สมการที่ 1

$L_2 = L\cos(\alpha + \theta)$ ---------- สมการที่ 2

สมการที่ 1 หารด้วย สมการที่ 2 ; $\frac{L1}{L2} = \frac{Lcos\,\alpha}{Lcos(\alpha+\theta)}$

จากเอกลักษณ์ตรีโกณ ; $\cos(x+y) = \cos x \cos y - \sin x \sin y$

แทนค่าเอกลักษณ์ตรีโกณ ; $\frac{L1}{L2} = \frac{\cos\alpha}{\cos(\alpha+\theta)}$

$$\frac{L_1}{L_2} = \frac{\cos\alpha}{\cos\alpha\cos\theta - \sin\alpha\sin\theta}$$

$$L_1(\cos\alpha\cos\theta - \sin\alpha\sin\theta) = L_2(\cos\alpha)$$

$$\frac{L_2}{L_1} = \frac{\cos\alpha\cos\theta - \sin\alpha\sin\theta}{\cos\alpha}$$

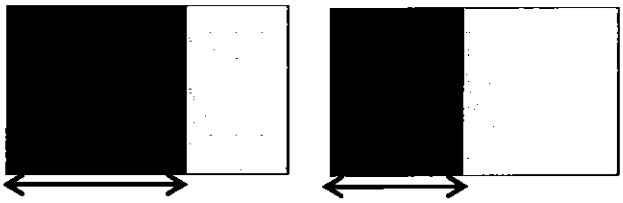
$$L_2/L_1 = \cos\theta - \tan\alpha\sin\theta$$

$$\alpha = \tan^{-1}\left(\frac{\cos\theta - L_2/L_1}{\sin\theta}\right)$$

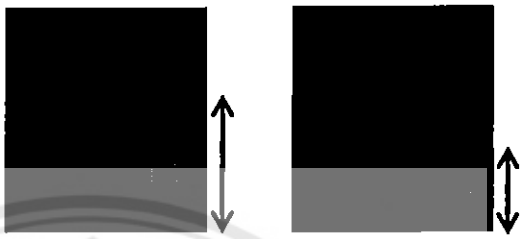
จากสมการที่ 1 ; $$L = \frac{L1}{\cos\alpha}$$

หมายเหตุ สูตรที่หาได้นั้นสามารถใช้ได้ทั้ง รูปที่หมุนในแนวระนาบและแนวดิ่ง (ทั้งรูปทรงลูกบาศก์และทรงพีระมิด (เฉพาะภาพใน 2 มิติที่ต้องการหาความยาวด้าน (L) ใน 1 มิติ เท่านั้น))

เมื่อได้สูตรหาความยาวในแนวระนาบและแนวดิ่งมาแล้ว **จากนั้นทำการทดลอง** โดยหาความยาวโดยใช้สูตรที่ได้โดยเริ่มตั้งแต่ความยาวในแนวระนาบ **ซึ่งจะใช้รูปอินพุท** อย่างน้อยสองรูปขึ้นไป ซึ่งรูปที่ใช้ทดลอง คือ มุมเริ่มต้น มุมแอลฟ่า **สองภาพต่างกัน แต่** มุมเบต้า ต้องเป็น 0 ซึ่งจะได้ความยาวด้าน ที่เกิดขึ้นมา จากนั้น**ก็ทำการหาความยาวใน** แนวดิ่ง ซึ่งจะใช้รูปอินพุท อย่างน้อยสองรูปขึ้นไป ซึ่งรูปที่ใช้ทดลอง **คือ มุมเริ่มต้นมุม** เบต้า สองภาพต่างกัน แต่ มุมแอลฟ่า ต้องเป็น 0 ซึ่งจะได้ความยาว**ด้านที่เกิดขึ้นมา**



(ก)

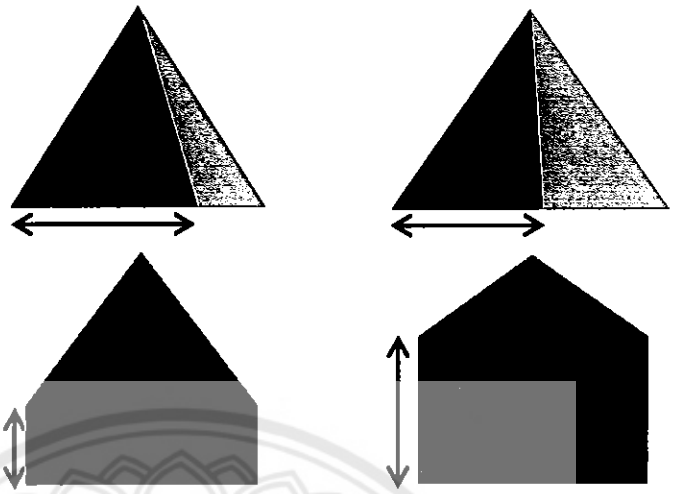


(ข)

รูปที่ 3.16 (ก) แสดงความยาวด้านที่ใช้ในการคำนวณของรูปที่ปรับค่ามุมในแนวระนาบ
(ข) แสดงความยาวด้านที่ใช้ในการคำนวณของรูปที่ปรับค่ามุมในแนวดิ่ง

เมื่อทำการทดลองจะได้ผลลัพธ์ เป็นค่าของมุมแอลฟ่า , ค่าของมุมเบต้า และค่าความยาวด้าน ซึ่งค่าดังกล่าว คือ ค่าของมุมเริ่มต้นของรูปที่เกิดจากการหาค่ามุมแอลฟ่า (มุมเบต้า เท่ากับ 0) , ค่าของมุมเริ่มต้นของรูปที่เกิดจากการค่ามุมเบต้า (มุมแอลฟ่า เท่ากับ 0) และ ค่าความยาวที่ได้จากการคำนวณ ซึ่งจะมีค่าใกล้เคียงกับค่าความยาวจริงของภาพที่เห็นเพียงด้านเดียว โดยก่อนการรันไฟล์ทุกครั้ง จะต้องทำการปรับค่า มุมเซตา และ มุมแกมม่า ด้วยทุกครั้ง (มุมเซตา คือ ผลต่างระหว่างมุมแอลฟ่าของรูปอินพุต 2 รูป และ มุมแกมม่า คือ ผลต่างระหว่างมุมเบต้าของรูปอินพุต 2 รูป)

- รูปทรงพีระมิด



รูปที่ 3.17 แสดงความยาวด้านที่ใช้ในการคำนวณทั้งในแนวระนาบและแนวดิ่ง

ในการวิเคราะห์หาสูตรที่จะใช้ในการคำนวณหาค่าความยาวด้านใน 1 มิติ ของรูปทรงพีระมิด นั้น สามารถใช้สูตรเดียวกันกับรูปทรงลูกบาศก์ได้ แต่จะเข้าไปแก้ไขในส่วนของ ฟังก์ชั่น FindCornerCube โดยให้ทำการปรับค่าที่ nocorner จาก 4 เป็น 3 และปรับค่า resizefactor จาก 0.15 และสุดท้ายเนื่องจากมุมมีการกระจายตัว จึงจะให้ไฟล์ที่ใช้ในการรัน จะตัดในส่วนของ test_function ออก สามารถเขียนได้ดังนี้

```
clear all ;
clc;
close all;
warning('off');

disp('Picture1==========================================================');
imrgb1 = imread('testcube_A0.jpg');
Face_A = FindCornerCube(imrgb1,1,1)

imrgb2 = imread('testcube_A5.jpg');
Face_B = FindCornerCube(imrgb2,2,1)
```

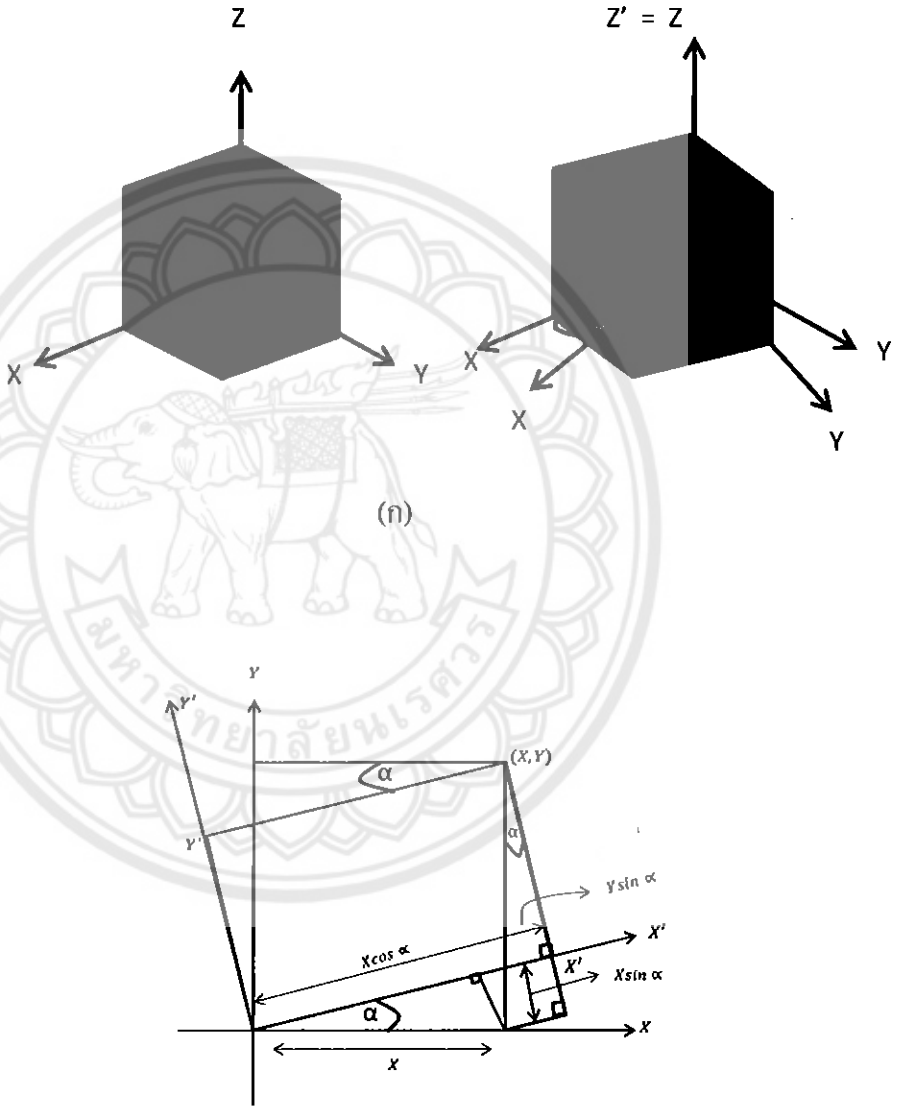
```
imrgb3 = imread('testcube_A7.jpg');
Face_C = FindCornerCube(imrgb3,3,1)

theta = 20

L1 =
L2 =

%Calculate alpha and Length
alpha = atand( (cosd(theta) - L2/L1)/sind(theta) )
L = L1/cosd(alpha)
```

หมายเหตุ ค่า L1 และ L2 ได้จากข้อมูลที่เกิดจากการรัน ซึ่งในการรัน จะได้ตำแหน่งของจุด แล้วนำมาหาระยะห่าง หลังจากนั้นจึงค่อยนำค่าที่ได้มาใส่ในไฟล์ที่ใช้รันอีกครั้ง

3.3.3.2 การหาค่าความยาวด้านใน 2 มิติของรูปใน 2 มิติ

- รูปทรงลูกบาศก์

วิเคราะห์หาสูตรที่ใช้ในการหาความยาวด้านของรูปใน 2 มิติ



(ข)

รูปที่ 3.18 (ก) แสดงการหมุนในแนวระนาบของรูปทรงและแนวแกน

(ข) แสดงการดูภาพตามแกนเดียว

จากรูปที่ 3.18 (ข) สามารถเขียนเป็นสมการได้ดังนี้

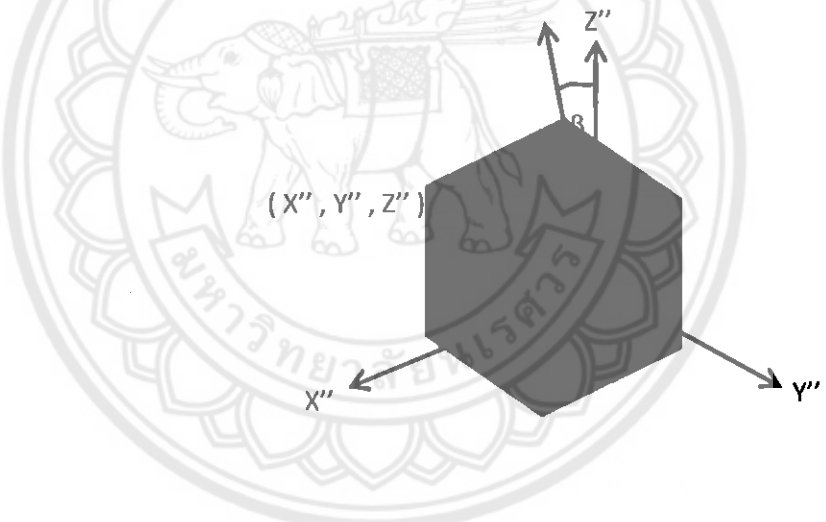
$$X' = X\cos\alpha + Y\sin\alpha$$

$$Y' = -X\cos\alpha + Y\sin\alpha$$

จัดรูปให้อยู่ในรูปแบบของเมทริกซ์ได้ ดังนี้

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z' \end{bmatrix}$$

$$\vec{r'} = R_z(\alpha) \cdot \vec{r}$$



รูปที่ 3.19 แสดงการหมุนในแนวดิ่งของรูปทรงและแนวแกน

จากรูปที่3.19 สามารถเขียนเป็นสมการได้ดังนี้

$$X'' = X'\cos\beta + Z'\sin\beta$$

$$Z'' = -X'\sin\beta + Z'\cos\beta$$

$$Y'' = Y'$$

จัดรูปให้อยู่ในรูปแบบของเมทริกซ์ได้ดังนี้

$$\begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}$$

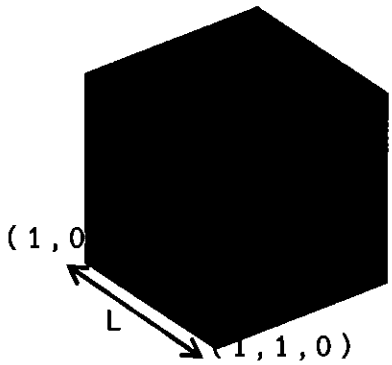
$$\vec{r}'' = R_y(\beta)\cdot\vec{r}$$

$$\vec{r}'' = R_y(\beta)\cdot R_z(\alpha)\cdot\vec{r}$$

$$\begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix} = \begin{bmatrix} \cos\alpha\cos\beta & \sin\alpha\cos\beta & \sin\beta \\ -\sin\alpha & \cos\alpha & 0 \\ -\cos\alpha\cos\beta & -\sin\alpha\cos\beta & \cos\beta \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix} = \begin{bmatrix} X\cos\alpha\cos\beta + Y\sin\alpha\cos\beta + Z\sin\beta \\ -X\sin\alpha + Y\cos\alpha \\ -X\cos\alpha\cos\beta - Y\sin\alpha\cos\beta + Z\cos\beta \end{bmatrix} \begin{matrix} \rightarrow \text{X บน Z} \\ \\ \rightarrow \text{Y บน Z} \end{matrix}$$

เนื่องจากดูตามแกน X ค่าที่จะนำมาใช้ จะใช้เฉพาะ Y กับ Z เท่านั้น



รูปที่ 3.20 แสดงตำแหน่งที่ใช้ในการหาระยะห่างระหว่างจุด

จากรูปที่ 3.20 เมื่อทำการแทนค่าตำแหน่ง (1,0,0) และ (1,1,0) ลงในสมการ แล้วนำมาหาระยะห่างระหว่างจุด ได้ดังนี้

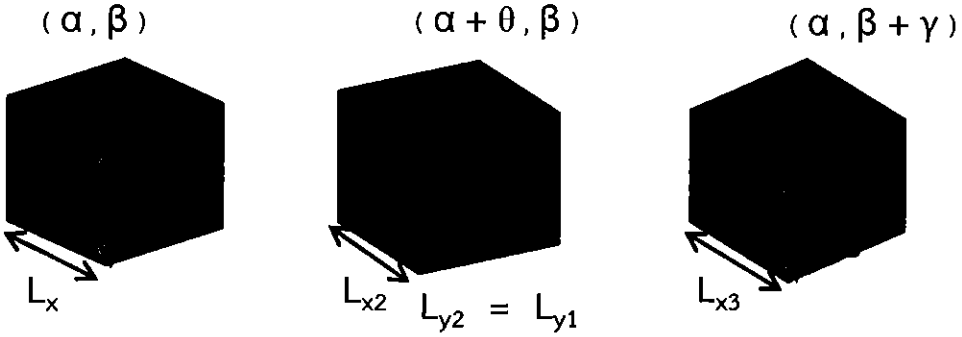
$$\sqrt[2]{(x_1 - x_2)^2 + (y_1 - y_2)^2} = \sqrt[2]{\cos^2\alpha + \sin^2\alpha \sin^2\beta}$$

$$= \sqrt[2]{\cos^2\alpha + \sin^2\alpha \sin^2\beta \pm \sin^2\beta}$$

$$= \sqrt[2]{1 + \sin^2\alpha (\sin^2\beta - 1)}$$

$$= \sqrt[2]{1 + \sin^2\alpha \cos^2\beta}$$

เมื่อได้สูตรระยะห่างระหว่างจุด หรือ นั่นก็คือ ค่าความยาวด้านของรูปใน 2 มิติ แล้ว ต่อไปจะนำสูตรนี้ไปใช้ในการหาสูตรที่ใช้หาค่าของมุมแอลฟ่า ดังนี้



รูปที่ 3.21 ภาพแสดงความยาวด้านและมุมของภาพที่ใช้ในการคำนวณ

จากรูปที่ 3.21 เมื่อแทนสูตรดังกล่าว จะสามารถเขียนเป็นสมการใหม่ได้ดังนี้

$$L_1 = L\sqrt[2]{1-\sin^2\alpha\cos^2\beta} \quad \text{---------- สมการที่ 1}$$

$$L_2 = L\sqrt[2]{1-\sin^2(\alpha+\theta)\cos^2\beta} \quad \text{---------- สมการที่ 2}$$

$$L_3 = L\sqrt[2]{1-\sin^2\alpha\cos^2(\beta+\gamma)} \quad \text{---------- สมการที่ 3}$$

ในการหาสูตรที่ใช้กับแนวระนาบจะใช้ สมการที่ 1 กับ 2 จะได้ดังนี้

สมการที่ 1 ยกกำลัง 2 ; $L_1^2 = L^2(1-\sin^2\alpha\cos^2\beta)$

สมการที่ 2 ยกกำลัง 2 ; $L_2^2 = L^2(1-\sin^2(\alpha+\theta)\cos^2\beta)$

กำหนดให้ $l_1 = \frac{L_1}{L}$ และ $l_2 = \frac{L_2}{L}$

แทนค่า l_1 และ l_2 พร้อมทั้งนำ สมการที่ 2 หารด้วย สมการที่ 1 จะได้

$$\frac{1-l_2^2}{1-l_1^2} = \frac{\sin^2(\alpha+\theta)\cos^2\beta}{\sin^2\alpha\cos^2\beta}$$

กำหนดให้ $k_1^2 = \frac{1-l_2^2}{1-l_1^2}$ พร้อมทั้ง แทนค่า จะได้

$$k_1^2 = \frac{\sin^2(\alpha+\theta)\cos^2\beta}{\sin^2\alpha\cos^2\beta}$$

$$k_1^2 = \frac{\sin^2(\alpha+\theta)}{\sin^2\alpha}$$

$$k_1^2 \cdot \sin^2\alpha = \sin^2(\alpha+\theta)$$

$$k_1 \cdot \sin\alpha = \sin(\alpha+\theta)$$

จากเอกลักษณ์ตรีโกณ ; $\sin(x+y) = \sin x\cos y - \cos x\sin y$

แทนค่าเอกลักษณ์ตรีโกณ ; $k_1 \cdot \sin\alpha = \sin\alpha\cos\theta + \cos\alpha\sin\theta$

$$(k_1 - \cos\alpha) \cdot \sin\alpha = \cos\alpha \sin\theta$$

$$\frac{(k_1 - \cos\alpha) \cdot \sin\alpha}{\cos\alpha} = \sin\theta$$

$$(k_1 - \cos\alpha) \cdot \tan\alpha = \sin\theta$$

$$\tan\alpha = \frac{\sin\theta}{k_1 - \cos\alpha}$$

$$\alpha = \tan^{-1}\left(\frac{\sin\theta}{k_1 - \cos\alpha}\right)$$

และ

$$L = \frac{L_1}{\sqrt[2]{1 - \sin^2\alpha \cos^2\beta}}$$

ในส่วนการหาสูตรที่ใช้กับแนวดิ่งจะใช้ สมการที่ 1 กับ 3 จะได้ดังนี้

สมการที่ 1 ยกกำลัง 2 ; $L_1^2 = L^2 (1 - \sin^2\alpha \cos^2\beta)$

สมการที่ 3 ยกกำลัง 2 ; $L_3^2 = L^2 (1 - \sin^2\alpha \cos^2(\beta + \gamma))$

กำหนดให้ $l_1 = \frac{L_1}{L}$ และ $l_3 = \frac{L_3}{L}$

แทนค่า l_1 และ l_3 พร้อมทั้งนำ สมการที่ 3 หารด้วย สมการที่ 1 จะได้

$$\frac{1 - l_3^2}{1 - l_1^2} = \frac{\sin^2\alpha \cos^2(\beta+\gamma)}{\sin^2\alpha \cos^2\beta}$$

กำหนดให้ $k_2^2 = \frac{1 - l_3^2}{1 - l_1^2}$ พร้อมทั้ง แทนค่า จะได้

$$k_2{}^2 = \frac{\sin^2\alpha \cos^2(\beta+\gamma)}{\sin^2\alpha \cos^2\beta}$$

$$k_2{}^2 \cdot \sin^2\alpha \cos^2\beta = \sin^2\alpha \cos^2(\alpha+\theta)$$

$$k_2 \cdot \sin\alpha \cos\beta = \sin\alpha \cos(\beta+\gamma)$$

$$k_2 \cdot \cos\beta = \cos(\beta+\gamma)$$

จากเอกลักษณ์ตรีโกณมิติ $\cos(x+y) = \cos x \cos y - \sin x \sin y$

แทนค่าเอกลักษณ์ตรีโกณมิติ ; $k_2 \cdot \cos\beta = \cos\beta\cos\gamma - \sin\beta\sin\gamma$

$$(k_2 - \cos\gamma) \cdot \cos\beta = -\sin\beta\sin\gamma$$

$$\frac{(k_2 - \cos\gamma) \cdot \cos\beta}{\sin\beta} = -\sin\gamma$$

$$\frac{(k_2 - \cos\gamma)}{\tan\beta} = -\sin\gamma$$

$$k_2 - \cos\gamma = \tan\beta \cdot -\sin\gamma$$

$$\tan\beta = \frac{\cos\gamma - k_2}{\sin\gamma}$$

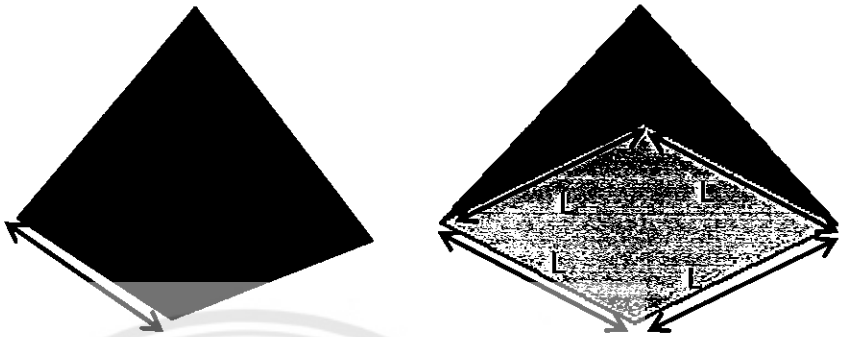
$$\beta = \tan^{-1}\left(\frac{\cos\gamma - k_2}{\sin\gamma}\right)$$

และ

$$L = \frac{L_1}{\sqrt[2]{1 - \sin^2\alpha \cos^2\beta}}$$

- **รูปทรงพีระมิด**

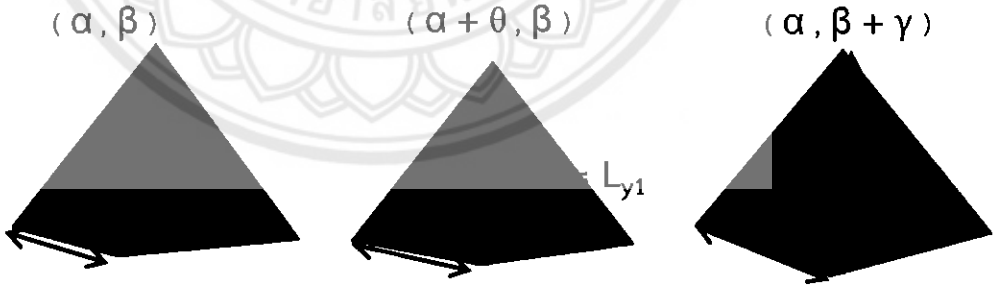
วิเคราะห์หาสูตรที่ใช้ในการหาความยาวด้านของรูปใน 2 มิติ



รูปที่ 3.22 แสดงลักษณะฐานของรูปทรงพีระมิด

จากรูปที่ 3.12 จะเห็นได้ว่า ฐานของรูปทรงพีระมิดมีลักษณะเป็นรูปสี่เหลี่ยมจัตุรัส เพราะฉะนั้นในการหาสูตรที่ใช้กับรูปในแนวระนาบ จึงสามารถใช้สูตรเดียวกันกับสูตรในแนวระนาบของรูปทรงลูกบาศก์ได้ นั่นก็คือ $\alpha = \tan^{-1}(\frac{\sin\theta}{k_1-\cos\alpha})$

และ $L = \frac{\sqrt[2]{1+\sin^2\alpha\cos^2\beta}}{L_1}$



รูปที่ 3.23 ภาพแสดงความยาวด้านและมุมของภาพที่ใช้ในการคำนวณ

เนื่องจากรูปทรงพีระมิดมีจุดยอดอยู่ที่ตำแหน่ง $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{\sqrt{2}}\right)$ เพราะฉะนั้นเมื่อแทนค่าตำแหน่งของจุดยอดลงไปในเมทริกซ์ ต่อไปนี้

$$\begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix} = \begin{bmatrix} X\cos\alpha\cos\beta + Y\sin\alpha\cos\beta + Z\sin\beta \\ -X\sin\alpha + Y\cos\alpha \\ -X\cos\alpha\cos\beta - Y\sin\alpha\cos\beta + Z\cos\beta \end{bmatrix}$$

จะได้ ความยาวด้าน (L_{yl}) ดังนี้

$$L_1 = \frac{L}{2}\sqrt[2]{(\sin\alpha - \cos\beta)^2 + (\cos\alpha\sin\beta + \sin\alpha\sin\beta + \sqrt{2}\cos\beta)^2}$$

จากการหาค่ามุมในแนวระนาบ จะทำให้เราทราบค่ามุมแอลฟ่า (α) ซึ่งจะทำให้สามารถหาค่าความยาว L ได้ดังนี้ (อ้างอิงสมการที่ 1 และ 2 ของรูปทรงลูกบาศก์)

$$L_1 = L\sqrt[2]{1 - \sin^2\alpha\cos^2\beta} \quad \text{---------- สมการที่ 1}$$

$$L_2 = L\sqrt[2]{1 - \sin^2(\alpha+\theta)\cos^2\beta} \quad \text{---------- สมการที่ 2}$$

สมการที่ 1 ยกกำลัง 2 ; $L_1^2 = L^2(1 - \sin^2\alpha\cos^2\beta)$

สมการที่ 2 ยกกำลัง 2 ; $L_2^2 = L^2(1 - \sin^2(\alpha+\theta)\cos^2\beta)$

กำหนดให้ $l_1 = \frac{L_1}{L}$ และ $l_2 = \frac{L_2}{L}$

$$l_1^2 = 1 - \sin^2\alpha\cos^2\beta$$

$$l_1^2 - 1 = -\sin^2\alpha\cos^2\beta \quad \text{---------- สมการที่ 3}$$

$$l_2^2 = 1 - \sin^2(\alpha+\theta)\cos^2\beta$$

$$l_2^2 - 1 = -\sin^2(\alpha+\theta)\cos^2\beta \quad \text{---------- สมการที่ 4}$$

สมการที่ 4 หารด้วยสมการที่ 3

$$\frac{1 - l_2^2}{1 - l_1^2} = \frac{\sin^2(\alpha+\theta)}{\sin^2\alpha}$$

$$\frac{1 - l_1^2}{\sin^2\alpha} = \frac{1 - l_2^2}{\sin^2(\alpha+\theta)}$$

แทนค่า l_1 และ l_2

$$\frac{1-\left(\frac{L_1}{L}\right)^2}{\sin^2\alpha} = \frac{1-\left(\frac{L_2}{L}\right)^2}{\sin^2(\alpha+\theta)}$$

$$\frac{L^2-{L_1}^2}{\sin^2\alpha} = \frac{L^2-{L_1}^2}{\sin^2(\alpha+\theta)}$$

$$\left(\frac{1}{\sin^2\alpha}-\frac{1}{\sin^2(\alpha+\theta)}\right)L^2 = \frac{{L_1}^2}{\sin^2\alpha}-\frac{{L_1}^2}{\sin^2(\alpha+\theta)}$$

เพราะฉะนั้นค่าความยาวด้าน จะเท่ากับ ;

$$L = \sqrt[2]{\frac{\left(\frac{{L_1}^2}{\sin^2\alpha}-\frac{{L_1}^2}{\sin^2(\alpha+\theta)}\right)}{\left(\frac{1}{\sin^2\alpha}-\frac{1}{\sin^2(\alpha+\theta)}\right)}}$$

เมื่อทราบทั้งค่าความยาวด้านและค่ามุมแอลฟ่า ก็จะสามารถหาค่ามุมเบต้าได้ ดังนี้

$$L_1 = \frac{L}{2}\sqrt[2]{(\sin\alpha-\cos\beta)^2+(\cos\alpha\sin\beta+\sin\alpha\sin\beta+\sqrt{2}\cos\beta)^2}$$

$$\sqrt[2]{\left(\frac{L_1}{L/2}\right)^2-(\sin\alpha-\cos\beta)^2} = \cos\alpha\sin\beta+\sin\alpha\sin\beta+\sqrt{2}\cos\beta$$

กำหนดให้ $k_3 = \sqrt[2]{\left(\frac{L_1}{L/2}\right)^2-(\sin\alpha-\cos\beta)^2}$;

$$k_3 = \cos\alpha\sin\beta+\sin\alpha\sin\beta+\sqrt{2}\cos\beta$$

นำ $\cos\beta$ หารตลอด ;

$$\sec\beta\cdot k_3 = \cos\alpha\tan\beta+\sin\alpha\tan\beta+\sqrt{2}$$

$$\sec^2\beta \cdot k_3{}^2 = ((\cos\alpha + \sin\alpha)\tan\beta + \sqrt{2})^2$$

$$(1 + \tan^2\beta) \cdot k_3{}^2 = (\cos\alpha + \sin\alpha)^2 \tan^2\beta +$$

$$2\sqrt{2}\ (\cos\alpha + \sin\alpha)\ \tan\beta + 2$$

$$0 = ((\cos\alpha + \sin\alpha)^2 - k_3{}^2)\tan^2\beta$$
$$+ 2\sqrt{2}\ (\cos\alpha + \sin\alpha)\tan\beta + (2 - k_3{}^2)$$

กำหนดให้ $AA = ((\cos\alpha + \sin\alpha)^2 - k_3{}^2$

$$BB = 2\sqrt{2}(\cos\alpha + \sin\alpha)$$

$$CC = 2 - k_3{}^2$$

จะได้ สมการใหม่ดังนี้ ;

$$0 = AA\tan^2\beta + BB\tan\beta + CC$$

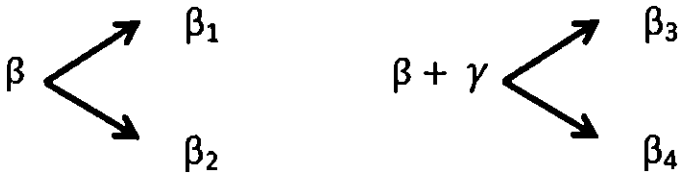
ซึ่งสามารถหาค่ามุมเบต้าได้ คือ

$$\beta = \tan^{-1}\left(\frac{-BB \pm \sqrt[2]{BB^2 - 4 \cdot AA \cdot CC}}{2}\right)$$

เมื่อสามารถค่ามุมเบต้าได้ ก็จะสามารถหาค่ามุมเบต้าบวกแกมม่า $(\beta + \gamma)$ ได้เช่นกัน โดยทำในลักษณะเดียวกันจะได้สมการของค่ามุมเบต้าบวกแกมม่า ดังนี้

$$\beta + \gamma = \tan^{-1}\left(\frac{-BB \pm \sqrt[2]{BB^2 - 4 \cdot AA \cdot CC}}{2}\right)$$

สุดท้ายแล้วผลลัพธ์ที่ได้ของค่ามุมในแนวดิ่ง จะมีด้วยกันทั้งหมด 4 ค่า ซึ่งจะต้องนำมาทำการจับคู่กัน ดังนี้



รูปที่ 3.24 ผลลัพธ์ทั้งหมดของค่ามุมเบต้า

จากรูปที่ 3.24 ให้นำค่ามุมทางฝั่งของมุมเบต้าบวกแกมม่าลบด้วยค่ามุมแกมม่า จะเหลือแต่ค่ามุมเบต้า จากนั้นให้ทำการจับคู่และลบกัน ดังนี้

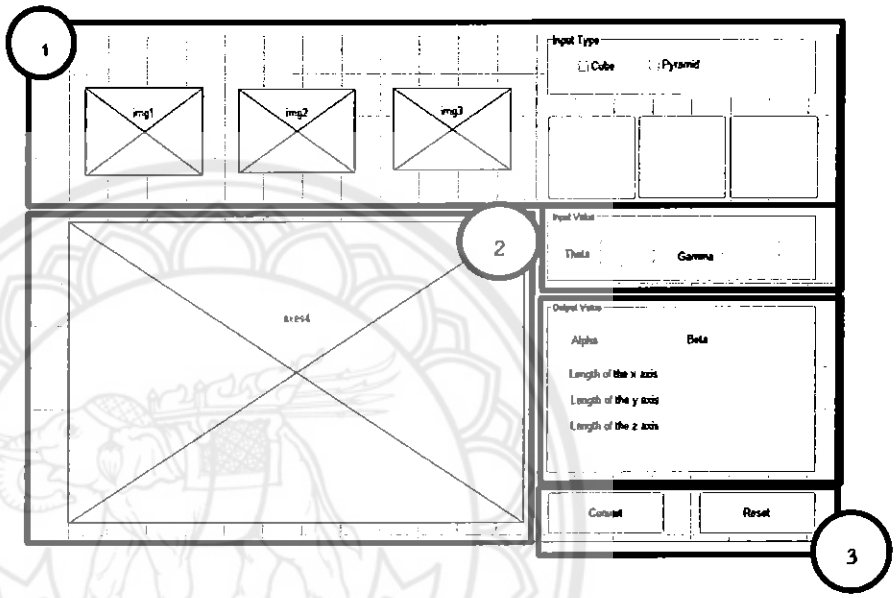
$$\beta_1 - \beta_3 \ , \ \beta_1 - \beta_4 \ , \ \beta_2 - \beta_3 \text{ และ } \beta_2 - \beta_4$$

หลังจากนั้น ให้ดูค่าที่ใกล้เคียงค่า 0 มากที่สุด ก็จะได้ชุดข้อมูลค่ามุมในแนวดิ่งคู่ที่ถูกต้อง

3.3.4 การสร้างตัวโปรแกรมสร้างภาพ 3 มิติกลับคืนจากภาพ 2 มิติ

3.3.4.1 โปรแกรมสร้าง GUI เพื่อแสดงผลลัพธ์ของงาน

- รูปแบบการสร้างหน้า GUI โดยจะแบ่งเป็นสองฝั่ง คือ ฝั่งล่างเป็นเอาต์พุตในการแสดงผล ฝั่งบนคือ อินพุทนำเข้า และแสดงผลภาพอินพุท ดังนี้



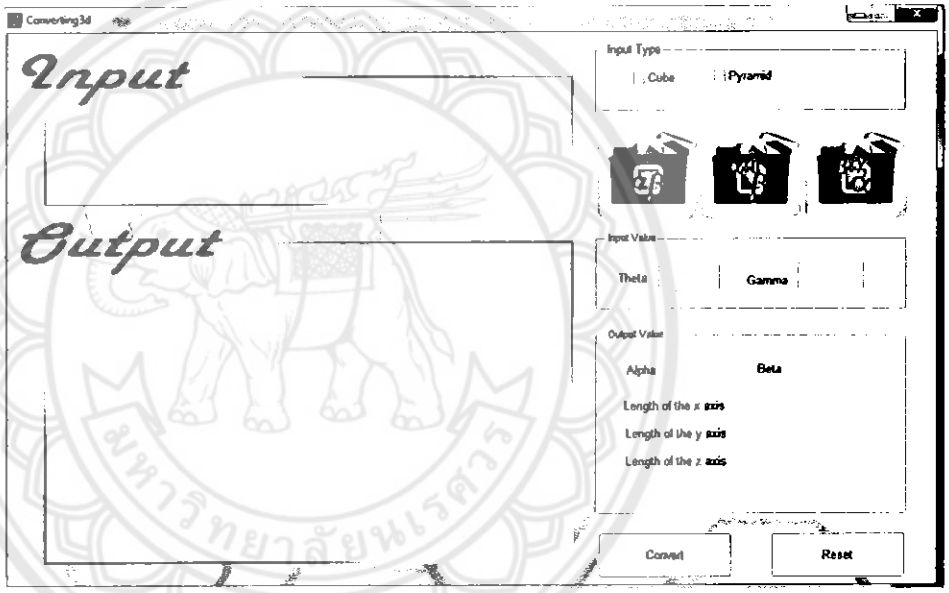
รูปที่ 3.25 สัดส่วนการแสดงผล GUI

จากรูปที่ 3.25 สามารถอธิบายได้สัดส่วนของโปรแกรมไว้ 3 ส่วน ซึ่งแต่ละส่วนทำหน้าที่ต่างกันได้ดังนี้

- ส่วนที่ 1 (กรอบสีแดง) จะมีปุ่ม (Button) 3 ปุ่ม ไว้สำหรับทำการเพิ่มรูปอินพุท (โดยปุ่มหนึ่งปุ่มจะเพิ่มรูปได้หนึ่งรูป) มีพื้นที่ว่าง (axes) อยู่ 3 ส่วน โดยใช้ชื่อว่า img1 , img2 และ img3 ไว้แสดงผลรูปลงในนั้น โดยการเพิ่มรูปแรกจะอยู่ในพื้นที่ว่าง (axes) ที่มีชื่อว่า img1 จากนั้นจะเป็น รูปสองและสาม ตามลำดับ สุดท้ายในส่วนกลุ่มฟังก์ชัน (panel) ที่มีชื่อว่า ชนิดของอินพุต (Input Type) มีฟังก์ชัน check block สองช่อง คือ ทรงลูกบาศก์ (cube) และทรงพีระมิด (pyramid) เพื่อให้ผู้ใช้เลือกว่าอินพุทที่เข้ามาประมวลผลในฟังก์ชั่นไหน

- ส่วนที่ 2 (กรอบสีน้ำเงิน) จะมีพื้นที่ว่างหนึ่ง ชื่อว่า axes4 มีไว้สำหรับแสดงผลลัพธ์ที่เป็นรูปสามมิติโดยสามารถหมุน (Rotate) ได้ และมีส่วนของกลุ่มฟังก์ชัน (panel) ที่มีชื่อว่า ค่าของผลลัพธ์ (Output Value) ในส่วนนี้จะแสดงค่ามุมแอลฟ่า และ มุมเบต้า และความยาว L ในแต่ละแกนที่ประมวลผลได้มาแสดง

- ส่วนที่ 3 (กรอบสีเขียว) จะมีปุ่มอยู่สองปุ่ม ชื่อว่า แปลงภาพ (convert) และเริ่มต้นใหม่ (reset) โดยปุ่มแปลงภาพ จะเป็นการประมวลผล อินพุทที่ได้มาลงในส่วนที่ 2 ส่วนปุ่มเริ่มต้นใหม่ มีไว้ เริ่มต้นโปรแกรมใหม่



รูปที่ 3.26 แสดงหน้า GUI ที่ทำการรันแล้ว

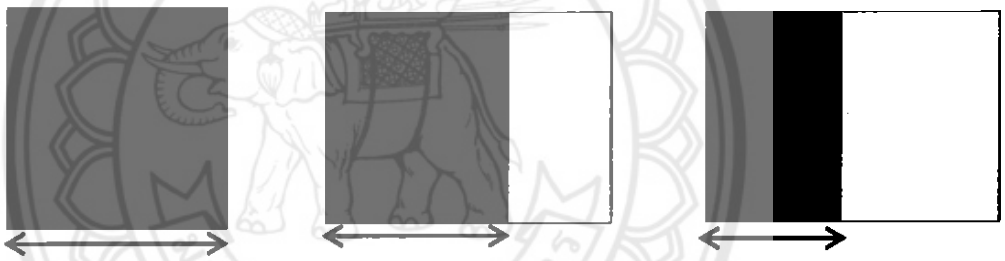
บทที่ 4

ผลการดำเนินการ

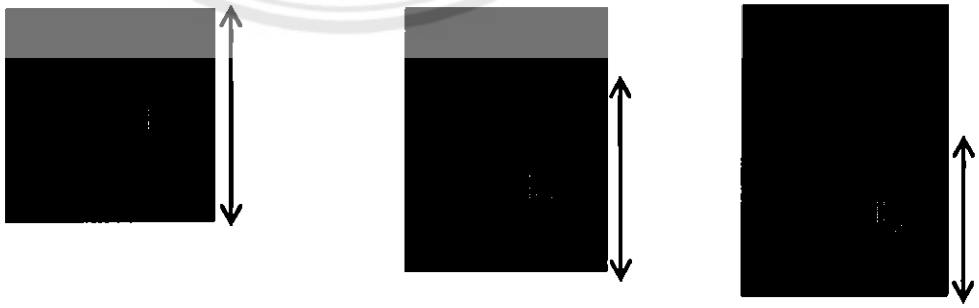
เนื้อหาในบทนี้ได้อธิบายถึงผลการทดลองตั้งแต่ขั้นตอนการทดลองหาความยาวใน 1 มิติ ด้วยภาพ 2 มิติ (Test case) และความยาวของรูปทรงมุมใดๆ จากการบันทึกความยาวและคำนวณด้วยโปรแกรม ท้ายสุดคือผลการทดลองกับชุดทดลอง 2 ชุด ซึ่งมีรายละเอียดดังนี้

4.1 ความยาวด้านใน 1 มิติ ด้วยภาพใน 2 มิติ จากการบันทึกความยาวและคำนวณด้วยโปรแกรม

4.1.1 ทรงลูกบาศก์



(ก)



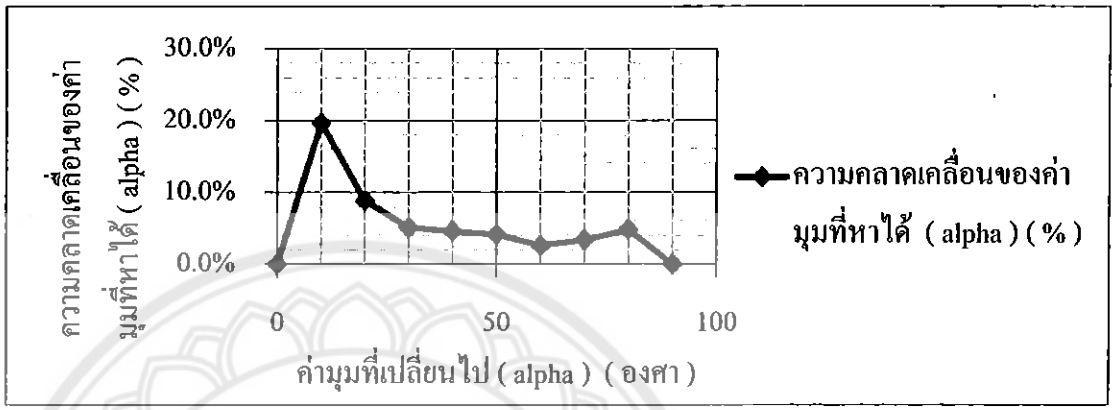
(ข)

รูปที่ 4.1 (ก) ความยาวด้านที่ใช้ในการคำนวณในแนวระนาบ

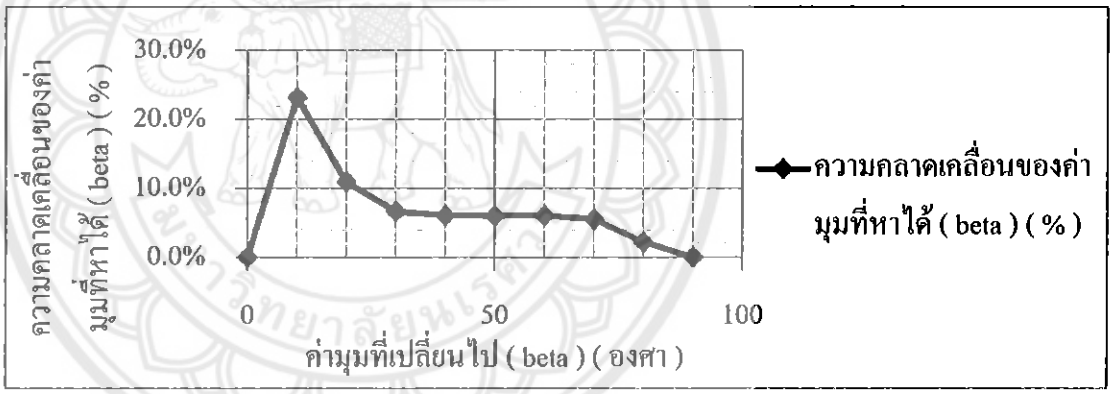
(ข) ความยาวด้านที่ใช้ในการคำนวณในแนวดิ่ง

- กราฟแสดงความคลาดเคลื่อนของค่ามุมเริ่มต้นแอลฟ่า (α) และ มุมเบต้า (β) ที่วัดได้

เมื่อทำการรันหลายๆครั้ง โดยเปลี่ยนค่ามุมไปเรื่อยๆ สามารถนำมาเขียนเป็นกราฟได้โดยประมาณดังแสดงในรูปที่ 4.2



(ก)

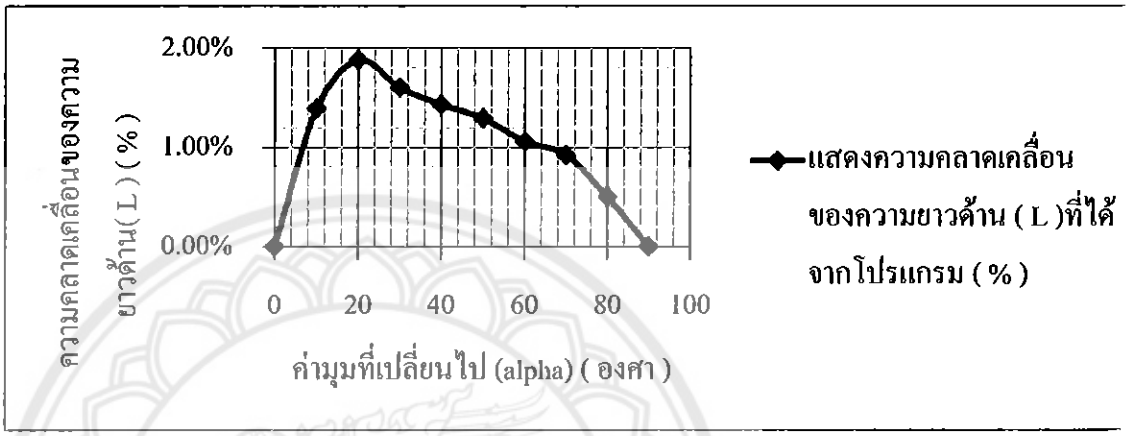


(ข)

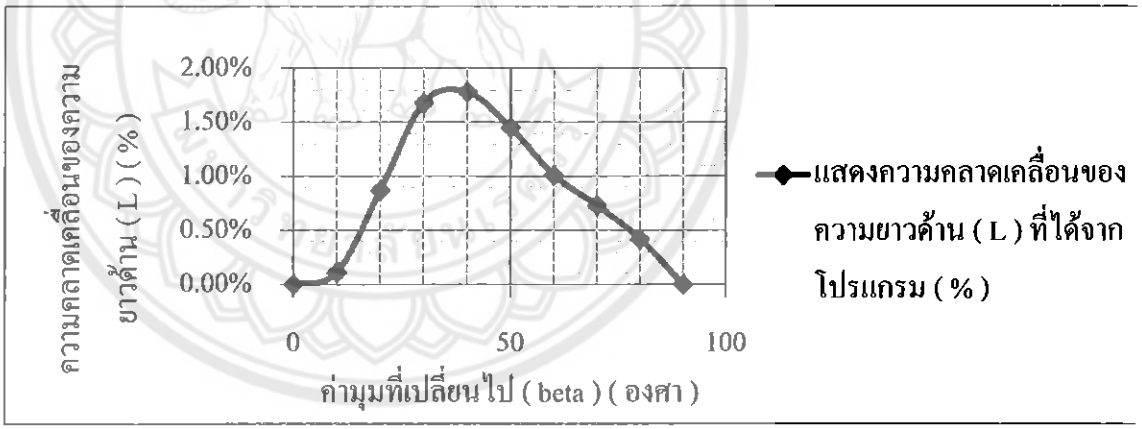
รูปที่ 4.2 (ก) กราฟแสดงความคลาดเคลื่อนของรูปที่ปรับค่ามุมในแนวระนาบ
(ข) กราฟแสดงความคลาดเคลื่อนของรูปที่ปรับค่ามุมในแนวดิ่ง

- กราฟแสดงความคลาดเคลื่อนของค่าความยาว (L) ที่วัดได้ เมื่อเทียบกับความยาว (L) จริง

เมื่อทำการรันหลายๆครั้ง โดยเปลี่ยนค่ามุมไปเรื่อยๆ สามารถนำมาเขียนเป็นกราฟได้โดยประมาณดังแสดงในรูปที่ 4.3



(ก)

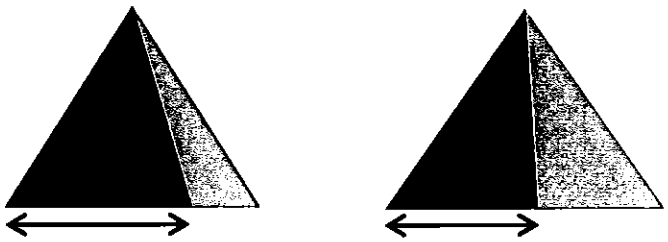


(ข)

รูปที่ 4.3 (ก) กราฟแสดงความคลาดเคลื่อนค่าความยาว (L) ที่วัดได้ เมื่อเทียบกับความยาว (L) จริง ของรูปที่ปรับค่ามุมในแนวระนาบ

(ข) กราฟแสดงความคลาดเคลื่อนความยาว (L) ที่วัดได้เมื่อเทียบกับความยาว (L) จริง ของรูปที่ปรับค่ามุมในแนวดิ่ง

4.1.2 ทรงพีระมิด



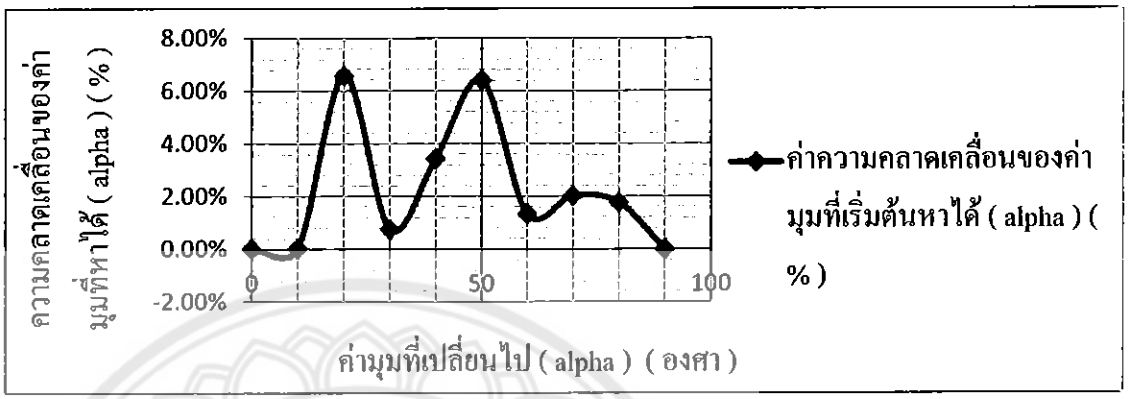
(ก)

(ข)

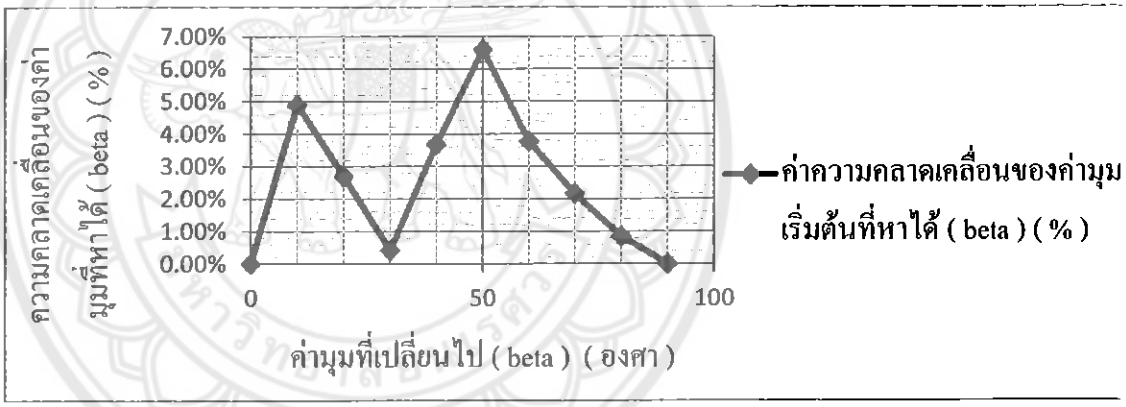
รูปที่ 4.4 (ก) ความยาวด้านที่ใช้ในการคำนวณในแนวระนาบ
(ข) ความยาวด้านที่ใช้ในการคำนวณในแนวดิ่ง

- กราฟแสดงความคลาดเคลื่อนของค่ามุมเริ่มต้นแอลฟ่า (α) และ มุมเบต้า (β) ที่วัดได้

เมื่อทำการรันหลายๆครั้ง โดยเปลี่ยนค่ามุมไปเรื่อยๆ สามารถนำมาเขียนเป็นกราฟได้โดยประมาณดังแสดงในรูปที่ 4.5



(ก)

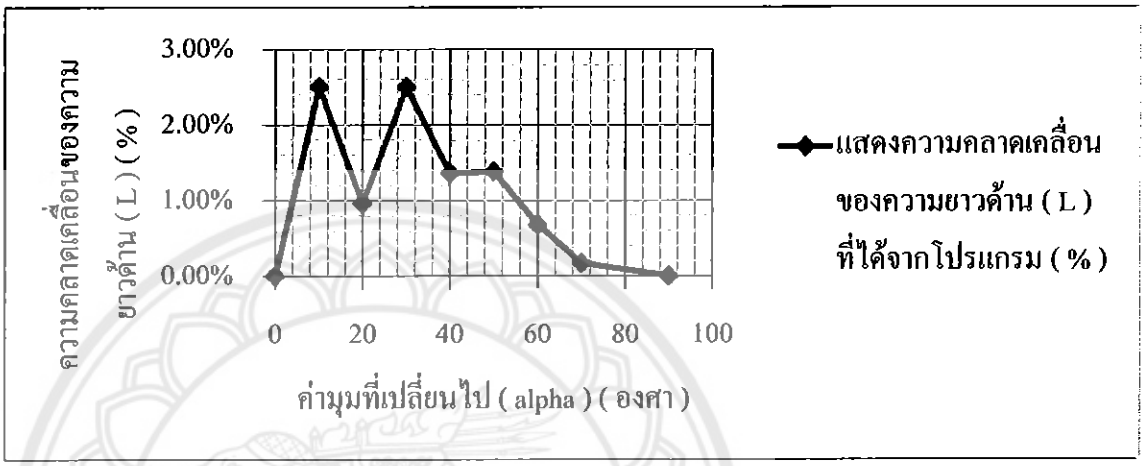


(ข)

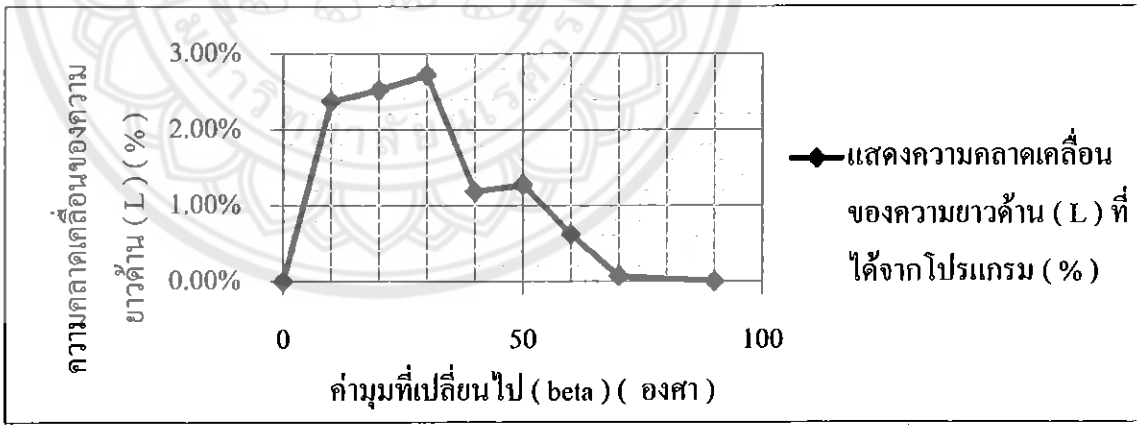
รูปที่ 4.5 (ก) กราฟแสดงความคลาดเคลื่อนค่ามุมเริ่มต้นของรูปที่ปรับค่ามุมในแนวระนาบ
(ข) กราฟแสดงความคลาดเคลื่อนค่ามุมเริ่มต้นของรูปที่ปรับค่ามุมในแนวดิ่ง

- กราฟแสดงความคลาดเคลื่อนของค่าความยาว (L) ที่วัดได้ เมื่อเทียบกับความยาว (L) จริง

เมื่อทำการรันหลายๆครั้ง โดยเปลี่ยนค่ามุมไปเรื่อยๆ สามารถนำมาเขียนเป็นกราฟได้โดยประมาณดังแสดงในรูปที่ 4.6



(ก)



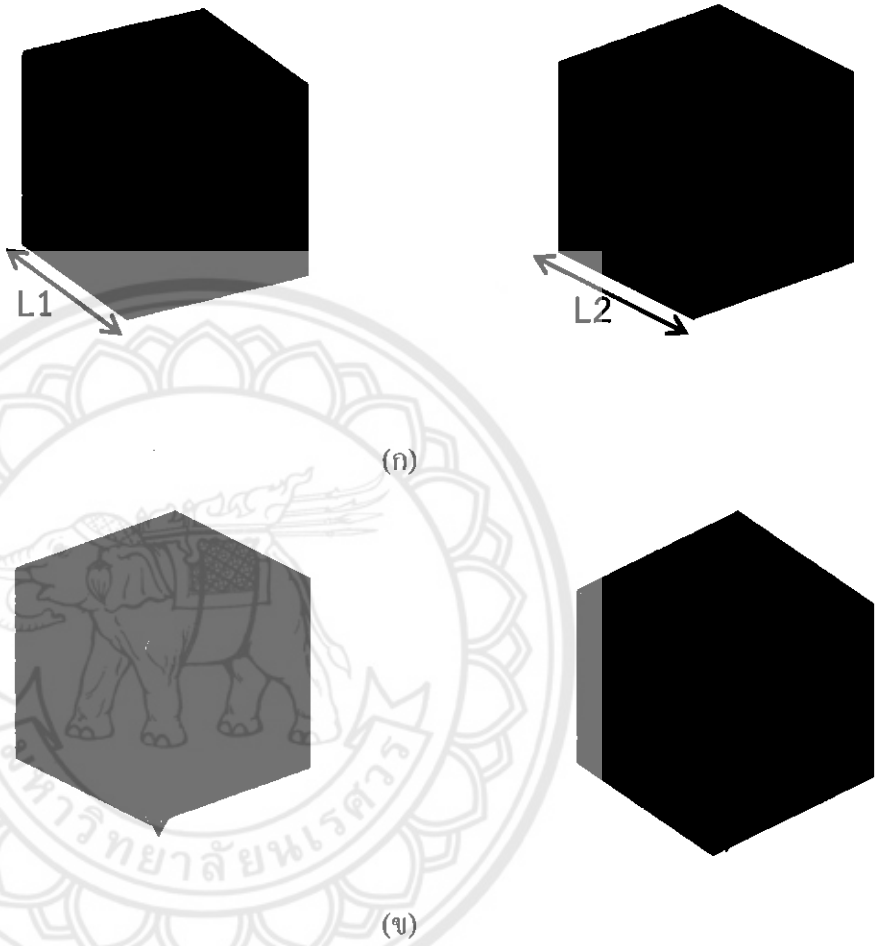
(ข)

รูปที่ 4.6 (ก) กราฟแสดงความคลาดเคลื่อนของค่าความยาว (L) ที่วัดได้ เมื่อเทียบกับความยาว (L) จริง ของรูปที่ปรับค่ามุมในแนวระนาบ

(ข) กราฟแสดงความคลาดเคลื่อนของค่าความยาว (L) ที่วัดได้เมื่อเทียบกับความยาว (L) จริงของรูปที่ปรับค่ามุมในแนวดิ่ง

4.2 ความยาวด้านของรูปทรงมุมใดๆ จากการบันทึกความยาวและคำนวณด้วยโปรแกรม

4.2.1 ทรงลูกบาศก์มุมใดๆ

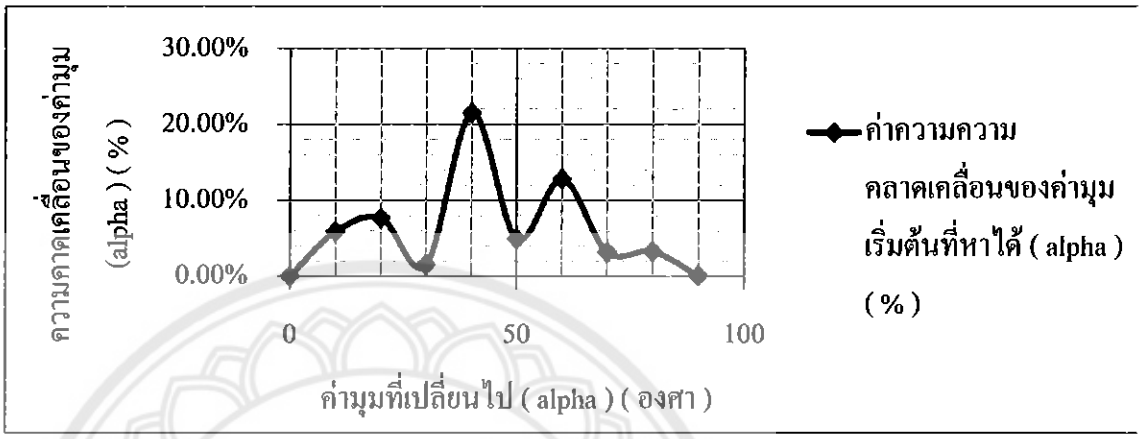


รูปที่ 4.7 (ก) ความยาวด้านที่ใช้ในการคำนวณในแนวระนาบ

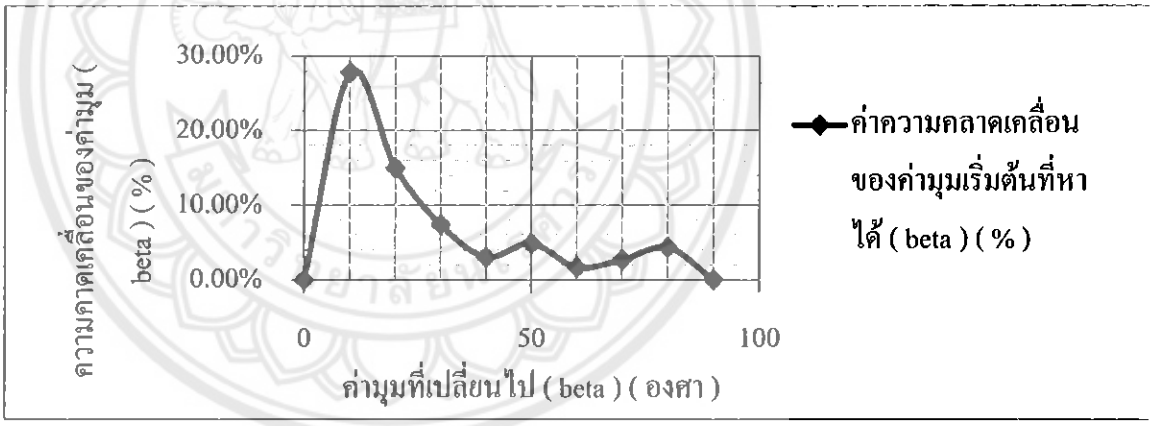
(ข) ความยาวด้านที่ใช้ในการคำนวณในแนวดิ่ง

- กราฟแสดงความคลาดเคลื่อนของค่ามุมเริ่มต้นแอลฟ่า (α) และ มุมเบต้า (β) ที่วัดได้

เมื่อทำการรันหลายๆครั้ง โดยเปลี่ยนค่ามุมไปเรื่อยๆ สามารถนำมาเขียนเป็นกราฟได้โดยประมาณดังแสดงในรูปที่ 4.8



(ก)

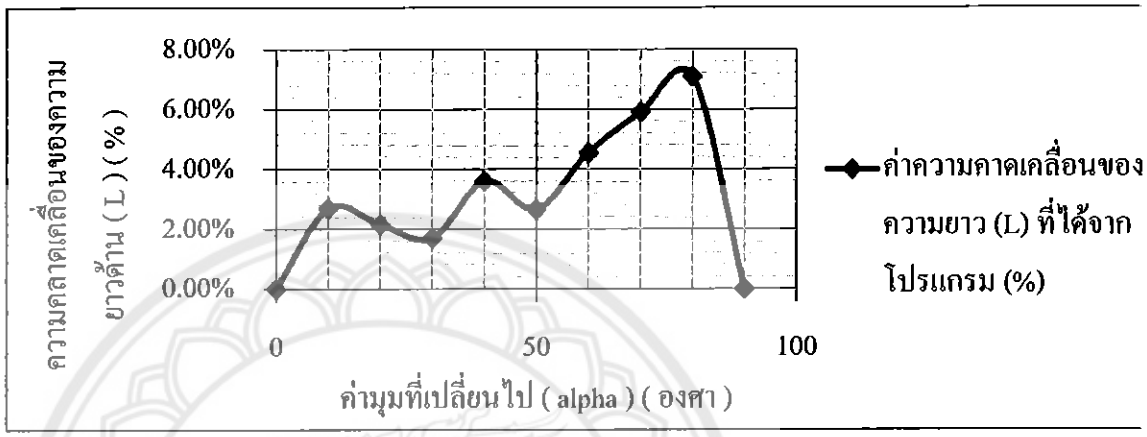


(ข)

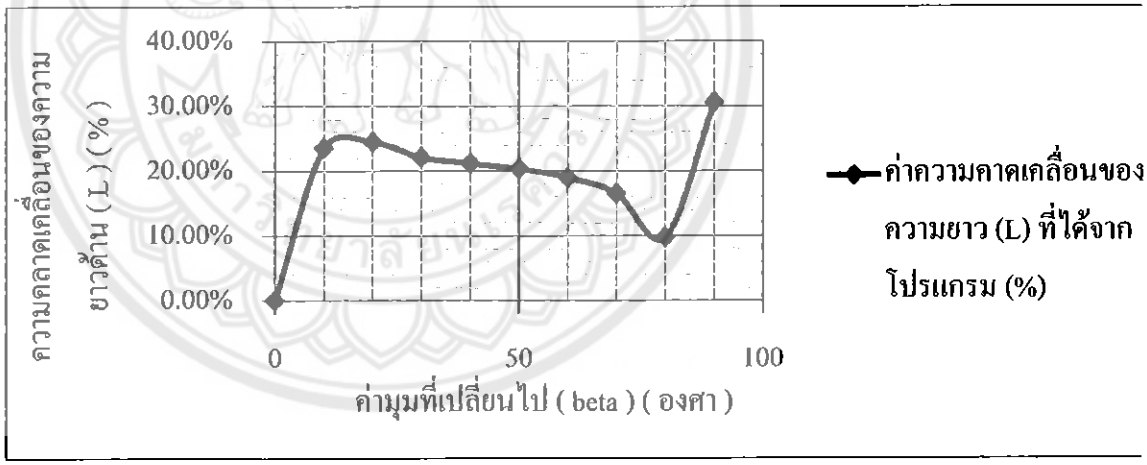
รูปที่ 4.8 (ก) กราฟแสดงความคลาดเคลื่อนค่ามุมเริ่มต้นของรูปที่ปรับค่ามุมในแนวระนาบ (ข) กราฟแสดงความคลาดเคลื่อนค่ามุมเริ่มต้นของรูปที่ปรับค่ามุมในแนวดิ่ง

- กราฟแสดงความคลาดเคลื่อนของค่าความยาว (L) ที่วัดได้ เมื่อเทียบกับความยาว (L) จริง

เมื่อทำการรันหลายๆครั้ง โดยเปลี่ยนค่ามุมไปเรื่อยๆ สามารถนำมาเขียนเป็นกราฟได้โดยประมาณดังแสดงในรูปที่ 4.9



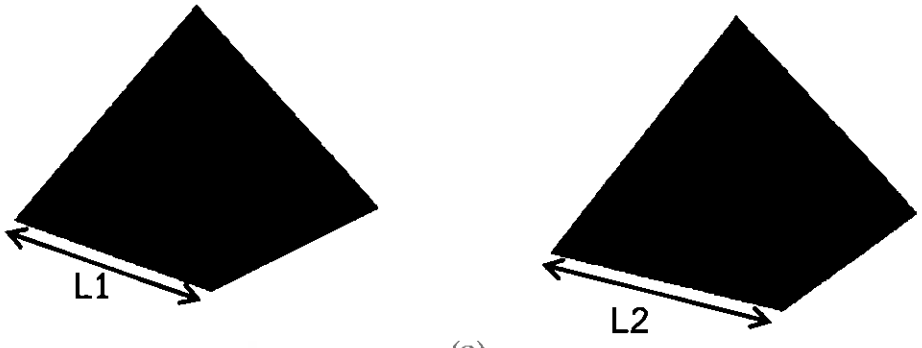
(ก)



(ข)

รูปที่ 4.9 (ก) กราฟแสดงความคลาดเคลื่อนของค่าความยาว (L) ที่วัดได้เทียบกับความยาว (L) จริง ของรูปที่ปรับค่ามุมในแนวระนาบ
(ข) กราฟแสดงความคลาดเคลื่อนของค่าความยาว (L) ที่วัดได้ เมื่อ เทียบกับความยาว (L) จริง ของรูปที่ปรับค่ามุมในแนวดิ่ง

4.2.2 ทรงพิระมิดมุมใดๆ



(ก)

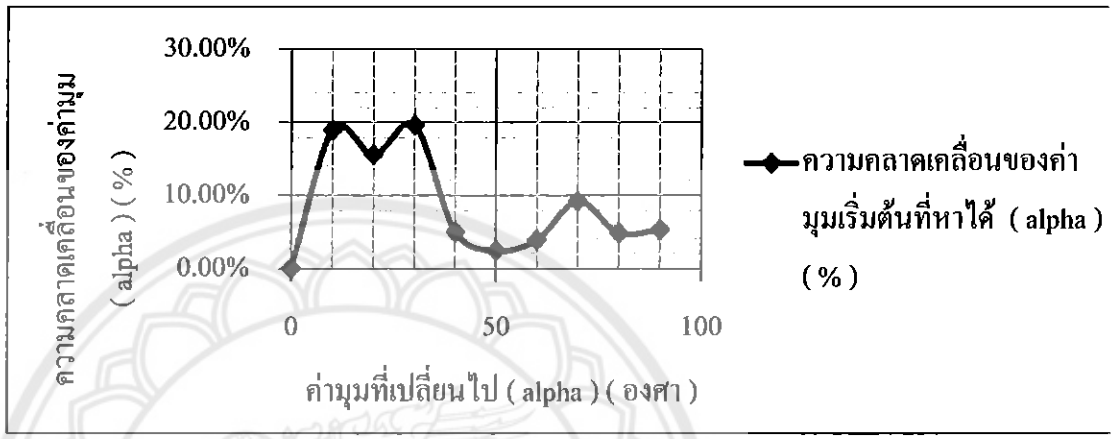
(ข)

รูปที่ 4.10 (ก) ความยาวด้านที่ใช้ในการคำนวณในแนว**ระนาบ**

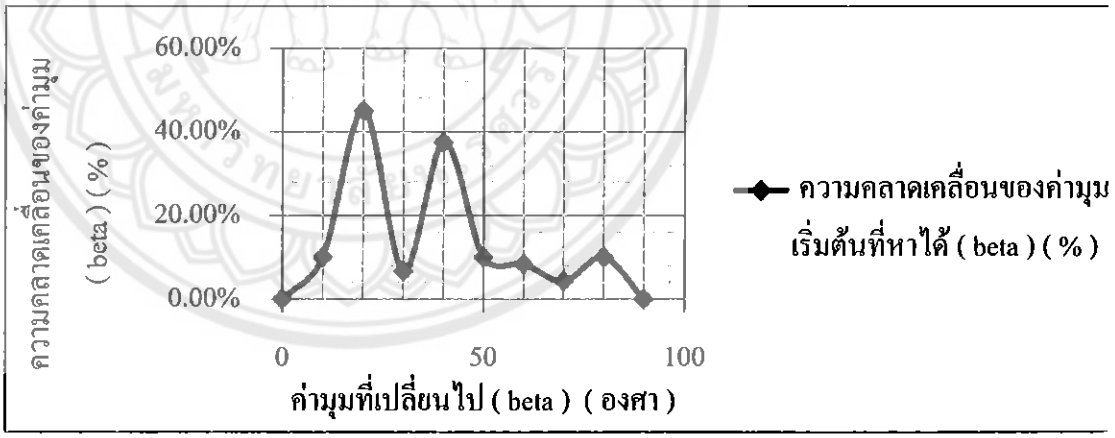
(ข) ความยาวด้านที่ใช้ในการคำนวณในแนว**ดิ่ง**

- กราฟแสดงความคลาดเคลื่อนของค่ามุมเริ่มต้นแอลฟ่า (α) และ มุมเบต้า (β) ที่วัดได้

เมื่อทำการรันหลายๆครั้ง โดยเปลี่ยนค่ามุมไปเรื่อยๆ สามารถนำมาเขียนเป็นกราฟได้โดยประมาณดังแสดงในรูปที่ 4.11 (ค่ามุมเบต้า (β) และ มุมแอลฟ่า (α) คงที่ = 40 องศา)



(ก)



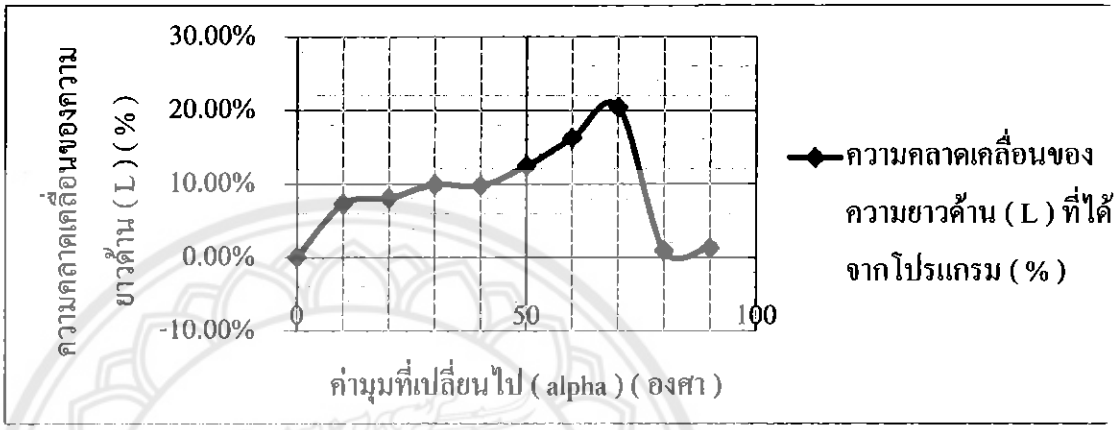
(ข)

รูปที่ 4.11 (ก) กราฟแสดงความคลาดเคลื่อนค่ามุมเริ่มต้นของรูปที่ปรับค่ามุมในแนวระนาบ

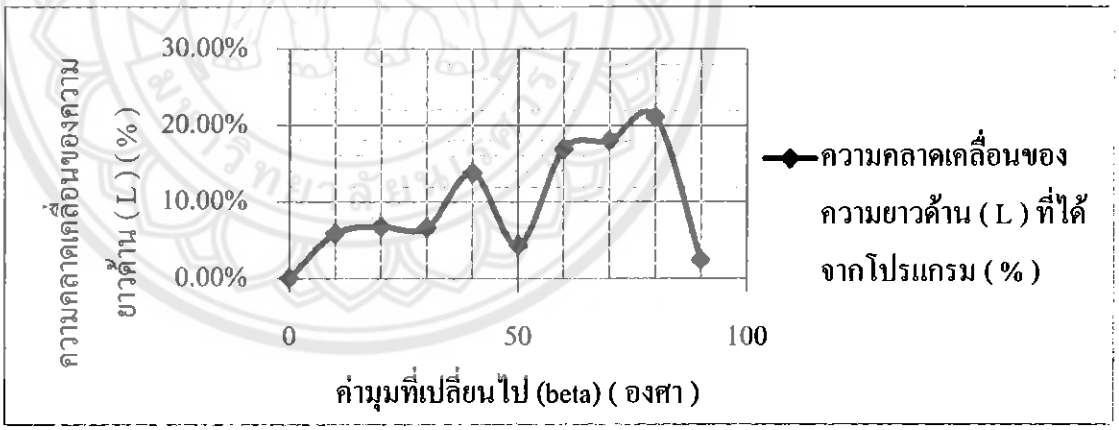
(ข) กราฟแสดงความคลาดเคลื่อนค่ามุมเริ่มต้นของรูปที่ปรับค่ามุมในแนวดิ่ง

- กราฟแสดงความคลาดเคลื่อนของค่าความยาว (L) ที่วัดได้ เมื่อเทียบกับความยาว (L) จริง

เมื่อทำการรันหลายๆครั้ง โดยเปลี่ยนค่ามุมไปเรื่อยๆ สามารถนำมาเขียนเป็นกราฟได้โดยประมาณดังดังแสดงในรูปที่ 4.12



(ก)



(ข)

รูปที่ 4.12 (ก) กราฟแสดงความคลาดเคลื่อนของค่าความยาว (L) ที่วัดได้ เทียบกับความยาว (L) จริง ของรูปที่ปรับค่ามุมในแนวระนาบ
(ข) กราฟแสดงความคลาดเคลื่อนของค่าความยาว (L) ที่วัดได้ เมื่อเทียบกับความยาว (L) จริง ของรูปที่ปรับค่ามุมในแนวดิ่ง

จากกราฟทั้งหมด จะเห็นได้ว่า ข้อมูลนั้นมีความคลาดเคลื่อนที่แตกต่างกันโดยทั้งนี้เป็นผลมาจากรูปอินพุตที่ใช้ในการประมวลผล ซึ่งจะสังเกตได้จาก ในส่วนของการหาความยาวด้านใน 1 มิติ ด้วยภาพใน 2 มิติ จากการบันทึกความยาวและคำนวณด้วยโปรแกรม (Test case) นั้น รูปอินพุตจะค่อนข้างสามารถหามุมของรูปได้อย่างชัดเจน จึงทำให้มีความคลาดเคลื่อนเพียงเล็กน้อย ซึ่งอยู่ในระดับประมาณ ไม่เกิน 4 % แต่หากเป็นในส่วนของการหาความยาวด้านของรูปทรงมุมใดๆ จากการบันทึกความยาวและคำนวณด้วยโปรแกรม มุมของรูปอินพุตที่ใช้นั้นมีการกระจายตัวมากกว่าในกรณีแรก และมีความคลาดเคลื่อนของมุมเริ่มต้นที่คำนวณได้บางค่าสูงถึงเกือบ 50 % เป็นเพราะรูปอินพุตบางรูปยังมีรอยต่อที่ไม่มีความกลมกลืน (ยังมีรอยแตกของรูป) ทำให้ส่งผลไปถึงการหาจุดที่ไม่ตรงกับความเป็นจริงซึ่งเป็นผลอย่างมากกับความยาวที่คำนวณได้ และอีกสาเหตุหนึ่งที่มุมเริ่มต้นที่ได้บางมุมมีค่าความคลาดเคลื่อนไปมากเนื่องจากความยาวที่ทำการวัดที่นำมาคิดใช้เพียงด้านเดียวในแต่ละรูปโดยที่ไม่ใช้ความยาวที่เหลืออีกด้านมาคิด ทำให้สามารถทราบได้ว่าเมื่อรูปหมุนไปแล้วความยาวอีกด้านจะเป็นเท่าไหร่ซึ่งเราสามารถใช้ความยาวอีกด้านมาคิดหามุมเริ่มต้นได้เช่นกันซึ่งมีความซับซ้อนมาก แต่มุมเริ่มต้นส่วนใหญ่ไม่มีผลต่อความยาวที่ได้จากโปรแกรมคำนวณเมื่อเทียบกับความยาวจริงมากนักดังนั้นจะเห็นได้ว่าค่าความคลาดเคลื่อนของความยาวที่คำนวณจากโปรแกรมเมื่อเทียบกับความยาวจริงมีประมาณ 30 % ซึ่งทำให้ผลจากการทดลองรูปมุมอื่นๆที่ไม่อยู่ในชุดทดสอบอยู่ในเกณฑ์ที่พอใช้ได้เนื่องจากส่วนสำคัญในการสร้างรูปสามมิติคือ ความยาว เมื่อหมุนแล้วความยาวในแนวระนาบจะต้องมีความใกล้เคียงกับความยาวจริง

4.3 ผลการทดลองกับชุดทดสอบ

4.3.1 ทรงลูกบาศก์มุมใดๆ

- รูปอินพุต

รูป	มุมเริ่มต้น (alpha) (องศา)	มุมเริ่มต้น (beta) (องศา)
1	40	20
2	30	20
3	40	30

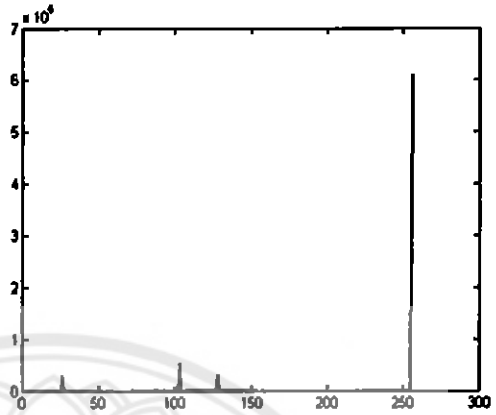
* มุมเซตา (θ) = 10 , มุมแกมม่า (γ) =10

ตารางที่ 4.1 ตารางแสดงข้อมูลรูปอินพุตทรงลูกบาศก์ทั้ง 3 รูป

1) จากตารางรูปอินพุต รูปที่ 1 แสดงดังรูป 4.13

รูปที่ 4.13 รูปอินพุตที่ 1 จากตารางรูปอินพุต

- จากรูปอินพุตที่ 1 ผ่านกระบวนการพลอตกราฟ ฮีสโตแกรม (Histogram) แล้วจะได้กราฟดังแสดงในรูปที่ 4.14



รูปที่ 4.14 กราฟฮีสโตแกรม (Histogram) จากรูปอินพุตที่ 1

- ผลการดึงค่าระดับสีเทา (gray level) จากกราฟ ฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

256 103 128 26

- ผลจากการเรียงลำดับค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

26 103 128 256

- ผลจากการจัดเก็บค่าลงในอาเรย์ใหม่ที่เริ่มต้นตำแหน่งแรกด้วย 0 ซึ่งจะได้ค่าดังนี้

0 26 103 128 256

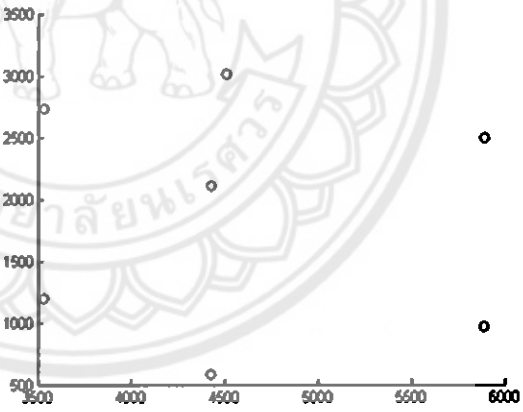
- ผลการตรวจสอบพื้นสีขาวที่มีค่าความสว่างเท่ากับ 256 ซึ่งจะได้ค่าดังนี้

0 26 103 128

- ผลจากการนำค่าระดับสีเทา (gray level) ผ่านกระบวนการแบ่งภาพตามค่าระดับสีเทา (gray level) ซึ่งจะได้ผลลัพธ์ดังแสดงในรูปที่ 4.15

รูปที่ 4.15 ผลการแบ่งภาพตามค่าระดับสีเทา (gray level)

- ผลจากการใช้ ฟังก์ชัน corner ตามภาพที่แบ่งและผ่านกระบวนการจัดการจุดใหม่ ซึ่งจะได้กราฟดังแสดงในรูปที่ 4.16



รูปที่ 4.16 กราฟจุดใหม่ที่ได้จากกระบวนการจัดการจุด

- จากรูปที่ 4.16 ได้ข้อมูลของจุดมาดังแสดงในตารางที่ 4.2

ชื่อจุด	ตำแหน่ง (x,y)
P1	3533,2733
P2	4427,2113
P3	5887,2500
P4	4507,3013
P5	3533,1200
P6	4427,580
P7	5887,967

ตารางที่ 4.2 ตารางแสดงข้อมูลจุด ณ ตำแหน่งใดๆบนรูปที่ 4.16

- จากตารางที่ 4.2 หาความยาวระหว่างจุด P5 และ P6 เป็นความยาวของ Lx1

Lx1 = sqrt(((P5(x)-P6(x))^2)+(P5(y)-P6(y))^2));

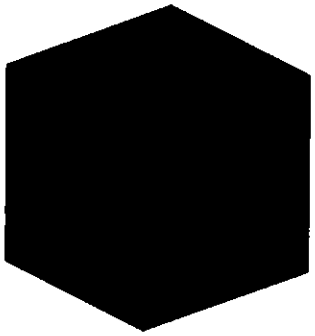
Lx1 = 1,087

- จากตารางที่ 4.2 หาความยาวระหว่างจุด P2 และ P6 เป็นความยาวของ Ly1

Ly1 = sqrt(((P2(x)-P6(x))^2)+(P2(y)-P6(y))^2));

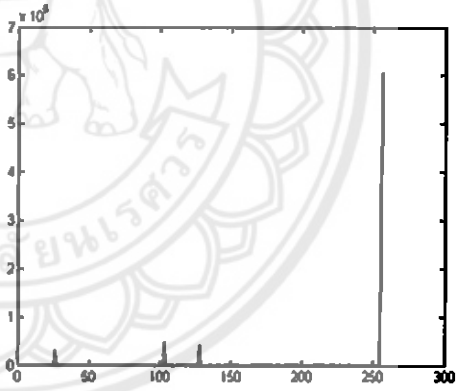
Ly1 = 1,533

2) จากตารางรูปอินพุต รูปที่ 2 ซึ่งแสดงดังรูป 4.17



รูปที่ 4.17 รูปอินพุตที่ 2 จากตารางรูปอินพุต

- จากรูปอินพุตที่ 2 ผ่านกระบวนการพลอตกราฟ ฮีสโตแกรม (Histogram) แล้วจะได้กราฟดังแสดงในรูปที่ 4.18



รูปที่ 4.18 กราฟ ฮีสโตแกรม (Histogram) จากรูปอินพุตที่ 2

- ผลการดึงค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

256 103 128 26

- ผลจากการเรียงลำดับค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

26 103 128 256

- ผลจากการจัดเก็บค่าลงในอาเรย์ใหม่ที่เริ่มต้นตำแหน่งแรกด้วย 0 ซึ่งจะได้ค่าดังนี้

0 26 103 128 256

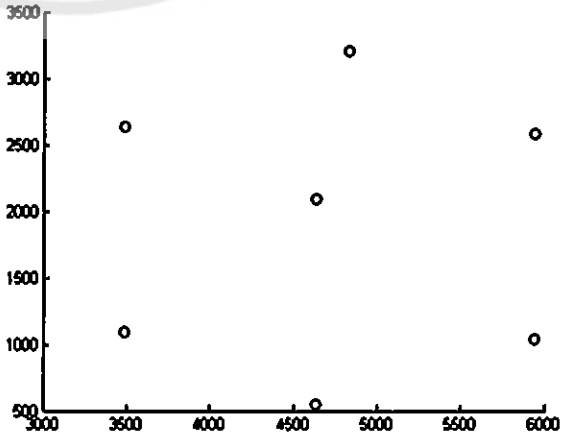
- ผลการตรวจสอบพื้นสีขาวที่มีค่าความสว่างเท่ากับ 256 ซึ่งจะได้ค่าดังนี้

0 26 103 128

- ผลจากการนำค่าระดับสีเทา (gray level) ผ่านกระบวนการแบ่งภาพตามค่าระดับสีเทา (gray level) ซึ่งจะได้ผลลัพธ์ดังแสดงในรูปที่ 4.19

รูปที่ 4.19 ผลการแบ่งภาพตามค่าระดับสีเทา (gray level)

- ผลจากการใช้ฟังก์ชัน corner ตามภาพที่แบ่งและผ่านกระบวนการจัดการจุดใหม่ ซึ่งจะได้กราฟดังแสดงในรูปที่ 4.20



รูปที่ 4.20 กราฟจุดใหม่ที่ได้จากกระบวนการจัดการจุด

- จากรูปที่ 4.20 ได้ข้อมูลของจุดมาดังแสดงในตารางที่ 4.3

ชื่อจุด	ตำแหน่ง (x,y)
P1	3487,2640
P2	4633,2093
P3	5940,2587
P4	4827,3207
P5	3487,1094
P6	4633,547
P7	5940,1041

ตารางที่ 4.3 ตารางแสดงข้อมูลจุด ณ ตำแหน่งใดๆ**บนรูปที่** 4.20

- จากตารางที่ 4.3 หาความยาวระหว่างจุด P5 และ P6 เป็น**ความยาวของ** Lx2

Lx2 = sqrt(((P5(x)-P6(x))^2)+(P5(y)-P6(y))^2));

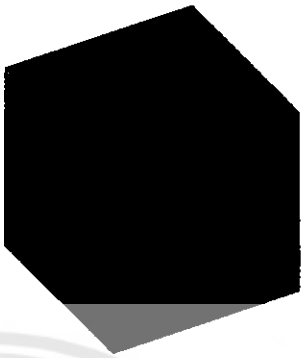
Lx2 = 1,269

- จากตารางที่ 4.3 หาความยาวระหว่างจุด P2 และ P6 เป็น**ความยาวของ** Ly2

Ly2 = sqrt(((P2(x)-P6(x))^2)+(P2(y)-P6(y))^2));

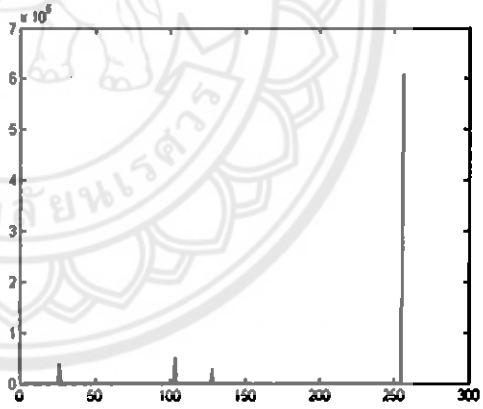
Ly2 = 1,546

3) จากตารางรูปอินพุต รูปที่ 3 แสดงดังรูปที่ 4.21



รูปที่ 4.21 รูปอินพุตที่ 3 จากตารางรูปอินพุต

- จากรูปอินพุตที่ 3 ผ่านกระบวนการพลอตกราฟฮีสโตแกรม (Histogram) แล้วจะได้กราฟดังแสดงในรูปที่ 4.22



ภาพที่ 4.22 กราฟ ฮีสโตแกรม (Histogram) จากรูปอินพุตที่ 3

- ผลการดึงค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

256 103 26 128

- ผลจากการเรียงลำดับค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

26 103 128 256

- ผลจากการจัดเก็บค่าลงในอาเรย์ใหม่ที่เริ่มต้นตำแหน่งแรกด้วย 0 ซึ่งจะได้ค่าดังนี้

0 26 103 128 256

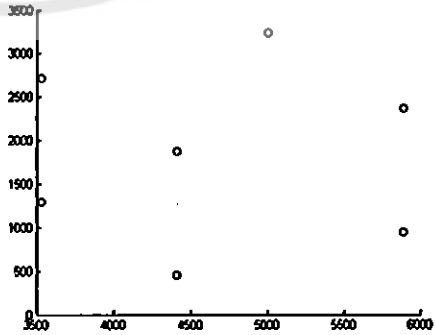
- ผลการตรวจสอบพื้นสีขาวที่มีค่าความสว่างเท่ากับ 256 ซึ่งจะได้ค่าดังนี้

0 26 103 128

- ผลจากการนำค่าระดับสีเทา (gray level) ผ่านกระบวนการแบ่งภาพตามค่าระดับสีเทา (gray level) ซึ่งจะได้ผลลัพธ์ดังแสดงในรูปที่ 4.23

รูปที่ 4.23 ผลการแบ่งภาพตามค่า ระดับสีเทา (gray level)

- ผลจากการใช้ฟังก์ชัน corner ตามภาพที่แบ่งและผ่านกระบวนการจัดการจุดใหม่ ซึ่งจะได้กราฟดังแสดงในรูปที่ 4.20



รูปที่ 4.24 กราฟจุดใหม่ที่ได้จากกระบวนการจัดการจุด

- จากรูปที่ 4.20 ได้ข้อมูลของจุดมาดังแสดงในตารางที่ 4.4

ชื่อจุด	ตำแหน่ง (x,y)
P1	3533,2720
P2	4413,1880
P3	5887,2373
P4	5000,3240
P5	3533,1293
P6	4413,453
P7	5887,946

ตารางที่ 4.4 ตารางแสดงข้อมูลจุด ณ ตำแหน่งใดๆบนรูปที่ 4.20

- จากตารางที่ 4.4 หาความยาวระหว่างจุด P5 และ P6 เป็นความยาวของ Lx3

Lx3 = sqrt(((P5(x)-P6(x))^2)+(P5(y)-P6(y))^2));

Lx3 = 1,216

- จากตารางที่ 4.4 หาความยาวระหว่างจุด P2 และ P6 เป็นความยาวของ Ly3

Ly3 = sqrt(((P2(x)-P6(x))^2)+(P2(y)-P6(y))^2));

Ly3 = 1,427

4) กระบวนการหาความยาวด้านและค่ามุมเริ่มต้นของรูปทรงลูกบาศก์

- จากความยาว Lx1 และ Lx2 นำไปหามุมแอลฟ่า (α) (องศา)

k = sqrt((1-Lx2^2)/(1-Lx1^2))

k = sqrt((1-1,269^2)/(1-1,087^2))

k = 1.1673

alpha = atand((sind(theta))/k-(cosd(theta)))

alpha = atand((sind(10))/1.1673-(cosd(10)))

alpha = 39.8971 ≈ 40 องศา

- จากความยาว Ly1 และ Ly3 นำไปหามุมเบต้า (β) (องศา)

k2 = sqrt((1-(Ly3^2))/(1-(Ly1^2)))

k2 = sqrt((1-(1,427^2))/(1-(1,533 ^2)))

k2 = 0.9309

beta = atand((cosd(gamma)-k2)/(sind(gamma)))

beta = atand((cosd(10)- 0.9309)/(sind(10)))

beta = 17.2602 ≈ 17 องศา

- จากมุมแอลฟ่า (α) (องศา) และ มุมเบต้า (β) (องศา) ที่ได้นำไปหาความยาวจริงแกน x และ y ได้

Lx = Lx2/sqrt(1-(sind(alpha+theta).^2)*(cosd(beta).^2)))

Lx = 1,269/sqrt(1-(sind(40+10).^2)*(cosd(17).^2)))

Lx = 1,866

Ly = Ly1/sqrt(1-(sind(alpha).^2)*(cosd(beta).^2)))

Ly = 1,533/sqrt(1-(sind(40).^2)*(cosd(17).^2)))

Ly = 1,944

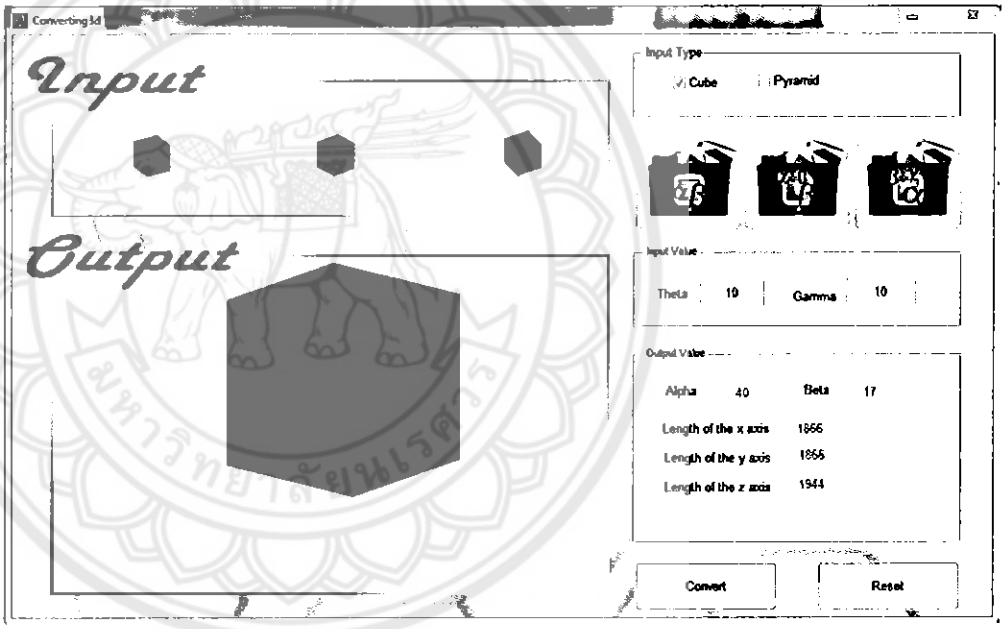
5) ผลจากกระบวนการการแปลงภาพ

- รายละเอียดเมื่อแปลงรูปแล้วซึ่งแสดงในตารางที่ 4.5

รูป	มุมเริ่มต้น (alpha) (องศา)	มุมเริ่มต้น (beta) (องศา)	ความยาวตามแนวแกน x	ความยาวตามแนวแกน y	ความยาวตามแนวแกน z
1	40	17	1,866	1,866	1,944

ตารางที่ 4.5 ตารางแสดงรายละเอียดเมื่อแปลงรูปแล้ว

- หน้าจอแสดงผลลัพธ์ดังแสดงในรูปที่ 4.25



รูปที่ 4.25 แสดงผลลัพธ์การทำงานของโปรแกรมทรงลูกบาศก์

4.3.2 ทรงพิระมิดมุมใดๆ

- รูปอินพุต

รูป	มุมเริ่มต้น (alpha) (องศา)	มุมเริ่มต้น (beta) (องศา)
1	40	20
2	30	20
3	40	30

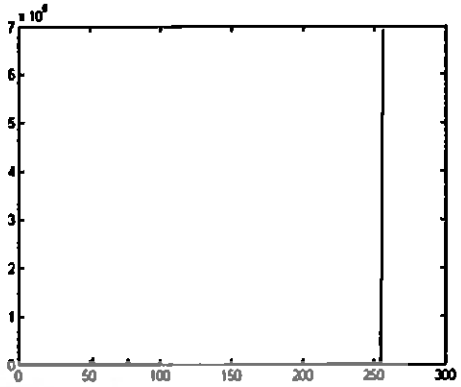
* มุมเซตา (θ) = 10 , แกมม่า (γ) = 10

ตารางที่ 4.6 ตารางแสดงข้อมูลรูปอินพุตทรงพิระมิดทั้ง 3 รูป

1) จากตารางรูปอินพุต รูปที่ 1 ซึ่งแสดงดังรูปที่ 4.26

รูปที่ 4.26 รูปอินพุตที่ 1 จากตารางรูปอินพุตทรงพิระมิด

- จากรูปอินพุตที่ 1 ผ่านกระบวนการพลอตกราฟฮีสโตแกรม (Histogram) แล้วจะได้กราฟดังแสดงในรูปที่ 4.27



รูปที่ 4.27 กราฟ ฮีสโตแกรม (Histogram) จากรูปอินพุตทรงพิระมิดที่ 1

- ผลการดึงค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

256 52 77

- ผลจากการเรียงลำดับค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

55 77 256

- ผลจากการจัดเก็บค่าลงในอาเรย์ใหม่ที่เริ่มต้นตำแหน่งแรกด้วย 0 ซึ่งจะได้ค่าดังนี้

0 55 77 256

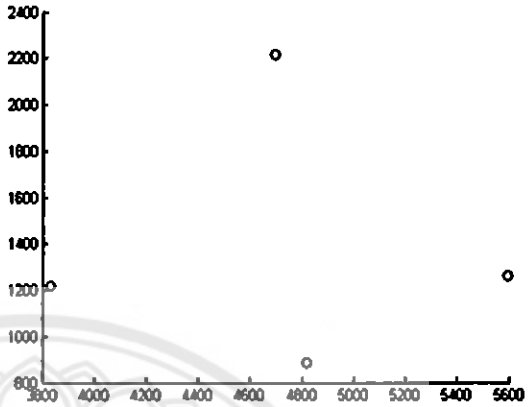
- ผลการตรวจสอบพื้นสีขาวที่มีค่าความสว่างเท่ากับ 256 ซึ่งจะได้ค่าดังนี้

0 55 77

- ผลจากการนำค่าระดับสีเทา (gray level) ผ่านกระบวนการแบ่งภาพตามค่าระดับสีเทา (gray level) ซึ่งจะได้ผลลัพธ์ดังแสดงในรูปที่ 4.28

รูปที่ 4.28 ผลการแบ่งภาพทรงพิระมิดที่ 1 ตามค่า ระดับสีเทา (gray level)

- ผลจากการใช้ฟังก์ชัน corner ตามภาพที่แบ่งและผ่านกระบวนการจัดการจุดใหม่ ซึ่งจะได้กราฟดังแสดงในรูปที่ 4.29



รูปที่ 4.29 กราฟจุดใหม่ที่ได้จากกระบวนการจัดการจุด (พิระมิด 1)

- จากรูปที่ 4.25 ได้ข้อมูลของจุดมาดังแสดงในตารางที่ 4.7

ชื่อจุด	ตำแหน่ง (x,y)
P1	3833,1220
P2	4820,887
P3	5593,1260
P4	4693,2213

ตารางที่ 4.7 ตารางแสดงข้อมูลจุด ณ ตำแหน่งใดๆบนรูปที่ 4.25

- จากตารางที่ 4.7 หาความยาวระหว่างจุด P1 และ P2 เป็นความยาวของ Lx1

Lx1 = sqrt(((P1(x)-P2(x))^2)+(P1(y)-P2(y))^2));

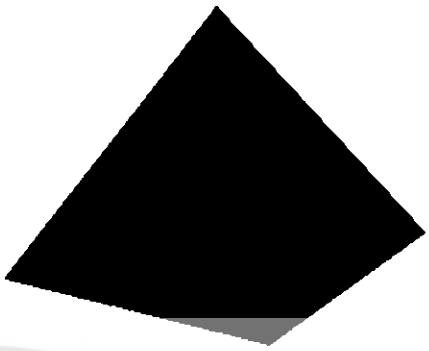
Lx1 = 1,041

- จากตารางที่ 4.7 หาความยาวระหว่างจุด P2 และ P4 เป็นความยาวของ Ly1

Ly1 = sqrt(((P2(x)-P4(x))^2)+(P2(y)-P4(y))^2));

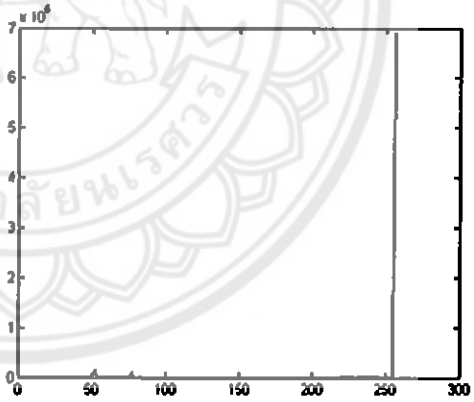
Ly1 = 1,332

2) จากตารางรูปอินพุต รูปที่ 2 ซึ่งแสดงดังรูปที่ 4.30



รูปที่ 4.30 รูปอินพุตที่ 2 จากตารางรูปอินพุตทรงพิระมิด

- จากรูปอินพุตที่ 2 ผ่านกระบวนการพลอตกราฟฮีสโตแกรม (Histogram) ได้ดังกราฟดังแสดงในรูปที่ 4.31



รูปที่ 4.31 กราฟฮีสโตแกรม (Histogram) จากรูปอินพุตทรงพิระมิดที่ 2

- ผลการดึงค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

256 52 77

- ผลจากการเรียงลำดับค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

55 77 256

- ผลจากการจัดเก็บค่าลงในอาเรย์ใหม่ที่เริ่มต้นตำแหน่งแรกด้วย 0 ซึ่งจะได้ค่าดังนี้

0 55 77 256

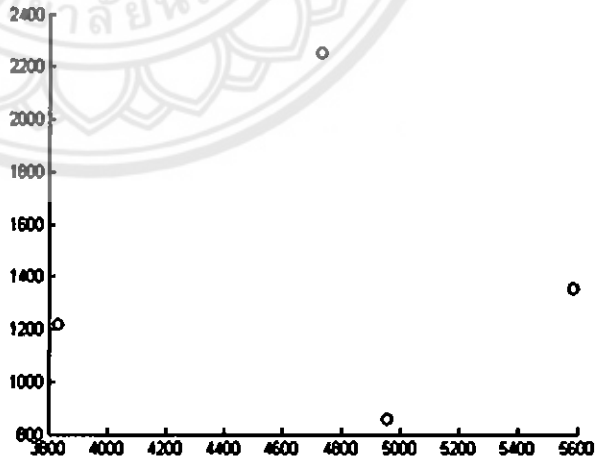
- ผลการตรวจสอบพื้นสีขาวที่มีค่าความสว่างเท่ากับ 256 ซึ่งจะได้ค่าดังนี้

0 55 77

- ผลจากการนำค่าระดับสีเทา (gray level) ผ่านกระบวนการแบ่งภาพตามค่าระดับสีเทา (gray level) ซึ่งจะได้ผลลัพธ์ดังแสดงในรูปที่ 4.32

รูปที่ 4.32 ผลการแบ่งภาพทรงพิระมิดที่ 2 ตามค่าระดับสีเทา (gray level)

- ผลจากการใช้ corner ตามภาพที่แบ่งและผ่านกระบวนการจัดการจุดใหม่ ซึ่งจะได้กราฟดังแสดงในรูปที่ 4.33



รูปที่ 4.33 กราฟจุดใหม่ที่ได้จากกระบวนการจัดการจุด (พิระมิด 2)

- จากรูปที่ 4.29 ได้ข้อมูลของจุดมาดังแสดงในตารางที่ 4.8

ชื่อจุด	ตำแหน่ง (x,y)
P1	3833,1220
P2	4953,860
P3	5587,1353
P4	4733,2253

ตารางที่ 4.8 ตารางแสดงข้อมูลจุด ณ ตำแหน่งใดๆบนรูปที่ 4.29

- จากตารางที่ 4.8 หาความยาวระหว่างจุด P1 และ P2 เป็นความยาวของ Lx1

Lx1 = sqrt(((P1(x)-P2(x))^2)+(P1(y)-P2(y))^2));

Lx1 = 1,176

- จากตารางที่ 4.8 หาความยาวระหว่างจุด P2 และ P4 เป็นความยาวของ Ly1

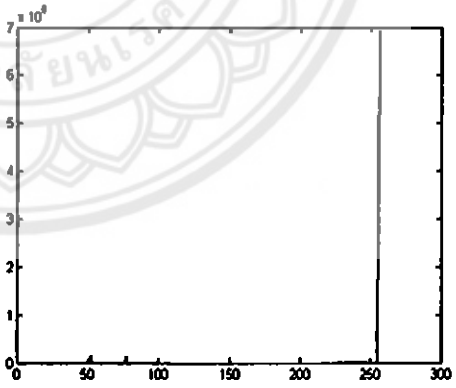
Ly1 = sqrt(((P2(x)-P4(x))^2)+(P2(y)-P4(y))^2));

Ly1 = 1,410

3) จากตารางรูปอินพุต รูปที่ 3 ซึ่งแสดงในรูปที่ 4.34

รูปที่ 4.34 รูปอินพุตที่ 3 จากตารางรูปอินพุตทรงพิระมิด

- จากรูปอินพุตที่ 3 ผ่านกระบวนการพลอตกราฟฮีสโตแกรม (Histogram) ได้กราฟดังแสดงในรูปที่ 4.35



รูปที่ 4.35 กราฟฮีสโตแกรม (Histogram) จากรูปอินพุตทรงพิระมิดที่ 3

- ผลการดึงค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

256 52 77

- ผลจากการเรียงลำดับค่าระดับสีเทา (gray level) จากกราฟฮีสโตแกรม (Histogram) ซึ่งจะได้ค่าดังนี้

55 77 256

- ผลจากการจัดเก็บค่าลงในอาเรย์ใหม่ที่เริ่มต้นตำแหน่งแรกด้วย 0 ซึ่งจะได้ค่าดังนี้

0 55 77 256

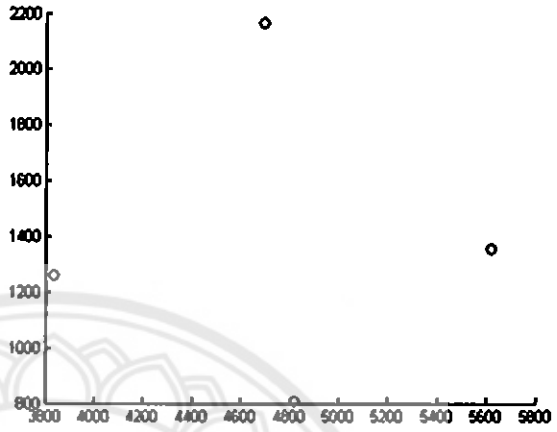
- ผลการตรวจสอบพื้นสีขาวที่มีค่าความสว่างเท่ากับ 256 ซึ่งจะได้ค่าดังนี้

0 55 77

- ผลจากการนำค่าระดับสีเทา (gray level) ผ่านกระบวนการแบ่งภาพตามค่าระดับสีเทา (gray level) ซึ่งจะได้ผลลัพธ์ดังแสดงในรูปที่ 4.36

รูปที่ 4.36 ผลการแบ่งภาพทรงพิระมิดที่ 3 ตามค่าระดับสีเทา (gray level)

- ผลจากการใช้ฟังก์ชัน corner ตามภาพที่แบ่งและผ่านกระบวนการจัดการจุดใหม่ ซึ่งจะได้กราฟดังแสดงในรูปที่ 4.37



รูปที่ 4.37 กราฟจุดใหม่ที่ได้จากกระบวนการจัดการจุด (พิระมิดที่ 3)

- จากรูปที่ 4.37 ได้ข้อมูลของจุดมาดังแสดงในตารางที่ 4.9

ชื่อจุด	ตำแหน่ง (x,y)
P1	3833,1260
P2	4820,807
P3	5620,1353
P4	4693,2167

ตารางที่ 4.9 ตารางแสดงข้อมูลจุด ณ ตำแหน่งใดๆบนรูปที่ 4.33

- จากตารางที่ 4.9 หาความยาวระหว่างจุด P1 และ P2 เป็นความยาวของ Lx3

Lx3 = sqrt(((P1(x)-P2(x))^2)+(P1(y)-P2(y))^2));

Lx3 = 1,085

- จากตารางที่ 4.9หาความยาวระหว่างจุด P2 และ P4 เป็นความยาวของ Ly3

Ly3 = sqrt(((P2(x)-P4(x))^2)+(P2(y)-P4(y))^2));

Ly3 = 1,365

4) กระบวนการหาความยาวด้านและค่ามุมเริ่มต้นของรูปทรงพีระมิด

- จากความยาว Lx1 และ Lx2 นำไปหามุมแอลฟ่า (α) (องศา)

k = sqrt((1-Lx2^2)/(1-Lx1^2))

k = sqrt((1-1,176^2)/(1-1,041^2))

k = 1.1286

alpha = atand((sind(theta))/k-(cosd(theta)))

alpha = atand((sind(10))/ 1.1286-(cosd(10)))

alpha = 39.7248 $\approx$ 40 องศา

- จากความยาว Lx1 และ Lx2 นำไปหาความยาวจริง L

X = ((((Lx2^2))/(sind(alpha)*sind(alpha)) -
(((Lx1^2))/(sind(alpha+theta)*sind(alpha+theta)))));

Y = (1/(sind(alpha)*sind(alpha))) -
(1/(sind(alpha+theta)*sind(alpha+theta)));

Lx = sqrt(abs(X/Y)))

X = ((((1,176^2))/(sind(40)*sind(40)) -
(((1,041^2))/(sind(40+10)sind(40+10)))));

Y = (1/(sind(40)*sind(40))) - (1/(sind(40+10)*sind(40+10)));

L = sqrt(abs(1,496/0.7162)))

L = 1,446

- จากมุมแอลฟ่า (α) (องศา) และ L ที่ได้นำไปหามุมเบต้า (β) (องศา)

k3 = sqrt((Ly1/(L/2))^2 - (sind(alpha) - cosd(alpha))^2);

k3 = sqrt((1,332/(1,446/2))^2 - (sind(40) - cosd(40))^2);

k3 = 1.8382

AA = ((((cosd(alpha) + sind(alpha))^2) - (k3^2)))

AA = ((((cosd(40) + sind(10))^2) - (1.8382^2)))

AA = -1.3942

BB = (2*sqrt(2)*(cosd(alpha) + sind(alpha)))

BB = (2*sqrt(2)*(cosd(40) + sind(40)))

BB = 3.9848

CC = (2-(k3^2))

CC = (2-(1.8382^2))

CC = -1.3790

beta1 = abs(round(atand(((-3.9848) + (sqrt(abs(3.9848^2 - (4*-1.3942*-1.3790))))) / (2*AA))))

beta1 = abs(round(atand(((-3.9848) + (sqrt(abs(BB^2 - (4*AA*CC))))) / (2*-1.3942))))

beta1 = 22 องศา

beta2 = abs(round(atand(((-BB) - (sqrt(abs(BB^2 - (4*AA*CC))))) / (2*AA))))

beta2 = abs(round(atand(((-3.9848) - (sqrt(abs(3.9848^2 - (4*1.3942*-1.3790))))) / (2*-1.3942))))

beta2 = 68 องศา

- จากมุมเบต้า (β) (องศา) ที่ได้สองค่า นำไปเปรียบเทียบกับมุมเบต้า (β) (องศา) 2 ค่า ของรูปที่ 3

ค่ามุมเบต้า (β) (องศา) ที่ได้จากรูปที่ 3

beta3 = 26 องศา

beta4 = 64 องศา

As = [26-gamma 64-gamma]

As = [26-10 64-10]

As = [16 54]

ผลการจับคู่มุมเบต้า (β) (องศา) ทั้งสองภาพดังนี้

ชุดแรก = [22 26]

ชุดที่สอง = [68 54]

เปรียบเทียบผลต่างของสองค่าเข้าใกล้ 0 มากที่สุดเลือกชุดมุมเบต้า (β) (องศา) นั้น

ผลต่างชุดแรก = |22-26|

ผลต่างชุดแรก = |-4| = 4

ผลต่างชุดที่สอง = |68-54|

ผลต่างชุดที่สอง = |14| = 14

ดังนั้นผลต่างชุดแรกเข้าใกล้ 0 มากกว่าเลือกมุมเบต้า (β) (องศา) รูปแรกของชุดนั้น

beta = 22 องศา

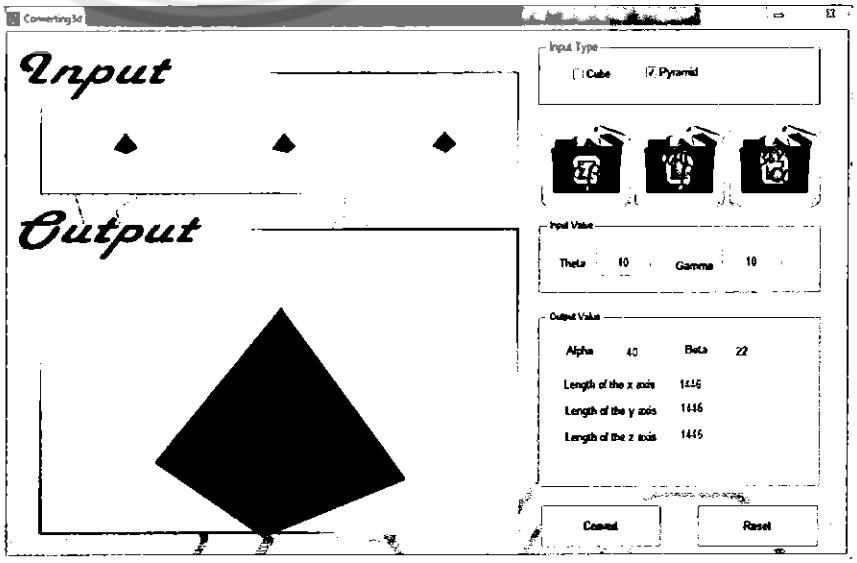
5) ผลจากกระบวนการการแปลงภาพ

- รายละเอียดเมื่อแปลงรูปแล้วซึ่งแสดงข้อมูลดังตารางที่ 4.10

รูป	มุมเริ่มต้น (alpha) (องศา)	มุมเริ่มต้น (beta) (องศา)	ความยาวตามแนวแกน x	ความยาวตามแนวแกน y	ความยาวตามแนวแกน z
1	40	22	1446	1446	1446

ตารางที่ 4.10 ตารางแสดงข้อมูลเมื่อแปลงรูปแล้ว

- หน้าจอแสดงผลลัพธ์ดังแสดงในรูปที่ 4.38



รูปที่ 4.38 แสดงผลลัพธ์การทำงานของโปรแกรมทรงพิระมิด

บทที่ 5
บทสรุป

5.1 วิเคราะห์และสรุปผลการทดลอง

เนื่องจากโครงงานนี้ใช้ความรู้เรื่อง การประมวลผลภาพและความรู้ทางด้านคณิตศาสตร์เป็นหลัก ทำให้โปรแกรมค่อนข้างมีข้อจำกัด และสิ่งที่ส่งผลต่อการทำงานของโปรแกรมเป็นอย่างมาก คือ ภาพ โดยภาพที่จะนำมาประมวลผลต้องเป็นภาพที่สามารถหามุมของรูปทรงเรขาคณิต (เฉพาะรูปทรงลูกบาศก์กับรูปทรงพีระมิด) ได้ชัดเจน อีกทั้งยังต้องระบุค่าความแตกต่างของมุมในแนวระนาบกับแนวดิ่งของภาพที่ใช้ประมวลผลด้วย นอกจากนี้โปรแกรมจะประมวลผลตามกระบวนการการคำนวณทางคณิตศาสตร์ ซึ่งส่งผลให้ข้อมูลมีความคลาดเคลื่อนได้

ในการทดลองคณะผู้จัดทำได้เน้นทำการทดลองในส่วนของภาพสองมิติที่ใช้ในการประมวลผลหาความยาวด้านในสองมิติ โดยได้ทำการทดลองในเรื่องของการปรับมุมในแนวระนาบและแนวดิ่งในหลายๆรูปแบบ และทำการทดสอบกับชุดทดลองจริง ผลที่ได้คือ โปรแกรมสามารถทำงานได้

5.2 ปัญหาและแนวทางแก้ไข

จากการทดลอง ทำให้ทราบถึงสาเหตุต่างๆ ที่ทำให้โปรแกรมเกิดข้อผิดพลาดเกิดขึ้น โดยส่งผลให้ไม่สามารถทำการสร้างภาพสามมิติกลับคืนได้อย่างสมบูรณ์ทุกค่ามุมในการหมุนรูป ซึ่งพบปัญหาอุปสรรคในการใช้งานและแนวทางแก้ไขปัญหาดังนี้

ปัญหาและอุปสรรค	แนวทางการแก้ไข
1. เนื่องจากโปรแกรมแมทแลบ (MATLAB) ไม่ใช่โปรแกรมกราฟฟิกโดยตรง เมื่อสร้างภาพออกมา ความชัดเจนของภาพลดลง	1. ใช้ซอฟท์แวร์อื่นร่วมด้วยเพื่อให้ได้ภาพที่ค่อนข้างสมบูรณ์
2. เนื่องด้วยความละเอียดของภาพมีผลต่อการทำงานการหาจุดมุมของภาพ	2. กำหนดความละเอียดของภาพให้เหมาะสมที่สุด

ตารางที่ 5.1 ตารางแสดงปัญหาและแนวทางการแก้ไข

ดังนั้นในทางปฏิบัติ ถ้าสามารถขจัดปัญหาและอุปสรรคดังกล่าวไปได้ ก็จะสามารถเพิ่มประสิทธิภาพการทำงานของโปรแกรมได้สมบูรณ์มากขึ้น

5.3 แนวทางการพัฒนาในอนาคต

1. ปรับปรุงให้โปรแกรมสามารถใช้งานกับภาพเรขาคณิตทรงอื่นๆ
2. ปรับปรุงให้สามารถวิเคราะห์พื้นผิวของภาพได้

เอกสารอ้างอิง

[1] Rafael C. Gonzalez and Richard E. Woods . (2008). **Digital Image Processing.** (3'rd Edition). United States of America: Prentice Hall

[2] "ระบบสี RGB" [online]. Available : http://bpiinc.wordpress.com/tag/rgb/

[3] "การหามุมโดยใช้เทคนิค The Harris Corner Detector" [online]. Available : http://en.wikipedia.org/wiki/Corner_detection

[4] "ระยะห่างระหว่างจุด" [online]. Available : http://dit.dru.ac.th/math/lesson5/waypoint.html

[5] "ฟังก์ชั่นตรีโกณมิติ" [online]. Available : http://th.wikipedia.org/wiki/%E0%B8%9F%E0%B8%B1%E0%B8%87%E0%B8%81%E0%B9%8C%E0%B8%8A%E0%B8%B1%E0%B8%99%E0%B8%95%E0%B8%A3%E0%B8%B5%E0%B9%82%E0%B8%81%E0%B8%93%E0%B8%A1%E0%B8%B4%E0%B8%95%E0%B8%B4

[5] "Projection" [online]. Available : http://www.bus.rmutt.ac.th/~suwat/Sheet3D.pdf

ภาคผนวก ก.

การสร้างรูปทรงลูกบาศก์ และ รูปทรงพีระมิด ด้วยโปรแกรม MATLAB

โค๊ด (Code) ที่ใช้ในการสร้างรูปทรงลูกบาศก์ และ รูปทรงพีระมิด

1. ทรงลูกบาศก์

```
clc
clear all

% Create the cube and save as jpg file.
figure(2);
patch([0 0 1 1],[0 1 1 0],[1 1 1 1],[0.1 0.1 0.1]); %top
patch([0 1 1 0],[0 0 0 0],[0 0 1 1],[0.2 0.2 0.2]); %fornt
patch([0 0 0 0],[0 1 1 0],[0 0 1 1],[0.3 0.3 0.3]); %left
patch([1 0 0 1],[1 1 1 1],[1 1 0 0],[0.4 0.4 0.4]);  %back
patch([1 1 1 1],[1 0 0 1],[1 1 0 0],[0.5 0.5 0.5]);  %right
patch([1 1 0 0],[1 0 0 1],[0 0 0 0],[0.6 0.6 0.6]); %bottom

camproj('orthographic')
theta = 40;
beta = 10;
campos([5,5,5])
camtarget([0,0,0])

axis vis3d off
axis square off
```

2. ทรงพีระมิด

```
% Create the pyramid and save as jpg file.
x = sqrt(2);
patch([0 0 1 1],[0 1 1 0],[0 0 0 0],[0.5 0.5 0.5]);
patch([0 0 0.5],[0 1 0.5],[0 0 1/x],[0.4 0.4 0.4]); % f
patch([0 1 0.5],[1 1 0.5],[0 0 1/x],[0.3 0.3 0.3]);
patch([1 1 0.5],[1 0 0.5],[0 0 1/x],[0.2 0.2 0.2]);
patch([1 0 0.5],[0 0 0.5],[0 0 1/x],[0.1 0.1 0.1]);

camproj('orthographic')
alpha = 40;
beta = 40;
campos([8,0,8])
camtarget([0,0,0])

axis vis3d off
axis square off
```

ภาคผนวก ข.

การหามุมของรูป

1. การหามุมของรูปทรงลูกบาศก์

```
function [ Pointface ] = find_corner_cube(imrgb)
%imrgb = imread('project_test/3.jpg');
imbw = rgb2gray(imrgb);

 % Extract basic information from the image
[sx,sy] = size(imbw);
maxval=max(double(imbw(:)));
minval=min(double(imbw(:)));

% Find Graylevels of sides by histogram
him = hist(double(imbw(:)),[minval:maxval]);
% plot(him)
[pval,val]=max(him); i=1;
while (pval > (0.01*sx*sy))  % Total area > 1% of image area
   v(i)=val; i=i+1;
   him(val)=0;
   [pval,val]=max(him);
end

if ( (length(v)<3) || (length(v)>10) )
   disp('Error in finding graylevels');
end

 v
% Each side detection
```

```
v=sort(v);
v=[0 v];
if (v(end)==256)    % Out of range / outer (inverse) side
   v=v(1:(end-1));

end

face = zeros(length(v)-1,sx,sy);

for i=1:(length(v)-1)
   fil1 = (imbw > v(i));
   fil2 = (imbw <= v(i+1));
   face(i,:,:) = fil1.*fil2;
end
for i=1:(length(v)-1)
figure(11)
subplot(2,2,i)
im = squeeze(face(i,:,:));
imshow(im)
end
% Find corner of each side stored in face

nocorner = 4; % number of detected corners
cornerface = zeros(length(v)-1,nocorner,2);     % 2 for x and y coordinates

for i=1:(length(v)-1)

   % Get the considered image of the side from face
   imside=squeeze(face(i,:,:));

%        se1 = strel('disk',4); se2 = strel('disk',7);
%     imside=imdilate(imerode(imside,se1),se2);
```

```
% Remove one line and one pixel noises by imerode and thenimdilate

% Do resize image to make the corner detection work properly
  se1 = strel('line',3,45);
%   imside=imerode(imside,se1);
  resizefactor = 0.15;
  imsidet = imresize(imside,resizefactor*[sx,sy]);
  imside = imresize(imsidet,[sx,sy]);
%   figure(11)
%   subplot(2,2,i)
%
%   imshow(imside)

    intr = flipud(imside);

%   theta = 180;
%   imsidet = imrotate(imsidet,theta);

  cxy = corner(intr,nocorner);
  cxy = round(cxy/resizefactor);

   if i==1
%       figure(5)
 %      hold on
 %      flipud(plot(cxy(:,1),cxy(:,2),'bo'));
     cornerface(i,:,:) = cxy;

  elseif i==2
 %      hold on
 %       plot(cxy(:,1),cxy(:,2),'ro');
     cornerface(i,:,:) = cxy;
```

```
    elseif i==3
%           hold on
%           plot(cxy(:,1),cxy(:,2),'go');
        cornerface(i,:,:) = cxy;

    end

end

Face2 = squeeze(cornerface(1,:,:));
Face3 = squeeze(cornerface(2,:,:));
Face4 = squeeze(cornerface(3,:,:));
point1 = [];
point2 = [];
point3 = [];
point4 = [];
for i = 1:4
  if (i == 1)
    point1 = [point1 Face2(i) Face2(i+4)];
  elseif (i == 2)
    point2 = [point2 Face2(i) Face2(i+4)];
  elseif (i == 3)
    point3 = [point3 Face2(i) Face2(i+4)];
  elseif (i == 4)
    point4 = [point4 Face2(i) Face2(i+4)];
  else
    disp('no point');
  end
end

checkpoint = min([point1(1) point2(1) point3(1) point4(1)]);
checkpoint2 = max([point1(1) point2(1) point3(1) point4(1)]);
```

```
p1 = [];
p2 = [];
p3 = [];
p4 = [];
p5 = [];
p6 = [];
p7 = [];

 checkpoint4 = max([point1(2) point2(2) point3(2) point4(2)]);
for j = 1:2
  if (point1(j) == checkpoint4)
     p4 = [p4 point1];

  elseif (point2(j) == checkpoint4)
     p4 = [p4 point2];

  elseif (point3(j) == checkpoint4)
     p4 = [p4 point3];

  elseif (point4(j) == checkpoint4)
     p4 = [p4 point4];

  else
      end
end

f2_point1 = [];
f2_point2 = [];
f2_point3 = [];
f2_point4 = [];
for i = 1:4
```

```
  if (i == 1)
    f2_point1 = [f2_point1 Face3(i) Face3(i+4)];
  elseif (i == 2)
    f2_point2 = [f2_point2 Face3(i) Face3(i+4)];
  elseif (i == 3)
    f2_point3 = [f2_point3 Face3(i) Face3(i+4)];
  elseif (i == 4)
    f2_point4 = [f2_point4 Face3(i) Face3(i+4)];
  else
    disp('no point');
  end
end

f3_point1 = [];
f3_point2 = [];
f3_point3 = [];
f3_point4 = [];
for fi = 1:4
  if (fi == 1)
    f3_point1 = [f3_point1 Face4(fi) Face4(fi+4)];
  elseif (fi == 2)
    f3_point2 = [f3_point2 Face4(fi) Face4(fi+4)];
  elseif (fi == 3)
    f3_point3 = [f3_point3 Face4(fi) Face4(fi+4)];
  elseif (fi == 4)
    f3_point4 = [f3_point4 Face4(fi) Face4(fi+4)];
  else
    disp('no point');
  end
end

checkpoint5 = [f2_point1(1) f2_point2(1) f2_point3(1) f2_point4(1)];
```

```
checkpoint5i = [f2_point1(1) f2_point2(1) f2_point3(1) f2_point4(1)];
checkpx2max = max([f2_point1(1) f2_point2(1) f2_point3(1) f2_point4(1)]);
checkpy2max = max([f2_point1(2) f2_point2(2) f2_point3(2) f2_point4(2)]);
checkpx2min = min([f2_point1(1) f2_point2(1) f2_point3(1) f2_point4(1)]);
checkpointp1 = [f3_point1(1) f3_point2(1) f3_point3(1) f3_point4(1)]

checkpx3min = min([f3_point1(1) f3_point2(1) f3_point3(1) f3_point4(1)]);
checkpy3max = max([f3_point1(2) f3_point2(2) f3_point3(2) f3_point4(2)]);

[P1_11] = min(checkpointp1);
AT = find(checkpointp1==P1_11);
checkpointp1(AT) = [];
[P1_12] = min(checkpointp1);
checkpointpt = [P1_11 P1_12]
for p = 1:2
   if (f3_point1(1)== checkpointpt(p)) & (f3_point1(2)== checkpy3max)
      p1 = [p1 f3_point1];
   end
   if (f3_point2(1)== checkpointpt(p)) & (f3_point2(2)== checkpy3max)
      p1 = [p1 f3_point2];
   end
   if (f3_point3(1)== checkpointpt(p)) & (f3_point3(2)== checkpy3max)
      p1 = [p1 f3_point3];
   end
   if (f3_point4(1)== checkpointpt(p)) & (f3_point4(2)== checkpy3max)
      p1 = [p1 f3_point4];
   end

end
[P3_11] = max(checkpoint5);
AU = find(checkpoint5==P3_11);
checkpoint5(AU) = [];
```

```
[P3_12] = max(checkpoint5);
checkpointp3 = [P3_11 P3_12];

for pxy2 = 1:2
   if (f2_point1(1)== checkpointp3(pxy2)) & (f2_point1(2)== checkpy2max)
      p3 = [p3 f2_point1];
   end
   if (f2_point2(1)== checkpointp3(pxy2)) & (f2_point2(2)== checkpy2max)
      p3 = [p3 f2_point2];
   end
   if (f2_point3(1)== checkpointp3(pxy2)) & (f2_point3(2)== checkpy2max)
      p3 = [p3 f2_point3];
   end
   if (f2_point4(1)== checkpointp3(pxy2)) & (f2_point4(2)== checkpy2max)
      p3 = [p3 f2_point4];
   end
end

datacal = [];
checkpoint5_1 = max(checkpoint5);
AI = find(checkpoint5==checkpoint5_1);
checkpoint5(AI) = [];
checkpoint5 = [checkpoint5];
u=unique(checkpoint5);
n=histc(checkpoint5,u);
if ((n(:) == 1))
     [Pm1] = min(checkpoint5i);
     ff = find(checkpoint5i==Pm1);
     checkpoint5i(ff) = [];
     [Pm2] = min(checkpoint5i);
   checkpoint6 = [Pm1 Pm2];
```

```
for j = 1:1
if ((f2_point1(j) == checkpoint6(j)) | (f2_point1(j) == checkpoint6(j+1)))
   datacal = [datacal f2_point1];
end
if ((f2_point2(j) == checkpoint6(j)) | (f2_point2(j) == checkpoint6(j+1)))
   datacal = [datacal f2_point2];
end
if ((f2_point3(j) == checkpoint6(j)) | (f2_point3(j) == checkpoint6(j+1)))
   datacal = [datacal f2_point3];
end
if ((f2_point4(j) == checkpoint6(j)) | (f2_point4(j) == checkpoint6(j+1)))
   datacal = [datacal f2_point4];
end

end
else
 b = min(checkpoint5i)
 for l = 1:1
   if (f2_point1(l) == b(l))
      datacal = [datacal f2_point1];
   end
   if (f2_point2(l) == b(l))
      datacal = [datacal f2_point2];
   end
   if (f2_point3(l) == b(l))
      datacal = [datacal f2_point3];
   end
   if (f2_point4(l) == b(l))
      datacal = [datacal f2_point4];
   end
 end
end
```

```
datacal
if (datacal(1)==datacal(3))
   checkp2x = [datacal(1) 0];
else
   checkp2x = [datacal(1) datacal(3)];
end
checkp2y = max([datacal(2) datacal(4)]);
checkp2x;
for pxy3 = 1:2
   if (f2_point1(1)== checkp2x(pxy3)) & (f2_point1(2)== checkp2y)
      p2 = [p2 f2_point1];
   end
   if (f2_point2(1)== checkp2x(pxy3)) & (f2_point2(2)== checkp2y)
      p2 = [p2 f2_point2];
   end
   if (f2_point3(1)== checkp2x(pxy3)) & (f2_point3(2)== checkp2y)
      p2 = [p2 f2_point3];
   end
   if (f2_point4(1)== checkp2x(pxy3)) & (f2_point4(2)== checkp2y)
      p2 = [p2 f2_point4];
   end
end
p2;
p1;
p3;
lengthface3 = round(sqrt(((datacal(1) - datacal(3))*(datacal(1) - datacal(3)))+((datacal(2)
- datacal(4))*(datacal(2) - datacal(4)))));
%
p5 = [p1(1) p1(2)-lengthface3];
p6 = [p2(1) p2(2)-lengthface3];
p7 = [p3(1) p3(2)-lengthface3];
```

```
p1;
p2;
p3;
p4;
p5;
p6;
p7;

Pointface = [p1;p2;p3;p4;p5;p6;p7];
end
```

2. การหามุมของรูปทรงพีระมิด

```
%  clc
%  clear all
function [ Pointface ] = find_corner_pyramid(imrgb)
%imrgb = imread('project_test/pyramidtest/3.jpg');
imbw = rgb2gray(imrgb);
%figure; imshow(imbw); colorbar;

% Extract basic information from the image
[sx,sy] = size(imbw);
maxval=max(double(imbw(:)));
minval=min(double(imbw(:)));

% Find Graylevels of sides by histogram
him = hist(double(imbw(:)),[minval:maxval]);
% figure(11)
% plot(him)

[pval,val]=max(him); i=1;
while (pval > (0.005*sx*sy))  % Total area > 1% of image area
        v(i)=val; i=i+1;
        him(val)=0;
        [pval,val]=max(him);
end
A = length(v)
if ( (length(v)<1) || (length(v)>10) )
        disp('Error in finding graylevels');
end

v
% Each side detection
```

```
v=sort(v);


v=[0 v];
if (length(v) >= 5)
        [Pm1] = min(v);
        ff = find(v==Pm1);
        v(ff) = [];
        [Pm2] = max(v);
        jj = find(v == Pm2);
        v(jj) = [];
        v = [v]
elseif (length(v) == 3)
        v = [v];


end
v
A = length(v)
if (v(end)==256)    % Out of range / outer (inverse) side
        v=v(1:(end-1));
end


face = zeros(length(v)-1,sx,sy);


for i=1:(length(v)-1)
   fil1 = (imbw > v(i));
   fil2 = (imbw <= v(i+1));
   face(i,:,:) = fil1.*fil2;
end
%
% for i=1:(length(v)-1)
% figure(12)
% subplot(2,2,i)
```

```
% im = squeeze(face(i,:,:));
% imshow(im)
% end

% Find corner of each side stored in face

nocorner = 3; % number of detected corners
cornerface = zeros(length(v)-1,nocorner,2);    % 2 for x and y coordinates

for i=1:(length(v)-1)

   % Get the considered image of the side from face
   imside=squeeze(face(i,:,:));

   % Remove one line and one pixel noises by imerode and then imdilate

 % Do resize image to make the corner detection work properly
%    se1 = strel('disk',3); se2 = strel('disk',7);
%    imside=imdilate(imerode(imside,se1),se2);
   resizefactor = 0.15;
   imsidet = imresize(imside,resizefactor*[sx,sy]);
   imside = imresize(imsidet,[sx,sy]);

   intr = flipud(imside);
%    theta = 180;
%    imsidet = imrotate(imsidet,theta);
   %imshow(imsidet);
   cxy = corner(intr,nocorner);
   cxy = round(cxy/resizefactor);
    %figure(66)

   if i==1
```

```
%      figure(66)
%       flipud(plot(cxy(:,1),cxy(:,2),'bo'));
       cornerface(i,:,:) = cxy;

   elseif i==2

%       hold on;
%       plot(cxy(:,1),cxy(:,2),'ro');
       cornerface(i,:,:) = cxy;

    elseif i==3

%       hold on;
%       plot(cxy(:,1),cxy(:,2),'go');
       cornerface(i,:,:) = cxy;

   end

end

face1 = squeeze(cornerface(1,:,:))
face2 = squeeze(cornerface(2,:,:))
FaceT = [face1;face2]

point1 = [];
point2 = [];
point3 = [];
point4 = [];
point5 = [];
point6 = [];
p1 = [];
p2 = [];
```

```
p3 = [];
p4 = [];
for i = 1:6
  if (i == 1)
    point1 = [point1 FaceT(i) FaceT(i+6)];
  elseif (i == 2)
    point2 = [point2 FaceT(i) FaceT(i+6)];
  elseif (i == 3)
    point3 = [point3 FaceT(i) FaceT(i+6)];
  elseif (i == 4)
    point4 = [point4 FaceT(i) FaceT(i+6)];
  elseif (i == 5)
    point5 = [point5 FaceT(i) FaceT(i+6)];
  elseif (i == 6)
    point6 = [point6 FaceT(i) FaceT(i+6)];
  else
    disp('no point');
  end
end
%check max point
Pmax = max([point1(1) point2(1) point3(1) point4(1) point5(1) point6(1)]);
%check min point
Pmin = min([point1(1) point2(1) point3(1) point4(1) point5(1) point6(1)]);

%check p1
 if (point1(1) == Pmin)
    p1 = [p1 point1];
 elseif (point2(1) == Pmin)
    p1 = [p1 point2];
 elseif (point3(1) == Pmin)
    p1 = [p1 point3];
 elseif (point4(1) == Pmin)
```

```
    p1 = [p1 point4];
elseif (point5(1) == Pmin)
    p1 = [p1 point5];
elseif (point6(1) == Pmin)
    p1 = [p1 point6];
else
    disp('no condition')
end
%check p3
if (point1(1) == Pmax)
    p3 = [p3 point1];
elseif (point2(1) == Pmax)
    p3 = [p3 point2];
elseif (point3(1) == Pmax)
    p3 = [p3 point3];
elseif (point4(1) == Pmax)
    p3 = [p3 point4];
elseif (point5(1) == Pmax)
    p3 = [p3 point5];
elseif (point6(1) == Pmax)
    p3 = [p3 point6];
else
    disp('no condition')
end
p1;
p3;

R = [point1;point2;point3;point4;point5;point6];

AT = find(R==p1(1));
AR = find(R==p3(1));
```

```
[R,ps] = removerows(R,[AT AR]);
% AS = [R(1) R(2) R(3) R(4)]
% tmp = abs(AS-p1(1))
% [idx idx] = min(tmp) %index of closest value
% closest = AS(idx)

Npoint1 = [];
Npoint2 = [];
Npoint3 = [];
Npoint4 = [];
for p = 1:4
  if (p == 1)
    Npoint1 = [Npoint1 R(p) R(p+4)];
  elseif (p == 2)
    Npoint2 = [Npoint2 R(p) R(p+4)];
  elseif (p == 3)
    Npoint3 = [Npoint3 R(p) R(p+4)];
  elseif (p == 4)
    Npoint4 = [Npoint4 R(p) R(p+4)];
  else
    disp('no point');
  end
end
RU = [Npoint1;Npoint2;Npoint3;Npoint4];
Rymin = [Npoint1(2) Npoint2(2) Npoint3(2) Npoint4(2)];
RyminT = min([Npoint1(2) Npoint2(2) Npoint3(2) Npoint4(2)]);
Rymax = [Npoint1(2) Npoint2(2) Npoint3(2) Npoint4(2)];
PI2 = [];
checkpoint2 = [];
[PI1] = min(Rymin);
VI = find(Rymin==PI1);
c = length(Rymin(VI))
```

```
if (c == 2)
 checkpoint2 = [Rymin(VI)];
else
 Rymin(VI) = [];
 [PI2] = min(Rymin);
 checkpoint2 = [PI1 PI2];
end
 checkpoint2
 datacal = [];

[PT1] = max(Rymax);
VT = find(Rymax==PT1);
Rymax(VT) = [];
[PT2] = max(Rymax);

 checkpoint4 = [PT1 PT2]

for m = 1:1
 if ((Npoint1(m+1) == checkpoint2(m)) | (Npoint1(m+1) == checkpoint2(m+1)))
     datacal = [datacal Npoint1];
 end
 if ((Npoint2(m+1) == checkpoint2(m)) | (Npoint2(m+1) == checkpoint2(m+1)))
     datacal = [datacal Npoint2];
 end
 if ((Npoint3(m+1) == checkpoint2(m))  | (Npoint3(m+1) == checkpoint2(m+1)))
     datacal = [datacal Npoint3];
 end
 if ((Npoint4(m+1) == checkpoint2(m)) | (Npoint4(m+1) == checkpoint2(m+1)))
     datacal = [datacal Npoint4];
 end
end
datacal;
```

```
AS = [datacal(1) datacal(3)];
tmp = abs(AS-p3(1));
[idx idx] = min(tmp); %index of closest value
closes = AS(idx) ;
datacal = [datacal(1) datacal(2);datacal(3) datacal(4)];

closes(1)
if (datacal(1) == datacal(3))
    p2 =[p2 datacal(1) min([datacal(2) datacal(4)])];
elseif (datacal(1) ~= datacal(3))
 if (datacal(1) == closes(1))
    p2 = [p2 datacal(1) datacal(3)];
 elseif (datacal(2) == closes(1))
    p2 = [p2 datacal(2) datacal(4)];
 end
end
%find p4
datacal2 = [];
for n = 1:1
 if ((Npoint1(n+1) == checkpoint4(n)) | (Npoint1(n+1) == checkpoint4(n+1)))
    datacal2 = [datacal2 Npoint1];
 end
 if ((Npoint2(n+1) == checkpoint4(n)) | (Npoint2(n+1) == checkpoint4(n+1)))
    datacal2 = [datacal2 Npoint2];
 end
 if ((Npoint3(n+1) == checkpoint4(n))  | (Npoint3(n+1) == checkpoint4(n+1)))
    datacal2 = [datacal2 Npoint3];
 end
 if ((Npoint4(n+1) == checkpoint4(n)) | (Npoint4(n+1) == checkpoint4(n+1)))
    datacal2 = [datacal2 Npoint4];
 end
```

```
end
datacal2
ASP = [datacal2(1) datacal2(3)];
tmp1 = abs(ASP-p1(1));
[idx1 idx1] = min(tmp1); %index of closest value
closes2 = ASP(idx1) ;
datacal2 = [datacal2(1) datacal2(2);datacal2(3) datacal2(4)];
datacal3 = max([datacal2(3) datacal2(4)])

closes2(1)
if (datacal2(1) == datacal2(3))
   p4 =[p4 datacal2(1) max([datacal2(2) datacal2(4)])];
elseif (datacal2(1) ~= datacal2(3))
 if (datacal2(3) == datacal3)
     p4 = [p4 datacal2(1) datacal2(3)];
 elseif (datacal2(4) == datacal3)
     p4 = [p4 datacal2(2) datacal2(4)];
 end
end
p2;
p4;

l1 = round(sqrt(((p1(1)-p3(1))^2)+((p1(2)-p3(2))^2)));
l1 = l1/2;
k = abs(p4(1)-p2(1));
ly = round(sqrt((l1^2-k^2)+p2(2)^2));
lm = abs(p4(2)-ly);
pm = [p4(1) ly];
p5 = [p1(1) p1(2)+lm];
p6 = [p2(1) p2(2)+lm];
p7 = [p3(1) p3(2)+lm];
```

```
Pointface = [p1;p2;p3;p4];
%
% figure(88)
% x = [p1(1) p2(1) p3(1) p4(1)];
% y = [p1(2) p2(2) p3(2) p4(2)];
%
% scatter(x,y,'bo')
end
```

ภาคผนวก ค.

การคำนวณหาค่าความยาวด้านและมุมเริ่มต้น

1. ทรงลูกบาศก์

```
Face_A = find_corner_cube(img1);
Face_B = find_corner_cube(img2);
Face_C = find_corner_cube(img3);
set(0,'DefaultFigureVisible','off');
str1 = get(handles.theta,'String');
theta = str2num(str1);
str2 = get(handles.gamma,'String');
grammar = str2num(str2);

Lengthx_FaceA = sqrt(((Face_A(5) - Face_A(6))*(Face_A(5) - Face_A(6)))+((Face_A(12) - Face_A(13))*(Face_A(12) - Face_A(13))));%Lx1
Lengthx_FaceB = sqrt(((Face_B(5) - Face_B(6))*(Face_B(5) - Face_B(6)))+((Face_B(12) - Face_B(13))*(Face_B(12) - Face_B(13))));%Lx2
Lengthx_FaceC = sqrt(((Face_C(5) - Face_C(6))*(Face_C(5) - Face_C(6)))+((Face_C(12) - Face_C(13))*(Face_C(12) - Face_C(13))));%Lx3
Lengthy_FaceA = sqrt(((Face_A(2) - Face_A(6))*(Face_A(2) - Face_A(6)))+((Face_A(9) - Face_A(13))*(Face_A(9) - Face_A(13))));%Ly1
Lengthy_FaceB = sqrt(((Face_B(2) - Face_B(6))*(Face_B(2) - Face_B(6)))+((Face_B(9) - Face_B(13))*(Face_B(9) - Face_B(13))));%Ly2
Lengthy_FaceC = sqrt(((Face_C(2) - Face_C(6))*(Face_C(2) - Face_C(6)))+((Face_C(9) - Face_C(13))*(Face_C(9) - Face_C(13))));%Ly3

Lx1 = round(Lengthx_FaceA)
Lx2 = round(Lengthx_FaceB)
```

```
Lx3 = Lx2
Ly1 = round(Lengthy_FaceA)
Ly2 = Ly1
Ly3 = round(Lengthy_FaceC)

k = sqrt((1-Lx2^2)/(1-Lx1^2))

alpha = atand((sind(theta))/k-(cosd(theta)))

alpha = round(alpha);
alpha = abs(alpha);
k2 = sqrt((1-(Ly3^2))/(1-(Ly1^2)))
set(handles.alpha,'String',num2str(alpha));

Beta = atand((cosd(grammar)-k2)/(sind(grammar)))
 Beta = round(Beta);
 Beta = abs(Beta);
set(handles.beta,'String',num2str(Beta));
Lx = round(Lx2/sqrt(1-(sind(alpha+theta).^2)*(cosd(Beta).^2)))
%   X = ((((Lx2^2))/(sind(alpha)*sind(alpha)) -
(((Lx1^2))/(sind(alpha+theta)*sind(alpha+theta)))));
%    Y = (1/(sind(alpha)*sind(alpha))) - (1/(sind(alpha+theta)*sind(alpha+theta)));
%    Lx = round(sqrt(abs(X/Y))) %???

%    Ly = round(Ly3/sqrt(1-(sind(alpha).^2)*(cosd(Beta+grammar).^2)))
   Ly = round(Ly1/sqrt(1-(sind(alpha).^2)*(cosd(Beta).^2)))
```

2. ทรงพีระมิด

```
Face_A = find_corner_pyramid(img1);
   Face_B = find_corner_pyramid(img2);
   Face_C = find_corner_pyramid(img3);

   str1 = get(handles.theta,'String');
   theta = str2num(str1);
   str2 = get(handles.gamma,'String');
   gamma = str2num(str2);

Lengthx_FaceA = sqrt(((Face_A(1) - Face_A(2))^2) + ((Face_A(5) - Face_A(6))^2)
);%Lx1
Lengthx_FaceB = sqrt(((Face_A(1) - Face_B(2))^2) + ((Face_A(5) - Face_B(6))^2)
);%Lx2
Lengthx_FaceC = sqrt(((Face_C(1) - Face_C(2))^2) + ((Face_C(5) - Face_C(6))^2)
);%Lx3
Lengthy_FaceA = sqrt(((Face_A(2) - Face_A(4))^2) + ((Face_A(6) - Face_A(8))^2)
);%Ly1
Lengthy_FaceB = sqrt(((Face_B(2) - Face_B(4))^2) + ((Face_B(6) - Face_B(8))^2)
);%Ly2
Lengthy_FaceC = sqrt(((Face_C(2) - Face_C(4))^2) + ((Face_C(6) - Face_C(8))^2)
);%Ly3

Lx1 = round(Lengthx_FaceA)
Lx2 = round(Lengthx_FaceB)
Lx3 = Lx2;
Ly1 = round(Lengthy_FaceA)
Ly2 = Ly1;
Ly3 = round(Lengthy_FaceC)

k = sqrt((1-Lx2^2)/(1-Lx1^2))
```

```
alpha = atand((sind(theta))/k-(cosd(theta)))

alpha = round(alpha);
alpha = abs(alpha)
k2 = sqrt((1-Ly3^2)/(1-Ly1^2))
ap = num2str(alpha)
set(handles.alpha,'String',ap);
X = ((((Lx2^2))/(sind(alpha)*sind(alpha)) -
(((Lx1^2))/(sind(alpha+theta)*sind(alpha+theta)))))
Y = (1/(sind(alpha)*sind(alpha))) -(1/(sind(alpha+theta)*sind(alpha+theta)))
Lx = round(sqrt(abs(X/Y))) %???
k3 = sqrt( (Ly1/(Lx/2))^2 - (sind(alpha) - cosd(alpha))^2)
U = sind(alpha)-cosd(alpha);
T = U^2;
I = Ly1/(Lx/2);
IU = I^2;
RR = sqrt(IU-T);

AA = (( ((cosd(alpha) + sind(alpha))^2) - (k3^2) ))
BB = (2*sqrt(2)*(cosd(alpha) + sind(alpha)))
CC = (2-(k3^2))
beta1 = abs(round( atand( ((-BB) + (sqrt(abs(BB^2 - (4*AA*CC))))) / (2*AA) ))) %
?????? ?????
beta2 = abs(round( atand( ((-BB) - (sqrt(abs(BB^2 - (4*AA*CC))))) / (2*AA) ) ))



k4 = sqrt(abs( ((Ly3/(Lx/2))^2) - (sind(alpha) - cosd(alpha))^2))
AA2 = ( ((cosd(alpha) + sind(alpha))^2) - (k4^2) );
BB2 = (2*sqrt(2)*(cosd(alpha) + sind(alpha)));
CC2 = 2-(k4^2);
```

```
beta3 = abs(round( atand( ((-BB2) + (sqrt(abs(BB2^2 - (4*AA2*CC2))))) / (2*AA2) ) ))
beta4 = abs(round( atand( ((-BB2) - (sqrt(abs(BB2^2 - (4*AA2*CC2))))) / (2*AA2) ) ))
beta31 = beta3-gamma;
beta41 = beta4-gamma;

YI = [beta31 beta41];
UI = abs(abs(YI(1)) - abs(beta1));
UT = abs(abs(YI(2)) - abs(beta1));
if (UI < UT)
   test1 = [beta1 YI(1)];
elseif (UI > UT)
   test1 = [beta1 YI(2)];
else
   test1 = [beta1 YI(1)];
end

RI = abs(YI(1) - beta2);
ST = abs(YI(2) - beta2);
if (RI < ST)
   test2 = [beta2 YI(1)];
elseif (RI > ST)
   test2 = [beta2 YI(2)];
end
fa = test1(1)-test1(2);
za = test2(1)-test2(2);

AS = [fa za];
tmp = 0;
[idx idx] = min(tmp); %index of closest value
closes = AS(idx);

[PT1] = closes;
```

```
VT = find(AS==PT1);
YU = [VT];
if (YU(1) == 1)
   YU = [test1(1) test1(2)+gamma];
elseif (YU(1) == 2)
   YU = [test2(1) test2(2)+gamma];
end

YU
% if (YU(1) > YU(2))
%    beta = YU(2)
% elseif (YU(1) < YU(2))
%    beta = YU(1)
% else
%    beta = YU(1)
% end
beta = YU(1)
Ly =Lx
```

ภาคผนวก ง.

การสร้างภาพสามมิติกลับคืนจากภาพสองมิติ

1. ทรงลูกบาศก์

```
x1 = 0;
y1 = 0;
z1 = 0;
x2 = round(x1 + Lx);
y2 = round(y1);
z2 = round(z1);
x3 = round(x1);
y3 = round(y1);
z3 = round(z2 + Ly);
x4 = round(x2);
y4 = round(y2);
z4 = round(z3);
x5 = round(x3);
y5 = round(y3 + Lx);
z5 = round(z3);
x6 = round(x4);
y6 = round(y4 + Lx);
z6 = round(z4);
x7 = round(x2);
y7 = round(y2 + Lx);
z7 = round(z2);
x8 = round(x1) ;
y8 = round(y1 + Lx);
z8 = round(z1);
% for alpha = alpha:160
```

```
axes(handles.axes4)
patch([x1 x2 x4 x3],[y1 y2 y4 y3],[z1 z2 z4 z3],[0.1 0.1 0.1]);%yellow
patch([x4 x2 x7 x6],[y4 y2 y7 y6],[z4 z2 z7 z6],[0.2 0.2 0.2]);%magenta
patch([x3 x4 x6 x5],[y3 y4 y6 y5],[z3 z4 z6 z5],[0.3 0.3 0.3]);%cyan
patch([x3 x1 x8 x5],[y3 y1 y8 y5],[z3 z1 z8 z5],[0.4 0.4 0.4]);%red
patch([x5 x8 x7 x6],[y5 y8 y7 y6],[z5 z8 z7 z6],[0.5 0.5 0.5]);%green
patch([x1 x2 x7 x8],[y1 y2 y7 y8],[z1 z2 z7 z8],[0.6 0.6 0.6]);%blue

xlabel('X');
ylabel('Y');
zlabel('Z');

axis square off
rotate3d on;
% camproj('orthographic')
%  view(alpha,Beta)

campos([cosd(alpha)*cosd(Beta),sind(alpha)*cosd(Beta),sind(Beta)])
camtarget([0,0,0])
% end
```

2. ทรงพีระมิด

```
x1 = 0;
y1 = 0;
z1 = 0;
x2 = x1+Lx
y2 = y1
z2 = z1
x7 = x2
y7 = y2+Lx
z7 = z2
x8 = x1
y8 = y1+Lx
z8 = z1
x3 = x2/2
y3 = y7/2
z3 = (z2+Lx)/sqrt(2)
x4 = x3
y4 = y3
z4 = z3
x6 = x4
y6 = y4
z6 = z4
x5 = x6
y5 = y6
z5 = z6
axes(handles.axes4) %
patch([x1 x2 x4 x3],[y1 y2 y4 y3],[z1 z2 z4 z3],[0.1 0.1 0.1]);%yellow
patch([x4 x2 x7 x6],[y4 y2 y7 y6],[z4 z2 z7 z6],[0.2 0.2 0.2]);%magenta
patch([x3 x4 x6 x5],[y3 y4 y6 y5],[z3 z4 z6 z5],[0.3 0.3 0.3]);%cyan
patch([x3 x1 x8 x5],[y3 y1 y8 y5],[z3 z1 z8 z5],[0.4 0.4 0.4]);%red
patch([x5 x8 x7 x6],[y5 y8 y7 y6],[z5 z8 z7 z6],[0.5 0.5 0.5]);%green
```

```
patch([x1 x2 x7 x8],[y1 y2 y7 y8],[z1 z2 z7 z8],[0.6 0.6 0.6]);%blue

xlabel('X');
ylabel('Y');
zlabel('Z');


axis square off
rotate3d on;
% camproj('orthographic')

view(3)
campos([cosd(alpha)*cosd(beta),sind(alpha)*cosd(beta),sind(bet)]);
% camtarget([0,0,0]);
```