รายงานโครงการวิจัยประจำปีงบประมาณ 2546

เรื่อง

การศึกษาและพัฒนาการคำนวณค่าคุณสมบัติของกล้องวีดีโอด้วยวิธีการแบบ Optical-based โดยอาศัยภาพมุมเดียวและหลายมุม

(A Study and Development of Optical-based Algorithms for Video Camera Parameter Estimation Using Single-view and Multiple-view Images)

โดย

รศ. ดร. นพพร โชติกกำธร

และ

นาย วรวิทย์ แสงเรือง

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ 10520

พ.ศ. 2547

บทคัดย่อ

ในปัจจุบัน การสร้างสรรค์ผลงานด้านภาพยนตร์และรายการโทรทัศน์มีความหลากหลายมากขึ้น ส่วนสำคัญของกระบวนการถ่ายทำได้แก่การใช้ระบบ virtual studio ซึ่งจะประกอบไปด้วยกล้องวีดีโอ และฉากซึ่งเป็น blue screen โดยกล้องจะจับภาพวัตถุหรือคนซึ่งจะอยู่หน้าฉาก แล้วนำภาพที่ได้นี้มาผ่านกระบวนการของการประมวลผลภาพ แล้วนำภาพพื้นหลังที่สร้างจากเทคนิคของคอมพิวเตอร์กราฟฟิกมาแทนที่ส่วนของฉากที่เป็น blue screen จะได้เป็นภาพวีดีโอที่สมบูรณ์ที่จะนำเสนอ ในการวิจัยนี้เป็นการถ่ายภาพจากกล้องวีดีโอซึ่งอาศัยฉากแบบ binary pattern ในการถ่ายทำ แล้วนำภาพที่ได้มาทำการวิเคราะห์ด้วยฟังก์ชั่นที่สร้างขึ้นด้วยโปรแกรม Matlab เพื่อหาค่าคุณสมบัติของกล้อง แล้วนำมาเปรียบเทียบกันระหว่างภาพที่ถ่ายในตำแหน่งที่แตกต่างกัน

ABSTRACT

In present , development in computer graphic and image processing helps us for a variety of producing movies and television programs.The important part of video production is the use of visual studio system which consist of the camera and the blue screen. The camera captures object and human in front of a blue screen. After that , camera will replaces the blue screen with the background pictures which is created by computer graphic technique. The result of theses processes is the complete video image that is ready to present. In the research, the blue screen binary pattern is used in the video production. The video captured images are analyzed by using functions which is developed in Matlab to find the camera parameters. Then bringing the results to compare between the images which captured in the different location.

กิตติกรรมประกาศ

โครงการการศึกษาและพัฒนาการคำนวณค่าคุณสมบัติของกล้องวีดีโอด้วยวิธีการแบบ Optical-based โดยอาศัยภาพมุมเดียวและหลายมุมได้รับความร่วมมือจากบุคคลหลายฝ่าย ดังนี้

ขอขอบพระคุณ รศ.ดร.นพพร โชติกกำธร อาจารย์ที่ปรึกษาวิชาโครงการพัฒนาระบบงาน ที่กรุณาให้คำแนะนำและเป็นที่ปรึกษา อันเป็นประโยชน์ต่อการพัฒนาระบบ

นอกจากนี้ข้าพเจ้าต้องขอกราบขอบพระคุณ บิดา มารดา และบุคคลในครอบครัว รวมทั้งขอขอบคุณเพื่อนๆ พี่ๆ IS 13.1 และที่ MVlab ทุกคนที่ให้ความช่วยเหลือในด้านต่างๆ จนการทำโครงการพัฒนาระบบนี้สำเร็จด้วยดี

โครงงานวิจัยนี้ได้รับการสนับสนุนทุนวิจัยจากเงินรายได้ของคณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ประจำปีงบประมาณ 2546

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาของปัญหา

การพัฒนาไปของคอมพิวเตอร์กราฟฟิก (computer graphic) และเทคนิคการประมวลผลภาพ (image processing) ส่งผลให้การสร้างสรรค์ผลงานด้านภาพยนตร์และรายการโทรทัศน์มีความแปลกใหม่และหลากหลายมากขึ้นเพื่อเป็นการดึงดูดความสนใจของผู้ชม โดยส่วนสำคัญของกระบวนการถ่ายทำได้แก่การใช้ระบบ virtual studio ซึ่งจะประกอบด้วยกล้องวีดีโอ และฉากซึ่งเป็นสีน้ำเงิน (blues screen) ตัวอย่างเช่น การนำเสนอข่าวพยากรณ์อากาศ ที่พิธีกรจะกล่าวรายงานข่าว ขณะที่เบื้องหลังจะแสดงภาพของแผนที่หรือรูปภาพแสดงสภาพอากาศ โดยในการถ่ายทำ พิธีกรหรือผู้นำเสนอจะยืนอยู่ข้างหน้าของ blue screen ซึ่งจะเรียกภาพนี้ว่าภาพ foreground และในส่วนของภาพ background นั้น จะถูกสร้างขึ้นจากเทคนิคของคอมพิวเตอร์กราฟฟิก แล้วนำมารวมกันกับภาพ foreground โดยจะนำภาพ background ที่สร้างขึ้นมาแทนที่ลงบน blue screen รวมเป็นภาพวีดีโอที่สมบูรณ์พร้อมจะนำเสนอ

การเคลื่อนไหวของกล้องวีดีโอ ได้แก่ การซูมเข้า-ออก (zooming) การส่ายกล้องไปด้านซ้าย-ขวา (panning) หรือ การเอียงขึ้น-ลงของกล้อง (tilting) ที่ทำให้ตำแหน่งและมุมมองของการจับภาพ (capture) ของกล้องเปลี่ยนแปลงไปดูเหมือนจะเป็นปัญหาที่สำคัญของเทคโนโลยี chromakeying แบบเก่าสำหรับการนำภาพ foreground กับภาพ background มารวมกันให้เกิดความถูกต้องและเหมาะสมกับตำแหน่งของการจับภาพของกล้องวีดีโอ

ดังนั้นจึงมีการนำเอา 2- tone blue screen ซึ่งเป็น blue screen แบบ binary pattern ที่ออกแบบให้ใช้ระดับของสีน้ำเงิน 2 ระดับ คือ ใช้สีน้ำเงินเข้ม และสีน้ำเงินอ่อนสร้างเป็น pattern เพื่อใช้ในการอ้างตำแหน่งบน blue screen และยังนำเอามาใช้ประโยชน์ในการวิเคราะห์ภาพที่ถ่ายเพื่อหามุมการหันเหที่เปลี่ยนไปของกล้องเพื่อลดข้อผิดพลาดที่เกิดจากการรวมกันระหว่างภาพ foreground และ background

1.2 วัตถุประสงค์ของการพัฒนาระบบ

1. ศึกษากรรมวิธีและอัลกอริทึ่มในการวิเคราะห์ภาพ
2. ศึกษาอัลกอริทึ่มในการคำนวณค่ามุมการหันเหของกล้องวีดีโอ
3. เพื่อนำเอาค่าที่ได้ไปใช้ในการสร้างภาพด้วยคอมพิวเตอร์กราฟฟิกมาแทนที่ส่วนของฉากพื้นหลังได้อย่างเหมาะสม

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. เรียนรู้เทคนิคของการประมวลผลภาพ และคอมพิวเตอร์กราฟฟิก
2. เพื่อช่วยลดข้อผิดพลาดในการสร้างภาพจากคอมพิวเตอร์กราฟฟิกเพื่อนำมาแทนที่ส่วนที่เป็นพื้นหลัง
3. เป็นแนวทางให้กับผู้ศึกษาการถ่ายทำด้วยกล้องวีดีโอในระบบ virtual studio
4. ได้ฟังก์ชันในการคำนวณหามุมการหันเหของกล้องวีดีโอที่สามารถนำไปใช้กับการทำงานด้านคอมพิวเตอร์กราฟฟิกที่เกี่ยวข้องกันได้

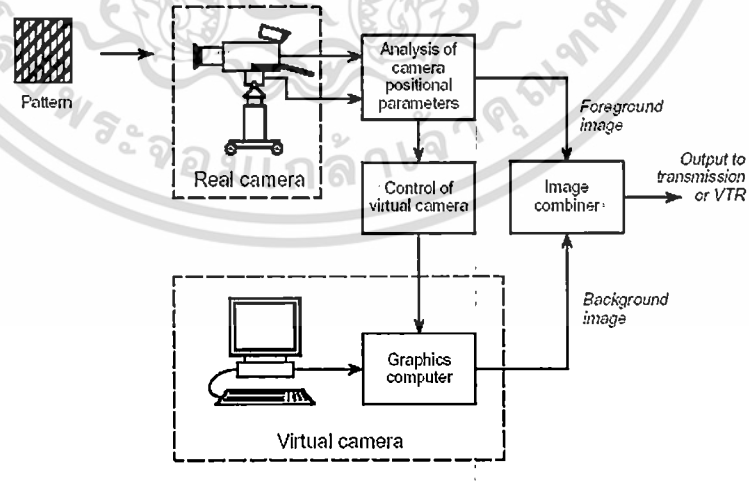
บทที่ 2

หลักการในการหามุมการหันเหของกล้องวีดีโอ

2.1 ระบบ Virtual Studio

เป็นระบบที่ใช้ในการถ่ายทำภาพยนตร์โดยส่วนประกอบที่สำคัญ คือ ฉาก , กล้องวีดีโอ อุปกรณ์ฉายแสง และวัตถุหรือนักแสดง ซึ่งเป็นการถ่ายทำโดยให้วัตถุหรือนักแสดง อยู่หน้าฉากซึ่งส่วนมากแล้วจะมี 2 รูปแบบ คือ ฉากแบบตัว U และฉากรูปตัว L แล้วใช้กล้องวีดีโอถ่ายทำ (real camera) หลังจากนั้นจะนำภาพที่ถ่ายนี้ไปทำการตกแต่งด้วยคอมพิวเตอร์กราฟฟิก โดยจะมีการสร้างภาพขึ้นมาแทนที่ส่วนที่เป็นฉากหลังแล้วนำมารวมกับภาพนักแสดงที่อยู่ด้านหน้าของฉากนั้น ซึ่งจะเรียกส่วนของ virtual object

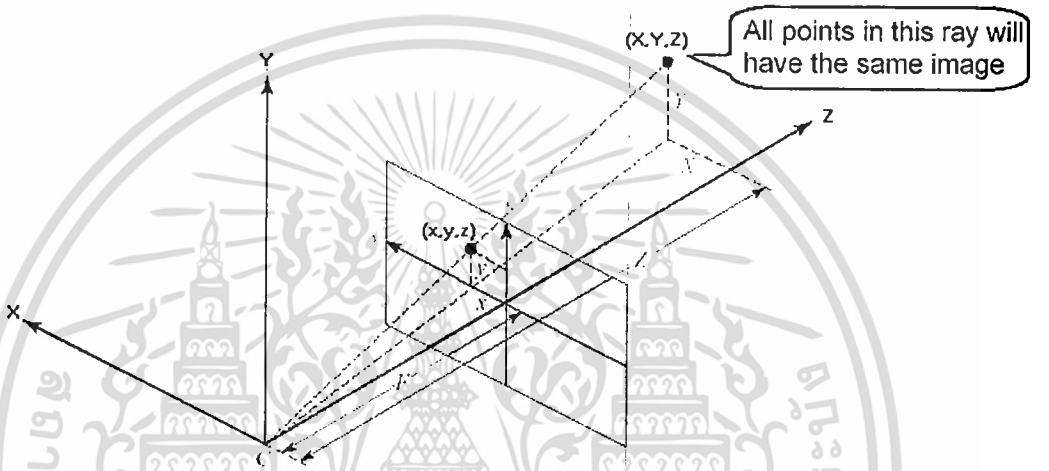
ภายหลังจึงมีการคิดค้นฉากหรือ blue screen ที่มีส่วนของการอ้างอิงตำแหน่งได้ เช่น มีจุดอ้างอิงตำแหน่ง หรือ สร้าง blue screen ที่มีลวดลาย (pattern)ขึ้น โดยหลังจากถ่ายทำแล้วก็จะแยกส่วนของภาพ blue screen นำไปทำการวิเคราะห์เพื่อให้ได้ค่าparameter ที่สามารถช่วยในการสร้างภาพด้วยคอมพิวเตอร์กราฟฟิกให้สัมพันธ์กับการเปลี่ยนแปลงตำแหน่งของกล้องมากขึ้นองการสร้างภาพนี้ว่า virtual camera



รูปที่ 2.1 แสดงขั้นตอนการทำงานของระบบ Virtual Studio

2.2 Perspective Projection

เมื่อกำหนดให้ระนาบของภาพที่ถ่าย(image plane)ขนานกับระนาบ XY โดยมีความลึกเป็น Z = f ซึ่ง f คือ ความยาวโฟกัส (focal length) และให้จุดเริ่มต้นของการถ่ายภาพอยู่ที่พิกัด (0,0,0) และ ฉากจะอยู่ในแนวขนานกับระนาบของภาพที่ถ่าย และกำหนดให้พิกัดเปลี่ยนแปลงตามการเปลี่ยนตำแหน่งของการจับภาพของกล้อง



รูปที่ 2.2 แสดง Perspective Projection

เมื่อกล้องทำการถ่ายภาพ จุดภาพทุกจุดที่กล้องจับภาพได้ที่จะถูกฉาย (projected) ลงบนกล้องได้สมการ projection model ดังนี้

$$x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z} \qquad (2.2.1)$$

โดย (X ,Y , Z) คือ ตำแหน่งโคออร์ดิเนตของจุดภาพที่ถ่าย และ (x , y , z) คือโคออร์ดิเนตของจุดภาพที่ถูก project ลงบนกล้อง

เมื่อกล้องมีการเปลี่ยนตำแหน่งหรือมีการหันเหไปจากเดิม จุดทุกจุดที่ถูก project ลงบนกล้องก็จะเปลี่ยนไปในแต่ละแกน การเปลี่ยนแปลงของจุดในรูปแบบ 3 มิติ จะทำให้ได้มาซึ่งการหมุนแบบ 3 มิติ (3-D rotation) และการเปลี่ยนแปลงตำแหน่งแบบ 3 มิติ (3-D translation) กำหนดให้ R เป็น rotation matrix ขนาด 3×3 โดยมี r_1 , r_2 , r_3 เป็นคอลัมน์เวคเตอร์ จะได้ $R = [r_1$

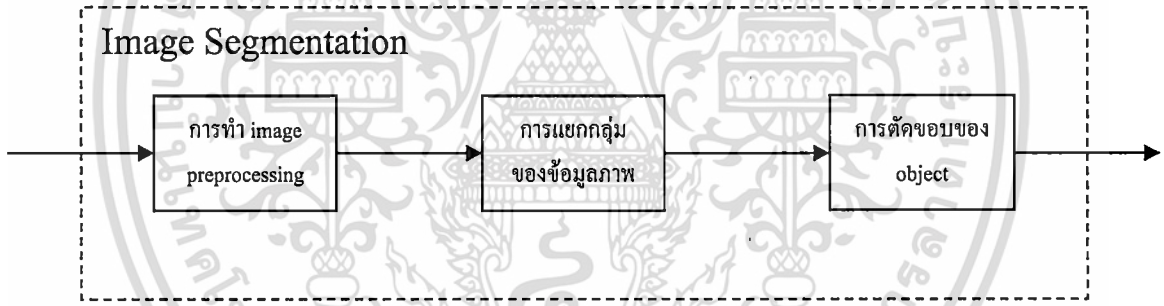
r_2 r_3] และ T เป็น translation vector ขนาด 3×1 ที่ซึ่งจะใช้ในการหาการเปลี่ยนแปลงตำแหน่งและการหันเหของกล้องแบบ 3 มิติ

ถ้ากำหนดให้ P แทน $[x\ y\ z]^T$ ซึ่งเป็นโคออร์ดิเนตของจุดภาพแล้วโคออร์ดิเนตใหม่ของจุดนั้นจะเท่ากับ

$$P' = P*R + T \qquad (2.2.2)$$

2.3 Image Segmentation

กระบวนการ segmentation เป็นขั้นตอนหนึ่งของการวิเคราะห์ภาพ(image analysis) เพื่อพยายามจะทำให้การวิเคราะห์ภาพมีความง่ายขึ้น โดยการแยก หรือ เน้น(highlight) ส่วนของภาพที่สำคัญที่เราสนใจออกมาจากภาพต้นฉบับ กลุ่มของข้อมูลภาพที่ได้มาจะต้อง ไม่สูญเสียรายละเอียดที่สำคัญของภาพไปเมื่อเทียบกับรูปต้นฉบับ โดยกระบวนการ segmentation สามารถแบ่งเป็นขั้นตอนหลักๆ ได้ 3 ขั้นตอน ดังนี้



รูปที่ 2.3 กระบวนการ image segmentation

2.3.1 การทำ image preprocessing

เป็นการแยกเอาส่วนข้อมูลที่เป็นส่วนที่ทำให้ภาพเกิดความไม่ชัดเจนออก ซึ่งเป็นส่วนของข้อมูลภาพที่ไม่ได้อยู่ในวัตถุประสงค์ที่ต้องการ อาจเป็นพวก nois ต่างๆ เช่น จุดสีดำหรือจุดสีขาวบนภาพซึ่งมีความไม่ชัดเจนว่าเป็นจุดภาพ(pixel)ในบริเวณนั้น วิธีการแยกอาจทำโดยใช้การเทียบข้อมูลของจุดภาพนั้นกับจุดภาพรอบๆ แล้วทำการวิเคราะห์ซึ่งอาจจะดูจากค่าเฉลี่ยของค่าสีรอบๆจุดนั้นแล้วทำการขจัด noise ออกโดยอาจจะแทนด้วยค่าสีที่สอดคล้องกับค่าสีที่อยู่รอบๆ จุดนั้น

2.3.2 การแยกกลุ่มของข้อมูลภาพ

เป็นการแยกกลุ่มของข้อมูลภาพ โดยกลุ่มของภาพที่แยกออกมานั้นจะแยกตามคุณลักษณะของภาพตามวัตถุประสงค์ที่ต้องการ คุณลักษณะส่วนใหญ่ที่ใช้ในการจำแนกกลุ่มของข้อมูลภาพอาจเป็นการใช้ค่าความสว่างและค่าสีของจุดภาพ ซึ่งเป็นลักษณะทั่วๆไปที่รูปภาพแต่ละรูปพึงจะมี

แต่ลักษณะนี้ไม่เหมาะกับการใช้แยกแยะวัตถุออกจากรูปแบบ binary เนื่องจากมีเพียงแค่ 2 ค่า คือ 1 กับ 0 ไม่มีแสงสว่างที่เห็นได้อย่างชัด ดังนั้นลักษณะนี้จึงเหมาะสำหรับรูปภาพแบบ gray scale ขึ้นไป ซึ่งจะมีความแตกต่างของสีและแสงที่พอจะแยกแยะได้

ในกรณีที่รูปเป็นแบบ gray scale นั้น histogram จะเป็นสิ่งที่เหมาะสมกับการใช้หาลักษณะที่สุดเพราะ histogram จะแสดงให้เห็นถึงค่าความสว่างเฉพาะของภาพ ซึ่งสามารถใช้เป็นเครื่องมือที่ช่วยในการแยกกลุ่มของข้อมูลภาพได้

ฮีสโทแกรม (Histogram) คือ กราฟที่ใช้ระบุกลุ่มของตัวเลขโดยที่ตัวเลขเหล่านั้นคือปริมาณของจุดภาพ(pixel)ที่มีค่าของสีต่างกันออกไป กราฟ histogram นั้นแนวแกน X จะแทนด้วยค่าของสีและแนวแกว Y จะแทนด้วยจำนวน pixel ของสีแต่ละสี

histogram สามารถใช้ระบุได้ว่าวัตถุชนิดใดมีความหนาแน่นของสีในช่วงใดมากหรือน้อย ซึ่งจะช่วยให้สามารถแยกวัตถุออกตามช่วงของสีได้ อีกทั้งยังใช้เป็นตัวระบุความชัดเจนของรูปได้อีกด้วย ถ้า histogram กราฟมีการกระจายในแนวแกน X น้อยก็จะมีความคมชัดต่ำ ในทางกลับกันถ้ามีการกระจายในแนวแกน X สูง ก็จะมีความคมชัดค่อนข้างสูง

ในกรณีที่รูปเป็นแบบ grey scale นั้นรูปจะมี histogram กราฟเพียงชุดเดียวแต่ถ้าเป็นรูปแบบ RGB แล้ว histogram กราฟจะมี 3 ชุดคือแต่ละชุดสำหรับแต่ละสีเพื่อที่จะสามารถบ่งบอกถึงความเข้มข้นของแต่ละสีได้

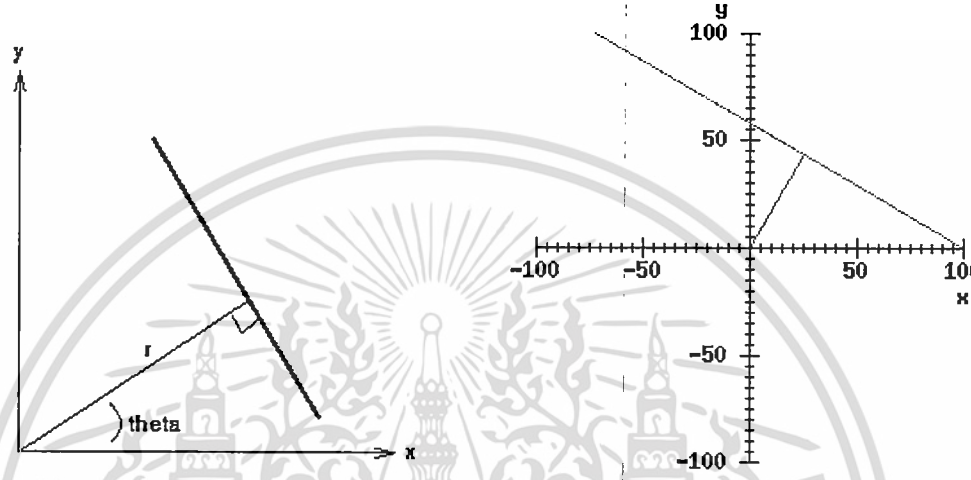
2.3.3 การตัดขอบของ object ของกลุ่มข้อมูลภาพ

จาก 2 ขั้นตอนที่ผ่านมาจะทำให้ได้กลุ่มของข้อมูลภาพที่ตรงตามวัตถุประสงค์ของการแยกส่วนออกมา ซึ่ง object ของกลุ่มของภาพที่ได้นั้น จะมีส่วนที่เป็นขอบที่เกินออกมาซึ่งเกิดจากการตัดแบบหยาบๆ ในขั้นตอนของการแยกกลุ่มของข้อมูลภาพ ซึ่งในขั้นตอนนี้จะเป็นการตัดขอบของ object ของกลุ่มข้อมูลภาพที่เราแยกออกมาเพื่อให้มีความเรียบเนียนมากขึ้น

2.4 Hough Transform

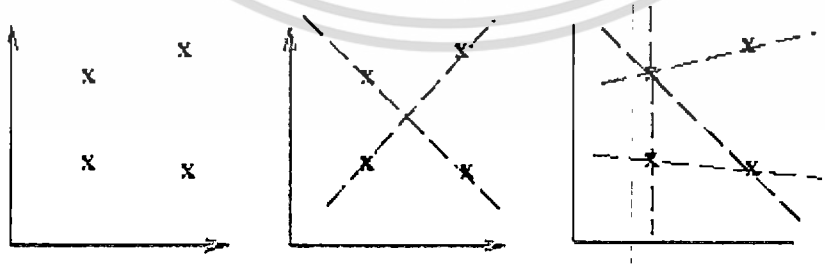
Hough Transform เป็นวิธีการเดียวกับ Radon Transform ซึ่งเป็นเทคนิคที่สำคัญของกระบวนการประมวลผลภาพที่นิยมใช้ในการหาแนวเส้นตรง (line detection) ที่ซ่อนอยู่ภายในภาพ โดยหลักการจะเป็นการนำภาพขาวดำ (binary image) ซึ่งเป็นภาพที่ผ่านกระบวนการหาขอบของภาพ (edge detection)แล้ว นำมาหาผลรวมของจำนวนจุดภาพ (pixel) ณ ตำแหน่งโคออดิเนต(x,y) ของภาพ ที่อยู่ในแนวของเส้นการสแกนที่ตั้งฉากกับเส้นที่ลากจากจุดศูนย์กลางของการสแกนที่มีรัศมีเท่ากับ r และทำมุม theta กับแกน x โดยจุดศูนย์กลางของการสแกนจะอยู่ที่จุดกลางของภาพ โดย

$$x\cos\theta + y\sin\theta = r \qquad (2.4.1)$$

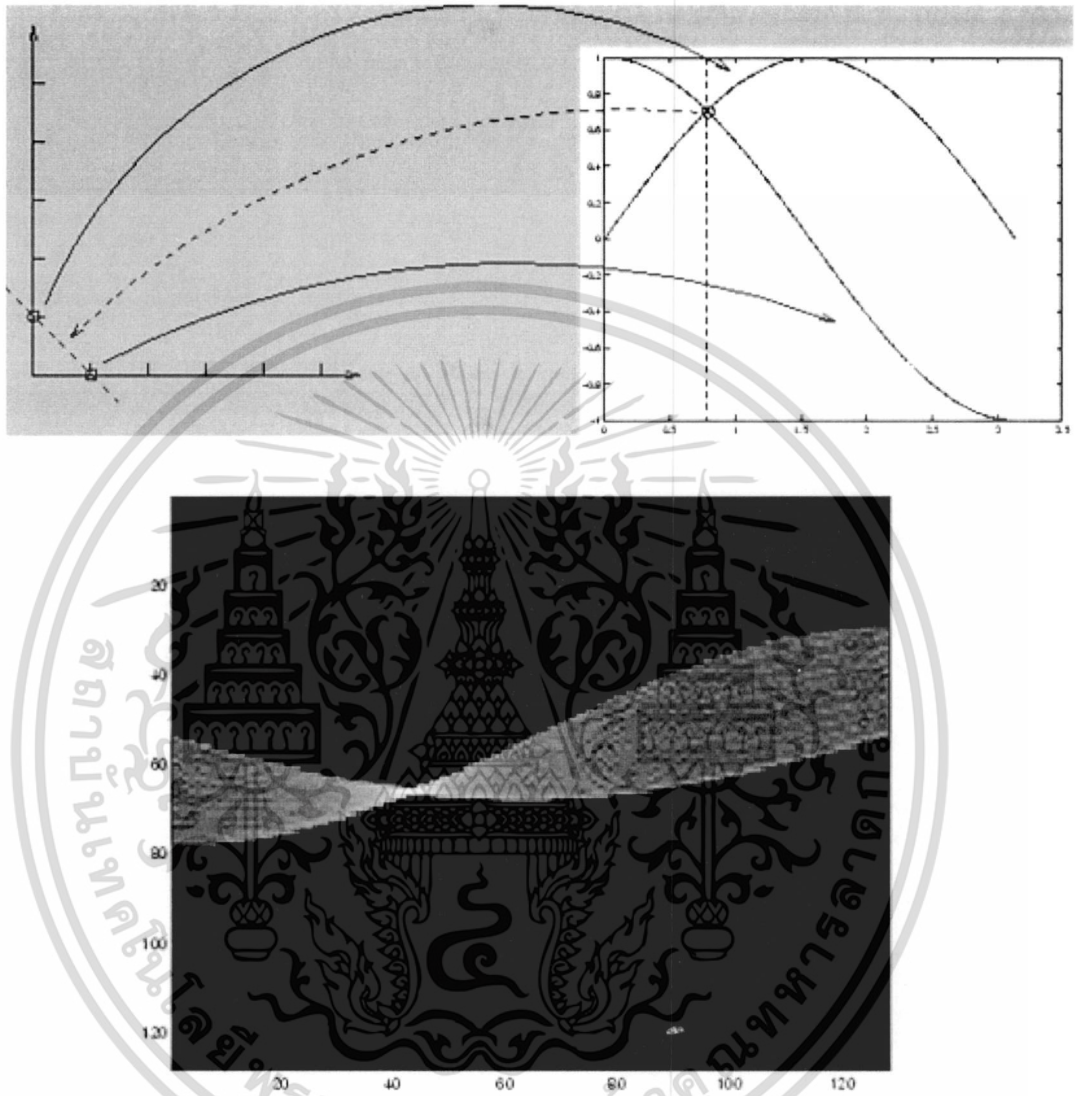


รูปที่ 2.4 แสดงเส้นแนวการสแกนที่มุมตั้งฉากกับเส้นตรงที่จุด(r,theta)

ในแต่ละแนวสแกนที่จุด(r,theta) จะให้ค่าซึ่งจะแสดงถึงความหนาแน่นของจุดภาพในแนวเส้นการสแกนนั้น โดยจะทำการสแกนเริ่มจาก (r,theta) ที่ (0,0) แล้วพิจารณาไปในทุกๆแนวสแกน ซึ่งค่าที่มีความหนาแน่นสูงจะมีความเป็นไปได้สูงที่จะเป็นแนวของเส้นตรงในภาพ



รูปที่ 2.5 แสดงความเป็นไปได้ที่จะเกิดเป็นเส้นตรงของแนวสแกนจุดภาพ

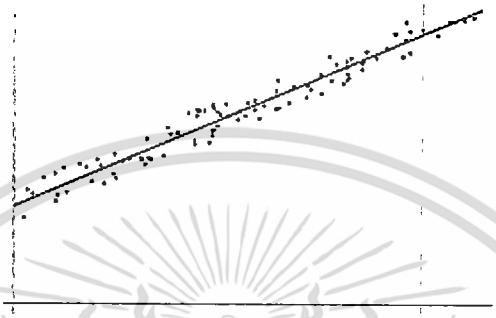


รูปที่ 2.6 กราฟของ Hough Transform แสดงค่าความหนาแน่นจุดภาพ

จากกราฟของ hough transform แสดงถึงค่าความหนาแน่นของจุดภาพในแนวสแกนต่างๆ โดยแกน x คือ ค่า theta แกน y คือรัศมี r ซึ่งจุดที่มีค่าความหนาแน่นของจุดภาพสูงจะเป็นจุดที่มีการตัดกันมากของเส้นกราฟที่ได้จากการสแกน หากมองจากกราฟจะเห็นว่าเป็นจุดที่มีความสว่างสูงอย่างเห็นได้ชัด ซึ่งจุดที่เป็นจุดตัดนั้นจะเป็นจุด(r,theta) ที่มีความเป็นไปได้ที่จะเกิดเป็นแนวเส้นตรงมากที่สุดในแนวสแกนนั้น

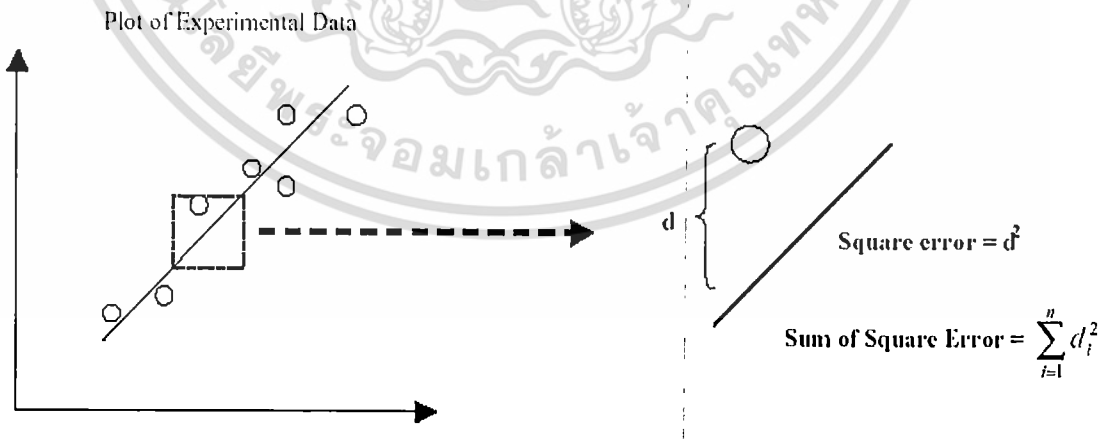
2.5 วิธีกำลังสองน้อยที่สุด (Least Squares method)

การประมาณค่าข้อมูลในแผนภาพที่มีข้อมูลกระจายอยู่ จะใช้การลากเส้นตรงเพื่อให้เส้นนั้นบอกแนวโน้มของกลุ่มข้อมูลที่กระจายกันอยู่ จะใช้สมการเส้นตรง $y = ax + b$ ในการประมาณค่าเพื่อให้ได้เส้นตรงที่เข้ากับข้อมูลได้ดีที่สุด



รูปที่ 2.7 เส้นตรงที่ใช้ประมาณค่าเพื่อดูแนวโน้มของกลุ่มข้อมูล

การหาค่า a และ b จากข้อมูล $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ มีวิธีการที่เป็นที่นิยมใช้คือ วิธีกำลังสองน้อยที่สุด (least squares method) วิธีนี้จะให้ค่าประมาณ a และ b ที่ทำให้ผลรวมของกำลังสองของความแตกต่างระหว่างค่าที่กะประมาณจากความสัมพันธ์ที่สร้างขึ้น กับค่าที่สังเกตได้ของข้อมูลทุก ๆ ค่า (Sum of Square Error) มีค่าน้อยที่สุด ถ้ากำหนดให้ d คือค่าของผลต่างของค่าประมาณกับค่าที่สังเกตได้ของข้อมูล ดังนั้น



รูปที่ 2.8 การหาค่า Sum of Square Error

จากสมการเส้นตรง

$$y_i = ax_i + b \ , (i = 1 \ldots n) \qquad (2.5.1)$$

จะได้สมการเส้นตรงที่ผ่านจุด(x_i,y_i) แต่ละจุดที่สังเกตได้ของข้อมูล สามารถหาค่า a,b ได้จาก

$$\text{ให้ } Y = \begin{bmatrix} y_1 \\ . \\ . \\ . \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} X_1 & 1 \\ . & . \\ . & . \\ . & . \\ X & 1 \end{bmatrix}, \quad C = \begin{bmatrix} a \\ b \end{bmatrix}, \quad E = \begin{bmatrix} d_1 \\ . \\ . \\ . \\ d_n \end{bmatrix}$$

$$Y = XC + E \qquad (2.5.2)$$

$$X^TY = X^TXC + X^TE \qquad (2.5.3)$$

$$(X^T X)^{-1} X^TY = IC + (X^T X)^{-1} X^TE \qquad (2.5.4)$$

จะได้ C ค่าที่ดีที่สุดเมื่อให้ E มีค่าน้อยที่สุด ดังนั้นให้E เป็น 0

ดังนั้น $C = (X^T X)^{-1} X^TY$ (2.5.5)

2.6 Rotation matrix

การหมุนของจุดภาพที่เกิดจากการถ่ายภาพจากกล้องที่หมุนไปในสามมิติรอบแกน X Y และ Z โดยเราสามารถแยกการหมุนในแต่ละแนวแกนแสดงโดยใช้ rotation matrix ดังนี้

การหมุนรอบแกน X

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \qquad (2.6.1)$$

การหมุนรอบแกน Y

$$R_y = \begin{bmatrix} \cos(\gamma) & 0 & -\sin(\gamma) \\ 0 & 1 & 0 \\ \sin(\gamma) & 0 & \cos(\gamma) \end{bmatrix} \qquad (2.6.2)$$

การหมุนรอบแกน Z

$$R_z = \begin{bmatrix} \cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.6.3)$$

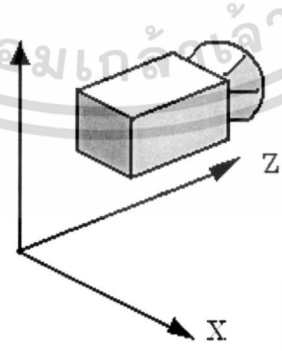
เหตุที่แยกออกมาเป็นสามส่วนนี้ก็เพื่อให้ง่ายต่อการทำความเข้าใจ และเห็น rotation matrix ของการหมุนรอบแต่ละแกน ส่วน rotation matrix ที่ทำการคำนวณออกมาได้นั้นจะเป็นเมทริกซ์ที่เกิดจากการรวมกันของ rotation matrix ของการหมุนรอบทั้ง 3 แนวแกน ซึ่ง R_{xyz} มาจาก

$$R_{xyz} = R_x \times R_y \times R_z$$

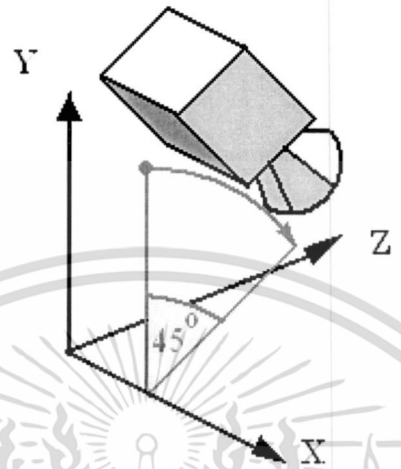
จะได้ rotation matrix

$$R_{xyz} = \begin{bmatrix} \cos(\gamma)\cos(\beta) & -\cos(\gamma)\sin(\beta) & -\sin(\gamma) \\ -\sin(\theta)\sin(\gamma)\cos(\beta) + \cos(\theta)\sin(\beta) & \sin(\theta)\sin(\gamma)\sin(\beta) + \cos(\theta)\cos(\beta) & -\sin(\theta)\cos(\gamma) \\ \cos(\theta)\sin(\gamma)\cos(\beta) + \sin(\theta)\sin(\beta) & -\cos(\theta)\sin(\gamma)\sin(\beta) + \sin(\theta)\cos(\beta) & \cos(\theta)\cos(\gamma) \end{bmatrix} \qquad (2.6.4)$$

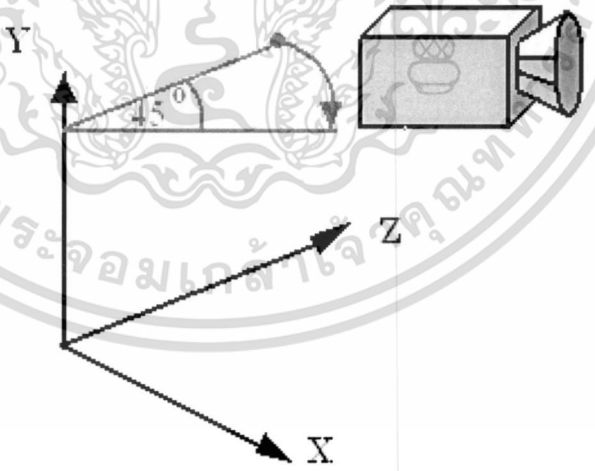
จาก rotation matrix R_{xyz} เราจะสามารถแก้สมการหาค่ามุม (, (, (ซึ่งเป็นมุมของการหมุนรอบแนวแกน X , Y และ Z ตามลำดับ ทำให้ทราบการหันเหไปของกล้องวีดีโอเมื่อเทียบกับแกนทั้ง 3



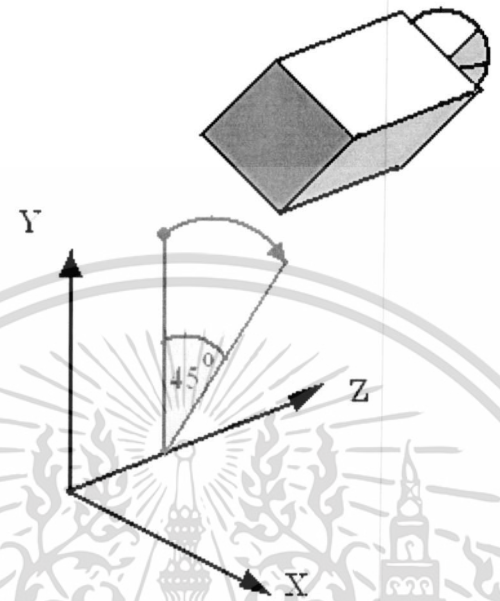
□รูปที่ 2.9 แสดงตำแหน่งของกล้องก่อนการหันเห



รูปที่ 2.10 กล้องหมุนรอบแกน X 45 องศา



รูปที่ 2.11 กล้องหมุนรอบแกน Y 45 องศา



รูปที่ 2.12 เมื่อหมุนรอบแกน Z 45 องศา

บทที่ 3
การพัฒนาอัลกอริทึ่มและทดสอบ

ในบทนี้จะกล่าวถึงการออกแบบและพัฒนาอัลกอริทึ่มรวมถึงการทดสอบ โดยเริ่มต้นจะต้องมีการออกแบบและสร้างฉากแบบ Binary Pattern ขึ้นมา หลังจากนั้นจะนำฉากนี้ไปใช้ในการถ่ายภาพด้วยกล้องวีดีโอแล้วนำภาพที่ได้นี้ ไปทำการวิเคราะห์ซึ่งจะเป็นข้อมูลภาพเบื้องต้นสำหรับใช้ในการดำเนินการ ซึ่งต้องมีการออกแบบวิธีการสำหรับการประมวลผลภาพเพื่อให้ได้อัลกอริทึ่มที่สามารถทำการคำนวณหาค่าตำแหน่งและมุมที่เปลี่ยนไปของกล้องจากภาพที่ถ่ายได้

3.1 การออกแบบฉาก binary pattern

ในการออกแบบจะเริ่มจากกำหนดขนาดของฉากที่จะใช้ จากนั้นเมื่อได้ความกว้างและความสูงของฉากแล้วก็ทำการแบ่งพื้นที่นี้ออกเป็นรูปสี่เหลี่ยมผืนผ้าที่มีขนาดเท่าๆกัน จำนวน N×M บล็อก ให้เหมาะสมกับขนาดของฉาก แล้วจะทำการสร้างเป็นลวดลาย(pattern) โดยในการศึกษาและทดลองนี้จะใช้ฉากสีน้ำเงิน(blue screen) ซึ่งจะมีลวดลายเป็นบล็อกสีน้ำเงินเข้มหรือสีน้ำเงินอ่อนสลับกันไป ซึ่งลวดลายของฉากจะมาจาก binary matrix ขนาด N×M ที่ทำการคำนวณได้

จาก binary matrix ที่สร้างเป็นฉาก binary pattern ขึ้นมา ทำให้ในแต่ละเฟรมที่กล้อง capture ภาพมานั้นจะเกิดเป็น submatrix ย่อย ๆ ของฉาก binary map ซึ่งขอบเขตของการ capture นั้น จะกำหนดไว้ว่าในแต่ละเฟรมจะต้องมีขนาดของ submatrix ปรากฏอยู่อย่างน้อยที่สุด n × m บล็อก ซึ่งส่วนของ pattern ใน submatrix ที่ได้จากการ capture นั้น จะปรากฏเพียงครั้งเดียวภายในฉากนี้เนื่องจาก binary pattern จะทำให้ แต่ละ submatrix เกิดความแตกต่างกัน (unique)

3.2 การสร้าง Binary matrix

จะใช้ทฤษฎี Galois field และคุณสมบัติของ primitive polynomial ในการสร้าง cyclic maximal length code สมมติให้ h เป็น minimal polynomial ของ primitive element ใน $GF(p^n)$ โดย p คือจำนวนเฉพาะ จะได้ cyclic code c ที่มีความยาว $p^n - 1$ ซึ่งเราจะเรียกว่า h เป็น primitive polynomial ของ degree n และเรียก cyclic code ที่ได้ว่า maximal length code หรือ maximal length sequence

สำหรับการสร้าง binary sequence จะกำหนดให้ $p \equiv 2$ จากที่กล่าวไปข้างต้นจะเขียนได้เป็น $GF(2^n)$ สำหรับทุกค่า ของ n และสามารถรับประกันได้ว่า binary pattern นั้น จะมีคุณสมบัติที่แสดงความแตกต่างกัน (uniqueness) ของแต่ละ $n \times 1$ pattern ภายใน sequence ขนาด $(2^n-1) \times 1$ ซึ่งทำให้เราทราบว่าภายใน 1 คอลัมน์ที่มีขนาด $(2^n-1) \times 1$ ของ binary matrix จะไม่มี $n \times 1$ pattern ที่ซ้ำกัน แต่ที่กล่าวมานี้เป็นเพียงแค่มุมมองแบบ 1 มิติเท่านั้น ในขั้นตอนต่อไปจะต้องสร้างเป็น binary matrix แบบ 2 มิติ ซึ่งจะต้องทำให้ ส่วนของ $n \times m$ pattern ที่อยู่ใน binary matrix ขนาด $N \times M$ นั้นมีความไม่ซ้ำกัน โดยต้องสร้าง maximal length sequence ขึ้นมาอีกหนึ่งชุด ซึ่ง sequence ชุดนี้จะไม่เป็น binary sequence เพราะจะเป็นตัวที่บอกถึงการเลื่อนตำแหน่งไปของ binary sequence ซึ่งอธิบายได้ ดังนี้

กำหนดให้ c เป็น maximal length sequence (binary sequence) ที่มีความยาว $2^n - 1$ และ ให้ c_r และ c_s เป็นจำนวนของการเลื่อนตำแหน่งของ c (shift versions of c) ที่ซึ่ง $0 \leq s,r \leq 2^n - 2$ และกำหนดให้ $B \equiv [b_{ij}]$ และให้ b_j เป็นคอลัมน์ที่ j ของ binary matrix B สำหรับแต่ละคู่ของคอลัมน์ของ B ให้ $b_j \equiv c_r$ และ $b_{j+1} \equiv c_s$ และ $d_j \equiv s - r$ ซึ่งจะหมายถึง ผลต่างของการเลื่อนตำแหน่งของคอลัมน์ที่ j กับ j+1 ดังนั้น

- สอง submatrices ของ B ที่มี b_{i1j1} และ b_{i2j2} เป็นอิลิเมนต์ตัวแรกของแต่ละ submatrix ตามลำดับจะแตกต่างกัน ถ้า $d_{j1} \neq d_{j2}$
- ถ้าให้ d_j นิยามว่าเป็น $[d_j\ d_{j+1} \ldots d_{j+m-2}]$ ซึ่งคือ เวคเตอร์ที่บอก shift difference ของ m คอลัมน์ที่ติดกันแล้ว สมาชิกของ d_j สามารถได้มาจาก maximal length sequence ของ primitive polynomial ใน $GF((2^n-1)^{m-1})$

ขั้นตอนในการสร้าง binary map สามารถสรุปอัลกอริทึ่มได้ดังนี้

Algorithm

1.) กำหนดขนาดของ n*m submatrix ซึ่งเป็นขนาดที่เล็กที่สุดที่กล้องจับภาพได้

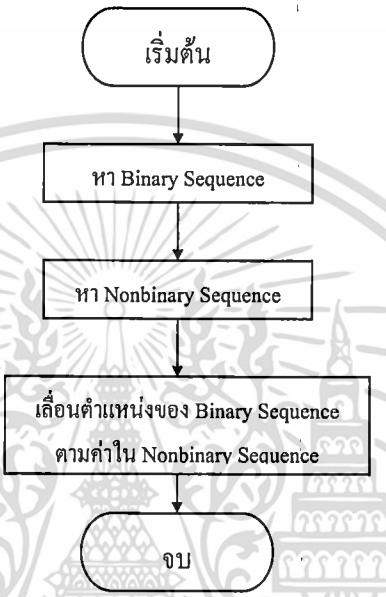
2.) หา primitive polynomial ของ degree n ใน $GF(2^n)$ และทำการคำนวณหา maximal length sequence

3.) กำหนดให้ maximal length sequence ที่ได้จากข้อ 2 ซึ่งจะเป็น binary sequence ให้เป็นคอลัมน์แรกสำหรับการเลื่อนตำแหน่ง (shift)

4.) หา primitive polynomial ของ degree $2^n - 1$ ใน $GF((2^n-1)^{m-1})$ และทำการคำนวณหา maximal length sequence ซึ่ง nonbinary sequence ที่ได้นี้จะเป็น ผลต่าง

ของการเลื่อนตำแหน่งของคอลัมน์ที่ติดกันของ binary sequence เรียกว่า "shift difference sequence"

5.) เริ่มทำการเลื่อนตำแหน่งของ binary sequence จากคอลัมน์แรก ตามค่าที่ได้จาก shift difference sequence จะได้เป็น binary map



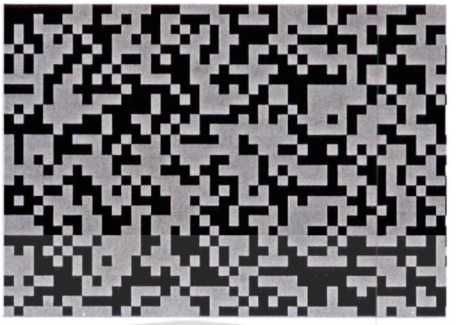
รูปที่ 3.1 ขั้นตอนการสร้าง Binary Map

นำ binary map ที่ได้มาสร้างเป็น blue screen โดยทำการกำหนดขนาดของความกว้าง และความสูงของแต่ละช่อง ของ pattern และให้ ค่าที่ได้จาก binary map ค่า 0 แทน สีน้ำเงินอ่อน และค่า 1 จะแทนด้วยสีน้ำเงินเข้ม

เพื่อที่จะทำให้เข้าใจมากยิ่งขึ้น ในตารางที่ 3.1 แสดงถึงการเลื่อนตำแหน่งของ sequence ตาม shift difference sequence ตามลำดับ และตัวอย่างที่ใช้ในตารางสมมติให้ n = 3 ดังนั้น sequence ที่จะนำมา shift จะมีความยาวเท่ากับ $2^n - 1 = 2^3 - 1 = 7$ สำหรับการสร้าง shift difference sequence และ และการ shift sequence นั้นจะใช้การ modulo-$(2^n - 1)$ ดังนั้นค่าที่ได้จะมีค่าไม่เกินความยาวของ shift sequence และการ shift เมื่อครบรอบ ก็จะวนกลับมาเริ่มที่ตัวแรกใหม่

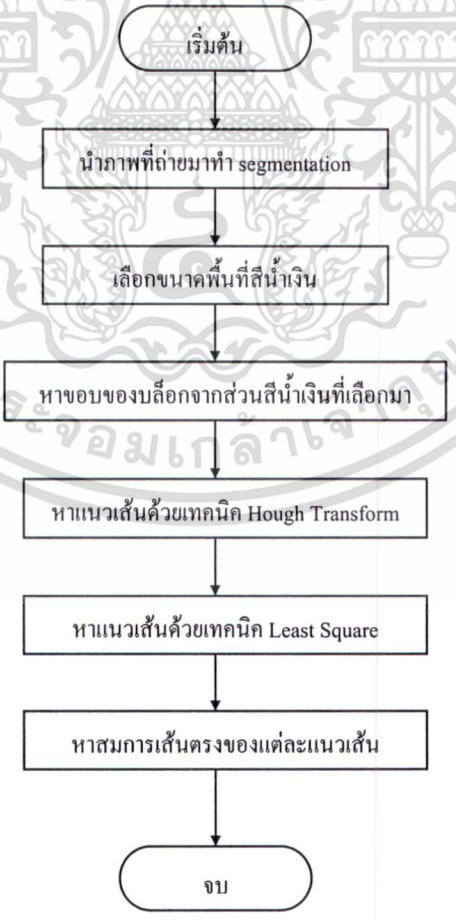
d_j		0	1	1	2	3	...
s	1	1	2	3	5	1	...

ตารางที่ 3.1 แถวบน คือ Shift Differencees Sequence แถวล่าง คือ Shift Sequence



รูปที่ 3.2 ภาพตัวอย่างของ binary pattern ที่ออกแบบขึ้นมา

จากหลักการที่ได้กล่าวไปนี้ ทำให้เราได้วิธีที่สามารถสร้าง blue screen binary pattern ที่มี pattern ที่ไม่ซ้ำกันตามขนาดที่ได้กำหนดไว้ เพื่อนำไปใช้ในการศึกษาและทดลองได้ โดยขั้นตอนการดำเนินการมีดังนี้



รูปที่ 3.3 ขั้นตอนการหาค่าพารามิเตอร์จาก Blue Screen Binary Map

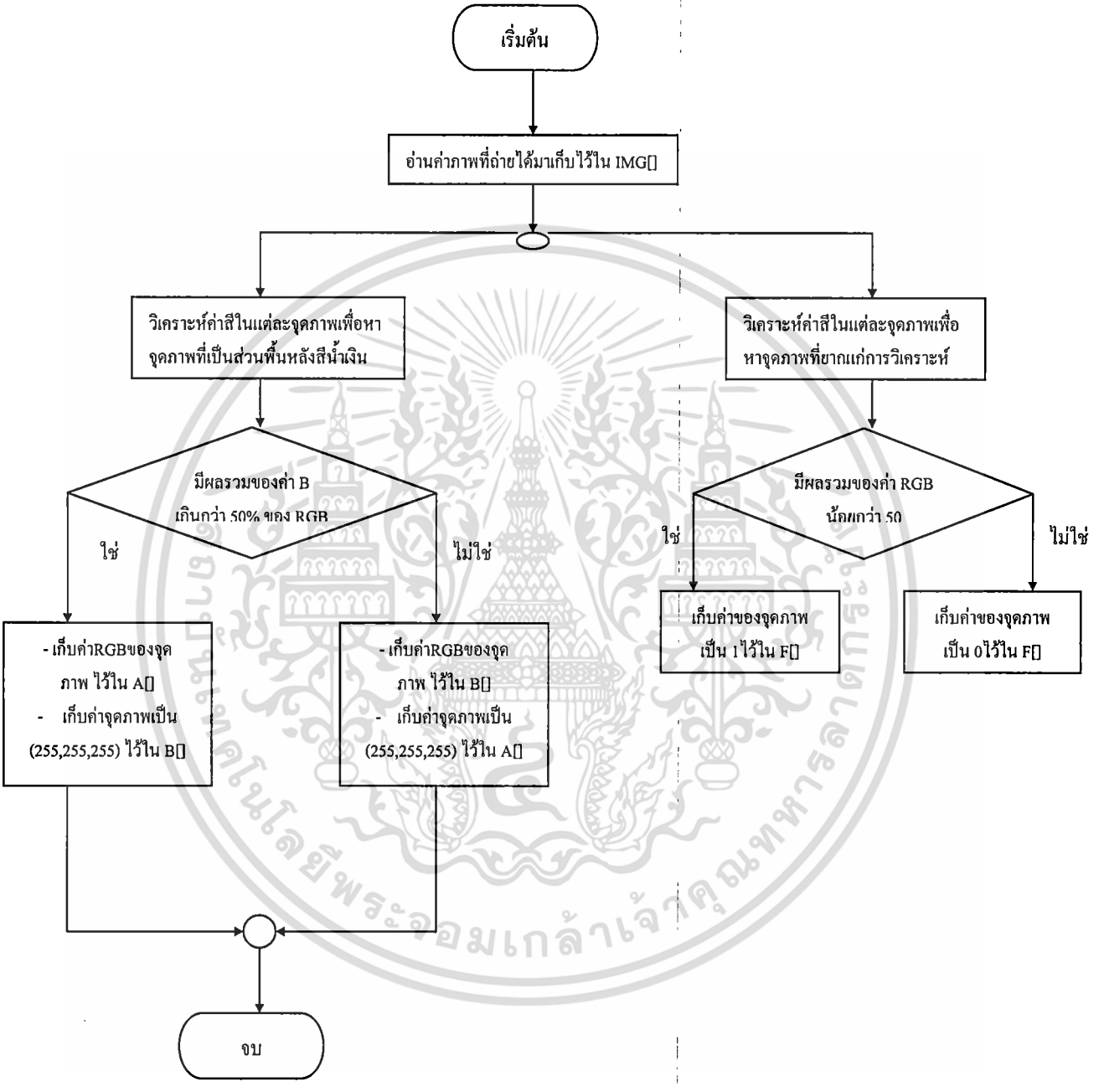
3.3 การทำ segmentation

การทำ segmentation เป็นการแยกเอาส่วนที่เราสนใจออกจากภาพ หรืออาจกล่าวได้ว่าเป็นการวิเคราะห์และประมวลผลภาพเพื่อทำการแยกภาพพื้นหลังกับภาพวัตถุออกจากกัน เมื่อได้ภาพที่ถ่ายด้วยกล้องวีดีโอที่ถ่ายภาพคนหรือวัตถุที่มีพื้นหลังเป็นฉาก binary pattern จะใช้วิธีการแยกส่วนของภาพโดยดูจากอัตราส่วนของค่า RGB ซึ่ง ได้แก่ ค่าสีแดง (R) ค่าสีเขียว (G) และค่าสีน้ำเงิน (B) ของแต่ละจุดภาพ แล้วทำการจัดกลุ่มของข้อมูลแยกภาพส่วนหน้า (foreground image) และภาพส่วนหลัง(background image) ออกจากกัน เพื่อที่จะได้นำข้อมูลภาพส่วนของ background ไปทำการวิเคราะห์หาค่าตำแหน่งและมุมที่เปลี่ยนไป ในขั้นตอนต่อๆไป โดยอัลกอริทึ่มในการแยกภาพทั้งสองส่วนออกจากกันมีดังนี้

Algorithm

1) อ่านค่าภาพที่ถ่ายได้มาเก็บไว้ใน IMG[]
2) แยกเก็บภาพส่วนที่มีความสว่างของภาพต่ำ(เงามืด) ซึ่งจะยากต่อการวิเคราะห์อัตราส่วนของค่าสีออกจากภาพ โดยให้จุดภาพที่มีผลรวมของค่า RGB น้อยมาก กำหนดให้ถ้าน้อยกว่า 50 ให้เก็บค่าเป็น 1 และ จุดภาพอื่นๆ มีค่าเป็น 0 เก็บไว้ใน F[]
3) หาอัตราส่วนของค่าสีน้ำเงิน(B) เทียบเป็นร้อยละกับค่า RGB ของแต่ละจุดภาพใน IMG[]
4) หากมีค่าของสีน้ำเงินเกินกว่าร้อยละ 50 แสดงว่าเป็นส่วนของฉากหลังสีน้ำงิน (blue screen) ให้ A[] เก็บค่า RGB ในส่วนของพื้นหลัง และให้ส่วนที่เป็นวัตถุ (foreground) เป็นสีขาวโดยเก็บค่า RGB เป็น (255:255:255)
5) หากมีค่าของสีน้ำเงินน้อยกว่าร้อยละ 50 แสดงว่าเป็นส่วนของวัตถุ ให้ B[] เก็บค่า RGB ของส่วนที่เป็นวัตถุไว้และเก็บค่า RGB ของส่วนที่เป็นพื้นหลังให้เป็นสีขาว คือ (255:255:255)

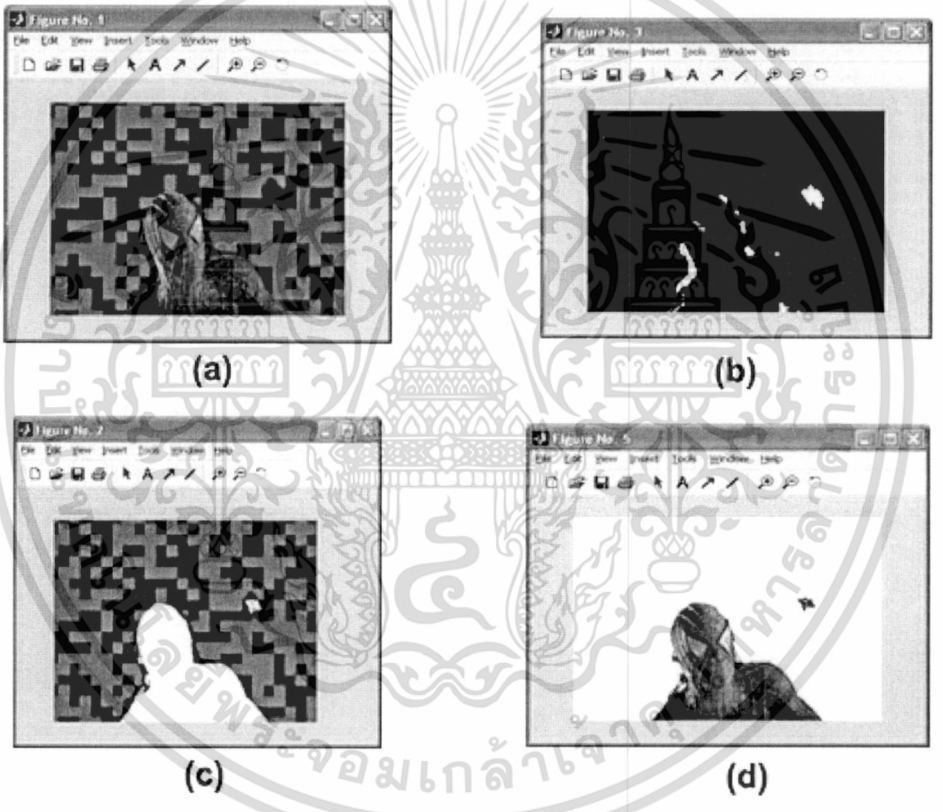
โดย A[] ที่เก็บภาพส่วนของพื้นหลัง จะถูกนำไปใช้ในการวิเคราะห์หาการหันเหของกล้องวีดีโอในขั้นตอนต่อไป ส่วน B[] ที่เก็บภาพวัตถุ และ F[] ที่เก็บส่วนที่เป็นจุดที่มี threshold ต่ำ จะเอาไว้ใช้ในขั้นตอนของการนำภาพวัตถุกับภาพพื้นหลังที่สร้างขึ้นให้มีความเหมาะสมกับตำแหน่งการหันเหไปของกล้องมารวมกัน



รูปที่ 3.4 ขั้นตอนการทำ segmentation

ผลการทดสอบ

จากขั้นตอนที่ผ่านมาจะสามารถแยกส่วนของภาพออกได้เป็น 3 ส่วน คือ ส่วนที่เป็นวัตถุ , ส่วนที่เป็นพื้นหลัง และส่วนที่เป็นส่วนเงามืดที่ยากต่อการแยกแยะว่าเป็นส่วนวัตถุหรือส่วนพื้นหลัง จะสังเกตได้ว่าวัตถุที่อยู่หน้าฉากสีน้ำเงินนั้นควรจะเป็นวัตถุที่มีอัตราส่วนของสีน้ำเงินไม่มาก เพราะจะไม่สามารถแยกได้ว่าเป็นส่วนพื้นหลังหรือส่วนวัตถุ ส่วนภาพที่จะนำไปใช้ในการวิเคราะห์ในขั้นตอนต่อไป คือ ส่วนภาพของพื้นหลังที่เก็บไว้ใน A[]



รูปที่ 3.5 (a) ภาพต้นฉบับ (b) ส่วนของภาพที่ยากต่อการวิเคราห์
(c) ส่วนของภาพ background (d) ส่วนของภาพ foreground

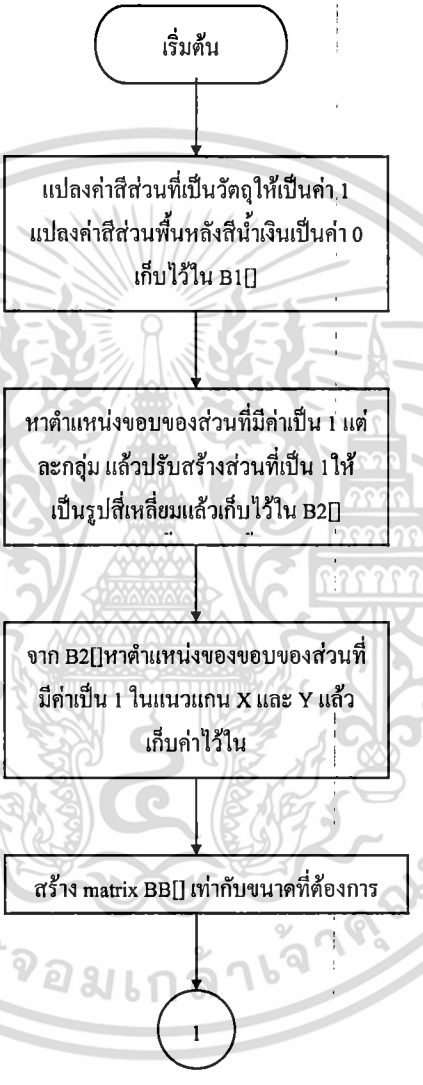
3.4 การเลือกพื้นที่สีน้ำเงินจากส่วนของพื้นหลัง

จากภาพที่ทำการ segmentation แล้ว จะได้ส่วนของภาพ foreground และภาพ background แยกออกจากกันแล้ว สมมติให้เก็บไว้ใน I[] ในขั้นตอนนี้จะนำส่วนของ I[] นี้ มาทำการเลือกส่วนพื้นที่สีน้ำเงินที่ใหญ่พอที่จะใช้ทำการคำนวณได้ โดยจะทำการกำหนดขนาดของพื้นที่สีน้ำเงินที่จะนำไปใช้ไว้ก่อน แล้วทำตามขั้นตอน ดังนี้

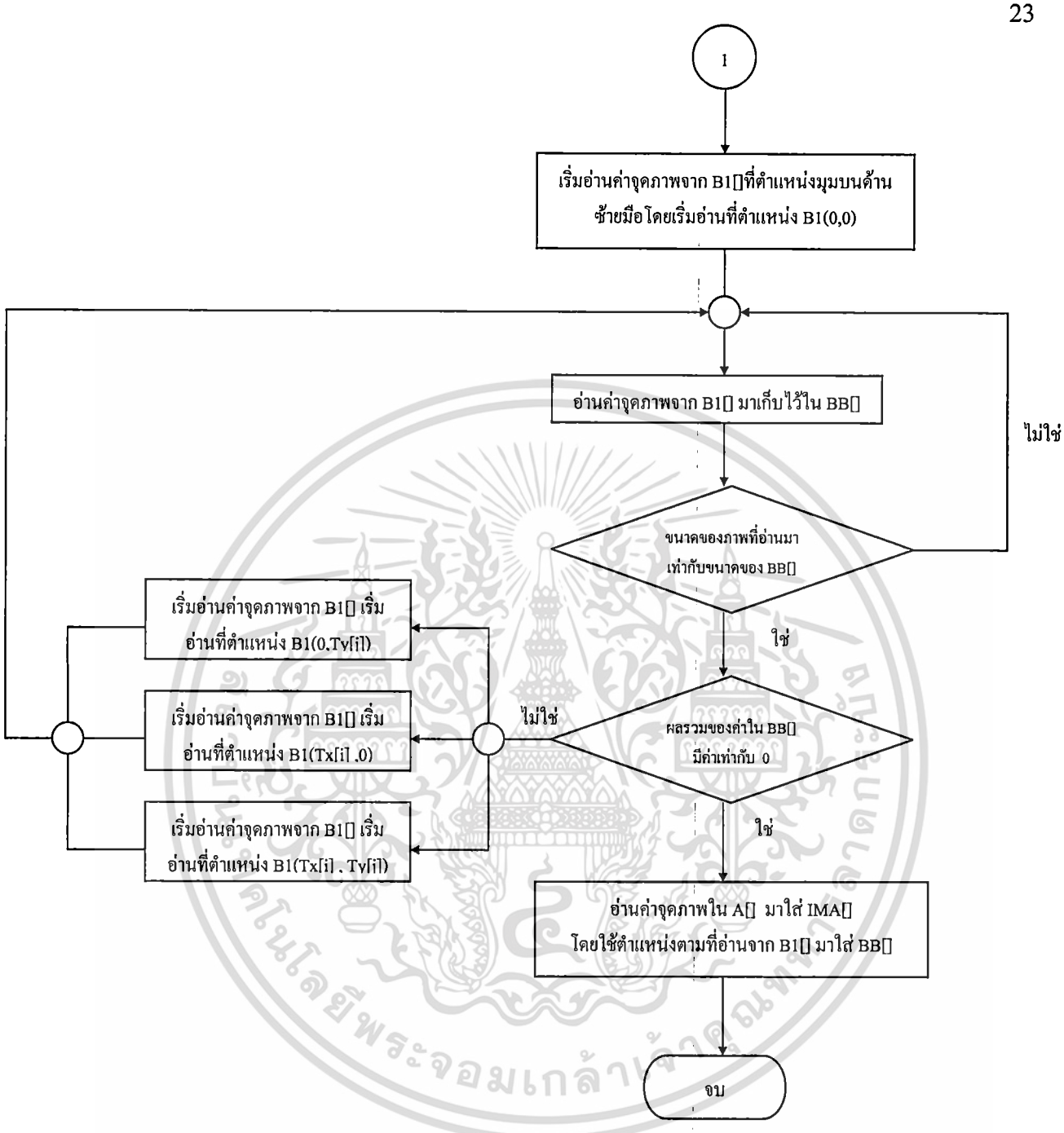
Algorithm

1) แปลงภาพแบบ RGB ให้เป็นภาพขาวดำ โดยให้ส่วนที่เป็น foreground เป็นสีขาวให้เก็บค่าสีเป็น 1 แล้วให้ส่วนที่เป็น background สีน้ำเงินทั้งหมดเป็นสีดำเก็บค่าสีเป็น 0 เก็บภาพนี้ไว้ใน B1[]
2) จาก B1[] หาตำแหน่ง (x_{min} , x_{max}) และตำแหน่ง (y_{min} , y_{max}) ของขอบกลุ่มของจุดภาพที่เป็นสีขาวแต่ละกลุ่ม (ส่วนที่มีค่าเป็น 1) แล้วนำมาปรับสร้างกลุ่มของจุดภาพสีขาวแต่ละกลุ่มให้เป็นรูปสี่เหลี่ยม จากนั้นเก็บภาพนี้ไว้ใน B2[]
3) จาก B2[] จะได้ตำแหน่งของขอบของบล็อกในแนวตั้งเก็บไว้ใน Tx[] และแนวนอนเก็บไว้ใน Ty[] จะใช้ค่าใน Tx[] และ Ty[] เป็นแนวในการหาพื้นที่สีน้ำเงินที่ต้องการ
4) สร้าง matrix BB[] ขนาดเท่ากับขนาดพื้นที่สีน้ำเงินที่ต้องการ
5) อ่านค่าจุดภาพจาก B1[] เริ่มจากตำแหน่ง B1(0,0) ซึ่งเป็นจุดมุมบนด้านซ้าย แล้วเอาค่าในจุดภาพไปเก็บไว้ใน BB[]
6) เก็บค่าจุดภาพจาก B1[] ไปไว้ใน BB[] จนเท่ากับขนาดของ BB[]
7) ตรวจสอบผลรวมของค่าสีใน BB[] ถ้ามีค่าเป็น 0 แสดงว่าได้ส่วนสีน้ำเงินขนาดที่ต้องการแล้ว
8) ถ้าผลรวมของค่าสีใน BB[] ไม่เป็น 0 ให้ย้อนกลับไปทำที่ข้อ 5 โดยให้จุดเริ่มต้นของการอ่านภาพเริ่มจากตำแหน่งตามค่าใน Tx[] ซึ่งจะเริ่มอ่านในแนวนอนก่อน โดยเริ่มที่ตำแหน่ง B1(Tx[i] , 0) (i คือตำแหน่งของขอบของบล็อกมีค่าตั้งแต่ 1 ถึง n) ไล่ไปจนกว่าจะพบว่าผลรวมของค่าสีใน BB[] เป็น 0 จึงหยุด
9) ถ้ายังไม่พบส่วนที่มีผลรวมของค่าสีใน BB[] เป็น 0 อีกให้อ่านในแนวตั้ง โดยเริ่มที่ตำแหน่ง B1(0 , Ty[j]) (j คือตำแหน่งของขอบของบล็อกมีค่าตั้งแต่ 1 ถึง m) ไล่ไปจนกว่าจะพบว่าผลรวมของค่าสีใน B[] เป็น 0 จึงหยุด
10) ถ้ายังไม่พบส่วนที่มีผลรวมของค่าสีใน BB[] เป็น 0 อีกให้อ่านในแนวตั้ง โดยเริ่มที่ตำแหน่ง B1(Tx[i] , Ty[j]) (i , j คือตำแหน่งของขอบของบล็อกมีค่าตั้งแต่ 1 ถึง n และ1 ถึง m ตามลำดับ) ไล่ไปจนกว่าจะพบว่าผลรวมของค่าสีใน B[] เป็น 0 จึงหยุด

จากนั้นให้เก็บตำแหน่งของจุดภาพใน B1[] ที่ตรงกับส่วนของจุดภาพที่อยู่ในกรอบสี่เหลี่ยมใน BB[] มาเก็บไว้ใน matrix ใหม่ กำหนดให้เป็น matrix IMA[] เพื่อเก็บส่วนพื้นหลังสีน้ำเงินที่จะนำไปใช้ในขั้นตอนต่อไป



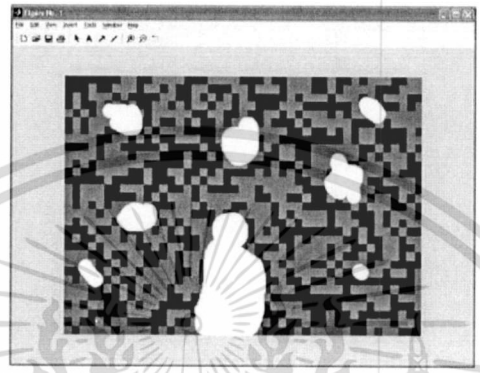
รูปที่ 3.6 ขั้นตอนการเลือกพื้นที่สีน้ำเงินจากส่วนพื้นหลัง (ส่วนที่ 1)



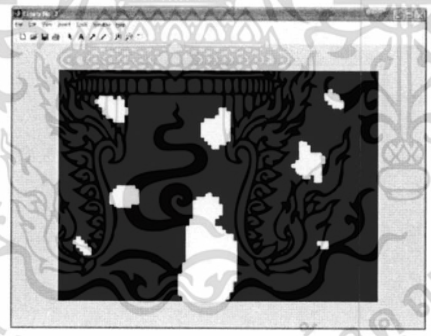
รูปที่ 3.7 ขั้นตอนการเลือกพื้นที่สีน้ำเงินจากส่วนพื้นหลัง (ส่วนที่ 2)

ผลการทดสอบ

ผลจากการแปลงภาพที่ได้จากขั้นตอนการ segmentation เป็นภาพขาวดำ โดยภาพที่ผ่านขั้นตอนการ segmentation จะเก็บไว้ใน A[] และภาพที่แปลงเป็นภาพขาวดำนั้นจะเก็บไว้ใน B1[]

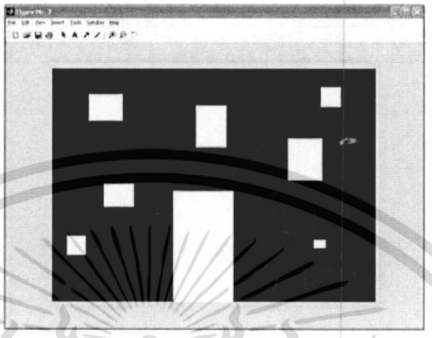


รูปที่ 3.8 ภาพที่ได้จากขั้นตอนการ segmentation



รูปที่ 3.9 เมื่อแปลงภาพให้เป็นภาพขาวดำ

จาก B1[] เมื่อหาตำแหน่ง จุดมุม(X_{min} , Y_{min}) และ (X_{max} , Y_{max}) ด้านของกลุ่มที่เป็นจุดสีขาวแต่ละกลุ่มแล้ว ก็จะมาทำการสร้างเป็นกรอบสี่เหลี่ยม แสดงได้ดังรูปที่ 3.10 แล้วจะเก็บภาพนี้ไว้ใน B2[] เพื่อนำไปใช้สำหรับการหาตำแหน่งของแนวการหาส่วนสีน้ำเงินที่ต้องการ

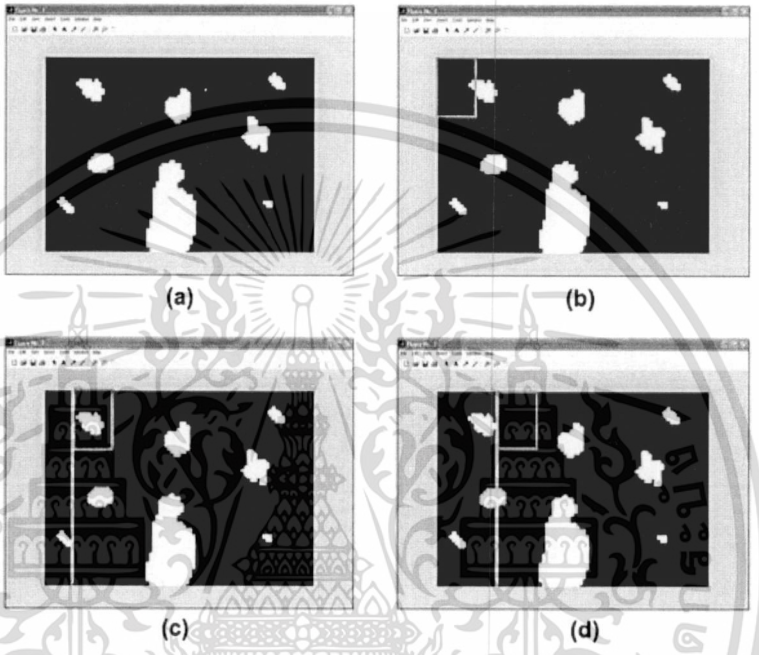


รูปที่ 3.10 หลังจากการสร้างส่วนที่เป็นสีขาวให้เป็นรูปสี่เหลี่ยม

ผลของการหาแนวของการหาส่วนสีน้ำเงินที่ต้องการ ซึ่งจะเก็บค่าตำแหน่งของเส้นแนวแกน X ไว้ใน Tx[] และในแนวแกน Y ไว้ใน Ty[]

รูปที่ 3.11 แนวการหาส่วนสีน้ำเงิน (a) ในแนวแกน X (b) ในแนวแกน Y

สำหรับการไล่หาพื้นที่ส่วนสีน้ำเงินตามขนาดที่ต้องการนั้น จะต้องมีการอ่านค่าจาก B1[] มาใส่ไว้ใน BB[] ดังที่ได้อธิบายไว้ในอัลกอริทึ่ม เพื่อนำมาหาผลรวมค่าจุดภาพใน BB[] ที่มีค่าเป็น 0 ซึ่งก็คือส่วนที่เป็นภาพสีดำทั้งหมด จากนั้นก็จะใช้ตำแหน่งภาพที่พบว่าเป็นส่วนสีดำนี้ นำไปอ่านจุดภาพในภาพต้นฉบับ A[] ซึ่งจะแสดงไว้ในรูปที่ 3.12 และ รูปที่ 3.13



รูปที่ 3.12 (a) ภาพจาก B1[] (b) การหาส่วนสีน้ำเงินครั้งที่ 1 (c) การหาส่วนสีน้ำเงินครั้งที่ 2 (d) การหาส่วนสีน้ำเงินครั้งที่ 3

รูปที่ 3.13 (a) พบส่วนพื้นหลังที่ต้องการใน B1[] (b) ภาพที่ได้จาก A[] ตัดมาใส่ใน IMA[]

3.5 การหาขอบของบล็อกภายในส่วนพื้นหลัง

หลังจากได้ส่วนของภาพพื้นหลังสีน้ำเงินที่เลือกมาแล้วก็จะทำการ หาแนวของเส้นขอบของบล็อกของ pattern ของภาพฉากที่เรียงกันอยู่ซึ่งจะเปรียบเสมือนเป็นเส้นตาราง(grid line) โดยส่วนของเส้นเหล่านี้จะมีความสำคัญมากสำหรับการวิเคราะห์ภาพเพื่อหาค่าตำแหน่งและมุมที่เปลี่ยนไปของกล้อง เนื่องจากการวิเคาระห์หาตำแหน่งการหันเหจะต้องดูจากการเบี่ยงเบนไปของเส้นของตาราง ขั้นตอนการหาของขอบบล็อกซึ่งเป็นขั้นตอนเริ่มต้นของการหาการเบี่ยงเบนของเส้นตารางมีขั้นตอนดังนี้

- ทำการแปลงภาพที่ได้มาเป็น gray scale
- ทำการหาขอบของบล็อกด้วยกระบวนการ edge detection
- เมื่อผ่านขั้นตอนของ edge detection แล้ว จะได้ภาพขาวดำ(binary image) ซึ่งจะมีเส้นที่เป็นขอบของบล็อกซึ่งแนวเส้นขอบของบล็อกแต่ละเส้นจะไม่เชื่อมต่อกันเนื่องจากมีส่วนของ binary pattern ที่สลับกันอยู่

รูปที่ 3.14 (a) ภาพพื้นหลังสีน้ำเงิน (b) แปลงเป็น gray scale (c) เมื่อผ่าน edge detection

3.6 การหาแนวเส้นตารางโดยใช้ Hough Transform

ในการหาแนวเส้นตรงจะนำภาพที่ผ่านขั้นตอนของการหาขอบภาพของบล็อก (edge detection) นำมาหาแนวของเส้นที่ไม่ต่อเนื่องกันที่อยู่ในภาพดังรูปที่ 3.4 เพื่อดูว่ากลุ่มของเส้นขอบของบล็อกใดเป็นแนวเส้นเดียวกันบ้าง เพื่อใช้ในการวิเคราะห์การเปลี่ยนแปลงไปของแนวเส้นขอบของบล็อกในขั้นตอนต่อๆไป โดยในการหาแนวของเส้นตรงนั้นจะนำเอาเทคนิคของ Hough Transform หรือเรียกอีกอย่างว่าเทคนิค Radon Transform เข้ามาช่วยในการหาแนวเส้นที่ซ่อนอยู่ในภาพ

Algorithm

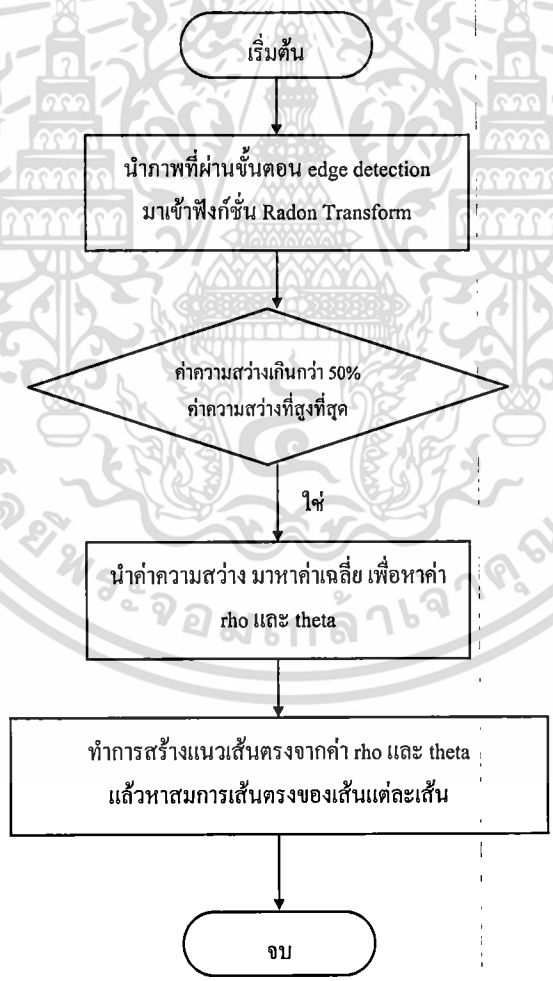
1.) นำภาพที่เป็น gray scale ที่ผ่านขั้นตอนการหาขอบของบล็อกแล้ว มาเข้าฟังก์ชันของ Radon Transform ในโปรแกรม Matlab โดยกำหนดให้รัศมีของการสแกนเริ่มตั้งแต่ 0 องศา จนถึง 179 องศา

2.) จากค่าความสว่างของแต่ละแนวสแกนให้เลือกเฉพาะส่วนที่มีค่าความสว่างเกินกว่าร้อยละ 50 ของค่าความสว่างที่สูงที่สุด

3.) จากแต่ละกลุ่มของค่าความสว่างที่ได้นำมาหาค่าเฉลี่ยโดยวิธีการหามวลรวม จะได้ค่ารัศมีของการสแกน (rho) และองศาสแกน (theta) ของแต่ละกลุ่มของค่าความสว่าง

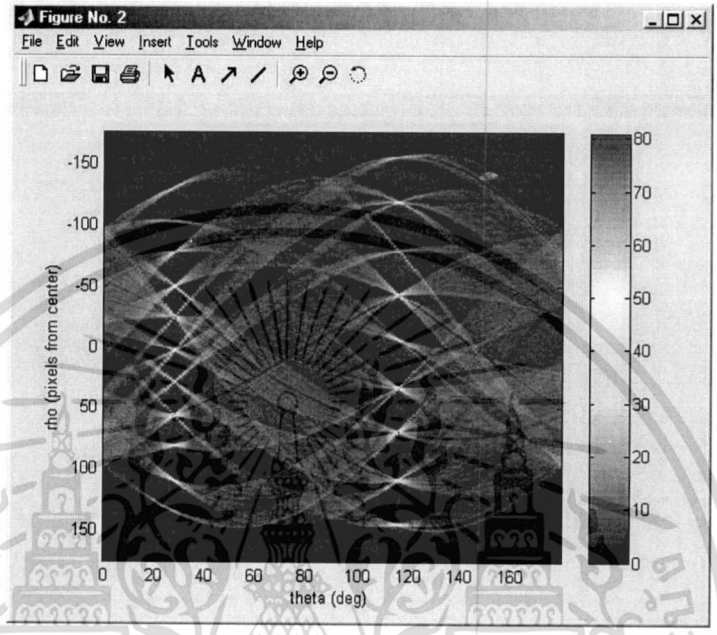
4.) นำค่า rho และ theta ที่ได้มาทำการสร้างแนวเส้นตรง

5.) หาค่าความชันของเส้นตรงแต่ละเส้นโดยใช้สมการที่ 2.5.5



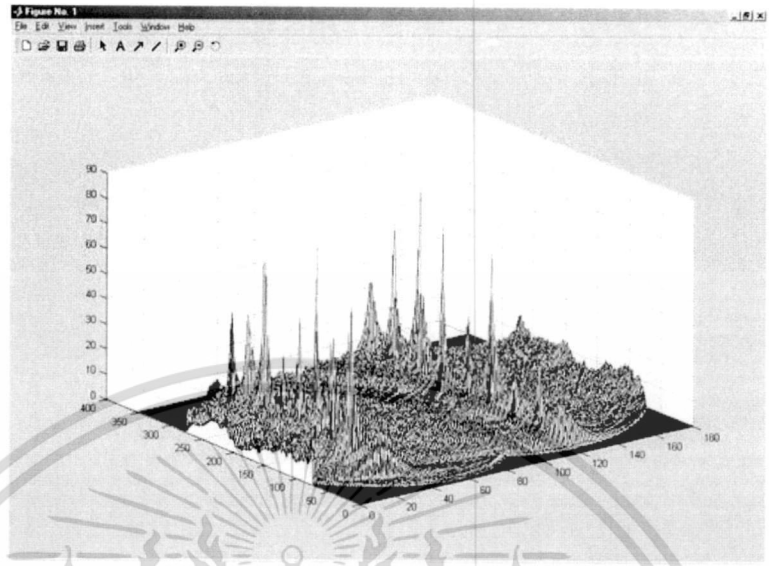
รูปที่ 3.15 ขั้นตอนการหาแนวเส้นโดยใช้เทคนิค Hough Transform

ผลการทดสอบ

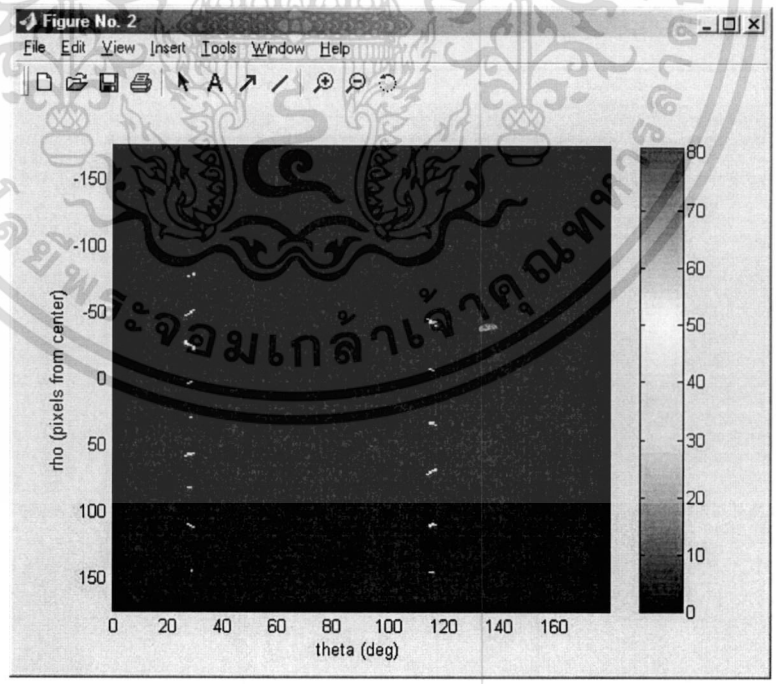


รูปที่ 3.16 กราฟที่ได้จาก Radon Transform

จากกราฟที่ได้จากเทคนิค Radon Transform จะสังเกตได้ว่ามีส่วนที่มีความสว่างที่เกิดจากการสแกนในระยะต่างๆ และในหลายๆ แนวสแกน ซึ่งผลที่ได้ออกมานั้นจะแสดงค่าของความชัดเจนของส่วนที่อาจจะเป็นเส้นตรง หากลองพล็อตกราฟนี้แบบสามมิติดังรูปที่ 3.17 จะพบว่าข้อมูลของระดับความสว่างที่เกิดขึ้นนั้นมีหลายระดับ ซึ่งยากต่อการแบ่งแยกว่าบริเวณใดที่เป็นบริเวณที่มีความน่าจะเป็นที่จะเป็นแนวเส้นตรงของขอบของบล็อก จากการศึกษาและทดลองพบว่าส่วนที่มีความน่าจะเป็นที่จะเป็นเส้นที่ชัดเจนนั้น จะเป็นส่วนที่มีความสว่างสูงๆ เพราะฉะนั้นจึงสนใจข้อมูลที่มีค่าความสว่างของกราฟ เกินกว่าร้อยละ 50 ของจุดที่มีค่าความสว่างสูงที่สุด เพื่อจะตัดส่วนของข้อมูลให้เหลือน้อยลงเพื่อที่จะวิเคราะห์ได้ง่ายขึ้น



รูปที่ 3.17 กราฟของ Radon Transform แบบ 3 มิติ



รูปที่ 3.18 กราฟที่ได้จาก radon Transform ที่ตัดส่วนของข้อมูลที่มีความสว่างน้อยออกไป

จากภาพข้างต้น จะเห็นกลุ่มของความสว่างที่มีความหนาแน่นสูงที่เกิดจากผลรวมของจำนวนของจุดภาพในแนวสแกนของ Radon Transform ที่มีความเป็นไปได้สูงที่จะเป็นแนวของเส้นตรง ในขั้นตอนนี้จะเป็นการหาตำแหน่ง rho (รัศมีของแนวสแกน)และ theta (องศาที่สแกน) ของเส้นที่จะลากตั้งฉากกับแนวสแกน เพื่อบอกแนวเส้นที่หาได้ในภาพ โดยจะใช้การหาค่าเฉลี่ยแทนค่าตำแหน่ง rho และ theta ของกลุ่มข้อมูลที่มีความสว่างเหล่านั้น การหาค่าเฉลี่ยจะใช้การหาโดยวิธีหามวลรวม โดย

$$L_x = (l_{xi}\, m_{xi})/M \quad \text{และ} \quad L_y = (l_{yj}\, m_{yj})/M \qquad (3.6.1)$$

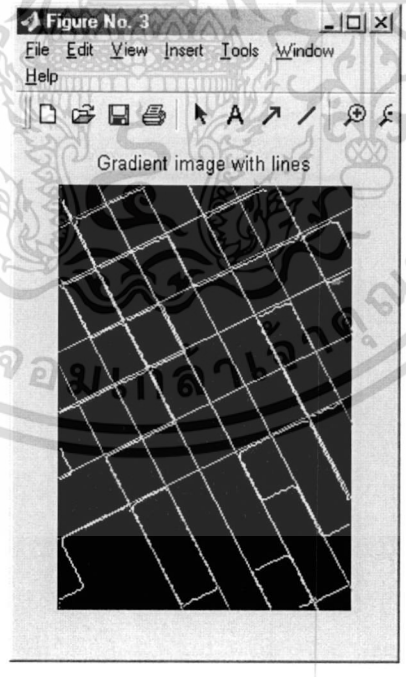
เมื่อ L_x และ L_y คือ ระยะเฉลี่ยในแนวแกน x และ y ตามลำดับ

l_{xi} และ l_{yj} คือ ระยะของจุดที่มีความสว่างในแนวแกน x และ y ตามลำดับ

m_{xi} และ m_{yj} คือ ค่าของความสว่างของจุดภาพในกราฟที่ได้จาก Radon Transform

M คือ ผลรวมของค่าความสว่างของจุดภาพในกลุ่มนั้น

จากวิธีนี้จะทำให้ได้ตำแหน่งของ rho และ theta ของแนวสแกนของ Radon Transform ที่เกิดเป็นเส้นตรง และเมื่อทดลองทำการสร้างแนวเส้นตามค่าที่ได้จาก Radon Transform จะได้แนวเส้นที่ใกล้เคียงกับความเป็นจริง ดังนี้



รูปที่ 3.19 แนวเส้นที่ทดลองสร้างขึ้นจากค่าที่ได้จาก Radon Transform

เมื่อทราบแนวเส้นตรงทั้งหมดแล้ว จะได้ตำแหน่งจุดโคออดิเนต (x,y) ของเส้นตรง จากจุดบนเส้นตรงสามารถนำมาหา ค่าความชันของเส้นตรง และ จุดตัดแกน y ได้โดยใช้สมการที่ 2.5.5 และจากแนวเส้นที่ได้จากขั้นตอนนี้จะสังเกตได้ว่าจะยังมีความคลาดเคลื่อนอยู่เล็กน้อยจึงต้องทำการหากลุ่มของจุดภาพในแนวเส้นตรงด้วยวิธีการ Least Square อีกครั้งหนึ่ง

3.7 การหาแนวเส้นตรงโดยใช้ Least Square

เพื่อต้องการให้มีความถูกต้องมากขึ้นของข้อมูลที่ได้ของแนวเส้นตรงแต่ละเส้น จะนำเอาเทคนิค Least Square มาใช้หาค่าสมการเส้นตรงของแนวเส้นตรงแต่ละเส้นจากจุดภาพ (pixel) ของแนวเส้นขอบของบล็อกแต่ละบล็อก เพื่อดูว่าจุดภาพใดเป็นจุดภาพที่อยู่ในแนวเส้นตรงเดียวกันบ้างแล้วมีค่าความชัน และค่าจุดตัดแกน y เป็นเท่าไหร่

ในขั้นตอนของการหาแนวเส้นตรงโดยวิธี Radon Transform ที่จะได้แนวเส้นตรงทำให้ทราบจุดปลายของเส้น จากจุดปลายของเส้นนี้นำมาหาความชันและจุดตัดแกน y ของเส้นตรงได้ ซึ่งจะทำให้ได้สมการเส้นตรงของเส้นแต่ละเส้น เมื่อได้สมการเส้นตรงแล้วจะทำการสร้างแนวเส้นของการอ่านจุดภาพ (pixel) เพื่อที่จะหาค่าสมการเส้นตรงให้ละเอียดยิ่งขึ้นจากจุดภาพที่กระจายกันอยู่ในแนวเส้นตรงแต่ละเส้น โดยในการอ่านจุดภาพนั้นจะทำการหาทีละแนวเส้น

Algorithm

1.) รับค่าจุดปลายของเส้น($x1_i$, $y1_i$) และ ($x2_i$, $y2_i$) จากแนวเส้นตรงที่หาได้จากเทคนิค Radon Transform (i คือลำดับของเส้นตรงที่หาได้มีค่าตั้งแต่ 1 ถึง N)

2.) นำตำแหน่งจุดปลายของเส้นทั้งหมดมาหาค่าความชัน a_i และจุดตัดแกน y b_i เพื่อสร้างสมการเส้นตรง $y = a_i x + b_i$ ของเส้นตรงแต่ละเส้นเพื่อนำไปใช้ในการสร้างแนวการอ่านจุดภาพ

3.) สร้างแนวเส้นการอ่านจุดภาพจากสมการเส้นตรงที่ได้ และทำให้การอ่านจุดภาพมีขนาดกว้างขึ้น โดยการอ่านจุด ภาพนั้นจะแบ่งเป็น 2 กรณี

- หากจะไล่อ่านค่าจากซ้ายไปขวา ให้กำหนดให้ตำแหน่ง $x1 < x2$
- หากจะไล่อ่านค่าจากขวาไปซ้าย ให้กำหนดให้ตำแหน่ง $x1 > x2$

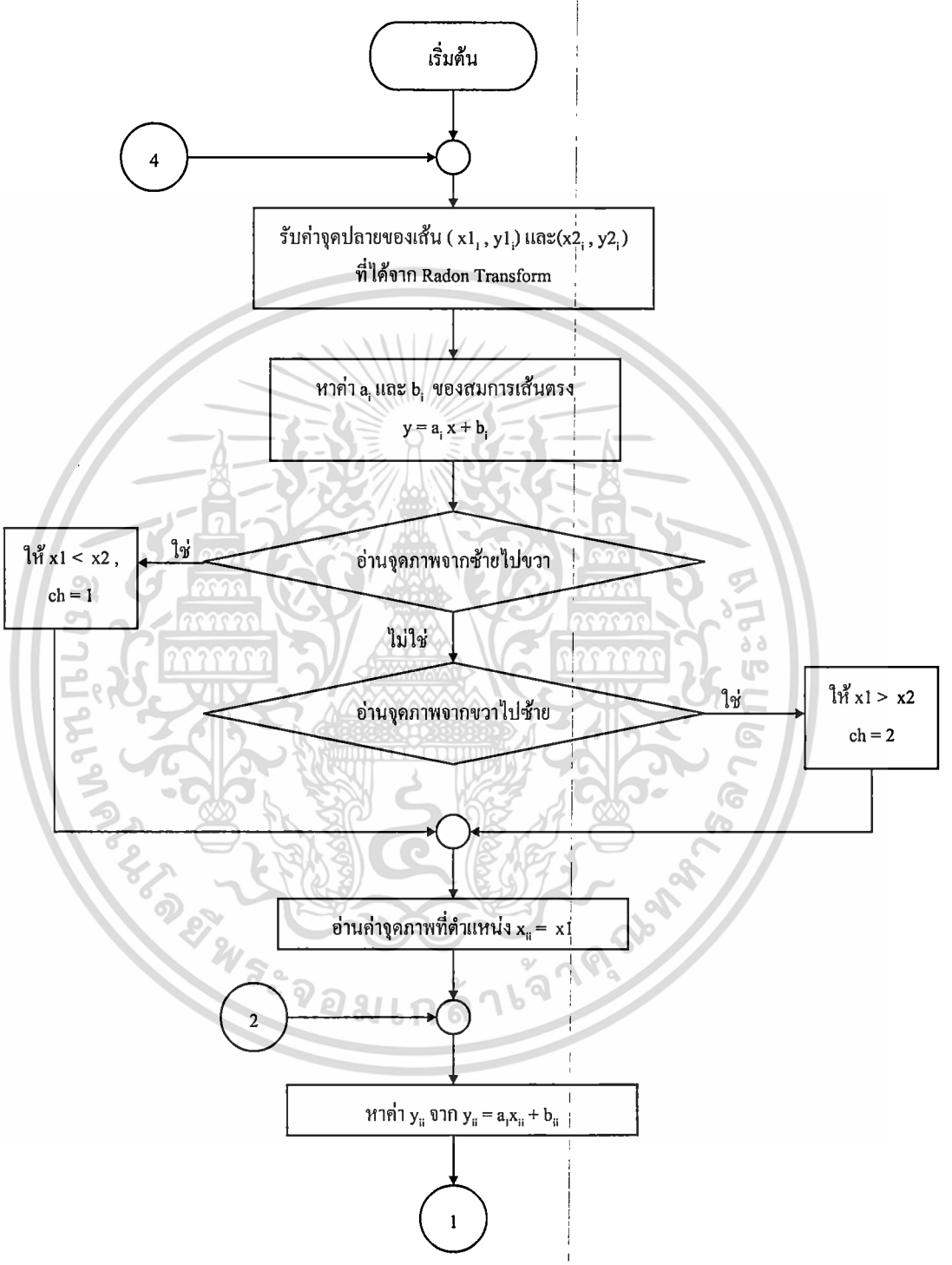
4.) ให้ทำการอ่านภาพที่จุด $x_{ij} = x1$ หาตำแหน่ง y_{ij} จาก $y_{ij} = a_i x_{ij} + b_i$ (j คือ ลำดับของตำแหน่งในแนวแกน x มีค่าตั้งแต่ 1 ถึง M ซึ่ง M คือขนาดของ ภาพในแนวแกน x)

5.) ที่จุด x_{ij} จะต้องทำการอ่านจุดภาพในแนวแกน y เพิ่มขึ้นเพื่อเป็นการทำให้ขนาดของการอ่านจุดภาพกว้างขึ้นโดยจะอ่านจุดภาพในแนวแกน y ที่ $y_{ij} - 5$ จนถึง $y_{ij} + 5$

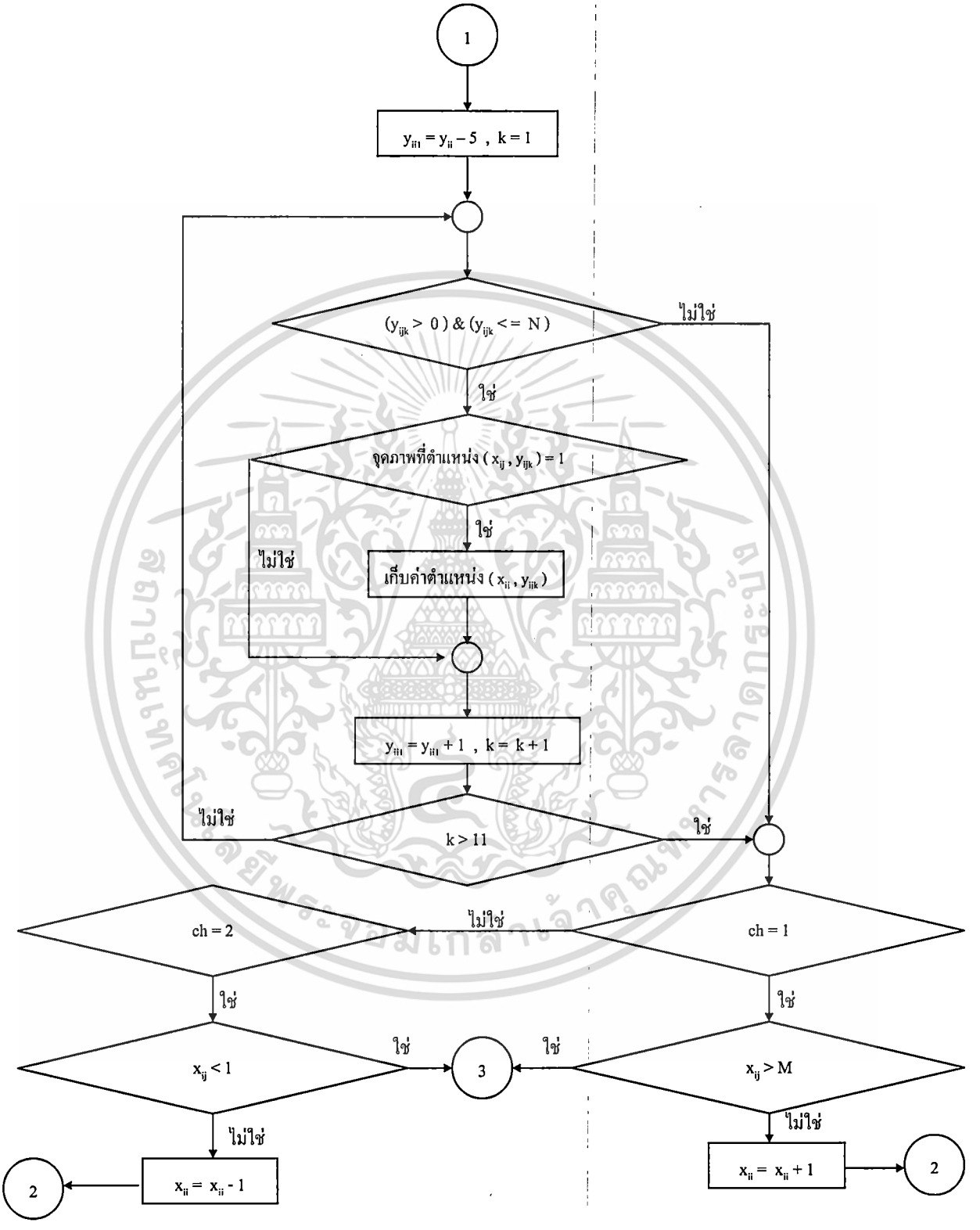
โดยจะตรวจด้วยว่าค่า $y_{ij} - 5$ จนถึง $y_{ij} + 5$ นั้นอยู่ในขนาดของภาพในแนวแกน y คือตำแหน่ง 1 ถึง N ซึ่ง N คือขนาดของภาพในแนวแกน y

6.) จาก $(x_{ij}, y_{ij} - 5)$ จนถึง $(x_{ij}, y_{ij} + 5)$ หากพบว่าที่จุดใดมีค่าของจุดภาพเป็น 1 ก็จะเก็บตำแหน่ง (x_{ij}, y_{ijk}) (k คือ ลำดับของ y มีค่าตั้งแต่ 1 ถึง 11 โดย y_{ijk} จะมีค่า $y_{ij} - 5$ จนถึง $y_{ij} + 5$) เพื่อนำไปหาเข้าสมการ least square เพื่อหาค่า ความชัน a_i และจุดตัดแกน y b_i

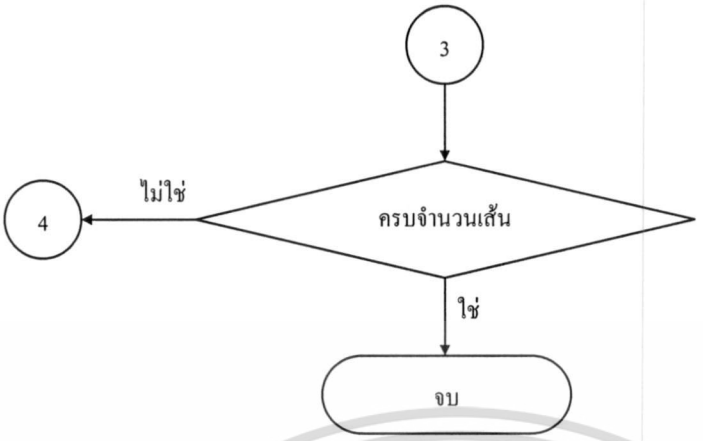
7.) ทำข้อ 4. ถึงข้อ 6. โดยเพิ่มค่า x_{ij} ทีละ 1 จน x_{ij} มีค่าเท่ากับ M ถ้าเป็นการอ่านค่าจากซ้ายไปขวา และถ้าเป็นการอ่านค่าจากขวาไปซ้ายให้ลดค่าทีละ 1 จน x_{ij} น้อยกว่า 0



รูปที่ 3. 20 ขั้นตอนการหาแนวเส้นตรงโดยใช้ Least Square (ส่วนที่ 1)



รูปที่ 3. 21 ขั้นตอนการหาแนวเส้นตรงโดยใช้ Least Square (ส่วนที่ 2)



รูปที่ 3. 22 ขั้นตอนการหาแนวเส้นตรงโดยใช้ Least Square (ส่วนที่ 3)

ผลการทดสอบ

จากการทดสอบโดยใช้อัลกอริทึ่มที่ได้กล่าวมาข้างต้น จะทำให้ได้ค่าตำแหน่งของจุดที่มีความเป็นไปได้ว่าจะอยู่ในแนวเส้นตรงเดียวกัน จากนั้นนำค่าตำแหน่งในแต่ละแนวมาเข้าสมการ least square จะได้ค่าความชัน a และจุดตัดแกน y b ของแนวเส้นตรงแต่ละแนวออกมา

รูปที่ 3.23 (a) ภาพที่ผ่านการหาขอบของเส้นแล้ว (b) เส้นสีแดงคือแนวการอ่านจุดภาพ (c) จุดภาพที่อ่านได้ในแนวการอ่านจุดภาพสีแดง

3.8 การวิเคราะห์แยกกลุ่มของเส้น

เมื่อได้ค่าความชันของแนวเส้นตรงแต่ละเส้นแล้ว จะนำค่าความชันมาใช้ในการแบ่งกลุ่มของเส้นตรงให้เป็น 2 แนว โดยการที่จะแบ่งกลุ่มของค่าความชันจะต้องแปลงค่าความชันเป็นค่าองศาของมุมของเส้นตรงก่อนโดยใช้สมการ

$$R = \arctan(m) \qquad (3.8.1)$$

โดยที่

R = ค่าเรเดียนของมุมของเส้นตรง

m = ค่าความชัน

เมื่อต้องการทราบค่ามุมเป็น θ ใช้สมการ

$$\theta = (R * 180)/\pi \qquad (3.8.2)$$

โดยที่

θ = ค่ามุมของเส้นตรง

เมื่อได้ค่ามุมของเส้นตรงแต่ละเส้นแล้วก็นำมาหาค่าเฉลี่ยของมุม แล้วใช้ค่าเฉลี่ยเป็นตัวแบ่งค่าค่ามุมออกเป็น 2 กลุ่ม

3.9 การหา Rotation Matrix

ในการหา rotation matrix นั้นจะนำเอาสมการเส้นตรงของเส้นแต่ละเส้นที่หาได้จากภาพมาใช้ โดยจากกลุ่มของเส้นตรงที่แยกไว้ 2 กลุ่มนั้น จะมีกลุ่มของ E_v ซึ่งเป็นกลุ่มของเส้นแนวตั้ง และ E_h เป็นกลุ่มของเส้นในแนวนอน ซึ่งจะสมมติให้กลุ่มใดกลุ่มหนึ่งเป็นเส้นแนวนอนหรือแนวตั้งก็ได้ และจาก E_v และ E_h นั้น จะได้ค่า $(a_{vi}\ ,\ b_{vi})$ และ $(a_{hj}\ ,\ b_{hji})$ ตามลำดับ ซึ่งเป็นค่าที่ได้จากสมการเส้นตรง $y = ax + b$ ดังนั้นจะใช้เวคเตอร์ $e = [a\ -1\ b]^T$ แทนเส้นตรงแต่ละเส้น

จากเวคเตอร์ของเส้นแต่ละเส้นนำมาหา unit-norm vector $\varepsilon = e / \| e \|$ โดยให้ $E_v = \{ \varepsilon_{vi}$ โดย iเป็นลำดับของเส้นตั้งแต่ 1 ถึง N$\}$ เป็นกลุ่มของเวคเตอร์ ε ของเส้นในแนวตั้ง ซึ่ง $\varepsilon_{vi} = e_{vi} / \| e_{vi} \|$ และ $e_{vi} = [\ a_{vi}\ -1\ \ b_{vi}]$ ซึ่งมาจาก $y = a_{vi}x + b_{vi}$ ส่วนในกลุ่มของเส้นในแนวนอน $E_h = \{ \varepsilon_{hj}$ โดย j เป็นลำดับของเส้นตั้งแต่ 1 ถึง M$\}$ ซึ่ง $\varepsilon_{hj} = e_{hj} / \| e_{hj} \|$ และ $e_{hj} = [\ a_{hj}\ -1\ \ b_{hj}]$ มาจาก $y = a_{hj}x + b_{hj}$

กำหนดให้ rotation matrix R เป็น matrix ขนาด 3×3 และให้ $r_1\ ,\ r_2\ ,\ r_3$ เป็นคอลัมน์เวคเตอร์ของ R โดย

$$R = [r_1\ r_2\ r_3] \qquad (3.8.1)$$

กำหนดให้ Q เป็น matrix ขนาด (n+m) × 3 ซึ่ง n และ m เป็นจำนวนของ cross product ในกลุ่มเวคเตอร์ E_v และ E_h ตามลำดับ โดย

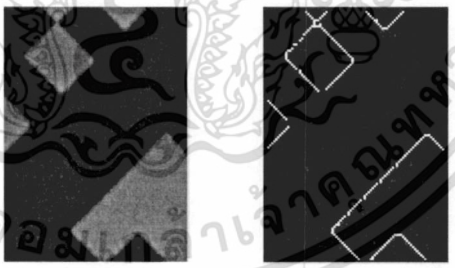
$$Q = [\ ((\varepsilon_{v1} \times \varepsilon_{v2})/(\| \varepsilon_{v1} \times \varepsilon_{v2} \|))\ \ ((\varepsilon_{v3} \times \varepsilon_{v4})/(\| \varepsilon_{v3} \times \varepsilon_{v4} \|))\ \ .....$$
$$((\varepsilon_{h1} \times \varepsilon_{h2})/(\| \varepsilon_{h1} \times \varepsilon_{h2} \|))\ \ ((\varepsilon_{h3} \times \varepsilon_{h4})/(\| \varepsilon_{h3} \times \varepsilon_{h4} \|))\ \ ....]^T \qquad (3.8.2)$$

- แต่ละคู่ของ cross product ใน E_v หรือ E_h จะเป็นค่าคงที่และมีค่าเท่ากับ $\pm r_2$ หรือ $\pm r_1$
- จากนั้นให้ $Q = USV^T$ ซึ่งเป็น singular value decomposition ของ Q จะได้

  $V = [v_1\ \ v_2\ \ v_3]$ และ $\text{diag}(S) = [s_1\ \ s_2\ \ s_3]$ โดย $s_1 = \max(\sqrt{n}, \sqrt{m})$ และ $s_2 = \max(\sqrt{n}, \sqrt{m})$ และ $s_3 = 0$ โดยถ้า $s_1 \equiv \sqrt{m}$ แล้ว $r_1 = v_1$, $r_2 = v_2$ และ $r_3 = v_3$ และ ถ้า $s_1 \equiv \sqrt{n}$ แล้ว $r_1 = v_2$, $r_2 = v_1$ และ $r_3 = v_3$

3.10 การหาแนวเส้นที่ขาดหายไป

ปัญหาอย่างหนึ่งสำหรับการวิเคราะห์ลวดลายของ binary map คือการขาดหายไปของแนวเส้นตรงที่สามารถหาได้จากขอบของบล็อก ซึ่งเกิดขึ้นจากในบริเวณนั้นมีส่วนของโทนสีของบล็อคเดียวกันทำให้ไม่สามารถหาขอบได้ ทำให้มีการหายไปของแนวเส้นตรงในส่วนนั้น



(a) (b)

รูปที่ 3.24 (a) ภาพที่นำมาวิเคราะห์ (b) แนวเส้นขอบของบล็อก

จากรูปที่ 3.24 (b) จะเห็นว่าหลังจากการหาแนวเส้นขอบของบล็อกแล้วพบว่าจะไม่สามารถทำการหาแนวเส้นของบริเวณตรงกลางของภาพนี้ได้ เนื่องจากในบริเวณนั้นเป็นบริเวณที่มีส่วนของโทนสีของลวดลายของ blue screen เป็นบริเวณใหญ่จนไม่สามารถที่จะทำการหาขอบได้ จึงต้องมีการคำนวณหาเส้นที่ขาดหายไป

กำหนดให้ ε_{v1} , ε_{v2} เป็นเวคเตอร์ที่เป็นสมาชิกของ E_v ที่สอดคล้องกับเส้นที่ตำแหน่ง x_1 และ x_2 ที่อยู่ใน X_v ตามลำดับ และเมื่อเราทราบระยะห่างของ x_1 และ x_2 โดยระยะห่าง

$$d_{21} = x_2 - x_1 \tag{3.10.1}$$

โดยที่เมื่อเราทราบ ε_{v3} ใน E_v ซึ่ง $d_{31} = x_3 - x_1$ สามารถหาระยะห่างระหว่างเส้นที่ 3 กับเส้นที่ 1 ด้วยสมการ

$$d_{31} = d_{21} \frac{\left\| \frac{\varepsilon_{v3}^T}{\varepsilon_{v3}^T r_1} - \frac{\varepsilon_{v1}^T}{\varepsilon_{v1}^T r_1} \right\|}{\left\| \frac{\varepsilon_{v2}^T}{\varepsilon_{v2}^T r_1} - \frac{\varepsilon_{v1}^T}{\varepsilon_{v1}^T r_1} \right\|} \tag{3.10.2}$$

โดยสมการนี้สามารถใช้หาระยะห่างระหว่างเส้น 3 เส้นได้ ในกรณีที่หาเส้นในแนวนอน E_h ก็จะแทนที่ v ด้วย h และแทนที่ r_1 ด้วย r_2

3.11 การหาจุดตัดของเส้นตรง

จากกลุ่มของเส้นตรงที่แยกได้ 2 กลุ่ม เราสามารถนำข้อมูลของความชันของเส้นตรงทั้งสองมาหาจุดตัดของเส้นได้ โดยนำสมการเส้นตรง $y = mx + b$ ดังนี้

ให้ $m_1x + b_1 = m_2x + b_2$ (3.11.1)

จะได้ $x = (b_2 - b_1) / (m_1 - m_2)$ (3.11.2)

เมื่อได้ค่า x แล้ว ก็นำไปแทนค่าในสมการเส้นตรงใดเส้นหนึ่งในระหว่างเส้นทั้งสองแนว จะได้ค่า y จะได้จุดตัด(x , y) ของเส้นตรงทั้งสองแนว จากจุดตัดของเส้นทั้งสองแนวนี้จะนำมาใช้เพื่อหาว่าบล็อกแต่ละบล็อกเป็นส่วนของสีน้ำเงินเข้ม หรือสีน้ำเงินอ่อน เพื่อใช้หาตำแหน่งของส่วนที่ capture มาว่าอยู่ส่วนใดของฉาก

Algorithm

1.) รับค่าความชัน ($a1_i$, $b1_i$) และ ($a2_j$, $b2_j$) โดย i = 1 ถึง N และ j = 1 ถึง M

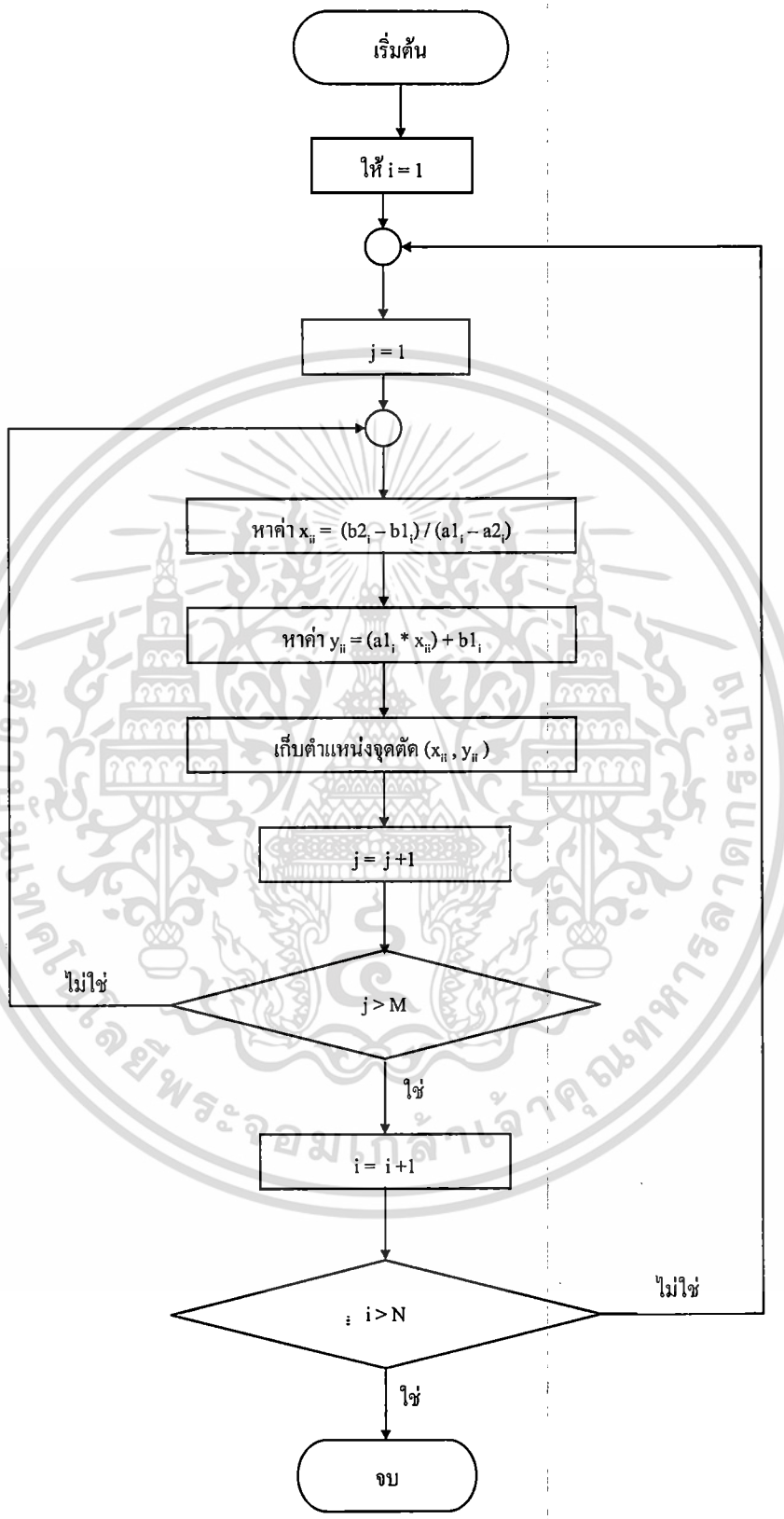
2.) ให้ i และ j เริ่มที่ 1

3.) หาค่า $x_{ij} = (b2_j - b1_i) / (a1_i - a2_j)$ โดย x_{ij} คือตำแหน่งจุดตัดของแนวเส้นที่ i กับ j

4.) หาค่า $y_{ij} = (a1_i * x_{ij}) + b1_i$

5.) เก็บตำแหน่งจุดตัด (x_{ij} , y_{ij})

6.) ย้อนทำข้อ 3 ถึง 5 จนกว่าจะครบจำนวนของ i และ j



รูปที่ 3.25 ขั้นตอนการหาจุดตัดของเส้น

3.12 การหา Binary Pattern

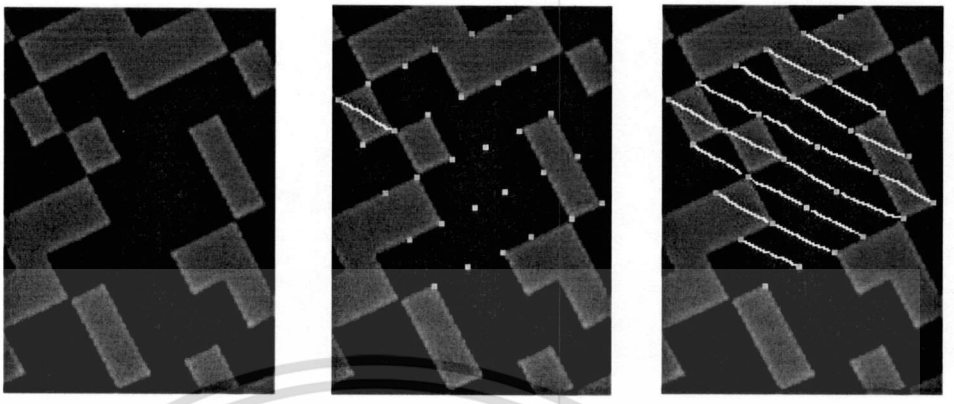
จากจุดตัดของเส้นตรงแต่ละเส้นเราจะนำจุดตัด มาหาค่าสีในแต่ละช่องตารางโดยการไล่หาค่าสีจะแปลงภาพที่ถ่ายมาเป็นภาพแบบ grayscale ก่อนเพื่อให้ง่ายต่อการดูระดับของความสว่างของจุดภาพ จากนั้นจะใช้การอ่านจุดภาพในแนวทแยงมุมจากจุดตัดจุดหนึ่งไปอีกจุดหนึ่งแล้วดูว่าค่าสีเฉลี่ยในแนวทแยงมุมนั้นเป็นส่วนของสีน้ำเงินเข้มหรือสีน้ำเงินอ่อน แล้วนำมาสร้างเป็น binary matrix แล้วนำเอา matrix นี้ไปเทียบหาใน binary map ว่าอยู่ส่วนใดของ binary map เพื่อที่จะทราบว่าตำแหน่งของเส้นที่พบในส่วนของภาพที่ capture มานั้นเป็น x_i มีค่าเท่าไหร่ใน X_v และ y_i มีค่าเท่าไหร่ใน Y_h

Algorithm

1.) เริ่มอ่านค่าที่ตำแหน่งจุดตัด (x_{ij} , y_{ij}) ไล่ในแนวทแยงมุมจนถึงจุดตัด ($x_{i+1, j+1}$, $y_{i+1, j+1}$)
2.) หาสมการเส้นตรงระหว่างจุดตัด (x_{ij} , y_{ij}) และ ($x_{i+1,j+1}$, $y_{i+1,j+1}$) ได้ค่า a_{ij} และ b_{ij}
3.) ให้เริ่มที่ x_{ij} แล้วหาค่าตำแหน่ง y_{ij} จาก $y_{ij} = a_{ij} * x_{ij} + b_{ij}$
4.) อ่านค่าจุดภาพที่ตำแหน่ง (x_{ij} , y_{ij}) และเก็บค่าสีของจุดภาพนั้นไว้ใน SumCol[]
5.) เพิ่มตำแหน่ง x_{ij} หนึ่งตำแหน่ง แล้วทำข้อ 3. และ 4 ไปจนถึงตำแหน่ง $x_{i+1, j+1}$
6.) หา ค่าเฉลี่ยของค่าสีที่อ่านจากจุดภาพ = (SumCol[] / จำนวนจุดภาพที่อ่าน)
7.) เอา ค่าเฉลี่ยของค่าสีที่อ่านจากจุดภาพ มาเทียบกับ ค่าเฉลี่ยของค่าสีของสีน้ำเงินเข้มและสีน้ำเงินอ่อน ซึ่งจะเป็นค่าที่ใช้แบ่งระดับของสีน้ำเงินเข้มกับอ่อน
8.) ถ้า ค่าเฉลี่ยค่าสีที่อ่านจากจุดภาพมีค่ามากกว่า ค่าที่แบ่งระดับสีน้ำเงิน ให้เก็บค่าเป็น 1 ไว้ใน submat[i,j]
9.) ถ้า ค่าเฉลี่ยค่าสีที่อ่านจากจุดภาพมีค่าน้อยกว่า ค่าที่แบ่งระดับสีน้ำเงิน ให้เก็บค่าเป็น 0 ไว้ใน submat[i,j]
10.) เริ่มทำข้อ 1 ถึง 9 โดยไล่ลำดับของการอ่านค่าในแนวทแยงมุมจนครบทุกจุดตัดของเส้น จะได้ค่า binary pattern เก็บไว้ใน submat[]

ผลการทดสอบ

เมื่อนำภาพที่ถ่ายมาผ่านฟังก์ชั่น rgb2gray() ในโปรแกรม matlab จะได้ภาพแบบ grayscale จากนั้นนำภาพนี้ไปหาค่าเฉลี่ยของค่าความสว่างของค่าสี 2 ระดับ จะได้ค่าที่จะสามารถแบ่งระดับของความสว่าง 2 ระดับ นำค่านี้ไปใช้ในการหาว่าในส่วนไหนในภาพที่เป็นส่วนสีน้ำเงินเข้ม หรือสีน้ำเงินอ่อน ซึ่งจะอยู่ในรูปของความสว่างแบบ grayscale

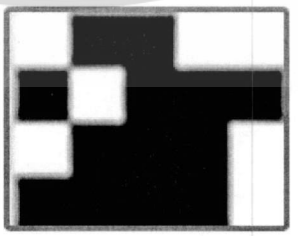


รูปที่ 3.26 (a) ภาพที่แปลงเป็น grayscale (b) จุดตัดที่หาได้ในขั้นตอนการหาจุดตัด (c) แนวเส้นการหาค่าสีเฉลี่ยในแนวทแยงมุม

จากภาพตัวอย่างในรูปที่ 3.26 เมื่อนำค่าใน submat[] ซึ่งได้จากขั้นตอนที่ผ่านมาสามารถแสดงได้ ดังนี้

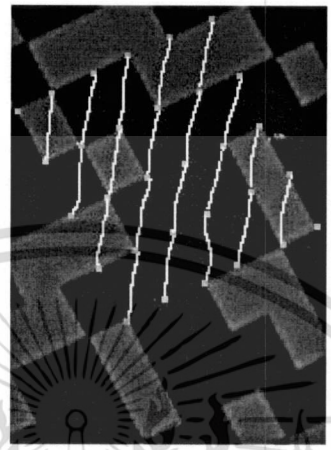
$$\text{submat} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

เมื่อกำหนดให้ ตำแหน่งที่มีความสว่างมากมีค่าเป็น 1 และตำแหน่งที่มีความสว่างน้องมีค่าเป็น 0 เมื่อนำค่าใน submat[] มาแสดงจะได้ ดังนี้



รูปที่ 3.27 ภาพ binary pattern ที่หาได้

อาจจะมีการหาค่าสีในอีกแนวก็ได้ ด้วยการหาค่าเฉลี่ยในแนวเฉียงขึ้น ดังรูป



รูปที่ 3.28 แนวการหาค่าเฉลี่ยสีในแนวเฉียงขึ้น

3.13 การหาตำแหน่งของ Binary Pattern Submatrix ใน Binary Map

เมื่อสามารถหาส่วนของ submatrix ได้แล้ว จะต้องมาดูว่าขนาดของ submatrix นั้นมีขนาดใหญ่ถึง 3*5 หรือไม่ เนื่องจากในขั้นตอนของการสร้าง binary map ได้กำหนดไว้ว่า แต่ละ submatrix จะต้องมีขนาด เกินกว่า 3*5 จึงจะรับประกันได้ว่าแต่ละ submatrix นั้น ไม่มีลวดลายที่ซ้ำกันใน binary map จากนั้นก็จะมาหาว่า submatrix ที่หามาได้นั้นอยู่ที่ ส่วนใดของ binary map เพื่อที่จะทำให้ทราบว่า แนวเส้นใน submatrix นั้นอยู่ในตำแหน่งใดใน X_v และ Y_h

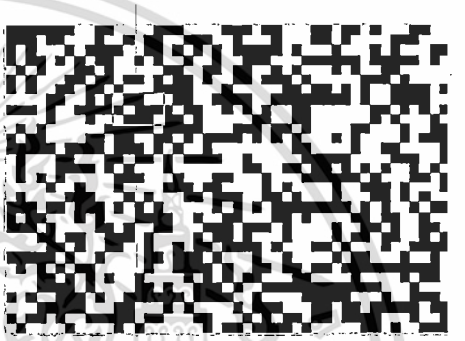
Algorithm

1.) รับค่า submat[] และค่า bimap[]
2.) ตรวจว่าขนาดของ submat ถึง 3*5 หรือไม่
3.) ถ้าเกินกว่า 3*5 ให้ตัดขนาดของ submat เหลือ 3*5
4.) ถ้าขนาดไม่พอให้ขึ้นข้อความว่า " ไม่สามารถหาตำแหน่งของ submatrix ได้"
5.) นำ submat[] ขนาด 3*5 ตรวจหาว่าอยู่ตำแหน่งใดของ binary map โดยตำแหน่งที่ได้นั้นจะเป็นตำแหน่งของสมาชิกตัวบนซ้ายของ submat[] ที่อยู่ใน bimap[]
6.) ถ้าหาไม่พบให้ขึ้นข้อความว่า " ไม่สามารถหาตำแหน่งของ submatrix ได้"

ผลการทดสอบ

จากการทดสอบเมื่อนำส่วนของ submat[] และ bimap[] มาแสดงเป็นภาพจะมองเห็นได้ชัดเจนยิ่งขึ้นว่า submatrix นั้นอยู่ในส่วนใดของ binary map และเมื่อรู้ตำแหน่งของ submat[] ซึ่งก็คือตำแหน่งของสมาชิกที่อยู่มุมบนด้านซ้ายมือ ของ submat[] แต่ตำแหน่งที่ได้นี้จะเป็นตำแหน่งของบล็อกสี่เหลี่ยมไม่ใช่ตำแหน่งของเส้น เพราะฉะนั้น จะต้องมีการคำนวณว่าเส้นขอบของบล็อกสี่เหลี่ยมที่อยู่ใน submatrix นั้นเป็นเส้นที่อยู่ในตำแหน่ง X_v และ Y_h

(a)



(b)

รูปที่ 3.29 (a) ส่วนของ submatrix (b) ตำแหน่งของ submatrix ใน binary map

ในการคำนวณหาตำแหน่งของแนวเส้นใน submatrix นั้น เมื่อได้ตำแหน่งมุมบนด้านซ้ายของ submatrix ว่าอยู่ตำแหน่งใดใน binary map ก็จะนำมาหาตำแหน่งของเส้นแนวตั้งที่ 1 กับเส้นแนวนอนที่ 1 ของ submatrix ได้

กำหนดให้ binary matrix ที่นำมาสร้างเป็น binary map มีขนาด 30*42 และขนาดของบล็อกสี่เหลี่ยมแต่ละบล็อกใน binary map มีขนาด 10*10 หน่วย เพราะฉะนั้นขนาดของ binary map จะเท่ากับ 300 * 420 หน่วย

ให้การถ่ายภาพเริ่มที่ตำแหน่ง จุดศูนย์กลางของภาพ จะอยู่ที่ตำแหน่ง

$$x_c = (\text{ขนาดของ map ในแนวแกน x})/2 \qquad (3.13.1)$$

$$y_c = (\text{ขนาดของ map ในแนวแกน y})/2 \qquad (3.13.2)$$

เพราะฉะนั้น ตำแหน่งจุดศูนย์กลางของภาพจะอยู่ที่ (420/2 , 300/2) และเมื่อนำมาพิจารณาจะทราบว่าตำแหน่งของ X_v ซึ่งเป็นตำแหน่งของเส้นในแนวตั้ง และ Y_h เป็นตำแหน่งของเส้นในแนวนอน จะได้

กลุ่มของเส้นในแนวตั้ง

$X_v = \{ -(420/2) , -(420/2)+10 , -(420/2)+ (2*10) , \ldots , (420/2)\}$

และกลุ่มของเส้นในแนวนอน

$Y_h = \{ -(300/2) , -(300/2)+10 , -(300/2)+ (2*10) , \ldots , (300/2)\}$

เมื่อได้ตำแหน่งจุดศูนย์กลางมาแล้วก็นำมาเปรียบเทียบกับตำแหน่งของ submatrix ที่หามาได้ จากภาพตัวอย่าง ในรูปที่ 3.29 หาตำแหน่งของ submatrix ได้เป็น แถวที่ 9 และคอลัมน์ที่ 12 จากนั้นนำมาหาตำแหน่งของแนวเส้นในแนวนอนและแนวตั้งที่ 1 ใน submatrix

- ตำแหน่งที่หาได้ในแนวนอน คือ แถวที่ 9 ดังนั้น เส้นขอบด้านบนของบล็อกแรกใน submatrix จะอยู่ที่ระยะ (9-1) คือ – (300/2) + (8*10)
- ตำแหน่งที่หาได้ในแนวตั้ง คือ คอลัมน์ที่ 12 ดังนั้น เส้นขอบด้านซ้ายมือของบล็อกแรกใน submatrix จะอยู่ที่ระยะ (12-1) คือ – (420/2) + (11*10)

ดังนั้นจะทราบได้ว่า เส้นที่หาได้ใน submatrix นั้นเป็นสมาชิกลำดับที่เท่าไหร่ใน กลุ่มของแนวเส้น X_v และ Y_h

3.14 การหาระยะโฟกัสของการถ่ายภาพ

ในที่กล่าวมาข้างต้นเป็นการหาค่า rotation matrix R และ translation matrix T เป็นกรณีที่ไม่ได้มีการนำค่า ความยาวโฟกัส f มาคิดด้วย ในกรณีที่นำค่า f มาคิดด้วยนั้น จะไม่สามารถใช้ค่า (a , b) ที่ได้จากสมการเส้นตรงมาคิดได้แต่จะต้องนำค่า f มาหารค่า b ก่อน จึงจะสามารถนำไปคำนวณหาค่า e ได้ โดย $e = [a \ -1 \ \beta/f]^T$

กำหนดให้ q_h เป็นผลของการ cross product ของแต่ละคู่ใดๆ ของสมาชิกใน E_h เมื่อกำหนดให้ q_{h12} เป็นเวคเตอร์ที่เก็บ 2 ค่าแรกใน q_h และ q_{h3} เก็บค่าตัวที่ 3 ในกรณีของ q_v ก็ทำเช่นเดียวกัน จากนั้นสามารถหา f โดยสมการ

$$f^2 = \frac{q_{h12} \cdot q_{v12}}{q_{h3} \cdot q_{v3}} \qquad (3.14.1)$$

สมการข้างต้นคือการหาความยาวโฟกัส โดยใช้ค่าที่เกิดจากการ cross product เพียงคู่เดียวใน E_v และ E_h เพื่อความถูกต้องมากยิ่งขึ้น จะใช้ค่าที่เกิดจากการ cross product หลาย ๆ คู่ มากขึ้นจะได้เป็นสมการ

$$f^2 = \frac{\sum_k (q_{hk3} q_{vk3} q_{hk2} q_{vk2})}{\sum_k (q_{hk3} q_{vk3})^2} \qquad (3.14.3)$$

เมื่อ k คือ จำนวนคู่ของการ cross product ใน E_v และ E_h เมื่อได้ค่า f แล้ว ก็นำไปหาค่า E_v และ E_h ตัวใหม่ โดยใช้ ค่า $b = \beta/f$

3.15 การหาค่ามุมการหมุนรอบแนวแกน XYZ

จาก rotation matrix R ที่หาได้นั้น จะสามารถนำมาหามุมการหมุนรอบแกน X , แกน Y และแกน Z ได้โดย rotation matrix ที่หาได้นี้จะเป็น rotation matrix ที่เกิดจากการคูณกันของ rotation matrix ของการหมุนรอบแกน X , แกน Y และแกน Z ตามลำดับ ซึ่งได้อธิบายไว้ในหัวข้อ 2.6 ดังนั้นถ้าเอาค่าที่ได้มาทำการแก้สมการจะได้ค่าองศาของมุมในแต่ละแนวแกน แต่ปัญหาอย่างหนึ่งคือ rotation matrix ที่หาได้นั้นยังมีความไม่แน่นอนของเครื่องหมายบวก หรือลบในแต่ละค่าซึ่งก่อนที่จะทำการแก้สมการหาค่ามุม จะต้องทำการปรับเครื่องหมายให้ตรงกับเครื่องหมายที่แท้จริงของ rotation matrix R_{xyz} เสียก่อนจึงจะทำการหาได้

จากสมการที่ 2.6.4 $R_{xyz}(1,3) = -\sin(\gamma_{radian})$ (3.15.1)

จะสามารถหาค่ามุม γ ได้โดยใช้

$$\gamma_{radian} = \arcsin(R(1,3)) \qquad (3.15.2)$$

แล้วแปลงเป็นองศาโดย

$$\gamma = (\gamma_{radian} * 180)/\pi \qquad (3.15.3)$$

เมื่อได้ค่า γ มาแล้วจะทำให้ทราบมุมของการหมุนรอบแกน Y หากต้องการทราบค่ามุมการหันเหของกล้องรอบแกน X และแกน Z ก็นำค่า มุม γ ไปแทนในตำแหน่งใดตำแหน่งหนึ่งใน rotation matrix

หากต้องการหาการหมุนรอบแกน X จะหาได้โดย

จากสมการที่ 2.6.4 $R_{xyz}(3,3) = \cos(\theta_{radian})\cos(\gamma_{radian})$ (3.15.4)

จะสามารถหาค่ามุม θ ได้โดยใช้

นำค่า γ_{radian} ไปแทนใน $\cos(\gamma_{radian})$

ดังนั้น $\cos(\theta_{radian}) = R(3,3)/\cos(\gamma_{radian})$ (3.15.5)

จากนั้นหา

$$\theta_{radian} = \arccos(R(3,3)/\cos(\gamma_{radian})) \quad (3.15.6)$$

แล้วแปลงเป็นองศาโดย

$$\theta = (\theta_{radian} * 180)/\pi \quad (3.15.7)$$

เมื่อได้ค่า θ มาแล้วจะทำให้ทราบมุมของการหมุนรอบแกน X

หากต้องการหาการหมุนรอบแกน Z จะหาได้โดย

จากสมการที่ 2.6.4 $R_{xyz}(1,1) = \cos(\gamma_{radian})\cos(\beta_{radian})$ (3.15.8)

จะสามารถหาค่ามุม β ได้โดยใช้

นำค่า γ_{radian} ไปแทนใน $\cos(\gamma_{radian})$

ดังนั้น $\cos\beta_{radian}) = R(1,1)/\cos(\gamma_{radian})$ (3.15.9)

จากนั้นหา

$$\beta_{radian} = \arccos(R(1,1)/\cos(\gamma_{radian})) \quad (3.15.10)$$

แล้วแปลงเป็นองศาโดย

$$\beta = (\beta_{radian} * 180)/\pi \quad (3.15.7)$$

เมื่อได้ค่า β มาแล้วจะทำให้ทราบมุมของการหมุนรอบแกน Z

บทที่ 4
ผลการวิจัย

การวิจัยในหัวข้อเรื่องการหามุมและตำแหน่งของกล้องวีดีโอจากฉากที่เป็น Binary Pattern จะใช้สร้างภาพจำลองขึ้นมาโดยใช้โปรแกรม 3D Studio Max 6 และการถ่ายภาพก็จะใช้กล้องที่มีในโปรแกรม 3D Studio Max 6 เพื่อให้ง่ายต่อการกำหนดค่าต่างๆ ที่จะใช้ทดสอบ และเป็นการลดปัญหาในเรื่องแสงและเงามืดในการถ่ายทำด้วยกล้องจริงๆ

4.1 เครื่องมือที่ใช้ในการทดสอบ

สำหรับการสร้างภาพในโปรแกรม 3D Studio Max 6 นั้น จะกำหนดค่าต่างๆ ดังต่อไปนี้

- กำหนดให้ฉากที่สร้างขึ้นเป็น plane กว้าง 300 ยาว 400 แล้วนำภาพ binary map ที่สร้างขึ้นมาแปะบน plane
- ให้ระนาบของ plane ขนานกับระนาบ XY
- ใช้กล้องแบบ Free โดยให้ตำแหน่งของการถ่ายกล้องเริ่มที่ตำแหน่ง (0,0,0) และมุมเริ่มต้นที่ (0,0,0)
- ใช้ เลนส์ 35 mm.
- ใช้โปรแกรม Matlab 6.1 ในการสร้างฟังชั่นเพื่อการทดสอบ

4.2 รูปแบบการทดลอง

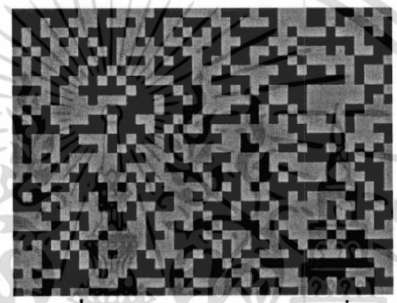
การทดลองจะออกแบบการทดลองเป็น 3 แบบ ดังต่อไปนี้

- **การทดลองที่ 1** : เมื่อมีการทดลองเปลี่ยนรูปแบบของ Binary Map แล้วจะมีผลต่อการหาค่ามุมอย่างไร
- **การทดลองที่ 2** : หากเพิ่มมุมในการหมุนของกล้องรอบแกนทั้ง 3 แกนแล้วจะมีผลต่อการหาค่ามุมอย่างไร
- **การทดลองที่ 3** : เป็นการทดสอบการหาค่ามุมโดยใช้วิธีการเฉลี่ยจากกล้อง 2 ตัว

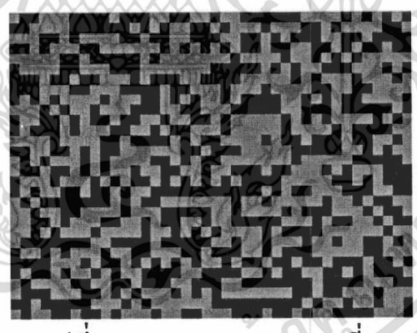
4.3 การทดลอง

4.3.1 การทดลองที่ 1

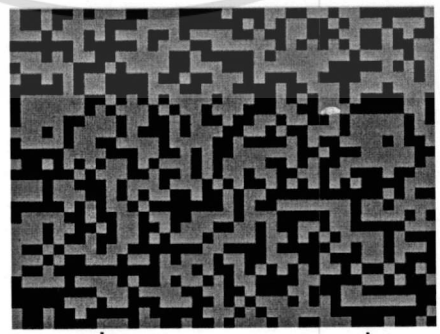
ทดลองเปลี่ยนรูปแบบของ binary map ที่นำมาทดลองเพื่อดูว่า เมื่อ pattern เปลี่ยนไปจะมีผลต่อการหามุมการหมุนของกล้องอย่างไร โดยใช้ binary map 3 รูปแบบ และในแต่ละตำแหน่งการหันเหของกล้องที่ทำการทดลองจะมีการเลื่อนตำแหน่งของการจับภาพไปในระยะใกล้เคียงกัน (ประมาณ 5 เซนติเมตร) แล้วนำค่ามุมที่ได้มาหาค่าเฉลี่ย แล้วดูการเปลี่ยนแปลงที่เกิดขึ้น โดยรูปแบบของ map ที่นำมาทดลองทั้ง 3 แบบ แสดง ดังรูป



รูปที่ 4.1 Binary Map แบบที่ 1



รูปที่ 4.2 Binary Map แบบที่ 2



รูปที่ 4.3 Binary Map แบบที่ 3

ผลการทดลองเมื่อใช้ binary map แบบที่ 1

ตารางที่ 4.1 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,0,30)

เฉลี่ย	**4.926158**	**0.571853**	**30.04212**
RMSE	**1.402658**	**0.874235**	**0.314396**

ตารางที่ 4.2 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,10,30)

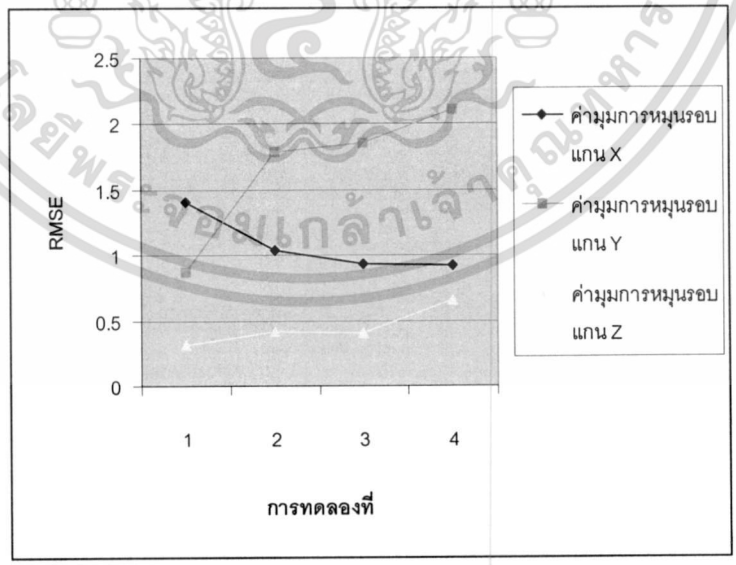
เฉลี่ย	**5.08657**	**9.590021**	**29.70713**
RMSE	**1.034257**	**1.784169**	**0.419749**

ตารางที่ 4.3 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,15,30)

เฉลี่ย	**5.52598**	**16.03722**	**30.06749**
RMSE	**0.92897**	**1.859766**	**0.405606**

ตารางที่ 4.4 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,20,30)

เฉลี่ย	**4.707328**	**19.78143**	**30.13701**
RMSE	**0.916365**	**2.108336**	**0.659109**



รูปที่ 4.4 กราฟแสดงแนวโน้มของค่า RMSE ที่ เมื่อใช้ binary map แบบที่ 1

ผลการทดลองเมื่อใช้ binary map แบบที่ 2

ตารางที่ 4.5 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,0,30)

เฉลี่ย	**6.05815**	**0.594521**	**29.86984**
RMSE	**1.106171**	**0.45626**	**0.20709**

ตารางที่ 4.6 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,10,30)

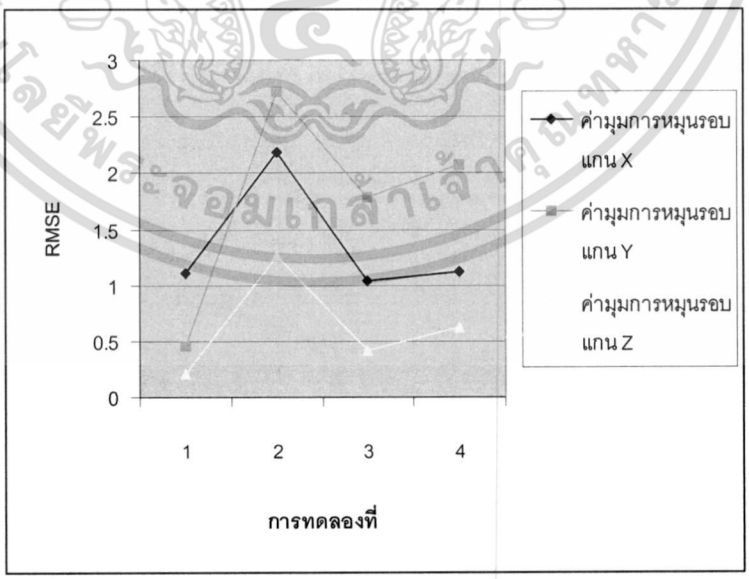
เฉลี่ย	**6.058646**	**9.851697**	**28.96079**
RMSE	**2.186915**	**2.721617**	**1.256107**

ตารางที่ 4.7 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,15,30)

เฉลี่ย	**5.67454**	**15.85947**	**30.03392**
RMSE	**1.11318**	**2.079612**	**0.62395**

ตารางที่ 4.8 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,20,30)

เฉลี่ย	**4.707328**	**19.78143**	**30.13701**
RMSE	**0.916365**	**2.108336**	**0.659109**



รูปที่ 4.5 กราฟแสดงแนวโน้มของค่า RMSE ที่ เมื่อใช้ binary map แบบที่ 2

ผลการทดลองเมื่อใช้ binary map แบบที่ 3

ตารางที่ 4.9 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,0,30)

เฉลี่ย	**6.042214**	**0.842921**	**29.79793**
RMSE	**1.677369**	**1.078818**	**0.399218**

ตารางที่ 4.10 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,10,30)

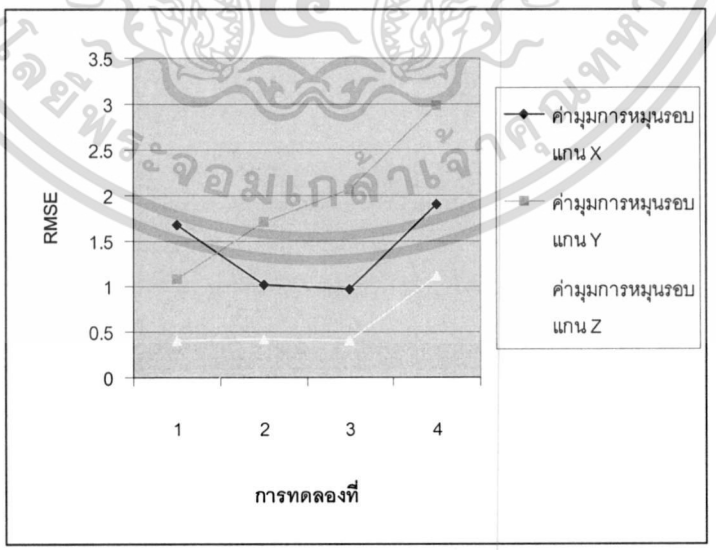
เฉลี่ย	**4.35998**	**9.56553**	**30.14604**
RMSE	**1.01847**	**1.702975**	**0.421728**

ตารางที่ 4.11 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,15,30)

เฉลี่ย	**5.194469**	**15.07896**	**29.92137**
RMSE	**0.96366**	**2.071635**	**0.407308**

ตารางที่ 4.12 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,20,30)

เฉลี่ย	**5.293066**	**19.22432**	**29.36258**
RMSE	**1.908526**	**2.97644**	**1.117311**



รูปที่ 4.6 กราฟแสดงแนวโน้มของค่า RMSE ที่ เมื่อใช้ binary map แบบที่ 3

4.3.1.1 สรุปผลการทดลองที่ 1

จากการทดลองเปลี่ยนแปลงค่ามุมและเปลี่ยนตำแหน่งของกล้องในการถ่ายทำในระยะและมุมต่างๆรวมทั้งยัง เปลี่ยนแปลงรูปแบบของ Binary Map แล้ว พบว่าการเปลี่ยนแปลงไปของ map ไม่ได้มีผลทำให้ค่าเฉลี่ยของมุมและค่า error ที่ได้เปลี่ยนแปลงไปในทางที่ดีขึ้น หรือ แย่ลงเลย ทั้ง 3 แบบ ให้ค่ามุมที่ยังไม่แสดงแนวโน้มที่จะบอกได้ว่า map แบบใดให้ผลที่ดีกว่าอีกแบบหนึ่ง แต่พบว่าค่า error ที่เกิดขึ้นที่การหมุนรอบแกน Y จะมีค่า error สูงที่สุด และค่า error ที่การหมุนรอบแกน X จะน้อยที่สุด

4.3.2 การทดลองที่ 2

จากที่พบว่า error ที่เกิดขึ้นมักเกิดจากจุดในแนวเส้นตรงมีน้อย ซึ่งเกิดจากการที่ในแนวเส้นตรงนั้นมีส่วนของขอบของบล็อกน้อย จึงให้การทดลองที่สองใช้รูปแบบ Binary Map แบบตารางหมากรุก เพื่อให้ส่วนของเส้นขอบของบล็อกมีความสมบูรณ์มากขึ้น เพื่อที่จะทดสอบว่าการเปลี่ยนแปลงมุมมีผลต่อการหาค่ามุมอย่างไร

รูปที่ 4.7 Binary Map แบบตารางหมากรุก

ตารางที่ 4.13 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (0 ,0,20)

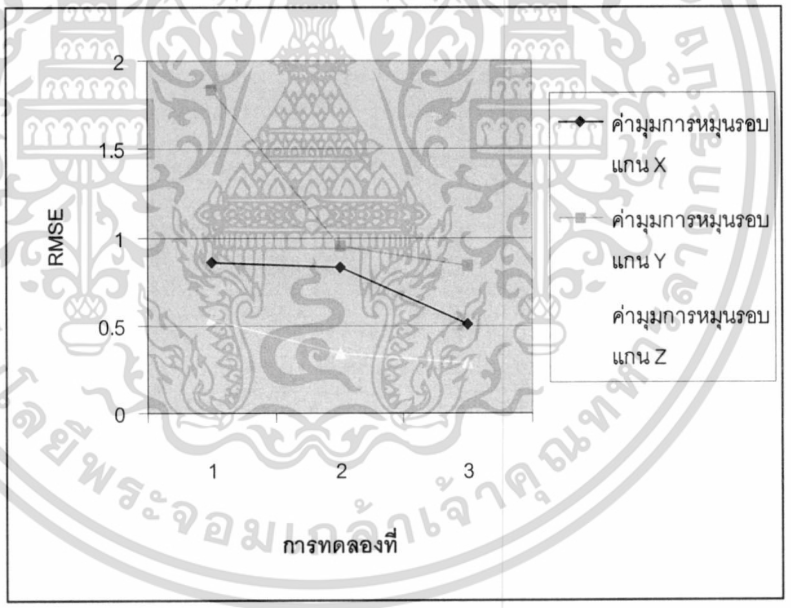
เฉลี่ย	**0.735771**	**1.532176**	**20.02042**
RMSE	**0.859252**	**1.835414**	**0.053413**

ตารางที่ 4.14 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (10,10,20)

เฉลี่ย	**10.16135**	**10.05579**	**19.97221**
RMSE	**0.827718**	**0.945269**	**0.337513**

ตารางที่ 4.15 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (20 ,20,20)

เฉลี่ย	**20.11962**	**19.96779**	**19.97895**
RMSE	**0.505772**	**0.84222**	**0.287014**



รูปที่ 4.8 กราฟแสดงแนวโน้มของค่า RMSE ที่ [(0,0,20) , (10,10,20) , (20,20,20)]

จากกราฟแกนนอน คือ

การทดลองที่ 1 : แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (0 ,0,30)

การทดลองที่ 2 : แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (10 ,10,30)

การทดลองที่ 3 : แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (20 ,20,30)

ตารางที่ 4.16 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (0 ,0,30)

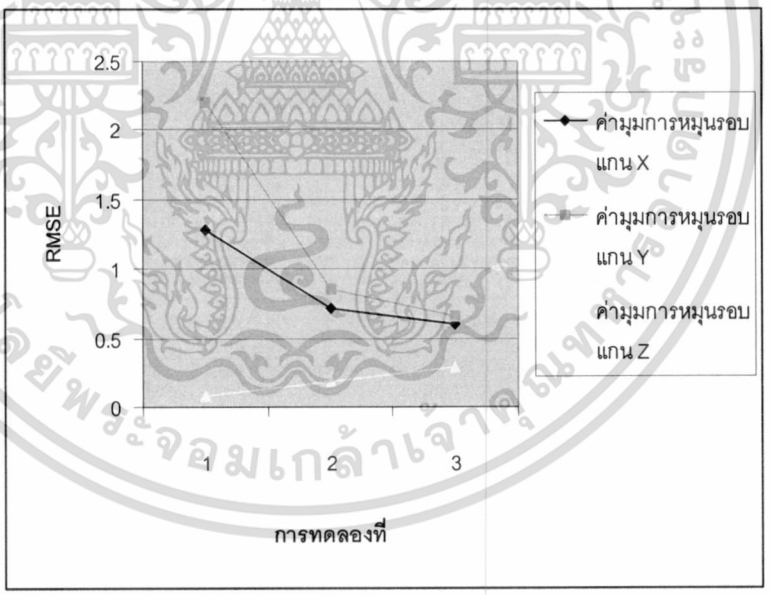
เฉลี่ย	**1.059285**	**1.672632**	**30.04685**
RMSE	**1.28152**	**2.201796**	**0.08524**

ตารางที่ 4.17 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (10,10,30)

เฉลี่ย	**10.00467**	**9.891831**	**29.95429**
RMSE	**0.713897**	**0.855182**	**0.182239**

ตารางที่ 4.18 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (20 ,20,30)

เฉลี่ย	**19.9846**	**19.8782**	**29.98850**
RMSE	**0.597763**	**0.653357**	**0.285233**



รูปที่ 4.9 กราฟแสดงแนวโน้มของค่า RMSE ที่ [(0,0,30) , (10,10,30) , (20,20,30)]

จากกราฟแกนนอน คือ

การทดลองที่ 1 : แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (0 ,0,30)

การทดลองที่ 2 : แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (10 ,10,30)

การทดลองที่ 3 : แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (20 ,20,30)

ตารางที่ 4.19 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (0 ,0,40)

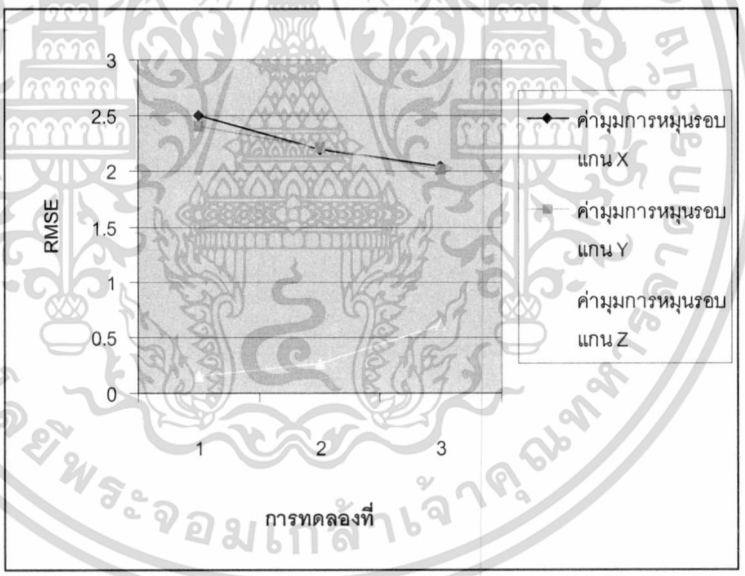
เฉลี่ย	**1.568936**	**1.17395**	**40.1121**
RMSE	**1.746067**	**1.39833**	**0.149984**

ตารางที่ 4.20 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (10,10,40)

เฉลี่ย	**8.741035**	**8.87146**	**40.0507**
RMSE	**2.20066**	**2.21354**	**0.256541**

ตารางที่ 4.21 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (20 ,20,40)

เฉลี่ย	**19.51613**	**19.64011**	**40.05929**
RMSE	**2.04494**	**2.019765**	**0.611516**



รูปที่ 4.10 กราฟแสดงแนวโน้มของค่า RMSE ที่ [(0,0,40) , (10,10,40) , (20,20,40)]

จากกราฟแกนนอน คือ

การทดลองที่ 1 : แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (0 ,0,40)

การทดลองที่ 2 : แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (10 ,10,40)

การทดลองที่ 3 : แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (20 ,20,40)

4.3.2.1 สรุปผลการทดลองที่ 2

จากแนวโน้มของค่า Root Mean Square Error (RMSE) นั้น สังเกตได้ว่าเมื่อมีการเพิ่มมุมในการหมุนรอบแกน x และแกน y เพิ่มมากขึ้น พบว่าค่า RMSE ในการหมุนรอบแกน x และ แกน y จะมีค่าลดลงควบคู่กัน ทั้งนี้เนื่องมาจากแนวเส้นมีการโน้มเอียงมากขึ้นทำให้การหาค่าความชันของเส้นนั้นมีความแตกต่างกันของแต่ละแนวเส้นชัดเจนขึ้น เมื่อนำมาหาค่ามุมแล้วจึงมีค่าความผิดพลาดลดลง

4.3.3 การทดลองที่ 3

ในการทดลองนี้จะเป็นการทดลองหาค่าเฉลี่ยโดยใช้กล้อง 2 ตัว แล้วเปรียบเทียบการเปลี่ยนแปลงของการหมุน ระหว่างกล้อง โดยสมมติให้มีการเปลี่ยนแปลงที่การหมุนรอบแกนเดียว คือที่การหมุนรอบแกน y ทดลองเก็บค่าผลการหามุมที่ระยะและมุมต่างๆ โดยให้การหมุนรอบแกน x และ z คงที่ แล้วเพิ่มที่การหมุนรอบแกน y นำเอาค่าที่ได้ในการถ่ายที่ตำแหน่งของการหมุนรอบแกน y ต่างกันมาเปรียบเทียบกัน แล้วทำการหาค่าเฉลี่ยของค่ามุมที่หาได้ในการหมุนรอบทั้ง 3 แกน

ตารางที่ 4.22 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,0,30)

ภาพ	มุม	ระยะ	รอบแกน X	รอบแกนY	รอบแกนZ
77101	(5,0,30)	(0,0,0)	3.722121	1.094399	30.18344
77102	(5,0,30)	(5,5,0)	4.225127	0.226916	30.07522
77103	(5,0,30)	(5,0,0)	4.45683	0.098066	30.15496
77104	(5,0,30)	(-5,-5,0)	2.469672	0.490078	30.20489
77105	(5,0,30)	(-5,0,0)	3.038618	0.798025	29.83684
77106	(5,0,30)	(4,4,0)	3.926357	0.388414	30.08797
77107	(5,0,30)	(-4,-4,0)	4.413327	0.125521	30.1537
77108	(5,0,30)	(-4,0,0)	5.989634	1.842627	30.68845
77109	(5,0,30)	(0,4,0)	4.749762	0.611643	29.84891
77110	(5,0,30)	(3,3,0)	4.350965	0.262983	30.1873
77111	(5,0,30)	(3,0,0)	3.909846	0.06631	30.11714
77112	(5,0,30)	(-3,-3,0)	5.49902	1.083831	30.31058
77113	(5,0,30)	(-3,0,0)	4.616199	0.712716	29.80235
77114	(5,0,30)	(6,6,0)	5.04065	0.063673	30.25225
77115	(5,0,30)	(6,0,0)	4.174688	0.664438	30.05877
77116	(5,0,30)	(-6,-6,0)	4.734625	0.721821	30.16175
77117	(5,0,30)	(-6,0,0)	4.145682	1.169497	30.00141
77118	(5,0,30)	(7,7,0)	5.058235	0.076003	30.33615
77119	(5,0,30)	(7,0,0)	4.682061	0.751196	30.17622
77120	(5,0,30)	(-7,-7,0)	5.095062	0.551938	30.0706
		เฉลี่ย	**4.414924**	**0.59005**	**30.13544**
		error	**0.972613**	**0.742867**	**0.23397**

ตารางที่ 4.23 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,10,30)

ภาพ	มุม	ระยะ	รอบแกน X	รอบแกนY	รอบแกนZ
76101	(5,10,30)	(0,0,0)	4.412861	7.039106	29.09875
76102	(5,10,30)	(5,5,0)	8.77553	15.29304	30.18151
76103	(5,10,30)	(5,0,0)	7.7144	15.05229	30.73664
76104	(5,10,30)	(-5,-5,0)	4.98755	9.398945	29.6775
76105	(5,10,30)	(-5,0,0)	7.987943	13.63842	29.8433
76106	(5,10,30)	(4,4,0)	8.020574	13.0822	29.39223
76107	(5,10,30)	(-4,-4,0)	6.074612	11.115	29.8524
76108	(5,10,30)	(-4,0,0)	8.624562	14.57864	29.91117
76109	(5,10,30)	(0,4,0)	4.400399	7.845551	29.48683
76110	(5,10,30)	(3,3,0)	9.003719	16.64296	30.64933
76111	(5,10,30)	(3,0,0)	6.039549	11.65227	30.0926
76112	(5,10,30)	(-3,-3,0)	8.355287	15.51358	30.68528
76113	(5,10,30)	(-3,0,0)	5.018815	9.437277	29.75211
76114	(5,10,30)	(6,6,0)	5.993281	10.57161	29.5176
76115	(5,10,30)	(6,0,0)	6.917482	13.48233	30.48712
76116	(5,10,30)	(-6,-6,0)	6.622046	12.75617	30.2088
76117	(5,10,30)	(-6,0,0)	6.192589	10.6778	29.57861
76118	(5,10,30)	(7,7,0)	9.2861	14.3521	29.3197
76119	(5,10,30)	(7,0,0)	4.672369	9.849034	30.25124
76120	(5,10,30)	(-7,-7,0)	3.404355	6.86743	29.6855
		เฉลี่ย	**6.62520**	**11.94229**	**29.92041**
		error	**2.374662**	**3.45773**	**0.47202**

ตารางที่ 4.24 แสดงผลการทดลองที่มุมการหันเหรอบแกน XYZ (5 ,20,30)

ภาพ	มุม	ระยะ	รอบแกน X	รอบแกนY	รอบแกนZ
78101	(5,20,30)	(0,0,0)	4.549406	19.6391	29.71838
78102	(5,20,30)	(5,5,0)	4.348555	18.40644	29.75151
78103	(5,20,30)	(5,0,0)	4.747267	20.35354	30.30182
78104	(5,20,30)	(-5,-5,0)	3.223356	14.02689	28.53315
78105	(5,20,30)	(-5,0,0)	5.978372	22.54653	30.92394
78106	(5,20,30)	(4,4,0)	4.905791	19.2073	29.34274
78107	(5,20,30)	(-4,-4,0)	4.323673	18.21707	29.29684
78108	(5,20,30)	(-4,0,0)	4.138185	17.20182	29.04476
78109	(5,20,30)	(0,4,0)	3.462857	15.56812	29.00075
78110	(5,20,30)	(3,3,0)	4.277981	18.54738	29.6151
78111	(5,20,30)	(3,0,0)	3.091451	15.31897	29.21123
78112	(5,20,30)	(-3,-3,0)	4.17964	17.63317	29.62759
78113	(5,20,30)	(-3,0,0)	5.818601	22.31385	30.81807
78114	(5,20,30)	(6,6,0)	6.261448	23.0983	30.81452
78115	(5,20,30)	(6,0,0)	3.613973	17.29089	29.46295
78116	(5,20,30)	(-6,-6,0)	8.752516	25.32074	29.9688
78117	(5,20,30)	(-6,0,0)	8.200501	23.50317	29.1536
78118	(5,20,30)	(7,7,0)	4.428647	19.3316	29.82325
78119	(5,20,30)	(7,0,0)	6.731257	20.53609	28.46698
78120	(5,20,30)	(-7,-7,0)	3.352294	15.67009	29.18106
		เฉลี่ย	**4.919289**	**19.18655**	**29.60285**
		error	**1.545304**	**3.067716**	**0.786587**

1. เฉลี่ยค่ามุมการหมุนรอบแกน XYZ (5,0,30) เมื่อเทียบกับ (5,10,30) แล้วดูค่าเฉลี่ยเมื่อกล้องเปลี่ยนไป 10 องศา

ตารางที่ 4.25 การหมุนที่มุม (5,0,30) ก่อนการเทียบกับการหมุนที่มุม (5,10,30)

เฉลี่ยมุม (X,Y,Z)	**4.414924**	**0.59005**	**30.13544**
RMSE	**0.972613**	**0.742867**	**0.23397**

ตารางที่ 4.26 การหมุนที่มุม (5,10,30) ก่อนการเทียบกับการหมุนที่มุม (5,0,30)

เฉลี่ยมุม (X,Y,Z)	**6.62520**	**11.94229**	**29.92041**
RMSE	**2.374662**	**3.45773**	**0.47202**

เมื่อทำการเปรียบเทียบแบบเฉลี่ยมุมโดย เทียบที่แกน Y ที่ 0 องศา ให้แกน Y เพิ่มขึ้น 10 องศา เพื่อให้มีค่าเท่ากับการหมุน (5,10,30) แล้วทำการเฉลี่ยค่าในการหมุนรอบแนวแกนทั้ง 3 จะได้

ตารางที่ 4.27 การหมุนที่มุม (5,0,30) หลังจากการเทียบกับการหมุนที่มุม (5,10,30)

เฉลี่ยมุม (X,Y,Z)	**5.52006**	**1.266146**	**30.02793**
RMSE	**1.13367**	**1.90821**	**0.262744**

เมื่อทำการเปรียบเทียบแบบเฉลี่ยมุมโดย เทียบที่แกน Y ที่ 10 องศา ให้แกน Y ลดลง 10 องศา เพื่อให้มีค่าเท่ากับการหมุน (5,0,30) แล้วทำการเฉลี่ยค่าในการหมุนรอบแนวแกนทั้ง 3 จะได้

ตารางที่ 4.28 การหมุนที่มุม (5,0,30) หลังจากการเทียบกับการหมุนที่มุม (5,10,30)

เฉลี่ยมุม (X,Y,Z)	**5.52006**	**11.26615**	**30.02793**
RMSE	**1.13367**	**1.90821**	**0.262744**

2. เฉลี่ยค่ามุมการหมุนรอบแกน XYZ (5,10,30) เมื่อเทียบกับ (5,20,30) แล้วดูค่าเฉลี่ยเมื่อกล้องเปลี่ยนไป 10 องศา

ตารางที่ 4.29 การหมุนที่มุม (5,10,30) ก่อนการเทียบกับการหมุนที่มุม (5,20,30)

เฉลี่ยมุม (X,Y,Z)	**6.62520**	**11.94229**	**29.92041**
RMSE	**2.374662**	**3.45773**	**0.47202**

ตารางที่ 4.30 การหมุนที่มุม (5,20,30) ก่อนการเทียบกับการหมุนที่มุม (5,10,30)

เฉลี่ยมุม (X,Y,Z)	**4.919289**	**19.18655**	**29.60285**
RMSE	**1.545304**	**3.067716**	**0.786587**

เมื่อทำการเปรียบเทียบแบบเฉลี่ยมุมโดย เทียบที่แกน Y ที่ 10 องศา ให้แกน Y เพิ่มขึ้น 10 องศา เพื่อให้มีค่าเท่ากับการหมุน (5,20,30) แล้วทำการเฉลี่ยค่าในการหมุนรอบแนวแกนทั้ง 3 จะได้

ตารางที่ 4.31 การหมุนที่มุม (5,10,30) หลังจากการเทียบกับการหมุนที่มุม (5,20,30)

เฉลี่ย	**5.772245**	**10.56442**	**29.76163**
RMSE	**1.393027**	**2.24423**	**0.479133**

เมื่อทำการเปรียบเทียบแบบเฉลี่ยมุมโดย เทียบที่แกน Y ที่ 20 องศา ให้แกน Y ลดลง 10 องศา เพื่อให้มีค่าเท่ากับการหมุน (5,10,30) แล้วทำการเฉลี่ยค่าในการหมุนรอบแนวแกนทั้ง 3 จะได้

ตารางที่ 4.32 การหมุนที่มุม (5,20,30) หลังจากการเทียบกับการหมุนที่มุม (5,10,30)

เฉลี่ย	**5.772245**	**20.56442**	**29.76163**
RMSE	**1.393027**	**2.24423**	**0.479133**

3. เฉลี่ยค่ามุมการหมุนรอบแกน XYZ (5,10,30) เมื่อใช้การเฉลี่ยของการหมุนที่ (5,0,30) กับ (5,20,30)

ตารางที่ 4.33 การหมุนที่มุม (5,10,30) ที่หาได้

เฉลี่ย	**6.62520**	**11.94229**	**29.92041**
RMSE	**2.374662**	**3.45773**	**0.47202**

การเปรียบเทียบ ที่การหมุน(5,0,30) ที่แกน Y ที่ 0 องศา ให้แกน Y เพิ่มขึ้น 10 องศา เพื่อให้มีค่าเท่ากับการหมุน (5,10,30) และ ที่การหมุน(5,20,30) ที่แกน Y ที่ 20 องศา ให้แกน Y ลดลง 10 องศา เพื่อให้มีค่าเท่ากับการแล้วทำการเฉลี่ยค่าในการหมุนรอบแนวแกนทั้ง 3 จะได้

ตารางที่ 4.34 การหมุนที่มุม (5,10,30) หลังจากการเฉลี่ย

เฉลี่ย	**4.667106**	**9.888278**	**29.86915**
RMSE	**0.944907**	**1.519048**	**0.350769**

4.3.3.1 สรุปผลการทดลองที่ 3

ในการเปรียบเทียบแบบเฉลี่ยนั้นมีความต้องการที่จะดูการเปลี่ยนแปลงของค่า เปรียบเทียบระหว่างกล้อง 2 ตัว เพื่อให้ทราบว่าเมื่อกล้องเปลี่ยนไป กี่องศาแล้ว จะได้ค่าความผิดพลาดเป็นเท่าไหร่ หรือเป็นการดูค่าของมุมที่เราไม่ได้ทำการถ่ายภาพมา เช่น ถ่ายที่ (5,0,30) กับ (5,20,30) ก็สามารถหาว่าถ้ากล้องถ่ายที่ (5,10,30) จะหาค่ามุมออกมาได้เป็นเช่นไร มีค่า error เท่าใด

4.4 สรุปผลการทดลอง

จากการทดลองทั้ง 3 สามารถสรุปได้ว่าสิ่งที่มีผลต่อความคลาดเคลื่อนของผลการหาค่ามุมการหันเหของกล้องวีดีโอ คือ

1. จำนวนของเส้นและจำนวนของจุดภาพในแต่ละแนวเส้นมีผลต่อความถูกต้องของค่ามุมการหันเหของกล้อง พบว่าหากภาพที่นำมาทดสอบนั้นมีจำนวนของแนวเส้นที่พบมากพอสมควรและมีความชัดเจนของเส้นที่สมบูรณ์ รวมทั้งเส้นที่พบมีจำนวนของจุดภาพมากพอควรก็จะทำให้ผลการหาค่ามุมการหันเหของกล้องถูกต้องมากยิ่งขึ้น

2. เมื่อองศาของมุมการหันเหของกล้องเพิ่มขึ้นจะทำให้ความผิดพลาดของค่าที่หาได้ของมุมการหมุนรอบแกน x และแกน y มีค่าลดลง

3. การเฉลี่ยค่าการหามุมการหันเหของกล้องโดยใช้การหาค่าการหันเหจากกล้อง 2 ตัวที่ถ่ายในตำแหน่งการทำมุมที่ต่างกันจะทำให้ค่าความคลาดเคลื่อนของค่ามุมการหันเหลดลง

สำหรับการศึกษาและทดลองนี้เป็นการทดลองกับภาพที่ทำการถ่ายทำด้วยโปรแกรม 3d Studio Max ดังนั้นจึงเป็นภาพที่มีความชัดเจนของเส้นพอสมควร จึงบอกได้ว่าสิ่งที่มีผลต่อความถูกต้องของค่ามุมการหันเหที่หาได้คือ จำนวนจุดภาพบนแนวเส้นแต่ละเส้น ซึ่งการตัดส่วนของแนวเส้นแต่ละเส้นในการนำมาหาค่าความชัน และสมการเส้นตรงต้องทำให้ได้ความสมบูรณ์ของจุดภาพบนเส้นมากที่สุด และส่วนที่เป็นจุดที่มีการตัดกันของเส้นจะต้องทำการตัดออกให้เหลือแต่จุดภาพที่อยู่ในแนวเส้นที่จะวิเคราะห์ให้ได้จุดภาพมากที่สุดและมีความใกล้เคียงกับแนวเส้นนั้นมากที่สุด

4.5 สรุปผลการวิจัย

จากการวิจัยที่ได้ทำมาทั้งหมด โดยใช้ฟังก์ชั่นที่สร้างมาโดยโปรแกรม matlab แล้วทำการทดสอบกับภาพที่สร้างขึ้นมาด้วยโปรแกรม 3D Studio Max 6 พบว่าในส่วนของการอ่านแนวเส้นและจุดภาพนั้นเป็นกระบวนการที่สำคัญอย่างยิ่งที่จะทำให้ค่ามุมการเปลี่ยนแปลงไปของกล้องนั้นมีความถูกต้อง ซึ่งสามารถที่จะใช้ เทคนิคของ Hough Transform หาแล้วมาใช้กระบวนการของการไล่หาจุดภาพแล้วใช้วิธี Least Square เพิ่มความถูกต้องในการหาค่าพารามิเตอร์เพื่อนำมาสร้างเป็นสมการเส้นตรง แต่จำนวนจุดที่นำมาคิดในวิธีนี้จะต้องมากพอสมควรเพื่อที่จะให้ผลที่ถูกต้องมากยิ่งขึ้น นอกจากนี้เป็นแล้วค่าที่ออกมาอาจจะมีการคลาดเคลื่อนไปบ้างเนื่องจากการประมาณค่าของจุดภาพซึ่งจะต้องอ่านจุดภาพด้วยจำนวนเต็มจึงต้องมีการตัดส่วนที่เป็นทศนิยมทิ้งไป ซึ่งจะต้องหากรรมวิธีในการปรับปรุงและแก้ไขให้ได้ค่าที่เหมาะสมยิ่งขึ้น

บทที่ 5
สรุปผลการศึกษาและข้อเสนอแนะ

5.1 สิ่งที่ได้จากการวิจัย

จากการศึกษาและวิจัยในเรื่องการหาค่ามุมและตำแหน่งของกล้องวีดีโอจากฉากที่เป็น Binary Pattern นั้น ผู้ศึกษาวิจัยได้ทำการสร้างฟังก์ชันที่สามารถนำมาใช้ในกระบวนการวิเคราะห์และประมวลผลภาพด้วยโปรแกรม matlab ทำให้ได้ชุดฟังก์ชันที่สามารถนำมาใช้วิเคราะห์ภาพที่สร้างขึ้นมาด้วยเทคนิคของคอมพิวเตอร์กราฟิก ในโปรแกรม 3D Studiomax 6 ทั้งนี้ยังได้ความรู้เพิ่มเติมในเรื่องเหล่านี้ด้วย

- เทคนิคการสร้าง Blue Screen Binary Map
- เทคนิค Hough Transform
- เทคนิค Least Square
- กระบวนการทาง image processing

5.2 ปัญหาและอุปสรรค

ปัญหาและอุปสรรคที่เกิดขึ้นในการศึกษาวิจัย สามารถจำแนกเป็นข้อๆ ได้ ดังต่อไปนี้

1. ปัญหาเรื่องแหล่งของหนังสืออ้างอิงเกี่ยวกับกระบวนการ image processing นั้นหาได้ยากและมีน้อย ทำให้ต้องทำการศึกษาค้นคว้าจากแหล่งข้อมูลทางอินเตอร์เน็ตเท่านั้น
2. ปัญหาในเรื่องเครื่องคอมพิวเตอร์ที่นำมาใช้ในการศึกษาวิจัย เนื่องจากการดำเนินการทางด้านคอมพิวเตอร์กราฟิกและ image processing นั้น จำเป็นต้องใช้ความสามารถและทรัพยากรในการประมวลผลสูง เครื่องคอมพิวเตอร์ที่นำมาใช้ต้องเป็นเครื่องที่มีคุณภาพสูงพอสมควร

5.3 ข้อเสนอแนะ

เพื่อเป็นประโยชน์ต่อผู้ที่ศึกษาวิจัยในเรื่องนี้ จึงจะต้องขอเสนอข้อเสนอแนะเพื่อใช้ปรับปรุงคุณภาพของอัลกอริทึม และการดำเนินการบางอย่างเพื่อให้การวิจัยได้รับประสิทธิภาพสูงขึ้น

1. ภาพที่นำมาใช้ในการทดสอบควรเป็นภาพที่มีความชัดเจนพอสมควร เพื่อที่จะทำให้ได้ผลลัพธ์ที่มีความถูกต้องมากยิ่งขึ้น
2. การคำนวณทางคณิตศาสตร์บางอย่างในกระบวนการทดสอบนั้นมีบางครั้งที่ต้องมีการปัดเศษของค่า ซึ่งทำให้ผลลัพธ์ที่ได้เกิดความคลาดเคลื่อน จึงควรมีการหากรรมวิธีที่จะสามารถแก้ปัญหาในเรื่องนี้เพื่อให้ผลลัพธ์ที่ได้มีความน่าเชื่อถือได้มากยิ่งขึ้น
3. ภาพที่นำมาทดสอบควรเป็นภาพที่ถ่ายในมุมที่สามารถวิเคราะห์แยกกลุ่มของเส้นได้โดยจำเป็นต้องให้ขนาดของบล็อกของ pattern ในภาพมีขนาดใหญ่พอสมควร
4. เนื่องจากในรายงานการวิจัยฉบับนี้ยังไม่ได้กล่าวถึงการหาตำแหน่งของกล้องจึงเป็นส่วนที่จะต้องมีการศึกษาและพัฒนาต่อไป

บรรณานุกรม

Gregory, A.Baxes. 1994. **Digital Image Processing Principle and Application.** John Wiley & Sons.

Xirouhakis, Y. et al. 2001 **Efficient Optical Camera Tracking in Virtual Sets.** IEEE transaction on Image Processing,Vol 10.

Xirouhakis, Y. et al. 2000 **Optical Camera Tracking in Virtual Studio: Degenerate Case.** IEEE.

ภาคผนวก

ภาคผนวก ก
อธิบายฟังก์ชัน

1. Function main ()

เป็นฟังก์ชั่นหลักที่เอาไว้สำหรับรวบรวมฟังก์ชันต่างต่างๆที่ใช้ในการทำงาน โดยจะรับค่า

Input

ตำแหน่ง path ที่เก็บภาพที่จะทำการวิเคราะห์เอาไว้

Output

ค่าพารามิเตอร์ต่างๆ ที่ต้องการ

ตัวอย่างฟังก์ชัน

```
function [degreeX,degreeY,degreeZ]  = main(path)

% create  binary map
primpoly1 = [1 0 0 1 0 1];
input1 = [0 0 0 0 1];
primpoly2 = [1 1 12];
input2 = [0 1];
MAP =  createmap(primpoly1,input1,primpoly2,input2,31,1,45);



%-----------------------------------------------------------------%
I = imread(path);
[IMG,centerX,centerY] = cutframe(I,200,150);   % cut blue area
[thresh,GI] = twocolor(IMG);
[X] = edgeIMG(IMG);      % X = edge of picture
theta = (0:179)';
[xf,yf,radslp] = radontran(X,theta,0.50);    % output of radon (position of image [x1,x2] and
[y1,y2] in size of image )
[rad1,rad2] = groupslope(radslp); % separated slope to 2 groups
```

```
sa1 = sortrows(rad1',2);          % sort by b
sa2 = sortrows(rad2',2);
[rx,ry] = intersec(sa1,sa2);
XM = cutIntersec(X,rx,ry);
%-------------------------------------------------------------------------%
slpline = leastline(XM,xf,yf,radslp,centerX,centerY);   % slope by leastsquare
[a1,a2] = groupslope(slpline);      % separated slope to 2 groups
sa1 = sortrows(a1',2);          % sort by b
sa2 = sortrows(a2',2);
ss1 = sortE(sa1);
ss2 = sortE(sa2);
Ev1 = linevt(ss1);              %  set of unit-norm vector (vertical)
Eh1 = linevt(ss2);              %  set of unit-norm vector (horizontal)
f = calfocus(Ev1,Eh1);
Ev = linevtF(ss1,f);            %  set of unit-norm vector (vertical)
Eh = linevtF(ss2,f);            %  set of unit-norm vector (horizontal)
[Q,cn,cm] = crossvecnorm(Ev,Eh);          %  cross product of Ev and Eh
[U,S,V] = svd(Q,0);             % sigular value decomposition
V = getrotate(V,S,cn,cm);       %  get r from V
[LLv,LLh] = lostlineHV(Ev,Eh,V);       % find lost line

%-------------------------------------------------------------------------%
[rx,ry] = intersec(sa1,sa2);          % find intersection point
[cp1,ch1] = crosspix(GI,rx,ry,thresh,centerX,centerY);    % find  binary pattern
cp2 = crosspix2(GI,rx,ry,thresh,centerX,centerY);
cp = crosspix3(cp1,cp2,ch1);
 [px,py] = posinMap(cp,MAP);
[degreeX,degreeY,degreeZ] = degreeXYZ(V);    % find degree (X,Y,Z)
```

2. Function createmap ()

เป็นฟังก์ชั่นที่เอาไว้สร้าง binary map

Input

primpoly1	=	primitive polynomial ของ binary sequence
input1	=	ค่าเริ่มต้นที่ใช้สร้าง binary sequence
primpoly2	=	primitive polynomial ของ nonbinary sequence
input2	=	ค่าเริ่มต้นที่ใช้สร้าง nonbinary sequence

Output

Map	=	matrix ที่เก็บ binary map

ตัวอย่างฟังก์ชัน

```
function   map = createmap(primpoly1,input1,primpoly2,input2,p,beg,en)
        bi = Mseq(primpoly1,input1);     % create binary sequence
        nbi = nbis(primpoly2,input2,p);  %  create  nonbinary sequence
        map = bimap(bi,nbi,beg,en);      %  create map
```

3. Function cutframe ()

เป็นฟังก์ชั่นที่เอาไว้ค้นหาส่วนพื้นที่สีน้ำเงินที่ต้องการ

Input

I	=	ภาพที่จะวิเคราะห์เป็นแบบ RGB
SizeX	=	ขนาดในแนวแกน x ของส่วนสีน้ำเงินที่ต้องการ
SizeY	=	ขนาดในแนวแกน y ของส่วนสีน้ำเงินที่ต้องการ

Output

IMG	=	matrix ที่เก็บ ภาพพื้นหลังสีน้ำเงินที่ตัดออกมา
CenterX	=	จุดศูนย์กลางในแนวแกน x ของภาพต้นฉบับ
CenterY	=	จุดศูนย์กลางในแนวแกน y ของภาพต้นฉบับ

4. Function edgeIMG ()

ใช้สำหรับหาขอบของบล็อก

Input

IMG	=	matrix ที่เก็บ ภาพพื้นหลังสีน้ำเงินที่ตัดออกมา

Output

X	=	matrix ที่เก็บ ภาพเส้นขอบของบล็อก

5. Function radontran ()

เป็นฟังก์ชั่นที่ทำการหาแนวเส้นด้วยเทคนิค Hough Transform

Input

X	=	matrix ที่เก็บ ภาพเส้นขอบของบล็อก
theta	=	(0:179)'
per	=	เปอร์เซนต์ของการตัดค่าความสว่างเทียบกับความสว่างสูงที่สุด

Output

Xf	=	ตำแหน่ง (x_1 , x_2) ซึ่งเป็นจุดปลายของแนวเส้นแต่ละเส้นที่หาได้
Yf	=	ตำแหน่ง (y_1 , y_2) ซึ่งเป็นจุดปลายของแนวเส้นแต่ละเส้นที่หาได้
Radslp	=	ค่า slope a และ จุดตัดแกน b ของแนวเส้นแต่ละเส้นที่หาโดยแนวเส้นที่หาได้จาก Radon Transform (Hough Transform)

6. Function leastline ()

เป็นฟังก์ชั่นที่ทำการหาแนวเส้นด้วยวิธี least square

Input

X	=	matrix ที่เก็บ ภาพเส้นขอบของบล็อก
Xf	=	ตำแหน่ง (x_1 , x_2) ซึ่งเป็นจุดปลายของแนวเส้นแต่ละเส้นที่หาได้
Yf	=	ตำแหน่ง (y_1 , y_2) ซึ่งเป็นจุดปลายของแนวเส้นแต่ละเส้นที่หาได้
Radslp	=	ค่า slope a และ จุดตัดแกน b ของแนวเส้นแต่ละเส้นที่หาโดยแนวเส้นที่หาได้จาก Radon Transform (Hough Transform)
CenterX	=	จุดศูนย์กลางในแนวแกน x ของภาพต้นฉบับ
CenterY	=	จุดศูนย์กลางในแนวแกน y ของภาพต้นฉบับ

Output

slpline = ค่า slope a และ จุดตัดแกน b ของแนวเส้นแต่ละเส้นที่หาโดยแนวเส้นที่หาได้จากวิธี least square

7. Function groupslope ()

ใช้สำหรับการแบ่งแนวเส้นออกเป็น 2 กลุ่ม

Input

slpline = ค่า slope a และ จุดตัดแกน b ของแนวเส้นแต่ละเส้นที่หาโดยแนวเส้นที่หาได้จากวิธี least square

Output

a1 = ค่า slope a และ จุดตัดแกน b ของแนวเส้นแต่ละเส้นที่หาโดยแนวเส้นที่หาได้จากวิธี least square กลุ่มที่ 1

a2 = ค่า slope a และ จุดตัดแกน b ของแนวเส้นแต่ละเส้นที่หาโดยแนวเส้นที่หาได้จากวิธี least square กลุ่มที่ 2

8. Function linevt ()

แปลงสมการเส้นตรงเป็น unit-norm vector แบบไม่คิดความยาวโฟกัส

Input

ss1 หรือ ss2 = ค่า slope a และ จุดตัดแกน b ของแนวเส้นแต่ละเส้นที่หาโดยแนวเส้นที่หาได้จากวิธี least square กลุ่มที่ 1 หรือ กลุ่มที่ 2

Output

Ev1 หรือ Eh1 = ค่า Ev หรือ ค่า Eh แบบที่ยังไม่ได้คิดความยาวโฟกัส

9. Function calfocus ()

หาความยาวโฟกัส

Input

Ev1 และ Eh1 = ค่า Ev และ ค่า Eh แบบที่ยังไม่ได้คิดความยาวโฟกัส

Output

f = ค่าความยาวโฟกัส

10. Function linevt F()

แปลงสมการเส้นตรงเป็น unit-norm vector แบบคิดความยาวโฟกัส

Input

ss1 หรือ ss2	=	ค่า slope a และ จุดตัดแกน b ของแนวเส้นแต่ละเส้นที่หาโดยแนวเส้นที่หาได้จากวิธี least square กลุ่มที่ 1 หรือ กลุ่มที่ 2
f	=	ค่าความยาวโฟกัส

Output

Ev หรือ Eh	=	ค่า Ev หรือ ค่า Eh แบบที่คิดความยาวโฟกัส

11. Function crossvecnorm()

เป็นฟังก์ชั่นที่ใช้ในการ cross vector

Input

Ev และ Eh	=	ค่า Ev และ ค่า Eh แบบที่คิดความยาวโฟกัส

Output

Q	=	ค่า Q ซึ่งเก็บการ cross ของ vector ทั้งสองแนว
cn	=	จำนวนการ cross กันของเส้นในแนวตั้ง (Ev)
cm	=	จำนวนการ cross กันของเส้นในแนวนอน(Eh)

12. Function getrotate ()

เป็นฟังก์ชั่นที่ใช้ในการสลับคอลัมน์ของ rotation matrix ที่หามาได้

Input

V	=	rotation matrix ที่หาได้
S	=	เมทริกซ์ S ที่ได้จาก svd(Q)
cn	=	จำนวนการ cross กันของเส้นในแนวตั้ง (Ev)
cm	=	จำนวนการ cross กันของเส้นในแนวนอน(Eh)

Output

V	=	rotation matrix ที่ทำการปรับตำแหน่งคอลัมน์แล้ว

13. Function lostlineHV ()

เป็นฟังก์ชั่นที่ใช้ในการหาตำแหน่งของแนวเส้นที่ขาดหายไป

Input

Ev และ Eh	=	ค่า Ev และ ค่า Eh แบบที่คิดความยาวโฟกัส
V	=	rotation matrix ที่ทำการปรับตำแหน่งคอลัมน์แล้ว

Output

LLv	=	เส้นที่ขาดหายไปในแนวตั้ง
LLh	=	เส้นที่ขาดหายไปในแนวนอน

14. Function intersec ()

เป็นฟังก์ชั่นที่ใช้ในการหาตำแหน่งของจุดตัดกันของเส้น

Input

Sa1 และ Sa2	=	ค่าslope a และ จุดตัดแกน b ที่เรียงลำดับตามค่า b

Output

rx	=	ตำแหน่ง x ของจุดตัด
ry	=	ตำแหน่ง y ของจุดตัด

15. Function cutIntersec()

ใช้ในการตัดจุดที่เป็นบริเวณที่เป็นจุดตัดกันของเส้นตรง

Input

X	=	ภาพที่ผ่านการหาขอบของบล็อกแล้ว
rx	=	ตำแหน่ง x ของจุดตัด
ry	=	ตำแหน่ง y ของจุดตัด

Output

XM	=	ภาพที่ผ่านการตัดส่วนที่เป็นจุดตัดกันของเส้น

16. Function crosspix ()

เป็นฟังก์ชั่นที่ใช้ในการหาตำแหน่งของจุดตัดกันของเส้น

Input

GI	=	ภาพที่นำมาวิเคราะห์แบบ gray scale
thresh	=	ค่าที่ใช้เป็นค่าที่แบ่งความสว่างของภาพเป็น 2 กลุ่ม
rx	=	ตำแหน่ง x ของจุดตัด
ry	=	ตำแหน่ง y ของจุดตัด
CenterX	=	จุดศูนย์กลางในแนวแกน x ของภาพต้นฉบับ
CenterY	=	จุดศูนย์กลางในแนวแกน y ของภาพต้นฉบับ

Output

Cp	=	ส่วนของ submatrix ที่หาได้มีค่าเป็น binary

17. Function posinMAP ()

เป็นฟังก์ชั่นที่ใช้ในการหาตำแหน่งของ submatrix ในฉาก

Input

Cp	=	ส่วนของ submatrix ที่หาได้มีค่าเป็น binary
Map	=	matrix ที่เก็บ binary map

Output

Px	=	ตำแหน่ง x ของมุมบนด้านซ้ายมือของ submatrix
Py	=	ตำแหน่ง x ของมุมบนด้านซ้ายมือของ submatrix

ภาคผนวก ข

ตัวอย่างฟังก์ชั่น

1. function cut2()

เป็นฟังก์ชั่นที่ใช้ในการ segmentation แยกส่วนของภาพวัตถุและพื้นหลังออกจากกัน

Input

I	=	เมทริกซ์ที่เก็บภาพที่จะนำมา segmentation

Output

amap	=	เมทริกซ์ที่เก็บภาพส่วนพื้นหลังที่แยกออกมา
bmap	=	เมทริกซ์ที่เก็บภาพส่วนวัตถุที่แยกออกมา
errmap	=	เมทริกซ์ที่เก็บภาพส่วนที่ยากต่อการวิเคราะห์แยกส่วน

```
function  [amap ,bmap,errmap] = cut2(I)
% เก็บค่า RGB ของแต่ละจุดภาพ
  r = I(:,:,1);
  g = I(:,:,2);
  b = I(:,:,3);
  figure(1)
  imshow(I);
  [sx cx] = size(r); % หาขนาดของภาพ


% ทำการวิเคราะห์ทีละจุดภาพ
  for i = 1 : sx
    for j = 1 : cx
      % หาผลรวมของค่า RGB ในแต่ละจุดภาพ
       rgb = double(r(i,j))+double(g(i,j))+double(b(i,j));

      % ถ้าน้อยกว่า 50 ให้เก็บใน errmap เป็น 1 นอกนั้นเป็น 0
       if(rgb < 50)
          errmap(i,j) = 1;
       else
          errmap(i,j) = 0;
       end

      % หาค่าอัตราส่วนของค่าสีน้ำเงินเทียบกับค่า RGB
       if rgb == 0
         rgb = 1;   % หลีกเลี่ยง devide by zero (ให้เป็น 1 )
       end
       rat = (double(b(i,j))/rgb)*100;
```

```
% ถ้าอัตราส่วนของสีน้ำเงินน้อยกว่าร้อยละ 50
        if (rat < 50)

% amap เก็บภาพส่วนพื้นหลังให้จุดภาพนี้เป็นสีขาว
                    amap(i,j,1) = uint8(255);
    amap(i,j,2) = uint8(255);
    amap(i,j,3) = uint8(255);

                    % bmap เก็บภาพส่วนวัตถุให้เก็บค่า RGB ตามเดิม
                                bmap(i,j,1) = r(i,j);
    bmap(i,j,2) = g(i,j);
    bmap(i,j,3) = b(i,j);

   else

   % amap เก็บภาพส่วนพื้นหลังให้เก็บค่า RGB ตามเดิม
                            amap(i,j,1)=r(i,j);
    amap(i,j,2)=g(i,j);
    amap(i,j,3)=b(i,j);


    % bmap เก็บภาพส่วนวัตถุให้จุดภาพนี้เป็นสีขาว
    bmap(i,j,1) = uint8(255);
    bmap(i,j,2) = uint8(255);
    bmap(i,j,3) = uint8(255);
   end
  end
 end
end
figure(2)
imshow(amap);
figure(3)
imshow(errmap);
figure(5)
imshow(bmap);
```

2. function cutframe()

ใช้สำหรับตัดส่วนพื้นหลังสีน้ำเงินตามขนาดที่ต้องการ ซึ่งต้องกำหนดขนาดที่ต้องการไว้ก่อน

input

p	=	เก็บภาพ RGB ส่วนของพื้นหลัง (ให้ส่วนที่เป็นวัตถุเป็นสีขาว)
rx	=	ขนาดในแนวแกน x ของส่วนพื้นหลังสีน้ำเงินที่ต้องการ
ry	=	ขนาดในแนวแกน y ของส่วนพื้นหลังสีน้ำเงินที่ต้องการ

output

BB	=	ส่วนพื้นหลังสีน้ำเงินตามขนาดที่ต้องการ
CenterX	=	จุดศูนย์กลางภาพในแกน x
CenterY	=	จุดศูนย์กลางภาพในแกน y

```
function [BB,centerX,centerY] = cutframe(p,ry,rx)

  M(:,:) = 0;
  I = imread(p);              % อ่านภาพ p มาเก็บไว้ใน I
   [sr,sc,map] = size(I);                  % หาขนาดของB

  % แยกค่าสี RGB ในแต่ละจุดภาพ

  R(:,:) = I(:,:,1);          % เก็บค่าสีแดง (R) ของแต่ละจุดภาพในภาพ I ไว้ใน R
  G(:,:) = I(:,:,2); % เก็บค่าสีเขียว (G) ของแต่ละจุดภาพในภาพ I ไว้ใน G
  B(:,:) = I(:,:,3); % เก็บค่าสีน้ำเงิน(B) ของแต่ละจุดภาพในภาพ I ไว้ใน B


  for i = 1:sr
   for j = 1:sc
      if (r(i,j)== uint8(255))&(g(i,j)== uint8(255))&
          (b(i,j)== uint8(255))
        B1(i,j)= 1;           % ให้จุดภาพที่มีค่า RGB เป็น (255:255:255) มีค่าเป็น 1 เก็บไว้
                              ใน B1
      else
        B1(i,j) = 0;          % ให้จุดภาพที่มีค่า RGB ไม่เป็น(255:255:255) มีค่าเป็น 0 เก็บ
                              ไว้ใน B1
      end;
    end;
   end;


   M2 = bwmorph(B1,'majority');          % กำจัดจุดภาพที่กระจายออกจากกลุ่ม

    figure(3)
   imshow(M2);

   [ML1,num1] = bwlabel(M2);                % จัดกลุ่มของจุดภาพ


   if num1 == 0          % ถ้าจำนวนกลุ่มของจุดภาพเป็น 0 (แสดงว่ามีแต่ส่วนของพื้นหลัง)
     % เก็บตำแหน่งของส่วนพื้นหลังสีน้ำเงินที่จะตัดไปวิเคราะห์
```

```
% ในกรณีนี้จะตัดส่วนมุมบนด้านซ้ายตามขนาดที่ต้องการ

ay1 = 1;
ay2 = ry;
ax1 = 1;
ax2 = rx;


else        % จำนวนของกลุ่มของจุดภาพไม่เป็น 0 (มีทั้งส่วนพื้นหลังและส่วนวัตถุ)
      % ใช้ฟังก์ชั่น boundbox() หาแนวการอ่านจุดภาพในแนวแกน x และ y
       [B2,ty,tx] = boundbox(ML1,num1);
       %ใช้ฟังก์ชั่น croparea() หาตำแหน่งของส่วนสีน้ำเงินที่ได้ขนาดตามต้องการ
       [ay1,ay2,ax1,ax2] = croparea(B1,ty,tx,ry,rx);
end;

% ตัดส่วนพื้นหลังสีน้ำเงินจาก I ตามตำแหน่งที่หาได้
BB = I(ay1:ay2,ax1:ax2,:);

% จุดศูนย์กลางของภาพที่ถ่ายมา
[rs,cs,map] = size(I);
centerX = cs/2;                 % x = column
centerY = rs/2;                 % y = row
```

3. function boundbox()

เป็นฟังก์ชั่นหนึ่งที่จะถูกเรียกใช้ในฟังก์ชั่น cutframe() ใช้สำหรับหาแนวการอ่านจุดภาพ

input

ML1	=	ภาพที่มีการใส่หมายเลขของกลุ่มของจุดภาพไว้
num	=	จำนวนของกลุ่มของจุดภาพ

output

B2	=	ภาพที่มีการตีกรอบสีขาวในส่วนที่เป็นวัตถุ
tx	=	แนวการอ่านจุดภาพในแนวแกน x
ty	=	แนวการอ่านจุดภาพในแนวแกน y

```
function [B2,ty,tx] = boundbox(ML1,num)
   [by,bx] = size(ML1);
   B1 = ML1;
   count = 1;
   k = 1;
```

```
% ทำการหาแนวเส้นของการอ่านจุดภาพที่ละกลุ่มของจุดภาพ
for lb = 1:num
   [y1,x1]= find(ML1 == lb); %หาตำแหน่ง (x,y) ที่หมายเลขกลุ่มของจุดภาพเท่ากับ lb
    srty = sort(y1);          % เรียงลำดับตำแหน่งในแนวแกน y ที่หาได้ในกลุ่มนี้
    srtx = sort(x1);          % เรียงลำดับตำแหน่งในแนวแกน x ที่หาได้ในกลุ่มนี้

    mx(lb,1) = max(srty);     % หาตำแหน่ง y ที่มากที่สุดในกลุ่มนี้

     % ถ้าตำแหน่ง y ไม่เท่ากับ 0 และ ไม่เท่ากับขนาดของภาพที่ถ่ายมา เก็บตำแหน่งของ y ไว้เป็น
       แนวการอ่านจุดภาพในแนวแกน y
    if (mx(lb,1) ~= 0)&(mx(lb,1) ~= by)
        ty(count) = mx(lb,1);
        count = count + 1;
    end;


    mx(lb,2) = max(srtx);     % หาตำแหน่ง x ที่มากที่สุดในกลุ่มนี้

     % ถ้าตำแหน่ง x ไม่เท่ากับ 0 และ ไม่เท่ากับขนาดของภาพที่ถ่ายมา เก็บตำแหน่งของ x ไว้เป็น
       แนวการอ่านจุดภาพในแนวแกน x
    if (mx(lb,2) ~= 0)&(mx(lb,2) ~= bx)
        tx(k) = mx(lb,2);
        k = k + 1;
    end;


    mn(lb,1) = min(srty);     % หาตำแหน่ง y ที่น้อยที่สุดในกลุ่มนี้

     % ถ้าตำแหน่ง y ไม่เท่ากับ 0 และ ไม่เท่ากับขนาดของภาพที่ถ่ายมา เก็บตำแหน่งของ y ไว้เป็น
       แนวการอ่านจุดภาพในแนวแกน y

    if (mn(lb,1) ~= 0)&(mn(lb,1) ~= by)
        ty(count) = mn(lb,1);
        count = count + 1;
    end;


    mn(lb,2) = min(srtx);     % หาตำแหน่ง y ที่น้อยที่สุดในกลุ่มนี้

     % ถ้าตำแหน่ง x ไม่เท่ากับ 0 และ ไม่เท่ากับขนาดของภาพที่ถ่ายมา เก็บตำแหน่งของ x ไว้เป็น
       แนวการอ่านจุดภาพในแนวแกน x

    if (mn(lb,2) ~= 0)&(mn(lb,2) ~= bx)
        tx(k) = mn(lb,2);
        k = k + 1;
```

```
        end;

        % สร้างกรอบสี่เหลี่ยมสีขาวในตำแหน่งของกลุ่มจุดภาพนี้
        B2(mn(lb,1):mx(lb,1), mn(lb,2):mx(lb,2))= 1;
    end;
```

4. function croparea()

เป็นฟังก์ชั่นหนึ่งที่จะถูกเรียกใช้ในฟังชั่น croparea() ใช้สำหรับตัดส่วนพื้นหลังตามขนาดที่กำหนดแล้วหาว่าเป็นส่วนพื้นหลังสีน้ำเงินตามหรือไม่

input

B1	=	ภาพขาวดำที่ส่วนวัตถุเป็นสีขาวส่วนพื้นหลังเป็นสีดำ
tx	=	แนวการอ่านจุดภาพในแนวแกน x
ty	=	แนวการอ่านจุดภาพในแนวแกน y
rx	=	ขนาดในแนวแกน x ของส่วนพื้นหลังสีน้ำเงินที่ต้องการ
ry	=	ขนาดในแนวแกน y ของส่วนพื้นหลังสีน้ำเงินที่ต้องการ

output

ay1	=	ตำแหน่งของ y1 ของส่วนสีน้ำที่ได้ขนาดตามต้องการ
ay2	=	ตำแหน่งของ y1 ของส่วนสีน้ำที่ได้ขนาดตามต้องการ
ax1	=	ตำแหน่งของ x1 ของส่วนสีน้ำที่ได้ขนาดตามต้องการ
ax2	=	ตำแหน่งของ x2 ของส่วนสีน้ำที่ได้ขนาดตามต้องการ

```
function [ay1,ay2,ax1,ax2] = croparea(B1,ty,tx,ry,rx)
    stx = sort(tx);     % เรียงลำดับของแนวการอ่านจุดภาพในแนวแกน x จากน้อยไปมาก
    sty = sort(ty);     % เรียงลำดับของแนวการอ่านจุดภาพในแนวแกน y จากน้อยไปมาก
    lngthy = length(ty);      % หาจำนวนของจุดที่เป็นแนวการอ่านจุดภาพ
    lngthx = length(tx);
    [sr,sc] = size(B1);       % หาขนาด B1 (ภาพที่รับเข้ามา)
    check = 1;                % ยังไม่พบส่วนสีน้ำเงินตามขนาดที่ต้องการ
    BB = B1(1:ry,1:rx);       % อ่านจุดภาพมาเก็บไว้ใน BB
    if sum(BB)== 0            % ถ้าผลรวมของค่าสีในจุดภาพเป็น 0
       check = 2;             % ให้ check = 2 แสดงว่าเจอส่วนพื้นหลังสีน้ำเงินที่ต้องการแล้ว
       ay1 = 1;
       ay2 = ry;
       ax1 = 1;
       ax2 = rx;
    end;
```

```
 if check == 1

% เลื่อนการอ่านจุดภาพทีละตำแหน่งตาม sty
% ให้ตำแหน่งการอ่านในแนวแกน x เป็น 1
    for i = 1:lngthy
        yy = sty(i)+1;        % ค่าตำแหน่งของแนวการอ่านจุดภาพในแนวแกน y +1
        if (yy + ry) < sr
            BB =  B1(yy :(yy + ry),1:rx);
            summ = sum(BB);
            if summ == 0
                check = 2;
                ay1 = yy;
                ay2 = yy + ry;
                ax1 = 1;
                ax2 = rx;
            end;
        end;
    end;
end;


if check == 1

% เลื่อนการอ่านจุดภาพทีละตำแหน่งตาม stx
% ให้ตำแหน่งการอ่านในแนวแกน y เป็น 1

    for j = 1:lngthx
        xx = stx(j)+1;   % ค่าตำแหน่งของแนวการอ่านจุดภาพในแนวแกน x +1
        if (xx + rx) < sc
            BB =  B1(1:ry, xx :(xx + rx));
            summ = sum(BB);
            if summ == 0
                check = 2;
                ay1 = 1;
                ay2 = ry;
                ax1 = xx;
                ax2 = xx + rx;
            end;
        end;
    end;
end;

if check == 1

% เลื่อนการอ่านจุดภาพทีละตำแหน่งตาม sty
% เลื่อนการอ่านจุดภาพทีละตำแหน่งตาม stx

    for i = 1:lngthy
       for j = 1:lngthx
           yy = sty(i)+1; % ค่าตำแหน่งของแนวการอ่านจุดภาพในแนวแกน x +1
```

```
                xx = stx(j)+1; % ค่าตำแหน่งของแนวการอ่านจุดภาพในแนวแกน x +1
                 if ((yy + ry) < sr)&((xx + rx) < sc)
                     BB =  B1(yy :(yy + ry), xx :( xx + rx));
                         if sum(BB)== 0
                             check = 2;
                             ay1 = yy;
                             ay2 = yy + ry;
                             ax1 = xx;
                             ax2 = xx + rx;
                         end;
                 end;
             end;
         end;
     end;
```

5. function intersec()

ใช้สำหรับหาจุดตัดของแนวเส้นตรง

Input

sa1 = ค่าความชัน a และจุดตัดแกน b ของแนวเส้นแนวนอน

sa2 = ค่าความชัน a และจุดตัดแกน b ของแนวเส้นแนวตั้ง

Output

X = ตำแหน่งในแนวแกน x ของจุดตัดของแนวเส้นตรง

Y = ตำแหน่งในแนวแกน y ของจุดตัดของแนวเส้นตรง

```
function [X,Y] = intersec(sa1,sa2)
    [ra1,ca1] = size(sa1);
    [ra2,ca2] = size(sa2);
    % ทำการหาจุดตัดของแนวเส้นที่ละคู่ของเส้นแนวตั้งกับแนวนอน
    for i = 1:ra1
        for j = 1:ra2
          % หา x = (b2-b1)/(a1-a2)
                    X(i,j) = (sa2(j,2) - sa1(i,2)) / (sa1(i,1) -
sa2(j,1));
          % หา y = ax + b จะได้ จุดตัดของเส้นในแนวแกน y
                     Y(i,j) = sa1(i,1)*X(i,j) + sa1(i,2);
        end;
    end;
```

6. function subfline()

เป็นฟังก์ชั่นที่จะถูกเรียกใช้ในฟังก์ชั่น leastline()ใช้สำหรับหาตำแหน่งของจุดภาพที่เป็นแนวเส้นเพื่อนำมาหาค่าความชัน a และจุดตัดแกนy bโดยวิธี least square

Input

R	=	ภาพที่นำมาวิเคราะห์หาแนวเส้นที่ผ่านขั้นตอน edge detection แล้ว
tx	=	ตำแหน่งในแนวแกน x ของจุดปลายของแนวเส้นตรงที่จะวิเคราะห์
ty	=	ตำแหน่งในแนวแกน y ของจุดปลายของแนวเส้นตรงที่จะวิคราะห์
ls	=	ค่าความชัน a จุดตัดแกนy b ของแนวเส้นที่วิเคราะห์

Output

px	=	ตำแหน่งในแนวแกน x ของจุดภาพที่เป็นแนวเส้นตรงที่เก็บมา
py	=	ตำแหน่งในแนวแกน y ของจุดภาพที่เป็นแนวเส้นตรงที่เก็บมา

```
function [px,py] = subfline(R,tx,ty,ls)
  a = ls(1,:);  % ค่าความชัน a
  b = ls(2,:);  % ค่าจุดตัดแกน y b
  [sr1,sr2] = size(R);
  ZR = zeros(size(R));
  posx(1) = 0;
  ch = 1;
  num = 1;

% ทำการปรับจุดเริ่มต้นการอ่าน
% ให้ cc = 1 เมื่อเป็นการอ่านจากซ้ายไปขวา
  if (ty(1) <= ty(2))&(tx(1) < tx(2))      % \>
     cc = 1;
  elseif (ty(1) >= ty(2))&(tx(1) > tx(2))
     cc = 1;
     tempy = ty(1);
     ty(1) = ty(2);                        % change ty1 to ty2
     ty(2) = tempy;                        % change ty2 to ty1

     tempx = tx(1);
     tx(1) = tx(2);                        % change tx1 to tx2
     tx(2) = tempx;                        % change tx2 to tx1

% ให้ cc = 2 เมื่อเป็นการอ่านจากขวาไปซ้าย
  elseif (ty(1) <= ty(2))&(tx(1) > tx(2))    % </
    cc = 2;

  elseif (ty(1) >= ty(2))&(tx(1) < tx(2))
    cc = 2;
    tempy = ty(1);
    ty(1) = ty(2);                         % change ty1 to ty2
```

```
    ty(2) = tempy;                              % change ty2 to ty1

    tempx = tx(1);
    tx(1) = tx(2);                              % change tx1 to tx2
    tx(2) = tempx;                              % change tx2 to tx1
  end;

%-----------------------------------------%
```

% กรณี อ่านจากซ้ายไปขวา

```
  if (cc == 1)
    x = floor(tx(1));
    x = x+1;
    while ch == 1
        ys = floor((a*x)+b);    % หาตำแหน่งของ y เมื่อทราบ x และค่า a กับ b
        if (ys > 6)&(ys <= sr1-6)  % ให้เริ่มทำการหาจุดภาพห่างจากขอบภาพ 6 จุดภาพ
            y = ys - 6;
            i = 1;
            while (i <= 13)    % ทำการอ่านจุดภาพเลื่อนไปในแนวแกน y จนครบ 13 จุดภาพ
            % ถ้าพบว่าเป็น 1 แสดงว่าเป็นจุดภาพที่เป็นแนวเส้นให้เก็บตำแหน่งของจุดภาพนี้ไว้
                if (R(y,x)== 1)
                    ZR(y,x) = 1;
                    posx(num) = x;
                    posy(num) = y;
                    num = num + 1;
                end;
                i = i+1;
                y = y+1;
                if (x >= tx(2)-1)|(y > sr1-6)
                    ch = 2 ;  % อ่านจุดภาพจนถึงสุดแนวเส้นการอ่านแล้ว
                end;
            end;

        end;
        x = x+1;
        if (x >= tx(2)-1)
            ch = 2; % อ่านจุดภาพจนถึงสุดแนวเส้นการอ่านแล้ว

        end;
    end;

%-----------------------------------------------------------%
```

% กรณี อ่านจากขวาไปซ้าย

```
  elseif (cc == 2)
    x = floor(tx(1));
    x = x-1;
    while ch == 1
```

```
        ys = floor((a*x)+b); % หาตำแหน่งของ y เมื่อทราบ x และค่า a กับ b

        if (ys > 6)&(ys <= sr1-6) % ให้เริ่มทำการหาจุดภาพห่างจากขอบภาพ 6 จุดภาพ

            y = ys - 6;
            i = 1;
            while (i <= 13) % ทำการอ่านจุดภาพเลื่อนไปในแนวแกน y จนครบ 13 จุดภาพ
            % ถ้าพบว่าเป็น 1 แสดงว่าเป็นจุดภาพที่เป็นแนวเส้นให้เก็บตำแหน่งของจุดภาพนี้ไว้
                if (R(y,x)== 1)
                    ZR(y,x) = 1;
                    posx(num) = x;
                    posy(num) = y;
                    num = num + 1;
                end;
                i = i+1;
                y = y+1;
                if (x < tx(2)+1)|(y > sr1-6)
                    ch = 2 ; % อ่านจุดภาพจนถึงสุดแนวเส้นการอ่านแล้ว
                end;
            end;

        end;
        x = x-1;
        if (x <= tx(2)+1)
            ch = 2 ; % อ่านจุดภาพจนถึงสุดแนวเส้นการอ่านแล้ว
        end;
    end;

%-----------------------------------------------%

end;
% figure(5)
% imshow(R);
figure(6)
imshow(ZR);
len = 0;
leng = len + length(posx);
if leng > 1
% จากตำแหน่งจุดภาพที่เก็บมาใช้หาความชันa และค่า b โดยวิธี least square
    ss = slp(posx',posy');
% ใช้ฟังก์ชั่น pixslope() ทำการตัดบางจุดในแนวเส้นตรงที่มีแนวโน้มของความชันไม่ตรงกันออก
    [ZZ,px,py] = pixslope(ZR,tx,ty,ss);
    figure(7)
    imshow(ZZ);
else
    px = 0;
    py = 0;
end;
```

ภาคผนวก ค

การใช้งานโปรแกรม

สำหรับการใช้งานโปรแกรมจะต้องทำการรันโปรแกรม Matlab แล้วทำตามขั้นตอนดังต่อไปนี้

1. ในหน้าจอ commad window ให้พิมพ์คำสั่งเพื่อให้ไปสู่ตำแหน่งที่เก็บฟังก์ชั่นไว้

   >> cd ตำแหน่งที่เก็บฟังก์ชั่น

   เช่น >> cd g:\ matlab02

2. เรียกฟังก์ชั่น main() ซึ่งจะรวมเอาทุก ๆ ฟังก์ชั่นที่ใช้ในการหาค่ามุมการหมุนของภาพเอาไว้ แล้วใส่ค่า input เป็นตำแหน่งที่เก็บภาพเอาไว้

   >> [degreeX,degreeY,degreeZ] = main('g:\11\pic1001.jpg');

   จะได้ output

   degreeX คือ ค่ามุมการหันเหของกล้องรอบแกน X

   degreeY คือ ค่ามุมการหันเหของกล้องรอบแกน Y

   degreeZ คือ ค่ามุมการหันเหของกล้องรอบแกน Z

3. หรือถ้าจะทดสอบทีละฟังก์ชั่นก็พิมพ์ชื่อฟังก์ชั่นแล้วใส่ค่าอินพุตและเอาท์พุตโดยอาจดูจากตัวอย่างแต่ละฟังก์ชั่นที่รวมไว้ในฟังก์ชั่น main()