# การพัฒนาอัลกอลิทึมในการหาจุดศูนย์กลาง ของวัตถุทรงกลมจากรูปภาพ

# Development of an Algorithm for Detection of Spherical Object Center from Image

โดย

นายเทพศิริ อลัมภินพงศ์

รหัส 45066022



อาจารย์ที่ปรึกษา

ผศ.ดร.นพพร โชติภกำธร



รายงานนี้เป็นส่วนหนึ่งของโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาเทคโนโลยีสารสนเทศ
ภาคเรียนที่ 2 ปีการศึกษา 2546
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

| | |
|---|---|
| **ชื่อหัวข้อ** | การพัฒนาอัลกอลิทึมในการหาจุดศูนย์กลางของวัตถุทรงกลมจากภาพ |
| **นักศึกษา** | นายเทพศิริ อสัมภินพงศ์ |
| **อาจารย์ที่ปรึกษา** | ผศ.ดร.นพพร โชติกกำธร |
| **ระดับการศึกษา** | วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ |
| **แขนงวิชา** | วิทยาการสารสนเทศ |
| **ปีการศึกษา** | 2546 |

# บทคัดย่อ

ในการหาจุดศูนย์กลางของภาพทรงกลมจากค่าของจุดศูนย์กลางมวลจากภาพที่ถ่าย โดยอาศัยกระบวนการประมวลผลภาพ (Image Processing) อาจเกิดความคลาดเคลื่อนได้ เนื่องจากการเกิดเงาขึ้นบนพื้นผิวของวัตถุทำให้ในขั้นตอนการทำ segmentation นั้นไม่สามารถระบุวัตถุได้ถูกต้อง ส่วนที่เป็นเงาบนพื้นผิววัตถุจะถูกกลืนเข้ากับพื้นหลัง รายงานฉบับนี้นำเสนอผลการพัฒนาอัลกอลิทึมในการหาจุดศูนย์กลางของวัตถุทรงกลมในรูปภาพ เพื่อให้มีความถูกต้องในการระบุตำแหน่งจุดศูนย์กลางของวงกลมภายใต้สภาพแวดล้อมที่เกิดแสงเงาขึ้นในภาพถ่ายได้ โดยอาศัยหลักการหาจุดตัดของเส้นตั้งฉากกับเส้นสัมผัสวงกลมในการหาจุดศูนย์กลาง และอัลกอลิทึมในการจำแนกขอบวัตถุทรงกลมด้านที่รับแสง และด้านที่เกิดเงาออกจากกัน โดยทดลองจากภาพถ่ายของวัตถุจริงจำนวน 19 ภาพ ผลการทดลองที่ได้ ค่าจุดศูนย์กลางที่หามาได้มีค่าคลาดเคลื่อนไปโดยเฉลี่ย 10.50%

| | |
|---|---|
| **Title** | Development of an Algorithm for Detection of Spherical Object Center from Image |
| **Student** | Mr. Tepsiri Asumpinpong |
| **Advisor** | Asst. Prof. Dr. Nopporn Chotikakamthorn |
| **Level of Study** | Master of Science in Information Technology |
| **Major** | Information Science |
| **Academic Year** | 2003 |

# ABSTRACT

Finding a center from spherical picture with the center of mass method, combined with image processing techniques may produce erroneous result due to shadow as appeared on the object surface. Image segmentation algorithm can't detect a whole object from the picture. This report describes development of algorithm for estimating a spherical object center from its image. The algorithm was developed such that the object center estimation accuracy is improved in the presence of shadow in the image. Intersection of lines tangent to the circle circumference is used for the estimation. In addition, an algorithm was developed to differentiate part of the circle circumference on a shadow side. Experiment was performed using 19 images of two(white and red) snooker balls, taken using a digital camera. The circle center estimation was found to differ from a correct position by 10.50 percent on average.

# สารบัญ

หน้า

# สารบัญ(ต่อ)

หน้า

# สารบัญตาราง

หน้า

# สารบัญภาพ

หน้า

# สารบัญภาพ(ต่อ)

หน้า

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาของปัญหา

หากจะกล่าวถึงการหาจุดศูนย์กลางของวงกลมอาจมีคำถามเกิดขึ้นว่าเหตุใดจึงต้องหา หรือว่าหาไปเพื่ออะไร หากมองการนำเอา Image Processing มาช่วยในการทำสื่อโฆษณาหรือภาพยนตร์ การ์ตูน เกมส์ ระบบที่นำมาใช้ในการทำสื่อที่กล่าวมาข้างต้น ก็จะมี Motion Caputure, Virtual Studio, Object Length Estimation, Zoom Estimation, Focal Length Estimation เป็นต้น

จะเห็นได้ว่าในบางครั้งมีความจำเป็นที่จะต้องหาตำแหน่งอ้างอิงของวัตถุ หรือตำแหน่งอ้างอิงของกล้อง เพื่อนำมาใช้ในการหาค่าระยะ หรืออัตราการซูม หรืออาจจะเป็นขนาดของวัตถุในภาพ ซึ่งการหาจุดศูนย์กลางวงกลมก็เป็นอีกวิธีหนึ่งซึ่งสามารถนำมาใช้กับเรื่องพวกนี้ได้

การหาจุดศูนย์กลางของวงกลมที่จะทำการพัฒนาอัลกอลิทึมนี้ เป็นการหาจุดศูนย์กลางของรูปวงกลมโดยนำเอาการทำ Image Processing เข้ามาเพื่อใช้ในการหาค่าที่แน่นอนของวัตถุทรงกลมในภาพซึ่งอาจมีการผิดเพี้ยนไปจากสภาวะรอบข้าง เช่น แสง เงา การสะท้อนของวัตถุ

## 1.2 วัตถุประสงค์ในการพัฒนาอัลกอลิทึม

1.2.1 เพื่อทำการหาตำแหน่งจุดศูนย์กลางที่แน่นอนของวัตถุทรงกลมที่อยู่ในภาพ

1.2.2 ลดความผิดพลาดในการหาตำแหน่งจุดศูนย์กลางของวัตถุในภาพเนื่องจากผลกระทบของสภาวะแวดล้อม เช่น แสง เงา และ การสะท้อนของวัตถุ

1.2.3 ค้นหาแนวทางในการหลีกเลี่ยงความคลาดเคลื่อนจากเงาของวัตถุเมื่อนำไปทำ Image Processing

1.2.4 ค้นหาแนวทางในการหลีกเลี่ยงความคลาดเคลื่อนจากแสงสะท้อนของวัตถุเมื่อนำไปทำ Image Processing

## 1.3 ขอบเขตของการพัฒนาอัลกอลิทึม

1.3.1 สามารถหาจุดศูนย์กลางของวงกลมจากภาพในสภาวะที่วัตถุในภาพถูกกลืนโดยเงาบางส่วนซึ่งจะทำให้รายละเอียดของวงกลมผิดเพี้ยนไป

1.3.2 สามารถหาจุดศูนย์กลางของวงกลมจากภาพในสภาวะที่วัตถุมีแสงสะท้อนจากแหล่งกำเนิดแสงได้

1.3.3 สามารถหาจุดศูนย์กลางของวงกลมที่มีจำนวน 1-2 ลูก ในรูปได้

1.3.4 สามารถหาจุดศูนย์กลางของวงกลมในกรณีที่เกิดการซ้อนทับกันของวัตถุ 2 ชิ้นได้

1.3.5 สามารถบอกได้ถึงความผิดพลาดของอัลกอลิทึมในการหาตำแหน่งจุดศูนย์กลางวงกลม

1.3.6 ค่าที่ได้จากการหาตำแหน่งจุดศูนย์กลางวงกลม อาจคลาดเคลื่อนด้วยอัตราความผิดพลาดที่ยอมรับได้ในระดับหนึ่ง

**1.4 ผลที่คาดว่าจะได้รับ**

ผลที่คาดว่าจะได้รับจากการพัฒนาอัลกอลิทึมคือ สามารถแก้ปัญหาการหาจุดศูนย์กลางของ รูปวงกลมที่ได้จากภาพซึ่งถูกรบกวนโดย แสง เงา และการสะท้อนของพื้นผิววัตถุ อีกทั้งยังสามารถ แยกแยะในกรณีที่มีวัตถุทรงกลมมากกว่า 1 ชิ้นในรูป และรวมไปถึงการหาตำแหน่งจุดศูนย์กลางของวัตถุที่ซ้อนทับกันนั้นได้

อัลกอลิทึมที่จะทำการพัฒนานั้นอาจไม่สามารถจับจุดศูนย์กลางที่ตรงตามตำแหน่งที่ถูกต้องได้ 100% แต่จะมีเปอร์เซ็นต์ความผิดพลาดที่อยู่ในระดับที่ยอมรับได้

# บทที่ 2

## การทำ Image Processing

### 2.1 อุปกรณ์ Motion Capture

Motion Capture คือชื่อของอุปกรณ์ที่ใช้จับการเคลื่อนไหวและส่งรูปแบบการเคลื่อนไหวนั้นออกมาเป็นข้อมูลเพื่อที่จะนำมาใช้ประมวลผลหรือแสดงผลต่อไป

การใช้งานตัว Motion Capture ส่วนใหญ่ที่จะพบเห็นกันคือในวงการบันเทิง การสร้างการเคลื่อนไหวของตัวการ์ตูน 3 มิติให้สมจริง ไม่ว่าจะเป็นหนังการ์ตูน หรือเกมในปัจจุบัน แทนการเขียนรูปแบบการเคลื่อนไหวของตัวการ์ตูนแต่ละตัวทีละเฟลม และอีกวงการหนึ่งที่จะพบเห็นการใช้ Motion Capture อย่างมากคือในวงการแพทย์ ใช้ตรวจหาการเคลื่อนไหวของร่างการมนุษย์ และการใช้งานอีกอย่างหนึ่งที่ไม่ค่อยเห็นกันได้คือการใช้ motion capture ตรวจการเคลื่อนไหวของมนุษย์ในสภาวะจำลองสภาวะหนึ่งเพื่อนำการเคลื่อนไหวนั้นมาประยุกต์ใช้กับโลกเสมือนจริง( Virtual reality world)

อุปกรณ์ Motion Capture แบ่งออกเป็น 4 ชนิดตามลักษณะการติดตามการเคลื่อนไหว ดังนี้

#### 2.1.1 Prosthetic-based

เป็นระบบที่ออกมาแรกสุดสำหรับจับการเคลื่อนไหวส่วนต่างๆของร่างกายมนุษย์ โดนระบบนี่จะมีส่วนของตัวตรวจจับ (tracker)ติดอยู่ตามส่วนต่างๆของข้อต่อต่างๆของร่างที่จะจับการเคลื่อนไหวและตัวตรวจจับตามส่วนต่างๆจะเชื่อมต่อกันโดย linear encoders

แต่ระบบนี้ไม่ค่อยเป็นที่นิยมเพราะจะเคลื่อนไหวไม่สะดวกเนื่องจากจะติด ตัว linear encoders ที่เชื่อมต่อตัวตรวจจับเข้าด้วยกัน

#### 2.1.2 Acoustic-based

จะใช้รูปแบบของคลื่นเสียงเพื่อช่วยในการตรวจจับตำแหน่งของส่วนต่างๆของร่างกาย โดยจะมีตัวส่งสัญญาณคลื่นเสียงติดตามส่วนต่างๆของร่ายกายและมีตัวรับสัญญาณ 3 ตัววางอยู่ในตำแหน่งที่แตกต่างกัน โดยที่ตัวส่งสัญญาณจะทำการค่อยๆทำการส่งคลื่นออกมาครั้งละตัวเป็นจังหวะคงที่เพื่อนำเอาค่าที่ได้จากการคำนวณว่าเสียงวิ่งจากตัวส่งถึงตัวรับทั้ง 3 ตัวใช้เวลา

เท่าใด ก็จะได้ออกมาเป็นตำแหน่งในรูป 3 มิติ ระบบการจับการเคลื่อนไหวชนิดนี้ส่วนใหญ่จะใช้ในการจับเป็นภาพนิ่ง (Snap shot) แทนที่จะทำการจับภาพเป็นลักษณะการเคลื่อนไหว

**2.1.3 Electromagnetic-based**

จะใช้คลื่นแม่เหล็กไฟฟ้าเป็นตัวช่วยในการตรวจจับตำแหน่งต่างๆของร่างกายโดยที่จะมีส่วนของอุปกรณ์รับสัญญาณแม่เหล็กไฟฟ้าติดอยู่ตามส่วนต่างๆของร่างกาย และรอบๆจะมีตัวสร้างสนามแม่เหล็ก อุปกรณ์รับสัญญาณแต่ละตัวจะมีสายข้อมูลส่งเข้าไปประมวลผลเพื่อให้ได้ตำแหน่งบนแกน 3 มิติของอุปกรณ์แต่ละตัวที่ติดยังส่วนต่างๆของร่างกาย

**2.1.4 Optical-based**

เป็นระบบที่ได้รับความนิยมมากที่สุดในช่วงหนึ่งถึงสองปีที่ผ่านมา ระบบนี้จะใช้กล้องดิจิตอล 3 ตัววางอยู่คนละตำแหน่งกัน และมีอุปกรณ์สะท้อนแสงติดไว้ตามส่วนต่างๆของร่างกายและข้อมูลที่ได้มาจะเป็นรูปของกล้องแต่ละตัว และนำภาพที่ถ่ายติดอุปกรณ์สะท้อนแสงนำมาวิเคราะห์หาตำแหน่งในรูป 3 มิติต่อไป

แต่ที่เราจะให้ความสนใจคือ Optical Motion Capture ซึ่งการที่จะหาตำแหน่งของวัตถุอ้างอิงหรือ Tracker จากรูปที่ถ่ายออกมาจากกล้องดิจิตอลนั้นจำเป็นจะต้องนำเอาไปผ่านการประมวลผลภาพหรือ Image Processing ต่างๆเพื่อให้ได้ค่าตำแหน่งของวัตถุอ้างอิงออกมา

**2.2 Image Segmentation**

คือกระบวนการในการวิเคราะห์รูปภาพเพื่อทำการแยกแยะหาวัตถุที่มีอยู่ในภาพ โดยการทำ highlight ให้แต่ละวัตถุมีสีที่ต่างกันออกไป การแยกหาวัตถุในภาพนั้น กลุ่มของวัตถุที่ได้มาจะต้อง ไม่สูญเสียรายละเอียดที่สำคัญของวัตถุเมื่อเทียบกับรูปจริงๆ เช่น ถ้ารับรูปวงกลมเข้ามา หลังจากการทำ Segmentation แล้ววัตถุที่ได้จะต้องเป็นวงกลม ไม่ใช่สี่เหลี่ยมหรือสามเหลี่ยม

การทำ Image Segmentation นั้นสามารถทำได้โดยมีขั้นตอนหลักๆด้วยกัน 3 ขั้นตอน คือ

**2.2.1 ขจัด noise ต่างๆที่มากับรูป**

เนื่องจากรูปที่ได้มาบางครั้งอาจมี noise ติดมาด้วยเพราะความไม่สมบูรณ์ของการจับภาพ noise ที่มากับรูปแบบ binary นั้นจะมาใน 2 รูปแบบคือ hole ซึ่งเป็นจุดสีดำอยู่ภายใน object และ snow ซึ่งเป็นจุดสีขาวๆอยู่บน background ตัวอย่างของรูปที่มี noise จะเห็นได้จากรูปที่ 2 ถ้านำรูปที่ยังไม่ได้ทำการขจัด noise ไปทำการ highlight จะทำให้ได้วัตถุออกมาไม่ตรงกับที่ควรจะได้ คือส่วนที่เป็น noise จะทำให้ลักษณะของวัตถุที่ได้มาผิดแปลกไป ดังนั้นการขจัด noise ก่อนจะไปทำการแยะแยะหาวัตถุนั้นจึงจำเป็นอย่างยิ่ง แต่ก่อนจะเข้าขั้นตอนในการขจัด noise นั้นมีกระบวนการที่ควรจะรู้คือ

1. Mask คือค่าคงที่ชุดหนึ่งที่ถูกกำหนดขึ้นมาใส่ลงใน matrix ขนาดจตุรัส ซึ่งส่วนใหญ่จะมีขนาดเป็นเลขคี่ตั้งแต่ 3 x 3 ขึ้นไป

mask จะถูกสร้างขึ้นเพื่อจุดประสงค์ในการนำทำการเทียบกับ pixel ในรูป ซึ่งจะทำให้รูปแบบของรูปภาพเปลี่ยนแปลงไป โดยใช้ operation and, or, xor เป็นตัวเปรียบเทียบสำหรับการใส่ค่าใหม่ลงใน pixel นั้นๆ ตัวอย่างเช่นในรูปแบบ binary จะมีค่าเพียง 2 ค่าคือ 0 คือสีดำ และ 1 คือสีขาว และ กำหนด matrix ขนาด 3 x 3 ที่มีค่าตัวแปรดังนี้

$$\begin{matrix} X_3 & X_2 & X_1 \\ X_4 & X & X_0 \\ X_5 & X_6 & X_7 \end{matrix}$$

กำหนดค่าบนรูปภาพเป็น I(r,c) โดยที่ r แทนด้วยค่า pixel ในแถวหรือตำแหน่งในแนวแกน Y ในรูป และ c แทนด้วยค่า pixel ในหลัก หรือตำแหน่งในแนวแกน X ในรูปภาพการคำนวณค่าใหม่ของ pixel นั้นจะได้ว่า

O(r,c) = 1 If X = I(r,c) and

$X_0$ = I(r,c+1) and

$X_1$ = I(r-1,c+1) and

$X_2$ = I(r-1,c) and

$X_3$ = I(r-1,c-1) and

$X_4$ = I(r,c-1) and

$X_5$ = I(r+1,c-1) and

$X_6$ = I(r+1,c) and

$X_7$ = I(r+1,c+1)

Else O(r,c) = 0

จากตัวอย่างข้างต้นจะเป็นการดูว่าจุด I(r,c) และจุดรอบๆนั้นมีค่าเหมือนกับค่าใน matrix ที่กำหนดมาหรือไม่ถ้าไม่ก็ให้ค่าใหม่เป็น 0 ถ้าใช่ก็ให้ค่าใหม่เป็น 1 การ mask จะต้องทำการเทียบให้หมดทั้งรูปภาพ ค่าข้างต้นที่ทำการกำหนดให้รูปภาพใหม่ และค่า operation and นั้นสามารถทำการเปลี่ยนแปลงได้ตามวัตถุประสงค์ที่ต้องการ

2. Dilation คือวิธีที่ใช้ในการเปลี่ยนแปลงรูปภาพซึ่งการนำ Dilation mask ไปเทียบกับรูปภาพ จะทำให้รูปภาพใหม่ที่ได้หลังจากการเทียบมีขนาดขยายใหญ่ขึ้นโดยที่ Dilation mask มีค่าดังนี้

0 0 0

0 0 0

0 0 0

เมื่อนำ Dilation mask ไปทำการเทียบแล้วค่าใหม่ในรูปที่ได้จะมีค่า

2.1. เป็น 0 ถ้าค่า pixel และจุดรอบๆในรูปเก่าที่ทำการเทียบด้วย mask นั้นมีค่าเท่ากันกับค่า mask

2.2. เป็น 1 ถ้าไม่เป็นไปตามข้อ 2.1

3. Erosion คืออีกวิธีที่ใช้ในการเปลี่ยนแปลงรูปภาพซึ่งจะนำ Erosion mask ไปเทียบกับรูปภาพ โดยจะทำให้รูปภาพใหม่ที่ได้หลังจากการเทียบมีขนาดเล็กลงโดยที่ Erosion mask มีค่าดังนี้

1 1 1

1 1 1

1 1 1

เมื่อนำ Erosion mask ไปทำการเทียบแล้วค่าใหม่ในรูปที่ได้จะมีค่า

3.1. เป็น 1 ถ้าค่า pixel และจุดรอบๆในรูปเก่าที่ทำการเทียบด้วย mask นั้นมีค่าเท่ากันกับค่า mask

3.2. เป็น 0 ถ้าไม่เป็นไปตามข้อ 3.1

4. Opening คือวิธีการเปลี่ยนแปลงรูปภาพที่นำเอาการทำ Dilation และ Erosion มารวมกันประยุกต์ใช้ หรือจะกล่าวได้อีกนัยหนึ่งว่า Dilation และ Erosion เป็นการเปลี่ยนแปลง

รูปขั้นพื้นฐานแต่การ Opening และ Closing เป็นอีกขั้นหนึ่งของการเปลี่ยนแปลงรูปซึ่งการ Opening และ Closing นั้นจะถูกนำมาใช้เพื่อทำการขจัด noise ที่เกิดขึ้นกับรูป

การทำ Opening นั้นรูปภาพที่ได้หลังจากการทำจะสามารถลบ noise ที่เป็น snow ที่มีอยู่ภายในรูปภาพได้โดยที่การทำ Opening นั้นจะทำได้โดยการทำ Erosion กับรูป แล้วนำรูปที่ได้ไปทำการ Dilation จะทำให้จุดขาวภายในภาพที่มีขนาดเพียง 1 จุดหายไป

ถ้าหากต้องการขจัด snow ที่มีค่ามากกว่า 1 pixel นั้นจะต้องนำรูปมาทำการ Erosion มากกว่า 1 ครั้งแล้วทำการ Dilation เป็นจำนวนครั้งเท่ากันกับการทำ Erosion แต่ถ้ายิ่งทำการ Erosion มากกว่า 1 ครั้งไปเท่าใดรายละเอียดของรูปก็จะยิ่งลดลงไปตามมากเท่านั้น ดังนั้นถ้าจะทำการขจัด noise ที่มีขนาดมากกว่า 1 pixel นั้นจึงต้องทำการคิดดูให้ดีๆว่าจะทำให้รายละเอียดของวัตถุในรูปลดลงไปจนทำให้สูญเสียลักษณะเฉพาะของวัตถุนั้นๆหรือไม่

5. Closing จะทำได้โดยนำเอารูปภาพไปทำการ Dilation แล้วจึงนำรูปที่ได้ไปทำการ Dilation หลังจากการทำ Closing แล้วนั้นจุดเล็กๆสีดำที่มีขนาด 1 pixel ภายในวัตถุจะถูกขจัดออกไป

การจะทำการขจัด hole ที่มีขนาดมากกว่า 1 pixel จะทำได้โดยการทำคล้ายๆกันกับที่ทำให้ Opening คือ ทำการ Dilation มากกว่า 1 ครั้งแล้วทำการ Erosion เท่ากับจำนวนครั้งที่ทำ Dilation แต่การทำเช่นนี้ต้องระมัดระวังมากกว่าในส่วนของการทำ Opening เพราะว่าวัตถุในรูปบางชนิดมี hole ซึ่งเป็นรายละเอียดของรูปไม่ใช่ noise เช่นรูปตัวอักษรไทยต่างๆที่มีหัว จะมีส่วนที่เป็น hole อยู่ภายในหากทำการ Closing มากเกินไปจะทำให้ hole ในส่วนหัวของตัวอักษรหายไปทำให้เสียลักษณะเฉพาะและต่อไปจะเป็นอุปสรรคต่อการจำแนกชนิดของวัตถุ

จากกระบวนการต่างๆที่บรรยายมาข้างต้น จะเห็นได้ว่าส่วนที่ใช้ในการขจัด noise จริงๆคือส่วนของ Opening และ Closing เท่านั้นส่วน Mask, Dilation และ Erosion นั้นเป็นพื้นฐานที่จะนำมาใช้ในการขจัด noise เท่านั้น การขจัด noise นั้นทำได้โดนการทำ Opening แล้วนำรูปที่ได้จากการ Opening มาทำ Closing ต่อก็จะสามารถขจัด noise ออกไปได้ส่วนหนึ่ง หรือจะทำ Closing แล้วค่อยทำ Opening ก็ได้แล้วแต่ผู้ที่จะนำไปประยุกต์ใช้

**2.2.2 การแยกแยะวัตถุออกจากรูป**

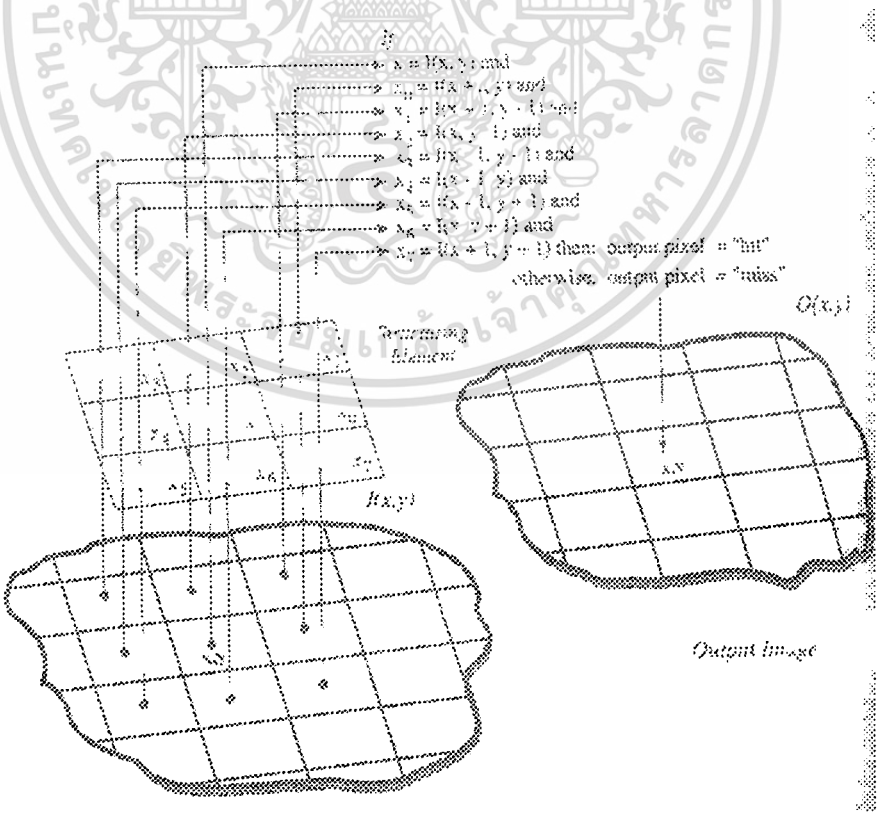
ทำได้โดยการดูว่า pixel ใดที่มีค่าเป็น 1 และเชื่อมต่อกันก็ให้ถือว่าเป็นวัตถุชนิดเดียวกันโดยจะทำการ highlight pixel ที่เชื่อมต่อกันให้เป็นสีเดียวกัน จะถือว่าวัตถุแต่ละชนิดมีสีที่

highlight แตกต่างกันทำให้สามารถแบ่งแยกรายละเอียดที่มีอยู่ภายในรูปออกเป็นวัตถุชนิดต่างๆกันได้

**2.2.3 การลดขนาดของวัตถุ**

การลดขนาดของวัตถุที่ได้ที่ได้ลงให้มีขนาดความกว้างเป็น 1 pixel ที่ทำเช่นนี้ก็เพื่อให้ง่ายต่อการนำเอาวัตถุที่ทำการลดขนาดลงไปเปรียบเทียบกับรูปแบบที่มีเก็บอยู่ในฐานข้อมูลเพื่อวิเคราะห์ออกมาให้ได้ว่าวัตถุนั้นจัดอยู่ในคลาสอะไร ส่วนใหญ่จะเห็นได้จากการทำระบบรู้จำตัวอักษร เนื่องจากตัวอักษรที่ได้มาจากการ scan เข้ามานั้นมีความหนาค่อนข้างมากดังนั้นจึงต้องลดขนาดลงเพื่อให้สามารถเทียบได้ใกล้เคียงกับขนาดของอักษรที่ได้ทำการบันทึกไว้

แต่ในกรณีที่จะนำมาประยุกต์ใช้กับตัว Motion Capture นั้น ระบบต้องการจับเอารูปของตัว Tracker ที่ติดอยู่กับร่างกายของผู้ที่เคลื่อนไหวดังนั้นการลดขนาดภาพให้เหลือเพียง 1 pixel นั้นจึงไม่ควรจะทำเพราะว่าจะทำให้จุดศูนย์กลางของตัว tracker ที่ได้มานั้นคลาดเคลื่อนออกไปได้ใน และอีกทั้งขนาดของ tracker ซึ่งเป็นทรงกลม ที่ทำการจับภาพมานั้นมีขนาดไม่เท่ากันถ้าทำการลดจนเหลือเพียง 1 pixel ก็จะเหลือเพียงจุดที่ tracker มีขนาดใหญ่สุดเท่านั้น การลดขนาดจึงไม่มีความจำเป็นต่อระบบ Motion Capture เลย



รูปที่ 2.1 แสดงการเปรียบเทียบ mask กับตำแหน่งในรูป

รูปที่ 2.2 แสดงรูปต้นแบบที่ยังไม่ได้ทำการขจัด noise

รูปที่ 2.3 แสดงรูปภาพหลังจากการเทียบด้วย Dilation mask

รูปที่ 2.4 แสดงรูปภาพหลังจากการเทียบด้วย Erosion mask

รูปที่ 2.5 แสดงรูปภาพหลังจากทำการ Opening

รูปที่ 2.6 แสดงรูปภาพหลังการทำ Closing

รูปที่ 2.7 แสดงรูปที่ทำการขจัด noise เรียบร้อยแล้ว

**2.3 Feature Extraction**

คือกระบวนการที่นำวัตถุที่ได้จากการ Segmentation มาทำการหาลักษณะเฉพาะของวัตถุแต่ละชิ้นเพื่อนำเอาลักษณะเฉพาะที่ได้เหล่านั้นไปเทียบกับ Knowledge base เพื่อให้รู้ว่าวัตถุนั้นๆจัดอยู่ในประเภทอะไร โดยที่ Knowledge base คือส่วนที่บันทึกความจริงเกี่ยวกับชนิดของวัตถุและลักษณะเฉพาะต่างๆ

การทำ Feature Extraction นั้นไม่ได้ทำได้แค่เพียงแยกแยะชนิดของวัตถุเท่านั้น แต่ยังสามารถนำลักษณะบางอย่างของวัตถุไปใช้กับกระบวนการอื่นๆได้อีกด้วย

ลักษณะส่วนใหญ่ที่ใช้ในการจำแนกประเภทของวัตถุมีดังต่อไปนี้

**2.3.1. ความสว่างและสี**

เป็นลักษณะทั่วๆไปที่รูปภาพแต่ละรูปพึงจะมี แต่ลักษณะนี้ไม่เหมาะกับการใช้แยกแยะวัตถุออกจากรูปแบบ binary เนื่องจากมีเพียงแค่ 2 ค่าคือ 1 กับ 0 ไม่มีแสงสว่างที่เห็นได้อย่างชัด ดังนั้นลักษณะนี้จึงเหมาะสำหรับรูปภาพแบบ gray scale ขึ้นไป ซึ่งจะมีความแตกต่างของสีและแสงที่พอจะแยกแยะได้

ในกรณีที่รูปเป็นแบบ gray scale นั้น histogram จะเป็นสิ่งที่เหมาะสมกับการใช้หาลักษณะที่สุดเพราะ histogram จะแสดงให้เห็นถึงค่าความสว่างเฉพาะของแต่ละวัตถุ

ในกรณีที่รูปเป็นแบบ RGB(red ,green ,blue) นั้น การหาลักษณะสีและความสว่างจะต้องทำการเปลี่ยน RGB ให้มาเป็น HSB( hue, saturation, intensity) โดยที่ H คือค่าของสี S คือค่าความเข้มของสี และ I คือค่าความสว่าง โดนค่าที่จะนำมาใช้เป็นตัวเปรียบเทียบคือ Hue และ Intensity

**2.3.2. พื้นที่**

จะเป็นตัวที่สามารถบ่งบอกถึงขนาดของวัตถุโดยจะนับจำนวน pixel ทั้งหมดของวัตถุนั้นๆ

**2.3.3. พื้นที่หลุม**

คือจำนวน pixel ทั้งหมดภายในวัตถุที่มีสีเป็นสีเดียวกับพื้นหลัง

**2.3.4. จุดศูนย์กลางของวัตถุ**

สามาถหาได้จาก

$$X_{Center} = \frac{\text{ผลรวมของค่า X ทั้งหมด}}{\text{พื้นที่รูป}} \quad (1)$$

$$Y_{Center} = \frac{\text{ผลรวมของค่า Y ทั้งหมด}}{\text{พื้นที่รูป}} \quad (2)$$

**2.3.5. เส้นรอบรูป**

มีลักษณะการเก็บอยู่ 2 แบบคือ

1. เก็บเป็น pixel คือเก็บค่า (x,y) ทั้งหมดที่เป็นเส้นรอบรูปของวัตถุ

2. เก็บเป็น chain code คือจะเก็บค่า (x,y) ของจุดเริ่มต้นบนเส้นรอบรูปเพียงจุดแล้วแล้วค่าต่อเป็นจะเป็น 0-7 ซึ่งเป็นตำแหน่งต่อไปของจุดบนเส้นรอบรูปโดยที่แต่ละค่าจะมีทิศทางดังนี้

| | | |
|---|---|---|
| 3 | 2 | 1 |
| 4 | | 0 |
| 5 | 6 | 7 |

โดยที่ค่า Chain code จะเก็บไปเรื่อยๆจะกระทั่งวนกลับมาจนถึงจุดเริ่มต้นอีกครั้ง

**2.3.6. ความยาวเส้นรอบรูป**

สาสามรถคิดได้โดยไล่ chain code โดยที่เส้นที่เป็น 0, 2, 4, 6 ให้มีค่าเป็น 1 ส่วน 1, 3, 5, 7 นั้นมีค่าเป็น $\sqrt{2}$ หรือ 1.414 แล้วทำการรวมค่าทั้งหมดที่ได้ก็จะได้ค่าความยาวเส้นรอบรูปออกมา

**2.3.7. Histogram**

คือกราฟที่ใช้ระบุกลุ่มของตัวเลขโดยที่ตัวเลขเหล่านั้นคือปริมาณ 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

การจะทำ Feature Extraction นั้นในบางครั้งใช้ลักษณะของวัตถุเพียงข้อเดียวก็สามารถหาชนิดของวัตถุนั้นๆได้ แต่ในบางครั้งก็ต้องใช้มากกว่า 1 ข้อเพื่อที่จะให้ได้มาซึ่งชนิดของวัตถุนั้น ดังนั้นการเลือกลักษณะที่เหมาะสมกับการหาชนิดของวัตถุนั้นจึงเป็นสิ่งสำคัญอย่างยิ่ง

**2.4 แนวทางการหาจุดศูนย์กลางของวงกลม**

จากการค้นหาจากแหล่งอ้างอิงทำให้พบว่าแนวทางการหาจุดศูนย์กลางของวงกลมนั้นมีได้หลายวิธีแต่ละวิธีนั้นจำเป็นจะต้องมีตำแหน่งของขอบของวงกลมนั้นๆจึงจะหามาได้ ซึ่งการที่จะหาขอบของวงกลมนั้นอาจหาได้โดยวิธีการทำ Edge detection

แนวทางการหาจุดศูนย์กลางวงกลมนี้จะทำการนำเอารูปภาพขาวเทา (gray scale image) มาทำการหาจุดศูนย์กลางของวงกลมซึ่งสามารถหาได้โดย

**2.4.1 Maximum Likelihood Estimation**

กำหนดให้ $r_i(a,b) = \sqrt{(x_i-a)^2+(y_i-b)^2}$ โดยที่ (a,b) คือจุดศูนย์กลางของวงกลม $r_i$ คือรัศมีแล้ว จุดศูนย์กลางและรัศมีของวงกลมที่มีเส้นขอบอยู่ที่ตำแหน่ง $(x_i,y_i)$ คือ

ค่าของ (a,b,r) ที่ทำให้สมการ

$$\sum_{i=1}^{N}[r_i(a,b)-r]^2$$

มีค่าน้อยที่สุด

ข้อเสียของวิธีนี้ก็คือยากต่อการวิเคราะห์และคำนวณหาค่าจากสมการ และในบางครั้งค่าที่มีค่าน้อยที่สุดก็ไม่สามารถหาค่าออกมาได้อีกด้วย

**2.4.2 The Delogne-Kåsa Estimator**

เป็นวิธีที่ ปรับปรุงมาจากวิธี Maximum Likelihood Estimation โดยที่หาค่าจุดศูนย์กลางและรัศมีจากสมการ

$$(\hat{a}_{DK}, \hat{b}_{DK}, \hat{r}_{DK}) = \arg\min_{(a,b,r)} \sum_{i=1}^{N}[r_i^2(a,b)-r^2]^2 \qquad (3)$$

ซึ่งจากสมการข้างต้นสามารถใช้การหา Least square เข้ามาช่วยในการหาค่า (a,b) ได้จาก

$$\mathbf{Z} = (\hat{a}_{\mathrm{DK}}, \hat{b}_{\mathrm{DK}})^T = \tfrac{1}{2}(\mathbf{S}+\mathbf{T})^{\#}(\mathbf{u}+\mathbf{v}).$$

$$\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2) = \mathbf{P}\mathbf{S}', \ \mathbf{T} = \mathbf{P}\mathbf{T}',$$

$$\mathbf{u} = \mathbf{P}\mathbf{u}' \text{ and } \mathbf{v} = \mathbf{P}\mathbf{v}'$$

$$\mathbf{S}' = \begin{pmatrix} a + r\cos\theta_1 & b + r\sin\theta_1 \\ \vdots & \vdots \\ a + r\cos\theta_N & b + r\sin\theta_N \end{pmatrix}, \mathbf{T}' = \begin{pmatrix} \xi_1 & \eta_1 \\ \vdots & \vdots \\ \xi_N & \eta_N \end{pmatrix} \qquad (4)$$

$$u_i' = (a + r\cos\theta_i)^2 + (b + r\sin\theta_i)^2,$$

$$v_i' = 2(a + r\cos\theta_i)\xi_i + 2(b + r\sin\theta_i)\eta_i + \xi_i^2 + \eta_i^2$$

ซึ่งสัญลักษณ์ # นั้นหมายถึง psudo-inverse ของเมทริกซ์นั้นๆ ซึ่งสามารถหาได้จาก $A^{\#} = (A^TA)^{-1}A^T$ เมื่อ $A^TA$ เป็น non-singular เมทริกซ์ และ P คือเมทริกซ์ขนาด N x N ซึ่งมาจาก

$P = I - (11^T/n)$

ซึ่ง I คือเมทริกซ์เอกลักษณ์ขนาด N x N

และ 1 คือเมทริกซ์หลักยาว N ซึ่งมีสมาชิกทั้งหมดเป็น 1

**2.4.3. การหาจุดศูนย์กลางของวงกลมจากการตัดกันของเส้นตั้งฉากกับเส้นสัมผัสวงกลม**

การหาจุดศูนย์กลางด้วยวิธีนี้เป็นอะไรที่ค่อนข้างที่จะไม่ซับซ้อนและง่ายต่อการหา อีกทั้งไม่จำเป็นจะต้องมีจุดบนเส้นขอบมากมายเพื่อนำมาหา เพียงแต่มี 3 จุดเป็นอย่างน้อยก็จะสามารถนำมาใช้หาจุดศูนย์กลางของวงกลมได้แล้ว การหาจุดศูนย์กลางของวงกลมวิธีนี้ทำได้โดย

1. หาจุด 2 จุดที่อยู่บนขอบของวงกลม $(x_1,y_1)$ และ $(x_2,y_2)$

2. หาสมการเส้นตรงที่ตั้งฉากกับจุดกึ่งกลางของเส้นที่ลากผ่านจุด $(x_1,y_1)$ และ $(x_2,y_2)$ ซึ่งสมการเส้นตรงที่ว่าจะผ่านจุดศูนย์กลางของวงกลมพอดี จาก

$$\text{ความชัน} = \frac{(y_2-y_1)}{(x_2-x_1)} \qquad (5)$$

$$\text{จุดกึ่งกลาง } (x_c,y_c) = \left(\frac{(x_1+x_2)}{2}, \frac{(y_1+y_2)}{2}\right) \qquad (6)$$

$$\text{ความชันของเส้นตั้งฉาก} = -\frac{(x_2-x_1)}{(y_2-y_1)} \qquad (7)$$

$$\text{สมการที่จะได้คือ } y = \text{ความชัน} \cdot x \qquad (8)$$

$$y - y_c = -\frac{(x_2-x_1)}{(y_2-y_1)}\cdot(x-x_c) \qquad (9)$$

$$x(x_2-x_1)+y(y_2-y_1) = y_c(y_2-y_1)+x_c(x_2-x_1) \qquad (10)$$

3. หาสมการเส้นตรงเช่นนี้สองเส้น แล้วนำมาหาจุดตัดของสมการสองสมการอีกทีก็จะได้จุดศูนย์กลางวงกลมออกมา

แต่การใช้จุดตัดของสมการเพียงสองสมการนั้นอาจได้ค่าจุดศูนย์กลางที่ไม่แม่นยำนัก เนื่องจากการความคลาดเคลื่อนในการหาจุดบนขอบของวงกลมดังนั้นเพื่อให้ได้ค่าที่ใกล้เคียงกับจุดศูนย์กลางของวงกลมนั้นๆ จึงได้มีการนำค่าของสมการเส้นตั้งฉากหลายๆเส้นมาเพื่อใช้ในการประมาณหาค่าจุดตัด

เพื่อให้ง่ายต่อการคำนวณ เราเลือกเอาการหาเมทริกซ์ผกผันมาใช้เป็นวิธีการในการแก้ปัญหาสมการเชิงเส้นในครั้งนี้ ซึ่งมีวิธีการทำโดย

หาสมการเส้นตั้งฉากกับจุดสัมผัสวงกลมให้อยู่ในรูปของ

$$ax+by = c \qquad (11)$$

จะได้เป็นว่า

$$a_1x+b_1y = c_1 \qquad (1)$$

$$a_2x+b_2y = c_2 \qquad (2)$$

$$a_3x+b_3y = c_3 \qquad (3)$$

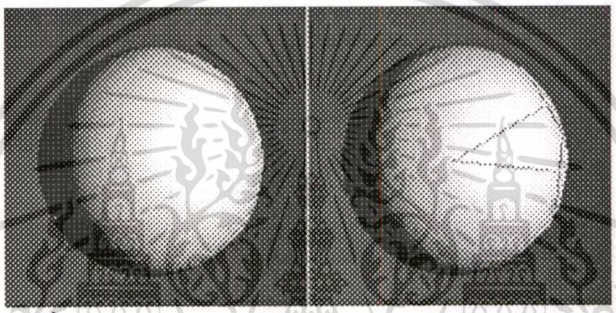
$$a_ix+b_iy = c_i \qquad (i)$$

$$\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \\ a_i & b_i \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \cdot \\ \cdot \\ \cdot \\ c_i \end{pmatrix} \qquad (12)$$

$$A = \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ \vdots & \vdots \\ a_i & b_i \end{pmatrix}, B = \begin{pmatrix} x \\ y \end{pmatrix}, C = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_i \end{pmatrix} \quad (13)$$

จะได้เป็นว่า $AB = C$

$$B = A^{-1}C \quad (14)$$



รูปที่ 2.8 แสดงตัวอย่างแนวทางการหาจุดศูนย์กลางวงกลม

แต่ว่าเมทริกซ์ inverse นั้นสามารถหาได้เพียงแต่สำหรับเมทริกซ์ที่เป็นเมทริกซ์จตุรัสเท่านั้น ดังนั้นจึงต้องหาเป็น psudo-inverse แทน ซึ่ง psudo-inverse เมทริกซ์นั้นคือเมทริกซ์ที่คล้ายคลึงกับเมทริกซ์ inverse เพียงแต่ว่าสามารถใช้กับเมทริกซ์ที่ไม่ใช่เมทริกซ์จตุรัสได้

Psudo-inverse หาได้โดย

$$A^{\#} = (A^T A)^{-1} A^T \quad (15)$$

แล้วนำมาคูณกับเมทริกซ์ C ก็จะสามารถประมาณค่าหาจุดศูนย์กลางของวงกลมได้ดังสมการ

$$\begin{pmatrix} x \\ y \end{pmatrix} = A^{\#} C \quad (16)$$

ค่าที่ได้จากการหานั้นขึ้นอยู่กับว่าสมการที่ใช้แต่ละสมการนั้นผ่านจุดที่อยู่บนขอบของวงกลมที่เราหามาหรือไม่ ถ้าจุดที่ได้จากการหาจุดบนขอบผิดพลาดไปมากๆ ฟังก์ชันที่ได้จากจุดสองจุดก็จะผิดพลาดไปด้วย ส่งผลกระทบให้ผลลัพธ์ที่ได้จากการหาโดยวิธีเมทริกซ์ผกผันนั้นผิดไปด้วยเช่นกัน ดังนั้นเราจึงต้องทำการศึกษาการหาจุดบนขอบวงกลมด้วย เพื่อให้ได้ค่าที่ใกล้เคียงกับจุดที่อยู่บนของจริงๆที่สุด

# บทที่ 3
# การวิเคราะห์ปัญหา และขั้นตอนการพัฒนาอัลกอลิทึม

3.1 การวิเคราะห์ปัญหา

วิเคราะห์ปัญหาข้างต้นที่เกิดขึ้นจากการกระทำโดยใช้ Image Processing พื้นฐานได้แก่การทำ threshold แล้วนำมาทำ segmentation และ feature extraction(โดยวิธีการหาจุดศูนย์กลางมวล) กับรูปที่ทำการสร้างขึ้นจากโปรแกรม 3d studio เพื่อทำการระบุหาตำแหน่งของวัตถุที่มีอยู่ในรูปแล้วทำการหาจุดศูนย์มวลของวัตถุที่ได้จากการทำ segmentation จะพบได้ว่าปัญหาที่เกิดขึ้นคือการทำ threshold เพื่อแยกวัตถุออกจากพื้นหลังนั้นไม่สามารถแยกวัตถุทั้งหมดออกจากพื้นหลังได้ จะมีส่วนด้านหนึ่งของวัตถุที่เกิดเงาขึ้นโดนกลืนไปเป็นพื้นหลัง ทำให้รูปของวัตถุผิดเพี้ยนไปดังรูปที่ 3.1 ทำให้การหาจุดศูนย์กลางจากการหาจุดศูนย์กลางมวลไม่ใช่จุดศูนย์กลางที่แท้จริงของวัตถุ

**3.2 ขั้นตอนการพัฒนาอัลกอลิทึม**

ขั้นตอนการพัฒนาอัลกอลิทึมที่ใช้อยู่นั้นเป็นแบบ Iterative คือ

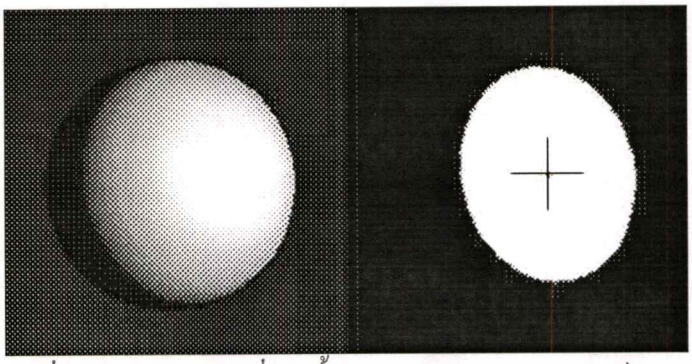
3.11 ทำการออกแบบอัลกอลิทึมเพื่อทำการแก้ไขปัญหาที่พบอยู่ในปัจจุบัน

3.12 นำอัลกอลิทึมที่ทำการออกแบบไปทำการ Implement

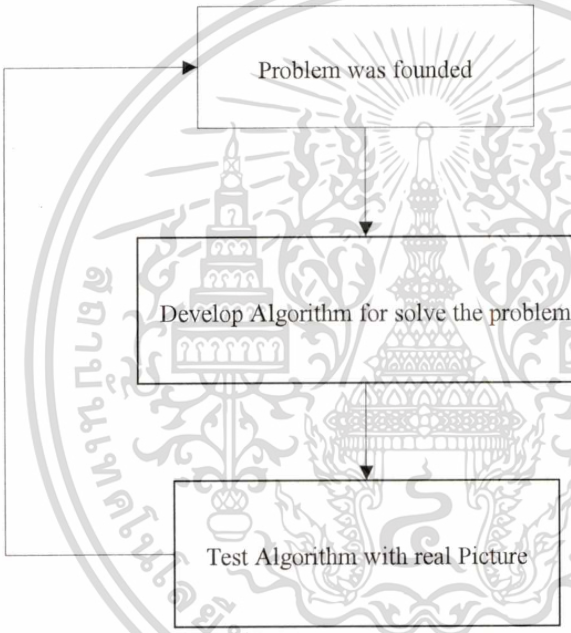
3.13 นำอัลกอลิทึมที่ได้ไปทำการทดสอบกับรูปภาพที่ได้จากการถ่ายรูปจริงในสภาวะที่อยู่ภายใต้ข้อตกลงที่ว่า

- วัตถุในรูปจะมีอยู่ 2 ชิ้นแต่ละชิ้นมีสีที่ต่างกัน(ในที่นี้กำหนดให้เป็นสีแดงและขาว)
- พื้นหลังที่ใช้จะต้องเป็นสีดำที่มีการสะท้อนของแสงต่ำ(ในที่นี้ใช้ผ้าสักหลาดสีดำ)

3.14 หากพบปัญหาที่เกิดขึ้นภายใต้อัลกอลิทึมที่ทำการออกแบบ กลับไปทำซ้ำที่การออกแบบอัลกอลิทึมอีกครั้งหนึ่ง



รูปที่ 3.1 แสดงปัญหาที่เกิดขึ้นจากการทำการหาจุดศูนย์กลางโดยการใช้วิธี Center Of Mass

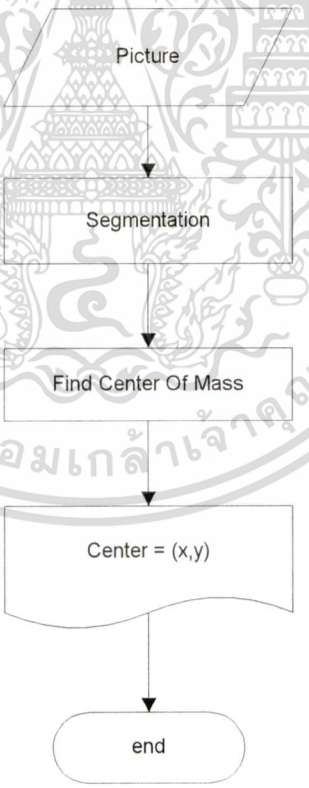



รูปที่ 3.2 แสดงขั้นตอนการพัฒนาอัลกอลิทึม

# บทที่ 4
## การพัฒนาอัลกอลิทึม

การพัฒนาอัลกอลิทึมที่เราจะใช้นั้นเป็นแบบ Iterate คือทำการออกแบบอัลกอลึทึมแล้วนำไปทำการทดสอบเมื่อทำการทดสอบแล้วพบปัญหาใหม่เกิดขึ้นก็นำไปทำการออกแบบอัลกอลิทึมใหม่ซึ่งครอบคลุมปัญหาที่เกิดขึ้นแล้วจึงนำไปทำการทดสอบอีกครั้ง

ส่วนการออกแบบอัลกอลิทึมเพื่อทำการแก้ปัญหาในแต่ละหัวข้อที่เกิดขึ้นนั้น อัลกอลิทึมจะถูกออกแบบให้เป็นฟังก์ชันย่อยๆแล้วนำมาเพิ่มในฟังก์ชันหลัก หรือเพิ่มฟังก์ชันย่อยในส่วนของฟังก์ชันที่เป็นปัญหาเพื่อใช้แก้ไขปัญหาที่เกิดขึ้น




รูปที่ 4.1แสดง Flow Chart ขอการอัลกอลิทึมหลัก

จำนวนครั้งและรูปแบบการพัฒนาอัลกอลิทึมจะเพิ่มเติมขึ้นตามปัญหาที่พบหลังจากการทดสอบอัลกอลิทึมซึ่งปัญหาต่างๆที่พบตลอดการทำการทดลองและรายละเอียดของแต่ละอัลกอลิทึมที่ถูกออกแบบเพิ่มเติมขึ้นมาจาก อัลกอลิทึมหลักมีดังนี้

**Problem 1 : ปัญหาจากการเกิดเงาบนวัตถุ**

จากการทดลองอัลกอลิทึมเบื้องต้นที่กล่าวมาในบทที่ 3 นั้นจะพบปัญหาที่เกิดขึ้นคือ การถูกกลืนของตัววัตถุโดยเงาทำให้ในช่วงการทำ threshold นั้นส่วนของวัตถุที่เป็นเงานั้นถูกระบุว่าเป็นพื้นหลังแทนที่จะเป็นวัตถุ ทำให้วัตถุที่ได้จากการ segmentation นั้นไม่ใช่พื้นที่ของวัตถุทั้งหมด เมื่อนำไปทำการหาจุดศูนย์กลางมวลแล้ว ทำให้ค่าที่ได้ออกมานั้นผิดเพี้ยนไปจากจุดศูนย์กลางจริงๆของวัตถุ

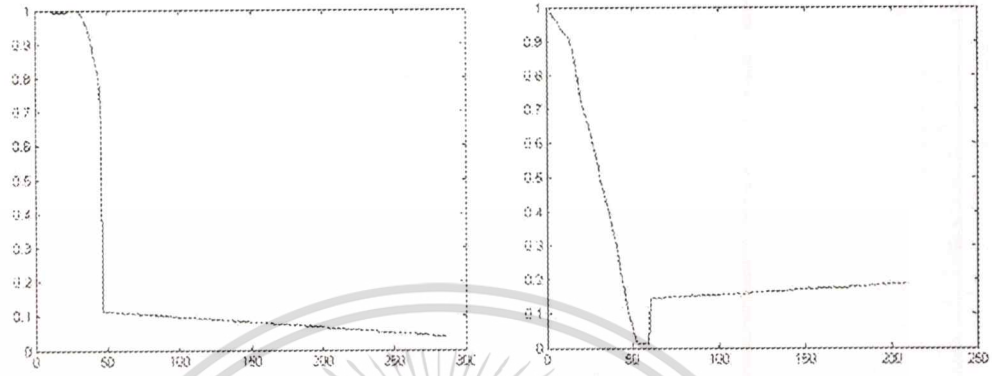
รูปที่ 4.2 แสดงปัญหาที่เกิดขึ้นใน Problem 1

**Algorithm 1 : การหาจุดศูนย์กลางของวัตถุทรงกลมในรูปโดยใช้จุดของเส้นตั้งฉากกับเส้นสัมผัสของขอบที่แท้จริง**

หลังจากการทดลองพบปัญหาที่ 1 นั้นหนทางแก้ไขที่นำมาใช้เพื่อให้หาจุดศูนย์กลางให้ได้เที่ยงตรงมากขึ้นคือ แนวทางการหาจุดศูนย์กลางจากการตัดกันของเส้นตั้งฉากกับเส้นสัมผัสวงกลม โดย

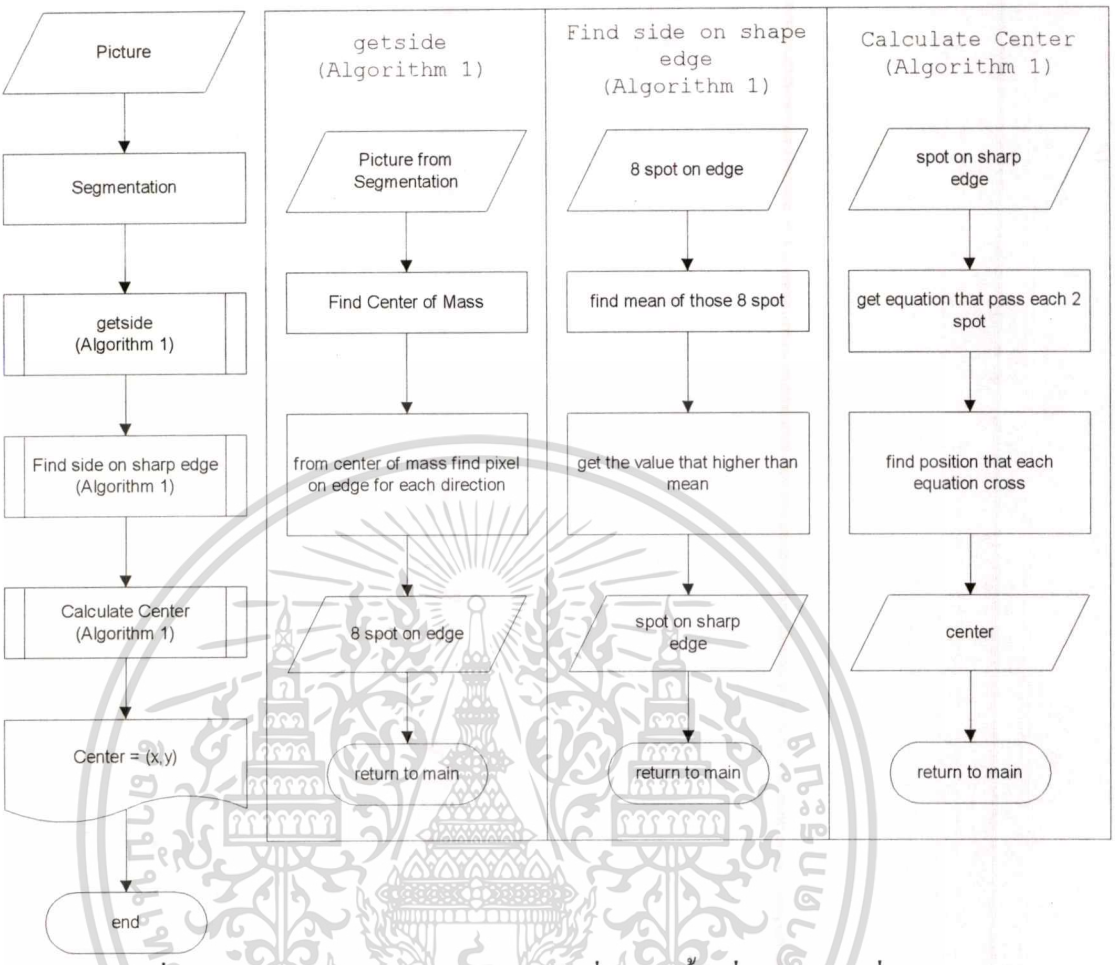
1. นำค่าที่ได้จากการหาจุดศูนย์กลางมวลเป็นจุดอ้างอิงแล้วทำการไล่ค่าความสว่างออกไปทั้ง 8 ทิศเพื่อหาค่าความแตกต่างกันของค่าความสว่างของพิกเซลที่ติดกัน เพื่อดูว่าจุดใดในแต่ละทิศเป็นจุดที่ตั้งอยู่บนเส้นขอบของวงกลมโดยดูจากค่าความแตกต่างสูงสุด ซึ่งจะทำให้ได้จุดที่คาดว่าจะอยู่บนเส้นขอบมาทั้งหมด 8 จุด
2. นำค่าความแตกต่างของทั้ง 8 จุดที่ได้มาหาค่า mean
3. ทำการเลือกเอาเฉพาะค่าที่มีค่าสูงกว่า mean มาเพื่อนำไปใช้ในการหาเส้นตั้งฉากของจุดสัมผัสแต่ละจุด สาเหตุที่ต้องทำการเลือกอีกทีเนื่องจากในบางจุดบนเส้นขอบที่ได้จากการหาโดยวิธีนี้อาจไม่ใช่จุดที่บนเส้นของจริงๆเนื่องจาก ค่าความต่างสูงสุดของวัตถุในด้านที่เป็นเงาจะอยู่ในส่วนที่เป็นเนื้อวัตถุส่วนที่เป็นเงา ไม่ใช่ส่วนขอบจริงๆของวัตถุ ดัง

นั้นจึงจำเป็นต้องมีการตัดส่วนนี้ทิ้งไปโดยดูจากค่าความต่างสูงสุดในส่วนที่เป็นขอบจริงจะมีค่าสูงกว่าค่าความต่างสูงสุดสูงกว่าในส่วนที่เป็นเงาดังกราฟในรูปที่ 4.3

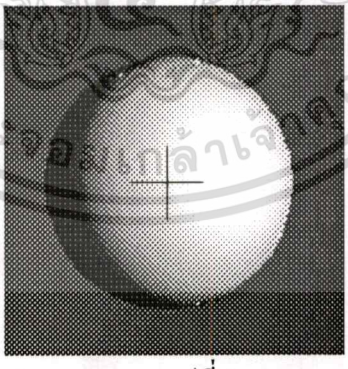


รูปที่ 4.3 กราฟแสดงค่าความสว่างของแต่ละพิกเซลในวัตถุระหว่างส่วนขอบด้านที่ไม่เป็นเงา (ซ้าย) กับด้านที่เป็นเงา(ขวา)

4. เมื่อได้ค่าที่อยู่บนเส้นขอบแล้วนำเอามาเข้าสมการเพื่อที่จะหาตำแหน่งจุดตัดของเส้นตั้งฉากกับกึ่งกลางของสมการเส้นตรงที่เชื่อมเส้นขอบ (จุดตัดของสมการเส้นตั้งฉากกับเส้นสัมผัส) ค่าจุดตัดสมการที่ได้จะเป็นค่าจุดศูนย์กลางของวงกลม(ซึ่งก็คือรูปทรงกลมภายในภาพ)ที่หาได้จากอัลกอลิทึมนี้




รูปที่ 4.4 แสดง flow chart ของอัลกอลิทึมที่พัฒนาขึ้นเพื่อแก้ปัญหาที่ 1



รูปที่ 4.5 แสดงผลจากการทำการทดลองรูปที่ทำการทดลองสร้างกับอัลกอลิทึมที่ 1

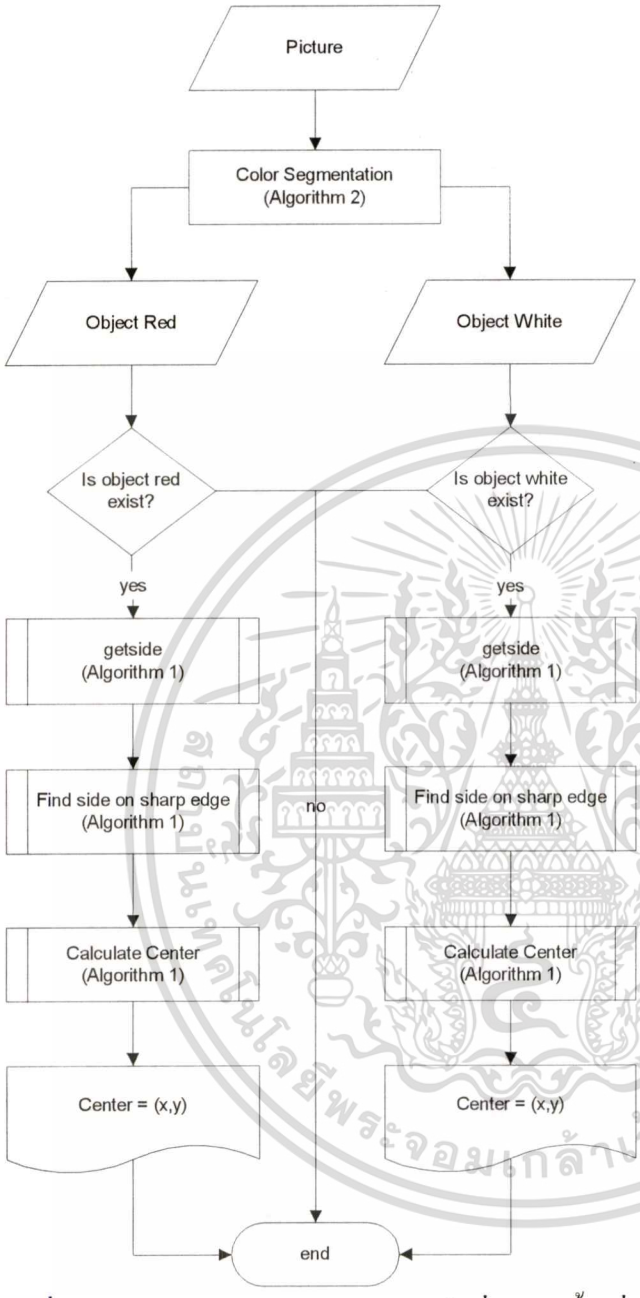
**Problem 2 : ปัญหาจากการตรวจจับวัตถุ 2 อันในภาพ**

ปัญหาต่อมาหลังจากที่ได้ทำการทดลองอัลกอลิทึมที่ทำการปรับปรุงข้างต้นแล้ว ไม่มีปัญหากับรูปที่สร้างขึ้นจาก 3d studio แต่ปัญหาที่เกิดขึ้นต่อมาก็คือการที่มีวัตถุ 2 อันอยู่ในรูปเดียวกัน ทำให้ต้องทำการเพิ่มเติมในส่วนของการทำ segmentation เพื่อทำการระบุวัตถุแยกจากกันเป็น 2 อัน

**Algorithm 2 : Color Segmentation**

เพื่อให้สามารถแยกวัตถุออกจากกันได้ในกรณีที่วัตถุอยู่ในสภาพซ้อนทับกันหรืออยู่ติดกัน จึงหลีกเลี่ยงการใช้วัตถุสีเดียวกัน ซึ่งทางผู้เขียนกำหนดวัตถุให้เป็นสีต่างกัน 2 สี คือ สีแดงและสีขาว อัลกอลิทึมที่นำมาใช้ในการทำ color segmentation เพื่อทำการแยกวัตถุ 2 สีออกจากกันมีดังนี้

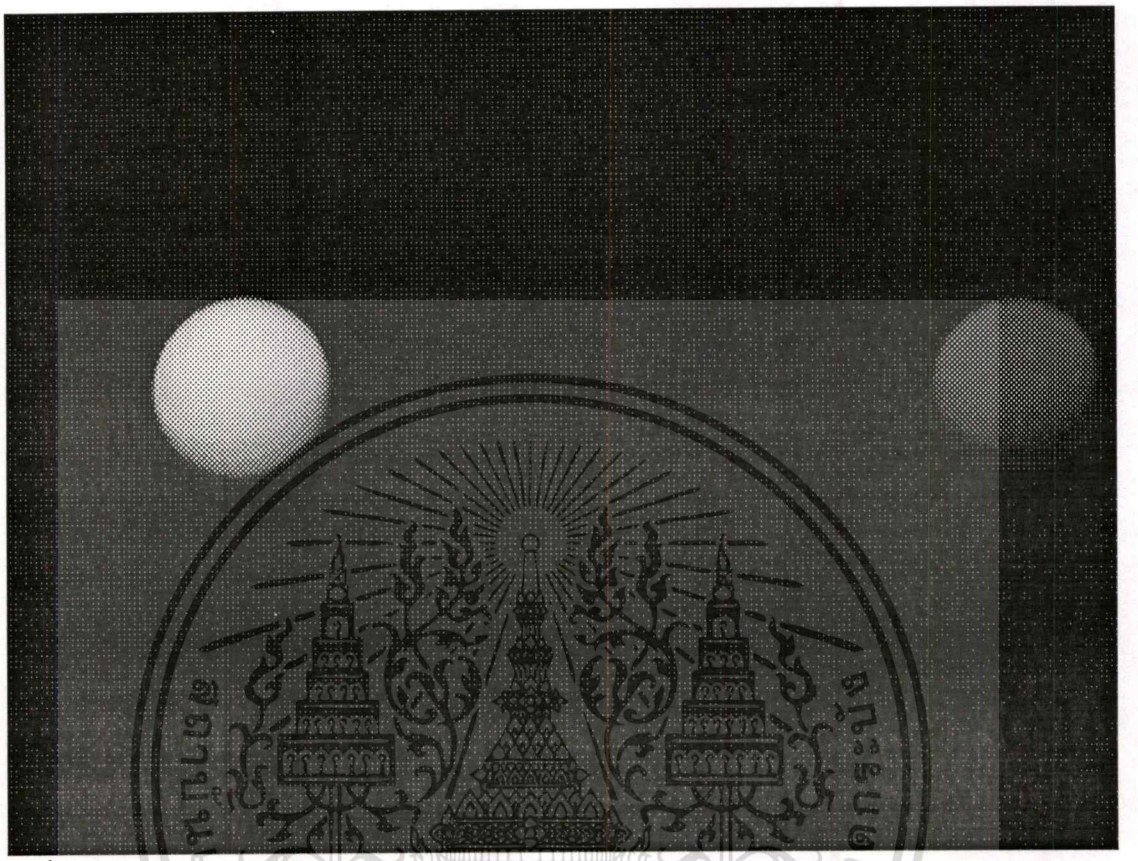
1. นำรูปที่ทำการอ่านมาได้มาทำการแยกสีออกเป็น 3 รูปคือส่วนรูปขาวดำของ red green และ blue ซึ่งแต่ละรูปก็คือความสว่างของแต่ละแม่สี ในภาพ
2. นำรูปขาวดำทั้ง 3 ส่วนที่ได้มาทำการ threshold เพื่อทำการแปลงให้อยู่ในรูปขาวดำ ซึ่งรูปขาวดำที่ได้จะแบ่งออกเป็นส่วนของวัตถุ(ส่วนที่เป็นสีขาว)และส่วนของพื้นหลัง(ส่วนที่เป็นสีดำ) ซึ่งหากเป็นวัตถุสีแดง หมายความว่าส่วนของรูปที่เป็นของ red นั้นเมื่อนำเอาค่าในตำแหน่งนั้นมาหักออกด้วยค่าในรูปของ green และ blue ก็จะยังคงมีค่าเป็นสีขาวอยู่ และหากส่วนของวัตถุนั้นเป็นสีขาวก็ต่อเมื่อ ค่าในแต่ละตำแหน่งของแต่ละรูปทั้ง 3 รูปมีค่าเป็น 1
3. แล้วจึงนำค่าตำแหน่งของวัตถุแต่ละสีไปทำการหาตำแหน่งบนเส้นขอบและจุดศูนย์กลางของภาพจากอัลกอลิทึมเดิม



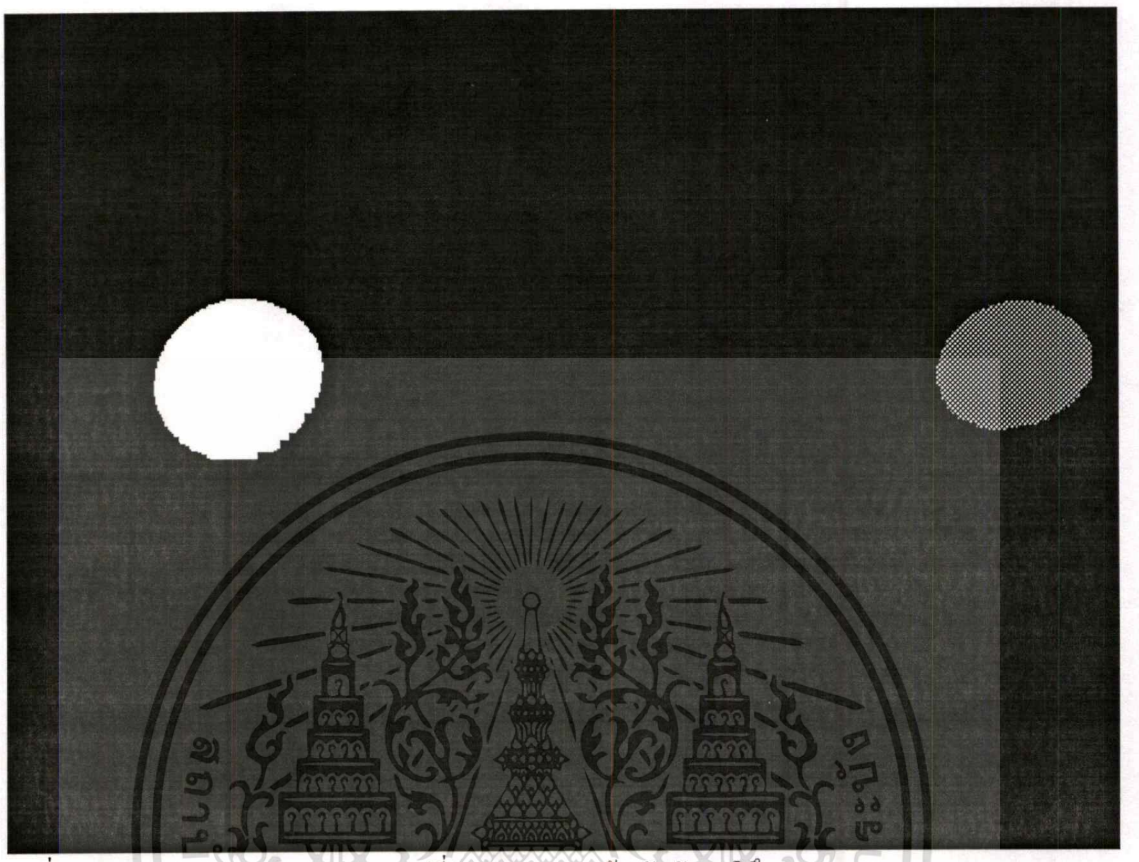

```
Color Segmentation
(Algorithm 1)

!-- picture contain with red green blue
level = graythresh(Picture[].red)
ObjRed[] = im2bw(Picture[].red,level)
level = graythresh(Picture[].green)
ObjGreen[] = im2bw(Picture[].green,level)
level = graythresh(Picture[].blue)
ObjcBlue[] = im2bw(Picture[].blue,level)
Object[] = ObjRed[]-ObjGreen[]-ObjBlue[]
for each spot in Object[i,j]
        if Object[i,j] < 0
                Object[i,j] = 0
ObjectRed[] = Object[]
ObjectWhite[] = ObjRed[]&ObjGreen[]&ObjBlue[]
return ObjectRed[],ObjectWhite[]
```

รูปที่ 4.6 แสดง flow chart ของอัลกอลิทึมที่พัฒนาขึ้นเพื่อแก้ปัญหาที่ 2 และ pseudo code ในส่วนของการทำ Color Segmentation



รูปที่ 4.7 แสดงรูปตัวอย่างก่อนเข้าทำในอัลกอลิทึม Color Segmentation



รูปที่ 4.8 แสดงผลจากการทำการทดลองรูปที่ทำการทดลองสร้างกับอัลกอลิทึม Color Segmentation

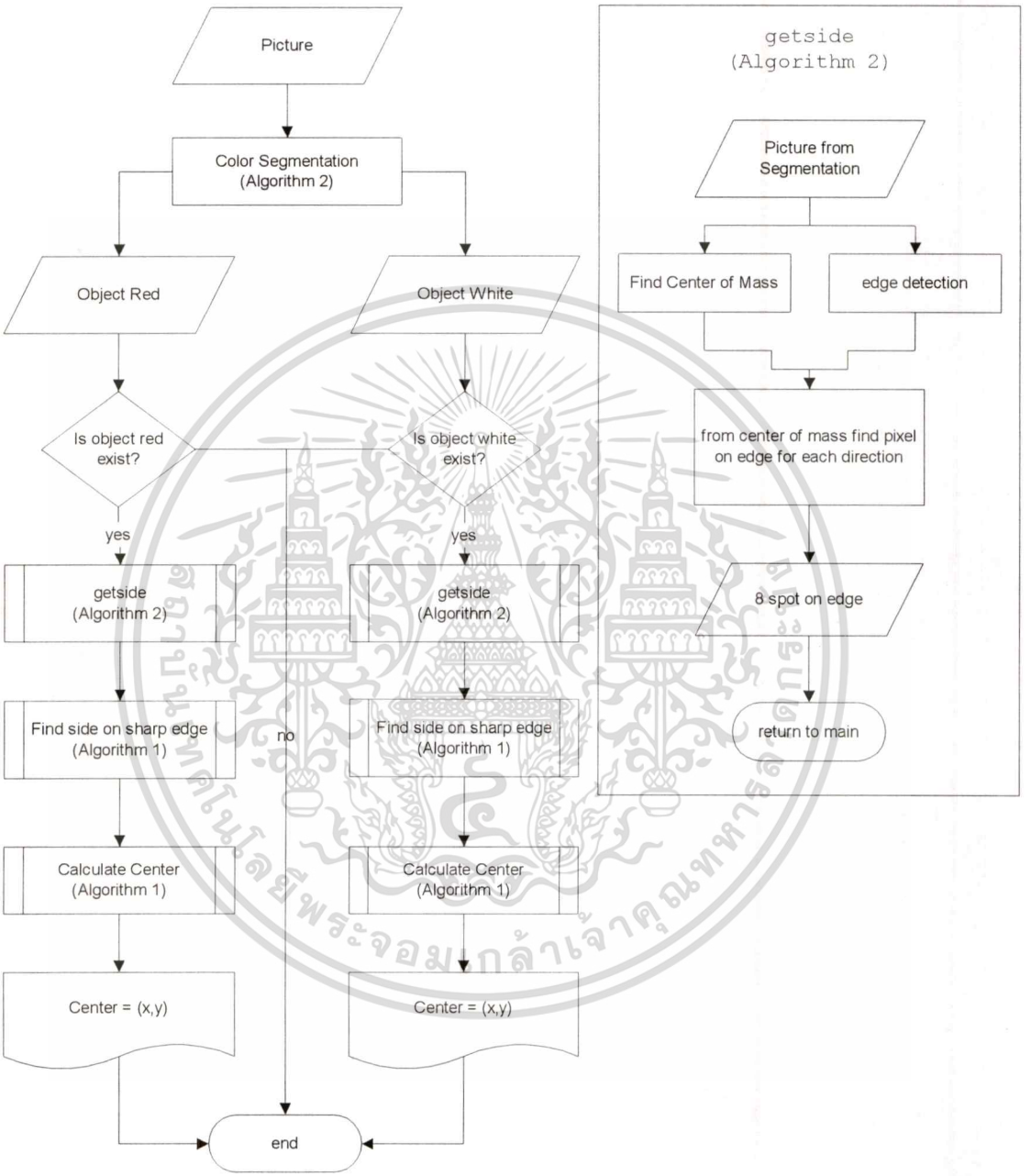
**Problem 3 : ปัญหาจากแสงสะท้อนบนพื้นผิวของวัตถุ**

เมื่อทำการทดลองอัลกอลิทึมข้างต้นที่กล่าวมากับรูปภาพที่ทำการสร้างขึ้นจากโปรแกรม 3d studio นั้นไม่พบปัญหาเกิดขึ้น สามารถระบุจุดศูนย์กลางของวัตถุทรงกลมภายในรูปภาพได้ใกล้เคียง มีค่าผิดพลาดเกิดขึ้นเล็กน้อย แต่หลังจากการนำเอาไปทดลองกับรูปถ่ายของวัตถุจริงซึ่งทางผู้เขียนได้เลือกนำเอาลูกสนุ๊กเกอร์มาทำการถ่ายภาพบนผ้าสักหลาดภายใต้สภาวะแสงต่างๆปัญหาที่พบตามมาก็คือ แสงไฟที่สะท้อนจากผิวของลูกสนุ๊กเกอร์สีแดง นั้นทำให้สีบนพื้นผิวลูกสนุ๊กเกอร์กลายเป็นสีขาวไป เมื่อนำไปทำการทดลองกับอัลกอลิทึมข้างต้นปัญหาจะพบตรงส่วนของ color segmentation เนื่องจากเราทำการเลือกวัตถุจากสีของวัตถุ เมื่อนำวัตถุไปทำการ detect หาเส้นขอบแล้ว แทนที่จะได้จุดบนเส้นขอบที่แท้จริงจุดที่ได้กลับเป็นจุดที่เป็นจุดที่แสงสะท้อนบนวัตถุสีแดง ทำให้เมื่อนำเอาไปคำนวณแล้วค่าของจุดศูนย์กลางที่ได้จากอัลกอลิทึมนั้นผิดเพี้ยนไป

**Algorithm 3 : การหาขอบที่แท้จริงโดยการนำเอา Edge Detection มาประยุกต์ใช้**

เพื่อที่จะทำการแก้ไขปัญหาการเกิดแสงสะท้อนของวัตถุทางผู้เขียนจึงเปลี่ยนวิธีการหาเส้นขอบของวัตถุโดยการนำเอาวิธี edge detection มาใช้โดยที่จากเดิมหาเส้นขอบจากการหาค่าแตกต่างสูงสุดมาใช้การไล่ไปจนกระทั่งพบขอบจากการทำ edge detection จากรูปขาวดำที่ได้จากการ

segmentation แล้วนำเอาค่าตำแหน่งบนเส้นขอบที่ได้ไปทำการหาจุดศูนย์กลางของวงกลมในรูปต่อไป

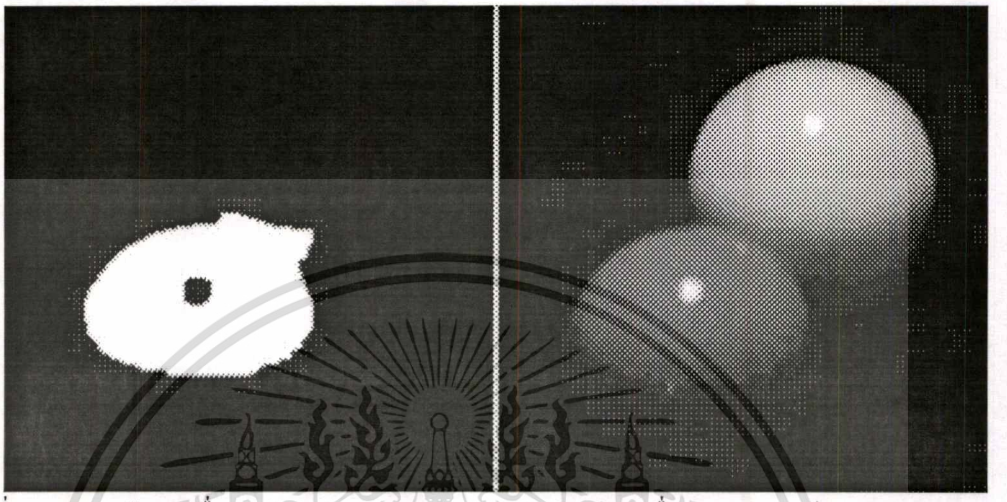



รูปที่ 4.9 แสดง flow chart ของอัลกอลิทึมที่พัฒนาขึ้นเพื่อแก้ปัญหาที่ 3

**Problem 4 : การสะท้อนของวัตถุชิ้นหนึ่งไปบนพื้นผิวของวัตถุอีกชิ้น**

หลังจากทำการทดลองอัลกอลิทึมที่กล่าวมาในข้างต้น สามารถแก้ไขปัญหาของแสงสะท้อนในกรณีที่วัตถุไม่ติดกันได้ แต่เมื่อในกรณีที่วัตถุอยู่ติดกันหรือซ้อนกันนั้น ปัญหาที่เกิดขึ้นอีกก็คือ สีของวัตถุหนึ่งไปสะท้อนลงบนผิวของวัตถุอีกอันหนึ่ง ทำให้การทำ color segmentation นั้นติดสีใน

ส่วนที่สะท้อนลงไปในวัตถุมาด้วย เมื่อนำมาทำการหาเส้นขอบจากอัลกอลิทึมที่ได้ทำการปรับปรุงในข้างต้น จะทำให้จุดบนเส้นขอบที่ได้ไปเป็นจุดในวัตถุสีขาวไม่ใช่จุดบนเส้นขอบจริงๆ เมื่อนำเอาจุดนั้นไปใช้หาจุดศูนย์กลางวงกลมทำให้จุดศูนย์กลางที่ได้เพี้ยนไป



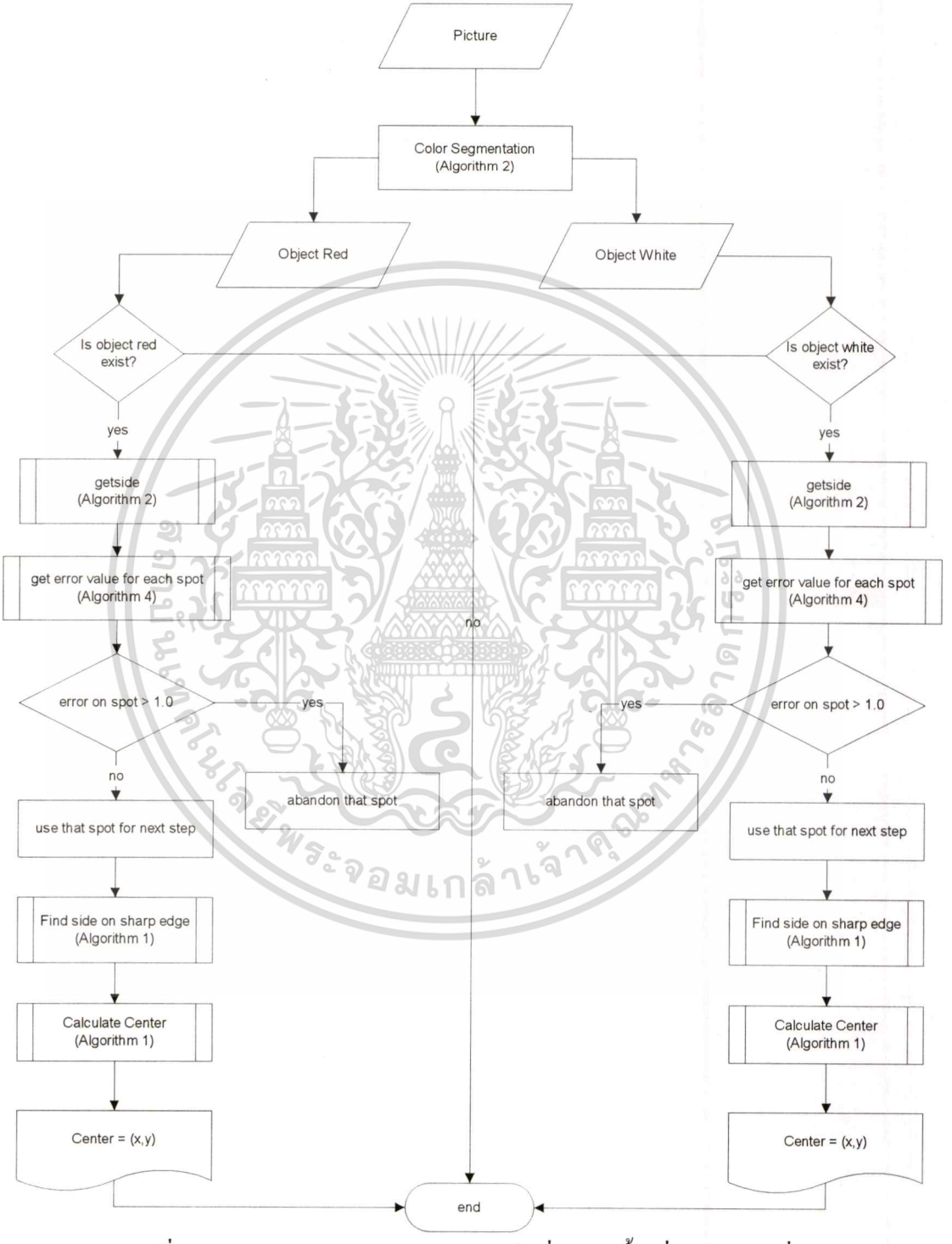
รูปที่ 4.10 แสดงปัญหาเนื่องจากการทำ Color Segmentation ผิดพลาดเนื่องจากการสะท้อนของแสง

**Algorithm 4 : การตัดขอบที่เกิด error ออกโดยดูจากค่าความคงที่ของเส้นโค้ง**

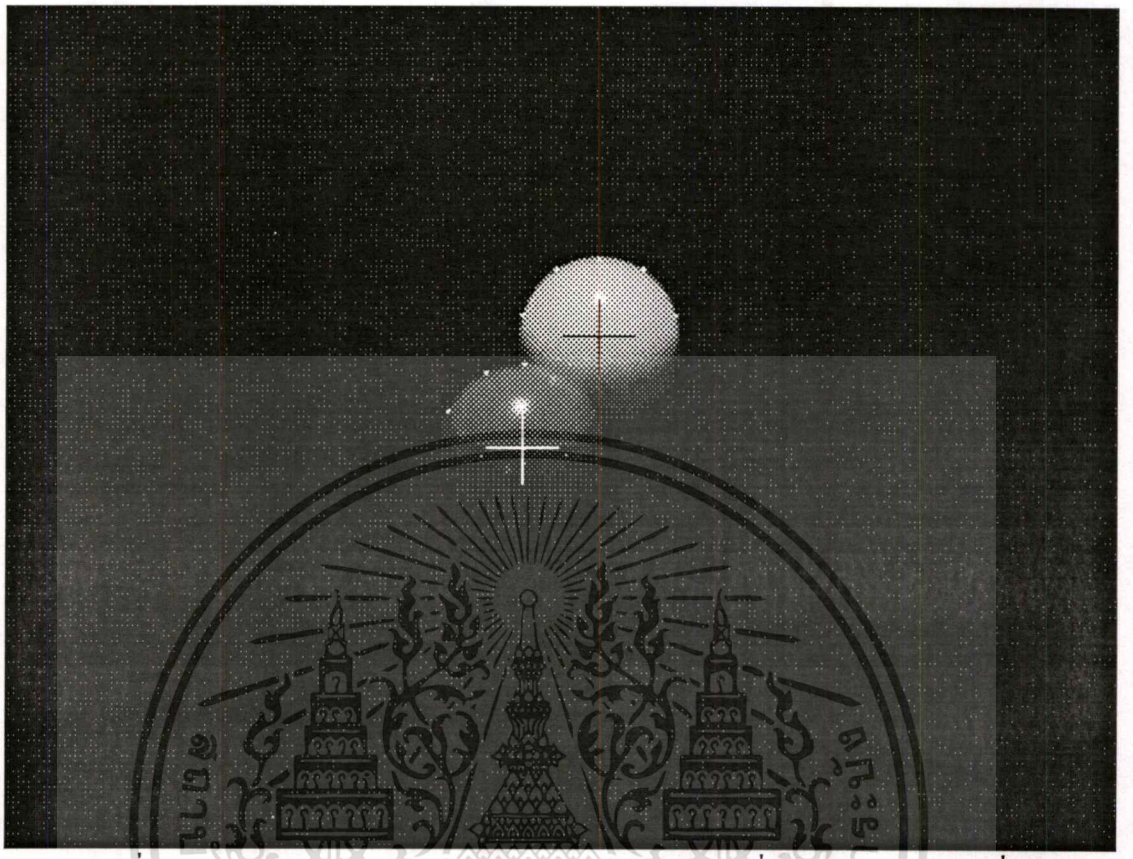
ปัญหาที่เกิดขึ้นนั้นผู้เขียนได้ทำการแก้ไขโดยสร้างฟังก์ชันย่อยเพื่อไว้ใช้ดูว่า ค่าบนเส้นขอบที่ได้นั้นใช่ค่าที่อยู่บนส่วนโค้งของขอบจริงๆหรือไม่หรือเป็นส่วน error ที่เกิดขึ้นจากแสงที่ตกกระทบไปยังอีกวัตถุหนึ่ง ซึ่งอัลกอลิทึมมีหลักการทำงานดังนี้

1. ทำการหาค่าจุดทั้งหมดที่เป็นเส้นขอบที่ได้จากการทำ edge detection
2. นำค่าขอบที่อยู่ระหว่างจุดที่ต้องการทำการหาค่า error กับจุดถัดไปเข้าไปทำฟังก์ชันหาจุดศูนย์กลางของวงกลมค่าที่ได้จากการนำค่าที่อยู่บนขอบ ไปเข้าฟังก์ชันนั้นเป็นเพียงจุดที่มีระยะห่างจากจุดบนเส้นขอบใกล้เคียงกันอาจไม่ใช่ค่าที่เป็นจุดศูนย์กลางของวงกลมก็ได้
3. นำตำแหน่งที่ได้จากการใช้ฟังก์ชันหาจุดศูนย์กลางของวงกลมไปทำการหาระยะห่างระหว่างจุดที่ได้กับจุดบนเส้นขอบทั้งหมด
4. นำค่าระยะห่างทั้งหมดมาหาค่าเฉลี่ย
5. หาค่า standard deviation โดยการหาจาก ผลรวมของ absolute(ระยะห่างเฉลี่ย-ระยะห่าง) หารด้วยจำนวนจุดเส้นขอบทั้งหมดที่นำเข้ามาคิดในฟังก์ชันนี้
6. ค่า standard deviation ที่ได้คือค่า error ที่เราต้องการ
7. หากค่า error ของตำแหน่งบนเส้นขอบใดๆมีค่าเกินกว่า 1.0 ให้ตัดทิ้ง ไม่นำไปใช้ในการคำนวณหาจุดศูนย์กลาง

ค่า 1.0 ที่นำมาใช้ในการตัดสินเกิดจากการทดลองเก็บข้อมูลจากรูปตัวอย่างแล้วนำมาทำการประมาณหาค่าแบ่งแยกระหว่างจุดที่ error กับจุดที่ไม่ error



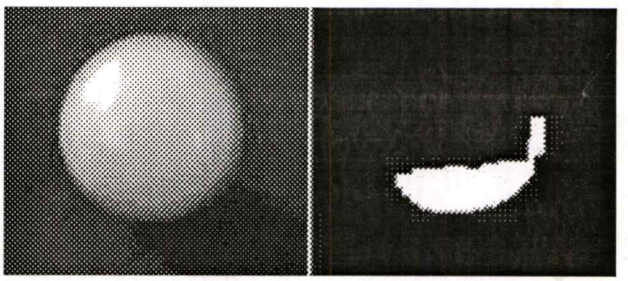

รูปที่ 4.11 แสดง flow chart ของอัลกอลิทึมที่พัฒนาขึ้นเพื่อแก้ปัญหาที่ 4



รูปที่ 4.12 แสดงผลจากการทำการทดลองรูปตัวอย่างในรูปที่ 4.10 กับอัลกอลิทึมที่ 4

**Problem 5 : ปัญหาที่เกิดขึ้นเนื่องจากสภาพแวดล้อมที่มีแสงสว่างสูง**

หลังจากการนำเอาอัลกอลิทึมที่ได้ไปทำการทดลองกับรูปที่ได้จากการถ่ายภาพโดยกล้องดิจิตอลแล้วสามารถแก้ไขปัญหาการสะท้อนของสีวัตถุหนึ่งไปยังอีกวัตถุหนึ่งได้ในระดับหนึ่ง แต่อย่างไรก็ได้ปัญหาที่พบตามมาจากการนำรูปมาใช้ในการทดลองคือ ในกรณีที่มีแหล่งกำเนิดแสงมากและขนาดใหญ่ ตัวอย่างเช่นแสงแดดจากหน้าต่างห้องมากระทบวัตถุสีแดง สีของวัตถุสีแดงในส่วนที่เกิดแสงตกกระทบนั้นจะเปลี่ยนเป็นสีขาว จากการนำไปทำ color segmentation รูปที่ได้ในส่วนของสีแดงจะถูกกลืนไปเป็นสีขาว เนื่องจากการ segmentation นั้นได้นำภาพขาวดำของส่วนแดง-(เขียว+น้ำเงิน) หากสีใดสีหนึ่งระหว่างเขียวและน้ำเงินมีความสว่างสูง(เนื่องจากปริมาณแสงตกกระทบวัตถุสูง)ส่วนนั้นก็จะถูกมองเป็นสีขาวไป



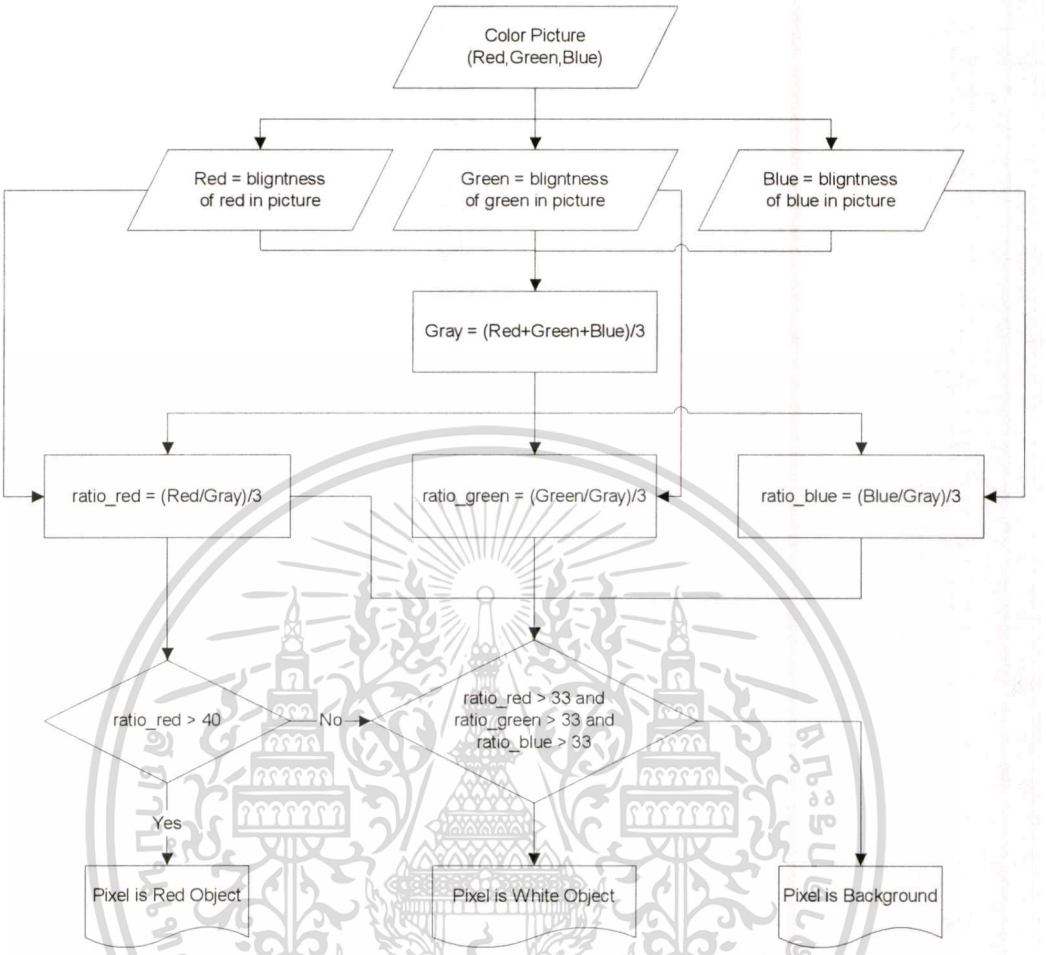
รูปที่ 4.13 แสดงปัญหาที่เกิดขึ้นจากการทำ color segmentation ในกรณีที่มีแสงตกกระทบลงวัตถุแดงในปริมาณสูง

**Algorithm 5 : การปรับปรุง Color Segmentation โดยใช้อัตราส่วนของแต่ละสีมาใช้ในการแยกแยะ**

การแก้ปัญหาที่ได้พบนั้นทำได้โดยการเขียนฟังก์ชันการทำ color segmentation ใหม่โดยที่ฟังก์ชันนี้มีพื้นฐานการคำนวณจากอัตราส่วนของสีต่างๆเพื่อนำเอาไปแยกสีแดงและสีขาวออกจากกันให้มีประสิทธิภาพมากขึ้น ฟังก์ชันดังกล่าวมีการทำงานดังนี้

1. ทำการ threshold เพื่อทำการแยกเอาวัตถุกับพื้นหลังออกจากกัน
2. นำส่วนที่เป็นวัตถุมาทำการพิจารณาดังนี้
3. หากเป็นสีแดง อัตราส่วนระหว่างความสว่างของสีแดงต่อสีอื่นๆ 50% ให้ถือว่าส่วนของวัตถุ ณ จุดนี้เป็นวัตถุสีแดง
4. ถ้าไม่ใช่ ให้ทำการตรวจสอบอัตราส่วนระหว่างค่าความสว่างของ สีแดง:สีเขียว:สีน้ำเงิน ซึ่งจะมีอัตราส่วนใกล้เคียงกัน จะถือว่าเป็นสีขาว
5. นำตำแหน่งของวัตถุแต่ละสีที่ได้จากการทำ color segmentation ใหม่นี้ไปทำการหาจุดศูนย์กลางต่อไป

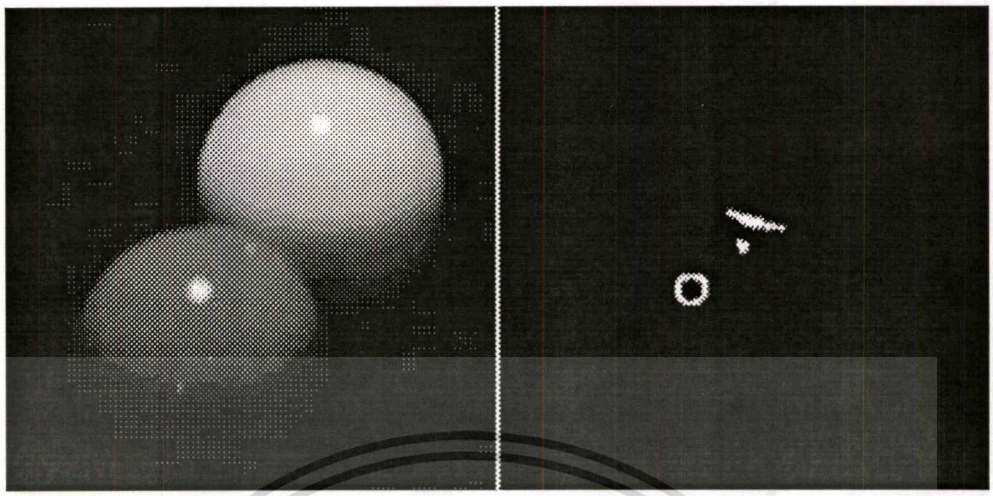
รูปที่ 4.14 แสดงผลจากการทำการทดลองรูปที่ทำการทดลองสร้างกับอัลกอลิทึมที่ 5




รูปที่ 4.15 flowchart แสดงการทำ Color Segmentation โดยดูจากอัตราส่วนของสีในรูปภาพ

**Problem 6 : ปัญหาต่อเนื่องจากการนำเอาอัลกอลิทึม Color Segmentation ใหม่มาใช้แทนอัลกอลิทึมเก่า**

จากการแก้ปัญหาในขั้นตอนที่ได้กล่าวมาแล้วจะสามารถระบุวัตถุสีต่างๆ(ในส่วนของขอบที่คมชัด)ได้อย่างถูกต้อง แต่ก็ยังมีปัญหาที่เกิดขึ้นอีกประการคือในกรณีที่แสงมีความสว่างสูง อัตราส่วนของสีแดงต่อสีอื่นๆเป็นไปแบบ 40:30:30 นั้นวัตถุจะไม่ถูก select เป็นสีใดเลยทั้งสีขาวแล้วสีแดง

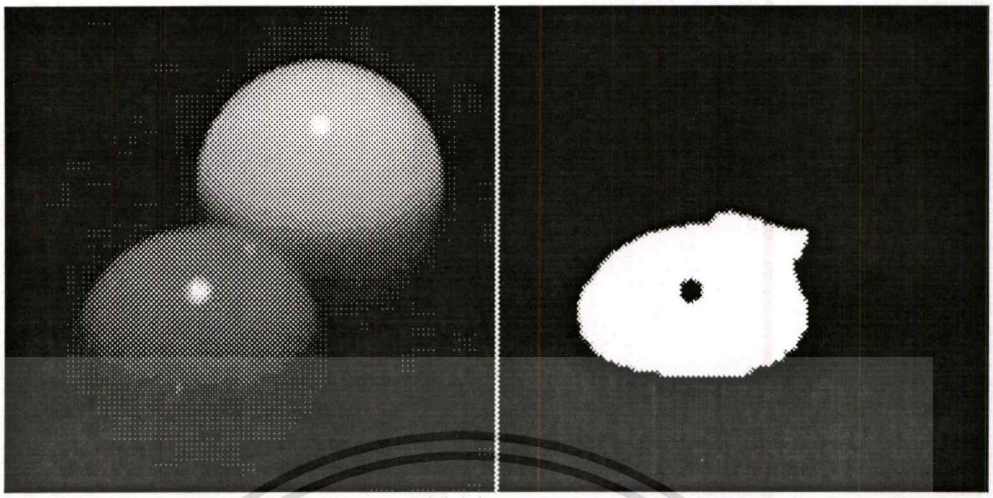


รูปที่ 4.16 แสดงผลจากการทำ Color Segmentation ในส่วนสีแดงจากรูปตัวอย่างด้วยอัลกอลิทึม Color Segmentation โดยการดูจากอัตราส่วนของสี

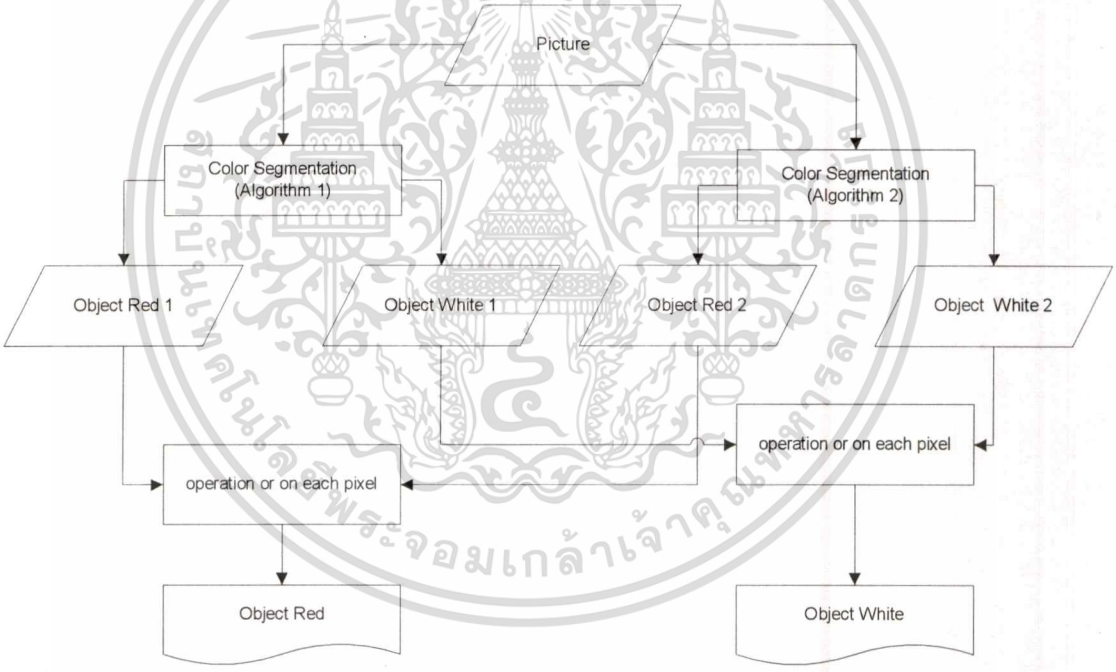
**Algorithm 6 : การรวมกันระหว่างอัลกอลิทึม Color Segmentation ใหม่และ Color Segmentation เก่า**

อัลกอลิทึมที่นำมาใช้ในการแก้ปัญหาสุดท้ายนี้เป็นเพียงการรวมกันระหว่างข้อดีของการทำ color selection ในแบบแรกและการทำ color selection แบบหลังกรณีของ color selection ในแบบแรกหากเป็นสีแดงสว่างทั้งวัตถุจะสามารถตรวจสอบเห็นได้อย่างชัดเจนแต่ปัญหาจะอยู่ที่ในกรณีที่เป็นสีแดงและมีแสงสว่างจำนวนมากสะท้อนอยู่บนพื้นผิวจะไม่สามารถตรวจสอบตัววัตถุเต็มๆเพื่อทำการหาขอบที่ถูกต้องได้

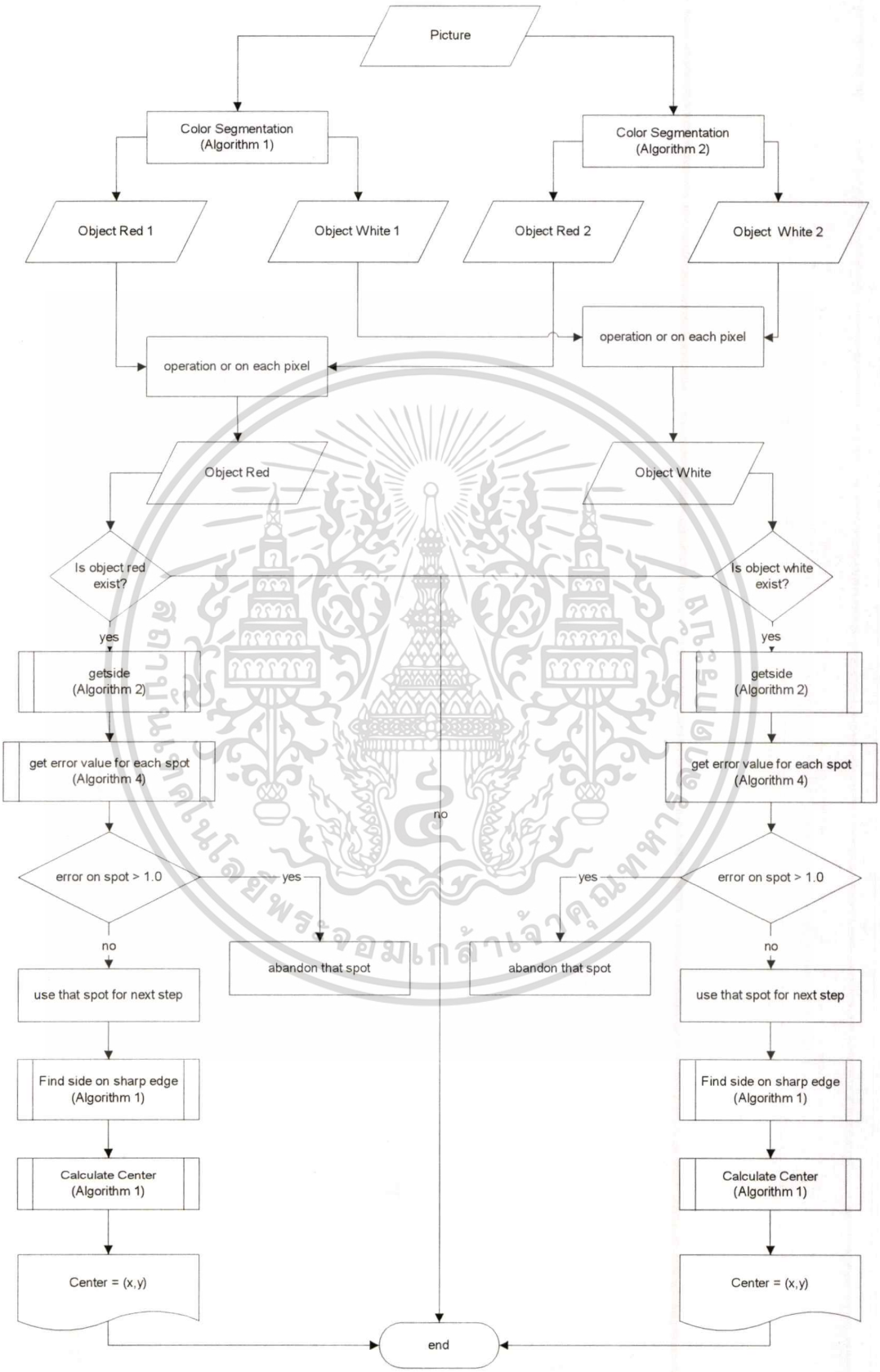
แต่ในกรณีของการทำ color selection แบบหลังนั้นจะสามารถตรวจสอบในกรณีกลับกันของการ segmentation ในแบบแรก เมื่อนำข้อดีของทั้งสองแบบมารวมกันก็จะทำให้ข้อเสียที่เกิดขึ้นหมดไป เมื่อทำการทดลองจะพบว่าเกิดข้อผิดพลาดในการหาจุดศูนย์กลางต่ำลงจากอัลกอลิทึมข้างต้น(การทดลองทำกับรูปภาพชุดเดียวกัน)



รูปที่ 4.17 แสดงผลจากการทำ Color Segmentation ในส่วนสีแดงจากรูปตัวอย่างด้วยอัลกอลิทึม Color Segmentation ที่รวมกันระหว่างของเก่าและใหม่




รูปที่ 4.18 แสดง flowchart ของการทำ color segmentation โดยรวมเอาอัลกอลิทึมเก่า(Algorithm 1) และอัลกอลิทึมใหม่(Algorithm 2)เข้าด้วยกัน




**รูปที่ 4.19 แสดง flowchart ของอัลกอลิทึมในการหาจุดศูนย์กลางหลังจากผ่านการพัฒนาแล้ว**

# บทที่ 5

## ผลจากการทำการทดลองและบทสรุป

การทดลองที่กล่าวถึงคือการทดสอบความแม่นยำในด้านต่างๆของอัลกอลิทึมกับรูปภาพที่ทำการถ่ายมาจากวัตถุภายใต้สภาวะแสงและตำแหน่งที่แตกต่างกันออกไปเพื่อให้ได้ผลพิสูจน์ว่าอัลกอลิทึมที่ทำการพัฒนานั้นสามารถนำไปใช้ได้จริงได้อย่างถูกต้องแล้วพึงพอใจในระดับหนึ่ง

สภาวะแสงที่ทำการทดลองนั้นมีทางผู้เขียนได้ทำการทดลองถ่ายรูปวัตถุภายใต้สภาวะแสงที่ต่างกันคือช่วงกลางวันซึ่งจะมีแหล่งแสงจากส่วนของหน้าต่างและช่วงกลางคืนซึ่งจะมีแหล่งแสงจากไฟนีออนแบบกลมโดยจะวางวัตถุให้อยู่ในตำแหน่งที่แตกต่างกันออกไปเพื่อให้เกิดการกลืนกินของเงาเป็นสัดส่วนที่ต่างกันออกไป

5.1 ข้อกำหนดเพิ่มเติม

5.11 วัตถุมีจำนวนมากที่สุดคือ 2 ชิ้นแต่ละชิ้นมีสีที่ต่างกันออกไป คือ สีแดงและสีขาว

5.12 พื้นหลังคือสีดำซึ่งมีการสะท้อนแสงต่ำ ในที่นี้ใช้ผ้าสักหลาดสีดำ

5.13 รูปภาพที่ได้ได้จากการถ่ายรูปด้วยกล้องดิจิตอลที่มีรายละเอียดขนาด 3 แสนพิกเซล 640x480 มีกำลังการซูมแบบ optical zoom สูงสุด 3 เท่า

**5.2 สมมติฐานเบื้องต้น**

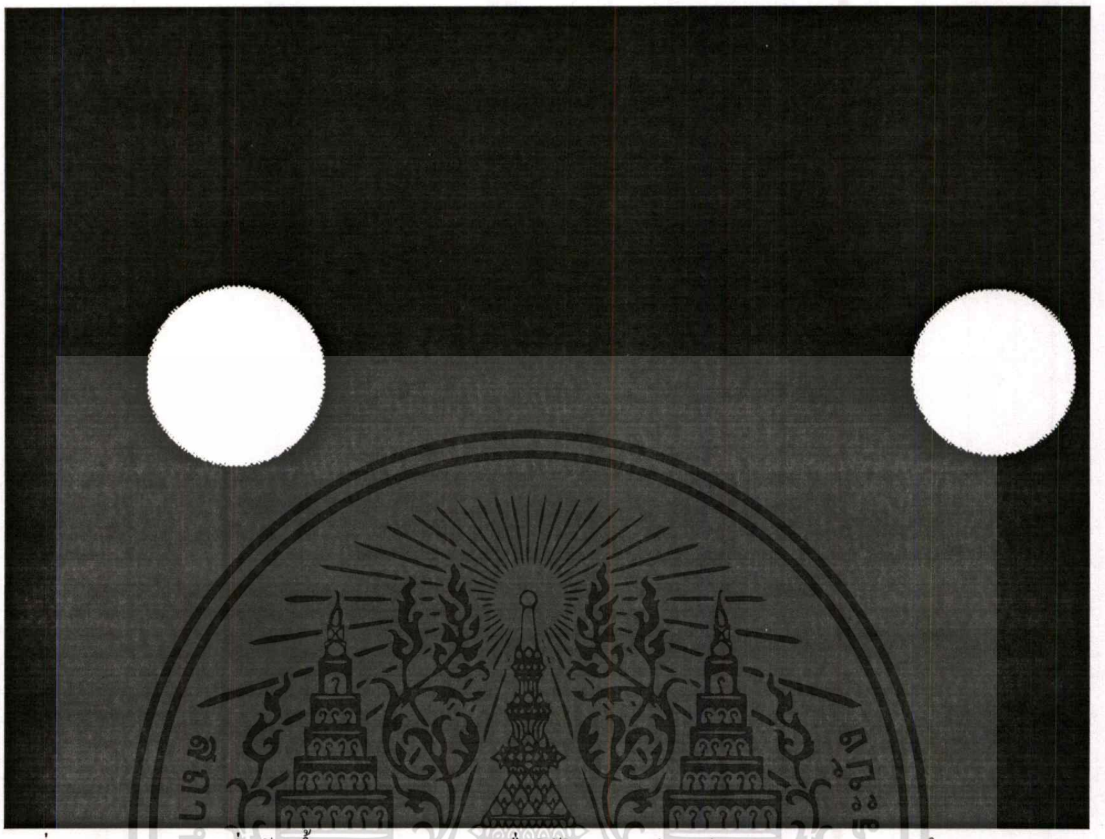
5.21 รูปที่ได้จากการถ่ายด้วยกล้องดิจิตอลมีคุณภาพสูง ทำให้เกิด noise ในปริมาณต่ำซึ่งสามารถทำการกำจัดได้โดยวิธีการ image processing

5.22 รูปที่ทำการถ่ายมาได้จะต้องมีอย่างน้อยส่วนหนึ่งเป็นขอบที่ไม่ได้เป็นส่วนที่เกิดเงา(ขอบส่วนที่คมชัด)

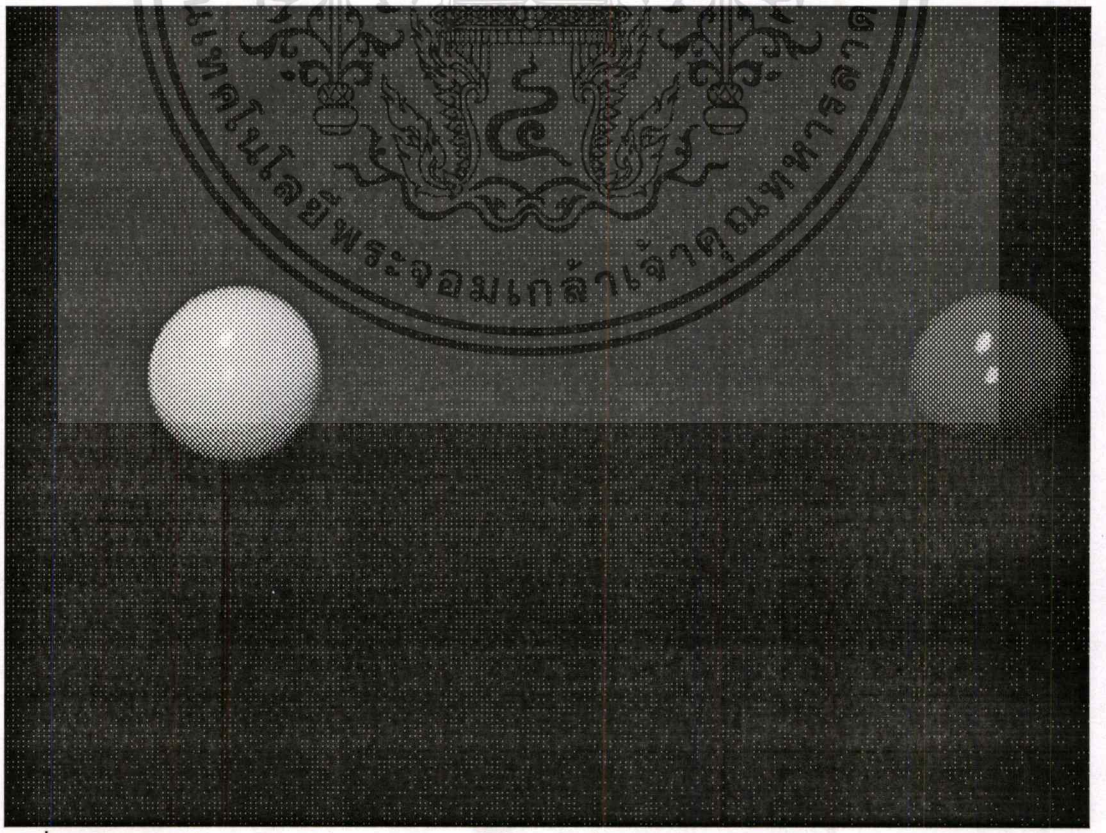
5.23 รูปที่ทำการหาวัตถุโดยโปรแกรม photoshop สามรถเชื่อถือได้ในระดับหนึ่ง

**5.3 ขั้นตอนการทดลอง**

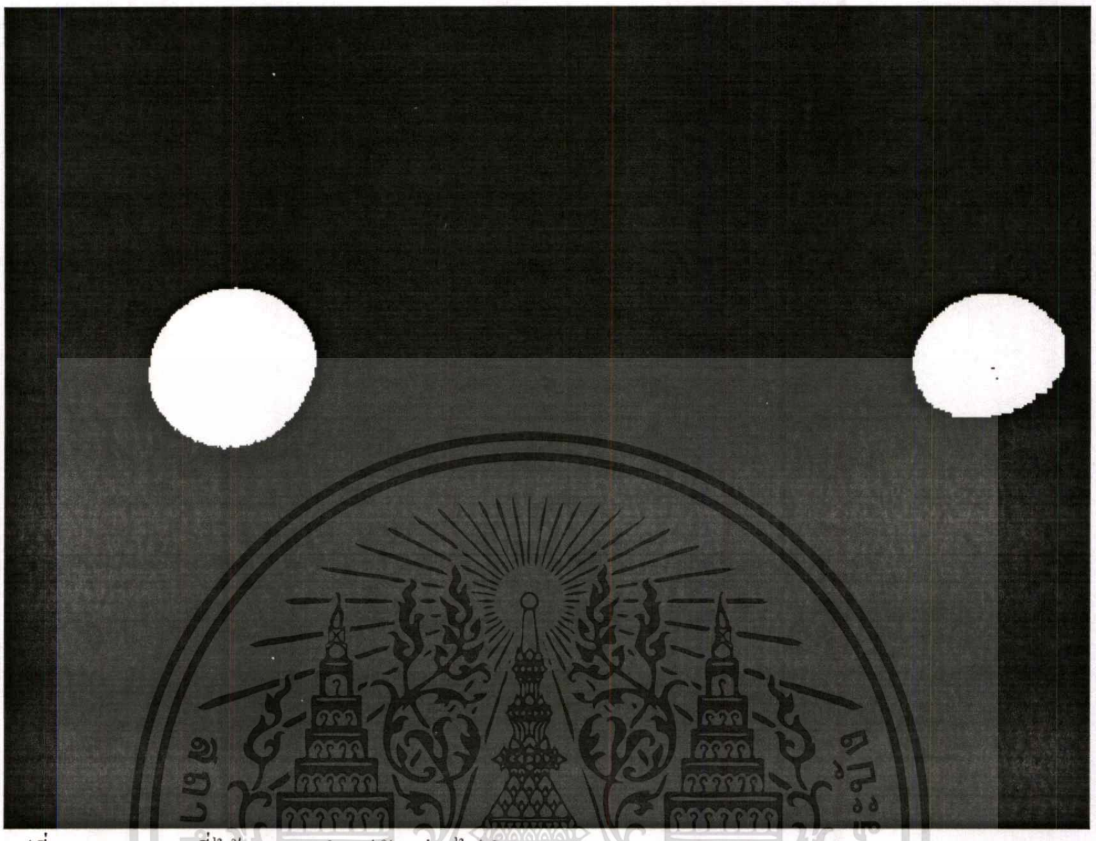
การทดลองจะมีขั้นตอนการทำงานคือ นำรูปภาพที่ทำการถ่ายจากวัตถุจริงดังรูปที่ xxx ไปผ่านขั้นตอนต่างๆของอัลกอลิทึมต่างๆ แล้วทำการเปรียบเทียบจุดศูนย์กลางที่ได้จากอัลกอลิทึมกับจุดศูนย์กลางที่ได้จากการหาจุดศูนย์กลางมวลจากรูปที่ทำการ select area โดย photoshop จากรูปจริง



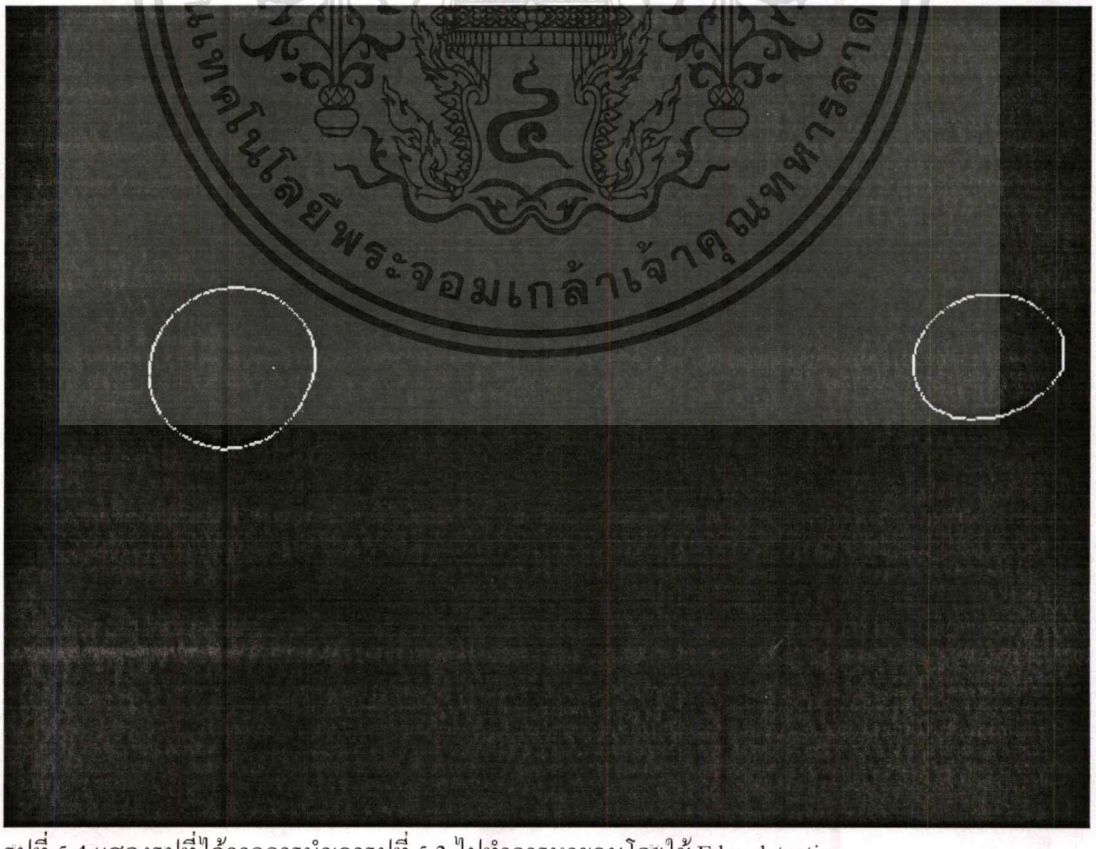
รูปที่ 5.1 แสดงรูปภาพที่สร้างขึ้นจากม photoshop เพื่อใช้ในการเปรียบเทียบกับการทำอัลกอลิทึม



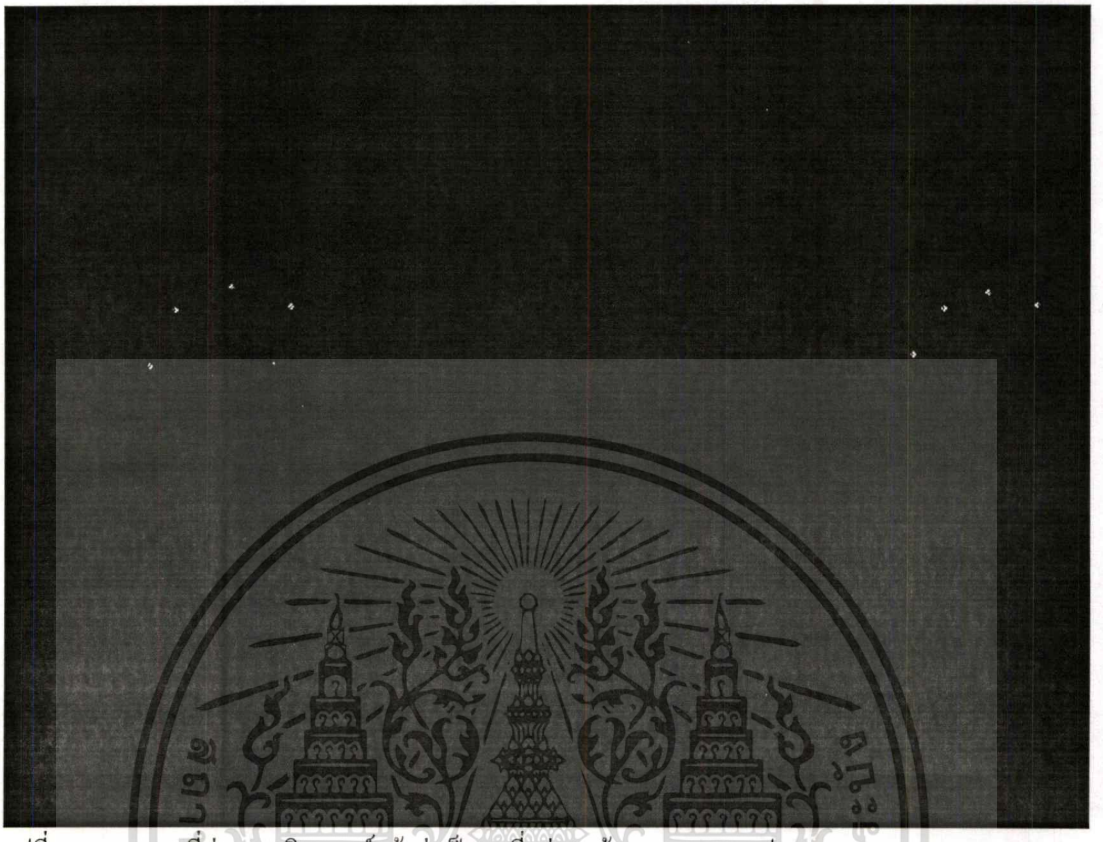
รูปที่ 5.2 แสดงรูปต้นฉบับก่อนเข้าทำการทดลองในอัลกอลิทึม



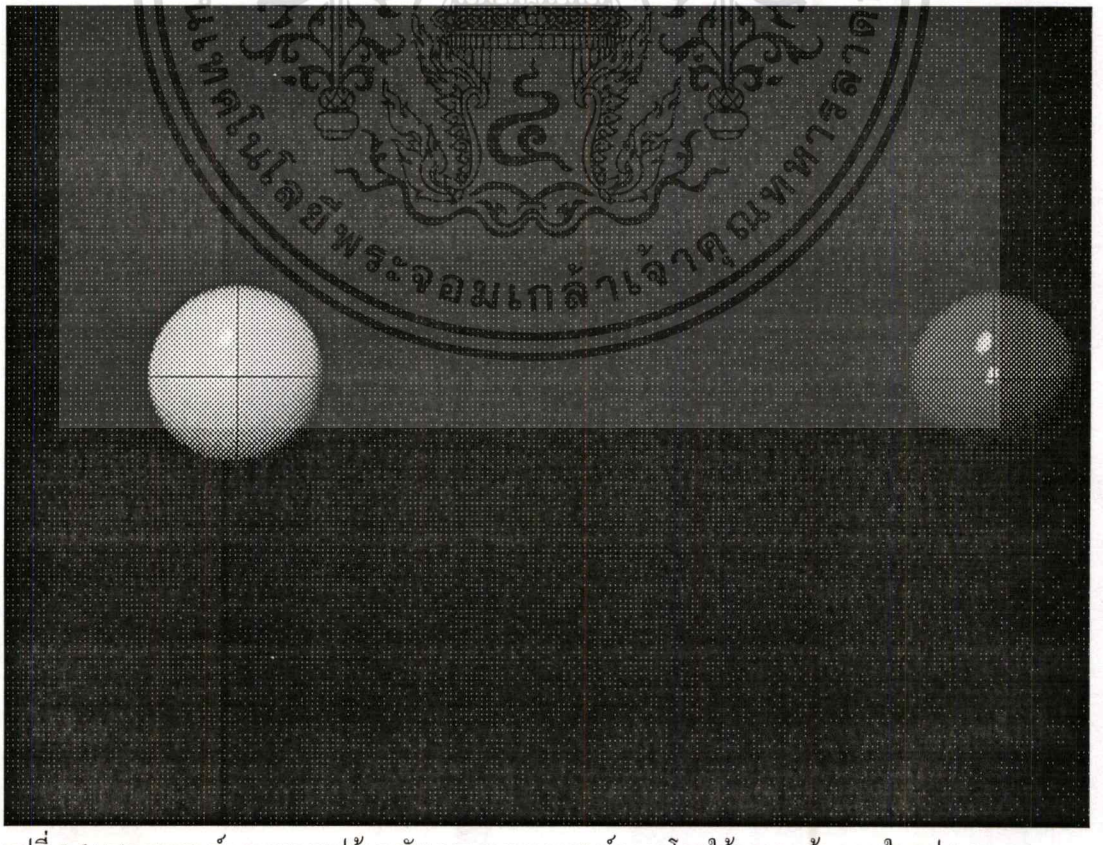
รูปที่ 5.3 แสดงภาพที่ได้จากการนำรูปตัวอย่างไปทำ Color segmentation



รูปที่ 5.4 แสดงรูปที่ได้จากการนำเอารูปที่ 5.3 ไปทำการหาขอบโดยใช้ Edge detection



รูปที่ 5.5. แสดงจุดที่ผ่านการวิเคราะห์แล้วว่าเป็นจุดที่อยู่บนเส้นขอบคมของรูปภาพทรงกลม



รูปที่ 5.6 แสดงจุดศูนย์กลางของรูปต้นฉบับ จากการหาจุดศูนย์กลางโดยใช้จุดบนเส้นขอบในรูป 5.5

5.4 ผลจากการทดลองด้วยอัลกอลิทึม

ผลที่ได้จากการทดลองด้วยอัลกอลิทึมนั้นจะผลออกเป็น 2 ส่วนคือ ส่วนที่ทำการหาตำแหน่งของวัตถุและอีกส่วนคือส่วนที่ทำการหาตำแหน่งจุดศูนย์กลางของวัตถุ โดยเทียบจากรูปที่ทำการประมาณตำแหน่งของวัตถุด้วยโปรแกรม photoshop จากรูปถ่ายที่นำมาใช้ทดลอง แล้วนำไปทำการเปรียบเทียบกันกับผลที่ได้จากการทดลอง

5.4.1 การเปรียบเทียบการหาตำแหน่งของวัตถุ

จะเทียบจากค่าพื้นที่ของวัตถุที่ทำการหาจากการทำ color segmentation ของอัลกอลิทึมกับขนาดของวัตถุที่ทำการ select จากการใช้โปรแกรม photoshop โดยที่จะผลที่ได้จะแบ่งเป็นค่าต่างๆ ดังนี้

1. area from algorithm คือค่าพื้นที่วัตถุที่หาได้จากอัลกอลิทึม color segmentation ที่ทำการพัฒนา
2. area from PS คือค่าพื้นที่ของวัตถุที่หาได้จากการทำการ select พื้นที่ของวัตถุจากโปรแกรม photoshop
3. over detection คือค่าพื้นที่ที่อัลกอลิทึมนั้นตรวจเจอ แต่ไม่ใช่เป็นพื้นที่ของวัตถุจริงๆ เช่น ในกรณีที่เกิดแสงกระทบของวัตถุหนึ่งไปยังอีกวัตถุหนึ่งเป็นต้น
4. non detection คือค่าของพื้นที่ที่อัลกอลิทึมนั้นไม่สามารถตรวจเจอ แต่มีพื้นที่นั้นในวัตถุจริงๆ เช่นในพื้นที่ที่ถูกกลืนจากการเกิดเงาบนวัตถุ
5. ค่า error คือค่าผิดพลาดที่เกิดขึ้นจากการทำการ detect ด้วย photoshop ซึ่งอาจจะทำการ detect เกินหรือ detect ขาดก็ได้

**ตารางที่ 5.1 แสดงผลการทดลองอัลกอลิทึมการ segmentation กับรูปภาพตัวอย่างในส่วนของวัตถุสีแดง**

| | area red | | | | |
|---|---|---|---|---|---|
| picture | algorithm | PS | over detect | non detect | error |
| | | | | | |
| 1 | 1941 | 3228 | 0 | 1287 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 2828 | 3852 | 0 | 1024 | 0 |
| 4 | 1808 | 3268 | 0 | 1460 | 0 |
| 5 | 1979 | 3313 | 0 | 1334 | 0 |
| 6 | 2138 | 3313 | 1 | 1176 | -2 |
| 7 | 1820 | 3472 | 3 | 1655 | -6 |
| 8 | 1462 | 5681 | 0 | 4219 | 0 |
| 9 | 2447 | 5808 | 584 | 3945 | -1168 |
| 10 | 3110 | 6221 | 0 | 3111 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 |
| 12 | 932 | 6005 | 0 | 5073 | 0 |
| 13 | 1363 | 5417 | 0 | 4054 | 0 |
| 14 | 1981 | 5481 | 0 | 3500 | 0 |
| 15 | 2918 | 7161 | 475 | 4718 | -950 |
| 16 | 3650 | 6948 | 0 | 3298 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 |
| 18 | 2026 | 6509 | 0 | 4483 | 0 |
| 19 | 1759 | 7604 | 0 | 5845 | 0 |

**ตารางที่ 5.2 แสดงผลการทดลองอัลกอลิทึมการ segmentation กับรูปภาพตัวอย่างในส่วนของวัตถุสีขาว**

| | area white | | | | |
|---|---|---|---|---|---|
| picture | algorithm | PS | over detect | non detect | error |
| | | | | | |
| 1 | 3494 | 3632 | 0 | 138 | 0 |
| 2 | 3374 | 3521 | 0 | 147 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 3790 | 3969 | 8 | 187 | -16 |
| 5 | 3125 | 3260 | 43 | 178 | -86 |
| 6 | 3270 | 3373 | 96 | 199 | -192 |
| 7 | 3337 | 3521 | 0 | 184 | 0 |
| 8 | 5100 | 6309 | 0 | 1209 | 0 |
| 9 | 4234 | 5745 | 41 | 1552 | -82 |
| 10 | 0 | 0 | 0 | 0 | 0 |
| 11 | 5088 | 7192 | 0 | 2104 | 0 |
| 12 | 4020 | 5284 | 11 | 1275 | -22 |
| 13 | 4818 | 6573 | 0 | 1755 | 0 |
| 14 | 4579 | 5957 | 28 | 1406 | -56 |
| 15 | 4482 | 6785 | 151 | 2454 | -302 |
| 16 | 0 | 0 | 0 | 0 | 0 |
| 17 | 4787 | 7620 | 0 | 2833 | 0 |
| 18 | 5171 | 7620 | 0 | 2449 | 0 |
| 19 | 5096 | 7345 | 12 | 2261 | -24 |

5.4.2 การเปรียบเทียบการหาตำแหน่งของจุดศูนย์กลางของวัตถุ

จะเทียบจากค่าระยะทางระหว่างค่าจุดศูนย์กลางที่หาได้จากอัลกอลิทึมและจุดศูนย์กลางที่หาได้จากจุดศูนย์กลางมวลของวัตถุที่ทำการ select มาจาก photoshop โดยจะมีค่าต่างๆดังนี้

1. algorithm X คือค่าในแนวคอลัมน์ของจุดศูนย์กลางที่ได้จากการหาโดยอัลกอลิทึม
2. algorithm y คือค่าในแนวแถวของจุดศูนย์กลางที่ได้จากการหาโดยอัลกอลิทึม
3. algorithm X คือค่าในแนวคอลัมน์ของจุดศูนย์กลางที่ได้โดยการหาจากจุดศูนย์กลางมวลของวัตถุที่ทำการ select มาจาก photoshop
4. algorithm y คือค่าในแนวแถวของจุดศูนย์กลางที่ได้ได้โดยการหาจากจุดศูนย์กลางมวลของวัตถุที่ทำการ select มาจาก photoshop

**ตารางที่ 5.3 แสดงผลการทดลองอัลกอลิทึมหาจุดศูนย์กลางกับรูปภาพตัวอย่างในส่วนของวัตถุสีแดง**

| | red center | | | | | |
|---|---|---|---|---|---|---|
| picture | algorithm | | PS | | radius | error |
| | x | y | x | y | | |
| 1 | 368 | 203 | 369 | 205 | 32.0815 | 2.24 |
| 2 | 0 | 0 | 0 | 0 | 0.0000 | 0.00 |
| 3 | 319 | 206 | 322 | 207 | 35.0391 | 3.16 |
| 4 | 431 | 198 | 433 | 200 | 32.2772 | 2.83 |
| 5 | 518 | 273 | 520 | 274 | 32.5013 | 2.24 |
| 6 | 370 | 225 | 371 | 226 | 32.5013 | 1.41 |
| 7 | 412 | 245 | 414 | 247 | 32.2591 | 2.83 |
| 8 | 320 | 154 | 317 | 155 | 42.5484 | 3.16 |
| 9 | 388 | 227 | 390 | 234 | 40.0292 | 7.28 |
| 10 | 346 | 219 | 346 | 217 | 44.5298 | 2.00 |
| 11 | 0 | 0 | 0 | 0 | 0.0000 | 0.00 |
| 12 | 512 | 266 | 536 | 277 | 43.7466 | 26.40 |
| 13 | 195 | 177 | 197 | 180 | 41.5481 | 3.61 |
| 14 | 352 | 212 | 349 | 205 | 41.7820 | 7.62 |
| 15 | 347 | 201 | 365 | 235 | 47.7610 | 38.47 |
| 16 | 356 | 212 | 356 | 214 | 47.0549 | 2.00 |
| 17 | 0 | 0 | 0 | 0 | 0.0000 | 0.00 |
| 18 | 242 | 182 | 241 | 193 | 45.5327 | 11.05 |
| 19 | 525 | 254 | 526 | 258 | 49.2241 | 4.12 |

**ตารางที่ 5.4 แสดงผลการทดลองอัลกอลิทึมหาจุดศูนย์กลางกับรูปภาพตัวอย่างในส่วนของวัตถุสีขาว**

| | white center | | | | | |
|---|---|---|---|---|---|---|
| picture | algorithm | | PS | | radius | error |
| | x | y | x | y | | |
| 1 | 366 | 245 | 366 | 246 | 34.0444 | 1.00 |
| 2 | 423 | 238 | 423 | 239 | 33.4882 | 1.00 |
| 3 | 0 | 0 | 0 | 0 | 0.0000 | 0.00 |
| 4 | 235 | 285 | 235 | 285 | 35.5595 | 0.00 |
| 5 | 225 | 205 | 226 | 206 | 32.2359 | 1.41 |
| 6 | 434 | 225 | 434 | 226 | 32.7784 | 1.00 |
| 7 | 349 | 244 | 349 | 245 | 33.4882 | 1.00 |
| 8 | 388 | 182 | 390 | 184 | 44.8317 | 2.83 |
| 9 | 321 | 197 | 322 | 198 | 42.7791 | 1.41 |
| 10 | 0 | 0 | 0 | 0 | 0.0000 | 0.00 |
| 11 | 348 | 217 | 349 | 218 | 47.8789 | 1.41 |
| 12 | 185 | 178 | 184 | 178 | 41.0506 | 1.00 |
| 13 | 542 | 291 | 541 | 291 | 45.7665 | 1.00 |
| 14 | 403 | 248 | 402 | 248 | 43.5683 | 1.00 |
| 15 | 317 | 185 | 319 | 180 | 46.4962 | 5.39 |
| 16 | 0 | 0 | 0 | 0 | 0.0000 | 0.00 |
| 17 | 337 | 200 | 338 | 199 | 49.2763 | 1.41 |
| 18 | 509 | 267 | 509 | 269 | 49.2763 | 2.00 |
| 19 | 204 | 172 | 204 | 173 | 48.3753 | 1.00 |

## 5.5 บทสรุป

จากผลที่ได้จากการทดลองจะเห็นได้ว่าอัลกอลิทึมที่ทำการพัฒนาขึ้นมานั้นไม่มีความจำเป็นที่จะต้องทำการ segmentation ได้ตรงกับพื้นที่ทั้งหมดของวัตถุก็สามารถที่จะตรวจสอบจุดศูนย์กลางของวัตถุทรงกลมได้ หากสามารถทำการหาขอบที่แท้จริงของวัตถุออกมาได้

ส่วนค่า error ที่เกิดขึ้นจากอัลกอลิทึมการหาจุดศูนย์กลางของวัตถุทรงกลมนั้นเกิดขึ้นเนื่องจากค่าที่ได้เป็นค่า pixel รายละเอียดจึงไม่สูงเท่าใดนัก

5.5.1. อัลกอลิทึมที่ใช้จริงในการทดลอง จะประกอบไปด้วย

- อัลกอลิทึมการทำ ColorSegmentation
- อัลกอลิทึมการหา Edge โดย EdgeDetection
- อัลกอลิทึมการเลือกตำแหน่งบนเส้นขอบที่แท้จริงมาเพื่อใช้ในการหาจุดศูนย์กลาง (getSide) ซึ่งภายในอัลกอลิทึมนี้จะมีส่วนของอัลกอลิทึมย่อยสำหรับหาค่า Error ที่ชื่อว่า StableValue รวมอยู่ด้วย
- อัลกอลิทึมการหาจุดศูนย์กลางวงกลมจากจุดบนเส้นขอบที่แท้จริงของวัตถุ

5.5.2. ข้อดีของอัลกอลิทึม

สามารถหาจุดศูนย์กลางของรูปวงกลมภายในภาพได้อย่างค่อนค่างถูกต้องใกล้เคียงกับค่าที่ประมาณด้วยสายตา ไม่ว่าจะเกิดเงาขึ้นหรือไม่ก็ตาม

5.5.3. ข้อเสียของอัลกอลิทึม

เป็นที่ทราบกันดีว่าการทำ Image Processing นั้นค่อนข้างจะใช้งานทรัพยากรสูงอีกทั้งค่อนข้างที่จะกินเวลาในการประมวลผลสูง เพื่อที่จะให้อัลกอลิทึมมีความแม่นยำในการหาจุดศูนย์กลาง อีกทั้งยังป้องกันความเสี่ยงเนื่องจากความผิดพลาดต่างๆเพื่อให้อัลกอลิทึมสามารถทำงานต่อไปได้ จึงต้องมีการเพิ่มความซับซ้อนให้อัลกอลิทึม ทำให้เกิด Overhead สูงขึ้น อีกทั้งอัลกอลิทึมมีจุดบอดอยู่ตรงที่ หากไม่ได้ตำแหน่งบนเส้นขอบของวัตถุที่ถูกต้องแล้ว การหาจุดศูนย์กลางก็จะผิดพลาดไปด้วย

5.5.4. ข้อเสนอแนะ

ปรับปรุงเพิ่มเติมในส่วนของการทำ Segmentation และ Threshold แล้วทำการทดสอบเพิ่มเติมในสภาวะอื่นๆเพื่อทำการหาจุดที่ต้องปรับปรุงของอัลกอลิทึม

# บรรณานุกรม

Edward, B. 1993. **Finding a Sphere From Four Points.** [online]. Available : http://www.research.microsoft.com/~hollasch/cgindex/geometry/sphere4pts.html

Gregory, A. B. 1994. **Digital Image Processing Principles And Applications USA**:John Wiley & Sons

John, C. R. 1999. **The Image Processing Handbook Third Edition London**: CRC Press LLC

Scott, G. and Owen, G. 1999 **A Practical Approach to Motion Capture: Acclaim's optical motion capture system.** [online]. Available : http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/motion_capture/motion_optical.htm.

Stefan, S. 1994. **Device Technology for Motion Capture and Applications in Interactive Environments.** [online]. Available :http://www.syscon.uu.se/Personnel/vs/Lecture4.pdf

Tae-Eun, K. et al. 1997. **Shape Recovery of Hybrid Reflectance Surface using Neural Network.** [online]. Available : http://www.computer.org/icip/8183/volumn3/81830416abs.htm

# ประวัติผู้เขียน

**ชื่อ นายเทพศิริ อสัมภินพงศ์**

**เกิดเมื่อวันที่ 27 กรกฎาคม 2522 ที่โรงพยาบาลคริสเตียน ถนนสีลม จังหวัดกรุงเทพมหานคร**

**ประวัติการศึกษา**

**ศึกษาระดับมัธยม 1-6 จากโรงเรียนเตรียมอุดมศึกษาพัฒนาการสายวิทย์-คณิต**

**จบการศึกษาปริญญาตรีวิทยาศาสตรบัณฑิตจาก สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง คณะวิทยาศาสตร์ สาขาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ แขนงวิชาวิทยาการคอมพิวเตอร์**

**ปัจจุบันกำลังศึกษาอยู่ ณ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังคณะเทคโนโลยีสารสนเทศ สาขาเทคโนโลยีสารสนเทศ**

# ภาคผนวก

**Main()**

Input Picture[]

Object[] = ColorSegment(Picture[])

Edge[] = EdgeDetection(Object[])

rightEdge[] = getSide(edge[],Picture)

(x,y) = findcenter(rightedge[])

คำอธิบาย

อัลกอลิทึมข้างต้นเป็นอัลกอลิทึมหลักสำหรับการตรวจหาจุดศูนย์กลางของรูปวัตถุทรงกลม โดยจะรวมเอาแต่ละอัลกอลิทึมย่อยๆเข้าด้วยกัน ซึ่งจะประกอบด้วยอัลกอลิทึมต่างๆดังนี้

*Input Picture[] :*

อ่านรูปภาพจากไฟล์รูปให้มาอยู่ในรูป matrix Picture[] ซึ่งมีขนาด 3 มิติขนาด กว้าง x ยาว x 3 ซึ่ง 3 นั้นคือค่าความสว่างของแต่ละสี red green blue

*Object[] = ColorSegment(Picture[]) :*

นำรูปที่อ่านได้มาทำการแยกวัตถุออกจากพื้นหลัง ผลลัพธ์ที่ได้จะเป็น matrix Object[] ขนาด กว้าง x ยาว x 2 ซึ่ง Object[กว้าง,ยาว,1] เป็นวัตถุส่วนสีแดง และ Object[กว้าง,ยาว,2] เป็นวัตถุส่วนสีขาว

ภายใน matrix จะมีค่าเพียง 1 และ 0 ซึ่งค่า 1 หมายความว่า pixel ในส่วนนั้นในรูปจริงเป็นวัตถุ และ 0 หมายความถึง pixel ในรูปจริงเป็นพื้นหลัง

*Edge[] = EdgeDetection(Object[])*

จะทำการแยกหาขอบของวัตถุแต่ละสีที่ทำการ segment มาได้

rightEdge[] = getSide(edge[],Picture)

จะทำการนำเอาขอบที่หามาได้ไปเทียบกับรูปจริงเพื่อการแยกว่าขอบด้านใดเป็นขอบในด้านที่คมชัด

(x,y) = findcenter(rightedge[])

จะนำเอาขอบในส่วนที่คมชัดไปทำการหาจุดศูนย์กลาง

**Segment(Picture[])**

!-- Make to grayscale picture

if Picture is not grayscale

grayPic[] = rgb2gray(Picture[])

else

grayPic[] = Picture[]

!-- get threshold level

level = graythresh(grayPic[])

!-- get object area from picture

object[] = im2bw(grayPic[],level)

!-- im2bw is algorithm that consider the blightness of picture if blightness above level count as 1 else count as 0

return object[]

คำอธิบาย

เป็นอัลกอลิทึมแรกที่นำมาใช้ในการทำ segmentation แต่อัลกอลิทึมนี้สามารถทำงานได้เพียง segment วัตถุชิ้นเดียวออกจากภาพ ซึ่งอัลกอลิทึมมีหลักการทำงานดังนี้

ทำการเช็คว่ารูปที่อ่านเข้ามาเป็นสีหรือเป็นขาวเทา ถ้าเป็นสีให้ทำการเปลี่ยนเป็นขาวเทาก่อน โดยคำสั่ง rgb2gray(Picture[])

เมื่อเป็นรูปขาวดำแล้วก็ทำการหาค่า threshold โดยเก็บไว้ในตัวแปร level

นำรูปขาวเทาที่ได้ไปทำการตรวจสอบแต่ละจุดว่าจุดใดมาค่ามากกว่าหรือน้อยกว่าค่า threshold ที่หามาได้โดยที่จุดที่มีค่ามากกว่าค่า threshold ให้เป็น 1 และน้อยกว่าให้มีค่าเป็น 0 (ซึ่งการทำงานเหล่านี้จะเป็นฟังก์ชันที่มีอยู่แล้วใน mathlab ชื่อว่า im2bw() )

**EdgeDetection(object[])**

```
!-- get edge by usesing edge command
P_edge[] = edge(object[])
!-- in this case we use canny algorithm
!-- P_edge Contain coordinate (x,y) of each spot on edge
return P_edge[]
```

คำอธิบาย

เป็นอัลกอลิทึมทำการหาค่าขอบของวัตถุโดยค่าที่ทำการ return ออกมานั้นเป็นค่าจุด(x,y)ทุกจุดที่เป็นขอบของวัตถุ(ซึ่งฟังก์ชันนี้มีอยู่แล้วใน mathlab ชื่อว่า edge() )

**getSide(P_edge[])**

```
!-- get Right edge
for each value in P_edge[]
        find slope on each spot
mean = sum(absolute(slope[]))/numberof(slope[])
j = 1
for each value in P_edge[]
        if slope at P_edge[i] > mean
                UseEdge[j].x = P_edge[i].x
                UseEdge[j].y = P_edge[i].x
                j++
return UseEdge[]
```

คำอธิบาย

อัลกอลิทึม getSide() นั้นเป็นอัลกอลิทึมเพื่อใช้แยกแยะขอบที่แท้จริงของวัตถุออกจากขอบทั้งหมดที่หามาได้โดยจะมีการทำงานดังนี้

หาจุดอ้างอิงจากตำแหน่งกลางของวัตถุโดยหาได้จาก center of mass ของวัตถุที่ได้จากการทำ segment หรือ colorsegment

มองเป็นเส้นตรงที่ลากจากจุดอ้างอิงไปตัดยังเส้นขอบทั้ง หาความชันของความสว่างของแสงที่อยู่ทั้ง 8 ตำแหน่งมาหาค่าเฉลี่ย

ทำการแยกค่าที่มีค่าความชันต่ำกว่าค่าเฉลี่ยออกแล้วนำค่าที่มีค่าความชันสูงกว่าเพื่อไปใช้ในการหาจุดศูนย์กลางต่อไป

**findcenter(UseEdge[])**

```
n = size of UseEdge[]
        for i = 1 to n
                if i < n
                        a[i,1],a[i,2],b[i] =
findEquation(UseEdge[i].x,UseEdge[i].y,UseEdge[i+1].x,UseEdge[i+1].y);
                else
                        a[i,1],a[i,2],b[i] =
findEquation(UseEdge[i].x,UseEdge[i].y,UseEdge[1].x,UseEdge[1].y);
!-- findEquation return a,b,c of equation ax+by = c
!-- find pseudo inverse of a
pa[] = pinv(a[])
(x,y) = pa[]*b[]
return (x,y)
```

คำอธิบาย

เป็นอัลกอลิทึมที่ใช้ในการแก้สมการจากทฤษฎีการหาจุดศูนย์กลางจากจุดตัดของเส้นตั้งฉากกับจุดกึ่งกลางของเส้นที่ลากเชื่อมต่อกันระหว่างจุดบนเส้นขอบที่แท้จริง โดยนำเอาค่าสัมประสิทธิ์ของสมการเส้นตรงที่ตั้งฉากนั้นมาใช้คิด โดยสัมประสิทธิ์หาได้จากอัลกอลิทึม findEquation ซึ่งจะ return ค่าออกมาเป็น a b และ c

ส่วนคำสั่ง pinv นั้นคือคำสั่งที่ใช้ในการหา pseudo inverse ของ matrix ดังที่กล่าวไว้ในส่วนของทฤษฎี

```
findEquation(sx1,sy1,sx2,sy2)
if(sx1<sx2)
   x1=sx1
   y1=sy1
   x2=sx2
   y2=sy2
else
   x1=sx2
   y1=sy2
   x2=sx1
   y2=sy1
end
cx = (x1+x2)/2
cy = (y1+y2)/2
a = x2-x1
b = y2-y1
c = (cx*(x2-x1))+(cy*(y2-y1))
return a,b,c
```

คำอธิบาย

เป็นอัลกอลิทึมที่ใช้หาค่าสัมประสิทธิ์ของสมการเส้นตั้งจากที่ลากกึ่งกลางระหว่าสมการที่ลากผ่านจุด (sx1,xy1) และ (sx2,xy2)

**ColorSegment(Picture[])**

```
!-- picture contain with red green blue
level = graythresh(Picture[].red)
ObjRed[] = im2bw(Picture[].red,level)
level = graythresh(Picture[].green)
ObjGreen[] = im2bw(Picture[].green,level)
level = graythresh(Picture[].blue)
ObjcBlue[] = im2bw(Picture[].blue,level)
Object[] = ObjRed[]-ObjGreen[]-ObjBlue[]
for each spot in Object[i,j]
        if Object[i,j] < 0
                Object[i,j] = 0
ObjectRed[] = Object[]
ObjectWhite[] = ObjRed[]&ObjGreen[]&ObjBlue[]
return ObjectRed[],ObjectWhite[]
```

คำอธิบาย

Selectcolor เป็นการทำ Color Segmentation แบบที่คาดว่าในสีแดงจะต้องไม่มีสีเขียวและสีน้ำเงินปนอยู่มาก และสีขาวคือสีที่มีทั้งสีแดง สีเขียว และสีน้ำเงินอยู่ในปริมาณที่เท่าๆกันซึ่งมีหลักการทำงานดังนี้

รับรูปสีเข้ามา ภายในรูปสีจะประกอบด้วย ตารางสีแดง สีเขียวและสีน้ำเงิน

ทำการแยกแต่ละตารางออกมาจะได้เป็นรูปภาพขาวดำของแต่ละสี

นำภาพขาวดำของแต่ละสีไปทำการ segment โดยการหาค่า threshold แล้วนำไปเข้าฟังก์ชัน im2bw

เมื่อได้วัตถุตามตารางสีออกมาแล้วทำการหาสีที่ต้องการได้โดย

สีแดง = วัตถุในส่วนของ(สีแดง - สีเขียว – สีน้ำเงิน)

สีเขียว = วัตถุในส่วนของ(สีแดง - สีเขียว – สีน้ำเงิน)

สีน้ำเงิน = วัตถุในส่วนของ(สีแดง - สีเขียว – สีน้ำเงิน)

สีขาว = วัตถุในส่วนของ(สีแดง & สีเขียว & สีฟ้า)

เมื่อได้ตามสีที่ต้องการแล้วก็ทำการ return ชุดข้อมูลที่ได้กลับไป

```
selectcolor2(Picture[])
xgray[]= rgb2gray(Picture[])
level = graythresh(xgray[])
level = level*255
xnewred[] = zeroes matrix size =Picture[]
xnewwhite[] = zeroes matrix size =Picture[]
r = max row of Picture
c = max column of Picture
for i = 1 to r
   for k = 1 to c
      if xgray[i,k] > level,
         ratio_red = (x[i,k].red/3)/xgray[i,k];
         ratio_green = (x[i,k].green/3)/xgray[i,k];
         ratio_blue = (x[i,k].blue/3)/xgray[i,k];
         if ratio_red > 0.40
                  xnewred[i,k] = 1;
         else
           if (ratio_red < 0.40) & (ratio_green < 0.40) &(ratio_blue < 0.40)
                  xnewwhite[i,k] = 1
return xnewred[],xnewwhite[]
```

คำอธิบาย

selectcolor2 เป็นอัลกอลิทึมที่ได้ปรับปรุงเพิ่มเติมจาก color segmentation เนื่องจากมีบางกรณีที่ไม่สามารถ Detect วัตถุสีแดงได้ในส่วนที่เป็นขอบคมเพราะมีแสงตกกระทบมากเกินไป

selectcolor2 นั้นอยู่ภายใต้แนวความคิดที่ว่า วัตถุที่มีสีแดงจะมีอัตราส่วนระหว่างสีแดง:สีเขียว:น้ำเงินในอัตราส่วน 4:3:3 ในขณะที่วัตถุสีขาวจะมีอัตราส่วนระหว่าสีเป็น 3.3 : 3.3 : 3.3

selectcolor2 มีหลักการทำงานดังนี้

ทำการอ่านภาพสีเข้ามา

ทำการแปลงรูปสีให้เป็นรูปขาวดำ และทำการแยกรูปสีออกเป็น 3 สีคือ แดง เขียว และน้ำเงิน

หาอัตราส่วนระหว่างสีต่อความสว่างทั้งหมดโดย

อัตราส่วนของสี = (ค่าความสว่างของสีในจุดนั้น/ค่าความสว่างของรูปขาวดำในจุดนั้น)/3

เมื่อค่าในอัตราส่วนของสีแดง > 40% หรือ 0.4 ให้ทำการบันทึกว่าตำแหน่งนั้นเป็นวัตถุสีแดง

เมื่อค่าในอัตราส่วนของทั้ง 3 สีมีค่า < 40% ให้ทำการบันทึกว่าตำแหน่งนั้นเป็นวัตถุสีขาว

**stableValue(Part_Edge)**

```
(x,y) = findcenter(Part_Edge[])
n = number of Part_Edge
mean_radius = 0
!--sum of all radius
for i = 1 to n
        radius[i] = length between (x,y) and spot in Part_Edge[i]
        mean_radius = mean_radius + radius[i]
mean_radius = mean_radius/n
s_value = 0
for i = 1 to n
        s_value = s_value + (abs(mean_radius - radius[i]))
s_value = s_value/n
return s_value
```

คำอธิบาย

อัลกอลิทึม stableValue นั้นไว้หาค่าความเรียบของเส้นโค้งว่ามีค่ามากกว่าที่กำหนดไว้หรือไม่ ถ้ามีมากกว่าแสดงว่าขอบที่อยู่ในค่าที่นำมาใช้ในการหา stableValue นั้นไม่ใช่ขอบคมของวัตถุ

อัลกอลิทึมจะถูกนำไปใช้ในส่วนของ getSide เพื่อทำการตัดค่าขอบที่ไม่เรียบทิ้งไปซึ่งมีหลักการทำงานดังนี้

รับช่วงขอบที่ทำการ detect เข้ามา

นำไปทำการหาค่าจุดศูนย์กลางของวงกลมจากช่วงขอบเหล่านั้น(จุดที่ได้ไม่ใช่จุดที่เป็นจุดศูนย์กลางของรูปวงกลมเสมอไปแต่เป็นจุดที่มีระยะห่างเฉลี่ยกับช่วงขอบทุกๆจุดเท่ากัน)

นำไปทำการหาระยะห่างระหว่างจุดที่หาได้กับจุดที่อยู่ในช่วงขอบที่ทำการรับเข้ามา

หาค่า mean

นำค่าระยะห่างที่ได้มาทำการลบออกจากค่า mean เพื่อทำการหาค่า error ของแต่ละจุด

ค่า stableValue = Sum error/จำนวนจุดขอบทั้งหมดที่รับเข้ามา