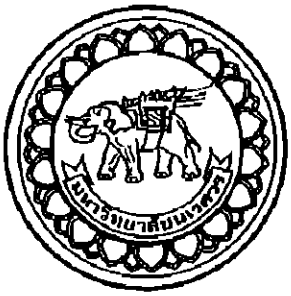


โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ

Circle Detection

นายปริญญา มาลาโรจน์ รหัส 46361978

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2549

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	**โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ**
ผู้ดำเนินโครงงาน	**นายปริญญา มาลาโรจน์**
อาจารย์ที่ปรึกษา	**ดร.พนมขวัญ ริชะมงคล**
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2549

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

..... ประธานกรรมการ
(ดร.พนมขวัญ ริชะมงคล)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น)

..... กรรมการ
(อาจารย์ศิริพร เดชะศิลารักษ์)

หัวข้อโครงงาน	โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ		
ผู้ดำเนินโครงงาน	นายปริญญา	มาลาโรจน์	รหัส 46361978
อาจารย์ที่ปรึกษา	ดร.พนมขวัญ	ริยะมงคล	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2549		

.....

บทคัดย่อ

การตรวจจับวัตถุที่เป็นวงกลมภายในภาพอาจจะนำไปใช้ในการตรวจสอบในโรงงานโดยใช้การตรวจสอบอัตโนมัติแทนการใช้คน ในงานอุตสาหกรรมหรือการแพทย์ จะช่วยประหยัดเวลาและประหยัดค่าใช้จ่าย การตรวจจับวัตถุที่เป็นวงกลมภายในภาพสามารถทำได้โดยแยกเป็น 2 ขั้นตอนหลักๆ คือ หาจุดศูนย์กลางของวงกลมและหาเส้นรอบวง โดยก่อนที่จะหาจุดศูนย์กลางได้เราต้องหาเส้นขอบของภาพ (Edge) จากนั้นก็ลากเส้นตั้งฉากจากจุดทุกจุดบนเส้นขอบนั้น บริเวณที่เส้นไปตัดกันมากที่สุดให้เป็นจุดศูนย์กลางของวงกลม จากนั้นก็นำจุดนั้นไปหาเส้นรอบวงโดยกำหนดช่วงความยาวของรัศมี การวาดวงกลมจะวาดอยู่ในช่วงรัศมีที่เป็นไปได้ ถ้าวาดแล้วเจอเส้นขอบของภาพก็จะถือว่ารัศมีนั้นคือรัศมีที่แท้จริง ผลลัพธ์ที่ได้จากโครงงานนี้ คือโปรแกรมตรวจจับวัตถุวงกลมภายในภาพที่สามารถตรวจจับวงกลมภายในภาพได้

Project Title	Circle Detection		
Name	Mr.Parinya	Malaroje	ID 46361978
Project Advisor	Dr.Panomkhawn	Riyamongkol	
Major	Computer Engineering.		
Department	Electrical and Computer Engineering.		
Academic Year	2006		

Abstract

Circle detection is one of the key problems in industrial vision applications such as automatic inspection. There is two-step of circle detection. We use the idea of Circle Hough Transform to find the circle(s) in an image. First, the centre of the circle(s) in the image will be found. Later, we get 'hot spot' that can predict the centre of circle. Then, the program enhances the 'hot spot' to find the real centre of circle and accumulate into 1 dimension value to find the radii. We draw the circumference between minimum radii to maximum radii to find the real radii. If the edge is found, the real circumference will be found and the circle in the image can be detected. The result of this project is the circle detection program that can detect the circle(s) in the image.

กิตติกรรมประกาศ

โครงงานวิศวกรรมคอมพิวเตอร์ สำเร็จได้ด้วยดีก็เนื่องจากความอนุเคราะห์จากท่านอาจารย์ที่ปรึกษาโครงงาน คือ อาจารย์พนมขวัญ ริยะมงคล ที่ให้ความกรุณาแนะนำวิธีในการทำงานให้เข้าใจถึงการศึกษาอย่างเป็นระบบขั้นตอน อีกทั้งสละเวลาเพื่อตรวจสอบการทำงานและ ชี้แนวทางแก้ไขในทุกขั้นตอนตลอดการทำโครงงาน และขอบคุณเพื่อนๆ ที่ไม่ได้เอ่ยนามที่คอยให้คำแนะนำต่างๆ

ปริญญา มาลาโรจน์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

Circle Detection หรือการตรวจจับวัตถุที่เป็นวงกลมภายในภาพอาจจะนำไปใช้ในการตรวจสอบในโรงงานโดยใช้การตรวจสอบอัตโนมัติแทนการใช้คน โดยจะตรวจสอบว่าวงกลมนั้นได้ขนาดหรือเป็นวัตถุวงกลมตามที่ต้องการหรือไม่

การตรวจจับวัตถุวงกลมภายในภาพนั้น เราต้องหาจุดศูนย์กลางของวงกลมให้ได้ก่อน แล้วจากนั้นจึงหารัศมีของวงกลม แต่ก่อนที่เราจะหาจุดศูนย์กลางวงกลมและรัศมีของวงกลมได้ เราต้องหา เส้นขอบของภาพ (edge) ซึ่งจะใช้การหาเส้นขอบภาพเข้ามาช่วย เมื่อเราได้เส้นขอบของภาพแล้วเราจึงจะสามารถนำไปคำนวณหาจุดศูนย์กลางและรัศมีของวงกลมได้

ประโยชน์ของการใช้การตรวจจับวัตถุวงกลมภายในภาพในงานอุตสาหกรรม คือ สามารถช่วยประหยัดเวลาในการทำงาน เพราะสามารถทำได้รวดเร็วกว่าการใช้คนในการตรวจสอบนอกจากนี้ยังช่วยประหยัดค่าใช้จ่ายในการจ้างพนักงานมาทำงานตรวจสอบนี้

1.2 วัตถุประสงค์โครงงาน

1. เพื่อศึกษาการใช้โปรแกรม MATLAB
2. เพื่อศึกษาการหาเส้นขอบภาพ
3. เพื่อศึกษาทฤษฎีการตรวจจับวัตถุที่เป็นวงกลมในภาพ โดยใช้ Hough transform
4. เพื่อให้ได้โปรแกรมตรวจจับวัตถุที่เป็นวงกลมในภาพ

1.3 ขอบเขตการทำงาน

1. เขียนโปรแกรมตรวจจับวัตถุวงกลมภายในภาพโดยใช้ MATLAB เป็นเครื่องมือ
2. ใช้ทฤษฎีการหาเส้นขอบของภาพมาช่วยในการหาเส้นขอบของภาพ
3. ใช้ทฤษฎี Hough transform ช่วยในการตรวจจับวงกลม

1.4 ขั้นตอนของการดำเนินงานโครงงาน

1. ศึกษาทฤษฎีของการหาเส้นขอบภาพ
2. ศึกษาทฤษฎีของ Hough transform เพื่อการตรวจหาวงกลมภายในภาพ
3. ศึกษาการใช้งาน MATLAB เบื้องต้น
4. นำทฤษฎีที่ได้ศึกษามาประยุกต์ใช้ในการเขียนโปรแกรมตรวจจับวัตถุที่เป็นวงกลมภายในภาพ
5. ติดตามผลการทำงาน
6. ทดสอบการทำงานของโปรแกรมและปรับปรุง
7. ตรวจทานและแก้ไขข้อผิดพลาดของโปรแกรม
8. จัดทำรูปเล่มรายงาน

1.5 แผนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงาน

รายการ	ปี 2549					ปี2550		
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1. ศึกษาทฤษฎีของการทำ Edge detection	←	—	→					
2. ศึกษาทฤษฎีของ Hough transform เพื่อการตรวจหาวงกลมภายในภาพ	←	—	→					
3. ศึกษาการใช้งาน MATLAB เบื้องต้น	←	—	→					
4. นำทฤษฎีที่ได้ศึกษามาประยุกต์ใช้ในการเขียนโปรแกรมตรวจจับวัตถุที่เป็นวงกลมภายในภาพ				←	→			
5. ติดตามผลการทำงาน				←	→			
6. ทดสอบการทำงานของโปรแกรมและปรับปรุง						←	→	
7. ตรวจทานและแก้ไขข้อผิดพลาดของโปรแกรม						←	→	
8. จัดทำรูปเล่มรายงาน		←	—	—	—	—	→	

1.6 ผลที่คาดว่าจะได้รับ

1. สามารถใช้โปรแกรม MATLAB ในการเขียนโปรแกรมตรวจจับวัตถุวงกลมภายในภาพได้
2. สามารถนำทฤษฎีการหาเส้นขอบภาพ และ Hough Transform ไปประยุกต์ใช้ในงานอื่นได้

1.7 รายละเอียดงบประมาณของโครงงาน

1. ค่าถ่ายเอกสาร	500 บาท
2. ค่าทำรูปเล่มรายงาน	500 บาท
รวมเป็นเงิน	1,000 บาท

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ภาพดิจิตอล (Digital Image) [1]

ภาพดิจิตอล คือ ภาพที่ได้จากการถ่ายด้วยกล้องดิจิตอล, การสแกนภาพโดยเครื่องแสกน จะแตกต่างจากภาพอะนาลอค ในหลายแง่มุมคือ ความละเอียด, สี, คุณภาพชิ้นงาน, ขั้นตอนการทำงาน และการจัดการ ตลอดจนถึงค่าใช้จ่าย ในธรรมชาติ แสงสีเป็นพลังงานคลื่นแม่เหล็กไฟฟ้าที่มีสมบัติของคลื่น และสมบัติเฉพาะทางไฟฟ้า และแม่เหล็กอยู่รวมกัน โดยที่มีความยาวคลื่นที่ตาของมนุษย์สามารถมองเห็นได้ในช่วง 380-780 นาโนเมตร ในช่วงความยาวคลื่นที่ตามองเห็นแตกต่างกันนี้คือแสงสีที่ต่างกันในสเปคตรัมของสี ดวงตามนุษย์สามารถมองเห็นสีที่แตกต่างกันได้จากที่ดวงตาของมนุษย์มีเซลล์รับแสงสีชนิดโคนที่ต่างกัน 3 ชนิดคือ โคนแดง โคนน้ำเงิน และโคนเขียว ที่ดูดแสงสีในช่วงคลื่น 570, 445, 535 นาโนเมตร การแปลผลสีอื่น ๆ ที่ต่างไปจากนี้จะเกิดขึ้นได้จากการผสมแสงสีทั้ง 3 ในอัตราส่วนต่าง ๆ กัน

2.1.1 รูปร่างของภาพ (Image Shape)

วัตถุที่มีอยู่ตามธรรมชาติและที่มนุษย์สร้างขึ้นมีรูปร่างที่แตกต่างกันไป ทั้งที่เป็นรูปทรงเรขาคณิตและไม่เป็นรูปทรงเรขาคณิตในศาสตร์ของการประมวลผลภาพนั้น การกำหนดขอบเขตของภาพทุกภาพให้อยู่ในรูปสี่เหลี่ยม (Rectangular image model) เป็นวิธีที่นิยมใช้กันมากที่สุด เนื่องจากทำให้การอ่านภาพ การจัดเก็บข้อมูลภาพในหน่วยความจำ และการแสดงภาพออกทางอุปกรณ์ต่าง ๆ เป็นไปได้อย่างมีประสิทธิภาพ

การเก็บข้อมูลภาพสามารถทำได้โดยการจองหน่วยความจำของเครื่องคอมพิวเตอร์ไว้ในรูปของตัวแปรอะเรย์ (Array) โดยค่าในแต่ละช่องของอะเรย์แสดงถึงคุณสมบัติของจุดภาพ (pixel) และตำแหน่งของช่องอะเรย์เป็นตัวกำหนดตำแหน่งของจุดภาพ (pixel)

2.1.2 พิกเซล (Pixel)

พิกเซล (Pixel) เป็นคำผสมของคำว่า Picture กับคำว่า Element หรือหน่วยพื้นฐานของภาพเทียบได้กับ "จุดภาพ" 1 จุด แต่ละพิกเซลเปรียบได้กับสี่เหลี่ยมเล็กๆ ที่บรรจุค่าสี โดยถูกกำหนดตำแหน่งไว้บนเส้นกริดของแนวแกน x และแกน y หรือในตารางเมตริกซ์สี่เหลี่ยม ภาพบิตแมปจะประกอบด้วยพิกเซลหลายๆ พิกเซล (pixel)

จำนวนพิกเซลของภาพแต่ละภาพ จะเรียกว่า ความละเอียด หรือ Resolution โดยจะเทียบจำนวนพิกเซลกับความยาวต่อนิ้ว ดังนั้นจะมีหน่วยเป็น พิกเซลต่อนิ้ว (ppi: pixels per inch)

หรือจุดต่อนิ้ว (dpi: dot per inch) ภาพขนาดเท่านั้นแต่มีความละเอียดต่างกัน แสดงว่าจำนวนพิกเซลต่างกัน และขนาดของจุดพิกเซลก็ต่างกันด้วย

2.1.3 ความละเอียดในการแสดงผล (Resolution)

คำนี้สามารถใช้ได้กับสถานการณ์ที่แตกต่างกัน เช่น ความละเอียดของการแสดงผลของเครื่องพิมพ์ หรือความละเอียดในการแสดงผลของจอภาพ ดังนั้นความละเอียดในการแสดงผลจึงหมายถึง จำนวนหน่วยต่อพื้นที่

ความละเอียดในการแสดงผล เป็นความสามารถในการปรับระยะการแสดงความละเอียดของภาพดิจิตอลระยะห่างของความถี่ในการแสดงภาพ (ความถี่ในการทำ sampling) จะถูกระบุในรูปของความละเอียดในการแสดงผล ซึ่งหมายถึง จุดต่อนิ้ว (dpi) หรือ พิกเซลต่อนิ้ว (ppi) เป็นคำทั่วไปที่ใช้เรียกหรือบ่งบอกว่ามีการแสดงภาพอยู่ที่ระดับความละเอียดในการแสดงผลที่เท่าไร แต่อยู่ในขอบเขตจำกัด การเพิ่มความถี่ในการแซมปลิ้งก็เป็นการเพิ่มความละเอียดในการแสดงผลด้วยเช่นกัน

สมมุติให้ภาพเป็นตัวแปรแบบอะเรย์ขนาด M x N (M แถว และ N คอลัมน์) ที่ใช้เก็บภาพขนาด M x N จุด (M จุดในแนวนอน และ N จุดในแนวตั้ง) ค่าสี (หรือความสว่าง ในกรณีที่เป็นภาพ grey level) ของจุดภาพในแถวที่ 5 คอลัมน์ที่ 4 จะตรงกับค่าของภาพ (5,4) จะเห็นว่าเราใช้ตำแหน่งของจุดภาพทั้งสองแกนเป็นตัวชี้ค่าข้อมูลในอะเรย์

จากการใช้หน่วยความจำเพื่อการเก็บภาพในลักษณะที่กล่าวมา เนื้อที่ที่ใช้ในการเก็บภาพสามารถคำนวณได้จาก M x N x g เมื่อ g เป็นจำนวนเต็มที่แทนจำนวนบิตของข้อมูลในแต่ละจุดภาพ ตัวอย่างถ้า g มีค่าเท่ากับ 8 บิต

เราจะสามารถเก็บความแตกต่างของระดับสีที่เป็นไปสูงสุด 256 ระดับ ค่า M และ N จะเป็นตัวบอกถึงความละเอียดของภาพ สำหรับคอมพิวเตอร์ทั่วไปในระบบ VGA (Video Graphic Array) จะมีขนาด 640x480, 800x600 และ 1024x768 จุด เป็นต้น การกำหนดความละเอียดจะขึ้นอยู่กับงานที่จะใช้ ในงานบางอย่างใช้ความละเอียดแค่ 30 x 50 จุด ก็พอแล้วแต่ในงานบางชนิด ใช้ความละเอียดถึง 1000 x 1000 จุด ก็ยังไม่พอ

2.1.4 บิต (BIT)

Bit ย่อมาจาก Binary Digit หมายถึง หน่วยความจำที่เล็กที่สุดของคอมพิวเตอร์ประกอบด้วยตัวเลข 2 จำนวน คือ 0 หมายถึงปิด และ 1 หมายถึงเปิด หรือสีขาวและสีดำ

ความลึกของบิต (Bit Depth) หมายถึง จำนวนบิตที่ใช้ในแต่ละพิกเซลในกราฟิกแบบบิต-แมปสีของพิกเซลถูกบันทึกโดยใช้บิต ถ้าใช้สีมากก็แสดงสีได้มากขึ้น ถ้ามีหน่วยความจำ 2 บิต ในการเก็บรวบรวมข้อมูล สามารถใช้สีได้ทั้งหมด 2 เท่ากับ 4 สี คือ สามารถกลับสีได้ 4 วิธี คือ 00, 01, 10 และ 11 ถ้ามี 2 บิต สามารถสร้างสีให้กับพิกเซลทั้งหมด 4 เฉดสี

จำนวนสีสูงสุดที่เป็นไปได้ของแต่ละจุดภาพขึ้นอยู่กับจำนวนบิตที่ใช้ เมื่อมีการกำหนดให้ขนาดของบิตต่อจุด มากขึ้นจะทำให้จำนวนของสีมากขึ้นด้วย ตัวอย่างเช่น 1 บิต = 2^1=2 สี, 2 บิต = 2^2=4 สี, 4 บิต = 2^4=16 สี, 8 บิต = 2^8=256 สี, 16 บิต = 2^{16}=65536 สี

ภาพขาว-ดำอยู่ในรูปพิกเซล ที่แต่ละพิกเซลจะมี 1 บิต ซึ่งแสดงได้ 2 ระดับสี คือ ขาวและดำ โดยค่า 0 จะเป็นสีดำ และ 1 จะเป็นสีขาว หรืออาจจะตรงกันข้าม

ภาพระดับเทา เป็นการเรียงของพิกเซลที่ใช้ข้อมูลแบบหลายบิต อยู่ในช่วงระหว่าง 2- 8 บิต หรือมากกว่านั้น

ภาพสี แบบทั่วไปนั้นจะมีค่า ความลึกบิต อยู่ในช่วง 8 – 24 บิต หรือมากกว่า ภาพที่มี 24 บิต นั้นคือ บิต จะถูกแบ่งออกเป็น 3 กลุ่ม 8 สำหรับสีแดง 8 สำหรับสีเขียว 8 สำหรับสีน้ำเงิน สีทั้งหมดจะถูกรวมกันเพื่อแสดงสีอื่นๆ ภาพ 24-บิต สามารถแสดงค่าสีได้ถึง 16.7 ล้านสี (2^{24}) สำหรับ เครื่องสแกน นั้นได้เพิ่ม จำนวน บิตในการจับภาพเอกสารเป็น 10 บิต หรือมากกว่านั้น เพราะบ่อยครั้งหลังจากจับภาพเอกสารที่ 8 บิต จะมี สัญญาณรบกวน รวมเข้าไปด้วย และเพื่อให้ภาพที่ออกนั้นมีความเหมือนใกล้เคียงกับที่มนุษย์ต้องการ อาจจะมีการใช้สีถึง 36, 48, 64 บิต แต่ในคอมพิวเตอร์ทั่วๆ ไปมักจะใช้สีไม่เกิน 32 บิต

2.2 รูปแบบของไฟล์ภาพ (Digital file format) [1] [2]

ภาพดิจิตอลที่จัดเก็บ (save) ในรูปแบบต่าง ๆ กันมีอยู่มากมายดังนี้ TIFF, BMP, GIF, JPEG, PSD ฯลฯ

2.2.1 ไฟล์สกุล GIF (CompuServe Graphics Interchange Format)

เป็นไฟล์ชนิดบิตแมปสามารถนำมาแสดงเป็นรูปกราฟิกได้ในทุกระบบและเก็บรายละเอียดสีได้ไม่เกิน 8 บิต มีความละเอียดของจุดสี (Pixel) สูงสุด 64,000 x 64,000 จุด ไฟล์ .GIF มีขนาดไม่ใหญ่มากนักแต่มีข้อจำกัดคือ สามารถแสงดสีได้สูงสุด 256 สีเท่านั้น เว็บเพจที่มีรูปเคลื่อนไหวต่าง ๆ จะมีนามสกุล .GIF โดยนำเอาภาพต่าง ๆ เป็นเฟรมมาผ่านโปรแกรมทำภาพเคลื่อนไหวเช่น GIF animation แล้วจะได้ภาพเคลื่อนไหวตามต้องการ

ไฟล์สกุล GIF เป็นไฟล์กราฟิกมาตรฐานที่ทำงานบนอินเทอร์เน็ต มักจะใช้เมื่อ

- ต้องการไฟล์ที่มีขนาดเล็ก
- จำนวนสีและความละเอียดของภาพไม่สูงมากนัก
- ต้องการพื้นแบบโปร่งใส
- ต้องการแสดงผลแบบโครงร่างก่อน แล้วค่อยแสดงผลแบบละเอียด
- ต้องการนำเสนอภาพแบบภาพเคลื่อนไหว

จุดเด่น

- มีขนาดไฟล์ต่ำ
- สามารถทำพื้นของภาพให้เป็นพื้นแบบโปร่งใสได้ (Transparent)
- มีระบบแสดงผลแบบหยาบและค่อยๆขยายไปสู่ละเอียดในระบบอินฟาเรด (Interlace)
- มีโปรแกรมสนับการสร้างจำนวนมาก
- เรียกดูได้กับตัวแสดงผลกราฟิคทุกตัว
- ความสามารถด้านการนำเสนอแบบภาพเคลื่อนไหว (Gif Animation)

จุดด้อย

- แสดงสีได้เพียง 256 สี

ไฟล์ .GIF มี 2 สกุล ได้แก่

- GIF87 พัฒนาขึ้นในปี ค.ศ. 1987 เป็นไฟล์กราฟิกรุ่นแรกที่สนับสนุนการนำเสนอบนอินเทอร์เน็ต เป็นไฟล์ที่มีขนาดเล็กและแสดงผลสีได้เพียง 256 สี และกำหนดให้แสดงผลแบบโครงร่างได้ (Interlace)
- GIF89A พัฒนาขึ้นในปี ค.ศ. 1989 เป็นไฟล์กราฟิกที่พัฒนาต่อจาก GIF87 โดยเพิ่มความสามารถการแสดงผลแบบพื้นโปร่งใส (Transparent) และการสร้างภาพเคลื่อนไหว (GIF Animation) ซึ่งเป็นไฟล์กราฟิกที่มีความสามารถพิเศษ โดยนำเอาไฟล์ภาพหลายๆไฟล์มารวมกัน และนำเสนอภาพเหล่านั้นโดยอาศัยการหน่วงเวลา มีการใส่รูปแบบการนำเสนอลักษณะต่างๆ (Effects) ในลักษณะภาพเคลื่อนไหว

นอกจากนี้ ยังมีคุณสมบัติพิเศษอีก 3 ประการ คือ

- Interlaced คือ การแสดงภาพออกเป็นช่วงๆ แทนที่จะไล่จากบนลงล่างเหมือนปกติ
- Progressive คือ การแสดงภาพหยาบก่อนแล้วจึงค่อยๆ ชัดขึ้น
- Transparent คือ การแสดงภาพโดยไม่มีพื้นฉากหลัง

2.2.2 ไฟล์สกุล JPG (Joint Photographer's Experts Group)

ไฟล์ JPEG หรือ JPG ถูกพัฒนาขึ้นเพื่อใช้งานกับภาพที่มีสีสันสดใสและมีความละเอียดสูงมาก สามารถแสดงสีได้ถึง 16.7 ล้านสี ไฟล์มีขนาดเล็กมาก นิยมใช้กันมาก

เป็นอีกไฟล์หนึ่งที่นิยมใช้บนอินเตอร์เน็ตมักใช้กรณี

- ภาพที่ต้องการนำเสนอมีความละเอียดสูง และใช้สีจำนวนมาก (สนับสนุนสีถึง 24 บิต)
- ต้องการบีบไฟล์ตามความต้องการของผู้ใช้

- ไฟล์ชนิดนี้มักจะใช้กับภาพถ่ายที่นำมาสแกน และต้องการนำไปใช้บนอินเทอร์เน็ต เพราะให้ความคมชัดและความละเอียดของภาพสูง

จุดเด่น

- สนับสนุนสีได้ถึง 24 บิต
- สามารถกำหนดค่าการบีบไฟล์ได้ตามที่ต้องการ
- มีระบบแสดงผลแบบหยาบและค่อยๆ ขยายไปสู่ละเอียดในระบบ โปรเกรสซิฟ
- มีโปรแกรมสนับสนุนการสร้างจำนวนมาก
- เรียกดูได้กับ ตัวแสดงผลกราฟิคทุกตัว
- ตั้งค่าการบีบไฟล์ได้ (compress files)

จุดด้อย

- ทำให้พื้นของรูปโปร่งใสไม่ได้

JPEG เป็นไฟล์รูปภาพที่ถูกบีบขนาดเหมือนกันไฟล์ GIF จะมีขนาดเล็กกว่าไฟล์รูปแบบ GIF มาก โดยใช้เทคนิค LOSSY ซึ่งคุณภาพของภาพจะต่ำลงมาตามขนานที่บีบ แต่ไม่จำกัดจำนวนสี มักใช้ในลักษณะของภาพถ่าย

ข้อเสียของการบีบไฟล์ (Compress File)

กำหนดค่าการบีบไฟล์ไว้สูง (1 - 10) แม้ว่าจะช่วยให้ขนาดของไฟล์มีขนาดต่ำ แต่ก็มีข้อเสียคือ เมื่อมีการส่งภาพจากเซิฟเวอร์ (Server) ไปแสดงผลที่ไคลเอนท์ (Client) จะทำให้การแสดงผลช้ามาก เพราะต้องเสียเวลาในการคลายไฟล์ ดังนั้นการเลือกค่าการบีบไฟล์ ควรกำหนดให้เหมาะสมกับภาพแต่ละภาพ

2.2.3 ไฟล์สกุล PNG (Portable Network Graphics)

เป็นไฟล์ที่ถูกพัฒนาขึ้นมาแทนไฟล์ GIF และไฟล์ JPG ในอนาคต ไฟล์ PNG สามารถใช้ได้กับทุกระบบปฏิบัติการ

จุดเด่น

- สนับสนุนสีได้ถึงตามค่าทรูคัลเลอร์16 บิต, 32 บิต หรือ 64 บิต)
- สามารถกำหนดค่าการบีบไฟล์ได้ตามที่ต้องการ
- มีระบบแสดงผลแบบหยาบและค่อยๆ ขยายไปสู่ละเอียด (Interlace)
- สามารถทำพื้นโปร่งใสได้

จุดด้อย

- หากกำหนดค่าการบีบไฟล์ไว้สูง จะใช้เวลาในการคลายไฟล์สูงตามไปด้วย แต่ขนาดของไฟล์จะมีขนาดต่ำ
- ไม่สนับสนุนกับ ตัวแสดงผลกราฟิค รุ่นเก่า สนับสนุนเฉพาะ IE 4 และ Netscape 4
- ความละเอียดของภาพและจำนวนสีขึ้นอยู่กับ Video Card

โดยสรุปก็คือ รูปแบบของภาพที่นิยมก็คือ .jpg กับ .gif เพราะว่าไฟล์มีขนาดเล็กมากและ .gif สามารถนำมาทำเป็นภาพเคลื่อนไหวได้ อย่างไรก็ตามยังมีอีกรูปแบบหนึ่งคือ .png แต่ยังมีปัญหาคือ browser บางตัวยังไม่สามารถแสดงผลได้

2.2.4 ไฟล์สกุล BMP MS-Windows bitmap format

ไฟล์ BMP เป็นรูปแบบพื้นฐานที่ใช้งานได้ดีกับโปรแกรมที่ทำงานภายใต้วินโดว์ ซึ่งมีความเร็วสูง เป็นไฟล์มีขนาดใหญ่ เพราะไม่ได้รับการบีบย่อข้อมูลเหมือนกับ .GIF หรือ .JPG ไฟล์ BMP ที่เห็นบ่อยๆ คือ ภาพวอลล์เปเปอร์ที่แสดงบนจอภาพของวินโดว์ โดยสามารถแสดงได้ตั้งแต่ 2, 16, 256 และ 16 ล้านสี

2.2.5 ไฟล์สกุล TIFF Tagged Image File Format

สามารถใช้กับระบบปฏิบัติการหลายชนิด สามารถจัดเก็บภาพสีทั้งโหมด Index color, RGB และ CMYK รวมถึงขาวดำ และ ภาพระดับเทา ภาพส่วนใหญ่จะเป็นภาพที่เกิดจากการสแกนรูปภาพต่าง ๆ และจะจัดเก็บเป็นไฟล์นามสกุล .TIF

ไฟล์แบบ TIFF เป็นรูปแบบที่มีคุณภาพความคมชัดของภาพสูงที่สุด ไม่ว่าจะย่อหรือขยายภาพคุณภาพที่แสดงก็ยังคงเดิม แต่จะมีขนาดใหญ่เนื่องจากมีการรวมเอาข้อมูลจากบิตแมป วันที่และเวลาที่ไฟล์ถูกสร้าง รวมทั้งซอฟท์แวร์ที่ใช้ เหมาะสมในวงการสื่อสิ่งพิมพ์

แต่ในทางปฏิบัติ การทำงานกับภาพจะใช้รูปแบบการจัดเก็บที่เป็นสากลนิยมคือ JPEG และ GIF ความแตกต่างของ 2 รูปแบบ ในทางปฏิบัติคือ การจัดเก็บงานในรูปแบบ GIF จะทำให้งานมีขนาดความจุเล็ก เนื่องจากรูปที่จัดเก็บแบบ GIF เป็น Index color คือให้สีน้อยกว่าปกติ ส่วนรูปที่จัดเก็บแบบ JPEG จะสามารถควบคุมคุณภาพจัดเก็บได้ 3 ระดับคือ ต่ำ กลาง และสูง ถ้าภาพที่ต้องการจัดเก็บมีความละเอียดต่ำการเก็บแบบ GIF และ JPEG จะมองไม่เห็นความแตกต่าง แต่ภาพ JPEG จะมีความจุมากกว่า GIF และรูปแบบการจัดเก็บอีกรูปแบบคือ PSD เป็นรูปแบบของ Photoshop การจัดเก็บวิธีนี้จะทำให้ภาพเก็บข้อมูลทุกอย่างรวมถึงการทำงาน layers ด้วยทำให้ง่ายต่อการทำงานเมื่อเปิดภาพขึ้นมาทำงานใหม่ สำหรับคุณภาพของภาพดิจิทัล (Digital image) ถูกกำหนดโดย จำนวนสีที่ใช้, ชนิดของการจัดเก็บ, จำนวน level, จำนวน บิต ของภาพ และความละเอียด (Resolution) ความหมายของจำนวนสีและชนิดของการจัดเก็บได้กล่าวไว้แล้ว สำหรับจำนวน level คือ ระดับการไล่สีจากจุดที่อ่อนสู่จุดที่เข้ม ภาพที่ใช้จำนวน level มาก จะให้คุณภาพที่

ดีดังรูป โดยส่วนใหญ่ภาพที่มีคุณภาพดีปกติจะใช้ 256 levels แต่อุปกรณ์อินพุต (Input Devices) อาจสร้างภาพได้คุณภาพมากกว่านี้คือ 1,024 levels ถึง 16,384 levels

สำหรับจำนวน บิต ของภาพจะมีผลต่อคุณภาพของภาพ โดยที่ภาพ 1 บิต จะแสดงผลของสีได้ 2 สีคือ ขาว-ดำ ในแต่ละพิกเซลในขณะทำงาน 8 บิต จะสามารถสร้างความเข้ม-จางให้กับภาพได้ 256 ระดับ ภาพระดับ RGB คือ ภาพที่ประกอบจากภาพ 8 บิต 3 channel รวมกัน จึงถือเป็นภาพ 24 บิต ((28)3) และเป็นภาพที่ต่างจาก CMYK และ Monochrome ภาพ 24 บิต (ภาพ 32 บิต) (8 บิต) ซอฟแวร์ที่ใช้ตกแต่งภาพ เช่น Photoshop จะเป็นซอฟแวร์ที่สามารถจัดการกับภาพ 24 บิต 8 บิต และ 32 บิต

2.3 กระบวนการวัดทางสถิติ (Statistical Operations) [3]

ที่ได้กล่าวมาแล้วเกี่ยวกับการรับภาพเข้าสู่คอมพิวเตอร์และรูปแบบของการนำภาพที่อยู่ในหน่วยความจำไปแสดงผลด้วยอุปกรณ์ต่างๆ จะเห็นได้ว่ามีกระบวนการทำได้หลาย ๆ วิธี เพื่อที่จะให้ได้ผลตามที่ต้องการ

ในการแบ่งกระบวนการประมวลผลเกี่ยวกับภาพแบ่งได้หลักๆ 3 อย่างด้วยกันคือ ระดับต่ำ, ระดับปานกลาง และระดับสูง โดยขึ้นอยู่กับการกระทำเกี่ยวกับบิตของภาพ ซึ่งอาจต้องผ่านการประมวลระดับต่ำ, ระดับปานกลาง และ ระดับสูง ตามลำดับเพื่อให้ได้ผลลัพธ์ออกมาดีที่สุด

การประมวลระดับต่ำ จะจัดการกับภาพแบบขาว-ดำ โดยปกติจะทำการสร้างภาพที่สองขึ้นมาโดยให้มีเฉพาะข้อมูลที่ต้องการ ส่วนไหนที่ไม่ต้องการจะทำการตัดออกไป

การประมวลระดับปานกลาง เป็นการประมวลเกี่ยวกับการบ่งบอกว่าภาพมีลักษณะรูปร่าง พื้นที่ หรือจุดของภาพจากการประมวลผลแบบระดับต่ำ

การประมวลระดับสูง เป็นการรู้ลักษณะทั่วๆ ไปที่จำเป็นของภาพ เช่น มีการเชื่อมต่อกันของรูปร่างอะไรบ้าง เพื่อที่จะทำให้รู้รูปร่างที่แท้จริงของวัตถุ ผลของการประมวลผลในระดับนี้จะนำไปสู่การวิเคราะห์ภาพด้วย

ในส่วนของนี้จะกล่าวถึงการประมวลผลภาพในระดับอย่างต่ำ ซึ่งจะขึ้นอยู่กับตำแหน่งต่างของพิกเซล

2.3.1 การแปลงภาพสู่ระดับสีเทา (Grey-Level Transformations)

การแปลงภาพสีเป็นระดับเทาได้นั้นทำได้หลายวิธีเช่น การใช้สมการ 2.17 แปลงค่า RGB ให้เป็นค่าเฉลี่ยแล้วแทนลงไปในพิกเซลนั้นๆซึ่งจะได้จำนวนบิตที่ใช้แทนระดับความเข้มของภาพเท่ากับ 24 บิตเหมือนเดิม หรืออีกวิธีโดยการเปลี่ยนจากภาพสี RGB เป็นภาพ ระดับเทา ซึ่งจำนวนบิตที่ใช้แทนระดับความเข้มของภาพจะเหลือ 8 บิต โดยจะมีการคูณด้วยค่าคงที่ไปที่แต่ละสีของ

RGB ซึ่งค่าคงที่นั้นโดยความจริงแล้วอาจจะไม่ใช่ตัวเลขที่ตายตัวเสมอไปแต่โดยทั่วไปแล้วมักเป็นดังสมการ

$$I = 0.299 \times red + 0.587 \times green + 0.114 \times blue \qquad (2.1)$$

โดยที่ I คือค่าของความเข้มมีค่าอยู่ในช่วงตั้งแต่ 0 (สีดำ) ไปจนถึง 255 (สีขาว) แสดงได้ดังรูปที่ 2.1

รูปที่ 2.1 แสดงการไล่สีระดับเทา (Gray scale)

2.3.2 การทำ Threshold

ใช้เพื่อ เปลี่ยนทิศทางของภาพ โดยทำการแยกกลุ่มของภาพออกเป็นส่วน ๆ

กระบวนการทำ โดยปกติ การทำ Threshold จำทำการเปรียบเทียบกับค่าของ threshold โดยถ้ามากกว่าจะเปลี่ยนค่าพิกเซลที่ตำแหน่งนั้นให้เป็นค่าสูงสุด และถ้าน้อยกว่าจะเปลี่ยนค่าพิกเซลตำแหน่งนั้นให้เป็นค่าต่ำสุด จึงทำให้ได้ภาพมีแค่สองระดับคือ สูงสุด หรือ ต่ำสุด

ตัวอย่างการแปลงภาพระดับเทา เป็นภาพขาวดำ

ตารางที่ 2.1 ตารางแสดงค่าในภาพระดับเทา

47	230	170	237	71	219	124	30	53	113
66	111	40	170	74	145	132	44	78	187
199	190	141	224	160	103	170	241	198	196
162	195	192	15	111	160	2	97	224	162
88	152	210	175	229	62	155	157	22	54
216	223	117	196	27	6	250	74	1	83
190	231	88	99	39	28	195	197	145	7
118	206	95	252	114	52	54	136	215	13
1	145	88	183	97	72	152	91	92	117
228	4	28	253	245	207	28	173	210	39

จากตารางสามารถคำนวณค่า Threshold ได้เป็น 128 เมื่อนำไปเข้ากระบวนการ Threshold จะได้ค่าออกมา ดังตาราง

ตารางที่ 2.2 ตารางแสดงค่าในภาพขาวดำหลังการทำ Threshold

0	1	1	1	0	1	0	0	0	0
0	0	0	1	0	1	1	0	0	1
1	1	1	1	1	0	1	1	1	1
1	1	1	0	0	1	0	0	1	1
0	1	1	1	1	0	1	1	0	0
1	1	0	1	0	0	1	0	0	0
1	1	0	0	0	0	1	1	1	0
0	1	0	1	0	0	0	1	1	0
0	1	0	1	0	0	1	0	0	0
1	0	0	1	1	1	0	1	1	0

ความผิดพลาดจากการทำ Threshold

โดยปกติความผิดพลาดจะเกิดจากการผลกระทบจากพื้นหน้าหรือพื้นหลังของภาพโดยที่เมื่อทำการแยกระดับของสีเทาออกเป็นกลุ่ม ความผิดพลาดหลัก ๆ ที่พบได้

ชนิดที่ 1 ทุกพิกเซลไม่เข้ามาอยู่ตามกลุ่มที่ต้องการ

ชนิดที่ 2 บางพิกเซล ไม่ควรจะเข้ามาอยู่ในกลุ่มที่ไม่ต้องการ

กระบวนการ Threshold ถูกนำไปใช้ในการสร้างภาพขาวดำ โดยอาจมีกระบวนการต่างๆ เพิ่มเข้ามาด้วยเพื่อทำให้คุณภาพของภาพดียิ่งขึ้น

2.4 ทฤษฎีการหาขอบ (Edge Detection Methods) [4]

การหาขอบภาพ (Edge Detection) เป็นการหาเส้นรอบวัตถุที่อยู่ในภาพ เมื่อทราบเส้นรอบวัตถุ เราจะสามารถคำ นวณหาพื้นที่ (ขนาด) หรือรู้จำชนิดของวัตถุนั้นได้ อย่างไรก็ตาม การหาขอบภาพที่ถูกต้องสมบูรณ์ไม่ใช่เป็นเรื่องที่ง่าย โดยเฉพาะอย่างยิ่งการหาขอบของภาพที่มีคุณภาพต่ำ มีความแตกต่างระหว่างพื้นหน้ากับพื้นหลังน้อย หรือมีความสว่างไม่สม่ำเสมอทั่วภาพขอบภาพเกิดจากความแตกต่างของความเข้มแสงจากจุดหนึ่งไปยังอีกจุดหนึ่ง หากต่างนี้มีค่ามาก ขอบภาพก็จะเห็นได้ชัด ถ้าความแตกต่างมีค่าน้อย ขอบภาพก็จะไม่ชัดเจน

ถ้าต้องการหาขอบภาพในแนวนอนอย่างง่าย วิธีการก็คือหาผลต่างระหว่างจุดหนึ่งกับจุดที่อยู่ข้างล่าง (หรือข้างบน) ของจุดนั้น ดังนี้

$$Y_{diff}(x,y) = I(x,y) - I(x,y+1) \tag{2.2}$$

โดยที่ Y_{diff} คือค่าความแตกต่างในแนวแกนตั้ง และ $I(x,y)$ คือค่าความเข้มแสงของจุดภาพที่ตำแหน่ง (x,y)

การหาขอบภาพในแนวตั้งก็สามารถหาได้เช่นเดียวกันคือ

$$X_{diff}(x,y) = I(x,y) - I(x-1,y) \quad (2.3)$$

โดยที่ X_{diff} คือค่าความแตกต่างในแนวนอน

บางครั้งเราต้องการรวมผลต่างของค่าความแตกต่างในแนวแกนนอน และแกนตั้งเข้าด้วยกัน เพื่อที่จะได้มีตัววัดความแรงของขอบภาพ (Gradient Magnitude) เพียงตัวเดียว เนื่องจากค่าความแตกต่างอาจมีค่าเป็นบวกหรือลบ ดังนั้น การบวกค่าความแตกต่างของทั้งสองแกนอาจทำ ให้ขอบภาพเกิดการหักล้างกันเอง ในทางปฏิบัติ เราจะต้องนำ ค่าสัมบูรณ์ (Absolute Value) หรือค่ากำลังสอง (Squared Value) ของค่าความแตกต่างของทั้งสองแกนมาบวกกันแทน นอกจากหาความแรงของขอบภาพแล้ว การหาทิศทางของขอบภาพ (Gradient Direction) ก็มีประโยชน์เช่นกัน การหาทิศทางของขอบภาพสามสามารถทำ ได้โดยการใช้สมการ ต่อไปนี้

$$GD(x,y) = tan^{-1}\left(\frac{Y_{diff}(x,y)}{X_{diff}(x,y)}\right) \quad (2.4)$$

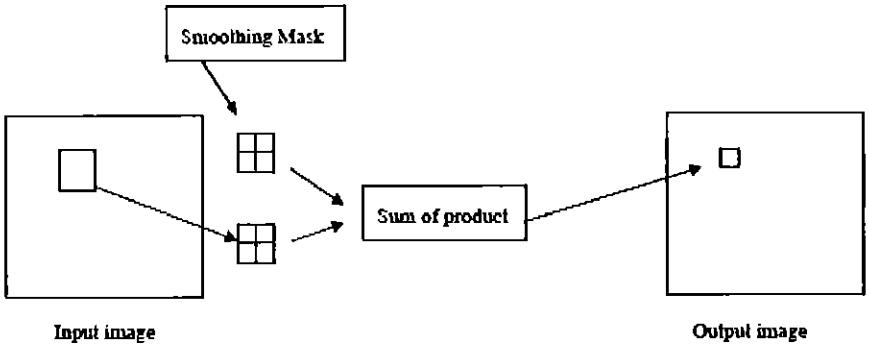
$GD(x,y)$ คือทิศทางของขอบภาพที่ตำแหน่ง (x,y) Y_{diff} คือค่าความแตกต่างในแนวแกนตั้ง และ X_{diff} คือค่าความแตกต่างในแนวนอน

2.5 ตัวกรองสำหรับทำให้ภาพเบลอ [5]

ตัวกรองสำหรับทำให้ภาพเบลอเป็นตัวกรองที่ใช้สำหรับทำให้ภาพมีความเบลอขึ้นและกำจัด สัญญาณรบกวน ซึ่งมี 2 หลักใหญ่ ๆ คือ ตัวกรองเชิงเส้น และตัวกรองไม่เชิงเส้น

2.5.1 ตัวกรองเชิงเส้น

เป็นการทำคอนโวลูชระหว่างภาพตั้งต้นกับเทมเพลตดังอธิบายโดยใช้แผนได้ดังรูป

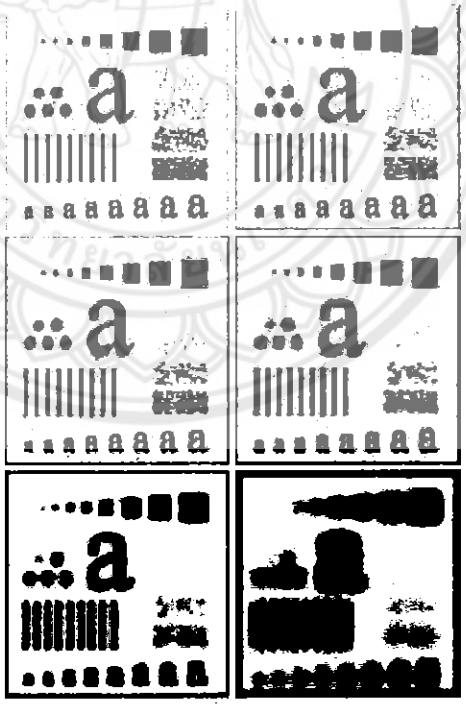


รูปที่ 2.2 รูปแสดงการหาด้วยตัวกรองเชิงเส้น

$\frac{1}{9}\times$ | 1 | 1 | 1 |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$\frac{1}{16}\times$ | 1 | 2 | 1 |
|---|---|---|
| 2 | 4 | 2 |
| 1 | 2 | 1 |

รูปที่ 2.3 รูปแสดงเทมเพลตแต่ละแบบ



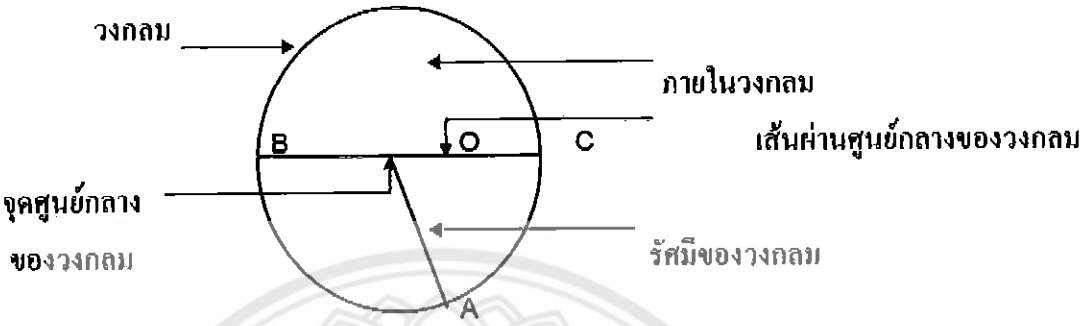
รูปที่ 2.4 รูปแสดงภาพผลลัพธ์เมื่อผ่านการทำโดยใช้ตัวกรองเชิงเส้นด้วยเทมเพลตต่าง ๆ

ข้อดี สามารถกำจัดสัญญาณรบกวนต่าง ๆ ได้

ข้อเสีย รายละเอียดเล็ก ๆ ของภาพที่มีลักษณะคล้าย สัญญาณรบกวน จะหายไปด้วย

2.6 วงกลม [6]

วงกลม คือ เซตของจุดทุกจุดซึ่งอยู่ห่างจากจุดคงที่ จุดหนึ่งบนระนาบเดียวกันเป็นระยะทางเท่า ๆ กัน



รูปที่ 2.5 ส่วนประกอบของวงกลม

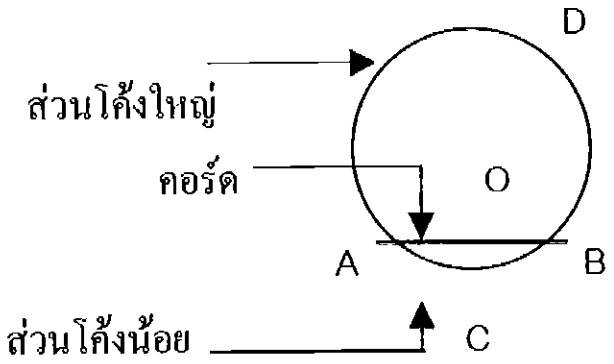
จากรูป 2.6 จุด O เป็นจุดคงที่ เรียกจุด O นี้ว่า จุดศูนย์กลางของวงกลม A เป็นจุดอยู่บนวงกลม เรียก OA ว่า รัศมีของวงกลม B และ C เป็นจุดอยู่บนวงกลม และ BC ผ่านจุดศูนย์กลาง O เรียก BC ว่า เส้นผ่านศูนย์กลางของวงกลม

2.6.1 จุดศูนย์กลาง (Center) คือ จุดคงที่ซึ่งอยู่ภายในวงกลม อยู่ห่างจากจุดทุกจุดที่ประกอบเป็นวงกลมเป็น ระยะทางเท่า ๆ กัน บนระนาบเดียวกัน

2.6.2 รัศมี (Radius) คือ เส้นตรงที่ลากจากจุดศูนย์กลางไปพบจุด ๆ หนึ่งบนเส้นรอบวง

2.6.3 เส้นผ่านศูนย์กลาง (Diameter) คือ เส้นตรงที่ลากจากจุดบนเส้นรอบวงผ่านจุดศูนย์กลางของวงกลมไปตัดจุด จุดอีกจุดหนึ่งบนเส้นรอบวง

2.6.4 คอร์ด (Chord) คือ ส่วนของเส้นตรงที่มีจุดปลายทั้งสองอยู่บนวงกลม คอร์ดจะแบ่งวงกลมออกเป็น 2 ส่วน

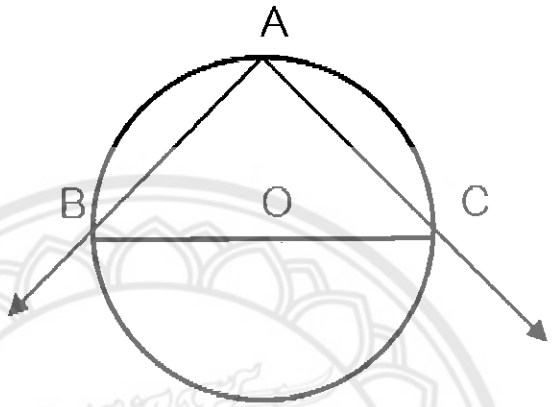


รูปที่ 2.6 เส้นคอร์ดของวงกลม

2.6.5 เส้นสัมผัสวงกลม คือ เส้นตรงที่ลากตัดวงกลมเพียงจุดเดียวเท่านั้น

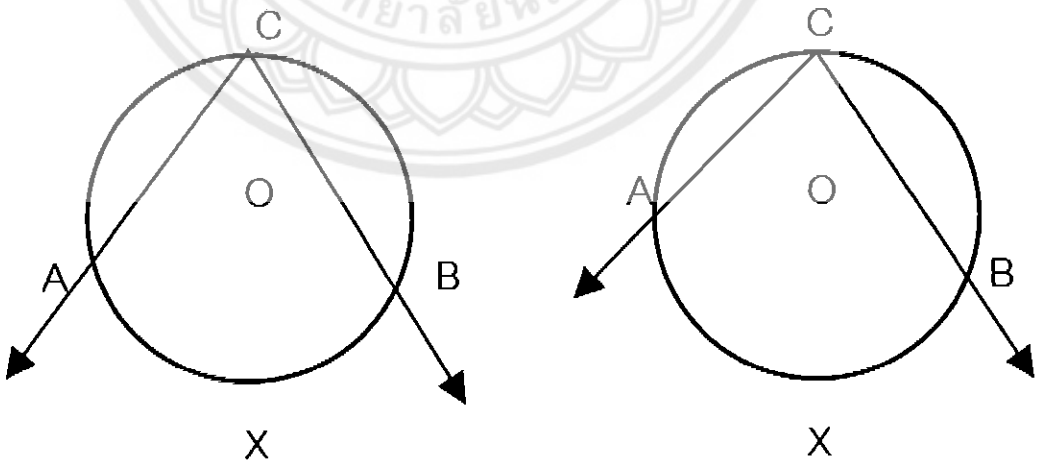
2.6.7 มุมในส่วนต่าง ๆ ของวงกลม

2.6.7.1 มุมในครึ่งวงกลม คือ มุมที่มีจุดยอดอยู่บนวงกลมและแขนทั้งสองของมุมผ่านจุดปลายทั้งสองของเส้นผ่านศูนย์กลาง



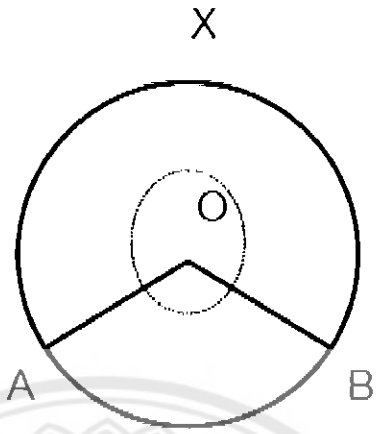
รูปที่ 2.7 มุมในครึ่งวงกลม

2.6.7.2 มุมในส่วนโค้งวงกลม คือ มุมที่มีจุดยอดมุมอยู่บนวงกลมและแขนทั้งสองข้างของมุมตัดวงกลม



รูปที่ 2.8 มุมในส่วนโค้งวงกลม

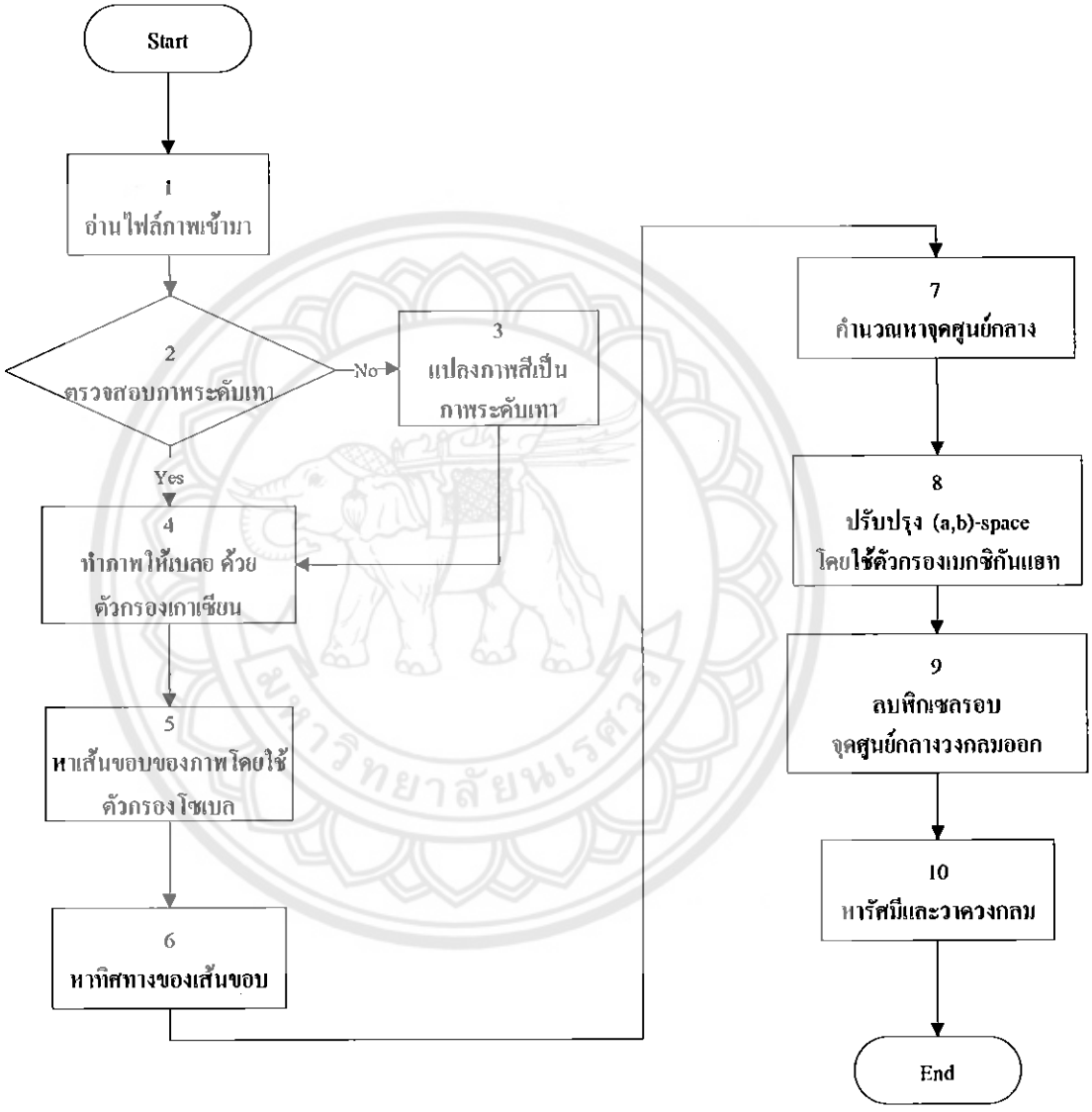
2.6.7.3 มุมที่จุดศูนย์กลางของวงกลม คือ มุมที่มีจุดศูนย์กลางเป็นจุดยอดมุมและมีรัศมีเป็นแขนของมุม



รูปที่ 2.9 มุมที่จุดศูนย์กลางของวงกลม

บทที่ 3
ขั้นตอนการทดลอง

3.1 แผนผังโปรแกรม [7]



รูปที่ 3.1 แผนผังโปรแกรม

3.2 รายละเอียดแผนผังโปรแกรม

3.2.1 อ่านไฟล์ภาพเข้ามาในโปรแกรม

อ่านไฟล์ภาพเข้ามาในโปรแกรมโดยใช้คำสั่งของ MATLAB คือ

ชื่อตัวแปร = imread('ชื่อไฟล์.นามสกุล')

3.2.2 ตรวจสอบภาพระดับเทา (gray-scale)

ตรวจสอบภาพที่อ่านเข้ามา ถ้าภาพที่อ่านเข้ามาเป็นภาพสีก็จะทำการแปลงภาพสีให้เป็นภาพระดับเทา (Gray-scale image) เพื่อให้ง่ายต่อการคำนวณและประมวลผล แต่ถ้าเป็นภาพระดับเทา (Gray-scale image) อยู่แล้วก็จะทำขั้นตอนต่อไปได้เลย

3.2.3 แปลงภาพสีเป็นภาพระดับเทา (Gray-scale image)

แปลงภาพสีที่อ่านเข้ามาเป็นภาพระดับเทา (gray-scale image) โดยใช้คำสั่ง MATLAB คือ

ชื่อตัวแปร = rgb2gray(ตัวแปรที่เก็บภาพสี)

3.2.4 ทำภาพให้เบลอ ด้วย ตัวกรองเกาเซียน (Gaussian smooth filter)

ทำภาพให้เบลอเพื่อลดสัญญาณรบกวนภายในภาพ ก่อนนำภาพไปหาเส้นขอบของภาพ โดยเราจะใช้ตัวกรองเกาเซียน (Gaussian smooth filter) จากสูตร

$$smoothfilter[i,j] = \frac{1}{(\sqrt{2\pi}\sigma)^2} \cdot e^{-\frac{i^2+j^2}{2\sigma^2}} \qquad (3.1)$$

เมื่อ (i,j) คือ พิกัดของพิกเซล และ σ คือ ขนาดของฟิลเตอร์

3.2.5 หาเส้นขอบของภาพโดยใช้ตัวกรองโซเบล (Sobel filter)

หาเส้นขอบของภาพในแนวตั้งโดยคอนโวลูทภาพกับเทมเพลตแนวตั้งของตัวกรองโซเบล (Vertical Sobel filter) ซึ่งเทมเพลตแนวตั้งของตัวกรองโซเบล (Vertical Sobel filter) คือ $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ และ หาเส้นขอบของภาพในแนวนอนโดยคอนโวลูทภาพกับเทมเพลตแนวนอนของตัวกรองโซเบล (Horizontal Sobel filter) ซึ่งเทมเพลตแนวนอนของตัวกรองโซเบล (Horizontal Sobel filter) คือ $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ แล้วนำมานำเส้นขอบของภาพในแนวตั้งและแนวนอนมารวมกันจากสูตร

$$sobel_{merge}[i,j] = \sqrt{(sobel_{vert}[i,j])^2 + (sobel_{horiz}[i,j])^2} \qquad (3.2)$$

เมื่อ $sobel_{merge}$ คือ เส้นขอบของภาพ $sobel_{vert}$ คือ เส้นขอบของภาพแนวตั้ง และ $sobel_{horiz}$ เส้นขอบของภาพแนวนอน

3.2.6 หาทิศทางของเส้นขอบ (Direction map)

เราสามารถคำนวณหาทิศทางของเส้นขอบของภาพจาก เส้นขอบของภาพในแนวนอนและเส้นขอบของภาพในแนวตั้ง (Horizontal and Vertical sobel filter) ในรูปของมุม ซึ่งมีค่าอยู่ระหว่าง $(-\pi, \pi)$ โดยคำนวณได้จากสูตร

$$\theta[i,j] = tan^{-1} \frac{sobel_{vert}[i,j]}{sobel_{horiz}[i,j]} \quad (3.3)$$

เมื่อ θ คือ ทิศทางของเส้นขอบของภาพในแต่ละจุด

(i,j) คือ พิกัดของพิกเซล

$sobel_{horiz}$ คือ เส้นขอบของภาพในแนวนอน

$sobel_{vert}$ คือ เส้นขอบของภาพในแนนตั้ง

3.2.7 คำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

ณ จุด (a,b) ใดๆ บนภาพ นำมาคำนวณหาค่าระยะห่างจากเส้นขอบของภาพโดยใช้สูตร

$$r = \sqrt{(a - P_x)^2 + (b - P_y)^2} \quad (3.4)$$

เมื่อ r คือ ระยะห่างจากจุด (a,b) ไปยังจุด (P_x, P_y) โดยที่ (a,b) เป็นพิกัดที่คาดว่าเป็นจุดศูนย์กลางของวงกลม และ (P_x, P_y) คือ พิกัดของเส้นขอบของภาพที่อยู่ในขอบเขตตั้งแต่จุด (a-maxR,b-maxR) ไปจนถึง (a+maxR,b+maxR)

หลังจากได้ค่า r มาแล้ว ก็นำเพิ่มค่าใน (a,b,r)-space

$$A(a,b,r) \leftarrow A(a,b,r) + 1/r \quad (3.5)$$

เมื่อ $A(a,b,r)$ เป็นอะเรย์ 3 มิติที่จะเก็บค่าของจำนวนเส้นที่รัศมี r ต่างๆ ณ จุด (a,b)

แปลง (a,b,r)-space ไปเป็น (a,b)-space โดยให้

$$C(a,b) = \max\{A(a,b,minR), A(a,b,minR+1), \ldots, A(a,b,maxR)\} \quad (3.6)$$

เมื่อ $C(a,b)$ คือ จะเก็บค่าของจำนวนเส้นที่มากที่สุดระหว่าง minR ถึง maxR ณ จุด (a,b) เพื่อใช้หาจุดศูนย์กลางของวงกลม เรียก C(a,b) ว่า (a,b)-apace

3.2.8 ปรับปรุง (a,b)-space โดยใช้ตัวกรองเมกซิกันแฮท (Mexican hat filter)

จากการคำนวณหาจุดศูนย์กลางโดยใช้ Circle Hough Transform จะได้กลุ่มของพิกเซลที่คาดว่าจะเป็นจุดศูนย์กลางของวงกลม แต่กลุ่มของพิกเซลนั้นจะค่อยข้างกระจาย เราจึงใช้ตัวกรองเมกซิกันแฮท ขนาด 17x17 (17x17 Mexican hat filter) ซึ่งเป็นตัวกรองที่ช่วยทำให้ภาพเบลออย่างหนึ่งช่วยทำให้ จุดที่เป็นจุดศูนย์กลางจริงเด่นขึ้นมา

```
mexhatlarge = [0  0  0  0  0  0 -1 -1 -1 -1 -1  0  0  0  0  0  0;
               0  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0  0;
               0  0 -1 -1 -1 -2 -3 -3 -3 -3 -3 -2 -1 -1 -1  0  0;
               0  0 -1 -1 -2 -3 -3 -3 -3 -3 -3 -3 -2 -1 -1  0  0;
               0 -1 -1 -2 -3 -3 -3 -2 -3 -2 -3 -3 -3 -2 -1 -1  0;
               0 -1 -2 -3 -3 -3  0  2  4  2  0 -3 -3 -3 -2 -1  0;
              -1 -1 -3 -3 -3  0  4 10 12 10  4  0 -3 -3 -3 -1 -1;
              -1 -1 -3 -3 -2  2 10 18 21 18 10  2 -2 -3 -3 -1 -1;
              -1 -1 -3 -3 -3  4 12 21 24 21 12  4 -3 -3 -3 -1 -1;
              -1 -1 -3 -3 -2  2 10 18 21 18 10  2 -2 -3 -3 -1 -1;
              -1 -1 -3 -3 -3  0  4 10 12 10  4  0 -3 -3 -3 -1 -1;
               0 -1 -2 -3 -3 -3  0  2  4  2  0 -3 -3 -3 -2 -1  0;
               0 -1 -1 -2 -3 -3 -3 -2 -3 -2 -3 -3 -3 -2 -1 -1  0;
               0  0 -1 -1 -2 -3 -3 -3 -3 -3 -3 -3 -2 -1 -1  0  0;
               0  0 -1 -1 -1 -2 -3 -3 -3 -3 -3 -2 -1 -1 -1  0  0;
               0  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0  0;
               0  0  0  0  0  0 -1 -1 -1 -1 -1  0  0  0  0  0  0];
```

รูปที่ 3.2 ตัวกรองเมกซิกันแฮท ขนาด 17x17

3.2.9 ลบพิกเซลรอบจุดศูนย์กลางวงกลมออก

หลังจากทำการปรับปรุง (a,b)-space แล้ว จะได้กลุ่มของจุดศูนย์กลางวงกลมโดยในกลุ่มของพิกเซลเหล่านี้จะมีอยู่ 1 พิกเซล ที่มีค่าของจำนวนเส้นตรงที่ลากผ่านมากที่สุด เราจึงต้องทำการลบเอาพิกเซลรอบๆออกไปเพื่อให้ได้พิกเซลที่เป็นจุดศูนย์กลางจริงๆ ของวงกลม

3.2.10 หารัศมีและวาดวงกลม

หลังจากได้จุดศูนย์กลางของวงกลมแล้ว นำจุดนั้นมาคำนวณหารัศมีโดยจะเพิ่มค่าของเส้นขอบที่จุด P ซึ่ง P เป็นจุดที่อยู่บนเส้นขอบและห่างจากจุดศูนย์กลาง (a,b) เป็นระยะรัศมี r ดังในสูตร

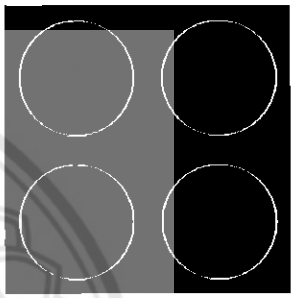
$$R(r) = \sum_{Pecircle(r)} E(P) \tag{3.7}$$

โดยที่ r จะอยู่ระหว่าง minR ถึง maxR เมื่อได้ค่า R(r) มาแล้วก็ทำการปรับค่า R(r) ที่มีค่าน้อยกว่า rthreshold ให้เป็น 0 เพื่อให้เหลือเฉพาะรัศมีของวงกลมจริง เมื่อได้ค่ารัศมีที่ถูกต้องแล้วก็วาดวงกลม

บทที่ 4
ผลการทดลองและวิเคราะห์ผลการทดลอง

4.1 ผลการทดลองสำหรับรูปที่สร้างขึ้นมาเอง

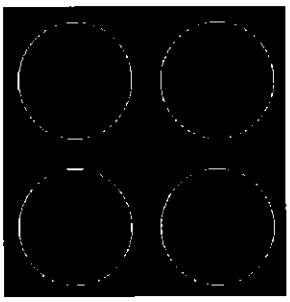
รูปที่ 4.1 ภาพที่มีรูปวงกลมซึ่งมีรัศมีเป็น 80 หน่วย จำนวน 4 วง



รูปที่ 4.2 ภาพเส้นขอบภาพของรูปที่ 4.1

รูปที่ 4.3 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

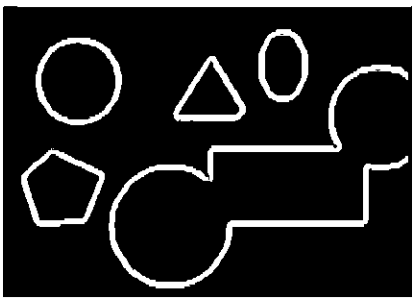
รูปที่ 4.4 ภาพจุดศูนย์กลางวงกลม



รูปที่ 4.5 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.1

ในรูปที่ 4.1 เป็นภาพวงกลมที่มีรัศมี 80 หน่วยจำนวน 4 วง หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับวงกลมได้ครบทั้ง 4 วง

รูปที่ 4.6 ภาพที่มีรูปทรงต่างๆ ทั้งวงกลม สามเหลี่ยม สี่เหลี่ยม ห้าเหลี่ยม และวงรี



รูปที่ 4.7 ภาพเส้นขอบภาพของรูปที่ 4.6

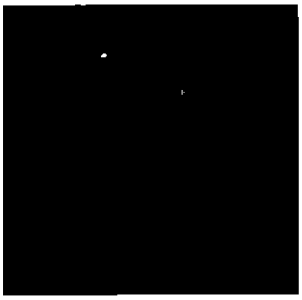
รูปที่ 4.8 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

รูปที่ 4.9 ภาพจุดศูนย์กลางวงกลม

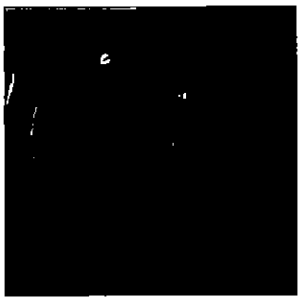
รูปที่ 4.10 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.6

ในรูปที่ 4.6 เป็นภาพที่มีรูปทรงต่างๆ ทั้งวงกลม สามเหลี่ยม สี่เหลี่ยม ห้าเหลี่ยม และวงรี หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับวงกลมได้ครบ

4.2 ผลการทดลองสำหรับรูปที่มีวงกลมภายในภาพเพียงวงเดียว



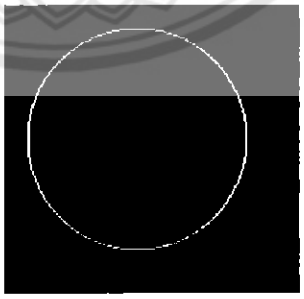
รูปที่ 4.11 ภาพลูกฟุตบอล



รูปที่ 4.12 ภาพเส้นขอบภาพของรูปที่ 4.11

รูปที่ 4.13 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

รูปที่ 4.14 ภาพจุดศูนย์กลางวงกลม

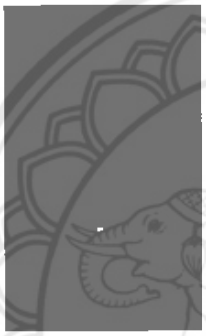


รูปที่ 4.15 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.11

ในรูปที่ 4.11 เป็นภาพลูกฟุตบอลซึ่งภายในภาพ ซึ่งภายในภาพมีรายละเอียดไม่มาก หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับลูกฟุตบอลได้

รูปที่ 4.16 ภาพรูปปั้นเด็กเล่นกงล้อ

รูปที่ 4.17 ภาพเส้นขอบภาพของรูปที่ 4.16



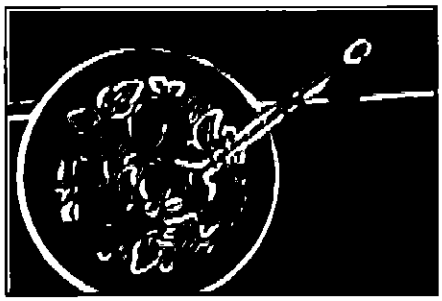
รูปที่ 4.18 ภาพหลังการคำนวณหาจุดศูนย์กลาง โดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

รูปที่ 4.19 ภาพจุดศูนย์กลางวงกลม

รูปที่ 4.20 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.16

ในรูปที่ 4.16 เป็นภาพรูปปั้นเด็กเล่นกงล้อ หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับกงล้อได้

รูปที่ 4.21 ภาพจานลายดอกไม้และไปป์



รูปที่ 4.22 ภาพเส้นขอบภาพของรูปที่ 4.21

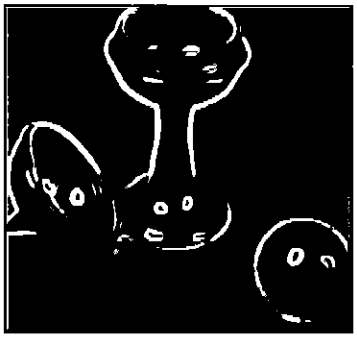
รูปที่ 4.23 ภาพหลังการคำนวณหาจุดศูนย์กลาง โดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

รูปที่ 4.24 ภาพจุดศูนย์กลางวงกลม

รูปที่ 4.25 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.21

ในรูปที่ 4.21 เป็นภาพจานลายดอกไม้และไปป์ หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับจานลายดอกไม้ได้

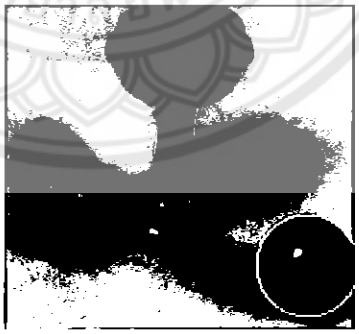
รูปที่ 4.26 ภาพลูกบอล



รูปที่ 4.27 ภาพเส้นขอบภาพของรูปที่ 4.26

รูปที่ 4.28 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

รูปที่ 4.29 ภาพจุดศูนย์กลางวงกลม



รูปที่ 4.30 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.26

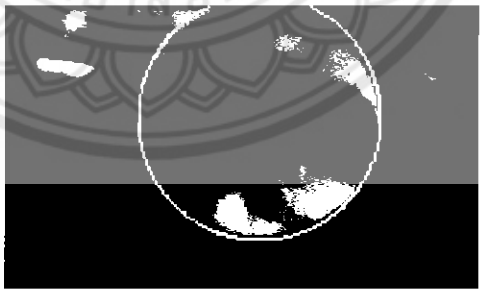
ในรูปที่ 4.26 เป็นภาพลูกบอล หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับลูกบอลได้

รูปที่ 4.31 ภาพเก้าอี้กวงกลม

รูปที่ 4.32 ภาพเส้นขอบภาพของรูปที่ 4.31

รูปที่ 4.33 ภาพหลังการคำนวณหาจุดศูนย์กลาง โดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

รูปที่ 4.34 ภาพจุดศูนย์กลางวงกลม

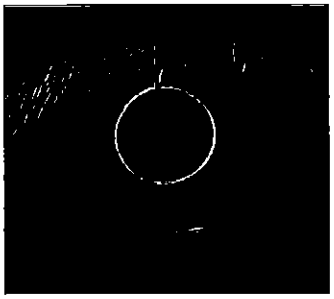


รูปที่ 4.35 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.31

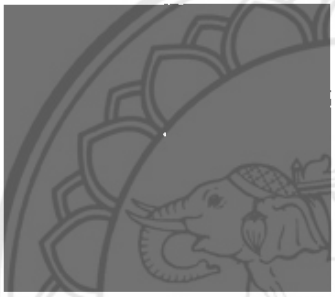
ในรูปที่ 4.31 เป็นภาพเก้าอี้วงกลมซึ่งมีท่อพีวีซีวางอยู่บนเก้าอี้ หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับเก้าอี้วงกลมได้

4.3 ผลการทดลองสำหรับรูปที่มีวงกลมภายในภาพมากกว่า 1 วง

รูปที่ 4.36 ภาพตาคน

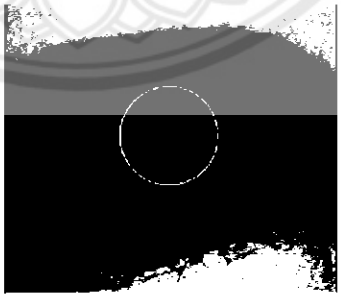


รูปที่ 4.37 ภาพเส้นขอบภาพของรูปที่ 4.36



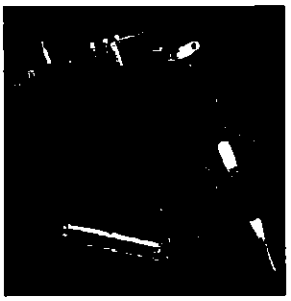
รูปที่ 4.38 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

รูปที่ 4.39 ภาพจุดศูนย์กลางวงกลม

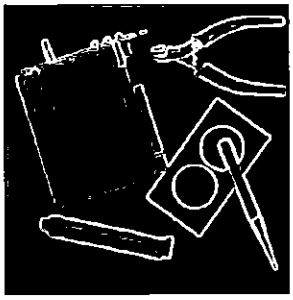


รูปที่ 4.40 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.36

ในรูปที่ 4.36 เป็นภาพตาคน หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับตาวงในได้ ส่วนตาวงนอกไม่สามารถตรวจจับได้เพราะเส้นขอบของภาพไม่ชัดเจน



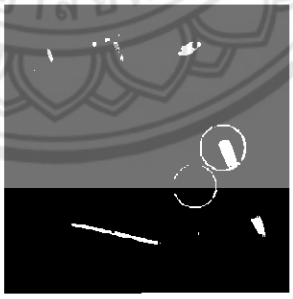
รูปที่ 4.41 ภาพเครื่องมือต่างๆ ซึ่งมีวงกลมในภาพ 2 วง



รูปที่ 4.42 ภาพเส้นขอบภาพของรูปที่ 4.41

รูปที่ 4.43 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

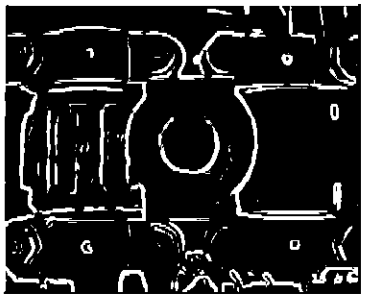
รูปที่ 4.44 ภาพจุดศูนย์กลางวงกลม



รูปที่ 4.45 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.41

ในรูปที่ 4.41 เป็นรูปเครื่องมือต่างๆ ซึ่งมีวงกลมในภาพ 2 วง หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับวงกลมได้ครบ 2 วง

รูปที่ 4.46 ภาพที่มีวงกลมซ้อนกัน 2 วง และมีรายละเอียดอื่นๆ ที่คล้ายวงกลม



รูปที่ 4.47 ภาพเส้นขอบภาพของรูปที่ 4.46

รูปที่ 4.48 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

รูปที่ 4.49 ภาพจุดศูนย์กลางวงกลม

รูปที่ 4.50 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.46

ในรูปที่ 4.46 เป็นภาพที่มีวงกลมซ้อนกัน 2 วง และมีรายละเอียดอื่นๆ ที่คล้ายวงกลม หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับวงกลมที่ซ้อนกันอยู่ 2 วงได้ แต่สำหรับรายละเอียดอื่นไม่สามารถตรวจจับได้ เนื่องจากเส้นขอบของภาพไม่ชัดเจน และบางวงไม่เป็นวงกลม

รูปที่ 4.51 ภาพผลส้มหลายลูก

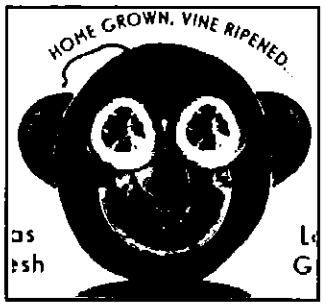
รูปที่ 4.52 ภาพเส้นขอบภาพของรูปที่ 4.51

รูปที่ 4.53 ภาพหลังการคำนวณหาจุดศูนย์กลาง โดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

รูปที่ 4.54 ภาพจุดศูนย์กลางวงกลม

รูปที่ 4.55 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.51

ในรูปที่ 4.51 เป็นภาพผลส้มหลายลูกภาพต้นฉบับเป็นภาพผลส้มหลายลูก หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับผลส้มได้ 2 ลูก ส่วนผลอื่นที่ไม่สามารถตรวจจับได้เพราะผลส้มไม่เป็นวงกลม



รูปที่ 4.56 ภาพผลไม้ที่นำมาวางเป็นรูปหน้าคน

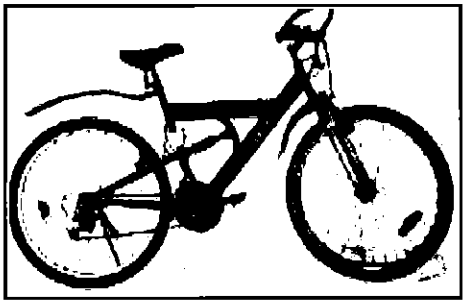
รูปที่ 4.57 ภาพเส้นขอบภาพของรูปที่ 4.56

รูปที่ 4.58 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

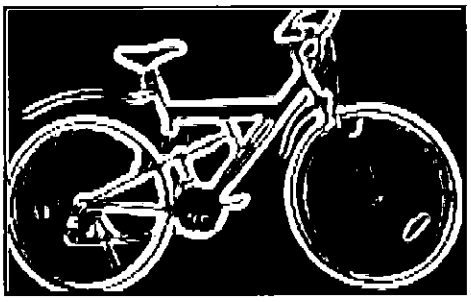
รูปที่ 4.59 ภาพจุดศูนย์กลางวงกลม

รูปที่ 4.60 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.56

ในรูปที่ 4.56 เป็นภาพผลไม้ที่นำมาวางเป็นรูปหน้าคน หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับวงกลมได้บางวงเพราะบางวงไม่ใช่วงกลมและเส้นขอบของภาพไม่ชัดเจน



รูปที่ 4.61 ภาพรถจักรยาน



รูปที่ 4.62 ภาพเส้นขอบภาพของรูปที่ 4.61

รูปที่ 4.63 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

รูปที่ 4.64 ภาพจุดศูนย์กลางวงกลม

รูปที่ 4.65 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.61

ในรูปที่ 4.61 เป็นภาพรถจักรยาน หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับล้อจักรยานได้ครบทั้ง 2 ล้อ แต่ไม่สามารถตรวจจับส่วนอื่นของจักรยานได้เพราะเส้นขอบของภาพไม่ชัดเจน

รูปที่ 4.66 ภาพฟองอากาศติดกัน 2 ฟอง

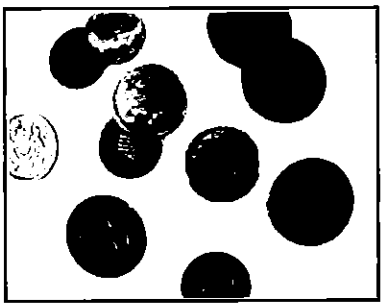
รูปที่ 4.67 ภาพเส้นขอบภาพของรูปที่ 4.66

รูปที่ 4.68 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

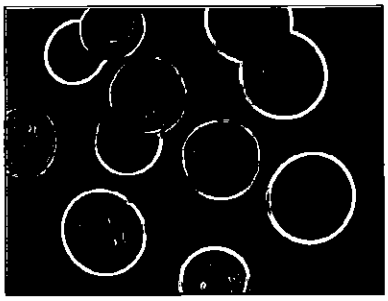
รูปที่ 4.69 ภาพจุดศูนย์กลางวงกลม

รูปที่ 4.70 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.66

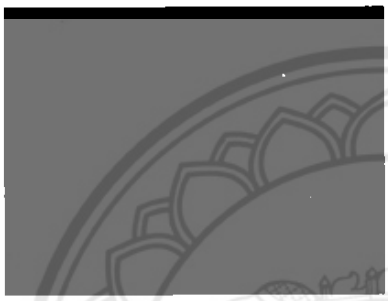
ในรูปที่ 4.66 เป็นภาพฟองอากาศติดกัน 2 ฟอง หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับฟองอากาศได้ทั้ง 2 ฟอง



รูปที่ 4.71 ภาพเหรียญซึ่งมีหลายเหรียญ

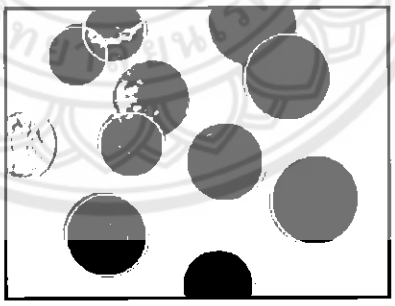


รูปที่ 4.72 ภาพเส้นขอบภาพของรูปที่ 4.71



รูปที่ 4.73 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space

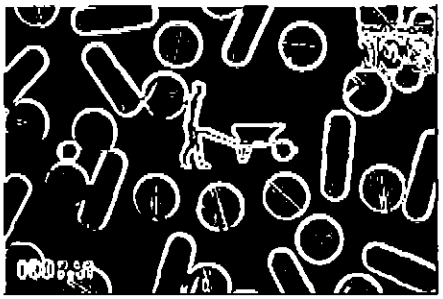
รูปที่ 4.74 ภาพจุดศูนย์กลางวงกลม



รูปที่ 4.75 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.71

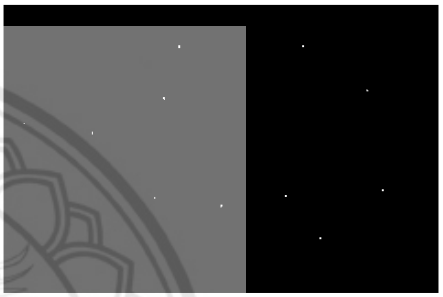
ในรูปที่ 4.71 เป็นภาพเหรียญซึ่งมีหลายเหรียญ หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับเหรียญได้บางเหรียญเท่านั้น เนื่องจากบางเหรียญไม่เป็นวงกลม เส้นขอบของภาพไม่ชัดเจนและสีของเหรียญกลืนกับสีพื้นหลัง

รูปที่ 4.76 ภาพเม็ดยา



รูปที่ 4.77 ภาพเส้นขอบภาพของรูปที่ 4.76

รูปที่ 4.78 ภาพหลังการคำนวณหาจุดศูนย์กลางโดยใช้ Circular Hough transform เก็บค่าไว้ใน (a,b)-space



รูปที่ 4.79 ภาพจุดศูนย์กลางวงกลม

รูปที่ 4.80 ผลลัพธ์ที่ได้หลังจากการรันโปรแกรมโปรแกรมตรวจจับวัตถุวงกลมภายในภาพจากรูปที่ 4.76

ในรูปที่ 4.76 เป็นภาพเม็ดยา หลังจากใช้โปรแกรมตรวจจับวัตถุวงกลมภายในภาพ ผลลัพธ์ที่ได้สามารถตรวจจับเม็ดยาได้บางเม็ดเท่านั้น เพราะขอบของภาพไม่ชัดเจน

ตารางที่ 4.1 สรุปผลการทดลองทั้งหมด

รูป	ขนาดรูป	จำนวนวงกลมในรูป	ตรวจจับได้	ความถูกต้อง
รูปที่ 4.1 ภาพวงกลม 4 วง	400x400	4	4	100 %
รูปที่ 4.6 ภาพรูปทรงต่างๆ	200x140	3	3	100 %
รูปที่ 4.11 ภาพลูกฟุตบอล	200x195	1	1	100 %
รูปที่ 4.16 ภาพรูปปั้นเด็กเล่นกงล้อ	122x200	1	1	100 %
รูปที่ 4.21ภาพจานและไปป์	200x132	1	1	100 %
รูปที่ 4.26 ภาพลูกบอล	200x185	1	1	100 %
รูปที่ 4.30 ภาพเก้าอี้	200x120	1	1	100 %
รูปที่ 4.36 ภาพตาคน	320x280	2	1	50 %
รูปที่ 4.41 ภาพเครื่องมือต่างๆ	255x256	2	2	100 %
รูปที่ 4.46 ภาพที่มีวงกลมซ้อนกัน 2 วง และรายละเอียดอื่นๆ	200x160	14	2	14.29 %
รูปที่ 4.51 ภาพผลส้ม	288x216	4	2	50 %
รูปที่ 4.56 ภาพผลไม้จัดเป็นรูปหน้าคน	200x188	5	3	60 %
รูปที่ 4.61 ภาพจักรยาน	200x123	3	2	66.67 %
รูปที่ 4.66 ภาพฟองอากาศ	160x200	2	2	100 %
รูปที่ 4.71 ภาพเหรียญ	320x241	11	8	72.73 %
รูปที่ 4.76 ภาพเม็ดยา	229x150	13	10	76.92 %

บทที่ 5
สรุปผล

5.1 สรุปผลการทดลอง

5.1.1 สรุปผลการทดลองสำหรับภาพที่สร้างขึ้นมาเอง

จากการทดลองรันโปรแกรมกับภาพที่เราสร้างขึ้นมาเอง จะสามารถตรวจจับวงกลมได้ง่าย เพราะรายละเอียดของภาพจะไม่มาก ถึงแม้ว่าภายในภาพจะมีภาพที่มีรูปร่างอื่นๆ นอกจากวงกลม ก็ยังสามารถตรวจจับวงกลมได้ทุกวง เพราะภาพที่เราสร้างขึ้นมาเองนั้นรายละเอียดจะไม่มาก ทำให้สามารถหาขอบของภาพชัดเจนจึงสามารถตรวจจับวงกลมได้ง่าย

5.1.2 สรุปผลการทดลองสำหรับภาพที่มีวงกลมภายในภาพวงเดียว

จากการทดลองรันโปรแกรมกับภาพที่มีวงกลมภายในภาพวงเดียว จะสามารถตรวจจับภาพได้ดี แต่ก็ขึ้นอยู่กับรายละเอียดของภาพและการหาเส้นขอบของภาพด้วย ถ้าภายในภาพมีรายละเอียดของภาพเยอะจะทำให้หาวงกลมได้ยากขึ้น และถ้าหลังจากหาเส้นขอบของภาพแล้วได้เส้นขอบไม่ชัด หรือขอบของภาพในส่วนที่เป็นวงกลมไม่เด่น จะทำให้หาวงกลมไม่เจอ

5.1.3 สรุปผลการทดลองสำหรับภาพที่มีวงกลมภายในภาพมากกว่า 1 วง

จากการทดลองรันโปรแกรมกับภาพที่มีวงกลมภายในภาพมากกว่า 1 วง จะสามารถตรวจจับวงกลมได้แต่อาจจะไม่ครบทุกวง ขึ้นอยู่กับขอบของภาพและรายละเอียดของภาพ ถ้าภาพมีรายละเอียดมากเกินไปจะทำให้หาเส้นขอบของภาพที่เป็นวงกลมได้ยาก และถ้าส่วนใดของภาพที่เป็นรูปวงแต่ไม่กลม จะไม่สามารถตรวจจับได้ ในวงที่สามารถตรวจจับได้ ถึงแม้ว่าจะเป็นวงกลมที่ไม่สมบูรณ์คือไม่ครบวง อาจจะแค่ 2 ใน 3 ของวง แต่ถ้าเส้นขอบของภาพชัดเจน เราก็สามารถตรวจจับได้

5.1.4 สรุปผลการทดลองเกี่ยวกับความสัมพันธ์ระหว่างเส้นขอบของวงกลมกับการตรวจจับวงกลม

จากการทดลองพบว่าความสมบูรณ์และความคมชัดของเส้นขอบของภาพมีผลต่อการหาจุดศูนย์กลางและรัศมีของวงกลม โดยถ้าเส้นขอบเป็นวงกลมสมบูรณ์และชัดเจนจะสามารถหาจุดศูนย์กลางและรัศมีได้แม่นยำกว่าภาพที่เส้นขอบไม่สมบูรณ์ ซึ่งถ้าเส้นขอบไม่สมบูรณ์อาจทำให้ไม่สามารถตรวจจับวงกลมวงนั้นได้

5.1.5 สรุปผลการทดลองเกี่ยวกับความแตกต่างของรัศมีของวงกลมภายในภาพ

จากการทดลองพบว่าภาพที่มีวงกลมภายในภาพมากกว่า 1 วง ถ้ารัศมีของวงกลมภายในภาพมีความแตกต่างกันมากจะทำให้ไม่สามารถตรวจจับวงกลมที่มีรัศมีขนาดเล็กได้ เพราะ เมื่อเราคำนวณหาจุดศูนย์กลางวงกลม เราจะนำค่าไปเก็บใน (a,b)-space แล้วหาค่าสูงสุดว่าจุด (a,b) ใดน่าจะเป็นจุดศูนย์กลาง แต่เมื่อมีวงกลมที่มีรัศมีแตกต่างกันมากๆ ค่าของจุดศูนย์กลางของวงเล็กจะถูกลบทิ้ง เนื่องจากค่าแตกต่างกันมากเกินไป

5.2 ปัญหาและอุปสรรคในการทำงาน

- เนื่องจากภายในโปรแกรมมีการวนลูปในการคำนวณค่อนข้างเยอะทำให้ในการประมวลผลแต่ละครั้งใช้เวลาในการประมวลผลค่อนข้างนาน ถ้าเป็นภาพที่มีรายละเอียดในภาพมากๆ หรือภาพที่มีขนาดใหญ่ จะใช้เวลาในการประมวลผลนานมาก
- ในการประมวลผลจะต้องมีการกรอก Parameter ต่างๆ ทั้งค่า Threshold ที่ใช้สำหรับการเช็คเส้นขอบของภาพ ค่าขนาดของ Smoothing filter ค่า minR และ maxR ซึ่งก็คือช่วงของรัศมีที่เป็นไปได้ ค่า ab-threshold ใช้สำหรับเลือกตัดจุดที่ไม่ใช่จุดศูนย์กลางออก (สัญญาณรบกวน) และค่า r-threshold ใช้สำหรับปรับรัศมีในวงกลมที่ซ้อนกันและมีจุดศูนย์กลางเดียวกัน ซึ่งในแต่ละภาพจะมีค่า Parameter ต่างๆ เหล่านี้ไม่เหมือนกัน เราต้องปรับค่าให้เหมาะสมกับแต่ภาพที่นำมาทดสอบ ทำให้มีการประมวลผลภาพ 1 ภาพอาจจะต้องทำการประมวลผลหลายครั้ง เพื่อนปรับเปลี่ยนค่า Parameter เหล่านี้ให้เหมาะกับแต่ละภาพ เพราะถ้าค่า Parameter เหล่านี้ไม่เหมาะสมกับภาพ ก็จะทำให้ไม่ได้ผลลัพธ์ตามต้องการ บางครั้งอาจจะทำให้ไม่สามารถตรวจจับวงกลมได้เลย
- ในภาพที่มีรายละเอียดในภาพเยอะ เช่นภาพวิว หลังจากหาเส้นขอบของภาพแล้ว จะมีสายเส้นของขอบของภาพเยอะ ทำให้การตรวจจับวงกลมทำได้ยาก และในบางครั้งจะไม่สามารถตรวจจับวงกลมได้เลย
- ในภาพที่มีรูปวง แต่ไม่ใช่วงกลมเช่นรูปโลก จะไม่สามารถตรวจจับได้

5.3 ข้อเสนอแนะ

- ในการตรวจหาวงกลมในภาพที่มีวงกลมหลายวง โดยมีความแตกต่างของรัศมีมากๆ เราจะไม่สามารถตรวจจับวงกลมที่มีรัศมีขนาดเล็กได้ เพราะในการคำนวณของเราจะใช้การเก็บรวบรวมทางสถิติ ว่าบริเวณใด ที่มีการตัดของเส้นตั้งฉากจากเส้นขอบของภาพมากที่สุด ถ้าวงที่มีรัศมีขนาดเล็ก ค่าบริเวณนั้นก็จะน้อยตาม ซึ่งเป็นผลทำให้ถูกตัดทิ้งไป ควรหาวิธีแก้ไขตรงจุดนี้ เพื่อให้สามารถตรวจจับวงกลมได้ดียิ่งขึ้น

- ในการคำนวณแต่ละครั้งใช้เวลาค่อนข้างมาก เพราะมีการวนลูปเพื่อคำนวณ ควรจะหาวิธีเพิ่มความเร็วในการคำนวณ
- ในการคำนวณต้องกรอกค่า Parameter หลายตัว ซึ่งในการตรวจจับวงกลมแต่ละภาพ เราอาจจะต้องคำนวณหลายครั้ง โดยเปลี่ยนค่า Parameter ไปเรื่อยๆ จนกว่าจะเจอค่าที่เหมาะสม ควรจะหาวิธีที่คำนวณหรือประมาณค่า Parameter ที่เหมาะสมก่อน เพื่อความรวดเร็วในการตรวจจับวงกลมมากขึ้น เพราะไม่ต้องเปลี่ยนค่า Parameter หลายๆ ครั้ง

เอกสารอ้างอิง

[1] Gregory A. Baxes. **Digital image processing.** New York : John Wiley & Sons, Inc. 1994.

[2] Maher A. Sid-Ahmed. **Image processing.** Singapore : McGraw-Hill, Inc. 1995.

[3] Andrian Low. **Introductory Computer Vision and Image Processing.** Singapore : McGraw-Hill Book Company. 1991.

[4] "Edge Detection." [Online]. Available : http://fivedots.coe.psu.ac.th/~montri/Teaching/240-373/Chapter8.pdf. 2007

[5] Ronald N. Brancewell. **Two – Dimensional Imaging.** New Jersey : Prentice-Hall, Inc. 1995.

[6] "วงกลม." [Online]. Available : http://linux.kr.ac.th/ebook/suvantee/b4.htm. 2007

[7] Jaroslav Borovicka. "Circle Detection Using Hough Transforms Documentation." [Online]. Available : http://linux.fjfi.cvut.cz/~pinus/bristol/imageproc/hw1/report.pdf. 2007

ประวัติผู้เขียนโครงงาน

ชื่อ นายปริญญา มาลาโรจน์

ภูมิลำเนา 485 หมู่ 2 ตำบลเมืองพาน อำเภอพาน
จังหวัดเชียงราย 57120

ประวัติการศึกษา

- จบมัธยมศึกษาจากโรงเรียนสามัคคีวิทยาคม จังหวัดเชียงราย
- ปัจจุบันกำลังศึกษาอยู่ระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E-mail: notiroy@msn.com