การประมวลผลภาพขั้นพื้นฐานด้วย ฟัซซี่มอร์โฟโลยี

Low-level Image Processing by Fuzzy Morphology

โดย

นายชายแดน สุทเธนทร์

นายอภิโชค เทียนวิไลรัตน์

นายอภิชาติ พรหมทัศน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์ปีการศึกษา 2542
ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง การประมวลผลภาพขั้นพื้นฐานด้วย ฟัซซี่มอร์โฟโลยี

Low-level Image Processing by Fuzzy Morphology

ผู้จัดทำ	นายชายแดน	สุทเธนทร์	40012085
	นายอภิโชค	เทียนวิไลรัตน์	40012114
	นายอภิชาติ	พรหมทัศน์	40013437

อาจารย์ที่ปรึกษา

....

(อาจารย์สาธิต อินทจักร์)

การประมวลผลภาพขั้นพื้นฐานด้วยฟัซซี่มอร์โฟโลยี

ผู้จัดทำ	นายชายแดน	สุทเธนทร์	40012085
	นายอภิโชค	เทียนวิไลรัตน์	40012114
	นายอภิชาติ	พรหมทัศน์	40013437

อาจารย์ที่ปรึกษา อาจารย์สาธิต อินทจักร์

บทคัดย่อ

การประมวลผลภาพขั้นพื้นฐานด้วยฟัซซี่มอร์โฟโลยี ที่นำเสนอในโครงงานนี้เป็นการประยุกต์ฟัซซี่เซตเข้ากับทฤษฎีมอร์โฟโลยี เพื่อใช้ในการประมวลผลภาพขั้นพื้นฐาน เช่น การกำจัดสัญญาณรบกวน, การปรับความเข้มของภาพและการหาขอบภาพเป็นต้น จากผลการทดลองฟัซซี่มอร์โฟโลยีสามารถประมวลผลกับภาพไบนารี่และภาพเกรย์สเกล ซึ่งเป็นการขยายแนวความคิดเดิมของทฤษฎีมอร์โฟโลยีที่ใช้ได้ดีกับภาพไบนารี่ ไปสู่การประมวลผลภาพเกรย์สเกลซึ่งได้ผลลัพธ์เป็นที่น่าพอใจ

Low-level Image Processing by Fuzzy Morphology

Staff	Mr. Chaidan	Sutthen	40012085
	Mr. Aphichoke	Tienvilairat	40012114
	Mr. Apichat	Promtus	40013437
Adviser	Mr. Sathit	Intajag	

ABSTRACT

In this thesis, the fuzzy morphology is described to apply with a low-level image processing. The morphology is a satisfaction when are employed to operate with a binary image. In our scheme, the fuzzy set is combines with morphology to operate on the gray scale images. From experiment results, the fuzzy morphology can be operate both binary and gray-scale, such as, removed noise, contrast, and edge detection etc., which proposed method have a good results.

กิตติประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถเสร็จสมบูรณ์ลุล่วงไปได้นั้น เนื่องจากได้รับความอนุเคราะห์จากอาจารย์ที่ปรึกษาคืออาจารย์สาธิต อินทจักร์ ที่คอยให้คำปรึกษา แนะแนวทาง และแหล่งข้อมูลเป็นอย่างดีตลอดมา จึงขอกล่าวคำขอบคุณเป็นอย่างสูงไว้ ณ. ที่นี้

นายชายแดน สุทเธนทร์
นายอภิโชค เทียนวิไลรัตน์
นายอภิชาติ พรหมทัศน์

สารบัญ

สารบัญ (ต่อ)

สารบัญภาพ

สารบัญภาพ(ต่อ)

บทที่ 1

บทนำ

วิธีการปรับปรุงภาพจะประกอบด้วยเทคนิคต่าง ๆ ที่จะปรับปรุงภาพหรือทำการเปลี่ยนแปลงภาพให้อยู่ในรูปแบบที่ต้องการ เพื่อทำการวิเคราะห์โดยมนุษย์หรือคอมพิวเตอร์ สำหรับประโยชน์ที่ได้รับจากการปรับปรุงภาพในการวิเคราะห์ด้วยคอมพิวเตอร์ก็คือ การกำจัดสัญญาณรบกวน การหาขอบภาพ การปรับปรุงภาพเพื่อให้เห็นความแตกต่างของวัตถุภายในภาพ การกำจัดสัญญาณรบกวนของภาพก็คือ การกำจัดสัญญาณที่ไม่พึ่งประสงค์ที่เกิดขึ้นในข้อมูลภาพให้หมดไป ส่วนการหาขอบภาพคือการดึงคุณลักษณะโครงร่างที่เด่นของวัตถุออกมา ลักษณะที่เด่นของวัตถุที่เรามองเห็นกันโดยทั่วไปก็คือสันหรือส่วนที่เป็นขอบภาพ

เทคนิคที่ได้นำมาใช้ในการปรับปรุงภาพมีอยู่มากมายแตกต่างกัน โดยขึ้นอยู่กับวิธีการที่นำมาปรับปรุงภาพ ในเรื่องของการกำจัดสัญญาณรบกวนของภาพจะมีวิธีการ เช่น การกำจัดสัญญาณรบกวนด้วยตัวกรองสัญญาณเชิงเส้น, การกำจัดสัญญาณรบกวนด้วยวิธีที่ไม่เป็นเชิงเส้น, การกำจัดสัญญาณรบกวนโดยใช้ตัวกรองสัญญาณค่ามัธยฐาน (Medium Filter) ในการหาขอบภาพด้วยวิธี Relaxation ในแต่ละวิธีก็จะมีความเหมาะสมในการใช้งานที่แตกต่างกันไป ในส่วนของโครงงานนี้ได้นำเสนอทฤษฎีของมอร์โฟโลยีมาประยุกต์ใช้งานร่วมกันกับฟัซซี่เซต เพื่อนำมาใช้ในการหาขอบภาพและการกำจัดสัญญาณรบกวน โดยเรียกวิธีการนี้ว่า ฟัซซี่มอร์โฟโลยี

1.1 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. ศึกษาทฤษฎีมอร์โฟโลยีขั้นพื้นฐานในการวิเคราะห์ภาพไบนารี
2. ศึกษาทฤษฎีฟัซซี่เซตเบื้องต้น
3. นำทฤษฎีมอร์โฟโลยีและฟัซซี่เซตมาประยุกต์ใช้งานร่วมกันในการวิเคราะห์ภาพเกรย์สเกล โดยวิธี กำจัดสัญญาณรบกวน การหาขอบภาพ และปรับระดับความเข้มของภาพ

1.2 ขอบเขตของปริญญานิพนธ์

ขอบเขตของปริญญานิพนธ์นี้ได้ โดยได้ออกเป็น 2 ส่วนได้แก่

ส่วนของทฤษฎี ซึ่งประกอบด้วย

- ทฤษฎีเซต และ ทฤษฎีฟัซซี่
- ฮีสโตแกรม

- ทฤษฎีฟัซซี่
- ฟังก์ชันการ์บอร์ (Garbor function)
- ฟังก์ชันเกาส์เซียน (Gaussian function)
- ทฤษฎีมอร์โฟโลยี

ในปริญญานิพนธ์ฉบับนี้ได้จำลองการทำงานของฟัซซี่มอร์โพโลยีด้วยโปรแกรม โดยนำสมการของตัวดำเนินการฟัซซี่ Erosion, Dilation, Opening, Closing และ Hit or Miss เขียนเป็นโปรแกรมเพื่อนำมาประมวลผลกับรูปภาพ โดยใช้โปรแกรมเดลไฟ เวอร์ชัน 4 (Delphi 4)

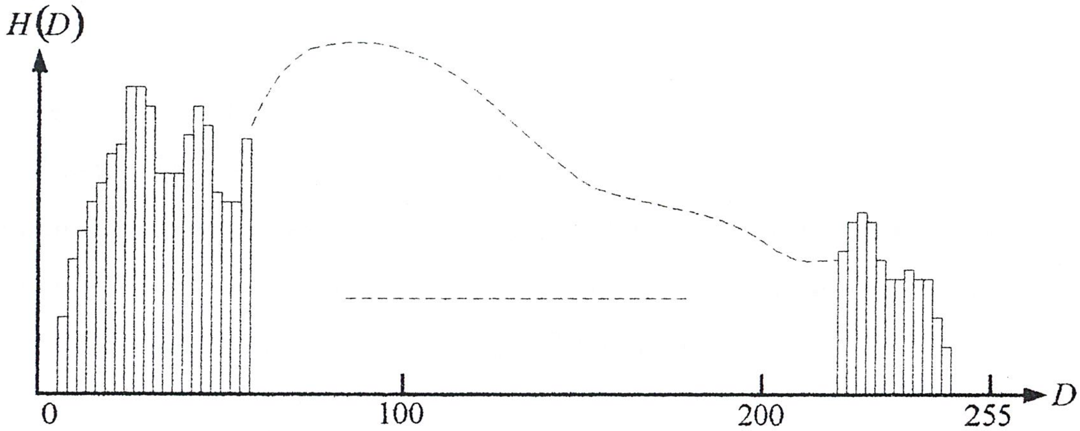
1.3 รายละเอียดและขั้นตอนการทำปริญญานิพนธ์

เริ่มจากการศึกษาค้นคว้าและหาข้อมูลในส่วนของทฤษฎีทั้งหมด ซึ่งสามารถแบ่งเนื้อหาออกได้เป็น 2 ส่วนใหญ่ ๆ คือ ส่วนแรกเป็นการศึกษาค้นคว้าเกี่ยวกับทฤษฎีของมอร์โฟโลยีและทฤษฎีของฟัซซี่ ซึ่งจะเป็นการศึกษาถึงทฤษฎีและแนวทางใช้งานพื้นฐานที่จะเป็นแนวทางในการนำทฤษฎีทั้งสองนำมาประยุกต์ใช้งานร่วมกัน และในส่วนที่สองเป็นการศึกษาค้นคว้าถึงทฤษฎีของฟัซซี่มอร์โฟโลยี ซึ่งจะเป็นส่วนที่ใช้ในการวิเคราะห์ภาพเกรย์สเกลตามวัตถุประสงค์ที่ได้กล่าวไว้ข้างต้น นอกเหนือจากนี้ยังมีเนื้อหาที่แตกต่างออกไปคือฮิสโตแกรม ซึ่งจะนำไปใช้หาจำนวนของจุดรูปภาพแต่ละระดับค่าสีเพื่อใช้ในการเปรียบเทียบการเปลี่ยนแปลงของรูปภาพหลังทำการประมวลผล ส่วนฟังก์ชันการ์บอร์และฟังก์ชันเกาส์เซียนเป็นฟังก์ชันที่นำไปใช้คำนวณหาค่าที่เหมาะสมของ SE ซึ่งจะขอกล่าวไว้ในส่วนนี้และมีหลักการและเนื้อหาดังนี้

1.3.1 ฮีสโตแกรม

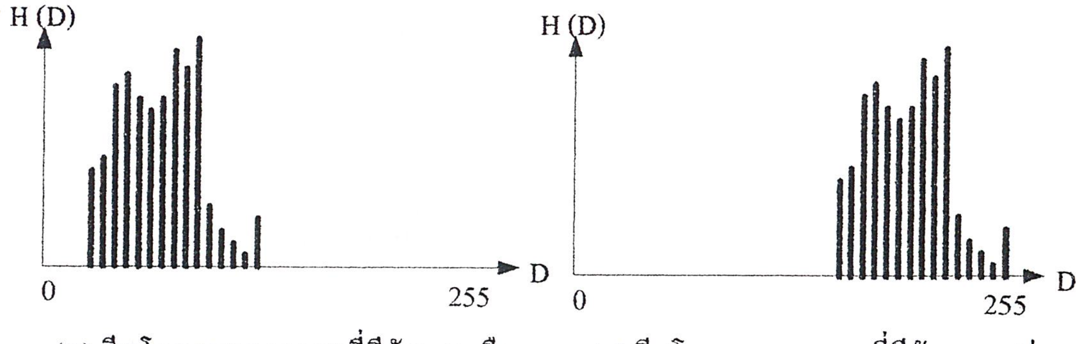
จากภาพเกรย์สเกลหรือภาพระดับสีเทาในแต่ละจุดภาพ ดังนั้นแต่ละจุดภาพจะถูกแทนค่าด้วยระดับสีเทาตั้งแต่ 0 ถึง 255 จากการเก็บข้อมูลจำนวนจุดภาพของแต่ละระดับสีเทาที่ปรากฏในภาพนั้น แล้วนำแสดงเป็นกราฟแท่งความถี่ของค่าระดับสีเทาทั้ง 256 ระดับ ซึ่งกราฟแท่งนี้เรียกว่าฮีสโตแกรมของภาพ ความสูงของกราฟแท่งจะแสดงเป็นจำนวนจุดภาพของแต่ละระดับของสีเทา ถ้าแต่ละแท่งของกราฟในฮีสโตแกรม คือ $H(D_i)$ โดย D_i เป็นค่าของระดับสีเทาที่มีค่าอยู่ระหว่าง 0 ถึง 255 $(0 \le D_i \le 255)$ ตัวอย่างของกราฟแท่งหรือฮีสโตแกรมแสดงได้ดังรูปที่ 1.1 โดยการพล็อตอาจทำในรูปของกราฟต่อเนื่องก็ได้ ตัวอย่างเช่นในรูปที่ 1.1 ถ้าหาก $H(D_i)$ เป็นจำนวนจุดภาพที่มีระดับสีเทาเท่ากับ D_i ดังนั้นจำนวนจุดภาพทั้งหมดในภาพดังกล่าวคือ N ซึ่งหาได้จาก

$$N = \sum_{i=0}^{255} H(D_i) \tag{1.1}$$



รูปที่ 1.1 ฮีสโตแกรมของรูปภาพ

การกระจายตัวของกราฟฮีสโตแกรมจะทำให้เราทราบลักษณะของภาพนั้น ๆ ได้ว่าเป็นภาพที่มีลักษณะเป็นภาพที่ความมืดหรือเป็นภาพที่มีความสว่าง โดยพิจารณาได้ดังนี้ ถ้าเป็นภาพที่มีลักษณะมืด ฮีสโตแกรมของภาพนั้น ๆ จะมีการกระจายตัวของแท่งกราฟอยู่ทางด้านขวาของกราฟเป็นส่วนใหญ่ ดังแสดงในรูปที่ 1.2 (ก) แต่ถ้าเป็นภาพที่มีความสว่าง ฮีสโตแกรมก็จะมีลักษณะเบนไปทางด้านซ้ายของกราฟ ดังแสดงในรูปที่ 1.2 (ข)



(ก) ฮีสโตแกรมของภาพที่มีลักษณะมืด (ข) ฮีสโตแกรมของภาพที่มีลักษณะสว่าง

รูปที่ 1.2 (ก) ฮีสโตแกรมของภาพที่มีลักษณะมืด
(ข) ฮีสโตแกรมของภาพที่มีลักษณะสว่าง

รูปร่างของฮีสโตแกรมจะเป็นข้อมูลพื้นฐานที่มีประโยชน์อย่างมาก จะนำมาใช้เป็นตัวเปรียบเทียบระหว่างรูปร่างอิสโตแกรมรูปต้นแบบกับฮีสโตแกรมของภาพ หลังจากผ่านการประมวลผลด้วยตัวดำเนินการ

1.3.2 ฟังก์ชันเกาส์เซียน (Gussian function)

ฟังก์ชันเกาส์เซียนเป็นฟังก์ชันที่ใช้ในการกรองสัญญาณเพื่อใช้ในการกำจัดรบกวนโดยที่กำหนดจากค่าความเบี่ยงเบนมาตรฐาน (σ) ที่ทำหน้าที่บอกถึงความกว้างของช่วงของความถี่ที่จะให้ผ่าน การกระจายของฟังก์ชันเกาส์เซียน (Gussian Distribution) เป็นสมการที่ใช้ในการอธิบายความน่าจะเป็น (Probability) ของเหตุการณ์ที่น่าเกิดขึ้นซึ่งมีสมการดังนี้

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{1}{2}(\frac{x^2+y^2}{\sigma})} \tag{1.2}$$

โดยที่ σ เป็นจำนวนจริงที่มีค่ามากกว่า 0

ในโครงงานนี้จะนำฟังก์ชันเกาส์เซียนมาใช้ในการคำนวณหาค่าความเป็นสมาชิกของ SE ที่เหมาะสมโดยจะขึ้นอยู่กับการกำหนดค่าของของ σ และ ขนาดของ SE ด้วย

กรณีที่กำหนดให้ $\sigma = 1$ ตัวกรองสัญญาณนี้จะถูกเรียกว่า ตัวกรองสัญญาณแบบเกาส์เซียน ซึ่งจะได้ผลลัพธ์ SE จะได้ดังรูปที่ 1.3

0.59	0.97	0.59
0.97	1.60	0.97
0.59	0.97	0.59

รูปที่ 1.3 แสดงผลลัพธ์ SE ขนาด 3×3 ที่ $\sigma = 1$

1.3.3 ฟังก์ชันการ์บอร์ (Gabor function)

ฟังก์ชัน Gabor (สาธิต, 2538) ในที่นี้เป็นฟังก์ชันที่ใช้ในการกรองสัญญาณภาพ โดยในส่วนของโครงงานนี้เราได้นำส่วนของฟังก์ชัน Gabor มาใช้เพื่อทำการหาขอบภาพ โดยแบบการจำลองการคำนวณที่พัฒนามาจากการทดลองทางด้านชีววิทยาและ จิตวิทยาที่เกี่ยวข้องกับระบบการมองเห็น ที่มีส่วนใกล้เคียงกับสัตว์เลี้ยงลูกด้วยนม โดยทั่วไปตัวกรองสัญญาณชนิดฟังก์ชัน Gabor นี้จะ

มีผลทำให้ความไม่แน่นในข้อมูลมีค่าที่ต่ำสุด ทำให้ผลฟังก์ชัน Gabor ถูกนำไปใช้ในการประมวลผลระดับต่ำในหลายขั้นตอนด้วยกัน เช่น การปรับปรุงภาพ, การหาขอบภาพ, การแบ่งแยกวัตถุภายในภาพ โดยเนื้อหาภาพ (Texture Classification) เป็นต้น

ฟังก์ชัน Gabor แสดงดังสมการที่ 1.3

$$S(x) = e^{\left(-\frac{1}{2}\frac{(x-x_0)^2}{\sigma} + j\omega_x x\right)} \tag{1.3}$$

โดยที่ σ และ x_0 เป็นส่วนเบี่ยงเบนมาตรฐานและค่าเฉลี่ยของฟังก์ชันเกาส์เซียนที่เกิดขึ้นภายในหนึ่งรอบ และ ω_x คือความถี่ของสัญญาณไซน์ จากสมการสามารถแยกได้สองส่วนคือส่วนที่เป็นฟังก์ชันคี่กับฟังก์ชันคู่ โดยที่จุดศูนย์กลางหรือค่าเฉลี่ยของแต่ละรอบเท่ากับศูนย์ $(x_0 = 0)$ สามารถแสดงดังสมการดังต่อไปนี้

$$S_{odd}(x) = e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \sin(\omega_x x) \tag{1.4}$$

$$S_{even}(x) = e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \cos(\omega_x x) \tag{1.5}$$

ตัวกรองสัญญาณที่เป็นฟังก์ชันคี่ (1.5) และฟังก์ชันคู่ (1.6) ของ Gabor ชนิดสองมิติ โดยมีทิศทางของตัวกรองสัญญาณตามมุม θ สามารถกำหนดได้โดย

$$S_{odd}(x,y) = e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} \sin(\omega(x\cos\theta + y\sin\theta)) \tag{1.6}$$

$$S_{even}(x,y) = e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} \cos(\omega(x\cos\theta + y\sin\theta)) \tag{1.7}$$

ที่ ω เป็นขนาดของความถี่เชิงมุมของเวกเตอร์ในระนาบสองมิติ

ในโครงงานนี้ฟังก์ชัน Gabor ถูกนำมาใช้คำนวณหาค่าความเป็นสมาชิกของ SE ที่ใช้ในการขอบภาพด้วยตัวดำเนินการ Hit or Miss โดยใช้ฟังก์ชัน Odd gabor เพื่อให้ได้ค่าความเป็น

สมาชิกของ SE เหมาะสม โดยจะขึ้นอยู่กับการกำหนดค่าของของ σ, θ และ ω และ ขนาดของ SE ด้วย

กรณีที่กำหนดให้ $\sigma = 1, \theta = 1$ และ $\omega = 1$ แสดงผลลัพธ์ของการคำนวณหาค่าความเป็นสมาชิกของ SE ดังรูปที่ 1.4

0.87	0.87	0.99
0.95	1.00	0.95
0.99	0.87	0.87

รูปที่ 1.4 แสดงผลลัพธ์ SE ขนาด 3×3 ที่ $\sigma = 1, \theta = 1$ และ $\omega = 1$ ด้วย Odd gabor

ในส่วนของโครงงานนี้เราได้นำส่วนของฟังก์ชัน Odd gabor เพื่อมาใช้ในการหาขอบภาพ

บทที่ 2
การประมวลผลภาพดิจิตอลขั้นพื้นฐานด้วยมอร์โฟโลยี

ในบทนี้จะเป็นการกล่าวถึงทฤษฎีต่างๆ ที่เกี่ยวข้องในการนำไปใช้งานในการวิเคราะห์ภาพโดยใช้ฟัซซี่มอร์โฟโลยี ซึ่งฟัซซี่มอร์โฟโลยีที่นำเสนอในที่นี้เกิดจากการนำทฤษฎีฟัซซี่เซตและทฤษฎีมอร์โฟโลยีมาประยุกต์เข้าด้วยกัน ในบทนี้จะแยกในส่วนของเนื้อหาออกเป็นสองส่วนคือในส่วนของทฤษฎีฟัซซี่เซตและทฤษฎีมอร์โฟโลยีและการนำทั้งสองมาประยุกต์ใช้งานร่วมกันโดยมีรายละเอียดดังต่อไปนี้

2.1 ฟัซซี่เซต

ฟัซซี่เซต คือ เซตที่ใช้อธิบายสิ่งที่มีความคลุมเครือและไม่สามารถกำหนดขอบเขตได้อย่างชัดเจน โดยสมาชิกภายในเซตจะมีค่าความเป็นสมาชิกอยู่ในช่วง 0 ถึง 1 ซึ่งกำหนดโดยฟังก์ชันความเป็นสมาชิก

2.1.1 ฟังก์ชันความเป็นสมาชิกของคลิส์ปเซต (The Characterestic Function)

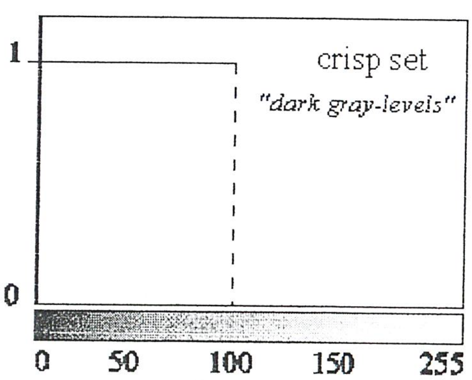
เป็นฟังก์ชันที่ใช้พิจารณาลักษณะความเป็นสมาชิกของเซตธรรมดา จะมีสมาชิกได้แก่ $\{0,1\}$ เขียนแทนด้วยสัญลักษณ์คือ $\mu_A : \cup \rightarrow \{0,1\}$ กำหนดเป็นเงื่อนไขได้คือ

$$\mu_A(x) = \begin{cases} 1 & if\ x \in A \\ 0 & 0\ otherwise \end{cases}$$

กำหนด μ_A ฟังก์ชันความเป็นสมาชิกของเซต A

x เป็นสมาชิกของเซต A

เพื่อให้เข้าใจมากขึ้นจะยกตัวอย่างการพิจารณาเซตของระดับสีเกรย์สเกล (Gray Scale) ดังรูปที่ 2.1



รูปที่ 2.1 คริส์ปเซต

กำหนดให้

A คือ สมาชิกของคริสัป

U คือ สมาชิกของเอกภพสัมพันธ์ มีสมาชิก คือ ค่าระดับสี 0- 255

จากรูปที่ 2.1 เป็นเซตของค่าระดับสีตั้งแต่ 0-255 เป็นเซตเอกภพสัมพันธ์ โดยเซต A เป็นเซตของสีดำคือค่าระดับสีที่อยู่ในช่วง 0- 100 จะเป็นสมาชิกของเซต A พิจารณาจากระดับความเป็นสมาชิกเท่ากับ 1 ส่วนค่าที่มีระดับสี 100-255 จะไม่เป็นสมาชิกของเซต A เนื่องจากค่าความเป็นสมาชิกเท่ากับ 0 จะเห็นว่าเซต A สามารถแบ่งแยกการเป็นสมาชิกได้อย่างชัดเจน คือการแบ่งจากความเป็นสมาชิกโดยพิจารณาค่าที่เท่ากับ 0 หรือ 1 ซึ่งก็คือคุณสมบัติของคริส์ปเซตนั้นเอง

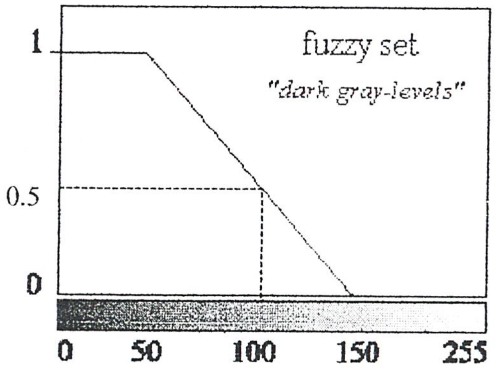
2.1.2 ฟังก์ชันความเป็นสมาชิกของฟัซซี่เซต (The membership Function)

ฟัซซี่เซตเป็นเซตที่แสดงถึงระดับความสำคัญของสมาชิกภายในกลุ่มแต่ละตัวกับความจำกัดความของเซตนั้นๆ โดยความสัมพันธ์นี้จะถูกแสดงในลักษณะของระดับความเป็นสมาชิกที่มีค่าอยู่ใน ช่วง [0,1] แทนที่จะแสดงว่าสมาชิกตัวใดเป็นสมาชิกหรือไม่เป็นสมาชิกของเซตนั้นอย่างเซตธรรมดา (ซึ่งมีค่าเป็น { 0,1 }) ถ้าเรากำหนดให้ U เป็นเซตของเอกภพสัมพัทธ์ และฟัซซี่เซต A มีสมาชิกเป็นของเซตเป็น $x_1, x_2, x_3, \ldots, x_n$ โดยที่ $x_i(i = 1, 2, 3, 4, \ldots, n)$ เป็นค่าฟัซซี่เซต ดังนั้นฟัซซี่เซต A สามารถแสดงในรูปแบบของความสัมพันธ์ในฟัซซี่เซตได้โดย $\mu_A(x_i)$ ความสัมพันธ์เช่นนี้ในทางทฤษฎีของฟัซซี่เซตจะเรียกว่าฟังก์ชันความเป็นสมาชิก

สามารถเขียนเป็นสัญลักษณ์ทางคณิตศาสตร์ได้โดย

$$\mu_A(x_i): U \rightarrow [0,1]$$

กำหนด $\mu_A(x_i)$ ก็คือฟังก์ชันความเป็นสมาชิกของเซต A



รูปที่ 2.2 แสดงความเป็นสมาชิกของฟัซซี่เซต

จากรูปที่ 2.2 เซต A มีสมาชิกคือระดับสีที่มีความเป็นสมาชิกอยู่ในช่วง [0-1] คือค่าระดับสีตั้งแต่ 0-150 ส่วนระดับสีที่มีความเป็นสมาชิกเท่ากับ 0 จะไม่เป็นสมาชิกของเซต A คือค่าระดับสีที่อยู่ในช่วง 151-255 จะเห็นได้ค่าความเป็นสมาชิกของค่าระดับสีจะลดลง ตัวอย่างจากค่าระดับสีที่ 100 ค่าความเป็นสมาชิกเท่ากับ 0.5 แต่ยังคงเป็นสมาชิกของเซต A ซึ่งแสดงให้เห็นว่าฟัซซี่เซตจะมีสมาชิกที่มีความยืดหยุ่นกว่าเซตธรรมดา

2.1.3 การกำหนดฟังก์ชันความเป็นสมาชิก

ในการกำหนดค่าความเป็นสมาชิกของฟัซซี่เซต สามารถกำหนดได้ 2 แบบคือ

1. การกำหนดฟังก์ชันความเป็นสมาชิกเชิงตัวเลข

หมายถึง การแสดงอัตราของความเป็นสมาชิกเชิงตัวเลขในเอกภพสัมพัทธ์ การกำหนดฟังก์ชันเชิงตัวเลขในโดเมนที่ต่อเนื่อง หมายถึงการแบ่งย่านในเอกภพสัมพัทธ์ออกเป็นส่วนๆ เรียกว่า “การควอนไตซ์เซชั่น(Quantization)” และการกำหนดระดับค่าความเป็นสมาชิกของเซต แต่ละค่าเป็นตัวเลข

2. การกำหนดฟังก์ชันความเป็นสมาชิกด้วยฟังก์ชันเฉพาะ

โดยที่ฟังก์ชันความเป็นสมาชิกจะจำแนกระดับของค่าความเป็นสมาชิกในเอกภพสัมพัทธ์ เพื่อทำหน้าที่แปลงปริมาณอินพุทที่อยู่ในรูปของคลิปเซตให้อยู่ในโดเมนของฟัซซี่เซต

หลักเกณฑ์ในการกำหนดฟังก์ชันความเป็นสมาชิก

1. การกระจายของข้อมูล หรือคุณลักษณะของความสัมพันธ์ระหว่างอินพุท และเอาต์พุทของกระบวนการควบคุมว่าเป็นสมาชิกเชิงเส้นหรือไม่เป็นเชิงเส้น ถ้าเป็นเชิงเส้นควรใช้ฟังก์ชันสามเหลี่ยม (ศิริชัย, 2542) ถ้าไม่เป็นเชิงเส้นควรใช้ฟังก์ชันเอ็กซ์โปเนนเชียล (ศิริชัย, 2542)

2. คุณสมบัติในการคำนวณ เช่น ความเร็วในการคำนวณขนาดของหน่วยความจำ

3. ความต่อเนื่อง และไม่ต่อเนื่องของข้อมูล โดยทั่วไปจำนวนฟังก์ชันที่ใช้กำหนดระดับของค่าลิงก์จีสติก (Linguistic) (ศิริชัย, 2542) ซึ่งเป็นค่าของตัวแปรที่กำหนดค่าย่านหรือขนาดปริมาณทางฟิสิกส์ที่สามารถกำหนดเป็นภาษามนุษย์ในที่นี่ไม่ควรไม่ต่ำกว่า 5 ถึง 7 ระดับ ถ้าต่ำกว่านี้จะเกิดความไม่ต่อเนื่อง และเกิดความผิดพลาดในผลลัพธ์ ถ้ามีจำนวนมากกว่านี้จะทำให้เพิ่มเวลาการคำนวณให้มากขึ้น

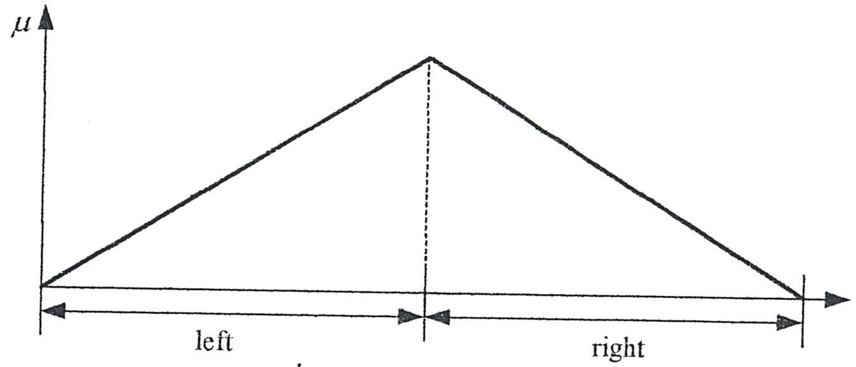
4. การกำหนดฟังก์ชัน แต่ละฟังก์ชันควรให้มีการทับกันระหว่างฟังก์ชัน (overlap) ของเซตที่อยู่ติดกัน เพื่อให้แน่ใจว่าการควบคุมต้องมีกฎควบคุมอย่างน้อย 1-2 กฎ

5. การกำหนดค่าพารามิเตอร์ของฟังก์ชันการเป็นสมาชิก

- ค่าสูงสุด (Peak Value) หมายถึง ค่าสูงสุดของฟังก์ชันความเป็นสมาชิกในแต่ละเทอม จะต้องมีค่าเท่ากับ 1

- ความกว้างทางซ้าย และขวา (Left and Right Width) หมายถึง ช่วงความกว้างระหว่างจุดสูงสุดและจุดศูนย์ของฟังก์ชันความเป็นสมาชิก หรือเรียกว่าซัพพอร์ตของค่าสมาชิกทั้งซ้าย และขวา ถ้าความกว้างทั้งสองเท่ากัน เรียกว่าฟังก์ชันนี้เป็นฟังก์ชันสมมาตรดังรูปที่ 2.3

- จุดตัด (Crosspoints) จุดที่ฟังก์ชัน 2 ฟังก์ชันที่อยู่ใกล้เคียงกันตัดกัน



รูปที่ 2.3 แสดงฟังก์ชันสมมาตร

จากรูปที่ 2.4 กำหนดให้

x เท่ากับจุดตัด

d_1 เท่ากับความกว้างระหว่างจุดเริ่มต้นและจุดตัดของฟังก์ชันที่ 1

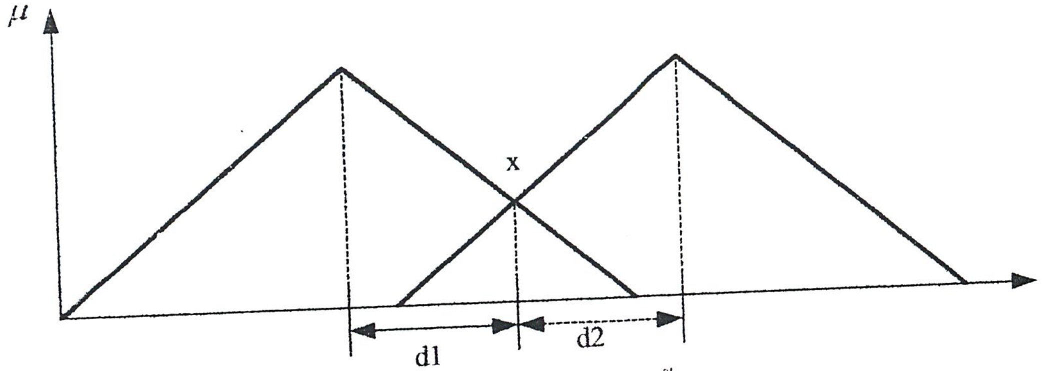
d_2 เท่ากับความกว้างระหว่างจุดเริ่มต้นและจุดตัดของฟังก์ชันที่ 2

ระดับของจุดตัด (Cross-point level) เท่ากับ ระยะระหว่างจุดตัดถึงแกน x

อัตราส่วนของจุดตัด (Cross-point ratio) เท่ากับ d_1/d_2

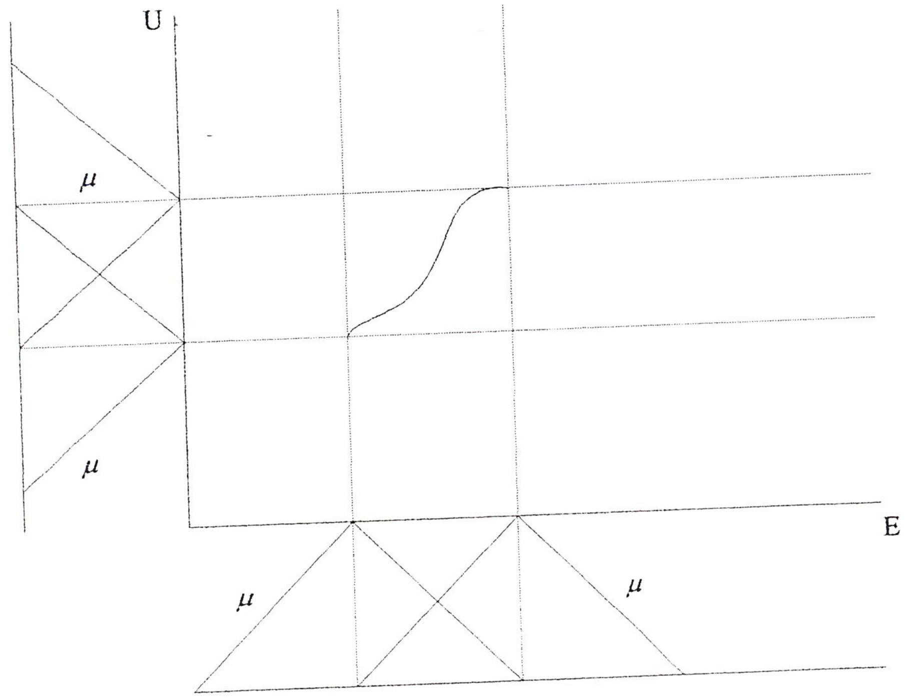
ในการออกแบบตัวควบคุมสำหรับระบบที่เป็นเชิงเส้น การใช้ฟังก์ชันความเป็นสมาชิกแบบสมมาตรมีระดับของจุดตัดเท่ากับ 0.5 และมีอัตราส่วนจุดตัดเท่ากับ 1 จะทำให้มีค่าสมรรถนะของระบบดี คือมีจุดพุ่งเกิน (overshoot) ต่ำ เวลาการเข้าถึง (Rise Time) เร็ว

ในส่วนของความต่อเนื่องของการควบคุม ควรจะใช้ระยะ d1 และ d2 ของทั้งสองฟังก์ชันที่ใกล้กันมีระยะเท่ากัน และที่ตำแหน่งที่ x ของค่าความเป็นสมาชิกของทั้งสองฟังก์ชันจะตรงกับตำแหน่งที่เป็นค่าสูงสุดพอดี

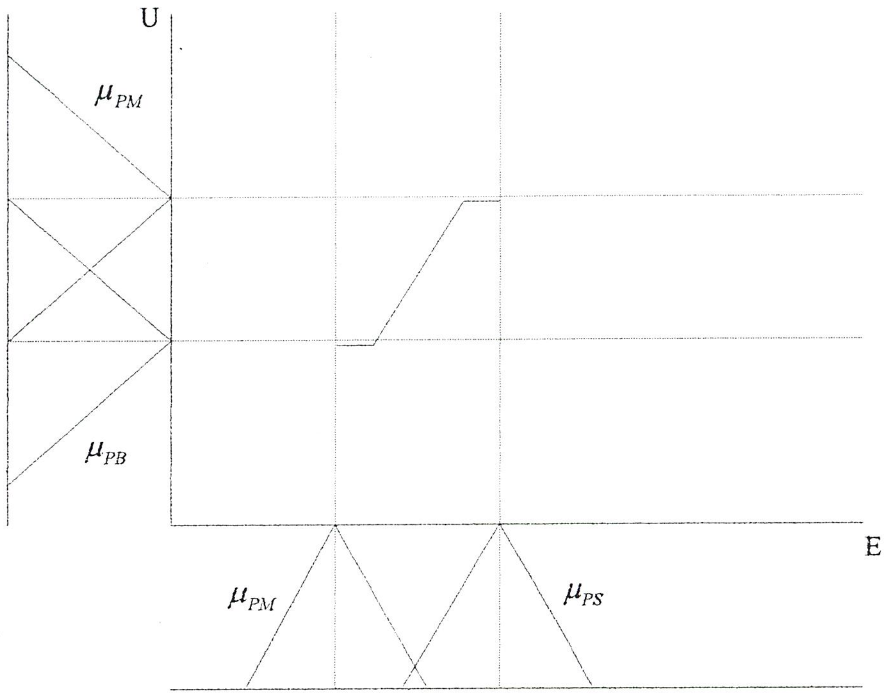


รูปที่ 2.4 แสดงความกว้างของจุดตัดทั้ง 2 ฟังก์ชัน

จะเห็นว่าความต่อเนื่อง แสดงดังรูปที่ 2.5 และไม่ต่อเนื่องดังจากรูปที่ 2.6 ของค่าเอาต์พุท จากการกำหนดระยะของจุดตัดและความกว้างทางซ้าย และทางขวาของฟังก์ชันความเป็นสมาชิก



รูปที่ 2.5 แสดงต่อเนื่องของเอาต์พุท



รูปที่ 2.6 แสดงไม่ต่อเนื่องของเอาต์พุท

2.2 ทฤษฎี Morphology

มอร์โฟโลยีเป็นทฤษฎีหนึ่งที่เกิดจากทฤษฎีของคณิตศาสตร์ที่จัดอยู่ในสาขาพีชคณิต โดยจะเป็นการศึกษารูปแบบและโครงสร้างของวัตถุ โดยในโครงงานนี้วัตถุที่นำมาใช้จะเป็นรูปภาพดิจิตอล ซึ่งมีโครงสร้างของทฤษฎีเซตมากำหนดเป็นเงื่อนไขของแต่ละตัวดำเนินการ โดยแรกๆ รูปภาพที่นำมาประมวลผลมีระดับค่าความเป็นสมาชิกอยู่ 2 ระดับคือ 0 กับ 1 หรือสีดำกับสีขาว โดยจะพิจารณามองรูปภาพเป็นข้อมูลเซต ถูกประมวลผลกับโครงสร้างพื้นฐาน (Structuring element :SE) ที่เป็นเซตอีกเซตหนึ่ง ซึ่งสามารถกำหนดค่าความเป็นสมาชิกได้ 2 ระดับคือ 0 และ 1 และสามารถกำหนดขนาดหรือมิติได้อย่างอิสระ

ทฤษฎีมอร์โฟโลยีเกิดจากการนำฟังก์ชันมากำหนดเงื่อนไขเพื่อสร้างเป็นตัวดำเนินการ จะมีฟังก์ชันที่นำมาใช้และจะกล่าวถึงหลักการและรายละเอียดแยกออกเป็นข้อๆ ดังต่อไปนี้

2.2.1 การทรานสเลชัน (Translation)

เป็นการย้ายฟัซซี่เซตไปที่ตำแหน่งที่ต้องการ โดยใช้จุดกำเนิด (0,0) เป็นตัวอ้างอิงการทรานสเลชันเซต A ที่ตำแหน่ง v ใดๆ เขียนเป็นสัญลักษณ์คือ $\Im(A;v)$

สามารถเขียนเป็นสมการได้ดังนี้

$$\mu_{\Im(A;v)}(x) = \mu_A(x-v) \tag{2.1}$$

เพื่อช่วยความเข้าใจมากขึ้น จะยกตัวอย่างประกอบในการอธิบาย

ตัวอย่างที่ 2.1 การทรานสเลชัน

กำหนด $$A = \begin{pmatrix} 0.2 & 0.5 \\ 0.3 & 0.3 \end{pmatrix}_{0,1}$$

$$\Im(A;1,1) = \begin{pmatrix} 0.2 & 0.5 \\ 0.3 & 0.3 \end{pmatrix}_{1,2}$$

จากตัวอย่างจะเห็นว่าสมาชิกของฟัซซี่เซต A ยังเหมือนเดิม แต่เมื่อทำการทรานสเลชันที่ 1,1 จะย้ายตำแหน่งจุดกำเนิดเดิม ไปที่ตำแหน่ง 1,1

หมายเหตุ พิกัดที่ห้อยที่ฟัซซี่เซตคือการบอกตำแหน่งของค่าความเป็นสมาชิกตัวแรก

2.2.2 การรีเฟลคชัน (The reflection)

การรีเฟลคชันเซต A ในฟัซซี่เซตจะใช้สัญลักษณ์คือ -A เขียนเป็นสมการได้ดังนี้

$$\mu_{-A}(x) = \mu_A(-x) \tag{2.2}$$

หลักการหาการรีเฟลคชันคือ นำเซตที่ต้องการมาทำการหมุนทั้งแนวแกน x และแกน y หรือหมุนทั้งแถวและหลัก โดยจะเอาจุดกำเนิด (0,0) เป็นจุดอ้างอิงที่ใช้หมุน สามารถพิจารณาได้จากตัวอย่าง ดังต่อไปนี้

ตัวอย่างที่ 2.2 การรีเฟลคชัน

$$\text{กำหนด} \quad A = \begin{pmatrix} 0.1 & 0.4 & 0.7 \\ 0.2 & 0.5 & 0.8 \\ 0.3 & \underrightarrow{0.6} & 0.9 \end{pmatrix}_{0,0}$$

1. นำเซต A มาหมุนในแกน X

$$= \begin{pmatrix} 0.7 & 0.4 & 0.1 \\ 0.8 & 0.5 & 0.2 \\ 0.9 & \underrightarrow{0.6} & 0.3 \end{pmatrix}_{0,-2}$$

2.นำผลที่ได้หมุนจากแกน X มาหมุนในแกน Y

$$\therefore \mu_{\underset{-}{A}}(-x) = \begin{pmatrix} 0.9 & 0.6 & 0.3 \\ 0.8 & 0.5 & 0.2 \\ 0.7 & \underrightarrow{0.4} & 0.1 \end{pmatrix}_{-2,2}$$

จากตัวอย่างจะเห็นว่าการรีเฟลคชัน จะทำให้มีการสลับตำแหน่งค่าความเป็นสมาชิกของเซต A ที่เกิดจากการหมุน แต่ค่าความเป็นสมาชิกที่จุดกำเนิดยังคงค่าเดิมจากตัวอย่างค่าที่จุดกำเนิดยังมีค่าความเป็นสมาชิกเป็น 0.1 โดยจะหมุนหลักหรือแถวก่อนก็ได้

2.2.3 การโบลดยูเนียน (Bold Union)

จะเกิดจากการนำเซต 2 เซต มากระทำกัน เขียนเป็นสัญลักษณ์ได้คือ $A \Delta B$ ซึ่งสามารถเขียนเป็นสมการได้ดังนี้

$$\mu_{A\Delta B}(x) = \min[1, \mu_A(x) + \mu_B(x)] \tag{2.3}$$

ตัวอย่างที่ 2.3 การ bold union

$$\text{กำหนด } A = \begin{pmatrix} 0.1 & 0.3 \\ 0.2 & 0.4 \end{pmatrix}_{(0,0)} \quad B = \begin{pmatrix} 0.2 & 0.6 \\ 0.4 & 0.8 \end{pmatrix}_{(0,0)}$$

จากสูตร $\mu_{A\Delta B}(x) = \min[1, \mu_A(x) + \mu_B(x)]$ นำค่าที่พิกัดต่างๆมาคำนวณตามสูตรจะได้

ที่ตำแหน่ง 0,0 จะได้ $\mu_{A\Delta B}(x) = \min\{\min[1, 0.2+0.1], \min[1,0.6+0.3], \min[1,0.4+0.2], \min[1,0.8+0.4]\}$

$= \min\{0.3,0.9,0.6,1.0\}$

$= 0.3$

ที่ตำแหน่ง 0,-1 จะได้ $\mu_{A\Delta B}(x) = \min\{\min[1, 0.2+0.2], \min[1,0.6+0.4], \min[1,0.4+0], \min[1,0.8+0]\}$

$= \min\{0.4,1.0,4,0.8\}$

$= 0.4$

ที่ตำแหน่ง 1,0 จะได้ $\mu_{A\Delta B}(x) = \min\{\min[1, 0.2+0.3], \min[1,0.6+0], \min[1,0.4+0.4], \min[1,0.8+0]\}$

$= \min\{0.5,0.6,0.8,0.8\}$

$= 0.5$

ที่ตำแหน่ง -1,1 จะได้ $\mu_{A\Delta B}(x) = \min\{\min[1, 0.2+0.4], \min[1,0.6+0], \min[1,0.4+0], \min[1,0.8+0]$

$= \min\{0.6,0.6,0.4,0.8\}$

$= 0.4$

$$\therefore \quad \mu_{A\Delta B}(x) = \begin{pmatrix} 0.3 & 0.5 \\ 0.4 & 0.4 \end{pmatrix}_{(0,0)}$$

จากตัวอย่างของการ bold union เป็นการนำค่าความเป็นสมาชิกของเซต โดยจะนำค่าที่จุดกำเนิด ของเซต B มากระทำทุกๆ พิกัดหรือตำแหน่งของเซต A โดยนำมารวมกันและนำค่าที่

ได้แต่ละค่ามาเปรียบเทียบหาค่าความเป็นสมาชิกที่น้อยที่สุดจะได้เป็นค่าของตำแหน่งนั้นๆ หาก ค่าเกิน 1.0 จะปัดลงให้เท่ากับ 1.0 ค่าที่อยู่นอกขอบเขตของเซตจะมีค่าเป็น 0 การ bold union มีหลักการเหมือนกับการยูเนียนในทฤษฎีเซต

2.2.4 การโบลดอินเตอร์เซ็กชั่น (Bold Intersection)

เกิดจากการนำเซต 2 เซต มากระทำ เขียนเป็นสัญลักษณ์ได้คือ $A \nabla B$ ซึ่งสามารถเขียนเป็นสมการได้ดังนี้

$$\mu_{A\nabla B}(x) = \max[0, \mu_A(x) + \mu_B(x) - 1]$$

หลักการ bold intersection ในฟัซซี่เซตมีหลักการเหมือนกับการอินเตอร์เซ็กชั่นในทฤษฎีของเซตทั่วไป

ตัวอย่างที่ 2.4 การ bold intersection

กำหนด $A = \begin{pmatrix} 0.5 & 0.1 \\ 0.6 & 0.2 \\ 0.1 & 0.3 \end{pmatrix}_{(1,1)}$ $B = \begin{pmatrix} 0.9 & 0.6 \\ 0.8 & 0.2 \end{pmatrix}_{(1,0)}$

จากสูตร $\mu_{A\nabla B}(x) = \max[0, \mu_A(x) + \mu_B(x) - 1]$ นำค่าที่พิกัดต่างๆมาคำนวณจะได้

ที่ 1,1 $\mu_{A\nabla B}(x) = \max\{\max[0, 0.6+0.5-1], \max[0,0.9+0-1], \max[0,0.8+0-1], \max[0,0.2+0.6-1]\}$

$= \max\{0.1,0,0,0\}$

$= 0.1$

ที่ 1,0 $\mu_{A\nabla B}(x) = \max\{\max[0,0.6+0.6-1], \max[0,0.9+0-1], \max[0,0.8+0-1], \max[0,0.2+0.1-1]\}$

$= \max\{0.2,0,0,0\}$

$= 0.2$

ที่ 1,-1 $\mu_{A\nabla B}(x) = \max\{\max[0, 0.6+0.1-1], \max[0,0.9+0-1], \max[0,0.8+0-1], \max[0,0.2+0-1]\}$

$= \max\{0,0,0,0\}$

$= 0$

ที่ 2,1 $\mu_{A\nabla B}$ (x) = max{max [0 , 0.6+0.1−1],max[0,0.9+0.5-1],max[0,0.8+0.6-1], max[0,0.2+0.2-1]}

= max{0,0.4,0.4,0}

= 0.4

ที่ 2,0 $\mu_{A\nabla B}$ (x) = max{max [0 , 0.6+0.2−1],max[0,0.9+0.6-1],max[0,0.8+0.1-1], max[0,0.2+0.3-1]}

= max{0,0.5,0,0}

= 0.5

ที่ 2,-1 $\mu_{A\nabla B}$ (x) = max{max [0 , 0.6+0.3−1],max[0,0.9+0.1-1],max[0,0.8+0-1], max[0,0.2+0-1]}

= max{0,0,0,0}

= 0

$$\therefore \quad \mu_{A\nabla B}(x) = \begin{pmatrix} 0.1 & 0.4 \\ 0.2 & 0.5 \\ 0 & 0 \end{pmatrix}_{(1,1)}$$

จากตัวอย่างของการ bold union และ bold intersection จะเห็นว่ามิติของเซต A และเซต B ไม่เท่ากัน แต่ผลลัพธ์พิจารณาเซตหลักมีขนาดของมิติเท่ากับเซต A นั้นเอง

2.2.5 ฟังก์ชันดัชนี (Index function)

เป็นฟังก์ชันที่ช่วยในการหาตัวดำเนินการเบื้องต้น ที่จะนำไปใช้หาตัวดำเนินการต่างๆ ของมอร์โฟโลยี (Mophology) กำหนดได้คือ

$$\vartheta : 2^U \times 2^U \rightarrow \{0,1\}$$

โดย

$$\vartheta(a,B) = \begin{cases} 1 \ldots\ldots if..A \subseteq B \\ 0 \ldots\ldots otherwise \end{cases} \qquad (2.4)$$

สูตรหาฟังก์ชันดัชนี หาได้ดังนี้

$$\vartheta(A,B) = \inf_{x\in A} \mu_B(x)$$
$$= \min[\inf_{x\in A} \mu B(x), \inf_{x\notin A} 1] \quad (2.5)$$
$$= \inf_{x\in u} \mu_{A^c \Delta B}(x)$$

จะเห็นว่าการหาได้ ฟังก์ชันดัชนี จะใช้ ฟังก์ชันที่กล่าวมาตามข้างต้น
คุณสมบัติของฟังก์ชันดัชนี

$$x \in A \Rightarrow \mu_{A^c \Delta B}(x) = \mu_B(x)$$
$$x \notin A \Rightarrow \mu_{A^c \Delta B}(x) = 1$$

เราสามารถนำหลักการของฟังก์ชันดัชนีมาหาตัวดำเนินการเบื้องต้นของมอร์โฟโลยี โดยจะนำฟังก์ชันที่กล่าวมาข้างต้นมาใช้งานร่วมกัน จะได้ดังนี้คือ

สมการของตัวดำเนินการ Erosion

$$\mu_{A\theta B}(x) = \vartheta(\Im(B;x), A) \quad (2.6)$$

สมการของตัวดำเนินการ Dilation

$$\mu_{A\oplus B}(x) = 1 - \vartheta(\Im(-B;x), A^C) \quad (2.7)$$

หาก

$$\vartheta^C(A,B) = 1 - \vartheta(A,B)$$
$$= \sup_{x\in u} \mu_{A \Delta B^c}(x) \quad (2.8)$$

ตัวอย่างที่ 2.5 การหาฟังก์ชันดัชนี

กำหนด $A = \begin{pmatrix} 0.2 & 0.9 \\ 0.5 & 0.1 \\ 0.7 & 0.5 \end{pmatrix}_{0,0}$ $B = \begin{pmatrix} 0.2 & 0.4 \\ 0.3 & 0.5 \end{pmatrix}_{0,0}$

สูตร $\vartheta(A,B) = \inf \mu_{A^c \Delta B}(x)$

$$A^C = \begin{pmatrix} 0.8 & 0.1 \\ 0.5 & 0.9 \\ 0.3 & .0.5 \end{pmatrix}_{0,0}$$

$A^C \Delta B = \min[1, \mu_{A^c} + \mu_B]$ นำค่าความเป็นสมาชิกที่พิกัดต่างๆ มาคำนวณ

ที่ตำแหน่ง 0,0 $A^C \Delta B = \min\{\min[1,0.8+0.2],\min[1,0.1+0.4],\min[1,0.5+0.3],\min[1,0.9+0.5]$

$= \min\{1,1,0.8,1\}$

$= 0.8$

ที่ตำแหน่ง 0,-1 $A^C \Delta B = \min\{\min[1,0.5+0.2],\min[1,0.9+0.4],\min[1,0.3+0.3],\min[1,0.5+0.5]$

$= \min\{0.7,1,0.6,1\}$

$= 0.6$

ที่ตำแหน่ง 0,-2 $A^C \Delta B = \min\{\min[1,0.3+0.2],\min[1,0.5+0.4],\min[1,0.0+0.3],\min[1,0.0+0.5]$

$= \min\{0.5,0.9,0.3,0.5\}$

$= 0.3$

ที่ตำแหน่ง 1,0 $A^C \Delta B = \min\{\min[1,0.1+0.2],\min[1,0.9+0.4],\min[1,0.0+0.3],\min[1,0.0+0.5]$

$= \min\ \{0.3,1,0.3,0.5\}$

$= 0.3$

ที่ตำแหน่ง 1,-1 $A^C \Delta B = \min\{\min[1,0.9+0.2],\min[1,0.5+0.4],\min[1,0.0+0.3],\min[1,0.0+0.5]$

$= \min\{1,0.9,0.3,0.5\}$

$= 0.3$

ที่ตำแหน่ง 1,-2 $A^C \Delta B = \min\{\min[1,0.5+0.2],\min[1,0.0+0.4],\min[1,0.0+0.3],\min[1,0.0+0.5]$

$= \min\{0.7,0.4,0.3,0.5\}$

$= 0.3$

จะได้ $$\mu_{A^c \Delta B}(x) = \begin{pmatrix} 0.8 & 0.3 \\ 0.6 & 0.3 \\ 0.3 & 0.3 \end{pmatrix}_{0,0}$$

$$\therefore \ \vartheta(A,B) = \inf_{x \in u} \begin{pmatrix} 0.8 & 0.3 \\ 0.6 & 0.3 \\ 0.3 & 0.3 \end{pmatrix}_{0,0}$$

$$= 0.3$$

2.3 ไบนารี่มอร์โฟโลยี (Binary Morphology)

จากหัวข้อที่ผ่านมาจะเป็นการกล่าวถึงหลักการของฟังก์ชันต่าง ๆ ที่นำมาเป็นเงื่อนไขในสมการของแต่ละตัวดำเนินการของทฤษฎีมอร์โฟโลยีที่แตกต่างออกกันไป ซึ่งในหัวข้อนี้จะเป็นการศึกษาถึงหลักการ และการนำตัวดำเนินการต่าง ๆ ของทฤษฎีมอร์โฟโลยีนำมาทำการวิเคราะห์ภาพที่เป็นไบนารี่ ซึ่งเพื่อให้ง่ายต่อการทำความเข้าใจ

ตัวดำเนินการของไบนารี่มอร์โฟโลยีที่ต้องทำการศึกษาได้แก่ Erosion, Dilation, Opening, Closing, Skeleton, Hit or Miss, Thinning และ Thickening รายละเอียดและหลักการมีดังนี้

2.3.1 การอีโรชั่น (Erosion)

เป็นตัวดำเนินการที่มีผลทำให้ขนาดของภาพในส่วนที่เป็นสีดำหรือมีค่าความเป็นสมาชิกที่เท่ากับ 1 ถูกลดทอนเมื่อมีการประมวลผล ลักษณะการลดทอนจะขึ้นอยู่กับการกำหนดค่าสมาชิกและการกำหนดตำแหน่งจุดออริจิลของ SE เขียนแทนด้วยสัญลักษณ์ A Θ B สามารถเขียนเป็นสมการได้คือ

$$A \,\Theta\, B = \{\, x : \Im(B;x) \subseteq A \,\} \tag{2.9}$$

กำหนด เซต A เป็นเซตของรูปภาพ

เซต B เป็นเซตของ SE

จากสมการมีหลักการคือนำ SE เลื่อนไปกระทำทุกๆ ตำแหน่งของรูปภาพ หากค่าความเป็นสมาชิกของรูปภาพกับ SE ดำเนินการสับเซตจะทำการเปลี่ยนค่าความเป็นสมาชิกตำแหน่งที่จุดกำเนิดของ SE ทับอยู่เป็น 1

2.3.4 การโคลสซิ่ง (Closing).

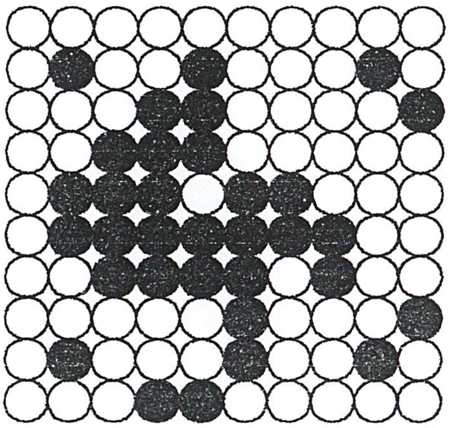
เป็นตัวดำเนินการที่เกิดได้จากการนำ SE มากระทำ Dilation แล้วนำผลลัพธ์ที่ได้มากระทำ Erosion กับ SE อีกครั้ง แทนด้วยสัญลักษณ์คือ $A \bullet B$ สามารถเขียนสมการได้ดังนี้

$$A \bullet B = (A \oplus B) \Theta B \qquad (2.13)$$

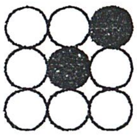
หรือ

$$A \bullet B = \{x : \forall y, x \in \Im(B\,;y) \Rightarrow \Im(B;y) \cap A \neq \phi\} \qquad (2.14)$$

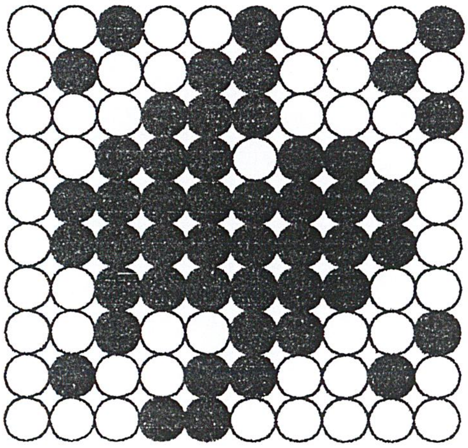
ตัวอย่างที่ 2.6 การ Erosion, Dilation, Openig และ Closing



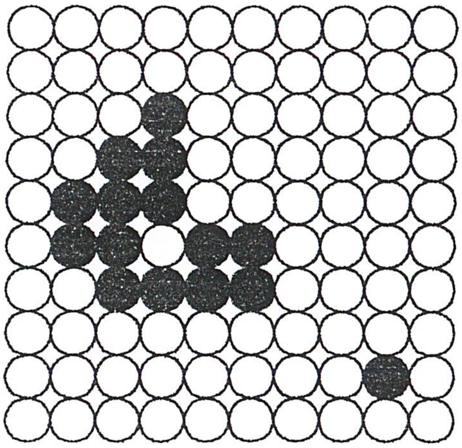
SET A



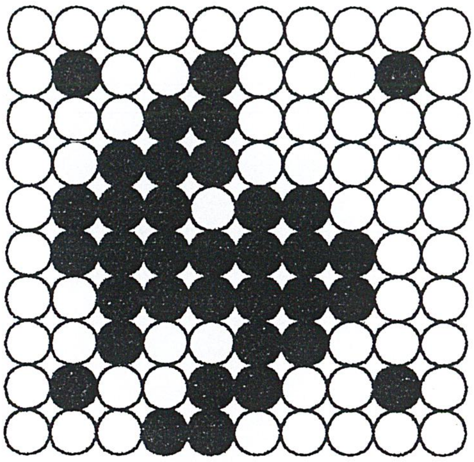
B .



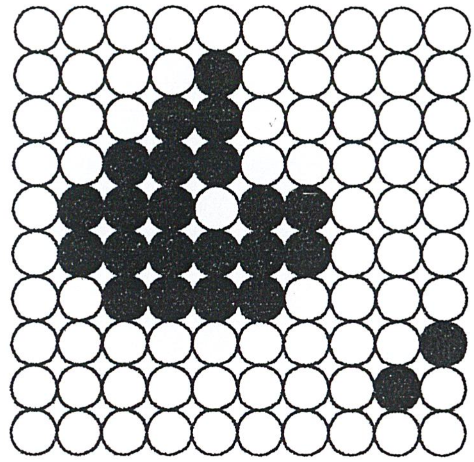
Ä ⊕ B



Ä ΘB



$(\ddot{A} \Theta B) \oplus B$



$(\ddot{A} \oplus B) \Theta B$

รูปที่ 2.7 แสดงตัวอย่างการดำเนินการ Erosion, Dilation, Opening และ Closing

การประมวลผลด้วยตัวดำเนินการ Closing ทำให้บางส่วนของรูปภาพที่ขาดหายกลับสมบูรณ์ใกล้เคียงเหมือนเดิม ซึ่งขึ้นอยู่กับการกำหนดค่าความเป็นสมาชิกและการกำหนดตำแหน่งจุดออริจิลของ SE

2.3.5 การสคีรีตอน (Skeleton)

เป็นตัวดำเนินการที่เกิดจากการนำตัวดำเนินการ Erosion และ Opening มาประมวลผล เขียนเป็นสัญลักษณ์คือ S(A) สามารถเขียนอยู่ในรูปสมการได้คือ

$$S(A) = \bigcup Sk(A) \tag{2.15}$$

และ

$$Sk(A) = \bigcup \{ (A \Theta(A \Theta kB) \circ B kB]\} \tag{2.16}$$

กำหนด เซต A เป็นเซตของรูปภาพ และเซต B เป็น SE ที่มากระทำ Erosion จำนวน k ครั้ง จนกระทั่งเป็นผลลัพธ์ที่ได้เป็นเซตว่าง มีรูปสมการดังนี้

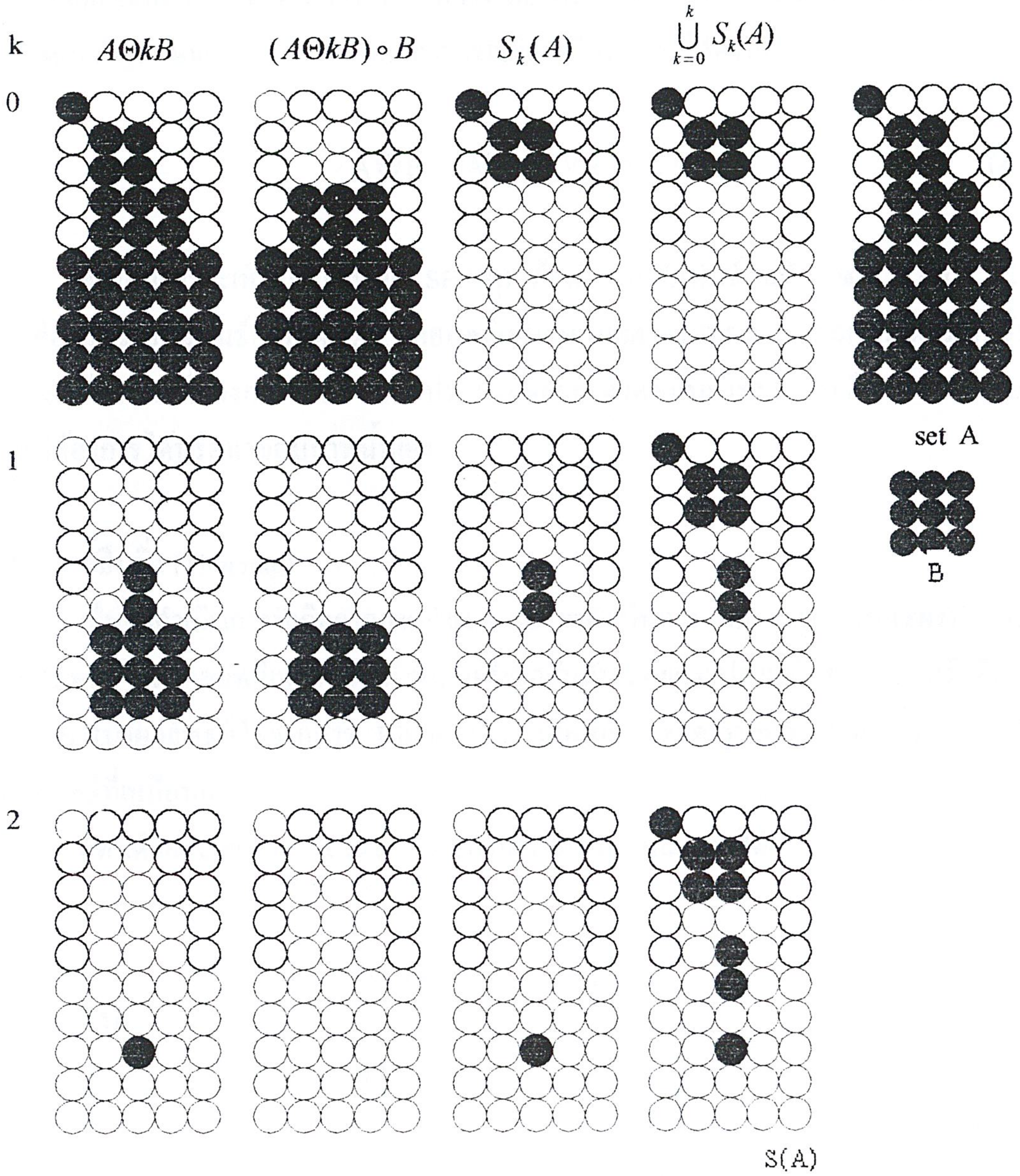
$$(A \Theta kB) = ((\ldots\ldots(A \Theta kB) \circ B) \Theta B) \tag{2.17}$$

โดย

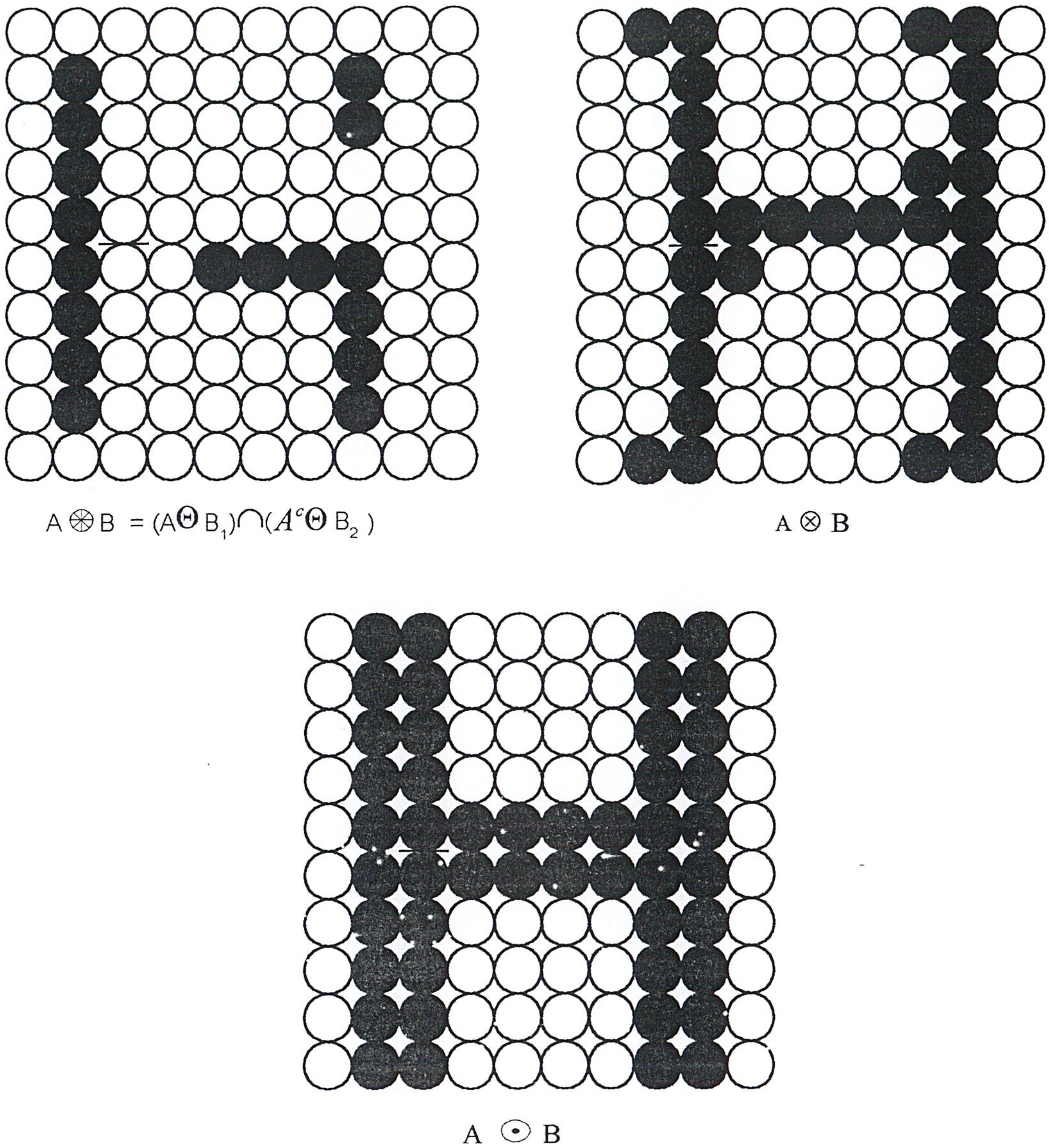
$$k = \max \{ k \mid (A \Theta kB) \neq \phi \} \tag{2.18}$$

ค่า k เป็นจำนวนครั้งสูงสุดที่เซต B กระทำ Eosion กับเซต A จนกระทั่งผลลัพธ์ที่ได้ไม่เป็นเซตว่าง

ตัวอย่างที่ 2.7 การสคีรีตอน



รูปที่ 2.8 แสดงการดำเนินการ Skeleton ใช้ค่า k = 2



รูปที่ 2.9 แสดงตัวอย่างการดำเนินการ Hit or Miss, Thinning และ Thickening

การประมวลผลแต่ละตัวดำเนินการจะมีเอกลักษณ์ในการใช้งานที่แตกต่างออกกันไป แต่มีอย่างหนึ่งที่เหมือนกันคือการกำหนดค่าความเป็นสมาชิก SE ต้องเหมาะสมและมีความสอดคล้องกับรูปภาพเพื่อประสิทธิภาพ จะเห็นว่าในบทนี้จะเป็นการประมวลผลกับรูปภาพที่เป็นสีขาวและสีดำเท่านั้น ส่วนนี้จะเป็นส่วนของหลักการพื้นฐานที่จะนำไปปรับปรุงพัฒนาไปใช้ประมวลผลกับรูปภาพเกรย์สเกลซึ่งจะกล่าวในบทต่อไป

บทที่ 3

ฟัซซี่มอร์โฟโลยี (Fuzzy Morphology)

เนื่องจากทฤษฎีมอร์โฟโลยีจะประมวลผลได้ดีกับภาพที่มีสีขาวดำ และต่อมาได้มีการพัฒนานำประมวลผลกับภาพเกรย์สเกล (Gray scale) เป็นภาพที่มีระดับสีอยู่ในช่วง [0-255] โดยนำทฤษฎีฟัซซี่มาประยุกต์ทำงานร่วมกันได้อย่างลงตัว เนื่องจากทั้งสองมีโครงสร้างที่เกิดจากทฤษฎีเซต แต่ยังคงรูปแบบของตัวดำเนินการเหมือนเดิม

มีรูปแบบของสมการดังนี้คือ

ตัวดำเนินการ (เซตที่ 1 , เซตที่ 2)

กำหนด ให้เซตที่ 1 เป็นเซตของรูปภาพ และกำหนดเซตที่ 2 เป็นเซตของ SE ประมวลด้วยตัวดำเนินการของมอร์โฟโลยี ซึ่งในที่นี้ได้แก่ Erosion, Dilation, Opening, Closing และ Hit Or Miss หลักการทฤษฎีของแต่ละตัวดำเนินการจะเหมือนกันกับมอร์โฟโลยีที่เป็นแบบไบนารีที่กล่าวมาตอนต้น จะแตกต่างกันในส่วนของการพิจารณาค่าความเป็นสมาชิกระดับข้อมูลของรูปภาพจาก จากระดับค่าความเป็นสมาชิก 0 และ 1 เปลี่ยนมาเป็นค่าความเป็นสมาชิกในช่วง [0 –1] นั่นเอง

ตัวดำเนินการมอร์โฟโลยี

3.1 ตัวดำเนินการ Erosion

การ Erosion ระหว่างเซต A กับเซต B เขียนแทนด้วยสัญลักษณ์ $\varepsilon(A,B)$ โดยมีรูปแบบสมการดังนี้

$$\begin{aligned}\mu_{\varepsilon(A,B)}(x) &= \vartheta(\Im(B;x),A) \\ &= \inf\{\, 1 \,,\, \min[\,(\,\mu_A(x)+\mu_{\Im(B^C;x)}(x)\,)\,\} \qquad (3.1)\end{aligned}$$

ตัวอย่างที่ 3.1 การ Erosion

กำหนด $A = \begin{Bmatrix} 0.2 & 1.0 & 0.8 & 0.1 \\ 0.3 & 0.9 & 0.9 & 0.2 \\ 0.1 & 0.9 & 1.0 & 0.3 \end{Bmatrix}_{0,0}$

$B = \{0.8 \quad 0.9\}_{0,0}$

จากสูตร $\mu_{\varepsilon(A,B)}(x) = \vartheta(\Im(B;x,A)$

$= \inf\{\,1\,,\min[\,(\mu_A(x)+\mu_{\Im(B^c;x)}(x)\,)\,\}$

ที่ตำแหน่ง (-1,0) $= \inf\{\,1\,,\min[\,(\,0+0.2\,)\,,(\,0.2+0.1\,)\,]\,\}$

$= \inf\{\,1\,,\min[\,(0.2\,)\,,(\,0.3\,)\,]\,\}$

$= \inf\{1\,,0.2\,\}$

$= 0.2$

ที่ ตำแหน่ง (0,0) $= \inf\{\,1\,,\min[\,(\,0.2+0.2\,)\,,(\,1+0.1\,)\,]\,\}$

$= \inf\{\,1\,,\min[\,(\,0.4\,)\,,(\,1.1\,)\,]\,\}$

$= \inf\{\,1\,,0.4\,\}$

$= 0.4$

ที่ตำแหน่ง (1,0) $= \inf\{\,1\,,\min[\,(\,1+0.2\,)\,,(\,0.8+0.1\,)\,]\,\}$

$= \inf\{\,1\,,\min[\,(\,1.2)\,,(\,0.9\,)\,]\,\}$

$= \inf\{\,1\,,0.9\,\}$

$= 0.9$

ที่ตำแหน่ง (2,0) $= \inf\{\,1\,,\min[\,(\,1+0.2)\,,(\,0.1+0.1\,)\,]\,\}$

$= \inf\{\,1\,,\min[\,(\,1.2)\,,(\,0.2)\,]\,\}$

$= \inf\{\,1\,,0.2\,\}$

$= 0.2$

ที่ตำแหน่ง (-1,1) = $\inf\{ 1 , \min [(0+0.2) , (0.3+0.1)] \}$

= $\inf \{ 1 , \min [(0.2) , (0.4)] \}$

= $\inf \{ 1 ,0.2 \}$

= 0.2

ที่ตำแหน่ง (0,1) = $\inf \{ 1 , \min [(0.3+0.2) , (0.9+0.1)] \}$

= $\inf \{ 1 , \min [(0.5) , (1.0)] \}$

= $\inf \{ 1 ,0.5 \}$

= 0.5

ที่ ตำแหน่ง (1,1) = $\inf \{ 1 , \min [(0.9+0.2) , (0.9+0.1)] \}$

= $\inf \{ 1 , \min [(1.1) , (1.0)] \}$

= $\inf \{ 1 ,1\}$

= 1

ที่ ตำแหน่ง (2,1) = $\inf \{ 1 , \min [(0.9+0.2) , (0.2+0.1)] \}$

= $\inf \{ 1 , \min [(1.1) , (0.3)] \}$

= $\inf \{ 1 ,0.3 \}$

= 0.3

ที่ตำแหน่ง (-1,2) = $\inf \{ 1 , \min [(0+0.2) , (0.1+0.1)] \}$

= $\inf \{ 1 , \min [(0.2) , (0.2)] \}$

= $\inf \{ 1 ,0.2 \}$

= 0.2

ที่ตำแหน่ง (0,2) = $\inf \{ 1 , \min [(0.1+0.2) , (0.9+0.1)] \}$

= $\inf \{ 1 , \min [(0.3) , (1.0)] \}$

= $\inf \{ 1 ,0.3 \}$

= 0.3

ที่ ตำแหน่ง (1,2) = $\inf \{ 1 , \min [(0.9+0.2) , (1+0.1)] \}$

= $\inf \{ 1 , \min [(1.1) , (1.1)] \}$

= $\inf \{ 1 ,1.1 \}$

= 1

ที่ตำแหน่ง (2,2) = inf { 1 , min [(1+0.2) , (0.3+0.1)] }

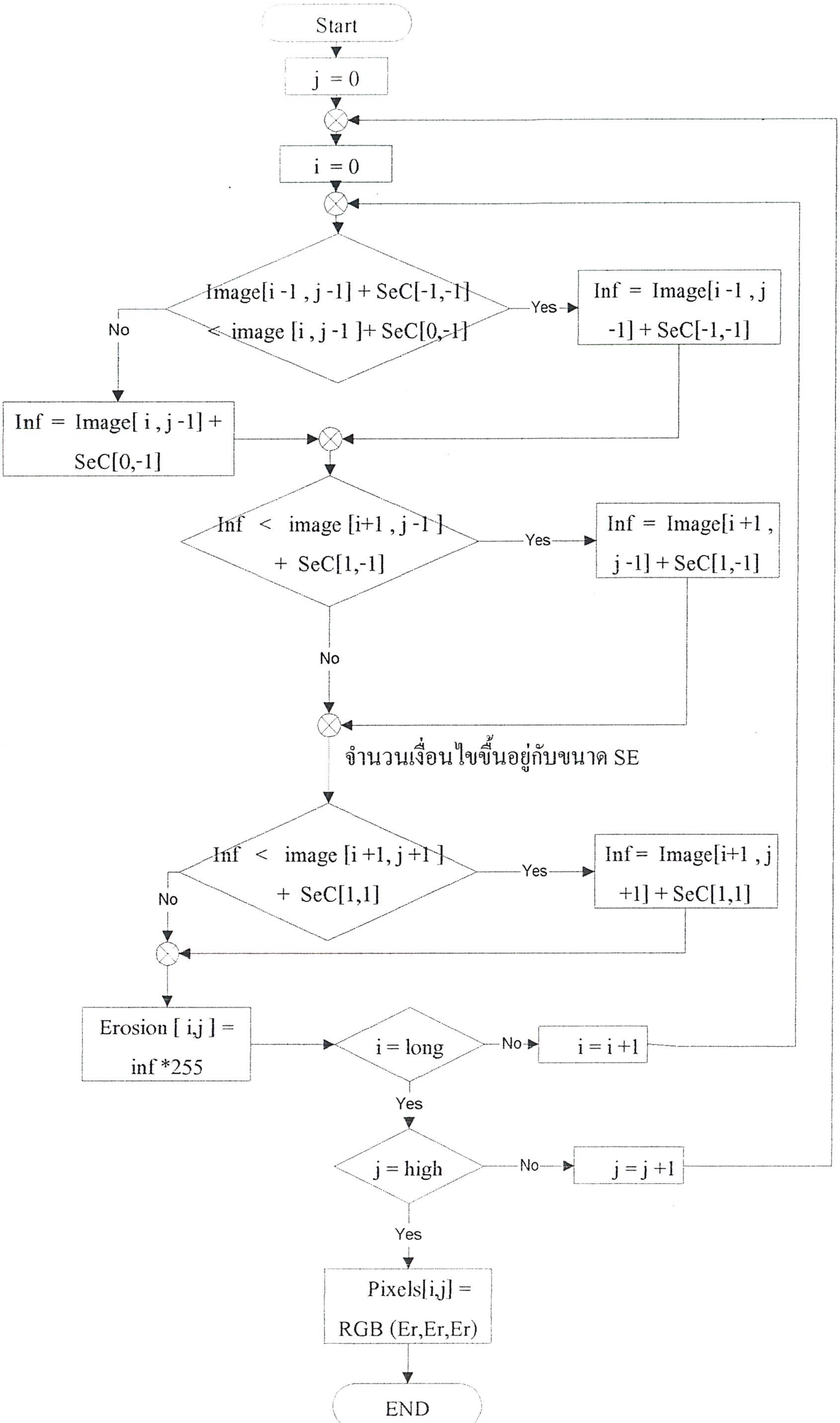
= inf { 1 , min [(1.2) , (0.4)] }

= inf { 1 ,0.4 }

= 0.4

$$\therefore\ \mu_{\varepsilon(A,B)}(x) = \begin{Bmatrix} 0.2 & 0.4 & 0.9 & 0.2 \\ 0.2 & 0.5 & 1.0 & 0.4 \\ 0.2 & 0.3 & 1.0 & 0.4 \end{Bmatrix}_{-1,0}$$

ส่วนของโปรแกรมที่ออกแบบเพื่อใช้ในการทำงานตามสมการของตัวดำเนินการ Erosion ซึ่งจะอธิบายโดยใช้แผนผังการทำงานดังรูปที่ 3.1 โดยจะดึงค่าของระดับความเป็นสมาชิกในแต่ละจุดมาใช้ โดยเก็บไว้ในตัวแปร Image และนำค่าระดับความเป็นสมาชิกแต่ละจุดมาบวกกับค่าของ SE ที่กำหนดไว้แต่ละจุดดังแสดงในแผนผัง และนำค่าที่ได้ที่ได้จากการคำนวณ มาเปรียบเทียบกันเพื่อหาค่า Inf หรือ ค่าต่ำสุด และนำค่าที่ได้ ไปคูณกับ 255 เพื่อทำการหาค่าความเข้มของสีระดับ gray scale ซึ่งมีระดับความเข้มตั้งแต่ 0 ถึง 255 ซึ่งจะเก็บไว้ในตัวแปร Erode[a,b] และ ส่งค่าที่ได้ไปยังแต่ละจุดของภาพ



รูปที่ 3.1 แผนผังแสดงการทำงานของโปรแกรมส่วน erosion

3.2 ตัวดำเนินการ Dilation

การ Dilation ระหว่างเซต A และเซต B เขียนแทนด้วยสัญลักษณ์ได้คือ $\mathcal{D}$(A,B) มีรูปแบบสมการดังนี้

$$\mu_{D(A,B)}(x) = \vartheta^{c}\ (\Im(-B;x), A^{c}) = \sup_{x\in u}\ \max[0, \mu_{\Im(-B;x)}(x) + \mu_A(x) - 1] \qquad (3.2)$$

ตัวอย่างที่ 3.2 การ Dilation

กำหนด $A = \begin{Bmatrix} 0.2 & 1.0 & 0.8 & 0.1 \\ 0.3 & 0.9 & 0.9 & 0.2 \\ 0.1 & 0.9 & 1.0 & 0.3 \end{Bmatrix}_{0,0}$

$B = \{0.8 \quad 0.9\}_{0,0}$

สูตร $\mu_{D(A,B)}(x) = \sup_{x\in u}\ \max[0, \mu_{\Im(-B;x)}(x) + \mu_A(x) - 1]$

ที่ตำแหน่ง (0,0) = sup max [0 , (0.9 + 0 -1), (0.8+0.2-1)]
= sup max [0 , -0.1, 0]
= 0

ที่ตำแหน่ง (1,0) = sup max [0 , (0.9 + 0.2 -1), (0.8+1-1)]
= sup max [0 , 0.1, 0.8]
= 0.8

ที่ตำแหน่ง (2,0) = sup max [0 , (0.9 + 1 -1), (0.8+0.8-1)]
= sup max [0 , 0.9, 0.6]
= 0.9

ที่ตำแหน่ง (3,0) = sup max [0 , (0.9 + 0.8 -1), (0.8+0.1-1)]
= sup max [0 , 0.7,- 0.1]
= 0.7

ที่ตำแหน่ง (4,0) = sup max [0 , (0.9 + 0.1 -1), (0.8+0-1)]

= sup max [0 , 0, -0.1-]

= 0

ที่ตำแหน่ง (0,1) = sup max [0 , (0.9 + 0 -1), (0.8+0.3-1)]

= sup max [0 , -0.1, 0.1]

= 0.1

ที่ตำแหน่ง (1,1) = sup max [0 , (0.9 + 0.3 -1), (0.8+0.9-1)]

= sup max [0 , 0.2, 0.7]

= 0.7

ที่ตำแหน่ง (2,1) = sup max [0 , (0.9 + 0.9 -1), (0.8+0.9-1)]

= sup max [0 , 0.8, 0.7]

= 0.8

ที่ตำแหน่ง (3,1) = sup max [0 , (0.9 + 0.9 -1), (0.8+0.2-1)]

= sup max [0 , 0.8, 0]

= 0.8

ที่ตำแหน่ง (4,1) = sup max [0 , (0.9 + 0.2 -1), (0.8+0-1)]

= sup max [0 , 0.1, 0]

= 0.1

ที่ตำแหน่ง (0,2) = sup max [0 , (0.9 + 0 -1), (0.8+0.1-1)]

= sup max [0 , -0.1, -0.1]

= 0

ที่ตำแหน่ง (1,2) = sup max [0 , (0.9 + 0.1 -1), (0.8+0.9-1)]

= sup max [0 , 0, 0.7]

= 0.7

ที่ตำแหน่ง (2,2) = sup max [0 , (0.9 + 0.9 -1), (0.8+1-1)]

= sup max [0 , 0.8, 0.8]

= 0.8

ที่ตำแหน่ง (3,2) = sup max [0 , (0.9 + 1 -1), (0.8+0.3-1)]

= sup max [0 , 0.9, 0.1]

= 0.7

ที่ตำแหน่ง (4,2) = sup max [0 , (0.9 + 0.3 -1), (0.8+0.-1)]

= sup max [0 , 0.2, -0.2]

= 0.2

$$\mu_{D(A,B)}(x) = \begin{Bmatrix} 0.0 & 0.8 & 0.9 & 0.7 & 0 \\ 0.1 & 0.7 & 0.8 & 0.8 & 0.1 \\ 0.0 & 0.7 & 0.8 & 0.9 & 0.2 \end{Bmatrix}_{0,0}$$

ส่วนของโปรแกรมที่ออกแบบเพื่อใช้ในการทำงานตามสมการของตัวดำเนินการ Dilation ซึ่งจะมีโครงสร้างของการทำงานของแผนผังคล้ายกับ โปรแกรมส่วน erosion ซึ่งจะอธิบายการทำงานของแผนผังดังรูปที่ 3.2 โดยจะดึงค่าของระดับความเป็นสมาชิกที่แต่ละจุดที่เก็บไว้ในตัวแปร Image มาทำการบวกกับค่า SE ที่กำหนดไว้ตามเงื่อนไข และนำค่าที่ได้ในแต่ละจุดนำมาเปรียบเทียบเพื่อหาค่า Sup หรือ ค่าสูงสุด และนำค่าที่ได้ไปคูณกับ 255 เพื่อทำการแปลงค่าที่ได้ให้เป็นค่าเข้มของสีระดับ gray scale ซึ่งมีระดับความเข้มตั้งแต่ 0 ถึง 255 ซึ่งจะเก็บไว้ในตัวแปร Dilate[a,b] และ ส่งค่าที่ได้กลับไปยังแต่ละจุดของภาพ

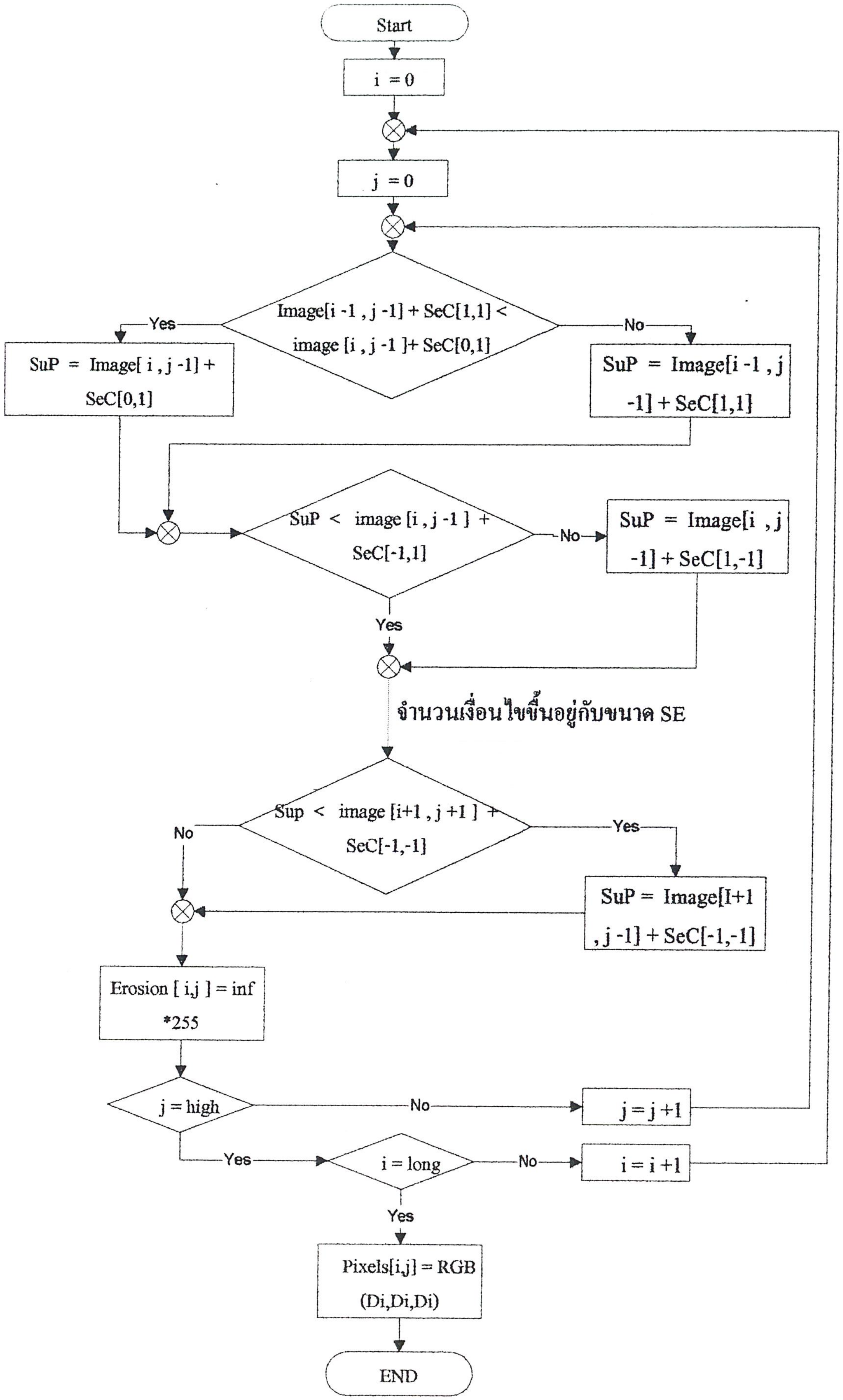
3.3 ตัวดำเนินการ Opening

เป็นตัวดำเนินการที่ได้จากการนำตัวดำเนินการ Erosion และ Dilation มากำหนดเป็นเงื่อนไข ตัวดำเนินการ Opening แทนด้วยสัญลักษณ์คือ $\mathcal{O}(A,B)$ มีรูปแบบสมการดังนี้

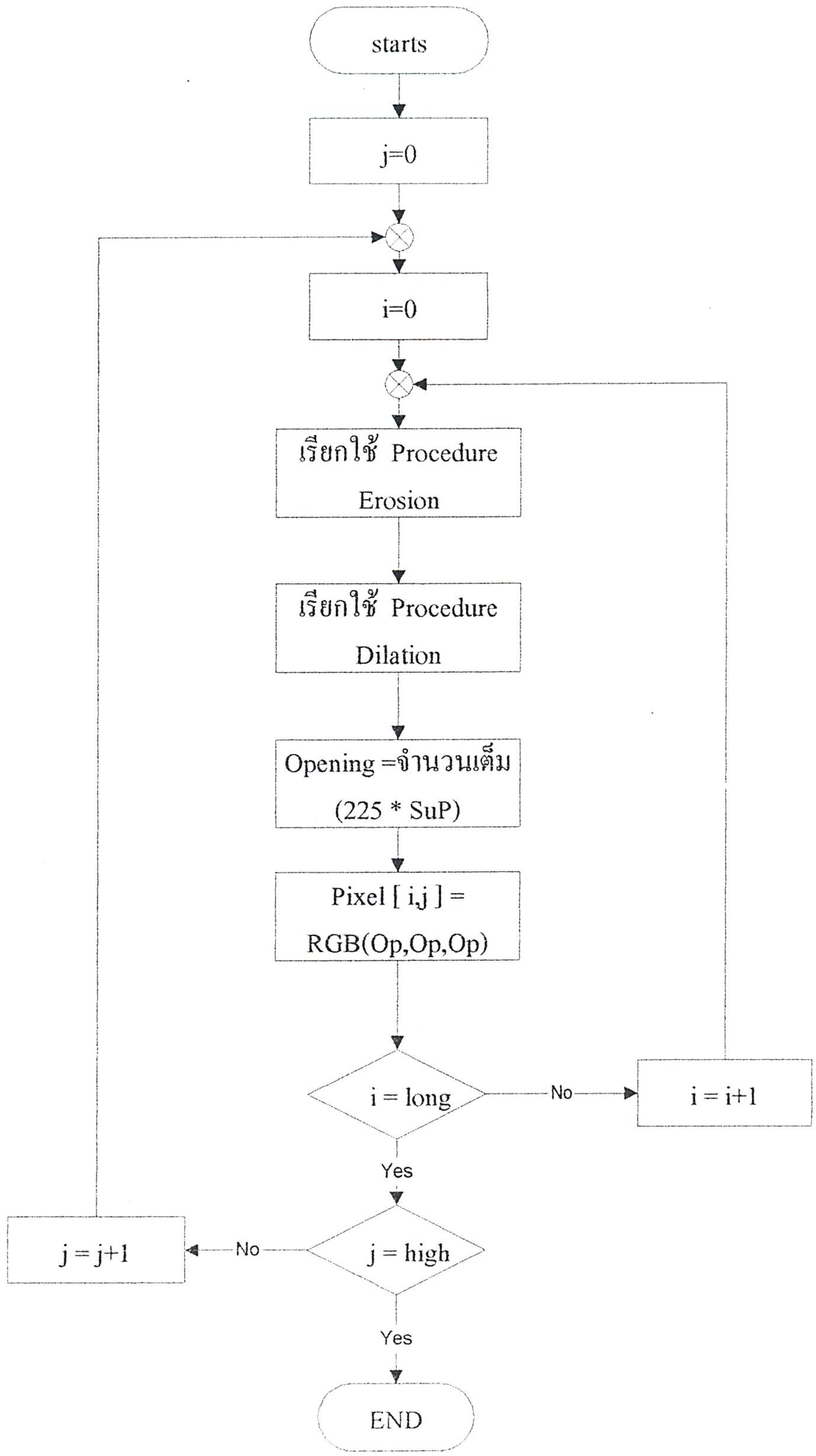
$$\mathcal{O}(A,B) = \mathcal{D}[\mathcal{E}(A,B),B] \tag{3.3}$$

เป็นตัวดำเนินการที่ใช้ในกำจัดสัญญาณรบกวนที่เกิดในภาพได้ ที่มีระดับค่าความเป็นสมาชิกสูงๆ หรือสัญญาณรบกวนที่มีค่าสูงๆ

ส่วนของโปรแกรมที่ทำการออกแบบเพื่อใช้งานตามคุณสมบัติของตัวดำเนินการ Opening คือ จะทำการดำเนินการกับรูปภาพและเมื่อได้ผลลัพธ์ออกมาจะนำไป อีกที่หนึ่งซึ่งจะได้เป็นผลลัพธ์ของ Opening ออกมา หลักการทำงานจะอธิบายได้ตามรูปที่ 3.3 ทำงานจะทำการเรียกโปรแกรมส่วนของ erosion มาทำงานก่อน และได้ผลลัพธ์ของการ Erosion แล้ว จึงเรียกใช้โปรแกรม dilation อีกที่หนึ่ง ค่าที่ได้ออกมาจะเป็นค่า Sup ของแต่ละจุดของภาพ นำค่าที่ได้ไปคูณกับ 255 เพื่อทำการแปลงค่าที่ได้ให้เป็นค่าเข้มของสีระดับ gray scale ซึ่งมีระดับความเข้มตั้งแต่ 0 ถึง 255 ซึ่งจะเก็บไว้ในตัวแปร Dilate[c,d] ส่งค่าที่ได้กลับไปยังแต่ละจุดของภาพ



รูปที่ 3.2 แผนผังแสดงการทำงานของโปรแกรมส่วน dilation



รูปที่ 3.3 แผนผังแสดงการทำงานของโปรแกรมส่วน opening

3.4 ตัวดำเนินการ Closing

ตัวดำเนินการ Closing เขียนแทนด้วยสัญลักษณ์และมีรูปแบบสมการดัง

$$\mathcal{C}(A,B) = \mathcal{E}[\mathcal{D}(A,-B),-B] \quad (2.44)$$

เป็นตัวดำเนินการที่สามารถกำจัดสัญญาณรบกวนที่มีระดับค่าความเป็นสมาชิกที่ต่ำ ๆ หรือสัญญาณรบกวนที่มีค่าต่ำๆ

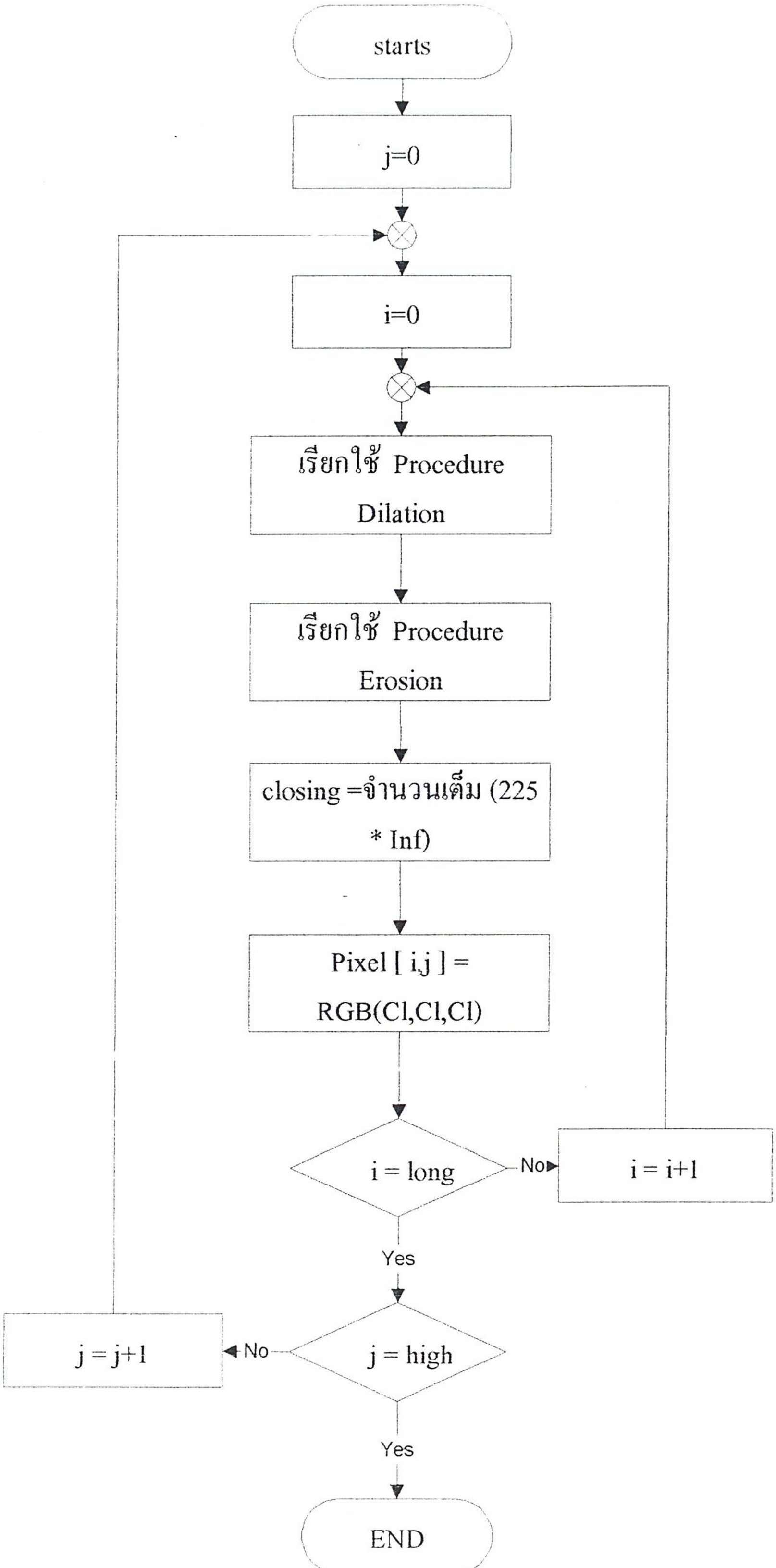
ส่วนของโปรแกรมที่ออกแบบมาทำงานตามคุณสมบัติของตัวดำเนินการ Closing ซึ่งมีหลักการทำงานคล้าย ๆ กับ โปรแกรมส่วนของ opening แต่ในตัวดำเนินการ Closing จะกระทำการ Dilation กับรูปภาพก่อน แล้วเมื่อได้ผลลัพธ์ออกมาจึงนำไปทำการ Erosion อีกที่ ซึ่งหลักการทำงานเป็นไปตาม แผนผังในรูป 3.4 จะอธิบายการทำงานโดยในขั้นแรกจะเรียกใช้ โปรแกรมในส่วนของ dilation ก่อน และเมื่อได้ผลลัพธ์ออกมาจึงเรียกใช้โปรแกรม erosion และผลลัพธ์ที่ได้ออกมาจะเป็นค่า Inf ของแต่ละจุดของภาพ นำค่าที่ได้ไปคูณกับ 255 เพื่อทำการแปลงค่าที่ได้ให้เป็นค่าเข้มของสีระดับ gray scale ซึ่งมีระดับความเข้มตั้งแต่ 0 – 255 ซึ่งจะเก็บไว้ในตัวแปร Erode[c,d]
ส่งค่าที่ได้กลับไปยังแต่ละจุดของภาพ

3.5 ตัวดำเนินการ Hit or Miss

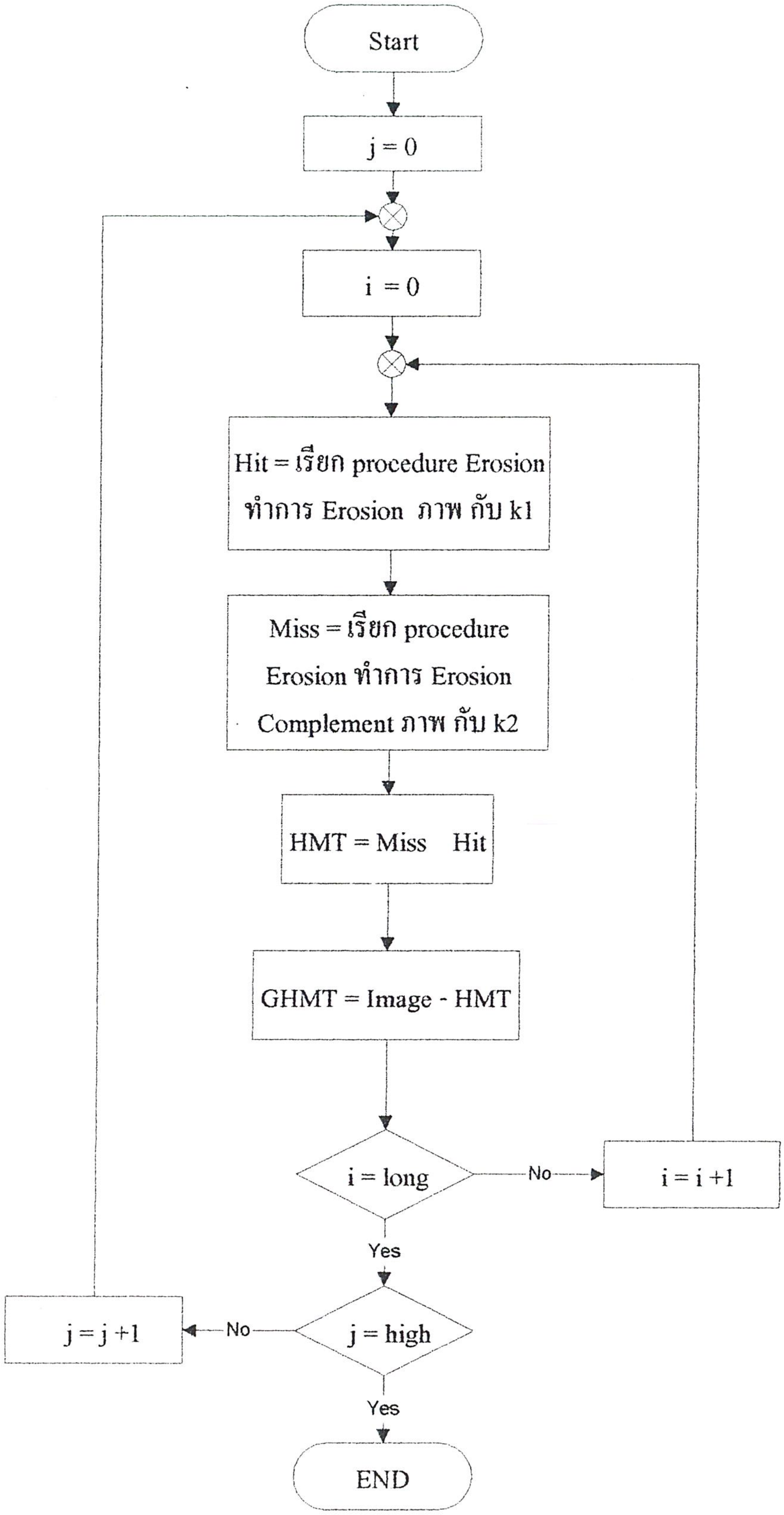
เป็นตัวดำเนินการสุดท้ายที่จะขอกล่าวและนำไปประยุกต์ใช้งานในการประมวลผลกับภาพเพื่อใช้หาขอบภาพ แทนด้วยสัญลักษณ์ $A \circledast (B_1, B_2)$ โดยมีเงื่อนไขว่า $B1 \cap B2 = \varnothing$ เขียนเป็นรูปสมการได้ดังนี้

$$A \circledast (B_1, B_2) = \mathcal{E}(A, B_1) \cap \mathcal{E}(A^C, B_2 \quad (3.4)$$

ส่วนของโปรแกรมที่ออกแบบขึ้นมาทำงานตามคุณสมบัติของ ตัวดำเนินการ Hit or Miss ซึ่งจะมีการดำเนินการสองส่วน คือ ส่วน Hit และส่วน Miss ดังแสดงในรูปที่ 3.5 การทำงานจะทำในส่วน Hit โดยจะได้จากการทำการ Erosion กับ SE ที่ได้กำหนดตามเงื่อนไขในโปรแกรมโดยค่าที่ได้จะเก็บในตัวแปร hit[a,b] และในส่วนที่สอง เป็นการ Miss ซึ่งเป็นการนำภาพ complement มาทำการ Erosion กับ SE ที่ได้กำหนดตามเงื่อนไขในโปรแกรม และเก็บไว้ในตัวแปร miss[a,b] เมื่อได้ผลลัพธ์ทั้งสองแล้วนำมาหาค่า HMT ซึ่งได้จากการนำ มาอินเตอร์เซคกันและเก็บค่าในตัวแปร hmt และนำไปหาค่า ghmt โดยการนำรูปภาพลบด้วย hmt การประมวลผลแต่ละตัวดำเนินการจะมีคุณสมบัติและการใช้งานที่แตกต่างออกกันไป



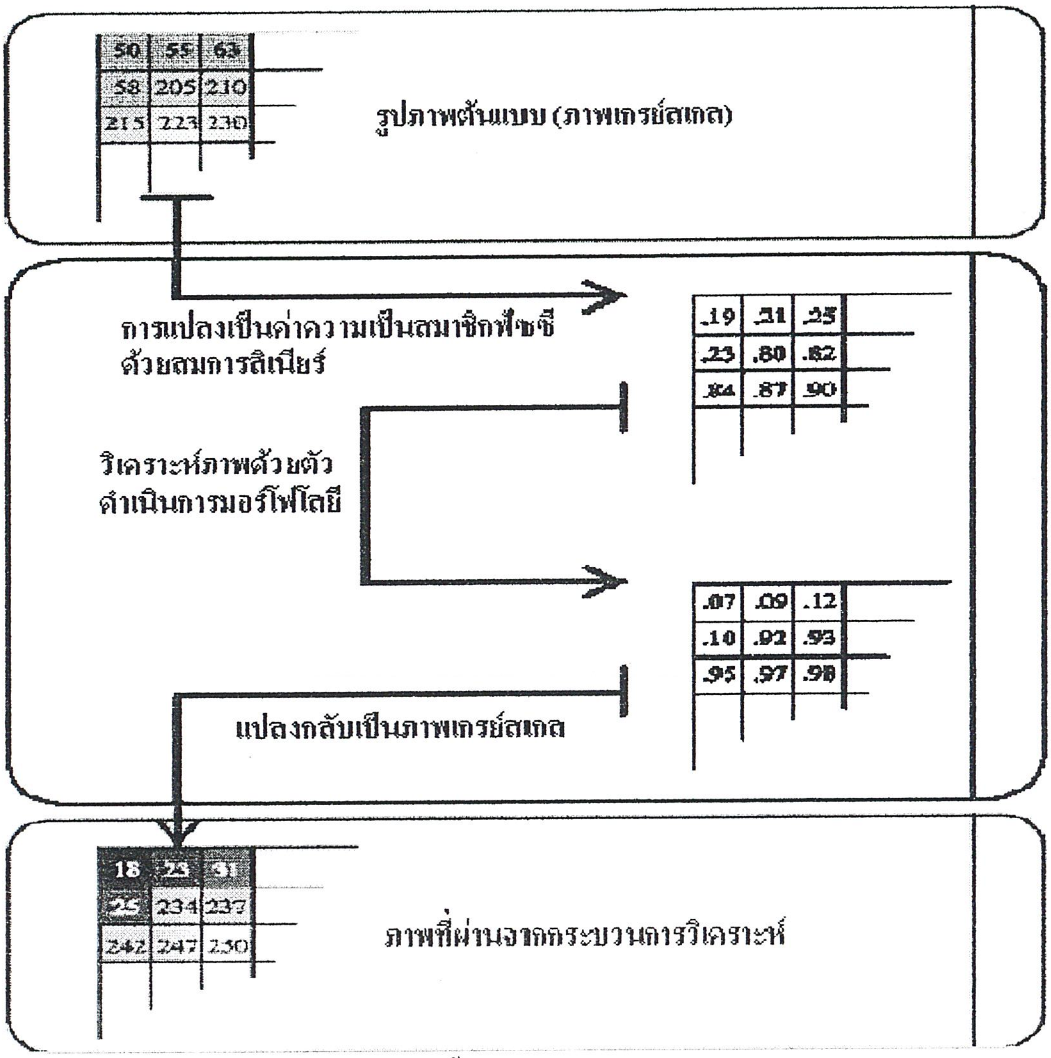
รูปที่ 3.4 แผนผังแสดงการทำงานของส่วนโปรแกรม closing



รูปที่ 3.5 แผนผังการแสดงการทำงานส่วนของโปรแกรม hit or miss

บทที่ 4
การประมวลผลภาพขั้นพื้นฐานด้วยฟัซซี่มอร์โฟโลยี

หลักการและผลการทดลอง



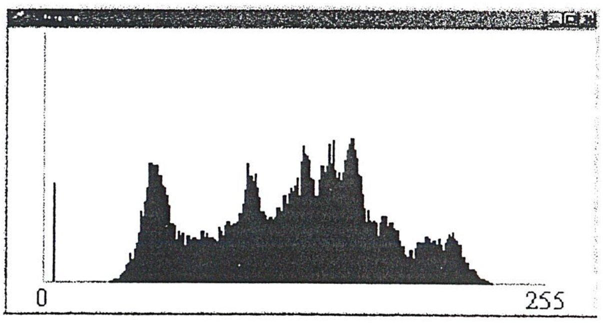
รูปที่ 4.1 แสดงขั้นตอนการทำงานของโปรแกรม

จากรูปที่ 4.1 เป็นการแสดงขั้นตอนการประมวลผลของมอร์โฟโลยีด้วยตัวดำเนินการต่างๆ ที่นำเสนอมาแล้วข้างต้น ซึ่งในบทนี้จะนำเสนอในส่วนของผลการทดลอง เริ่มจากนำภาพต้นแบบดังรูปที่ 4.2 มาทำการใส่สัญญาณรบกวนในข้อมูลภาพ ทำให้ภาพเปลี่ยนไปจากเดิมซึ่งในที่นี้เป็นสิ่งที่ไม่ต้องการ จะต้องกำจัดทำให้หายไปมากที่สุด เมื่อได้ภาพที่มีสัญญาณรบกวนแล้วต้องเอาภาพมาแปลงระดับค่าสีเป็นภาพเกรย์สเกล ซึ่งเป็นภาพที่มีค่าของระดับสี 0 - 255 แล้วนำมาแปลงค่าระดับสีเป็นค่าความเป็นสมาชิกของฟัซซี่เซตอีกครั้ง จะได้ค่าของระดับสีที่มีความเป็นสมาชิกในช่วง [0 – 1] ถึงจะนำไปเข้ากระบวนการประมวลผลด้วยตัวดำเนินการของมอร์โฟโลยีได้แก่

ได้แก่ Erosion, Dilation, Opening, Closing และ Hit or Miss ผลลัพธ์ที่ได้นำมาแปลงกลับเป็นภาพเกรย์สเกลดังเดิมหรือที่เรียกว่า Defuzzification ซึ่งเป็นกระบวนการประมาณค่าของตัวแปรเอาท์พุท โดยการอนุมานจากค่าความจริงในส่วนของ Antecedent ของแบบจำลองฟัซซี่เซต โดยค่าที่ได้จะเป็นตัวเลขจำนวนจริงที่อยู่ในโดเมนของตัวแปรเอาท์พุท และค่านี้จะเป็นค่าความคาดหวัง (Expected Value) ของตัวแปรฟัซซี่ ที่จะนำไปใช้ในการควบคุมต่อไป ในส่วนนี้การ Defuzzification ใช้หลักการนำค่าระดับสี 255 มาคูณค่าความเป็นสมาชิก ซึ่งจะเป็นการคืนเป็นรูปภาพเกรย์สเกลกลับดังเดิมหลังผ่านการประมวลผลของแต่ละตัวดำเนินการของมอร์โฟโลยี

ก่อนที่จะนำไปประมวลผลต้องมากำหนดค่าของ SE เนื่องจากเป็นส่วนที่นำไปประมวลผลกับภาพ ซึ่งเป็นเซตที่มีขนาด 5×5 หรือ 3×3 ตามความต้องการ

(ก) รูปภาพต้นแบบ LENNA



(ข) แสดงกราฟอิสโตแกรม

รูปที่ 4.2 (ก) รูปภาพต้นแบบ LENNA
(ข) แสดงกราฟฮิสโตแกรม

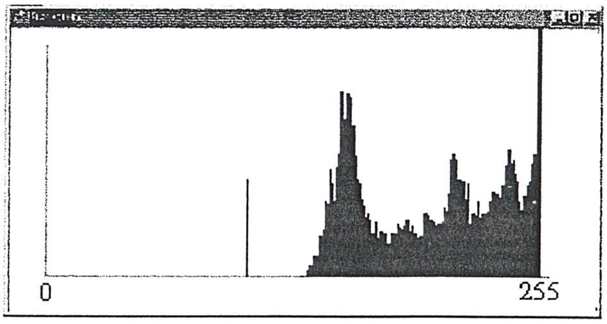
4.1 ตัวดำเนินการ Erosion

ผลที่ได้จากการประมวลผลของตัวดำเนินการ Erosion โดยการเปรียบเทียบกับรูปภาพต้นแบบที่แสดงดังรูป 4.2 (ก) จะเห็นได้ว่าเมื่อกำหนดให้ SE ขนาด 3×3 ที่มีค่า 0.6 ทั้งหมดทุกตำแหน่งดังรูปที่ 4.3 (ข) ผลที่ได้จะทำให้ภาพนั้นมีความสว่างเพิ่มขึ้นดังรูปที่ 4.3 (ก) ซึ่งสามารถพิจารณาได้จากฮิสโตแกรมดังรูปที่ 4.3 (ค) จะเห็นว่าลักษณะกราฟของฮิสโตแกรมจะมีการเลื่อนไปทางค่าสูงของข้อมูล และเมื่อกำหนดค่า SE ที่ 1.0 ทุก ๆ ตำแหน่งดังรูปที่ 4.4 (ข) ผลที่ได้จะเห็นว่ารูปภาพมีความสว่างน้อยลงกว่ารูปภาพเดิมดังรูปที่ 4.4 (ก) หากพิจารณาลักษณะของฮิสโตแกรมดังรูปที่ 4.4 (ค) จะเห็นว่าจำนวนข้อมูลที่มีค่าสูง ๆ จะมากขึ้น

(ก) รูปภาพที่ผ่านการ Erosion

0.6	0.6	0.6
0.6	0.6	0.6
0.6	0.6	0.6

(ข) แสดง SE ที่ 0.6



(ค) แสดงกราฟฮิสโตแกรม

รูปที่ 4.3 (ก) รูปภาพที่ผ่านการ Erosion

(ข) แสดง SE ที่ 0.6

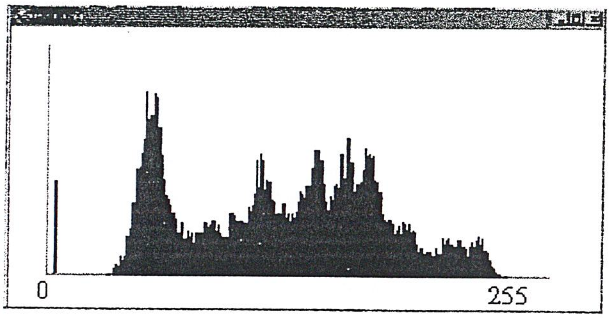
(ค) แสดงกราฟฮิสโตแกรม

สรุปได้ว่าเมื่อใช้ตัวดำเนินการ Erosion จะทำให้ความสว่างของรูปภาพจะมีการเปลี่ยนแปลง ซึ่งจะแปรผกผันกับการกำหนดค่า SE คือเมื่อกำหนดค่า SE ที่ค่าสูงจะทำให้ภาพมีความสว่างน้อยลง ตรงกันข้ามหากกำหนดค่า SE ต่ำ ๆ ภาพจะมีความสว่างมากขึ้นกว่าภาพต้นแบบ

(ก) รูปภาพที่ผ่านการ Eosion ที่ 1.0

1	1	1
1	1	1
1	1	1

(ข) กำหนด SE ที่ 1



(ค) แสดงการฮิสโตแกรม

รูปที่ 4.4 (ก) รูปภาพที่ผ่านการ Erosion
(ข) แสดง SE ที่ 1.0
(ค) แสดงกราฟฮิสโตแกรม

(ก) การหาขอบด้วย Erosion

1	1	1
1	1	1
1	1	1

(ข) กำหนด SE ที่ 1

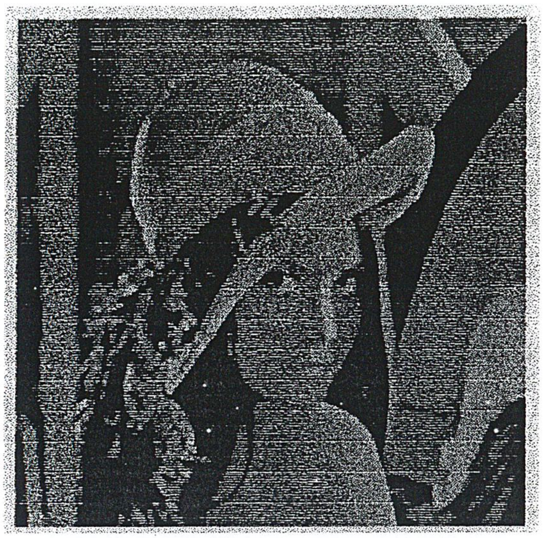
รูปที่ 4.5 (ก) การหาขอบด้วย Erosion
(ข) กำหนด SE ที่ 1

คุณสมบัติของการประมวลผล Erosion สามารถนำมาประยุกต์ใช้ในการหาขอบภาพได้ โดยใช้หลักการเอาผลค่าความเป็นสมาชิกของรูปภาพที่ผ่านการ Erosion มาลบกับค่าความเป็นสมาชิกของรูปภาพเดิมส่วนที่คงเหลืออยู่คือส่วนที่เป็นขอบภาพนั้นเองแสดงดังรูปที่ 4.5 (ก) โดยกำหนดค่าความเป็นสมาชิกที่ 1 ดังรูปที่ 4.5 (ข)

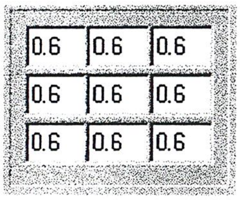
4.2 ตัวดำเนินการ Dilation

ผลที่ได้จากการประมวลผลของตัวดำเนินการ Dilation โดยการเปรียบเทียบกับรูปภาพต้นแบบจะเห็นได้ว่าเมื่อกำหนดให้ SE ขนาด 3×3 ที่มีค่า 0.6 ทั้งหมดทุกตำแหน่งแสดงดังรูปที่ 4.6 (ข) ผลที่ได้จะทำให้ภาพนั้นมีความเข้มเพิ่มขึ้นดังรูปที่ 4.6 (ก) ซึ่งสามารถพิจารณาได้จากฮิสโตแกรมแสดงดังรูปที่ 4.6 (ค) จะเห็นว่าลักษณะกราฟของฮิสโตแกรมจะมีการเลื่อนไปทางค่าต่ำของข้อมูล และเมื่อกำหนดค่า SE ที่ 1.0 ทุก ๆ ตำแหน่งดังรูปที่ 4.7 (ข) จะทำให้รูปภาพมีความเข้มลดลงดังรูปที่ 4.7 (ก) หากพิจารณาลักษณะของฮิสโตแกรมที่แสดงไว้ดังรูปที่ 4.7 (ค) จำนวนของข้อมูลที่มีค่าสูง ๆ จะมากขึ้น

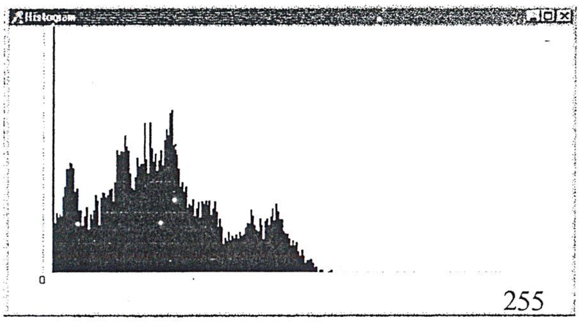
สรุปได้ว่าเมื่อใช้ตัวดำเนินการ Dilation ความเข้มของรูปภาพจะมีการเปลี่ยนแปลงความเข้มเพิ่มขึ้น ซึ่งจะแปรผกผันกับการกำหนดค่า SE คือเมื่อกำหนดค่า SE ที่ค่าสูงจะทำให้ภาพมีความเข้มลดลง ตรงกันข้ามหากกำหนดค่า SE ต่ำ ๆ ภาพจะมีความเข้มมากขึ้นกว่าภาพต้นแบบ



(ก) รูปภาพที่ผ่านการ Dilation ที่ 0.6



(ข) แสดง SE ที่ 0.6



(ค) แสดงการฮิสโตแกรม

รูปที่ 4.6 (ก) รูปภาพที่ผ่านการ Dilation

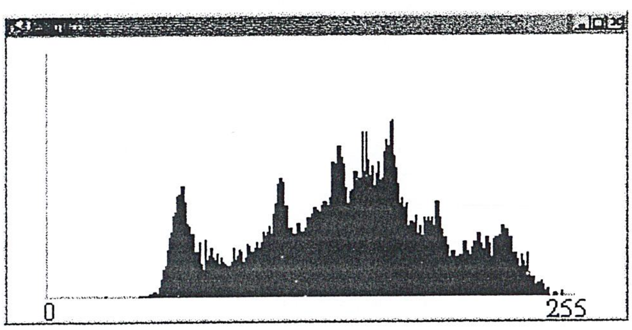
(ข) แสดง SE ที่ 0.6

(ค) แสดงกราฟฮิสโตแกรม

(ก) รูปภาพที่ผ่านการ Dilation ที่ 1.0

1	1	1
1	1	1
1	1	1

(ข) แสดง SE ที่ 1.0



(ค) แสดงการฮิสโตแกรม

รูปที่ 4.7 (ก) รูปภาพที่ผ่านการ Dilation

(ข) แสดง SE ที่ 1.0

(ค) แสดงกราฟฮิสโตแกรม

(ก) การหาขอบด้วย Dilation

1	1	1
1	1	1
1	1	1

(ข) กำหนด SE ที่ 1

รูปที่ 4.8 (ก) การหาขอบด้วย Dilation
(ข) กำหนด SE ที่ 1

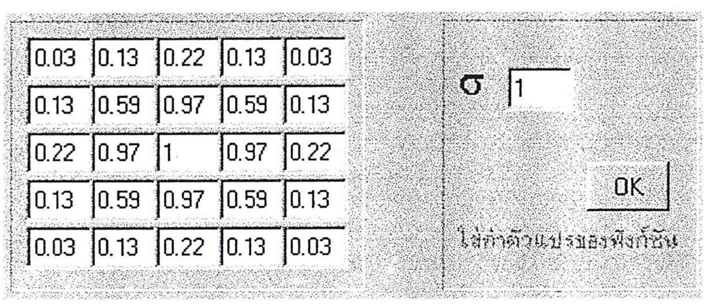
ตัวดำเนินการ Dilation ก็เป็นอีกตัวดำเนินการหนึ่งที่สามารถนำมาประมวลผลในการหาขอบภาพ ซึ่งใช้หลักการเหมือนกับการหาขอบภาพของตัวดำเนินการ Erosion ผลลัพธ์ที่ได้จากหาขอบภาพด้วยตัวดำเนินการ Dilation แสดงดังรูปที่ 4.8 (ก) และกำหนดค่าความเป็นสมาชิกของ SE ที่ 1 แสดงไว้ดังรูปที่ 4.8 (ข)

4.3 ตัวดำเนินการ Opening

เป็นตัวดำเนินการที่ใช้ในกำจัดสัญญาณรบกวนที่เกิดในข้อมูลภาพที่มีระดับค่าความเป็นสมาชิกสูงๆ หรือสัญญาณรบกวนสีขาว (White Noise) ในการประมวลผลจะนำฟังก์ชันเกาส์เซียนมาทำการคำนวณหาค่าความเป็นสมาชิกของ SE ที่เหมาะสม โดยใช้ค่าเบี่ยงเบนมาตรฐานที่ 1 แสดงดังรูปที่ 4.9 (ค) ผลลัพธ์ที่ได้จากกำจัดสัญญาณรบกวนจากรูปที่ 4.9 (ก) แสดงดังรูปที่ 4.9 (ข) ส่วนจุดที่กำจัดไม่หมดเนื่องจากเป็นจุดที่มีขนาดใหญ่กว่าขนาดของ SE สามารถกำจัดให้หมดได้โดยการขนาด SE ให้มีขนาดใหญ่กว่าขนาดของสัญญาณรบกวน

(ก) รูปภาพที่มี White noise

(ข) การ ประมวลผล Opening



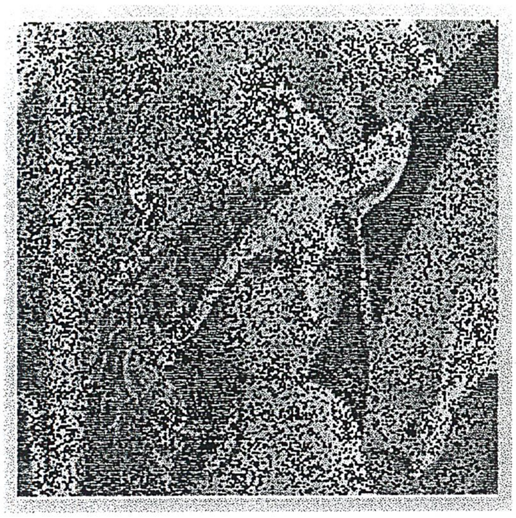
0.03	0.13	0.22	0.13	0.03
0.13	0.59	0.97	0.59	0.13
0.22	0.97	1	0.97	0.22
0.13	0.59	0.97	0.59	0.13
0.03	0.13	0.22	0.13	0.03

(ค) กำหนด SE โดยใช้ฟังก์ชันเกาส์เซียน ค่าเบี่ยงเบนมาตรฐานที่ 1

รูปที่ 4.9 (ก) ภาพที่มี White Noise
(ข) การประมวลผล Opening
(ค) กำหนด SE โดยใช้ฟังก์ชันเกาส์เซียน ค่าเบี่ยงเบนมาตรฐานที่ 1

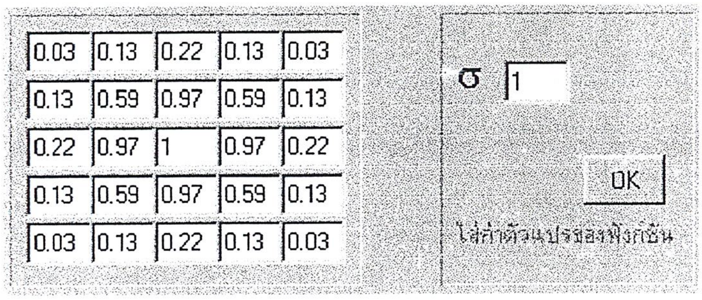
4.4 ตัวดำเนินการ Closing

เป็นตัวดำเนินการที่สามารถกำจัดสัญญาณรบกวน ที่มีระดับค่าความเป็นสมาชิกที่ต่ำๆ หรือสัญญาณรบกวนความถี่ต่ำ จากรูปที่ 4.10 (ก) เป็นภาพที่มีสัญญาณรบกวนความถี่ต่ำ ส่วนรูปที่ 4.10 (ข) เป็นภาพผลลัพธ์ที่ได้จากการประมวลผลของตัวดำเนินการ Closing โดยจะนำฟังก์ชันเกาส์เซียนหาค่าความเป็นสมาชิกของ SE โดยกำหนดค่าเบี่ยงเบนมาตราฐานที่ 1 แสดงดังรูปที่ 4.10 (ค) ซึ่งจะทำให้ได้ค่าความเป็นสมาชิกของ SE มีค่าที่เหมาะสม



(ก) ภาพที่มี Black Noise

(ข) การประมวลผล Closing



(ค) กำหนด SE โดยใช้ฟังก์ชันเกาส์เซียน ค่าเบี่ยงเบนมาตรฐานที่ 1

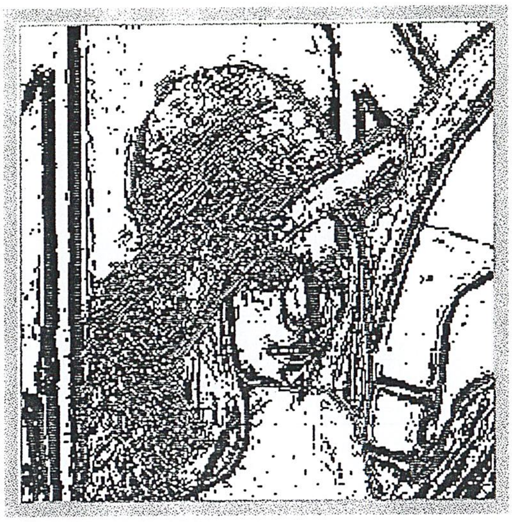
รูปที่ 4.10 (ก) ภาพที่มี Black Noise

(ข) การประมวลผล Closing

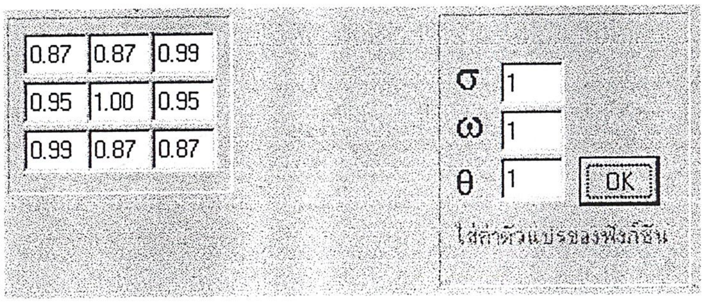
(ค) กำหนด SE โดยใช้ฟังก์ชันเกาส์เซียน ค่าเบี่ยงเบนมาตรฐานที่ 1

4.5 ตัวดำเนินการ Hit or Miss

การหาขอบภาพด้วยใช้ตัวดำเนินการ Hit or Miss สามารถหาได้ละเอียดและมีประสิทธิภาพมากน้อยเพียงใดขึ้นอยู่กับการกำหนดค่าความเป็นสมาชิกของ SE ในส่วนนี้จึงนำฟังก์ชัน Odd gabor มาคำนวณค่าความเป็นสมาชิกของ SE ที่เหมาะสม โดยกำหนด $\sigma = 1$, $\omega = 1$ และ $\theta = 1$ แสดงดังรูปที่ 4.11 (ข) ผลลัพธ์ที่ได้จากการประมวลผลแสดงดังรูปที่ 4.11 (ก) จะเห็นว่ารูปภาพที่ออกมาจะมีลายเส้นที่ละเอียดมากซึ่งลายเส้นนี้คือขอบของรูปภาพนี้เอง



(ก) การหาขอบด้วย Hit or Miss



(ข) การกำหนด SE โดยใช้ฟังก์ชัน Odd gabor กำหนด $\sigma = 1$, $\omega = 1$ และ $\theta = 1$

รูปที่ 4.11 (ก) การหาขอบภาพด้วย Hit or Miss

(ข) การกำหนด SE โดยใช้ฟังก์ชัน Odd gabor กำหนด $\sigma = 1$, $\omega = 1$ และ $\theta = 1$

ผลลัพธ์ที่ได้จากการประมวลผลของแต่ละตัวดำเนินการคงพอแสดงให้เห็นแล้วว่าการนำทฤษฎีมอร์โฟโลยีมาประยุกต์ใช้งานร่วมกับทฤษฎีฟัซซี่สามารถนำไปใช้งานได้อย่างมีประสิทธิภาพ

ทั้งหมดที่ได้กล่าวมาเป็นการนำตัวดำเนินการทั้งหมดมาใช้งานในการประมวลผล แต่ละตัวดำเนินการมีเอกลักษณ์ในการใช้งานที่แตกต่างกันไป ประสิทธิภาพที่ได้จะมากน้อยเพียงใดขึ้นอยู่กับการกำหนดค่าความเป็นสมาชิกของ SE ที่แตกต่างกันไปตามลักษณะโครงสร้างของรูปภาพที่นำมาประมวลผล

บทที่ 5

บทสรุปและวิจารณ์

ฟัซซี่มอร์โฟโลยีเกิดจากการรวมทฤษฎีสองทฤษฎีคือฟัซซี่เซตและมอร์โฟโลยีมาประยุกต์ใช้งานร่วมกัน โดยอาศัยตัวดำเนินการและ SE เพื่อนำมากำจัดสัญญาณรบกวนและหาขอบของภาพที่เป็นเกรย์สเกล การกำหนดค่าของ SE ได้อาศัยฟังก์ชัน Gussian และ ฟังก์ชัน Garbor เข้ามาช่วยเพื่อทำให้สามารถหาค่าที่เหมาะสมในการมากำจัดสัญญาณรบกวนและหาขอบของภาพ

ตัวดำเนินการ Erosion และ Dilation เป็นตัวดำเนินการพื้นฐานของมอร์โฟโลยีที่ต้องทำการศึกษาเป็นอันดับแรกซึ่งสามารถใช้งานได้ดีกับภาพไบนารี จึงได้พัฒนาร่วมกับฟัซซี่เซตเพื่อนำมาใช้กับภาพเกรย์สเกล เป็นตัวดำเนินการที่สามารถปรับระดับค่าความเป็นสมาชิกของรูปภาพให้มากขึ้นหรือน้อยลง โดยขึ้นอยู่กับกำหนดค่าความเป็นสมาชิกและขนาด SE แต่ความยุ่งยากก็จะตามมาในการคำนวณหาค่าความเป็นสมาชิกที่เหมาะสม

ตัวดำเนินการพื้นฐานของมอร์โฟโลยีทั้งสองตัวนี้ยังนำไปประยุกต์ใช้งานร่วมกัน เพื่อสร้างตัวดำเนินการใหม่ให้เกิดขึ้นคือตัวดำเนินการ Opening เป็นตัวดำเนินการที่เกิดจากการทำ Erosion แล้วผลลัพธ์ที่ได้มาทำการ Dilation อีกครั้งหนึ่งใช้ในการกำจัดสัญญาณรบกวนที่เกิดจากสัญญาณรบกวนที่เป็นโทนสีขาว ส่วนตัวดำเนินการ Closing เกิดจากการทำ Dilation แล้วนำผลลัพธ์ที่ได้มาทำการ Erosion อีกครั้ง ซึ่งเป็นตัวดำเนินการที่ใช้ในการกำจัดสัญญาณรบกวนที่เป็นโทนสีดำ ตัวดำเนินการ Opening และ Closing เป็นตัวดำเนินการของมอ์โฟโลยี ดังนั้นจึงได้มีการพัฒนาร่วมกับฟัตซี่เซตเพื่อเพื่อนำมาใช้กับภาพเกรย์สเกล

ในส่วนการหาขอบภาพยังสามารถใช้ตัวดำเนินการพื้นฐานของมอร์โฟโลยีที่หาขอบภาพได้ แต่มีอีกตัวดำเนินการอีกตัวที่มีประสิทธิภาพในการหาขอบภาพคือตัวดำเนินการ Hit or Miss ซึ่งยังมีโครงสร้างของตัวดำเนินการที่เกิดจากตัวดำเนินการพื้นฐานของมอร์โฟโลยีโดยจะใช้งานได้เฉพาะภาพไบนารี ดังนั้นจึงได้มีการพัฒนาร่วมกับฟัซซี่เซตเพื่อนำมาใช้กับภาพเกรย์สเกล

เนื่องจากการประมวลผลกระทำทุกๆ ต่อจุดของข้อมูลภาพภาพ ดังนั้นจะส่งผลทำให้จุดของข้อมูลภาพที่มีระดับค่าสีปกติผ่านการประมวลผลไปด้วยทำให้มีการเปลี่ยนแปลงไปจากเดิมซึ่งเป็นสิ่งที่ไม่ต้องการ ในการกำจัดสัญญาณรบกวนที่เกิดในข้อมูลภาพสามารถกระทำได้ดีกับสัญญาณรบกวนที่มีค่าความเป็นสมาชิกที่ระดับต่ำสุดหรือสีดำและสูงสุดหรือสีขาว เป็นการแสดงให้เห็นว่ายังคงจำกัดขอบเขตความสามารถอยู่ที่ค่าความเป็นสมาชิกของสัญญาณรบกวน ส่วนสัญญาณรบกวนที่อยู่บริเวณขอบภาพไม่สามารถกำจัดได้ซึ่งต้องทำการหาวิธีแก้ไขในส่วนนี้ การกำหนดค่าความเป็นสมาชิกและขนาดของโครงสร้างพื้นฐานที่เหมาะสมค่อนข้างจะยุงยาก ซึ่งเป็นปัจจัยหลักในการหาขอบและกำจัดสัญญาณรบกวน การแก้ปัญหาตรงส่วนนี้ผู้ใช้โปรแกรมต้อง

ทดลองเลือกค่าตัวแปรของแต่ละตัวกรองสัญญาณและจดบันทึกผลลัพธ์ที่ได้นำมาเปรียบเทียบเพื่อเลือกค่าที่ดีเหมาะ ต่อมาได้มีการพัฒนานำฟังก์ชันมาคำนวณหาค่าความเป็นสมาชิกเพื่อให้ผลลัพธ์ที่มีความเหมาะสมมากขึ้น

ภาคผนวก

ความหมายของสัญลักษณ์ที่ใช้

A	เซตของรูปภาพ
B	เซตของโครงสร้างพื้นฐาน (Structure Element:SE)
B_1	เซตย่อยโครงสร้างพื้นฐาน
B_2	เซตย่อยโครงสร้างพื้นฐาน
$\bigcup$	เอกภพสัมพัทธ์
$\cap$	การอินเตอร์เซ็กชัน
$\supseteq$	การสับเซต
$\cup$	การยูเนียน
(A^{c})	คอมพลีเมนต์ของ A
$\mu_A(x)$	ฟังก์ชันความเป็นสมาชิกของเซต A
$\Im$	ฟังก์ชัน Translation
ϑ	ฟังก์ชัน Index

ส่วนนี้เป็นสัญลักษณ์ของมอร์โฟโลยี

$\ominus$	ตัวดำเนินการของ Dilitlion
$\oplus$	ตัวดำเนินการของ Erosion
$\circledast$	ตัวดำเนินการของ Hit or Miss
$\circ$	ตัวดำเนินการของ Opening
$\bullet$	ตัวดำเนินการของ Closing
$\otimes$	ตัวดำเนินการของ Thinning
$\odot$	ตัวดำเนินการของ Thinckening

ตัวดำเนินการของ Dilationส่วนนี้เป็นสัญลักษณ์ของฟัซซี่มอร์โฟโลยี

	ตัวดำเนินการของ Dilitlion
	ตัวดำเนินการของ Erosion
	ตัวดำเนินการของ Opening
	ตัวดำเนินการของ Closing

โปรแกรมฟัซซี่มอร์โฟโลยี

```
procedure TForm1.GrayScale;
var k,l:integer;
begin
  Interval;
  for k:=0 to Long do
   for l:=0 to High do
    begin
    Level:=ColorToRgb(Image1.Canvas.Pixels[k,l]);
    Image2.Canvas.Pixels[k,l]:=RGB(Level,Level,Level);
    Image[k,l]:=Level/IntV;
  end;
end;

procedure Tform1.Interval;
var k,l:integer;
begin
  for k:=0 to Long do
   for l:=0 to High do
   begin
    Level:=ColorToRgb(Image1.Canvas.Pixels[k,l]);
    If level>max then max:=level;
    If level<min then min:=level;
    end;
  intv:=max-min;
end;

procedure TForm1.ImageComplement;
var i,j:integer;
```

```
begin
  for i:=0 to Long do
   for j:=0 to High do
   begin
    ImageC[i,j]:=1-Image[i,j];
   end;
end;
```

procedure TForm1.SeComplement;

```
var i,j:integer;
begin
  for i:=-2 to 2 do
   for j:=-2 to 2 do
   begin
    SeC[i,j]:=1-StrTofloat(Se[i,j]);
   end;
end;
```

procedure TForm1.Erosion;

```
var a,b:integer;
    inf:real;
begin
SeComplement;
if dimem=0 then
 begin
  {
  for a:=1 to long-1 do
   for b:=1 to high-1 do
   begin
```

```
if (Image[a-1,b-1]+SeC[-1,-1]) > (Image[a-1,b]+SeC[-1,0]) then
 Inf:=(Image[a-1,b]+SeC[-1,0])
else
 Inf:=(Image[a-1,b-1]+SeC[-1,-1]);
 if Inf> (Image[a-1,b+1]+SeC[-1,1]) then
  Inf:=(Image[a-1,b+1]+SeC[-1,1]);
 if Inf >(Image[a,b-1]+SeC[0,-1]) then
  Inf:=(Image[a,b-1]+SeC[0,-1]);
 if Inf >(Image[a,b]+SeC[0,0]) then
  Inf:=(Image[a,b]+SeC[0,0]);
 if Inf >(Image[a,b+1]+SeC[0,1]) then
  Inf:=(Image[a,b+1]+SeC[0,1]);
 if Inf >(Image[a+1,b-1]+SeC[1,-1]) then
  Inf:=(Image[a+1,b-1]+SeC[1,-1]);
 if Inf >(Image[a+1,b]+SeC[1,0]) then
  Inf:=(Image[a+1,b]+SeC[1,0]);
 if Inf >(Image[a+1,b+1]+SeC[1,1]) then
  Inf:=(Image[a+1,b+1]+SeC[1,1]);
 str(inf*256:3:0,Erode[a,b]);
 if strtoint(Erode[a,b])>255 then Erode[a,b]:='255';
 if strtoint(Erode[a,b])<0 then Erode[a,b]:='0';
 Image2.Canvas.Pixels[a,b]:=RGB(strtoint(Erode[a,b]),strtoint(Erode[a,b]),strtoint(Erode[a,b]));
 DataE[strtoint(Erode[a,b])]:=DataE[strtoint(Erode[a,b])]+1;
end;
}
end;
if Dimem=1 then
begin
{
```

```
for a:=2 to Long-2 do
 for b:=2 to High-2 do
 begin
  if (Image[a-2,b-2]+SeC[-2,-2]) > (Image[a-2,b-1]+SeC[-2,-1]) then
   Inf:=(Image[a-2,b-1]+SeC[-2,-1])
  else
   Inf:=(Image[a-2,b-2]+SeC[-2,-2]);
   if Inf > (Image[a-2,b]+SeC[-2,0]) then
    Inf:=(Image[a-2,b]+SeC[-2,0]);
   If Inf >(Image[a-2,b+1]+SeC[-2,1]) then
    Inf:=(Image[a-2,b+1]+SeC[-2,1]);
   If Inf >(Image[a-2,b+2]+SeC[-2,2]) then
    Inf:=(Image[a-2,b+2]+SeC[-2,2]);
   if Inf >(Image[a-1,b-2]+SeC[-1,-2]) then
    Inf:=(Image[a-1,b-2]+SeC[-1,-2]);
   if Inf >(Image[a-1,b-1]+SeC[-1,-1]) then
    Inf:=(Image[a-1,b-1]+SeC[-1,-1]);
   if Inf >(Image[a-1,b]+SeC[-1,0]) then
    Inf:=(Image[a-1,b]+SeC[-1,0]);
   if Inf >(Image[a-1,b+1]+SeC[-1,1]) then
    Inf:=(Image[a-1,b+1]+SeC[-1,1]);
   if Inf > (Image[a-1,b+2]+SeC[-1,2]) then
    Inf:=(Image[a-1,b+2]+SeC[-1,2]);
   if Inf >(Image[a,b-2]+SeC[0,-2]) then
    Inf:=(Image[a,b-2]+SeC[0,-2]);
   if Inf >(Image[a,b-1]+SeC[0,-1]) then
    Inf:=(Image[a,b-1]+SeC[0,-1]);
   if Inf >(Image[a,b]+SeC[0,0]) then
    Inf:=(Image[a,b]+SeC[0,0]);
   if Inf >(Image[a,b+1]+SeC[0,1]) then
```

```
Inf:=(Image[a,b+1]+SeC[0,1]);
if Inf >(Image[a,b+2]+SeC[0,2]) then
Inf:=(Image[a,b+2]+SeC[0,2]);
if Inf >(Image[a+1,b-2]+SeC[1,-2]) then
Inf:=(Image[a+1,b-2]+SeC[1,-2]);
if Inf >(Image[a+1,b-1]+SeC[1,-1]) then
Inf:=(Image[a+1,b-1]+SeC[1,-1]);
If Inf>(Image[a+1,b]+SeC[1,0]) then
Inf:=(Image[a+1,b]+SeC[1,0]);
if Inf> (Image[a+1,b+1]+SeC[1,1]) then
Inf:=(Image[a+1,b+1]+SeC[1,1]);
if Inf >(Image[a+1,b+2]+SeC[1,2]) then
Inf:=(Image[a+1,b+2]+SeC[1,2]);
if Inf >(Image[a+2,b-2]+SeC[2,-2]) then
Inf:=(Image[a+2,b-2]+SeC[2,-2]);
if Inf >(Image[a+2,b-1]+SeC[2,-1]) then
Inf:=(Image[a+2,b-1]+SeC[2,-1]);
if Inf >(Image[a+2,b]+SeC[2,0]) then
Inf:=(Image[a+2,b]+SeC[2,0]);
if Inf >(Image[a+2,b+1]+SeC[2,1]) then
Inf:=(Image[a+2,b+1]+SeC[2,1]);
if Inf >(Image[a+2,b+2]+SeC[2,2]) then
Inf:=(Image[a+2,b+2]+SeC[2,2]);
str(inf*256:3:0,Erode[a,b]);
If strtoint(Erode[a,b])>255 then Erode[a,b]:='255'
If strtoint(Erode[a,b])<0 then Erode[a,b]:='0';
Image2.Canvas.Pixels[a,b]:=RGB(strtoint(Erode[a,b]),strtoint(Erode[a,b]),strtoint(Erode[a,b]));
DataE[strtoint(Erode[a,b])]:=DataE[strtoint(Erode[a,b])]+1;
end;
}
```

```
end;
end;

procedure TForm1.Dilation;
var a,b:integer;
begin
ImageComplement;.
SeComplement;.
if dimem=0 then
begin
{
 for a:=1 to long-1 do
 for b:=1 to high-1 do
 begin
  if(ImageC[a-1,b-1]+SeC[-1,-1]) > (ImageC[a-1,b]+SeC[-1,0]) then
   Sup:=(ImageC[a-1,b]+SeC[-1,0])
  else
   Sup:=(ImageC[a-1,b-1]+SeC[-1,-1]);
  if Sup > (ImageC[a-1,b+1]+SeC[-1,1]) then
   Sup:=(ImageC[a-1,b+1]+SeC[-1,1]);
  If Sup >(ImageC[a,b-1]+SeC[0,-1]) then
   Sup:=(ImageC[a,b-1]+SeC[0,-1]);
  if Sup >(ImageC[a,b]+SeC[0,0]) then
   Sup:=(ImageC[a,b]+SeC[0,0]);
 if Sup >(ImageC[a,b+1]+SeC[0,1]) then
  Sup:=(ImageC[a,b+1]+SeC[0,1]);
 if Sup >(ImageC[a+1,b-1]+SeC[1,-1]) then
  Sup:=(ImageC[a+1,b-1]+SeC[1,-1]);
 if Sup >(ImageC[a+1,b]+SeC[1,0]) then
  Sup:=(ImageC[a+1,b]+SeC[1,0]);
```

```
    if Sup >(ImageC[a+1,b+1]+SeC[1,1]) then
     Sup:=(ImageC[a+1,b+1]+SeC[1,1]);
     str((1-Sup)*256:3:0,Dilate[a,b]);
     if strtoint(Dilate[a,b])>255 then Dilate[a,b]:='255';
     if strtoint(Dilate[a,b])<0 then Dilate[a,b]:='0';
     Image2.Canvas.Pixels[a,b]:=RGB(strtoint(Dilate[a,b]),strtoint(Dilate  [a,b]),strtoint(Dilate[a,b]));
     DataD[strtoint(Dilate[a,b])]:=dataD[strtoint(Dilate[a,b])]+1;
   end;
 end;
if dimem=1 then
begin
 for a:=2 to long-2 do
  for b:=2 to high-2 do
   begin
   if (ImageC[a-2,b-2]+SeC[-2,-2]) > (ImageC[a-2,b-1]+SeC[-2,-1]) then
    Sup:=(ImageC[a-2,b-1]+SeC[-2,-1])
   else
    Sup:=(ImageC[a-2,b-2]+SeC[-2,-2]);
    if Sup > (ImageC[a-2,b]+SeC[-2,0]) then
     Sup:=(ImageC[a-2,b]+SeC[-2,0]);
    If Sup >(ImageC[a-2,b+1]+SeC[-2,1]) then
     Sup:=(ImageC[a-2,b+1]+SeC[-2,1]);
    if Sup >(ImageC[a-2,b+2]+SeC[-2,2]) then
     Sup:=(ImageC[a-2,b+2]+SeC[-2,2]);
    if Sup >(ImageC[a-1,b-2]+SeC[-1,-2]) then
     Sup:=(ImageC[a-1,b-2]+SeC[-1,-2]);
    if Sup >(ImageC[a-1,b-1]+SeC[-1,-1]) then
     Sup:=(ImageC[a-1,b-1]+SeC[-1,-1]);
    if Sup >(ImageC[a-1,b]+SeC[-1,0]) then
     Sup:=(ImageC[a-1,b]+SeC[-1,0]);
```

```
if Sup >(ImageC[a-1,b+1]+SeC[-1,1]) then
 Sup:=(ImageC[a-1,b+1]+SeC[-1,1]);
if Sup > (ImageC[a-1,b+2]+SeC[-1,2]) then
 Sup:=(ImageC[a-1,b+2]+SeC[-1,2]);
if Sup >(ImageC[a,b-2]+SeC[0,-2]) then
 Sup:=(ImageC[a,b-2]+SeC[0,-2]);
if Sup >(ImageC[a,b-1]+SeC[0,-1]) then
 Sup:=(ImageC[a,b-1]+SeC[0,-1]);
if Sup >(ImageC[a,b]+SeC[0,0]) then
 Sup:=(ImageC[a,b]+SeC[0,0]);
if Sup >(ImageC[a,b+1]+SeC[0,1]) then
 Sup:=(ImageC[a,b+1]+SeC[0,1]);
if Sup >(ImageC[a,b+2]+SeC[0,2]) then
 Sup:=(ImageC[a,b+2]+SeC[0,2]);
if Sup >(ImageC[a+1,b-2]+SeC[1,-2]) then
 Sup:=(ImageC[a+1,b-2]+SeC[1,-2]);
if Sup >(ImageC[a+1,b-1]+SeC[1,-1]) then
 Sup:=(ImageC[a+1,b-1]+SeC[1,-1]);
if Sup >(ImageC[a+1,b]+SeC[1,0]) then
 Sup:=(ImageC[a+1,b]+SeC[1,0]);
if Sup > (ImageC[a+1,b+1]+SeC[1,1]) then
 Sup:=(ImageC[a+1,b+1]+SeC[1,1]);
if Sup >(ImageC[a+1,b+2]+SeC[1,2]) then
 Sup:=(ImageC[a+1,b+2]+SeC[1,2]);
if Sup >(ImageC[a+2,b-2]+SeC[2,-2]) then
 Sup:=(ImageC[a+2,b-2]+SeC[2,-2]);
if Sup >(ImageC[a+2,b-1]+SeC[2,-1]) then
 Sup:=(ImageC[a+2,b-1]+SeC[2,-1]);
If Sup >(ImageC[a+2,b]+SeC[2,0]) then
 Sup:=(ImageC[a+2,b]+SeC[2,0]);
```

```
 If Sup >(ImageC[a+2,b+1]+SeC[2,1]) then
  Sup:=(ImageC[a+2,b+1]+SeC[2,1]);
 if Sup >(ImageC[a+2,b+2]+SeC[2,2]) then
  Sup:=(ImageC[a+2,b+2]+SeC[2,2]);
  str((1-Sup)*256:3:0,Dilate[a,b]);
  if strtoint(Dilate[a,b])>255 then Dilate[a,b]:='255';
  if strtoint(Dilate[a,b])<0 then Dilate[a,b]:='0';
  Image2.Canvas.Pixels[a,b]:=RGB(strtoint(Dilate[a,b]),strtoint(Dilate[a,b]),strtoint(Dilate[a,b]);
  DataD[strtoint(Dilate[a,b])]:=dataD[strtoint(Dilate[a,b])]+1;
 end;
 }
 end;
end;
```

procedure Tform1.Opening;

```
var a,b,c,d:integer;
begin
ImageComplement;
SeComplement;  if dimem=0 then
 begin
 {
 for a:=1 to long-1 do
  for b:=1 to high-1 do
  begin
   {
  if (Image[a-1,b-1]+SeC[-1,-1]) > (Image[a-1,b]+SeC[-1,0]) then
   inf:=(Image[a-1,b]+SeC[-1,0])
  else
   inf:=(Image[a-1,b-1]+SeC[-1,-1]);
   if inf > (Image[a-1,b+1]+SeC[-1,1]) then
```

```
inf:=(Image[a-1,b+1]+SeC[-1,1]);
if inf >(Image[a,b-1]+SeC[0,-1]) then
inf:=(Image[a,b-1]+SeC[0,-1]);
if inf >(Image[a,b]+SeC[0,0]) then
inf:=(Image[a,b]+SeC[0,0]);
if inf >(Image[a,b+1]+SeC[0,1]) then
inf:=(Image[a,b+1]+SeC[0,1]);
if inf >(Image[a+1,b-1]+SeC[1,-1]) then
inf:=(Image[a+1,b-1]+SeC[1,-1]);
if inf >(Image[a+1,b]+SeC[1,0]) then
inf:=(Image[a+1,b]+SeC[1,0]);
if inf >(Image[a+1,b+1]+SeC[1,1]) then
inf:=(Image[a+1,b+1]+SeC[1,1]);
Erode[a,b]:=floattostr(inf);
if strtofloat(Erode[a,b])>1 then Erode[a,b]:='1';
if strtofloat(Erode[a,b])<0 then Erode[a,b]:='0';
}
end;
for c:=2 to long-2 do
for d:=2 to high-2 do
begin
{
if (strtofloat(Erode[c-1,d-1])+SeC[-1,-1]) > (strtofloat(Erode[c-1,d])
+SeC[-1,0]) then
sup:=(strtofloat(Erode[c-1,d])+SeC[-1,0])
else
sup:=(strtofloat(Erode[c-1,d-1])+SeC[-1,-1]);
if sup > (strtofloat(Erode[c-1,d+1])+SeC[-1,1]) then
sup:=(strtofloat(Erode[c-1,d+1])+SeC[-1,1]);
if sup >(strtofloat(Erode[c,d-1])+SeC[0,-1]) then
```

```
      sup:=(strtofloat(Erode[c,d-1])+SeC[0,-1]);
     if sup >(strtofloat(Erode[c,d])+SeC[0,0]) then
      sup:=(strtofloat(Erode[c,d])+SeC[0,0]);
     if sup >(strtofloat(Erode[c,d+1])+SeC[0,1]) then
     sup:=(strtofloat(Erode[c,d+1])+SeC[0,1]);
    if sup >(strtofloat(Erode[c+1,d-1])+SeC[1,-1]) then
     sup:=(strtofloat(Erode[c+1,d-1])+SeC[1,-1]);
    if sup >(strtofloat(Erode[c+1,d])+SeC[1,0]) then
     sup:=(strtofloat(Erode[c+1,d])+SeC[1,0]);
    if sup >(strtofloat(Erode[c+1,d+1])+SeC[1,1]) then
     sup:=(strtofloat(Erode[c+1,d+1])+SeC[1,1]);
     str(Sup*256:3:0,Dilate[c,d]);
if strtoint(Dilate[c,d])>255 then Dilate[c,d]:='255';
if strtoint(Dilate[c,d])<0 then Dilate[c,d]:='0';
Image2.Canvas.Pixels[c,d]:=RGB(strtoint(Dilate[c,d]),strtoint(Dilate[c,d]),strtoint(Dilate[c,d]));
dataO[strtoint(Dilate[c,d])]:=dataO[strtoint(Dilate[c,d])]+1;
}
    end;
 }
end;


if dimem=1 then
Begin
{
 for a:=2 to long-2 do
  for b:=2 to high-2 do
  begin
   {
  if (Image[a-2,b-2]+SeC[-2,-2]) > (Image[a-2,b-1]+SeC[-2,-1]) then
   inf:=(Image[a-2,b-1]+SeC[-2,-1])
```

```
else
 inf:=(Image[a-2,b-2]+SeC[-2,-2]);
 if inf > (Image[a-2,b]+SeC[-2,0]) then
  inf:=(Image[a-2,b]+SeC[-2,0]);
 if inf >(Image[a-2,b+1]+SeC[-2,1]) then
  inf:=(Image[a-2,b+1]+SeC[-2,1]);
 if inf >(Image[a-2,b+2]+SeC[-2,2]) then
  inf:=(Image[a-2,b+2]+SeC[-2,2]);
 if inf >(Image[a-1,b-2]+SeC[-1,-2]) then
  inf:=(Image[a-1,b-2]+SeC[-1,-2]);
 if inf >(Image[a-1,b-1]+SeC[-1,-1]) then
  inf:=(Image[a-1,b-1]+SeC[-1,-1]);
 if inf >(Image[a-1,b]+SeC[-1,0]) then
  inf:=(Image[a-1,b]+SeC[-1,0]);
 if inf >(Image[a-1,b+1]+SeC[-1,1]) then
  inf:=(Image[a-1,b+1]+SeC[-1,1]);
 if inf > (Image[a-1,b+2]+SeC[-1,2]) then
  inf:=(Image[a-1,b+2]+SeC[-1,2]);
 if inf >(Image[a,b-2]+SeC[0,-2]) then
  inf:=(Image[a,b-2]+SeC[0,-2]);
 if inf >(Image[a,b-1]+SeC[0,-1]) then
  inf:=(Image[a,b-1]+SeC[0,-1]);
 if inf >(Image[a,b]+SeC[0,0]) then
  inf:=(Image[a,b]+SeC[0,0]);
 if inf >(Image[a,b+1]+SeC[0,1]) then
  inf:=(Image[a,b+1]+SeC[0,1]);
 if inf >(Image[a,b+2]+SeC[0,2]) then
  inf:=(Image[a,b+2]+SeC[0,2]);
 if inf >(Image[a+1,b-2]+SeC[1,-2]) then
  inf:=(Image[a+1,b-2]+SeC[1,-2]);
```

```
if inf >(Image[a+1,b-1]+SeC[1,-1]) then
 inf:=(Image[a+1,b-1]+SeC[1,-1]);
if inf >(Image[a+1,b]+SeC[1,0]) then
 inf:=(Image[a+1,b]+SeC[1,0]);
if inf > (Image[a+1,b+1]+SeC[1,1]) then
 inf:=(Image[a+1,b+1]+SeC[1,1]);
if inf >(Image[a+1,b+2]+SeC[1,2]) then
 inf:=(Image[a+1,b+2]+SeC[1,2]);
if inf >(Image[a+2,b-2]+SeC[2,-2]) then
 inf:=(Image[a+2,b-2]+SeC[2,-2]);
if inf >(Image[a+2,b-1]+SeC[2,-1]) then
 inf:=(Image[a+2,b-1]+SeC[2,-1]);
if inf >(Image[a+2,b]+SeC[2,0]) then
 inf:=(Image[a+2,b]+SeC[2,0]);
if inf >(Image[a+2,b+1]+SeC[2,1]) then
 inf:=(Image[a+2,b+1]+SeC[2,1]);
if inf >(Image[a+2,b+2]+SeC[2,2]) then
 inf:=(Image[a+2,b+2]+SeC[2,2]);
 Erode[a,b]:=floattostr(inf);
if strtofloat(Erode[a,b])>1 then Erode[a,b]:='1';
if strtofloat(Erode[a,b])<0 then Erode[a,b]:='0';
}
end;
for c:=4 to long-4 do
 for d:=4 to high-4 do
  begin
   {
if (strtofloat(Erode[c-2,d-2])+SeC[-2,-2]) > (strtofloat(Erode[c-2,d-1])
+SeC[-2,-1]) then
 sup:=(strtofloat(Erode[c-2,d-1])+SeC[-2,-1])
```

```
else
sup:=(strtofloat(Erode[c-2,d-2])+SeC[-2,-2]);
if sup > (strtofloat(Erode[c-2,d])+SeC[-2,0]) then
sup:=(strtofloat(Erode[c-2,d])+SeC[-2,0]);
if sup >(strtofloat(Erode[c-2,d+1])+SeC[-2,1]) then
sup:=(strtofloat(Erode[c-2,d+1])+SeC[-2,1]);
if sup >(strtofloat(Erode[c-2,d+2])+SeC[-2,2]) then
sup:=(strtofloat(Erode[c-2,d+2])+SeC[-2,2]);
if sup >(strtofloat(Erode[c-1,d-2])+SeC[-1,-2]) then
sup:=(strtofloat(Erode[c-1,d-2])+SeC[-1,-2]);
if sup >(strtofloat(Erode[c-1,d-1])+SeC[-1,-1]) then
sup:=(strtofloat(Erode[c-1,d-1])+SeC[-1,-1]);
if sup >(strtofloat(Erode[c-1,d])+SeC[-1,0]) then
sup:=(strtofloat(Erode[c-1,d])+SeC[-1,0]);
if sup >(strtofloat(Erode[c-1,d+1])+SeC[-1,1]) then
sup:=(strtofloat(Erode[c-1,d+1])+SeC[-1,1]);
if sup > (strtofloat(Erode[c-1,d+2])+SeC[-1,2]) then
sup:=(strtofloat(Erode[c-1,d+2])+SeC[-1,2]);
if sup >(strtofloat(Erode[c,d-2])+SeC[0,-2]) then
sup:=(strtofloat(Erode[c,d-2])+SeC[0,-2]);
if sup >(strtofloat(Erode[c,d-1])+SeC[0,-1]) then
sup:=(strtofloat(Erode[c,d-1])+SeC[0,-1]);
if sup >(strtofloat(Erode[c,d])+SeC[0,0]) then
sup:=(strtofloat(Erode[c,d])+SeC[0,0]);
if sup >(strtofloat(Erode[c,d+1])+SeC[0,1]) then
sup:=(strtofloat(Erode[c,d+1])+SeC[0,1]);
if sup >(strtofloat(Erode[c,d+2])+SeC[0,2]) then
sup:=(strtofloat(Erode[c,d+2])+SeC[0,2]);
if sup >(strtofloat(Erode[c+1,d-2])+SeC[1,-2]) then
sup:=(strtofloat(Erode[c+1,d-2])+SeC[1,-2]);
```

```
if sup >(strtofloat(Erode[c+1,d-1])+SeC[1,-1]) then
 sup:=(strtofloat(Erode[c+1,d-1])+SeC[1,-1]);
if sup >(strtofloat(Erode[c+1,d])+SeC[1,0]) then
 sup:=(strtofloat(Erode[c+1,d])+SeC[1,0]);
if sup > (strtofloat(Erode[c+1,d+1])+SeC[1,1]) then
 sup:=(strtofloat(Erode[c+1,d+1])+SeC[1,1]);
if sup >(strtofloat(Erode[c+1,d+2])+SeC[1,2]) then
 sup:=(strtofloat(Erode[c+1,d+2])+SeC[1,2]);
if sup >(strtofloat(Erode[c+2,d-2])+SeC[2,-2]) then
 sup:=(strtofloat(Erode[c+2,d-2])+SeC[2,-2]);
if sup >(strtofloat(Erode[c+2,d-1])+SeC[2,-1]) then
 sup:=(strtofloat(Erode[c+2,d-1])+SeC[2,-1]);
if sup >(strtofloat(Erode[c+2,d])+SeC[2,0]) then
 sup:=(strtofloat(Erode[c+2,d])+SeC[2,0]);
if sup >(strtofloat(Erode[c+2,d+1])+SeC[2,1]) then
 sup:=(strtofloat(Erode[c+2,d+1])+SeC[2,1]);
if sup >(strtofloat(Erode[c+2,d+2])+SeC[2,2]) then
 sup:=(strtofloat(Erode[c+2,d+2])+SeC[2,2]);
 str(Sup*256:3:0,Dilate[c,d]);
  if strtoint(Dilate[c,d])>255 then Dilate[c,d]:='255';
if strtoint(Dilate[c,d])<0 then Dilate[c,d]:='0';
Image2.Canvas.Pixels[c,d]:=RGB(strtoint(Dilate[c,d]),strtoint(Dilate[c,d]),strtoint(Dilate[c,d]));
dataO[strtoint(Dilate[c,d])]:=dataO[strtoint(Dilate[c,d])]+1
 }
 end;
}
end;
end;
```

procedure TForm1.Closing;

```
var a,b,c,d:integer;
begin
ImageComplement ;
SeComplement.
if dimem=0 then
begin
 {
 for a:=1 to long-1 do
  for b:=1 to high-1 do
  begin
  {
  if (ImageC[a-1,b-1]+SeC[-1,-1]) > (ImageC[a-1,b]+SeC[-1,0]) then
   sup:=(ImageC[a-1,b]+SeC[-1,0])
   else
   sup:=(ImageC[a-1,b-1]+SeC[-1,-1]);
   if sup > (ImageC[a-1,b+1]+SeC[-1,1]) then
    sup:=(ImageC[a-1,b+1]+SeC[-1,1]);
   if sup >(ImageC[a,b-1]+SeC[0,-1]) then
    sup:=(ImageC[a,b-1]+SeC[0,-1]);
   if sup >(ImageC[a,b]+SeC[0,0]) then
    sup:=(ImageC[a,b]+SeC[0,0]);
   if sup >(ImageC[a,b+1]+SeC[0,1]) then
    sup:=(ImageC[a,b+1]+SeC[0,1]);
   if sup >(ImageC[a+1,b-1]+SeC[1,-1]) then
    sup:=(ImageC[a+1,b-1]+SeC[1,-1]);
   if sup >(ImageC[a+1,b]+SeC[1,0]) then
    sup:=(ImageC[a+1,b]+SeC[1,0]);
   if sup >(ImageC[a+1,b+1]+SeC[1,1]) then
    sup:=(ImageC[a+1,b+1]+SeC[1,1]);
   Dilate[a,b]:=floattostr(1-sup);
```

```
  if strtofloat(Dilate[a,b])>1 then Dilate[a,b]:='1';
  if strtofloat(Dilate[a,b])<0 then Dilate[a,b]:='0';
 }
 end;
 for c:=2 to long-2 do
  for d:=2 to high-2 do
   begin
  {
   if (strtofloat(Dilate[c-1,d-1])+SeC[-1,-1]) > (strtofloat(Dilate[c-1,d])
+SeC[-1,0]) then
    inf:=(strtofloat(Dilate[c-1,d])+SeC[-1,0])
   else
    inf:=(strtofloat(Dilate[c-1,d-1])+SeC[-1,-1]);
    if inf> (strtofloat(Dilate[c-1,d+1])+SeC[-1,1]) then
     inf:=(strtofloat(Dilate[c-1,d+1])+SeC[-1,1]);
    if inf >(strtofloat(Dilate[c,d-1])+SeC[0,-1]) then
     inf:=(strtofloat(Dilate[c,d-1])+SeC[0,-1]);
    if inf >(strtofloat(Dilate[c,d])+SeC[0,0])then
     inf :=(strtofloat(Dilate[c,d])+SeC[0,0]);
    if inf >(strtofloat(Dilate[c,d+1])+SeC[0,1]) then
     inf:=(strtofloat(Dilate[c,d+1])+SeC[0,1]);
    if inf >(strtofloat(Dilate[c+1,d-1])+SeC[1,-1]) then
     inf:=(strtofloat(Dilate[c+1,d-1])+SeC[1,-1]);
    if inf >(strtofloat(Dilate[c+1,d])+SeC[1,0]) then
     inf:=(strtofloat(Dilate[c+1,d])+SeC[1,0]);
    if inf >(strtofloat(Dilate[c+1,d+1])+SeC[1,1]) then
     inf:=(strtofloat(Dilate[c+1,d+1])+SeC[1,1]);
    str(inf*255:3:0,Erode[c,d]);
    if strtofloat(Erode[c,d])>255 then Erode[c,d]:='255';
    if strtofloat(Erode[c,d])<0 then Erode[c,d]:='0';
```

```
. Image2.canvas.pixels[c,d]:=rgb(strtoint(Erode[c,d]),strtoint(Erode[c,d]),strtoint(Erode[c,d]));
  dataC[strtoint(Erode[c,d])]:=dataE[strtoint(Erode[c,d])]+1;
   }
   end;
end;

if dimem=1 then
begin
for a:=1 to long-1 do
 for b:=1 to high-1 do
 begin
  {
  if (ImageC[a-2,b-2]+SeC[-2,-2]) > (ImageC[a-2,b-1]+SeC[-2,-1]) then
   sup:=(ImageC[a-2,b-1]+SeC[-2,-1])
  else
   sup:=(ImageC[a-2,b-2]+SeC[-2,-2]);
   if sup > (ImageC[a-2,b]+SeC[-2,0]) then
    sup:=(ImageC[a-2,b]+SeC[-2,0]);
   if sup >(ImageC[a-2,b+1]+SeC[-2,1]) then
    sup:=(ImageC[a-2,b+1]+SeC[-2,1]);
   if sup >(ImageC[a-2,b+2]+SeC[-2,2]) then
    sup:=(ImageC[a-2,b+2]+SeC[-2,2]);
   if sup >(ImageC[a-1,b-2]+SeC[-1,-2]) then
    sup:=(ImageC[a-1,b-2]+SeC[-1,-2]);
   if sup >(ImageC[a-1,b-1]+SeC[-1,-1]) then
    sup:=(ImageC[a-1,b-1]+SeC[-1,-1]);
   if sup >(ImageC[a-1,b]+SeC[-1,0]) then
    sup:=(ImageC[a-1,b]+SeC[-1,0]);
   if sup >(ImageC[a-1,b+1]+SeC[-1,1]) then
    sup:=(ImageC[a-1,b+1]+SeC[-1,1]);
```

```
if sup > (ImageC[a-1,b+2]+SeC[-1,2]) then
 sup:=(ImageC[a-1,b+2]+SeC[-1,2]);
if sup >(ImageC[a,b-2]+SeC[0,-2]) then
 sup:=(ImageC[a,b-2]+SeC[0,-2]);
if sup >(ImageC[a,b-1]+SeC[0,-1]) then
 sup:=(ImageC[a,b-1]+SeC[0,-1]);
if sup >(ImageC[a,b]+SeC[0,0]) then
 sup:=(ImageC[a,b]+SeC[0,0]);
if sup >(ImageC[a,b+1]+SeC[0,1]) then
 sup:=(ImageC[a,b+1]+SeC[0,1]);
if sup >(ImageC[a,b+2]+SeC[0,2]) then
 sup:=(ImageC[a,b+2]+SeC[0,2]);
if sup >(ImageC[a+1,b-2]+SeC[1,-2]) then
 sup:=(ImageC[a+1,b-2]+SeC[1,-2]);
if sup >(ImageC[a+1,b-1]+SeC[1,-1]) then
 sup:=(ImageC[a+1,b-1]+SeC[1,-1]);
if sup >(ImageC[a+1,b]+SeC[1,0]) then
 sup:=(ImageC[a+1,b]+SeC[1,0]);
if sup > (ImageC[a+1,b+1]+SeC[1,1]) then
 sup:=(ImageC[a+1,b+1]+SeC[1,1]);
if sup >(ImageC[a+1,b+2]+SeC[1,2]) then
 sup:=(ImageC[a+1,b+2]+SeC[1,2]);
If sup >(ImageC[a+2,b-2]+SeC[2,-2]) then
 sup:=(ImageC[a+2,b-2]+SeC[2,-2]);
If sup >(ImageC[a+2,b-1]+SeC[2,-1]) then
 sup:=(ImageC[a+2,b-1]+SeC[2,-1]);
if sup >(ImageC[a+2,b]+SeC[2,0]) then
 sup:=(ImageC[a+2,b]+SeC[2,0]);
if sup >(ImageC[a+2,b+1]+SeC[2,1]) then
 sup:=(ImageC[a+2,b+1]+SeC[2,1]);
```

```
If sup >(ImageC[a+2,b+2]+SeC[2,2]) then
 sup:=(ImageC[a+2,b+2]+SeC[2,2]);
 Dilate[a,b]:=floattostr(1-sup);
 if strtofloat(Dilate[a,b])>1 then Dilate[a,b]:='1';
if strtofloat(Dilate[a,b])<0 then Dilate[a,b]:='0';
}
  end;
for c:=3 to long-3 do
 for d:=3 to high-3 do
 begin
  {
  if (strtofloat(Dilate[c-2,d-2])+SeC[-2,-2]) > (strtofloat(Dilate[c-2,d-1])
+SeC[-2,-1]) then
   inf:=(strtofloat(Dilate[c-2,d-1])+SeC[-2,-1])
  else
   inf:=(strtofloat(Dilate[c-2,d-2])+SeC[-2,-2]);
   if inf > (strtofloat(Dilate[c-2,d])+SeC[-2,0]) then
    inf:=(strtofloat(Dilate[c-2,d])+SeC[-2,0]);
   if inf >(strtofloat(Dilate[c-2,d+1])+SeC[-2,1]) then
    inf:=(strtofloat(Dilate[c-2,d+1])+SeC[-2,1]);
   if inf >(strtofloat(Dilate[c-2,d+2])+SeC[-2,2]) then
    inf:=(strtofloat(Dilate[c-2,d+2])+SeC[-2,2]);
   if inf >(strtofloat(Dilate[c-1,d-2])+SeC[-1,-2]) then
    inf:=(strtofloat(Dilate[c-1,d-2])+SeC[-1,-2]);
   if inf >(strtofloat(Dilate[c-1,d-1])+SeC[-1,-1]) then
    inf:=(strtofloat(Dilate[c-1,d-1])+SeC[-1,-1]);
   if inf >(strtofloat(Dilate[c-1,d])+SeC[-1,0]) then
    inf:=(strtofloat(Dilate[c-1,d])+SeC[-1,0]);
   if inf >(strtofloat(Dilate[c-1,d+1])+SeC[-1,1]) then
    inf:=(strtofloat(Dilate[c-1,d+1])+SeC[-1,1]);
```

```
if inf> (strtofloat(Dilate[c-1,d+2])+SeC[-1,2]) then
 inf:=(strtofloat(Dilate[c-1,d+2])+SeC[-1,2]);
if inf >(strtofloat(Dilate[c,d-2])+SeC[0,-2]) then
 inf:=(strtofloat(Dilate[c,d-2])+SeC[0,-2]);
if inf >(strtofloat(Dilate[c,d-1])+SeC[0,-1]) then
 inf:=(strtofloat(Dilate[c,d-1])+SeC[0,-1]);
if inf >(strtofloat(Dilate[c,d])+SeC[0,0]) then
 inf:=(strtofloat(Dilate[c,d])+SeC[0,0]);
if inf >(strtofloat(Dilate[c,d+1])+SeC[0,1]) then
 inf:=(strtofloat(Dilate[c,d+1])+SeC[0,1]);
if inf >(strtofloat(Dilate[c,d+2])+SeC[0,2]) then
 inf:=(strtofloat(Dilate[c,d+2])+SeC[0,2]);
if inf >(strtofloat(Dilate[c+1,d-2])+SeC[1,-2]) then
 inf:=(strtofloat(Dilate[c+1,d-2])+SeC[1,-2]);
if inf >(strtofloat(Dilate[c+1,d-1])+SeC[1,-1]) then
 inf:=(strtofloat(Dilate[c+1,d-1])+SeC[1,-1]);
if inf >(strtofloat(Dilate[c+1,d])+SeC[1,0]) then
 inf:=(strtofloat(Dilate[c+1,d])+SeC[1,0]);
if inf> (strtofloat(Dilate[c+1,d+1])+SeC[1,1]) then
 inf:=(strtofloat(Dilate[c+1,d+1])+SeC[1,1]);
if inf >(strtofloat(Dilate[c+1,d+2])+SeC[1,2]) then
 inf:=(strtofloat(Dilate[c+1,d+2])+SeC[1,2]);
if inf >(strtofloat(Dilate[c+2,d-2])+SeC[2,-2]) then
 inf:=(strtofloat(Dilate[c+2,d-2])+SeC[2,-2]);
if inf >(strtofloat(Dilate[c+2,d-1])+SeC[2,-1]) then
 inf:=(strtofloat(Dilate[c+2,d-1])+SeC[2,-1]);
if inf >(strtofloat(Dilate[c+2,d])+SeC[2,0]) then
 inf:=(strtofloat(Dilate[c+2,d])+SeC[2,0]);
if inf >(strtofloat(Dilate[c+2,d+1])+SeC[2,1]) then
 inf:=(strtofloat(Dilate[c+2,d+1])+SeC[2,1]);
```

```
  if inf >(strtofloat(Dilate[c+2,d+2])+SeC[2,2]) then
   inf:=(strtofloat(Dilate[c+2,d+2])+SeC[2,2]);
   str(inf*255:3:0,erode[c,d]);
 if strtofloat(erode[c,d])>255 then erode[c,d]:='255';
 if strtofloat(Erode[c,d])<0 then Erode[c,d]:='0';
 Image2.canvas.pixels[c,d]:=rgb(strtoint(erode[c,d]),strtoint(erode[c,d]),strtoint(erode[c,d]));
 dataC[strtoint(Erode[c,d])]:=dataC[strtoint(Erode[c,d])]+1;
 }
  end;
 }
 end;
end;

procedure Tform1.HitorMiss;
var a,b:integer;
    hmt,mis,hi:real;
    ghmt:string;
begin
ImageComplement;
SeComplement; for a:=1 to long-1 do
 for b:=1 to high-1 do
  begin
  {
  if (ImageC[a-1,b-1]+SeC[-1,-1]) > (ImageC[a+1,b+1]+SeC[1,1]) then
   mis:=(ImageC[a+1,b+1]+SeC[1,1])
  else
   mis:=(ImageC[a-1,b-1]+SeC[-1,-1]);
  if mis > (ImageC[a-1,b+1]+SeC[-1,1]) then
   mis:=(ImageC[a-1,b+1]+SeC[-1,1]);
  if mis >(ImageC[a+1,b-1]+SeC[1,-1]) then
```

```
   mis:=(ImageC[a+1,b-1]+SeC[1,-1]);
   miss[a,b]:=floattostr(mis);
   }
  {
   if (Image[a,b-1]+SeC[0,-1])>(Image[a,b+1]+SeC[0,1]) then
   hi:=(Image[a,b+1]+SeC[0,1])
  else
   hi:=(Image[a,b-1]+SeC[0,-1]);
   if hi >(Image[a+1,b]+SeC[1,0]) then
   hi:=(Image[a+1,b]+SeC[1,0]);
   if hi >(Image[a-1,b]+SeC[-1,0]) then
   hi:=(Image[a-1,b]+SeC[-1,0]);
   hit[a,b]:=floattostr(hi);
   }
If strtofloat(hit[a,b])>strtofloat(miss[a,b]) then hmt:=strtofloat(miss[a,b]);        else hmt:=strtofloat(hit
[a,b]);
  str((image[a,b]-hmt)*255:3:0,ghmt);
. Image2.canvas.pixels[a,b]:=rgb(strtoint(ghmt),strtoint(ghmt),strtoint(ghmt));  dataH[ghmt]:=dataH
[ghmt]+1;
  end;
 end;
```

{..**Edging Detector**..}

```
procedure TForm1.Dilation3Click(Sender: TObject);
var a,b:integer;
begin
ImageComplement;
SeComplement;
if dimem=0 then
```

```
begin
 {
 for a:=1 to long-1 do
  for b:=1 to high-1 do
  begin
   if (ImageC[a-1,b-1]+SeC[-1,-1]) > (ImageC[a-1,b]+SeC[-1,0]) then
    Sup:=(ImageC[a-1,b]+SeC[-1,0])
   else
    Sup:=(ImageC[a-1,b-1]+SeC[-1,-1]);
    if Sup > (ImageC[a-1,b+1]+SeC[-1,1]) then
     Sup:=(ImageC[a-1,b+1]+SeC[-1,1]);
    if Sup >(ImageC[a,b-1]+SeC[0,-1]) then
     Sup:=(ImageC[a,b-1]+SeC[0,-1]);
    if Sup >(ImageC[a,b]+SeC[0,0]) then
     Sup:=(ImageC[a,b]+SeC[0,0]);
    if Sup >(ImageC[a,b+1]+SeC[0,1]) then
     Sup:=(ImageC[a,b+1]+SeC[0,1]);
    if Sup >(ImageC[a+1,b-1]+SeC[1,-1]) then
     Sup:=(ImageC[a+1,b-1]+SeC[1,-1]);
    if Sup >(ImageC[a+1,b]+SeC[1,0]) then
     Sup:=(ImageC[a+1,b]+SeC[1,0]);
    if Sup >(ImageC[a+1,b+1]+SeC[1,1]) then
     Sup:=(ImageC[a+1,b+1]+SeC[1,1]);
    str(((1-Sup)-image[a,b])*255:3:0,Dilate[a,b]);
    if strtofloat(Dilate[a,b])>255 then Dilate[a,b]:='255';
    if strtofloat(Dilate[a,b])<0then Dilate[a,b]:='0';
    Image2.Canvas.Pixels[a,b]:=RGB(strtoint(Dilate[a,b]),strtoint(Dilate  [a,b]),strtoint(Dilate[a,b]));
   end;
 }
end;
```

```
if dimem=1 then
begin
{
 for a:=2 to long-2 do
  for b:=2 to high-2 do
  begin
   if (ImageC[a-2,b-2]+SeC[-2,-2]) > (ImageC[a-2,b-1]+SeC[-2,-1]) then
    Sup:=(ImageC[a-2,b-1]+SeC[-2,-1])
   else
    Sup:=(ImageC[a-2,b-2]+SeC[-2,-2]);
    if Sup > (ImageC[a-2,b]+SeC[-2,0]) then
     Sup:=(ImageC[a-2,b]+SeC[-2,0]);
    if Sup >(ImageC[a-2,b+1]+SeC[-2,1]) then
     Sup:=(ImageC[a-2,b+1]+SeC[-2,1]);
    if Sup >(ImageC[a-2,b+2]+SeC[-2,2]) then
     Sup:=(ImageC[a-2,b+2]+SeC[-2,2]);
    if Sup >(ImageC[a-1,b-2]+SeC[-1,-2]) then
     Sup:=(ImageC[a-1,b-2]+SeC[-1,-2]);
    if Sup >(ImageC[a-1,b-1]+SeC[-1,-1]) then
     Sup:=(ImageC[a-1,b-1]+SeC[-1,-1]);
    if Sup >(ImageC[a-1,b]+SeC[-1,0]) then
     Sup:=(ImageC[a-1,b]+SeC[-1,0]);
    if Sup >(ImageC[a-1,b+1]+SeC[-1,1]) then
     Sup:=(ImageC[a-1,b+1]+SeC[-1,1]);
    if Sup > (ImageC[a-1,b+2]+SeC[-1,2]) then
     Sup:=(ImageC[a-1,b+2]+SeC[-1,2]);
    if Sup >(ImageC[a,b-2]+SeC[0,-2]) then
     Sup:=(ImageC[a,b-2]+SeC[0,-2]);
    if Sup >(ImageC[a,b-1]+SeC[0,-1]) then
     Sup:=(ImageC[a,b-1]+SeC[0,-1]);
```

```
        if Sup >(ImageC[a,b]+SeC[0,0]) then
         Sup:=(ImageC[a,b]+SeC[0,0]);
        if Sup >(ImageC[a,b+1]+SeC[0,1]) then
         Sup:=(ImageC[a,b+1]+SeC[0,1]);
        if Sup >(ImageC[a,b+2]+SeC[0,2]) then
         Sup:=(ImageC[a,b+2]+SeC[0,2]);
        if Sup >(ImageC[a+1,b-2]+SeC[1,-2]) then
         Sup:=(ImageC[a+1,b-2]+SeC[1,-2]);
        if Sup >(ImageC[a+1,b-1]+SeC[1,-1]) then
         Sup:=(ImageC[a+1,b-1]+SeC[1,-1]);
        If Sup >(ImageC[a+1,b]+SeC[1,0]) then
         Sup:=(ImageC[a+1,b]+SeC[1,0]);
        If Sup > (ImageC[a+1,b+1]+SeC[1,1]) then
         Sup:=(ImageC[a+1,b+1]+SeC[1,1]);
        if Sup >(ImageC[a+1,b+2]+SeC[1,2]) then
         Sup:=(ImageC[a+1,b+2]+SeC[1,2]);
        if Sup >(ImageC[a+2,b-2]+SeC[2,-2]) then
         Sup:=(ImageC[a+2,b-2]+SeC[2,-2]);
        if Sup >(ImageC[a+2,b-1]+SeC[2,-1]) then
         Sup:=(ImageC[a+2,b-1]+SeC[2,-1]);
        if Sup >(ImageC[a+2,b]+SeC[2,0]) then
         Sup:=(ImageC[a+2,b]+SeC[2,0]);
        if Sup >(ImageC[a+2,b+1]+SeC[2,1]) then
         Sup:=(ImageC[a+2,b+1]+SeC[2,1]);
        If Sup >(ImageC[a+2,b+2]+SeC[2,2]) then
         Sup:=(ImageC[a+2,b+2]+SeC[2,2]);
        str(((1-Sup)-image[a,b])*255:3:0,Dilate[a,b]);
    if strtofloat(Dilate[a,b])>255then Dilate[a,b]:='255';
    if strtofloat(Dilate[a,b])<0then Dilate[a,b]:='0';
    Image2.Canvas.Pixels[a,b]:=RGB(strtoint(Dilate[a,b]),strtoint(Dilate[a,b]),strtoint(Dilate[a,b]));
```

```
end;
 }
end;
end;

procedure TForm1.Erosion4Click(Sender: TObject);
var a,b:integer;
   inf:real;
begin
SeComplement;
if dimem=0 then
begin
{
for a:=1 to long-1 do
for b:=1 to high-1 do
begin
if (Image[a-1,b-1]+SeC[-1,-1]) > (Image[a-1,b]+SeC[-1,0]) then
 Inf:=(Image[a-1,b]+SeC[-1,0])
else
 Inf:=(Image[a-1,b-1]+SeC[-1,-1]);
 if Inf> (Image[a-1,b+1]+SeC[-1,1]) then
 Inf:=(Image[a-1,b+1]+SeC[-1,1]);
if Inf >(Image[a,b-1]+SeC[0,-1]) then
 Inf:=(Image[a,b-1]+SeC[0,-1]);
if Inf >(Image[a,b]+SeC[0,0]) then
 Inf:=(Image[a,b]+SeC[0,0]);
if Inf >(Image[a,b+1]+SeC[0,1]) then
 Inf:=(Image[a,b+1]+SeC[0,1]);
if Inf>(Image[a+1,b-1]+SeC[1,-1]) then
Inf:=(Image[a+1,b-1]+SeC[1,-1]);
```

```
   Inf:=(Image[a+2,b-1]+SeC[2,-1]);
  if Inf >(Image[a+2,b]+SeC[2,0]) then
   Inf:=(Image[a+2,b]+SeC[2,0]);
  if Inf >(Image[a+2,b+1]+SeC[2,1]) then
   Inf:=(Image[a+2,b+1]+SeC[2,1]);
  if Inf >(Image[a+2,b+2]+SeC[2,2]) then
   Inf:=(Image[a+2,b+2]+SeC[2,2]);
   str((image[a,b]-Inf)*255:3:0,Erode[a,b]);
   if strtofloat(Erode[a,b])>255 then Erode[a,b]:='255';
if strtofloat(Erode[a,b])<0 then Erode[a,b]:='0'
Image2.Canvas.Pixels[a,b]:=RGB(strtoint(Erode[a,b]),strtoint(Erode[a,b]),strtoint(Erode[a,b]));
}
 end;
end;
procedure TForm1.theshold1Click(Sender: TObject);
var a,b,bin:integer;
begin
interval;
for a:=0 to long do
 for b:=0 to high do
  begin
   bin:=ColorToRgb(Image2.Canvas.Pixels[a,b]);
   if bin>(min+(intv/2)) then
   begin
    Image1.Canvas.Pixels[a,b]:=clwhite;
    Image2.Canvas.Pixels[a,b]:=clwhite;
   end
  else
   begin
    Image1.Canvas.Pixels[a,b]:=clblack;
```

```
   if Inf >(Image[a+1,b]+SeC[1,0]) then
    Inf:=(Image[a+1,b]+SeC[1,0]);
   if Inf >(Image[a+1,b+1]+SeC[1,1]) then
    Inf:=(Image[a+1,b+1]+SeC[1,1]);
    str((image[a,b]-inf)*255:3:0,Erode[a,b]);
If strtofloat(Erode[a,b])>255 then Erode[a,b]:='255';
If strtofloat(Erode[a,b])<0 then Erode[a,b]:='0';
Image2.Canvas.Pixels[a,b]:=RGB(strtoint(Erode[a,b]),strtoint(Erode[a,b]),strtoint(Erode[a,b]));
end;
 }
end;
if Dimem=1 then
begin
{
 for a:=2 to Long-2 do
  for b:=2 to High-2 do
   begin
   if (Image[a-2,b-2]+SeC[-2,-2]) > (Image[a-2,b-1]+SeC[-2,-1]) then
    Inf:=(Image[a-2,b-1]+SeC[-2,-1])
   else
    Inf:=(Image[a-2,b-2]+SeC[-2,-2]);
    if Inf > (Image[a-2,b]+SeC[-2,0]) then
     Inf:=(Image[a-2,b]+SeC[-2,0]);
    if Inf >(Image[a-2,b+1]+SeC[-2,1]) then
     Inf:=(Image[a-2,b+1]+SeC[-2,1]);
    if Inf >(Image[a-2,b+2]+SeC[-2,2]) then
     Inf:=(Image[a-2,b+2]+SeC[-2,2]);
    if Inf >(Image[a-1,b-2]+SeC[-1,-2]) then
     Inf:=(Image[a-1,b-2]+SeC[-1,-2]);
    if Inf >(Image[a-1,b-1]+SeC[-1,-1]) then
```

```
Inf:=(Image[a-1,b-1]+SeC[-1,-1]);
if Inf >(Image[a-1,b]+SeC[-1,0]) then
 Inf:=(Image[a-1,b]+SeC[-1,0]);
if Inf >(Image[a-1,b+1]+SeC[-1,1]) then
 Inf:=(Image[a-1,b+1]+SeC[-1,1]);
if Inf > (Image[a-1,b+2]+SeC[-1,2]) then
 Inf:=(Image[a-1,b+2]+SeC[-1,2]);
if Inf >(Image[a,b-2]+SeC[0,-2]) then
 Inf:=(Image[a,b-2]+SeC[0,-2]);
if Inf >(Image[a,b-1]+SeC[0,-1]) then
 Inf:=(Image[a,b-1]+SeC[0,-1]);
if Inf >(Image[a,b]+SeC[0,0]) then
 Inf:=(Image[a,b]+SeC[0,0]);
if Inf >(Image[a,b+1]+SeC[0,1]) then
 Inf:=(Image[a,b+1]+SeC[0,1]);
if Inf >(Image[a,b+2]+SeC[0,2]) then
 Inf:=(Image[a,b+2]+SeC[0,2]);
if Inf >(Image[a+1,b-2]+SeC[1,-2]) then
 Inf:=(Image[a+1,b-2]+SeC[1,-2]);
if Inf >(Image[a+1,b-1]+SeC[1,-1]) then
 Inf:=(Image[a+1,b-1]+SeC[1,-1]);
if Inf >(Image[a+1,b]+SeC[1,0]) then
 Inf:=(Image[a+1,b]+SeC[1,0]);
if Inf > (Image[a+1,b+1]+SeC[1,1]) then
 Inf:=(Image[a+1,b+1]+SeC[1,1]);
if Inf >(Image[a+1,b+2]+SeC[1,2]) then
 Inf:=(Image[a+1,b+2]+SeC[1,2]);
if Inf >(Image[a+2,b-2]+SeC[2,-2]) then
 Inf:=(Image[a+2,b-2]+SeC[2,-2]);
if Inf >(Image[a+2,b-1]+SeC[2,-1]) then
```

```
    Image2.Canvas.Pixels[a,b]:=clblack;
   end;
  end;
 end;
end.
```

เอกสารอ้างอิง

สาธิต อินทจักร์, 2538, การหาขอบภาพโดยใช้แบบจำลองฟัซซี่และนิวตรอลเน็ตเวิร์ค, วิทยานิพนธ์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

ศิริชัย ปริตโตทกพร, 2542, การแยกกลุ่มข้อมูลทางการแพทย์โดยฟัซซี่มีนท์ และ การพิจารณาฮิสโตแกรม, วิทยานิพนธ์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

J. Serra, 1982, Mathematical morphology, vol.1, Academic press, London.

D. Sinha and E. R. Dougherty, Joural of visual communication and image representation, Vol.3, No.3, September, pp. 286-302, 1992

Arnold Kaufmann and Madan M. Gupta, Fuzzy Mathematical Models In engineering and management Science, Elsevier science Publishers B.V. 1988, NY. 1988