เอกสารประกอบการสอน

วิชา หัวข้อขั้นสูงในภาษาคอมพิวเตอร์

Advanced Topics in Programming Languages

(423411)

หัวข้อ การประมวลผลภาพดิจิตอลด้วยโปรแกรม MATLAB

Topic: Digital Image Processing Using MATLAB

อาจารย์ สุภาพร บุญฤทธิ์

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สำนักวิชาวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีสุรนารี

คำนำ

เอกสารประกอบการสอนวิชา หัวข้อขั้นสูงในภาษาคอมพิวเตอร์ (Advanced Topics in Programming Languages) ในหัวข้อ การประมวลผลภาพดิจิตอลด้วยโปรแกรม MATLAB (Digital Image Processing Using MATLAB) เล่มนี้จัดทำขึ้นเพื่อเป็นเอกสารประกอบการสอนวิชา 423411 หัวข้อขั้นสูงในภาษาคอมพิวเตอร์ สำหรับนักศึกษาชั้นปีที่ 4 สาขาวิชา วิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จำนวน 4 หน่วยกิต ซึ่งมีคำอธิบายรายวิชาดังนี้

แนวคิดเบื้องต้นของการประมวลผลภาพดิจิตอล (Over View of Digital Image Processing) การใช้งานโปรแกรม MATLAB เบื้องต้น (Introduction to MATLAB) พื้นฐานการประมวลผลภาพดิจิตอลด้วยโปรแกรม MATLAB (Fundamental of Digital Image Processing Using MATLAB) การปรับปรุงคุณภาพของภาพในโดเมนระยะทาง (Image Enhancement in Spatial Domain) การประมวลผลทางสัญฐานของภาพ (Morphological Image Processing) การประมวลผลภาพสี (Color Image Processing) และ การแบ่งส่วนภาพ (Image Segmentation)

อาจารย์ สุภาพร บุญฤทธิ์
สาขาวิชาวิศวกรรมคอมพิวเตอร์
สำนักวิชาวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
19 เมษายน 2553

ประมวลการสอน

รหัสวิชา 423411	**ชื่อรายวิชา** Advanced Topics in Programming Languages **Topic:** Digital Image Processing Using MATLAB (DIPUM)
หน่วยกิต	4 (4-0-8) คือ บรรยาย 4 ปฏิบัติ 0 และศึกษาด้วยตนเอง 8
สังกัด	สาขาวิชาวิศวกรรมคอมพิวเตอร์, สำนักวิชาวิศวกรรมศาสตร์, มหาวิทยาลัยเทคโนโลยีสุรนารี
หมวดวิชา	วิชาบังคับเลือก
การเรียน	แบบ Lecture (C)
ผู้สอน	อาจารย์สุภาพร บุญฤทธิ์ **ห้องทำงาน:** อาคารวิชาการ ชั้น 4 ห้อง B02 **Contact:** sbunrit@yahoo.com, sbunrit@sut.ac.th

คำอธิบายรายวิชา

แนวคิดเบื้องต้นของการประมวลผลภาพดิจิตอล (Over View of Digital Image Processing) การใช้งานโปรแกรม MATLAB เบื้องต้น (Introduction to MATLAB) พื้นฐานการประมวลผลภาพดิจิตอลด้วยโปรแกรม MATLAB (Fundamental of Digital Image Processing Using MATLAB) การปรับปรุงคุณภาพของภาพในโดเมนระยะทาง (Image Enhancement in Spatial Domain) การประมวลผลทางสัญฐานของภาพ (Morphological Image Processing) การประมวลผลภาพสี (Color Image Processing) และ การแบ่งส่วนภาพ (Image Segmentation)

รูปแบบการเรียนการสอน

จะเป็นลักษณะการบรรยายหลักการในเชิงทฤษฎีของแต่ละหัวข้อตามด้วยตัวอย่างเขียนโปรแกรมเพื่อ implement หลักการนั้นๆ ด้วย MATLAB รวมถึงการใช้งานคำสั่งด้านการประมวลผลภาพดิจิตอล (Image Processing Toolbox) ใน MATLAB เพื่อสร้างตัวอย่างการประยุกต์ใช้ทฤษฎี ให้นักศึกษาเกิดความเข้าใจในทฤษฎีและการใช้งานมากขึ้น จากนั้นนักศึกษาจะต้องเขียนโปรแกรมประยุกต์ใช้งานเทคนิคต่างๆ เพื่อแก้ปัญหาที่กำหนด

วัตถุประสงค์ในการเรียนการสอน

1. เพื่อให้ผู้เรียนเข้าใจหลักการประมวลผลภาพดิจิตอลเบื้องต้นและพื้นฐานการประมวลผลภาพดิจิตอลด้วยโปรแกรม MATLAB
2. เพื่อให้ผู้เรียนเข้าใจแนวคิดและทฤษฎีในการปรับปรุงคุณภาพของภาพในโดเมนระยะทาง (Image Enhancement in Spatial Domain) การประมวลผลทางสัญฐานของภาพ (Morphological Image Processing) การประมวลผลภาพสี (Color Image Processing) และการแบ่งส่วนภาพ (Image Segmentation)
3. เพื่อให้ผู้เรียนสามารถใช้งานคำสั่งด้านการประมวลผลภาพดิจิตอล (Image Processing Toolbox) ใน MATLAB เพื่อสร้างตัวอย่างการประยุกต์ใช้ทฤษฎี ให้เกิดความเข้าใจ

แผนการเรียนการสอน

1. แนะนำรายวิชา (2 ชั่วโมง)
2. แนวคิดเบื้องต้นของการประมวลผลภาพดิจิตอล (2 ชั่วโมง)
3. การใช้งานโปรแกรม MATLAB เบื้องต้น (2 ชั่วโมง)
4. การใช้งานคำสั่งด้านการประมวลผลภาพดิจิตอลในโปรแกรม MATLAB (4 ชั่วโมง)
5. พื้นฐานการประมวลผลภาพดิจิตอลด้วยโปรแกรม MATLAB (4 ชั่วโมง)
6. การแปลงค่าระดับความเทาและการปรับปรุงคุณภาพของภาพในโดเมนระยะทาง (8 ชั่วโมง)
7. การประมวลผลทางสัญฐานของภาพ (8 ชั่วโมง)
8. การประมวลผลภาพสี (6 ชั่วโมง)
9. การแยกส่วนภาพ (8 ชั่วโมง)
10. ตัวอย่างการประยุกต์ใช้ทฤษฎีกับ Application ต่างๆ (4 ชั่วโมง)

การประเมินผล

เข้าเรียน	5%
การบ้าน	25% (งานเดี่ยว 3 ชิ้น งานกลุ่ม กลุ่มละ 4 คน 1 ชิ้น)
Random QUIZ	15% (ประมาณ 4-5 ครั้ง)
สอบกลางภาค	25%
สอบปลายภาค	30%
รวม	**100%**

Text (ใช้อ้างอิงเป็นหลักในการสอน)

Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing Using MATLAB*, 1st edition, Prentice Hall, 2004

เอกสารอ่านประกอบ

Rafael C. Gonzalez and Richard E. Woods, *Digital Image* Processing, 2nd Edition, Prentice Hall, 2002.

Alasdair McAndrew, *An Introduction to Digital Image Processing with MATLAB*, School of Computer Science and Mathematics, Victoria University of Technology. (Lecture Note)

สารบัญ

บทที่ 1: แนวคิดเบื้องต้นของการประมวลผลภาพดิจิตอล
Over View of Digital Image Processing

ในการนำคอมพิวเตอร์มาใช้งานยุคแรกๆ นั้นเป็นการนำคอมพิวเตอร์เข้ามาช่วยงานทางด้านวิทยาศาสตร์ ข้อมูล (Data) ที่เกี่ยวข้องจึงเป็นเพียงข้อมูลตัวเลข (Numerical Data) เท่านั้น แต่ในเวลาต่อมาเมื่อคอมพิวเตอร์เข้ามามีบทบาทในชีวิตประจำวันของมนุษย์มากขึ้น ข้อมูลที่เป็นตัวอักษร (Textual Data) จึงเข้ามาเกี่ยวข้อง ซึ่งในปัจจุบันนี้ข้อมูลที่ใช้ได้ในคอมพิวเตอร์มีหลากหลายรูปแบบ ได้แก่ ข้อมูลเสียง (Voice) อันได้แก่ เสียงเพลง (Music) หรือเสียงมนุษย์พูด (Speech) รวมทั้งข้อมูลรูปภาพ (Images) ทั้งที่มีต้นกำเนิดจากธรรมชาติ (Natural Images) และที่ถูกสร้างขึ้นด้วยโปรแกรมสร้างภาพกราฟิกด้วยคอมพิวเตอร์ (Computer Graphics) ทั้งนี้ทั้งนั้น ข้อมูลที่ใช้ในการประมวลผลด้วยคอมพิวเตอร์ถือรวมเรียกว่าเป็นสัญญาณดิจิตอล (Digital Signal) ทั้งสิ้น

การประมวลผลสัญญาณกับงานด้านต่างๆ
Relationship of Signal Processing to Other Fields

ตั้งแต่ที่มนุษย์เริ่มมีความพยายามที่จะส่งหรือรับข้อมูลผ่านสื่ออิเล็กทรอนิคส์ (Electronic Media) โทรเลข (Telegraph) โทรศัพท์ (Telephone) โทรทัศน์ (Television) เรดาร์ (Radar) และผ่านสื่ออื่นๆ มนุษย์เรียนรู้ว่า สัญญาณเหล่านี้สามารถถูกรบกวนได้ด้วยระบบ (System) ที่ใช้ในการส่ง รับ หรือ ประมวลผลสัญญาณ เนื่องจากในบางครั้งระบบอาจมีความไม่สมบูรณ์ ทำให้สัญญาณถูกรบกวนได้ด้วยสัญญาณรบกวน (Noise) หรือถูกบิดเบือน หรือผิดเพี้ยน (Distortion) ในแบบต่างๆ

การทำความเข้าใจผลกระทบที่เกิดจากระบบที่มีต่อสัญญาณและหาหนทางในการแก้ไขให้สัญญาณมีความถูกต้องนั้นเป็นพื้นฐานของทฤษฎีการประมวลผลสัญญาณ (Fundamental of Signal Processing) ในบางครั้งสัญญาณเหล่านี้คือข้อความ (Message) ที่มนุษย์เราสร้างขึ้นเพื่อส่งต่อไปยังผู้อื่น ตัวอย่างเช่น ข้อความที่ส่งผ่านระบบโทรเลข โทรศัพท์ โทรทัศน์ เครือข่ายคอมพิวเตอร์ และอื่นๆ เป็นต้น หลักการทั่วไปในการสื่อสารแบบนี้ก็คือ การส่งข้อมูลข่าวสาร (Information) แฝงไปในรูปแบบสัญญาณเพื่อส่งไปยังผู้รับ โดยผู้รับจะทำการแปลงสัญญาณที่ได้รับมาเพื่อนำเอาข้อมูลที่แฝงไว้ในสัญญาณมาใช้งาน

ในบางครั้งสัญญาณที่เรานำมาใช้งานคือ สัญญาณที่ถูกสร้างขึ้นโดยธรรมชาติ ได้แก่ สัญญาณเสียง (Audio Signal) สัญญาณภาพ (Image Signal) และอื่นๆ โดยเราสามารถนำเอาสัญญาณดังกล่าวเข้ามาใช้งานด้วยการใช้อุปกรณ์ในการรับสัญญาณ (Acquisition Equipment) เช่น ไมโครโฟน (Microphone) กล้องถ่ายภาพดิจิตอล (Digital Camera) หรือกล้องวีดีโอ (Video Camera) เป็นต้น แต่ในบางครั้งสัญญาณที่เรานำมาใช้งานนั้นถูกสร้างขึ้นด้วยโปรแกรมคอมพิวเตอร์ ตัวอย่างเช่น โปรแกรมสร้างสัญญาณเสียง โปรแกรมสร้างเสียงเพลง โปรแกรมสร้างภาพกราฟิก และ

ในบางครั้งมนุษย์นำเราสัญญาณทั้งสองประเภทมาใช้รวมกัน โดยรับเอาสัญญาณธรรมชาติเข้ามาด้วยอุปกรณ์รับสัญญาณแล้วทำการประมวลผลสัญญาณด้วยคอมพิวเตอร์ สร้างเป็นสัญญาณผลลัพธ์แล้วส่งผ่านผลลัพธ์นั้นไปยังผู้รับ

การประมวลผลภาพดิจิตอลคืออะไร
What is Digital Image Processing?)

การประมวลผลภาพดิจิตอล (Digital Image Processing) เป็นคลาสย่อย (subclass) ของการประมวลผลสัญญาณ (Signal Processing) นั่นคือ เป็นการประมวลผลสัญญาณที่มี input ของระบบเป็นภาพ (image) เท่านั้น

- วัตถุประสงค์ของการประมวลผลภาพดิจิตอลสามารถแบ่งออกได้เป็น 2 ประเภทใหญ่ๆ คือ
  - เพื่อปรับปรุงคุณภาพให้มนุษย์สามารถมองเห็นรายละเอียดของภาพได้ชัดเจนมากยิ่งขึ้น (Image Quality Improvement)
  - เพื่อให้คอมพิวเตอร์สามารถตีความภาพได้ (Computer Interpretation)

การประมวลผลภาพดิจิตอลเริ่มต้นจากอุตสาหกรรมหนังสือพิมพ์ ซึ่งเป็นการส่งข้อมูลภาพที่ถูกเข้ารหัส (Coding) ด้วยอุปกรณ์การพิมพ์พิเศษ ที่มีจุดประสงค์ในการลดเวลาส่งถ่ายข้อมูลผ่านสายเคเบิล โดยการส่งผ่านสายเคเบิลใต้น้ำครั้งแรกระหว่างเมืองลอนดอน ประเทศอังกฤษ ไปยังเมืองนิวยอร์ก ประเทศสหรัฐอเมริกา ทำให้ระยะเวลาในการส่งผ่านข่าวแทนการส่งแบบดั้งเดิมที่ใช้เวลาเป็นอาทิตย์กลายเป็นใช้เวลาน้อยกว่าสามชั่วโมง อุปกรณ์ปลายทางจะทำการถอดรหัสข้อมูล (Decoding) แล้วสร้างเป็นภาพเหมือนต้นฉบับขึ้นมา วิธีการเข้ารหัสที่ใช้เรียกว่า รูปแบบฮาฟโทน (Halftone Pattern)

หลังจากนั้นเทคโนโลยีการประมวลผลภาพดิจิตอลก็ถูกพัฒนาเพิ่มขึ้นเรื่อยๆ ประวัติการพัฒนาของเทคโนโลยีด้านนี้ผูกติดกับการพัฒนาดิจิตอลคอมพิวเตอร์เป็นอย่างมาก เนื่องจากการประมวลผลภาพนั้นมีความต้องการเนื้อที่ในการเก็บข้อมูลและความเร็วในการประมวลผลสูงทำให้การพัฒนาขึ้นอยู่กับความเร็วของหน่วยประมวลผลกลาง (Central processing Unit, CPU) หน่วยความจาสำรอง (Data Storage) และการส่งถ่ายข้อมูล (Data Transmission)

หลังจากที่เทคโนโลยีคอมพิวเตอร์ได้รุดหน้าถึงขั้นที่เราสามารถประมวลผลภาพได้อย่างรวดเร็ว ราวๆ ปี ค.ศ. 1960 ก็ได้มีการนำเอาทฤษฎีการประมวลผลภาพไปประยุกต์ใช้กับงานด้านอวกาศ ในปี ค.ศ. 1964 ได้มีการส่งผ่านภาพถ่ายดวงจันทร์ผ่านยานอวกาศ Ranger 7 เมื่อวันที่ 31 กรกฎาคม ค.ศ. 1964 โดยภาพที่ส่งมานั้นถูกประมวลผลด้วยคอมพิวเตอร์เพื่อทำการแก้ไขข้อมูลส่วนที่ผิดเพี้ยน อันเกิดขึ้นในขบวนการส่งผ่านภาพหรือรับภาพ หลังจากนั้นก็มีการพัฒนาเทคโนโลยีการ

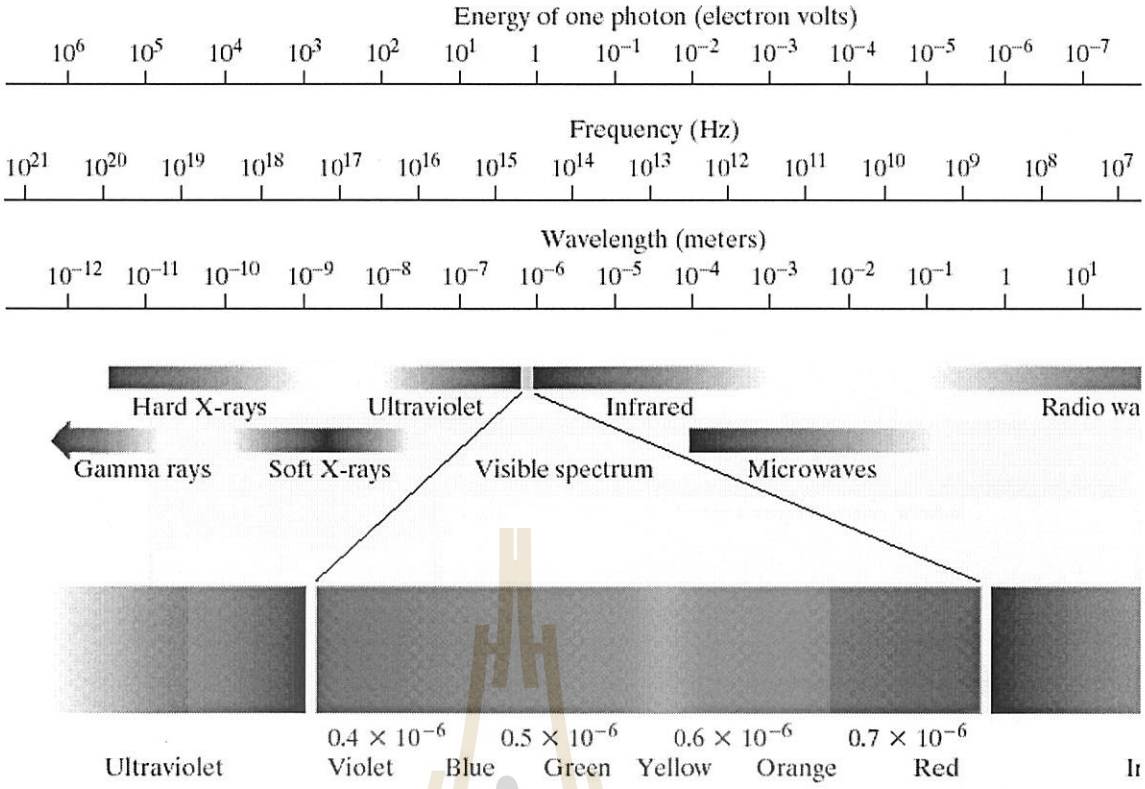
ประมวลผลภาพขึ้นเรื่อยๆ รวมทั้งได้มีการนำเทคโนโลยีใหม่ๆ มาใช้ในโครงการสำรวจดวงจันทร์ ดาวอังคาร และอื่นๆ

ในขณะเดียวกันกับที่มีการนำเทคโนโลยีการประมวลผลภาพดิจิตอลไปใช้กับโตรงการด้านอวกาศนั้น ก็ได้มีการนำเทคโนโลยีด้านนี้ไปใช้กับงานด้านภาพทางการแพทย์ (Medical Imaging) ด้วย รวมทั้งงานทางด้านการสำรวจโลก โดยการรักษาทางการแพทย์โดยอาศัยภาพ (Medical Diagnosis) นั้นเข้ามามีบทบาทอย่างสูงเมื่อมีการสร้างอุปกรณ์ที่เรียกว่า เครื่องซีที (Computerized Tomography, CT) ซึ่งสามารถทำการถ่ายเนื้อเยื่อภาพตัดขวาง 3 มิติได้

แหล่งกำเนิดของภาพดิจิตอล
Source of Digital Images

ในปัจจุบันนี้ทฤษฎีการประมวลผลภาพดิจิตอลและการใช้งานนั้นได้พัฒนาก้าวหน้าไปอย่างมาก ซึ่งสาขางานที่มีการนำเทคโนโลยีการประมวลผลภาพดิจิตอลไปใช้งานนั้นสามารถแบ่งออกได้ตามแหล่งกำเนิดของภาพ ดังต่อไปนี้ คือ

- ภาพที่เกิดจากคลื่นความถี่แม่เหล็กไฟฟ้าในแถบสเปกตรัมต่างๆ (Electromagnetic Spectrum)
  - คลื่นแถบความถี่แกรมมา (Gramma rays)
  - คลื่นแถบความถี่เอกซ์ (X-rays)
  - คลื่นแถบความถี่อัลตราไวโอเลต (Ultraviolet)
  - คลื่นแถบความถี่มองเห็น (Visible)
  - คลื่นแถบความถี่อินฟราเรด (Infrared)
  - คลื่นแถบความถี่ไมโครเวฟ (Microwaves)
  - คลื่นแถบความถี่วิทยุ (Radio waves)
- ภาพที่เกิดจากสัญญาณในช่วงคลื่นความถี่เสียง (Acoustic)
- ภาพที่เกิดจากสัญญาณในช่วงความถี่อัลตราโซนิค (Ultrasonic)
- ภาพที่เกิดจากเครื่องอิเล็กทรอนิกส์ที่ใช้อนุภาคอิเล็กตรอน (Electronic in the form of electron beams) ซึ่งใช้ในกล้องจุลทรรศน์แบบอิเล็กตรอน (Electron microscopy)
- ภาพที่เกิดจากการสังเคราะห์ด้วยคอมพิวเตอร์ (Computer's synthetic images)



รูปที่ 1.1 คลื่นความถี่แม่เหล็กไฟฟ้าในแถบสเปกตรัมต่างๆ

สาขาที่เกี่ยวข้องกับการประมวลผลภาพดิจิตอล

- การสื่อสารแบบดิจิตอล (Digital Communication) : เป็นสาขาที่เกี่ยวกับการส่งข้อมูลผ่านสื่อ
- การบีบอัดข้อมูล (Data Compression) : เป็นสาขาที่เกี่ยวกับการเข้ารหัส ถอดรหัส และบีบอัดข้อมูล
- คอมพิวเตอร์กราฟิก (Computer Graphics) : เป็นสาขาที่เกี่ยวกับการสังเคราะห์ภาพขึ้นมาใหม่ด้วยคอมพิวเตอร์
- การประมวลผลภาพ (Image Processing) : เป็นสาขาที่เกี่ยวกับการปรับปรุงคุณภาพของภาพ ซึ่งผลลัพธ์ที่ได้คือภาพที่มีคุณภาพเปลี่ยนไป
- คอมพิวเตอร์วิชัน (Computer Vision) : เป็นสาขาที่เกี่ยวกับการวิเคราะห์ข้อมูลภายในภาพ

บทที่ 2: แนะนำการใช้งานโปรแกรม MATLAB
Introduction to MATLAB

2.1 การใช้งานโปรแกรม MATLAB เบื้องต้น

- เปิดใช้งานโปรแกรม MATLAB จาก Windows
  - Double-Click ไอคอนบน Windows desktop จะได้ MATLAB desktop ดังรูปด้านล่าง

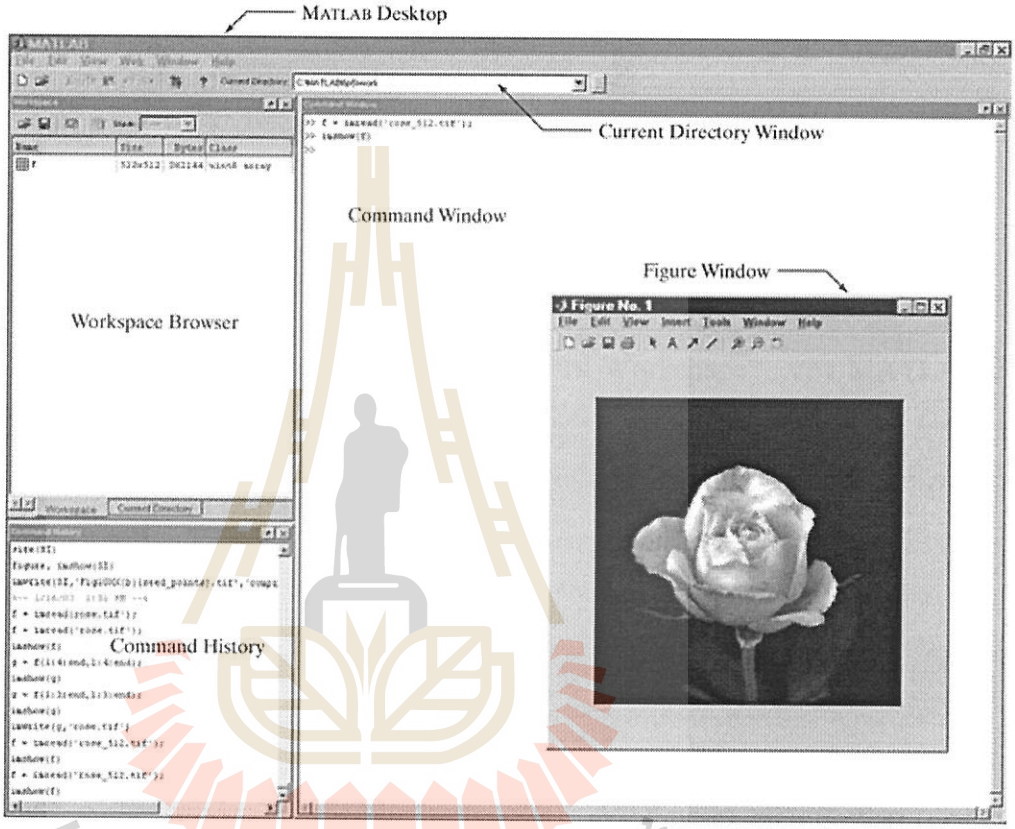


FIGURE 1.1 The MATLAB desktop and its principal components.

  - ตรงส่วน Command Window จะมี prompt พิเศษเป็นลักษณะ >> ปรากฏอยู่ หมายความว่าโปรแกรมกำลังรอรับคำสั่ง (command) อยู่ผ่าน prompt นี้

- การออกจากโปรแกรม MATLAB
  - เลือก **Exit MATLAB** จากเมนู **File** หรือ
  - พิมพ์ quit หรือ exit ตรง prompt ของ Command Window นั่นคือ

    >> quit

    หรือ

    >> exit

Note: ไม่แนะนำให้ปิดโปรแกรมโดย click ที่ close box ตรงมุมบนสุดด้านขวาของ MATLAB desktop

- ทดลองใช้การคำนวณทางเลขคณิตต่อไปนี้

Note: ในเอกสารนี้เมื่อใช้สัญลักษณ์ <Enter> หมายถึงให้กดปุ่ม Enter

สำหรับ MATLAB การกดปุ่ม Enter คือการสั่งให้ run หรือ execute คำสั่ง

- พิมพ์ 2+3 หลัง >> prompt ใน Command Window แล้วกดปุ่ม Enter นั่นคือ

```
>> 2+3                <Enter>
```

จะพบว่าผลลัพธ์ที่ได้คือ

```
ans =

    5
```

Note: ans เป็น default variable ของ MATLAB จะเก็บค่าล่าสุดที่ได้จากการคำนวณเสมอ

สังเกตว่าจะมีตัวแปร ans ปรากฏอยู่ในส่วน Workspace ของ MATLAB desktop

- ทดลองพิมพ์ต่อไปดังต่อไปนี้

```
>> 3-2                <Enter>
>> 2*3                <Enter>
>> 1/2                <Enter>
>> 2^3                <Enter>
>> 2\1                <Enter>

>> 3*4                <Enter>
>> 3*4;               <Enter>
>> ans                <Enter>
>> ans;               <Enter>
```

Note: การใส่ ; (semicolon) ไว้ที่ตอนท้ายสุดของแต่ละคำสั่ง คือไม่ต้องการให้แสดงค่าของผลลัพธ์ออกมาทางหน้าจอ

- ทดลองใช้ฟังก์ชันพื้นฐานทางคณิต

```
>> sqrt(2)            <Enter>
ans =

   1.4142

>> log2(16)           <Enter>
ans =
```

```
    4

>> log10(100)          <Enter>
ans =

    2

>> sin(pi/4)           <Enter>
ans =

   0.7071

>> sqrt(2)/2           <Enter>
ans =

   0.7071

>> format long         <Enter>
>> sqrt(2)/2           <Enter>
ans =

  0.70710678118655

>> format              <Enter>
>> sqrt(2)/2           <Enter>
ans =

   0.7071
```

Note: การคำนวณใน MATLAB จะเป็น double precision แต่ในการแสดงค่าจะ default เป็นทศนิยมแปดตำแหน่ง ถ้าต้องการให้แสดงค่าในลักษณะ double precision สามารถใช้คำสั่ง format long และเมื่อใช้คำสั่ง format (หรือ format short) ก็จะกลับไปแสดงค่าแบบ default

2.2 Variable และ Workspace

- ทดลองกำหนดค่าให้กับตัวแปร (variable) เพื่อทำการคำนวณดังตัวอย่างต่อไปนี้

```
>> a = 2;              <Enter>
>> b = 5^(3/2)         <Enter>
>> a + b               <Enter>
>> c = a + b;          <Enter>
>> c                   <Enter>
```

Note: การพิมพ์ชื่อของตัวแปร แล้วกด Enter จะแสดงค่าของตัวแปรตัวนั้นออกมา

- ตัวแปรทั้งหมดที่เรากำลังใช้งานจะแสดงอยู่ในส่วนของ Workspace ซึ่งเราสามารถใช้คำสั่งเรียกดูได้ที่ prompt เช่นกัน โดยใช้คำสั่ง who หรือ whos ดังตัวอย่าง

```
>> who                <Enter>

Your variables are:
a   ans  b   c

>> whos               <Enter>

Name      Size            Bytes  Class

a         1x1                8  double array
ans       1x1                8  double array
b         1x1                8  double array
c         1x1                8  double array
```

- สามารถใช้คำสั่ง clear ตามด้วยชื่อตัวแปรในการลบตัวแปรตัวแปรตัวนั้นออกจากระบบ หรือใช้คำสั่ง clear เฉยๆ หรือ clear all เพื่อลบตัวแปรทุกตัวออกไป ดังตัวอย่าง

```
>> clear a            <Enter>
>> who                <Enter>
Your variables are:

ans  b   c

>> clear all          <Enter>
>> who                <Enter>
```

- ชื่อตัวแปรใน MATLAB มีกฎคือ
  - ต้องประกอบด้วยตัวอักษร a-z, A-Z, ตัวเลข 0-9 และ underscore (_)
  - ต้องขึ้นต้นด้วยตัวอักษร
  - ความยาวเท่าไรก็ได้ แต่ MATLAB จะจำเพียง 31 ตัวแรก
  - เป็น Case sensitive (แตกต่างกันระหว่างพิมพ์ใหญ่และพิมพ์เล็ก) นั่นคือ balance, BALANCE และ BaLance ถือว่าเป็นตัวแปรที่ต่างกัน
  - ตัวอย่างชื่อตัวแปรที่ใช้ได้: r2d2 pay_day ProFit
  - ตัวอย่างชื่อตัวแปรที่ใช้ไม่ได้: pay-day 2a name$ _Profit

2.3 Arrays: Vectors และ Matrices

- MATLAB นั้นถูกออกแบบมาในลักษณะของ matrices
- A matrix คือ rectangular object (a table) ที่ประกอบด้วย rows และ columns
- A vector นั้นก็เป็น matrix ที่มีเพียง 1 row หรือ 1 column

- MATLAB จะจัดการกับ vectors และ matrices ในลักษณะเดียวกัน

➢ การสร้าง row vectors

```
>> x = [1 3 0 -1 5]          <Enter>
>> y = [5, 6, 7]             <Enter>
>> z = [130 -15]

>> a = [1 2 3]               <Enter>
>> b = [4 5]                 <Enter>
>> c = [a -b]                <Enter>

>> A = [1 3 7]               <Enter>
>> A = [A 0 -1]              <Enter>
```

Note: ใช้ [], ใช้ spaces หรือ commas ในการแยกสมาชิกแต่ละตัว

➢ การสร้าง vectors ด้วย colon operator

```
>> v1 = 1 : 10               <Enter>
>> v2 = 1 :  0.5 : 4         <Enter>
>> v3 = 10 : -1 : 1          <Enter>
>> v4 = 1 : 2 : 6            <Enter>
>> v5 = 0 : -2 : -5          <Enter>
```

➢ การสร้าง column vectors

```
>> x1 = [1 3 0 -1 5] '       <Enter>
>> y1= [5, 6, 7] '           <Enter>
>> z1 = [130 -15] '

>> x2 = [1; 3; 0; -1; 5]     <Enter>
>> y2 = [5; 6; 7]            <Enter>
>> z2 = [130; -15]           <Enter>

>> a = [1 2 3]               <Enter>
>> b = [4 5]                 <Enter>
>> c = [a -b] '              <Enter>
```

Note: ใช้การ transpose จาก row vector หรือใช้ semicolon ในการแยกสมาชิกแต่ละตัว

➢ การสร้าง Matrices

```
>> A = [1  2 3; 4 5 6]       <Enter>
A =

1    2    3
4    5    6

>> B = [3 0 -1; 2 5 -7; -1 4 8] <Enter>
```

```
>> C = 0:30:80;
>> table = [C', sin(C*pi/180) ']

>>Z1 = [1:3; 4:6; 7:9]
>>Z2 = Z1'
```

- ตัวอย่าง standards arrays (matrices) in MATLAB
  - zeros(M, N) generates an M-by-N matrix of zeros.
  - ones(M, N) generates an M-by-N matrix of ones.
  - true(M, N) generates an M-by-N matrix of logical ones.
  - false(M, N) generates an M-by-N matrix of logical zeros.
  - magic(N) generates an N-by-N matrix constructed from the integers 1 through N^2 with equal row, column, and diagonal sums.
  - rand(M, N) generates an M-by-N matrix with random entries, chosen from a uniform distribution on the interval (0.0,1.0).

Notes: *Wording ใน MATLAB*

Scalar: ตัวแปรที่มีขนาดของ array เป็น 1-by-1 (1X1)

Vector: ตัวแปรที่มีขนาดของ array เป็น 1-by-N (row vector) หรือ M-by-1 (column vector)

Matrix: ตัวแปรที่มีขนาดของ array เป็น M-by-N

นั่นคือทุกอย่างใน MATLAB มองเป็นลักษณะของ array (หรือ matrices)

2.4 การจัดการกับ Matrix

- การอ้างถึงสมาชิกใน matrix

```
>> a = [4 -2 -4 7; 1 5 -3 2; 6 -8 -5 -6; -7 3 0 1]        <Enter>
a =

     4    -2    -4     7
     1     5    -3     2
     6    -8    -5    -6
    -7     3     0     1

>>

>> a(2,3)
ans =
          -3


>> a(4,1)
ans =
          -7
```

```
>> a(end,1)
ans =
        -7

>> a(end:end)
ans =
         1

>> a(2,1:3)
ans =
         1     5    -3

>> a(2:4,3)
ans =
        -3
        -5
         0

>> a(2:3,3:4)
ans =
        -3     2
        -5    -6

>> a(3,:)
ans =
         6    -8    -5    -6

>> a(:,2)
ans =
        -2
         5
        -8
         3

>> a(:,:)

ans =

         4    -2    -4     7
         1     5    -3     2
         6    -8    -5    -6
        -7     3     0     1


>> a(1)
        ans =

                4

>> a(end)
```

```
ans =
     1

>> a(10)
ans =
    -3

>> 2:3:16
ans =
     2     5     8    11    14

>> a(2:3:16)
ans =
     1    -2     3    -5     2


>> position = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]'
position =
     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16
```

➢ การดำเนินการบน Matrix

```
>> a = [4 -2 -4 7; 1 5 -3 2; 6 -8 -5 -6; -7 3 0 1]
a =
     4    -2    -4     7
     1     5    -3     2
     6    -8    -5    -6
    -7     3     0     1

>> b = [2 4 -7 -4; 5 6 3 -2; 1 -8 -5 -3; 0 -6 7 -1]
b =
     2     4    -7    -4
     5     6     3    -2
     1    -8    -5    -3
     0    -6     7    -1

>> 2*a
ans =
     8    -4    -8    14
     2    10    -6     4
    12   -16   -10   -12
   -14     6     0     2

>> 2*a-3*b
ans =
     2   -16    13    26
   -13    -8   -15    10
```

```
  9    8    5   -3
-14   24  -21    5

>> a*b

ans =

   -6   -6   35   -7
   24   46   37   -7
  -33   52  -83   13
    1  -16   65   21

>> a.*b

ans =

    8   -8   28  -28
    5   30   -9   -4
    6   64   25   18
    0  -18    0   -1

>> a^2

ans =

  -59   35   10   55
  -23   53   -4   37
   28  -30   25   50
  -32   32   19  -42

>> a.^2

ans =

   16    4   16   49
    1   25    9    4
   36   64   25   36
   49    9    0    1

>> 1/a
??? Error using ==> /
Matrix dimensions must agree.

>> 1./a
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this warning.)

ans =

    0.2500   -0.5000   -0.2500    0.1429
```

```
    1.0000    0.2000   -0.3333    0.5000
    0.1667   -0.1250   -0.2000   -0.1667
   -0.1429    0.3333       Inf    1.0000
```

2.5 การ Plot ใน MATLAB

```
>> x = [0:0.1:2*pi];
>> plot(x, sin(x))
>> plot(x, sin(x),'.',x,cos(x),'o')

>> help plot
```

บทที่ 3: พื้นฐานการประมวลผลภาพดิจิตอลด้วยโปรแกรม MATLAB
Fundamental of Digital Image Processing Using MATLAB

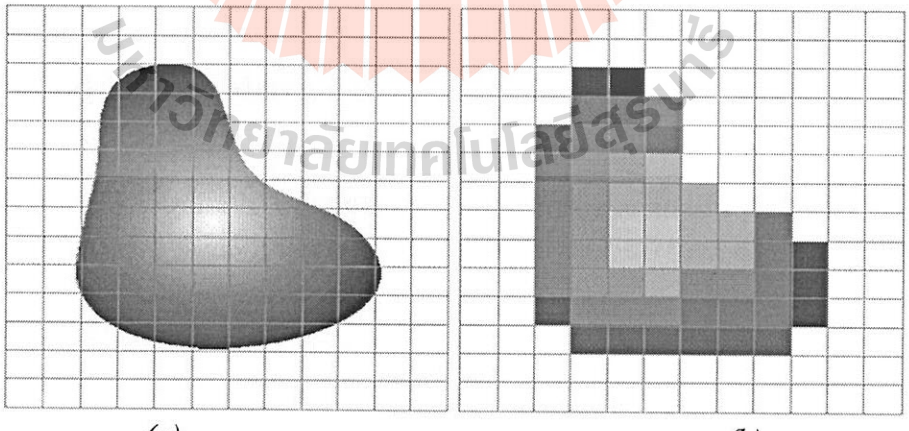
3.1 การแทนรูปภาพดิจิตอล (Digital Image Representation)

An image may be defined as a two-dimensional function, $f(x, y)$, where x and y are *spatial* (plane) *coordinates*, and the amplitude of f at any pair of coordinates (x, y) is called the *intensity* of the image at that point.

รูปภาพสามารถนิยามได้ในรูปของฟังก์ชัน $f(x, y)$ ใน 2 มิติเมื่อ x และ y เป็นของค่าระยะทางในแนวแกน x และแกน y และค่า amplitude ของ f ณ แต่ละพิกัด (x, y) เรียกว่า intensity ของภาพ ณ ตำแหน่งนั้นๆ

An image may be continuous with respect to the x- and y-coordinates, and also in amplitude. Converting such an image to digital form requires that the coordinates, as well as the amplitude, be digitized. Digitizing the coordinate values is called *sampling*; digitizing the amplitude values is called *quantization*. Thus, when x, y, and the amplitude values of f are all finite, discrete quantities, we call the image a *digital image*

จากลักษณะของพิกัด (x, y) และค่าของ amplitude แล้วรูปภาพถือว่ามีลักษณะที่ต่อเนื่อง ซึ่งการแปลงรูปภาพที่ต่อเนื่องไปเป็นแบบดิจิตอลต้องทำการดิจิไทซ์ (digitized) ทั้งค่าของพิกัดจุด (coordinates) และค่าของ amplitude การ ดิจิไทซ์ ค่าของพิกัดจุด (coordinates) นั้นเรียกว่าการ *sampling* ส่วนการ ดิจิไทซ์ ค่าของ amplitude เรียกว่าการ *quantization* ดังนั้นเมื่อ ค่า x, y, และค่า amplitude ของ f มีขนาดที่จำกัด นั่นคือเป็นค่าแบบดิสตรีต เราจึงเรียกรูปภาพที่ผ่านการ ดิจิไทซ์แล้วว่า *digital image* ดังตัวอย่างในรูปที่ 3.1(b) ซึ่งได้มาจากการ *sampling และการ quantization* ภาพที่ต่อเนื่องในรูปที่ 3.1(a)



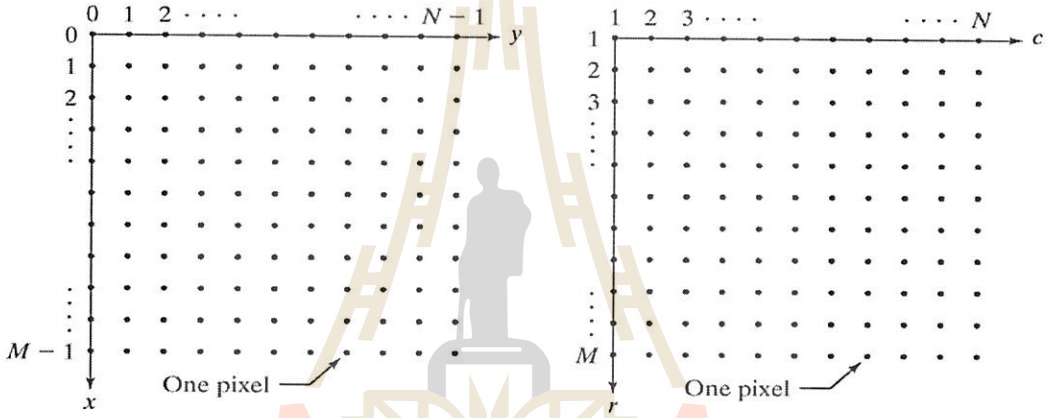
(a) *(b)*

รูปที่ 3.1 (a) ภาพที่มีลักษณะต่อเนื่อง (b) ภาพดิจิตอลที่ได้จากการ sampling และการ quantization

➢ **ข้อตกลงเกี่ยวกับรูปแบบของพิกัด (Coordinate Conventions)**

ผลลัพธ์ที่ได้จากการ sampling และ quantization รูปภาพนั้นถือว่าเป็นเมตริกซ์ของตัวเลขสมมุติรูปภาพ *f(x, y)* ผ่านการ sampling มาแล้วมีขนาด *M* แถว และ *N* คอลัมน์ นั่นคือเป็นรูปภาพที่มีขนาด *M* x *N* โดยค่าของพิกัดจุด *(x,y)* เป็นค่าแบบดิสครีต หนังสือเกี่ยวกับ image processing โดยทั่วไปนั้น กำหนดให้พิกัดเริ่มต้นของรูปภาพอยู่ที่ตำแหน่ง *(x, y)* = (0,0) ดังนั้นช่วงของค่าพิกัดในแกน *x* จึงอยู่ที่ 0 ถึง *M*-1 และ ช่วงของค่าพิกัดในแกน *y* อยู่ที่ 0 ถึง *N-1* ดังรูปที่ 3.2(a)

แต่รูปแบบของพิกัดจุกที่ใช้ใน Image Processing Toolbox (IPT) ของ MATLAB นั้นแตกต่างกันออกไปคือ ใช้สัญลักษณ์ *(r,c)* แทนแทนที่จะใช้ *(x,y)* โดย *r* แทน row และ *c* แทน column และกำหนดให้พิกัดเริ่มต้นของรูปภาพอยู่ที่ตำแหน่ง *(r, c)* = (1,1) แทน ช่วงของค่าพิกัดในแนว *r* จึงอยู่ที่ 0 ถึง *M* และช่วงของค่าพิกัดในแนว *c* อยู่ที่ 0 ถึง *N* ดังแสดงในรูปที่ 3.2(b)



(a) *(b)*

รูปที่ 3.2 (a) รูปแบบของพิกัดที่ใช้ในหนังสือ image processing ทั่วไป (b) รูปแบบของพิกัดที่ใช้ใน Image Processing Toolbox (IPT) ของ MATLAB

➢ **การแทนรูปภาพด้วย Matrices (Images as Matrices)**

จากรูปแบบของพิกัดที่ใช้ในหนังสือ image processing ทั่วไปสอดคล้องตามในรูป 3.1(a) นั้นเมื่อนำค่าของ amplitude *f(x,y)* ของแต่ละพิกัดมาแทนด้วย matrix จึงแสดงได้ดังด้านล่าง

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

ซึ่งสมาชิกแต่ละตัวใน matrix นั้นเรียกว่า *element, picture element, pixel,* หรือ *pel.*

ส่วนรูปแบบของพิกัดที่ใช้ใน MATLAB ซึ่งสอดคล้องตามในรูป 3.1(b) นั้นเมื่อนำค่าของ amplitude *f(x,y)* ของแต่ละพิกัดมาแทนด้วย matrix จึงแสดงได้ดังด้านล่าง

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix}$$

โดยที่ค่า ณ ตำแหน่ง $f(0,0)$ ที่อ้างอิงในหนังสือ image processing ทั่วไปจะเท่ากันกับค่า ณ ตำแหน่ง f(1,1) ของ MATLAB

3.2 การอ่านไฟล์รูปภาพเข้ามาใน MATLAB (Reading Images)

➢ ใช้ฟังก์ชัน imread ในการอ่านไฟล์รูปภาพเข้ามาใน MATLAB

Syntax:
imread('filename')

- ตัวอย่าง:

ถ้าภาพ rose_original.tif อยู่ใน current directory ใช้
>> f1 = imread('rose_original.tif');

ถ้าภาพ rose_original.tif ไม่ได้อยู่ใน current directory แต่อยู่ใน folder Myimages ของ drive C ใช้
>> f1 = imread('C:\Myimages\rose_original.tif');

ตัวอย่างข้างต้นเป็นการใช้คำสั่ง imread อ่าน file ภาพ ชื่อ rose_original.tif เข้ามาเก็บไว้ที่ตัว f1 ดังนั้นผลลัพธ์ที่ได้คือจะ มีตัวแปร f1 เกิดขึ้นใน Workspace ดังรูปแบบ

Name	Size	Bytes	Class
f1	1024x1024	1048576	uint8 array

ซึ่งข้อมูลที่แสดงใน Workspace ดังกล่าว สามารถใช้คำสั่ง whos ที่ Command Windows ได้เช่นกันดังรูปแบบ
>> whos f1
ผลลัพธ์ที่ได้คือ

Name	Size	Bytes	Class
f1	1024x1024	1048576	uint8 array

Grand total is 1048576 elements using 1048576 bytes

ถ้าต้องการดูเฉพาะขนาดของภาพสามารถใช้คำสั่ง
>> size(f1)
ผลลัพธ์ที่ได้คือ
ans =
1024 1024

ถ้าต้องการเอาขนาดของภาพมาเก็บไว้ที่ตัวแปร M และ N เมื่อ M คือแถว และ N คือคอลัมน์ สามารถใช้คำสั่ง

```
>> [M N] = size(f1)
```

ผลลัพธ์ที่ได้คือ

```
M =
     1024
N =
     1024
```

3.3 การแสดงผลรูปภาพ (Displaying Images)

- ใช้ฟังก์ชัน imshow ในการแสดงผลรูปภาพออกหน้าจอหลังจากได้อ่านไฟล์รูปนั้นเข้ามาแล้วด้วยฟังก์ชัน imread

Syntax:
imshow(f,G)
เมื่อ f คือตัวแปรที่เก็บข้อมูลภาพ (image array) และ G คือ ค่าตัวเลขที่บอกว่าต้องการให้แสดงผลออกมาด้วยค่าสีกี่ระดับ (intensity level)

Syntax:
imshow(f)
เป็นการแสดงผลโดยใช้ค่า G เป็นค่า default นั่นคือ 256 ระดับ

Syntax:
imshow(f, [low high])
เป็นการแสดงผลโดยกำหนดให้เป็นสีดำ (ค่า = 0) ในทุกจุดที่มีค่าน้อยกว่าหรือเท่ากับ low และสีขาว (ค่า = 255) ในทุกจุดที่มีค่ามากกว่าหรือเท่ากับ high

Syntax:
imshow(f, [])
เป็นการแสดงผลโดย ให้ค่า low คือค่าต่ำสุดของ array f และค่า high คือค่าสูงสุดของ array f

- ตัวอย่าง:

```
>> imshow(f1, 2)
>> imshow(f1, 10)
>> imshow(f1, 256), figure, imshow(f1)
>> imshow(f1, [100 150])
>> imshow(f1, [])
```

- ทดลองพิมพ์คำสั่งต่อไปนี้ แล้วสังเกตผลลัพธ์:

```
>> g1 = imread('chest_xray.tif');
>> imshow(g1), figure, imshow(g1, [])
```

3.4 การเขียนไฟล์รูปภาพ (Writing Images)

➢ ใช้ฟังก์ชัน imwrite ในการขียนไฟล์รูปภาพ (เมื่อต้องการ save ไฟล์รูปภาพเก็บไว้)

Syntax:
imwrite(f,'filename')

- ตัวอย่าง:

```
>> imwrite(f1, 'newrose.jpg');
>> imfinfo('rose_original.tif')
>> imfinfo('newrose.jpg')
```

3.5 Data Classes

Lists the various data classes supported by MATLAB and IPT for representing pixel values

Name	Description
`double`	Double-precision, floating-point numbers in the approximate range -10^{308} to 10^{308} (8 bytes per element).
`uint8`	Unsigned 8-bit integers in the range [0, 255] (1 byte per element).
`uint16`	Unsigned 16-bit integers in the range [0, 65535] (2 bytes per element).
`uint32`	Unsigned 32-bit integers in the range [0, 4294967295] (4 bytes per element).
`int8`	Signed 8-bit integers in the range [−128, 127] (1 byte per element).
`int16`	Signed 16-bit integers in the range [−32768, 32767] (2 bytes per element).
`int32`	Signed 32-bit integers in the range [−2147483648, 2147483647] (4 bytes per element).
`single`	Single-precision floating-point numbers with values in the approximate range -10^{38} to 10^{38} (4 bytes per element).
`char`	Characters (2 bytes per element).
`logical`	Values are 0 or 1 (1 byte per element).

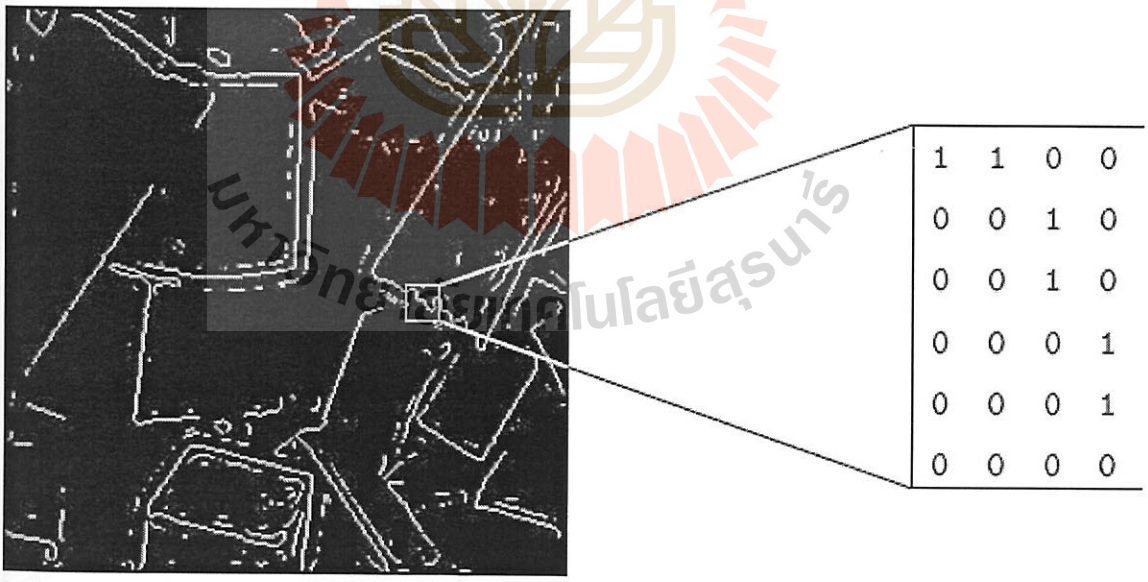
3.6 Image Types

สามารถแบ่งชนิดของรูปภาพดิจิตอลออกเป็น 4 ชนิดด้วยกันคือ Binary image, Intensity image, RGB image และ Indexed image

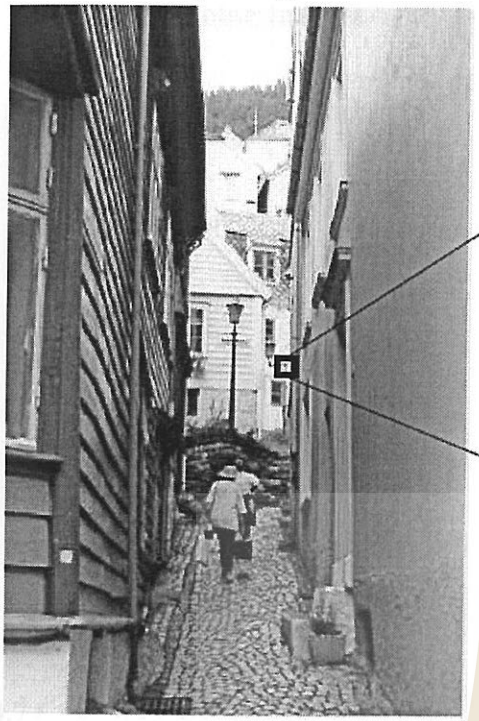
Image Type	**Storage Class**	**Interpretation**
Binary	logical	An array of zeros (0) and ones (1)
Intensity[1]	double	An array of floating-point values. The typical range of values is [0, 1].
	uint8 or uint16	An array of integers. The typical range of values is [0, 255] or [0, 65535].
Indexed[1]	double	An array of integers in the range [1, p]
	uint8 or uint16	An array of integers in the range [0, p-1]
RGB (Truecolor)	double	An m-by-n-by-3 array of floating-point values in the range [0, 1].
	uint8 or uint16	An m-by-n-by-3 array of integers in the range [0, 255] or [0, 65535].

➢ **Binary Images**

Binary images have a very specific meaning in MATLAB. A binary image is a logical array of 0s and Is. Thus, an array of 0s and Is whose values are of data class, say, uint8. is not considered a binary image in MATLAB.



➢ **Intensity Images**



230	229	232	234	235	232
237	236	236	234	233	234
255	255	255	251	230	236
99	90	67	37	94	247
222	152	255	129	129	246
154	199	255	150	189	241
216	132	162	163	170	239

➢ **RGB Images**

49	55	56	57	52	53
58	60	60	58	55	57
58	58	54	53	55	56
83	78	72	69	68	69
88	91	91	84	83	82
69	76	83	78	76	75
61	69	73	78	76	76

Red

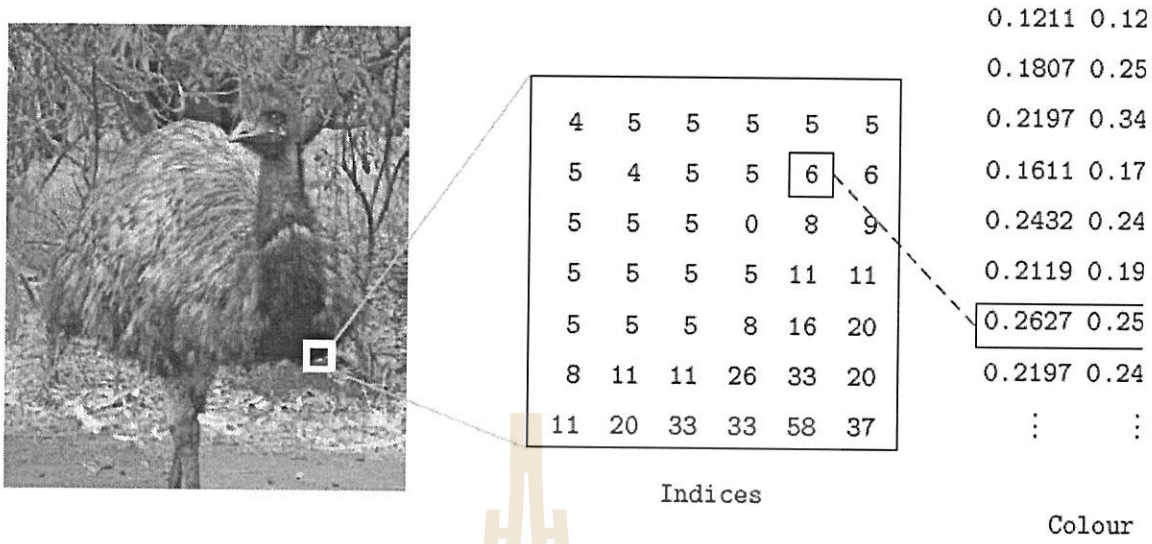
64	76	82	79	78	78
93	93	91	91	86	86
88	82	88	90	88	89
125	119	113	108	111	110
137	136	132	128	126	120
105	108	114	114	118	113
96	103	112	108	111	107

Green

66	80	77	80	87	
81	93	96	99	86	
83	83	91	94	92	
135	128	126	112	107	1
141	129	129	117	115	1
95	99	109	108	112	1
84	93	107	101	105	1

Blue

➢ **Index Color Images**



3.7 Converting between Data Classes and Image Types

➢ **Converting between Data Classes**

Name	Converts Input to:	Valid Input Image Data Classes
`im2uint8`	`uint8`	`logical`, `uint8`, `uint16`, and `double`
`im2uint16`	`uint16`	`logical`, `uint8`, `uint16`, and `double`
`mat2gray`	`double` (in range [0, 1])	`double`
`im2double`	`double`	`logical`, `uint8`, `uint16`, and `double`
`im2bw`	`logical`	`uint8`, `uint16`, and `double`

➢ **Converting between Image Classes and Types**

Function	Description
dither	Create a binary image from a grayscale intensity image by dithering; create an indexed image from an RGB image by dithering
gray2ind	Create an indexed image from a grayscale intensity image
grayslice	Create an indexed image from a grayscale intensity image by thresholding
im2bw	Create a binary image from an intensity image, indexed image, or RGB image, based on a luminance threshold
ind2gray	Create a grayscale intensity image from an indexed image
ind2rgb	Create an RGB image from an indexed image
mat2gray	Create a grayscale intensity image from data in a matrix, by scaling the data
rgb2gray	Create a grayscale intensity image from an RGB image
rgb2ind	Create an indexed image from an RGB image

3.8 Array Indexing

- ตัวอย่างการใช้ array indexing ใน MATLAB กับการเรียกดูข้อมูลภาพ:

```
>> f1 = imread('rose_original.tif');
   >> imshow(f1)

   >> f2 = f1(end:-1:1, :);
   >> figure, imshow(f2)

   >> f3 = f1(257:768, 257:768);
   >> figure, imshow(f3)

   >> f4 = f1(1:2:end, 1:2:end);
   >> figure, imshow(f4)

   >> f4 = f1(1:16:end, 1:16:end);
   >> figure, imshow(f4)

   >> plot(f1(512, :))
```

3.9 Operators in MATLAB

Table of array and matrix operators in MATLAB

Operator	Name	MATLAB Function	Comments and Examples
+	Array and matrix addition	plus(A, B)	a + b, A + B, or a + A.
–	Array and matrix subtraction	minus(A, B)	a – b, A – B, A – a, or a – A.
.*	Array multiplication	times(A, B)	C = A.*B, C(I, J) = A(I, J)*B(I, J).
*	Matrix multiplication	mtimes(A, B)	A*B, standard matrix multiplication, or a*A, multiplication of a scalar times all elements of A.
./	Array right division	rdivide(A, B)	C = A./B, C(I, J) = A(I, J)/B(I, J).
.\	Array left division	ldivide(A, B)	C = A.\B, C(I, J) = B(I, J)/A(I, J).
/	Matrix right division	mrdivide(A, B)	A/B is roughly the same as A*inv(B), depending on computational accuracy.
\	Matrix left division	mldivide(A, B)	A\B is roughly the same as inv(A)*B, depending on computational accuracy.
.^	Array power	power(A, B)	If C = A.^B, then C(I, J) = A(I, J)^B(I, J).
^	Matrix power	mpower(A, B)	See online help for a discussion of this operator.
.'	Vector and matrix transpose	transpose(A)	A.'. Standard vector and matrix transpose.
'	Vector and matrix complex conjugate transpose	ctranspose(A)	A'. Standard vector and matrix conjugate transpose. When A is real A.' = A'.
+	Unary plus	uplus (A)	+A is the same as 0 + A.
–	Unary minus	uminus (A)	–A is the same as 0 – A or –1*A.
:	Colon		Discussed in Section 2.8.

Table of relational operator

Operator	Name
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Table of logical operator

Operator	Name
&	AND
|	OR
~	NOT

Table of logical function

Function	Comments
`xor` (exclusive OR)	The `xor` function returns a 1 only if both operands are logically different; otherwise `xor` returns a 0.
`all`	The `all` function returns a 1 if all the elements in a vector are nonzero; otherwise `all` returns a 0. This function operates columnwise on matrices.
`any`	The any function returns a 1 if any of the elements in a vector is nonzero; otherwise any returns a 0. This function operates columnwise on matrices.

- ทดลองพิมพ์คำสั่งต่อไปนี้แล้วสังเกตผลลัพธ์:

```
>> A = rand(4,4)
>> A = A*10
>> A = round(A)
>> B = rand(4,4)
>> B = B*10
>> B = round(B)

>> C = A + B
>> D = A - B

>> E = A .* B
>> F = A * B

>> G = A ./ B
>> H = A / B

>> I = A .^2
>> J = A .^B
>> K = A^2

>> L = A'
>> M = A.'

>> N = -A
>> O =  +A

>> P = A>= 4
>> all(P)
>> any(P)

>> ~P
>> P | ~P
>> P & ~P
>> xor(P, ~P)
>> xor(P, P)
```

3.10 Image Arithmetic

Table of the image arithmetic supported by IPT in MATLAB

Function	Description
`imadd`	Adds two images; or adds a constant to an image.
`imsubtract`	Subtracts two images; or subtracts a constant from an image.
`immultiply`	Multiplies two images, where the multiplication is carried out between pairs of corresponding image elements; or multiplies a constant times an image.
`imdivide`	Divides two images, where the division is carried out between pairs of corresponding image elements; or divides an image by a constant.
`imabsdiff`	Computes the absolute difference between two images.
`imcomplement`	Complements an image. See Section 3.2.1.
`imlincomb`	Computes a linear combination of two or more images. See Section 5.3.1 for an example.

➢ **Adding Image:** imadd

Syntax:

Z = imadd(X,Y)

```
>> X = uint8([ 255 0 75; 44 225 100]);
>> Y = uint8([ 50 50 50; 50 50 50 ]);
>> Z = imadd(X,Y)
Z =
255   50  125
94  255  150
```

- ตัวอย่าง: Add a constant to an image.

```
>> I = imread('rice.tif');
>> J = imadd(I,100);
>> subplot(1,2,1), imshow(I)
>> subplot(1,2,2), imshow(J)
```

- ตัวอย่าง: Add two images together and specify an output class

```
>> I = imread('rice.tif');
>> imshow(I)

>> J = imread('cameraman.tif');
>> imshow(J)

>> K = imadd(I, J);
>> L = imadd(I, J, 'uint16');
>> imshow(K,[]), figure, imshow(L,[])
```

```
>> subplot(1,2,1); imshow(K,[]), xlabel('unit8');
>> subplot(1,2,2); imshow(L,[]), xlabel('uint16');
```

➢ **Subtracting Image:** imsubtract

Syntax:

```
Z = imsubtract(X,Y)
```

```
>> X = uint8([ 255 10 75; 44 225 100]);
>> Y = uint8([ 50 50 50; 50 50 50 ]);
>> Z = imsubtract(X,Y)
Z =
205   0   25
0   175   50
```

▪ ตัวอย่าง: Subtract a constant to an image.

```
>> I = imread('rice.tif');
>> J = imsubtract(I,85);
>> subplot(1,2,1), imshow(I)
>> subplot(1,2,2), imshow(J)
```

▪ ตัวอย่าง: Estimate and subtract the background of the rice image:

```
>> I = imread('rice.tif');
>> blocks = blkproc(I,[32 32],'min(x(:))');
>> background = imresize(blocks,[256 256],'bilinear');
>> Ip = imsubtract(I,background);
>> subplot(1,2,1); imshow(I,[]), xlabel('Before');
>> subplot(1,2,2); imshow(Ip,[]), xlabel('After');
```

➢ **Multiplying Images:** immultiply

Syntax:

```
Z = immultiply(X,Y)
```

```
>> I = imread('moon.tif');
>> J = immultiply(I,0.5);
>> subplot(1,2,1), imshow(I)
>> subplot(1,2,2), imshow(J)
```

➢ **Deviding Image:** imdivide

Syntax:

```
Z = imdivide(X,Y)
```

```
>> I = imread('rice.tif');
>> blocks = blkproc(I,[32 32],'min(x(:))');
>> background = imresize(blocks,[256 256],'bilinear');
>> Ip = imdivide(I,background);
>> imshow(Ip,[]) % [] = let imshow scale data automatically
```

➢ **Complement Image**: imcomplement
Syntax:
IM2 = imcomplement(IM)

▪ ตัวอย่าง: Create the complement of a uint8 array.

```
>> X = uint8([ 255 10 75; 44 225 100]);
>> X2 = imcomplement(X)
>> X2 =
    0   245   180
  211    30   155
```

▪ ตัวอย่าง: Reverse black and white in a binary image.

```
>> bw = imread('text.tif');
>> bw2 = imcomplement(bw);
>> subplot(1,2,1),imshow(bw)
>> subplot(1,2,2),imshow(bw2)
```

▪ ตัวอย่าง: Create the complement of an intensity image.

```
>> I = imread('bonemarr.tif');
>> J = imcomplement(I);
>> imshow(I), figure, imshow(J)
```

➢ **Absolute Different Image:** imabsdiff
Syntax:
Z = imabsdiff(X,Y)

```
>> X = uint8([ 255 10 75; 44 225 100]);
>> Y = uint8([ 50 50 50; 50 50 50 ]);
>> Z = imabsdiff(X,Y)
Z =
205   40   25
  6  175   50
```

▪ ตัวอย่าง: Display the absolute difference between a filtered image and the original.

```
>> I = imread('cameraman.tif');
>> J = uint8(filter2(fspecial('gaussian'), I));
>> K = imabsdiff(I,J);
>> imshow(K,[])
```

บทที่ 4: การปรับปรุงคุณภาพของภาพในโดเมนระยะทาง
Image Enhancement in Spatial Domain

เป้าหมายหลักของการปรับปรุงคุณภาพของภาพดิจิตอลนั้นคือการทำให้ภาพผลลัพธ์ที่ได้จากการปรับปรุงคุณภาพมีความเหมาะสมกว่าภาพต้นฉบับเพื่อที่จะนำไปใช้งานต่อในกระบวนการอื่นๆ ได้ ซึ่งในขั้นตอนของการปรับปรุงคุณภาพของภาพนั้นไม่ได้จำกัดจำนวนครั้งและเทคนิคที่จะนำมาใช้ อีกทั้งขั้นตอนในการปรับปรุงคุณภาพนั้นก็ไม่ได้กำหนดตายตัวว่าจะต้องเลือกใช้เทคนิคใดก่อนหรือหลัง ซึ่งถ้าลำดับของการใช้งานเทคนิคมีความแตกต่างกัน ภาพผลลัพธ์ที่ได้ก็จะแตกต่างกันไปด้วย ดังนั้นการจะเลือกใช้เทคนิคใดๆ จึงขึ้นอยู่กับดุลพินิจของผู้ใช้ทั้งสิ้น แน่นอนว่าแต่ละเทคนิคจะมีลักษณะความสามารถในการทำงานที่แตกต่างกัน ผู้ใช้จึงจำเป็นต้องเรียนรู้ถึงหลักการทำงานของแต่ละเทคนิคให้เข้าใจชัดเจนก่อนเลือกใช้งาน เพื่อจะได้เลือกใช้ได้เหมาะสมและบรรลุผลตามเป้าหมายที่ต้องการ

ในการปรับปรุงคุณภาพของภาพสามารถแบ่งออกได้เป็นเทคนิคใน 2 ประเภทใหญ่ๆ ด้วยกันตามโดเมนของข้อมูลภาพที่นำมาใช้งาน นั่นคือ

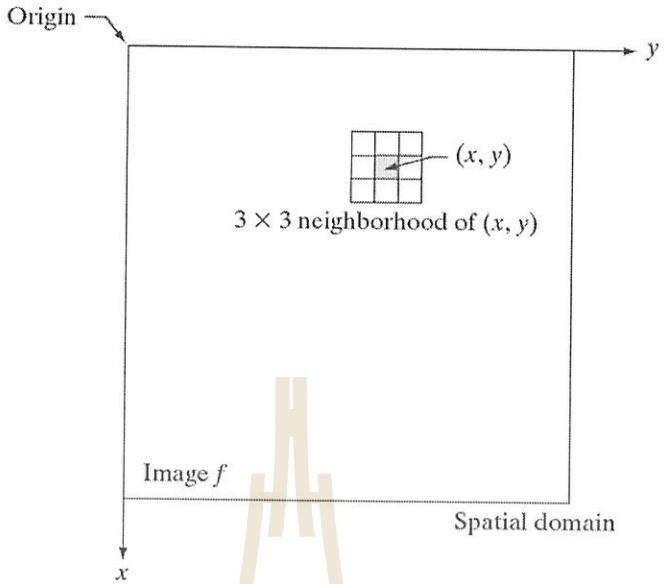
1. โดเมนระยะทาง (Spatial Domain) เป็นการประมวลผลโดยตรงกับพิกเซล (pixel) ของภาพ (เนื่องจากภาพดิจิตอลเกิดจากการชักตัวอย่าง (Sampling) ในแนวแกน x และ y ดังนั้นการประมวลผลในเชิงระยะทางจึงเกี่ยวข้องกับค่าในแนวแกนแกน x และ y นั่นคือเป็นการประมวลผลที่แต่ละพิกเซลของภาพโดยตรงนั่นเอง)
2. โดเมนความถี่ (Frequency Domain) เป็นการประมวลผลบนค่าองค์ประกอบความถี่ (Frequency Component) ซึ่งเป็นผลลัพธ์ที่ได้จากการแปลงข้อมูลภาพด้วยวิธีการแปลงแบบฟูริเยร์ (Fourier Transformation)

ในบทนี้เราจะกล่าวถึงเทคนิคต่างๆ ในโดเมนระยะทางเท่านั้น ส่วนเทคนิคบนโดเมนความถี่จะกล่าวถึงในบทต่อไป อย่างไรก็ตามผลลัพธ์ที่ได้จากเทคนิคของทั้งสองโดเมนนี้จะคล้ายคลึงกันมาก เพราะข้อมูลบนโดเมนของความถี่นั้นก็ได้มาจากการแปลงข้อมูลในโดเมนระยะทางไปเป็นข้อมูลความถี่นั่นเอง

4.1 พื้นฐานการปรับปรุงภาพในโดเมนระยะทาง
Background of Image Enhancement in Spatial Domain

จากที่เคยกล่าวไปแล้วว่ารูปภาพนั้นนิยามได้ในรูปของฟังก์ชัน $f(x, y)$ ใน 2 มิติเมื่อ x และ y เป็นของค่าระยะทางในแนวแกน x และแกน y และค่า amplitude ของ f ณ แต่ละพิกัด (x, y) เรียกว่า intensity ของภาพ ณ ตำแหน่งนั้นๆ รูปที่ 4.1 แสดงลักษณะการอ้างอิงตำแหน่งของรูปภาพ $f(x, y)$

โดยแสดงถึงลักษณะของneighborhood (พิกเซลข้างเคียง) ขนาด 3x3 ของพิกเซล ณ ตำแหน่ง *(x, y)* ใดๆ ของภาพ



รูปที่ 4.1 Neighborhood (พิกเซลข้างเคียง) ขนาด 3x3 ของพิกเซล ณ ตำแหน่ง (x, y) ใดๆ ของภาพดิจิตอล

เทคนิคต่างๆ ในการปรับปรุงคุณภาพของภาพ $f(x, y)$ นั้นแทนด้วยฟังก์ชัน T หรือเรียกอีกอย่างว่า ฟังก์ชันการแปลง (Transformation Function) ที่มีอินพุตของฟังก์ชันคือ $f(x, y)$ โดยฟังก์ชันการแปลงจะกระทำบนกลุ่มพิกเซลข้างเคียง ณ ตำแหน่ง (x, y) และผลลัพธ์ที่ได้จากการแปลงด้วยฟังก์ชัน T ดังกล่าวคือ ภาพเอาต์พุต $g(x,y)$ ซึ่งสามารถเขียนอธิบายเป็นสมการได้ดังสมการที่ 4.1

$$g(x,y) = T[f(x,y)] \qquad \text{สมการ 4.1}$$

ในการดำเนินการด้วยฟังก์ชัน T นั้น จะกำหนดพิกเซลข้างเคียงสำหรับ (x, y) ใดๆ โดยพิจารณาภายในพื้นที่สี่เหลี่ยมจัตุรัสหรือสี่เหลี่ยมผืนผ้าเล็กๆ ที่อยู่รอบๆ (x, y) ดังแสดงในรูปที่ 4.1 ผลลัพธ์ที่ได้จากการดำเนินการด้วยฟังก์ชัน T คือค่าระดับความเทา (intensity) ค่าใหม่ที่จะกำหนดให้กับพิกเซล (x, y) นั่นเอง นั่นคือ ในการปรับปรุงคุณภาพ ณ ตำแหน่งพิกเซล (x,y)ใดๆ จะใช้ค่าระดับความเทาของพิกเซลข้างเคียงของ (x, y) ในการคำนวณ

พื้นที่สี่เหลี่ยมจัตุรัสหรือสี่เหลี่ยมผืนผ้าเล็กๆ ที่กำหนดเป็นพิกเซลข้างเคียงของพิกเซล (x, y) ใดๆ นั้นอาจเรียกว่า มาส์ก (Masks) ตัวกรอง (Filters) วินโดส์ (Windows) แทมเพรต (Tamplates) หรือเคอเนลล์ (Kernels) โดยในแต่ละเทคนิคจะนำมาส์กที่มีค่าสัมประสิทธิ์ที่ต่างกัน (ทำให้ได้ผลลัพธ์ที่ต่างกันในแต่ละเทคนิค) มาคูณกับค่าระดับความเทาของพื้นที่ภาพที่มาส์กนี้ทับอยู่ ค่าระดับความเทาของพิกเซล (x, y) ที่อยู่ ณ ตำแหน่งตรงกลางมาส์กจะถูกกำหนดโดยค่าใหม่ที่ได้จากการดำเนินการด้วยฟังก์ชัน T การคำนวณค่าระดับความเทาใหม่นั้นจะทำกับทุกๆ พิกเซลของภาพซึ่งทำ

ได้โดยการกวาดมาส์กไปทั่วทั้งภาพ โดยเริ่มจากมุมบนซ้ายสุดของภาพ (Origin) แล้วขยับไปทีละพิกเซลไปทางขวามือ (ตามแนวแกน y ดังรูปที่ 4.1) เมื่อหมดหนึ่งแถวก็ขยับลงมาด้านล่างหนึ่งแถว (ตามแนวแกน x ดังรูปที่ 4.1) นั่นคือเป็นการเลื่อนมาส์กไปในทิศทางซ้ายไปขวา บนลงล่าง โดยที่พิกเซลบริเวณขอบของภาพนั้นมักจะกำหนดให้พิกเซลข้างเคียงที่อยู่นอกกรอบของภาพมีค่าระดับเทาเป็นศูนย์ (สีดำ) ทั้งหมด

ขนาดของมาส์กที่มีขนาดเล็กที่สุดคือ 1x1 ซึ่งหมายถึงการใช้ค่าระดับเทาของพิกเซล (x, y) เพียงค่าเดียว (ไม่พิจารณาพิกเซลข้างเคียง) ในการพิจารณาหาค่าระดับความเทาใหม่ให้กับตัวมันเอง เราเรียกฟังก์ชัน T ที่นำมาใช้งานในลักษณะนี้ว่า ฟังก์ชันการแปลงระดับความเทา (Gray-level or Intensity or Mapping Transformation Function) และเรียกการประมวลผลในลักษณะนี้ว่า การประมวลผลจุด (Point Processing) ซึ่งสามารถเขียนในรูปสมการได้ดังสมการที่ 4.2

$$s = T(r) \qquad \text{สมการ 4.2}$$

เมื่อ r คือ ระดับความเทาตั้งต้นของพิกเซล (x, y) และ s คือ ระดับความเทาใหม่ของพิกเซล (x, y)

สำหรับมาส์กที่มีขนาดใหญ่กว่า 1x1 (มักอยู่ในรูปแบบ 2n+1 x 2n+1) เช่น 3x3, 5x5, ... ซึ่งในการประมวลผลที่ใช้มาส์กขนาดใหญ่กว่า 1x1 นั้นเรียกว่าการประมวลผลมาส์กหรือการกรอง (Mask Processing or Filtering)

4.2 ฟังก์ชันพื้นฐานในการแปลงระดับความเทา
Basic Gray Level Transformation Functions

สำหรับฟังก์ชันพื้นฐานในการแปลงค่าระดับความเทาที่ใช้ในการแปลงค่าระดับความเทาของภาพ เพื่อเป็นการปรับปรุงคุณภาพของภาพให้ดีขึ้นได้นั้น สามารถแบ่งออกได้เป็น

1. ฟังก์ชันการแปลงแบบเชิงเส้น (Linear Transformation Functions)
2. ฟังก์ชันการแปลงแบบลอการึทึม (Log Transformation Functions)
3. ฟังก์ชันการแปลงแบบยกกำลัง (Power-law Transformation Functions)
4. ฟังก์ชันการแปลงแบบเชิงเส้นต่อเนื่อง (Piecewise -linear Transformation Functions)

- **ฟังก์ชันการแปลงแบบเชิงเส้น (Linear Transformation Functions)**

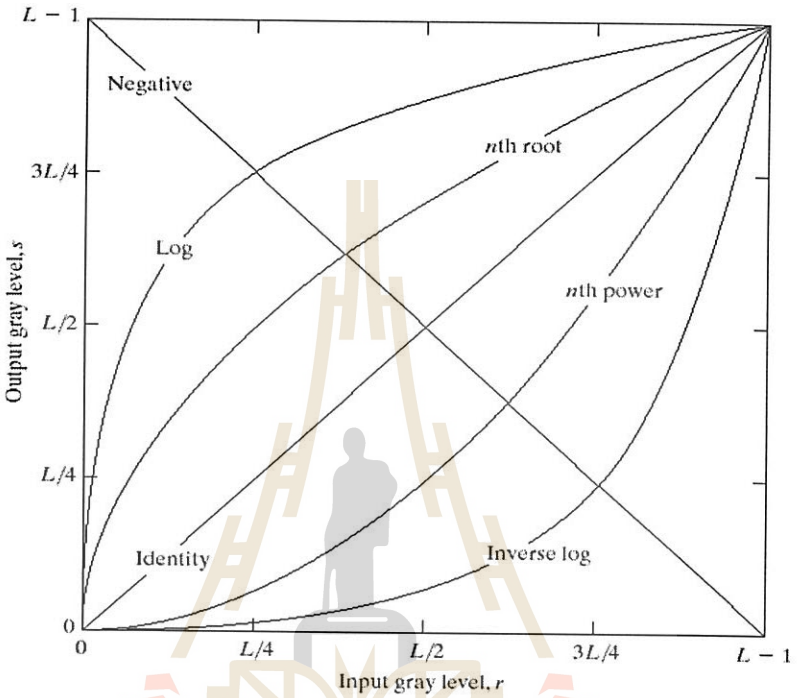
  ประกอบด้วย

  - ฟังก์ชันการแปลงเอกลักษณ์ (Identity Transformation Function) ผลลัพธ์จากการแปลงที่ได้คือภาพต้นฉบับนั่นเอง

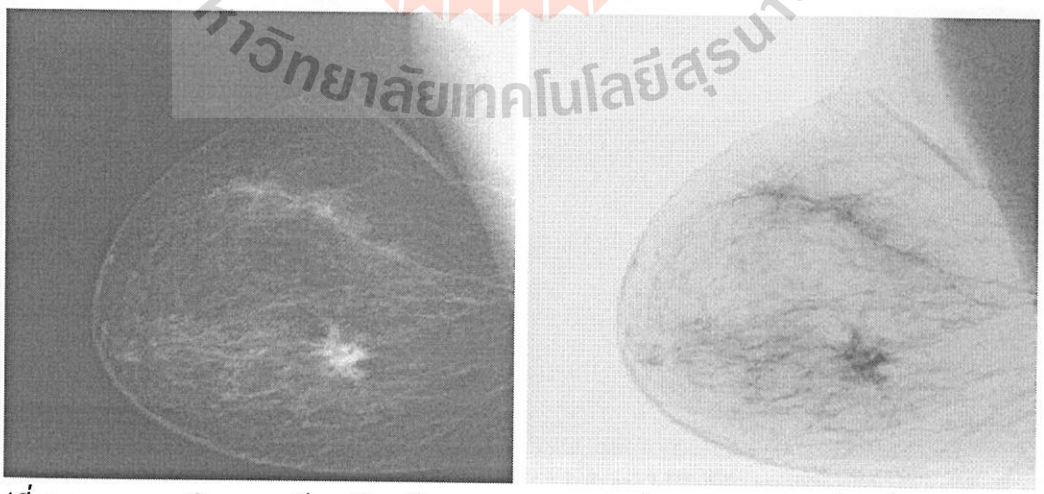
- ฟังก์ชันการแปลงค่ากลับ (Negative Transformation Function) เป็นการกลับค่าระดับเทา เช่นถ้าช่วงระดับความเทาของภาพคือ [0, L-1] แล้วฟังก์ชันการแปลงค่ากลับสามารถเขียนได้ดังสมการ

$$s = L - 1 - r$$

การใช้ฟังก์ชันนี้เหมาะสำหรับการทำให้บริเวณที่มีสีขาวหรือเทาบนพื้นภาพสีดำมีความเด่นชัดมาก ขึ้นโดยเฉพาะภาพที่องค์ประกอบสีดำมีจำนวนมาก



รูปที่ 4.2 Input และ Output ของค่าระดับเทาจากการใช้ฟังก์ชันการแปลงค่าระดับเทาแบบเชิงเส้น ลอการิทึมและยกกำลัง



รูปที่ 4.3 ภาพแมมโมแกรมต้นฉบับ (ซ้าย) และภาพผลลัพธ์(ขวา) หลังการใช้ฟังก์ชันการแปลงค่ากลับ

➢ **ฟังก์ชันการแปลงแบบลอการึทึม (Log Transformation Functions)**

ประกอบด้วย

- ฟังก์ชันการแปลงแบบลอการึทึม (Log Transformation Functions)

  เป็นการเพิ่มความสว่างให้แก่แต่ละค่าระดับความเทา r ที่มีค่าต่ำๆ (สังเกตได้ว่าความชันของฟังก์ชันช่วง r ต่ำๆ จะมีค่าสูง) ซึ่งจะทำให้ pixel ที่มีระดับความเทาต่ำหรือมืดกลับสว่างมากขึ้น นั่นคือ บริเวณค่า r ต่ำๆ จะถูกยืดค่าระดับความเทาออก (spreading) ในขณะที่ pixel ที่เป็นสีเทาสว่าง หรือมีค่า r สูงๆ จะถูกบีบให้มีค่าลดลง กล่าวคือ pixel ที่เคยสว่างมากจะถูกทำให้สว่างน้อยลง ทำให้บริเวณที่มีค่า r สูงๆ นั้นถูกบีบอัดค่าระดับเทาลง (Compressing) แต่ระดับความเทาค่าสูงสุด (สีขาว) และต่ำสุด (สีดำ) ยังคงมีค่าเท่าเดิม ไม่เปลี่ยนแปลง โดยที่สมการของฟังก์ชันการแปลงแบบลอการึทึมคือ

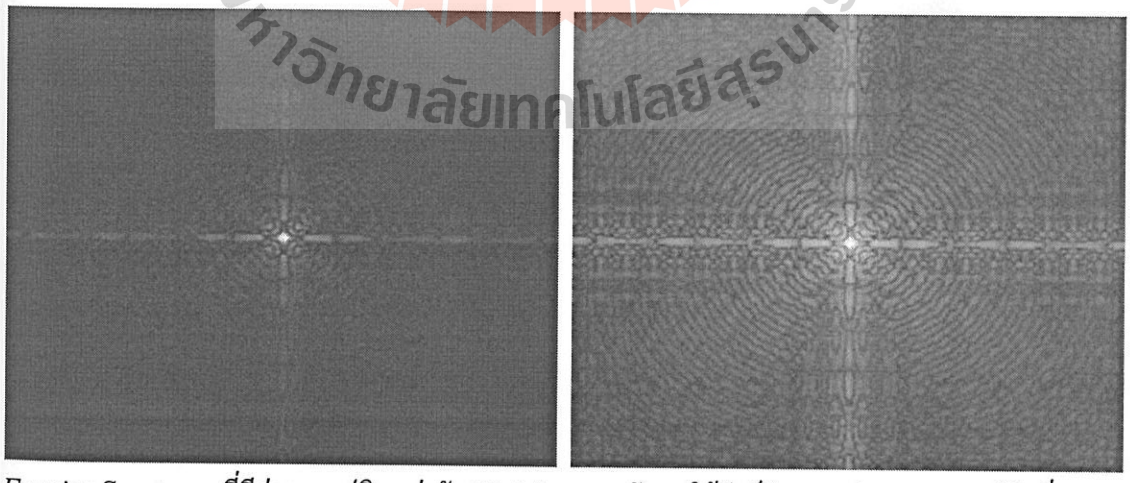
  $$s = c\log(1+r)$$

  เมื่อ c คือค่าคงที่และ $r >= 0$

  ภาพที่เหมาะจะใช้ฟังก์ชันนี้ในการเพิ่มคุณภาพการมองเห็นได้แก่ ภาพที่มีช่วงระดับความเทาที่กว้างมากๆ (ค่าระดับความสว่างสูงสุดมีค่ามากๆ) เช่น Fourier Spectrum

- ฟังก์ชันการแปลงแบบอินเวอร์สลอการึทึม (Inverse-log Transformation Functions)

  จะทำงานตรงกันข้ามกับการทำงานของฟังก์ชันการแปลงแบบลอการึทึม กล่าวคือ ฟังก์ชันการแปลงแบบอินเวอร์สลอการึทึมจะทำให้ pixel ที่มีระดับความเทาต่ำมีค่ายิ่งต่ำลงไปอีก ในขณะที่ pixel ที่สว่างถูกทำให้สว่างมากขึ้น



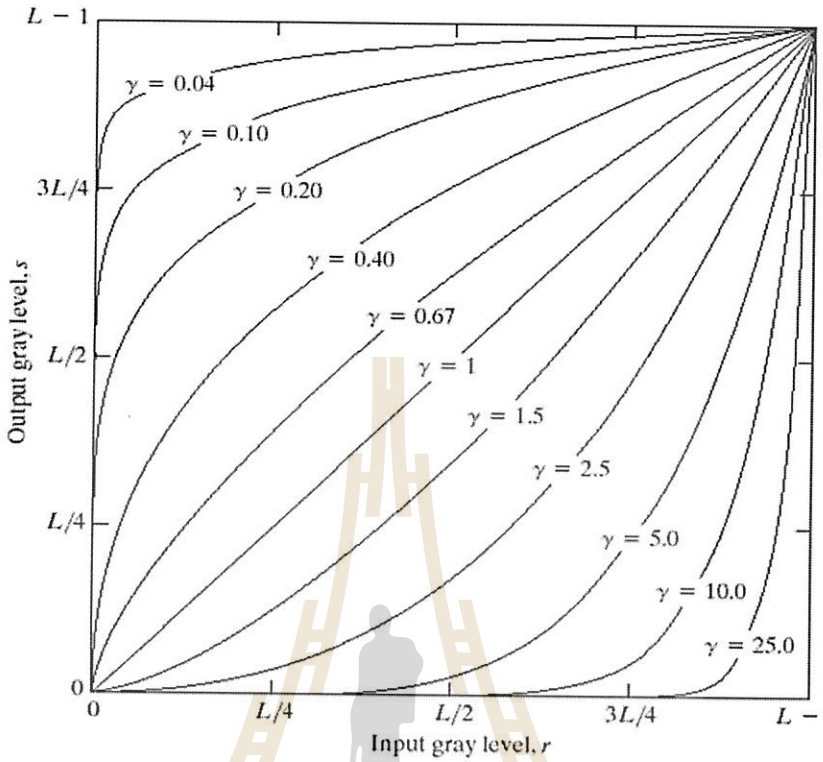
Fourier Spectrum ที่มีช่วงแอมปลิจูดเท่ากับ [0, 1.5 x 10^6]

หลังการใช้ฟังก์ชันการแปลงแบบลอการึทึมที่ c=1

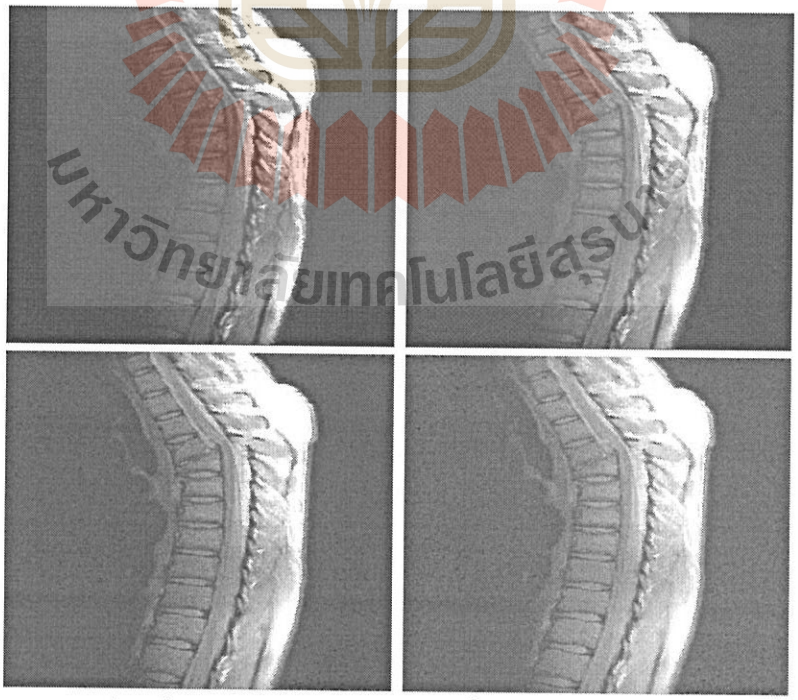
รูปที่ 4.4 ภาพ Fourier Spectrumหลังการใช้ฟังก์ชันการแปลงแบบลอการึทึมที่ c=1

- **ฟังก์ชันการแปลงแบบยกกำลัง (Power-law Transformation Functions)**

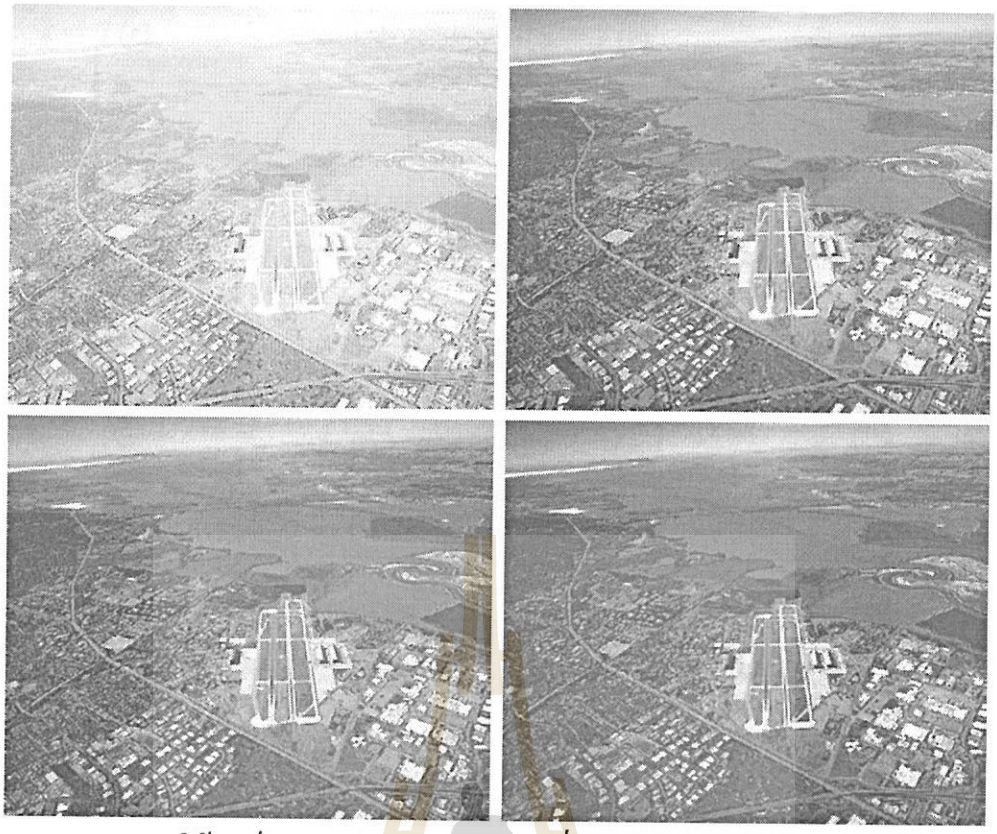
ฟังก์ชันการแปลงแบบยกกำลังนั้นอยู่ในรูปของสมการ $s = cr^{\gamma}$



รูปที่ 4.5 ฟังก์ชันการแปลงแบบยกกำลังที่มีค่า γ ต่างๆ กันและค่า c=1



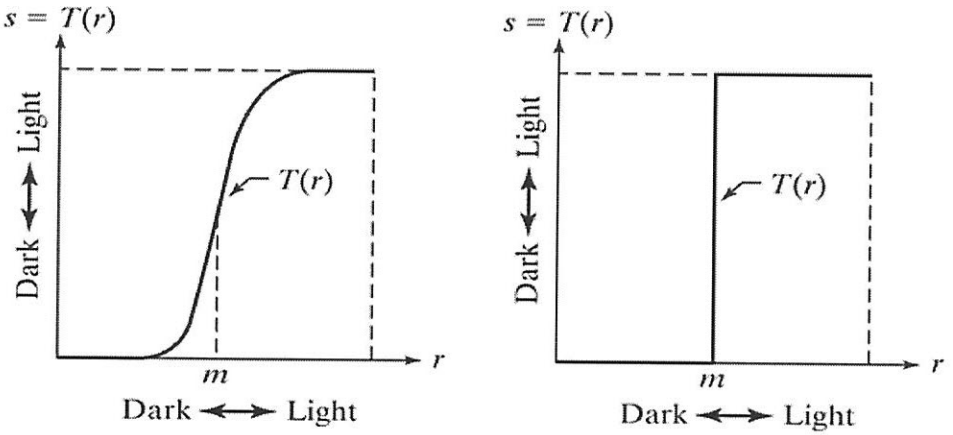
รูปที่ 4.6 ภาพจากการใช้ฟังก์ชันการแปลงแบบยกกำลังที่มีค่า r= 0.6, 0.4 และ 0.3 ตามลำดับ (ภาพบนซ้ายคือภาพต้นฉบับ) เมื่อค่า c=1



รูปที่ 4.7 ภาพจากการใช้ฟังก์ชันการแปลงแบบยกกำลังที่มีค่า r= 3, 4 และ 5 ตามลำดับ (ภาพบนซ้ายคือภาพต้นฉบับ) เมื่อค่า c=1

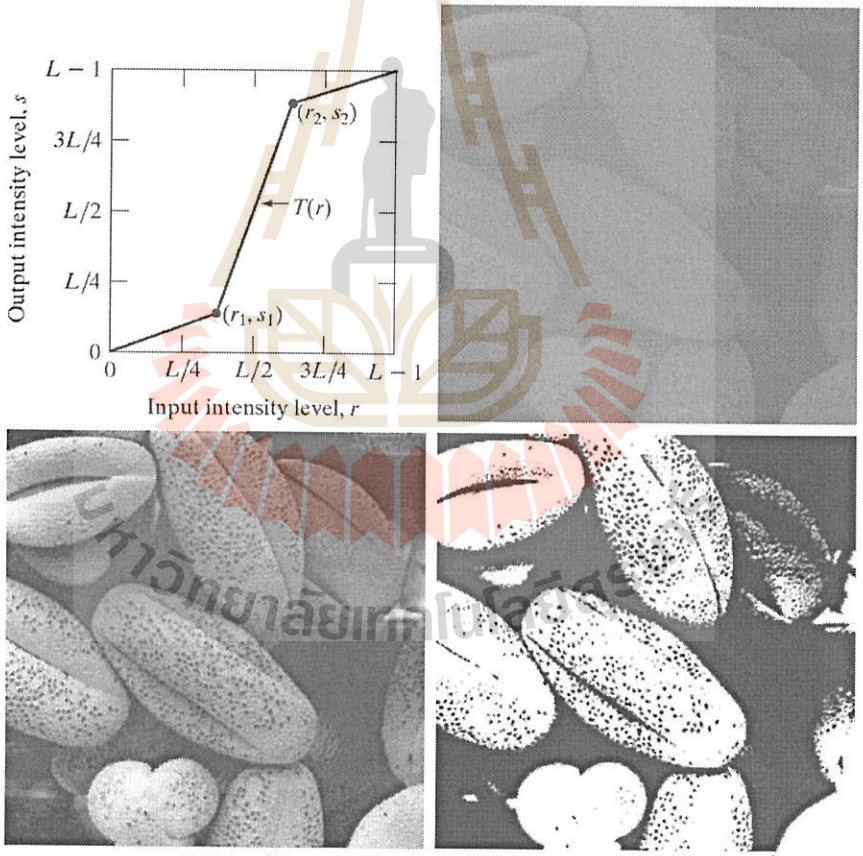
➢ **ฟังก์ชันการแปลงแบบเชิงเส้นต่อเนื่อง (Piecewise -linear Transformation Functions)**

- ฟังก์ชันการแปลงแบบยืดความคมชัดเชิงเส้นต่อเนื่อง (Piecewise-Linear Transformation Function) เป็นการแปลงแบบยืดความแตกต่างของระดับความเทาในลักษณะรูปที่ 4.8 ซ้าย ซึ่งค่าระดับเทา r ที่มีค่าต่ำกว่า m จะถูกบีบค่าต่ำลงหรือมืดลง ส่วนค่า r ที่มากกว่า m จะถูกขยายให้มีค่าสูงขึ้นหรือสว่างขึ้น
- ฟังก์ชันการแปลงแบบดีสครีต (Discrete Transformation Function) เป็นการแปลงแบบยืดระดับเทาที่ผลลัพธ์มีอยู่เพียง 2 ค่าคือ 0 และ 1 จึงทำให้ได้ภาพเป็นภาพขาวดำ จึงเรียกฟังก์ชันนี้ว่า ฟังก์ชันค่าแบ่ง (Thresholding Function)



(ซ้าย) ฟังก์ชันต่อเนื่องการแปลงแบบยืดระดับเทา *(ขวา) ฟังก์ชันดีสครีตการแปลงแบบยืดระดับเทา*

รูปที่ 4.8 ฟังก์ชันการแปลงแบบเชิงเส้นต่อเนื่อง



รูปที่ 4.9 ภาพจากการใช้ฟังก์ชันการแปลงแบบยืดความคมชัดเชิงเส้นต่อเนื่อง (ล่างซ้าย) และแบบฟังก์ชันค่าแบ่ง(ล่างขวา) เมื่อภาพบนขวาคือภาพต้นฉบับ

4.3 การประมวลผลอีสโตแกรม
Histogram Processing

➢ Histogram

Histogram หมายถึง กราฟที่แสดงจำนวน pixel ของสี หรือของ intensity ค่าต่างๆ

$$h(r_k) = n_k$$

เมื่อ r_k คือค่าระดับเทาที่ k และ

n_k คือจำนวนพิกเซลในภาพที่มีค่าระดับความเทาเท่ากับ r_k

- Function imhist();

Syntax:
h = imhist(f);
imhist(f)

h= imhist(f, b);
imhist(f, b)

Example 1: Simple plot image histogram

```
>> I = imread('breast.tif');
>> imshow(I)
>> imhist(I)

>> figure, imhist(I, 4)
>> J = imhist(I, 4)

>> figure, imhist(I, 10)
>> K = imhist(I, 10)
```

Example 2: Various ways (imhist, bar, stem, and plot) to plot an image histogram

```
>> I = imread('breast.tif');
>> imshow(I)
>> imhist(I)    % Plot by the default imhist function

>> h = imhist(I);
>> h1 = h(1:10:256);
>> horz = 1:10:256;

>> figure, bar(horz, h1)          % Plot by function bar
>> axis([0 255 0 5000])
>> set(gca, 'xtick', 0:50:255)          % gca: graphics current axis
>> set(gca, 'ytick', 0:2500:5000)

>> figure, stem(horz, h1, 'fill')          % Plot by function stem
```

```
>> axis([0 255 0 5000])
>> set(gca, 'xtick', 0:50:255)
>> set(gca, 'ytick', 0:2500:5000)

>> figure, stem(horz, h1, 'r--s', 'fill')   % Another plot by function stem
>> axis([0 255 0 5000])
>> set(gca, 'xtick', 0:50:255)
>> set(gca, 'ytick', 0:2500:5000)

>> h = imhist(I);
>> figure, plot(h)        % use the default value of plot function
>> axis([0 255 0 5000])
>> set(gca, 'xtick', 0:50:255)
>> set(gca, 'ytick', 0:2500:5000)

>> figure, plot(horz, h1, 'color', 'm', 'linestyle', 'none', 'marker', 'd')
%Another plot function
>> axis([0 255 0 5000])
>> set(gca, 'xtick', 0:50:255)
>> set(gca, 'ytick', 0:2500:5000)
```

Symbol	Color	Symbol	Line Style	Symbol	Marker
k	Black	–	Solid	+	Plus sign
w	White	--	Dashed	o	Circle
r	Red	:	Dotted	*	Asterisk
g	Green	-.	Dash-dot	.	Point
b	Blue	none	No line	x	Cross
c	Cyan			s	Square
y	Yellow			d	Diamond
m	Magenta			none	No marker

Attributes for function stem and plot.

- **ความคมชัด (Contrast)**

Contrast หมายถึงความคมชัดของภาพ เป็นความแตกต่างระหว่าง สีที่มืดที่สุดในภาพกับสีที่สว่างที่สุดในภาพ ซึ่งรูปแบบที่แตกต่างกันของ contrast (Different types of contrast) เช่น

- ภาพที่มืด (dark image) จะมี histogram กองอยู่ไปทางซ้าย
- ภาพที่สว่าง (bright image) จะมี histogram กองอยู่ไปทางขวา
- ภาพที่มี contrast ต่ำ (low contrast) จะมี histogram กระจุกกันอยู่ในช่วงแคบๆ
- ภาพที่มี contrast สูง (high contrast) จะมี histogram กระจายกันอยู่ในช่วงกว้างๆ

Example: Image with different types of contrast

```
>> I = imread('pollen1.jpg');
```

```
>> imhist(I)
>> J = imread('pollen2.jpg');
>> figure, imhist(J)
>> K = imread('pollen3.jpg');
>> figure, imhist(K)
>> L = imread('pollen4.jpg');
>> figure, imhist(L)

>> figure
>> subplot(4,2,1); imshow(I,[]), title('pollen1');
>> subplot(4,2,2); imhist(I), text(125, 4000, 'dark image');
>> subplot(4,2,3); imshow(J,[]), title('pollen2');
>> subplot(4,2,4); imhist(J),text(40, 4000, 'bright image');
>> subplot(4,2,5); imshow(K,[]), title('pollen3');
>> subplot(4,2,6); imhist(K), text(150, 4000, 'low contrast');
>> subplot(4,2,7); imshow(L,[]), title('pollen4');
>> subplot(4,2,8); imhist(L), text(100, 3000, 'high contrast');
```

➢ **Histogram Processing**

หมายถึงกระบวนการปรับปรุง intensity ของรูปภาพเพื่อให้ได้ histogram ที่มีลักษณะตามต้องการ เช่น

- Histogram equalization เป็นการทำให้ histogram กระจายกันอย่างสม่ำเสมอตลอด
- Histogram matching เป็นการทำให้ histogram มีลักษณะเหมือนกราฟที่กำหนดไว้

▪ **Histogram Equalization**

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j)$$

$$= \sum_{j=0}^{k} \frac{n_j}{N}$$

n_j = the number of pixels with intensity = j
N = the number of total pixels

ตัวอย่างการคำนวณ Histogram Equalization

Intensity	# pixels
0	20
1	5
2	25
3	10
4	15
5	5
6	10
7	10
Total	100

Accumulative Sum of P_r
20/100 = 0.2
(20+5)/100 = 0.25
(20+5+25)/100 = 0.5
(20+5+25+10)/100 = 0.6
(20+5+25+10+15)/100 = 0.75
(20+5+25+10+15+5)/100 = 0.8
(20+5+25+10+15+5+10)/100 = 0.9
(20+5+25+10+15+5+10+10)/100 = 1.0
1.0

Intensity (r)	No. of Pixels (n_j)	Acc Sum of P_r	Output value	Quantized Output (s)
0	20	0.2	0.2x7 = 1.4	1
1	5	0.25	0.25*7 = 1.75	2
2	25	0.5	0.5*7 = 3.5	3
3	10	0.6	0.6*7 = 4.2	4
4	15	0.75	0.75*7 = 5.25	5
5	5	0.8	0.8*7 = 5.6	6
6	10	0.9	0.9*7 = 6.3	6
7	10	1.0	1.0x7 = 7	7
Total	100			

Histogram Equalization ใน MATLAB

- Function histeq();

Syntax:

```
g = histeq(f, nlev);
```

f is the input image

nlev is the number of intensity levels specified for the output image (default value is 64 but should use 256).

Note: การใช้ ฟังก์ชัน histeq ของ Matlab นั้นถ้าไม่ใส่ parameter ที่กำหนด level จะ default ค่าให้เท่ากับ 64 ดังตัวอย่าง

```
>> histeq(I);
```

แต่เราควรใส่ parameter ที่กำหนด level เองให้เป็น 256 ซึ่งจะเป็นไปตามทฤษฎีของการทำ histogram equalization ดังตัวอย่าง

```
>> histeq(I, 256);
```

อย่างไรก็ตามผลลัพธ์ที่ได้จากการกำหนด level ทั้งสองแบบก็ไม่ได้แตกต่างกันมากนัก

Example1:

```
>> I = imread('tire.tif');
>> J = histeq(I); %default to 64 level
>> imshow(I), figure, imshow(J)
>> figure, imhist(J), ylim('auto')

>> K = histeq(I,256); %default to 64 level but should use 256
>> figure, imshow(K)
>> figure, imhist(K), ylim('auto')
```

Example2:

```
>> f = imread('pollen1.jpg');
>> imshow(f)
>> figure, imhist(f), ylim('auto')
>> g = histeq(f, 256);
>> figure, imshow(g);
>> figure, imhist(g), ylim('auto');
```

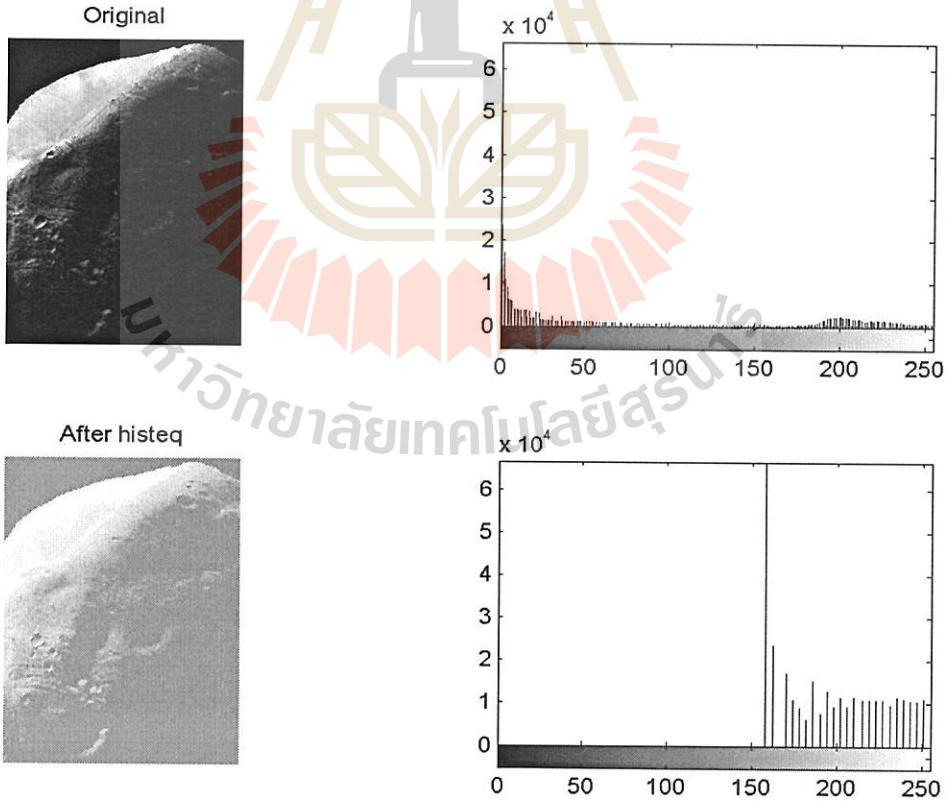
Example3:

```
>> I = imread('pollen1.jpg');
>> J = imread('pollen2.jpg');
>> K = imread('pollen3.jpg');
>> L = imread('pollen4.jpg');

>> figure
>> subplot(4,2,1); imshow(I,[]), title('pollen1');
>> subplot(4,2,2); imhist(I), text(125, 4000, 'dark image');
>> subplot(4,2,3); imshow(J,[]), title('pollen2');
>> subplot(4,2,4); imhist(J),text(40, 4000, 'bright image');
>> subplot(4,2,5); imshow(K,[]), title('pollen3');
>> subplot(4,2,6); imhist(K), text(150, 4000, 'low contrast');
>> subplot(4,2,7); imshow(L,[]), title('pollen4');
>> subplot(4,2,8); imhist(L), text(100, 3000, 'high contrast');
```

```
>> figure
>> subplot(4,2,1); histeq(I,256); II=histeq(I); title('histeq pollen1');
>> subplot(4,2,2); imhist(II);
>> subplot(4,2,3); histeq(J,256); JJ=histeq(J); title('histeq pollen2');
>> subplot(4,2,4); imhist(JJ);
>> subplot(4,2,5); histeq(K,256); KK=histeq(K); title('histeq pollen3');
>> subplot(4,2,6); imhist(KK);
>> subplot(4,2,7); histeq(L,256); LL=histeq(L); title('histeq pollen4');
>> subplot(4,2,8); imhist(LL);
```

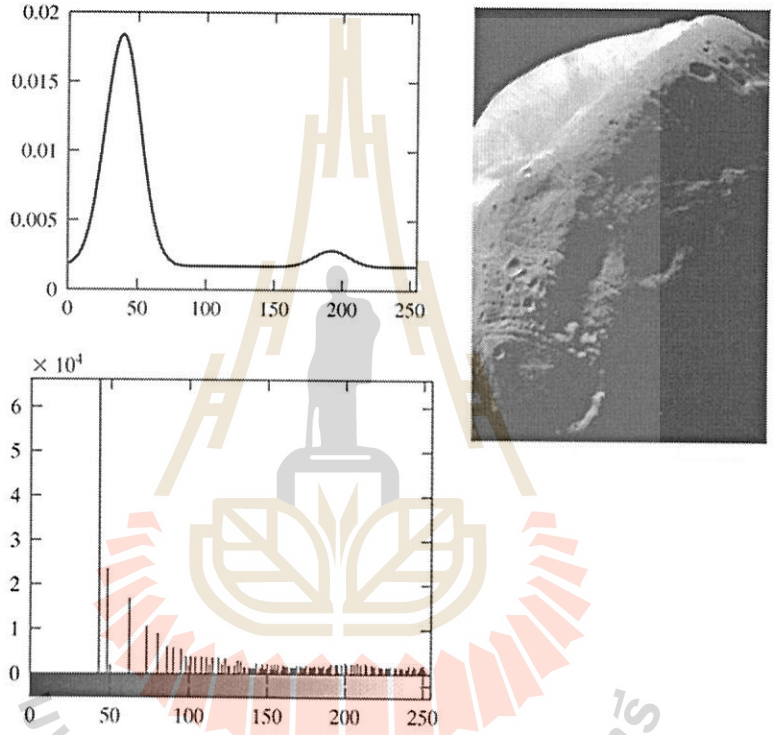
- **Histogram Matching**

ข้อจำกัดของ histogram equalization คือ เป็นการกำหนดฟังก์ชันความหนาแน่นความน่าจะเป็นของระดับความเทาของภาพผลลัพธ์เป็นลักษณะรูปแบบที่ตายตัว กล่าวคือ เป็นฟังก์ชันความหนาแน่นความน่าจะเป็นของระดับความเท่าแบบสม่ำเสมอเท่านั้น หมายความว่า ถึงแม้ค่า histogram ของภาพ input มีการเปลี่ยนแปลงไป ฟังก์ชันที่ใช้ในการแปลงค่าก็ไม่ได้มีการเปลี่ยนแปลงตามไปด้วย ทำให้ในบางครั้ง histogram equalization ก็ได้ผลลัพธ์ที่ไม่ดีในการปรับปรุงคุณภาพกับภาพบางแบบ ดังตัวอย่าง ภาพ Moon Phobos



การใช้ histogram equalization ที่ทำให้ภาพผลลัพธ์ที่ได้เป็นภาพแบบ low contrast ไป ซึ่งไม่ได้ทำให้การปรับปรุงคุณภาพของภาพดีขึ้น

ดังนั้นสำหรับภาพบางภาพหรือใน Applications บางชนิด เราอาจต้องการให้ภาพผลลัพธ์มีลักษณะเฉพาะของฟังก์ชันความหนาแน่นความน่าจะเป็นของระดับความเทาเป็นไปตามที่เราต้องการ ซึ่ง histogram equalization ไม่สามารถทำได้ แต่สามารถทำได้ด้วยวิธีที่เรียกว่า histogram matching หรืออาจเรียกว่า วิธีการกำหนดลักษณะเฉพาะของ histogram (histogram specification) ดังตัวอย่าง เมื่อภาพบนซ้ายคือ model ของลักษณะ histogram ของภาพผลลัพธ์ที่ผู้ใช้ต้องการ และภาพล่างซ้ายคือ histogram ที่ได้จากการทำ histogram matching ซึ่งได้ลักษณะของ histogram ที่ใกล้เคียงกับ model ที่ต้องการ ส่วนภาพทางขวาเป็นภาพผลลัพธ์ที่ได้จากการทำ histogram matching ซึ่งคุณภาพของภาพที่ได้ดีขึ้นมาก



Histogram Matching of Moon Phobos Image

Note: ใน Matlab นั้นก็สามารถใช้ฟังก์ชัน histeq ในการทำ histogram matching ได้แต่ต้องมี input ที่เป็น vector ของ model ของ histogram ที่ต้องการมาก่อน ดังตัวอย่าง

```
>> histeq(I, P);
```

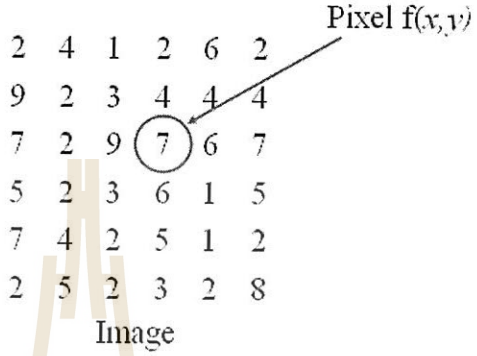
เมื่อ P เป็น vector (row or column matrix) ซึ่งเป็น model ของ histogram ที่ต้องการ

4.4 Spatial Filtering

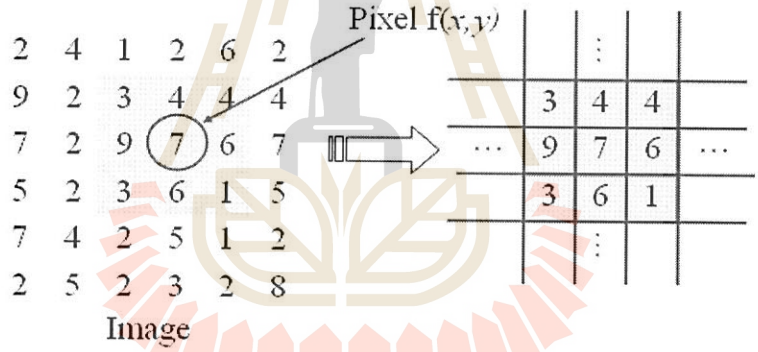
- **Fundamental of Spatial Filter**
  - **Basic Mask Processing**

Sometime we need to manipulate values obtained from neighboring pixels

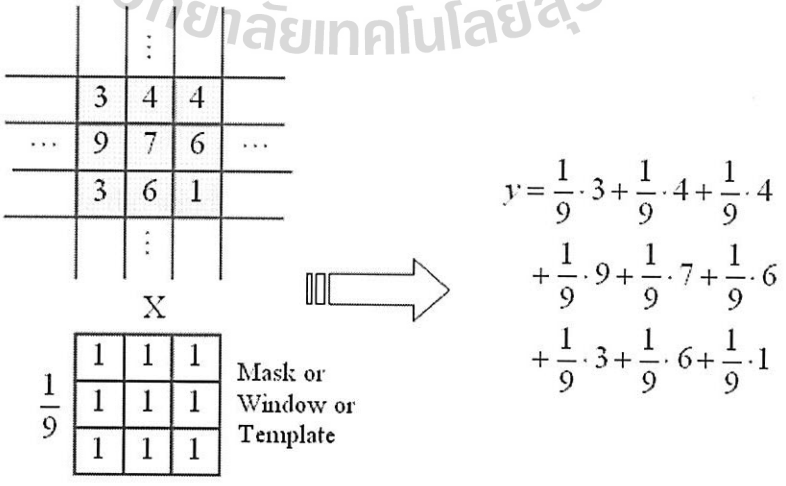
Example: How can we compute an average value of pixels in a 3x3 region center at a pixel *f*(x,y)?



Step 1. Selected only needed pixels

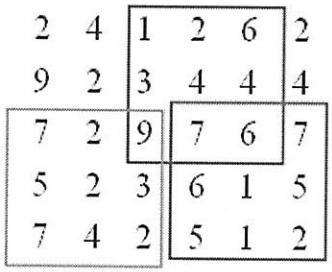


Step 2. Multiply every pixel by 1/9 and then sum up the values

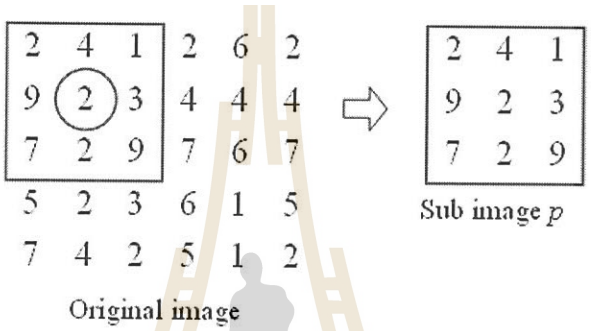


$$y = \frac{1}{9}\cdot 3 + \frac{1}{9}\cdot 4 + \frac{1}{9}\cdot 4 + \frac{1}{9}\cdot 9 + \frac{1}{9}\cdot 7 + \frac{1}{9}\cdot 6 + \frac{1}{9}\cdot 3 + \frac{1}{9}\cdot 6 + \frac{1}{9}\cdot 1$$

- **Mask processing to every pixels**



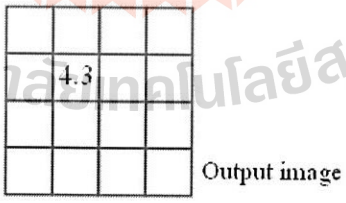
Step 1: Move the window to the first location where we want to compute the average value and then select only pixels inside the window.



Step 2: Compute the average value

$$y = \sum_{i=1}^{3}\sum_{j=1}^{3} \frac{1}{9} \cdot p(i, j)$$

Step 3: Place the result at the pixel in the output image



Step 4: Move the window to the next location and go to Step 2

Note: Wording *mask* was named by different words but the same meaning such as:

window, template, kernel, filter, filter mask or convolution filter.

Mask can be any size of m x n that $m = 2a+1$ and $n = 2b+1$ where a and b are nonnegative integers

➢ Spatial Filter in Matlab

Example 1: Simple computes mean value directly from the generated data

```
>> x = uint8(10*magic(5))

x =

  170  240   10   80  150
  230   50   70  140  160
   40   60  130  200  220
  100  120  190  210   30
  110  180  250   20   90

>> mean2(x(1:3,1:3))

ans =

  111.1111

>> mean2(x(1:3,2:4))

ans =

  108.8889
```

Example 2: Use *imfilter* function of IPT (Image Processing Toolbox)
Syntax:
g = imfilter(f, w, filtering_mode, boundary_options, size_options)

Note: See the detail syntax of imfilter function by using help (>> help imfilter)

```
>> f = uint8(10*magic(5))

f =

  170  240   10   80  150
  230   50   70  140  160
   40   60  130  200  220
  100  120  190  210   30
  110  180  250   20   90

>> w = ones(3,3)/9

w =

   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111
```

```
>> imfilter(f, w)

ans =

    77    86    66    68    59
    88   111   109   129   106
    67   110   130   150   107
    68   131   151   149    86
    57   106   108    88    39

>> f = uint8(zeros(5))

f =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0

>> f(3,3) = 1

f =

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

>> w = [1 2 3;4 5 6; 7 8 9]

w =

     1     2     3
     4     5     6
     7     8     9

>> imfilter(f,w)

ans =

     0     0     0     0     0
     0     9     8     7     0
     0     6     5     4     0
     0     3     2     1     0
     0     0     0     0     0


>> imfilter(f,w,'corr')
ans =

     0     0     0     0     0
     0     9     8     7     0
     0     6     5     4     0
```

```
    0    3    2    1    0
    0    0    0    0    0
```

```
>> imfilter(f,w,'conv')

ans =

    0    0    0    0    0
    0    1    2    3    0
    0    4    5    6    0
    0    7    8    9    0
    0    0    0    0    0
```

```
>> imfilter(f,w,'corr', 'same')

ans =

    0    0    0    0    0
    0    9    8    7    0
    0    6    5    4    0
    0    3    2    1    0
    0    0    0    0    0
```

```
>> imfilter(f,w,'corr', 'full')

ans =

    0    0    0    0    0    0    0
    0    0    0    0    0    0    0
    0    0    9    8    7    0    0
    0    0    6    5    4    0    0
    0    0    3    2    1    0    0
    0    0    0    0    0    0    0
    0    0    0    0    0    0    0
```

Example: Test for boundary options

```
>> imfilter(f,w)

ans =

    0    0    0    0    0
    0    9    8    7    0
    0    6    5    4    0
    0    3    2    1    0
    0    0    0    0    0
```

```
>> imfilter(f,w,0)

ans =

    0    0    0    0    0
    0    9    8    7    0
    0    6    5    4    0
```

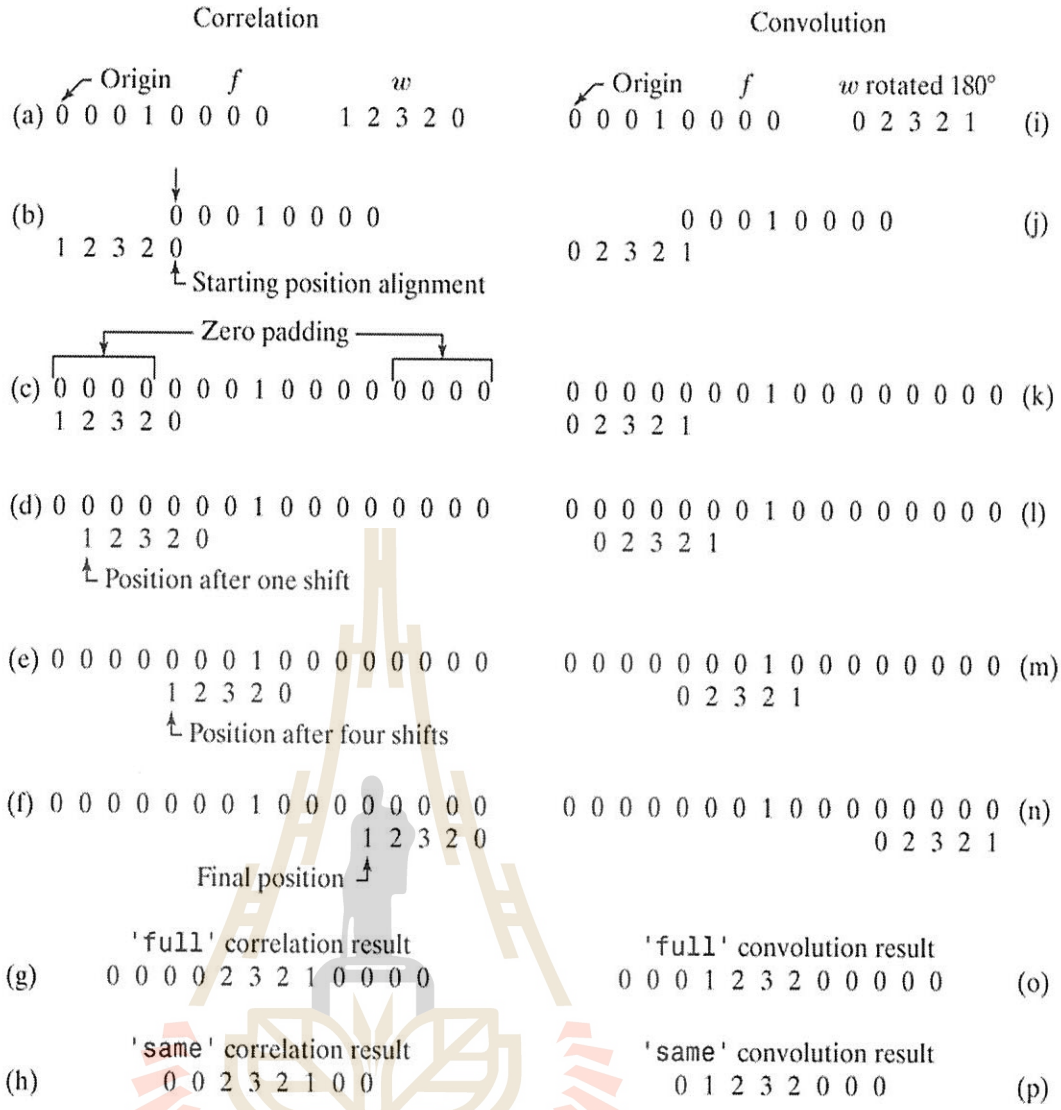
```
     0     3     2     1     0
     0     0     0     0     0
```

```
>> imfilter(f,w,1)

ans =

    17     6     6     6    21
    12     9     8     7    18
    12     6     5     4    18
    12     3     2     1    18
    29    24    24    24    33
```

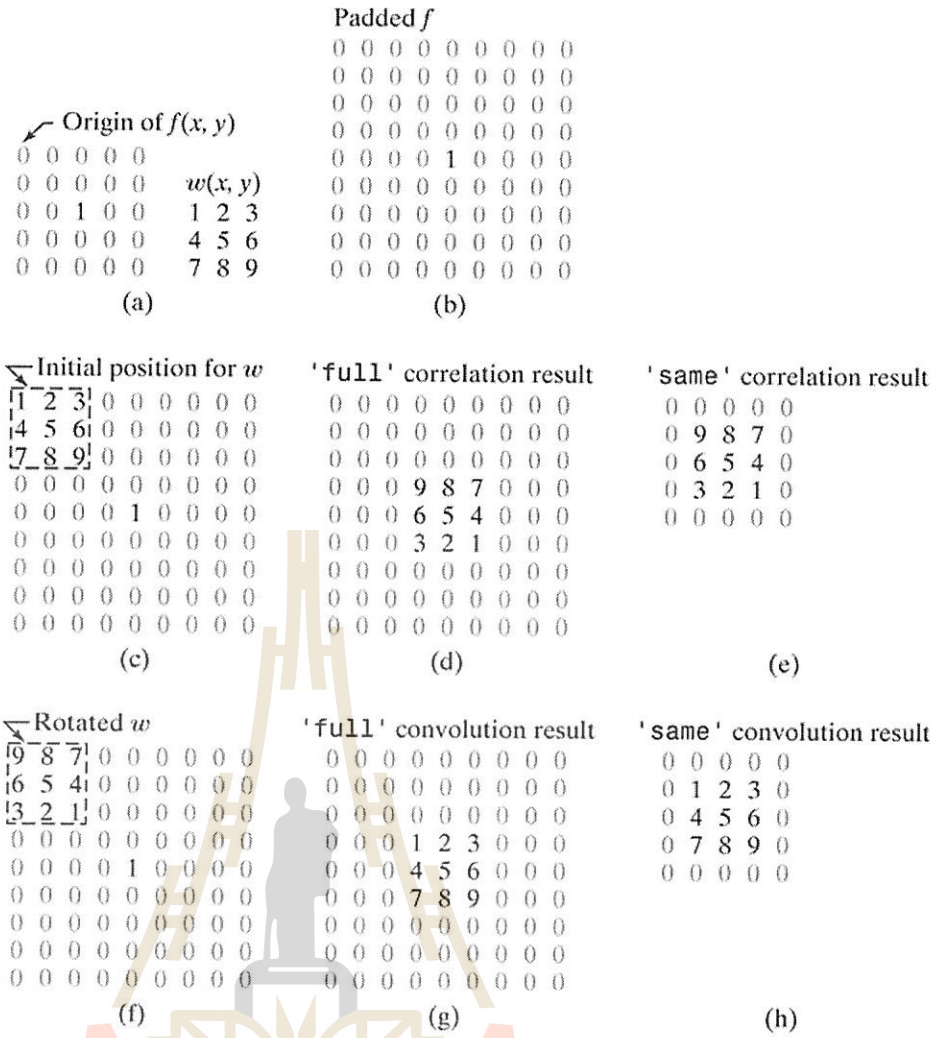
```
>> imfilter(f,w,255)

ans =

   255   255   255   255   255
   255     9     8     7   255
   255     6     5     4   255
   255     3     2     1   255
   255   255   255   255   255
```

Options	Description
Filtering Mode	
`'corr'`	Filtering is done using correlation (see Figs. 3.13 and 3.14). This is the default.
`'conv'`	Filtering is done using convolution (see Figs. 3.13 and 3.14).
Boundary Options	
`P`	The boundaries of the input image are extended by padding with a value, `P` (written without quotes). This is the default, with value 0.
`'replicate'`	The size of the image is extended by replicating the values in its outer border.
`'symmetric'`	The size of the image is extended by mirror-reflecting it across its border.
`'circular'`	The size of the image is extended by treating the image as one period a 2-D periodic function.
Size Options	
`'full'`	The output is of the same size as the extended (padded) image (see Figs. 3.13 and 3.14).
`'same'`	The output is of the same size as the input. This is achieved by limiting the excursions of the center of the filter mask to points contained in the original image (see Figs. 3.13 and 3.14). This is the default.

Options for function imfilter



One-dimension correlation and convolution

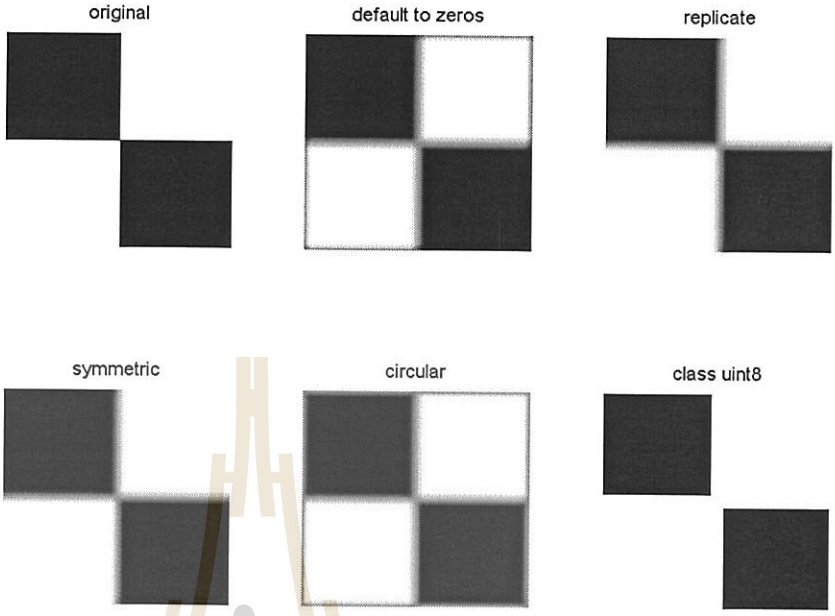


Two-dimension correlation and convolution

Example 3: Test boundary options of function *imfilter*

```
>> f = imread('block.tif');  %image of 'block.tif' can load from moodle
>> f = im2double(f);
>> subplot(2,3,1); imshow(f,[]), title('original');

>> w = ones(31);

>> gd = imfilter(f, w);
>> subplot(2,3,2); imshow(gd,[]), title('default to zeros');

>> gr = imfilter(f, w, 'replicate');
>> subplot(2,3,3); imshow(gr,[]), title('replicate');

>> gs = imfilter(f, w, 'symmetric');
>> subplot(2,3,4); imshow(gs,[]), title('symmetric');

>> gc = imfilter(f, w, 'circular');
>> subplot(2,3,5); imshow(gc,[]), title('circular');
```

```
>> f8 = im2uint8(f);
>> g8r = imfilter(f8, w, 'replicate');
>> subplot(2,3,6); imshow(g8r,[]), title('class uint8');
```



Result of Example 3

- **Linear Spatial Filter**
  - **Function *fspecial()*** ใช้ในการ generate filter mask ที่เป็นแบบ linear filter

    Syntax:

    w = fspecial('type', parameters)

    ซึ่ง type และ parameters สามารถกำหนดรูปแบบต่างๆ ได้จาก ตารางด้านล่าง โดยที่ในที่นี่จะกล่าวถึงเฉพาะ mask แบบต่อไปนี้คือ

    1. average ใช้ในการทำให้ภาพเลือนขึ้น (blurring image)
    2. gaussian ใช้ในการทำให้ภาพเลือนขึ้น (blurring image)
    3. laplacian ใช้ในการทำให้ภาพคมขึ้น (sharpening image)
    4. log ใช้ในการทำให้ภาพคมขึ้น (sharpening image)

Type	Syntax and Parameters
'average'	fspecial('average', [r c]). A rectangular averaging filter of size r × c. The default is 3 × 3. A single number instead of [r c] specifies a square filter.
'disk'	fspecial('disk', r). A circular averaging filter (within a square of size 2r + 1) with radius r. The default radius is 5.
'gaussian'	fspecial('gaussian', [r c], sig). A Gaussian lowpass filter of size r × c and standard deviation sig (positive). The defaults are 3 × 3 and 0.5. A single number instead of [r c] specifies a square filter.
'laplacian'	fspecial('laplacian', alpha). A 3 × 3 Laplacian filter whose shape is specified by alpha, a number in the range [0, 1]. The default value for alpha is 0.5.
'log'	fspecial('log', [r c], sig). Laplacian of a Gaussian (LoG) filter of size r × c and standard deviation sig (positive). The defaults are 5 × 5 and 0.5. A single number instead of [r c] specifies a square filter.
'motion'	fspecial('motion', len, theta). Outputs a filter that, when convolved with an image, approximates linear motion (of a camera with respect to the image) of len pixels. The direction of motion is theta, measured in degrees, counterclockwise from the horizontal. The defaults are 9 and 0, which represents a motion of 9 pixels in the horizontal direction.
'prewitt'	fspecial('prewitt'). Outputs a 3 × 3 Prewitt mask, wv, that approximates a vertical gradient. A mask for the horizontal gradient is obtained by transposing the result: wh = wv'.
'sobel'	fspecial('sobel'). Outputs a 3 × 3 Sobel mask, sv, that approximates a vertical gradient. A mask for the horizontal gradient is obtained by transposing the result: sh = sv'.
'unsharp'	fspecial('unsharp', alpha). Outputs a 3 × 3 unsharp filter. Parameter alpha controls the shape; it must be greater than 0 and less than or equal to 1.0; the default is 0.2.

Spatial filters (Mask types) supported by function *fspecial*

Example: Generate average masks

```
>> w = fspecial('average', [3 3])

w =

   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111

>> w = fspecial('average', 3)

w =

   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111

>> w = fspecial('average', [7 5])

w =
```

```
0.0286   0.0286   0.0286   0.0286   0.0286
0.0286   0.0286   0.0286   0.0286   0.0286
0.0286   0.0286   0.0286   0.0286   0.0286
0.0286   0.0286   0.0286   0.0286   0.0286
0.0286   0.0286   0.0286   0.0286   0.0286
0.0286   0.0286   0.0286   0.0286   0.0286
0.0286   0.0286   0.0286   0.0286   0.0286
```

Example: Generate gaussian masks

```
>> w = fspecial('gaussian', [5 5], 1)

w =

   0.0030   0.0133   0.0219   0.0133   0.0030
   0.0133   0.0596   0.0983   0.0596   0.0133
   0.0219   0.0983   0.1621   0.0983   0.0219
   0.0133   0.0596   0.0983   0.0596   0.0133
   0.0030   0.0133   0.0219   0.0133   0.0030

>> w = fspecial('gaussian', [5 5], 2)

w =

   0.0232   0.0338   0.0383   0.0338   0.0232
   0.0338   0.0492   0.0558   0.0492   0.0338
   0.0383   0.0558   0.0632   0.0558   0.0383
   0.0338   0.0492   0.0558   0.0492   0.0338
   0.0232   0.0338   0.0383   0.0338   0.0232

>> w = fspecial('gaussian', [7 7], 3)

w =

   0.0113   0.0149   0.0176   0.0186   0.0176   0.0149   0.0113
   0.0149   0.0197   0.0233   0.0246   0.0233   0.0197   0.0149
   0.0176   0.0233   0.0275   0.0290   0.0275   0.0233   0.0176
   0.0186   0.0246   0.0290   0.0307   0.0290   0.0246   0.0186
   0.0176   0.0233   0.0275   0.0290   0.0275   0.0233   0.0176
   0.0149   0.0197   0.0233   0.0246   0.0233   0.0197   0.0149
   0.0113   0.0149   0.0176   0.0186   0.0176   0.0149   0.0113
```

Example: Plot the distribution of gaussian masks by surface plot

```
>> w1 = fspecial('gaussian',[50 50], 3);
>> surf(1:50, 1:50, w1)

>> w2 = fspecial('gaussian',[50 50], 9);
>> figure, surf(1:50, 1:50, w2)
```

Example: Generate laplacian masks

```
>> w = fspecial('laplacian')

w =

    0.1667    0.6667    0.1667
    0.6667   -3.3333    0.6667
    0.1667    0.6667    0.1667

>> w = fspecial('laplacian',0.2)

w =

    0.1667    0.6667    0.1667
    0.6667   -3.3333    0.6667
    0.1667    0.6667    0.1667

>> w = fspecial('laplacian',1)

w =

    0.5000         0    0.5000
         0   -2.0000         0
    0.5000         0    0.5000
```

Example: Generate laplacian of Gaussian (loG) masks

```
>> w = fspecial('log')

w =

    0.0448    0.0468    0.0564    0.0468    0.0448
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0564    0.7146   -4.9048    0.7146    0.0564
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0448    0.0468    0.0564    0.0468    0.0448

>> w = fspecial('log',5,0.5)

w =

    0.0448    0.0468    0.0564    0.0468    0.0448
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0564    0.7146   -4.9048    0.7146    0.0564
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0448    0.0468    0.0564    0.0468    0.0448

>> w = fspecial('log',3,1)

w =

    0.1004   -0.0234    0.1004
   -0.0234   -0.3079   -0.0234
    0.1004   -0.0234    0.1004
```

➢ **Nonlinear Spatial Filter**

มาส์กแบบไม่เชิงเส้นนั้นเกิดจากการใช้ค่าทางสถิติ (statistic) เช่น ค่ากลาง ค่ามากสุด และค่าต่ำสุดของระดับความเทาของ pixel ที่อยู่ภายในมาส์กมากำหนดเป็นค่าระดับความเทาใหม่ของ pixel ในภาพ ณ ตำแหน่งที่อยู่ตรงกลางมาส์ก ในที่นี้จะกล่าวถึงเฉพาะมาส์กแบบค่ากลาง (median filter) ซึ่งใช้ในการทำให้ภาพเลือนขึ้น (blurring image) เช่นเดียวกันกับมาส์กแบบเชิงเส้นบางชนิด แต่ภาพที่ได้จากการใช้มาส์กแบบค่ากลางจะเลือนน้อยกว่าในขณะที่สามารถกำจัดสัญญาณรบกวนในภาพได้ผลดีมากกว่า โดยเฉพาะสัญญาณรบกวนชนิดเกลือและพริกไทย (salt and pepper noise)

- **Function *medfilt2()***

  Syntax:

  g = medfilt2(f, [m n], padopt)

4.5 การทำให้ภาพเลือน

Blurring Images

การทำให้ภาพเลือนหรือการเบลอภาพนั้น เรียกอีกอย่างว่าการเกลี่ยภาพ (Smoothing) ซึ่งนอกจากจะมีผลทำให้ความคมชัดบริเวณขอบภาพน้อยลงแล้วยังทำให้สัญญาณรบกวน (noise) ภายในภาพลดลงตามไปด้วย โดยทั่วไปแล้วการเบลอภาพนี้จะใช้กำจัดวัตถุขนาดเล็กๆ ในภาพก่อนที่จะทำการแยกส่วนวัตถุขนาดใหญ่ที่ต้องการ (Object Extraction) นอกจากนี้ยังใช้ในการเชื่อมต่อช่องว่างระหว่าง 2 บริเวณที่อยู่ติดกันเข้าด้วยกัน ซึ่ง 2 บริเวณนั้นอาจเป็นอาจเป็นบริเวณของวัตถุอันเดียวกันแต่เกิดความผิดพลาดบางประการที่ทำให้บริเวณทั้งสองขาดออกจากกัน เป็นต้น

หลักการทั่วไปของการทำภาพเบลอนี้ คือ การคำนวณหาค่าระดับความเทาค่าใหม่ให้ตำแหน่ง f(x,y) โดยคำนวณจากค่าเฉลี่ยของระดับความเทาของพิกเซลข้างเคียงภายในมาส์ก การใช้ค่าเฉลี่ยนั้นจะมีผลมากบริเวณที่เป็นเส้นขอบของวัตถุ ตัวอย่างเช่น ถ้าวัตถุมีสีขาวส่วนพื้นหลังภาพคือสีดำหรือเทาแล้ว ระดับความเทาของบริเวณเส้นขอบภาพจะถูกลดค่าลงทำให้มีสีขาวหม่น (เนื่องด้วยค่าเฉลี่ยของบริเวณภายในมาส์กจะมีค่าต่ำกว่าค่าระดับความเทาเดิมของพิกเซลบนเส้นขอบ) ทำให้ความแตกต่างของระดับความเทา ณ บริเวณขอบวัตถุลดลงไป

บริเวณขอบภาพและส่วนที่เป็นสัญญาณรบกวนนั้นเป็นบริเวณที่มีการเปลี่ยนแปลงค่าระดับความเทาสูง ถ้าเราพิจารณาถึงองค์ประกอบความถี่ในบริเวณนี้ก็ย่อมต้องมีค่าสูง การเบลอภาพจึงเป็นการทำให้ค่าองค์ประกอบความถี่สูงของภาพตั้งต้นหายไป ดังนั้น การเบลอภาพจึงเทียบเท่ากับการกรองเอาความถี่สูงออกไป เหลือไว้แต่ส่วนที่มีองค์ประกอบความถี่ต่ำ หรือเรียกได้ว่าเป็นตัวกรองแบบผ่านต่ำ (Low pass Filter)

ตัวอย่างการใช้งาน average mask แสดงดังรูปใน Example 4 และ Example 5 ด้านล่างใน Example 4 ภาพซ้ายบนเป็นภาพตั้งต้นขนาด 500 x 500 พิกเซล ที่มีองค์ประกอบสี่เหลี่ยมจัตุรัสที่อยู่ด้านบน ขนาด 3 x 3 , 5 x 5 , 9 x 9 , 15 x 15 , 25 x 25 , 35 x 35 , 45 x 45 , และ 55 x 55 พิกเซลตามลำดับ ระยะห่างระหว่างแต่ละสี่เหลี่ยมเท่ากับ 25 พิกเซล ตัวอักษร a ที่อยู่ด้านล่างมีขนาด 10 , 12 , 14 , 16 , 18 , 20 , 22 และ 24 จุด ตามลำดับ ตัวอักษร a ตัวใหญ่ด้านบนสุดของภาพมีขนาด 60 จุด บริเวณตรงกลางภาพประกอบด้วยกลุ่มเส้นตรงขนาด 5 x 100 พิกเซลเรียงขนานกัน โดยมีระยะห่างระหว่างกันเท่ากับ 20 พิกเซล กลุ่มภาพวงกลมที่อยู่เหนือกลุ่มภาพเส้นตรงมีขนาดเส้นผ่าศูนย์กลางเท่ากับ 25 พิกเซลและมีระดับความเทาพิ่มทีละ 20% ภาพตั้งต้นนี้มีพื้นภาพเป็นสีเทาและมีสี่เหลี่ยมผืนผ้าขนาด 50 x 120 พิกเซลที่ประกอบด้วยสัญญาณรบกวนภายในอยู่ด้านซ้ายมือของภาพ จากการเบลอภาพตั้งต้นนี้ด้วยมาส์กขนาด 3 x 3 , 5 x 5 , 9 x 9 , 15 x 15 และ 35 x 35 ผลลัพธ์ที่ได้คือ ภาพที่เหลือในรูปเรียงจากบนลงล่างซ้ายไปขวา ตามลำดับ จะเห็นได้ว่ามาส์กขนาดใหญ่จะทำให้วัตถุมีขนาดเล็กกว่าหายไปได้ และทำให้เส้นตรงที่เรียงขนานกันเกิดการเชื่อมต่อด้วยพื้นที่สีเทา

ตัวอย่างการใช้งาน gaussian mask แสดงดังรูปใน Example 6 และ Example 7 ค่าสัมประสิทธิ์ของมาส์กน้ำหนักเฉลี่ยนั้นจะมีค่าสูงสุดที่ตรงกลางและจะมีค่าลดลงเมื่อระยะทางเทียบกับตำแหน่งกลางมาส์กมีค่าสูงขึ้น เหตุผลที่เป็นเช่นนี้ก็เพื่อไม่ทำให้เกิดการเบลอภาพมากจนเกินไป เนื่องจากพิกเซลข้างเคียงที่อยู่ไกลๆ จะไม่มีผลมากเท่ากับพิกเซลที่อยู่ใกล้กว่า ซึ่งสอดคล้องกับหลักการของภาพทั่วไปที่ว่าพิกเซลที่อยู่ใกล้กันมักมีค่าระดับความเทาใกล้เคียงกัน

- **ตัวอย่างการทำให้ภาพเลือนโดยใช้ Linear Spatial Filter**

  Example 4 และ Example 5 เป็นตัวอย่างการทำให้ภาพเลือนโดยใช้มาส์ก average ส่วน Example 6 และ Example 7 เป็นตัวอย่างการทำให้ภาพเลือนโดยใช้มาส์ก Gaussian

Example 4: Blurring image with average masks to 'pattern.tif'

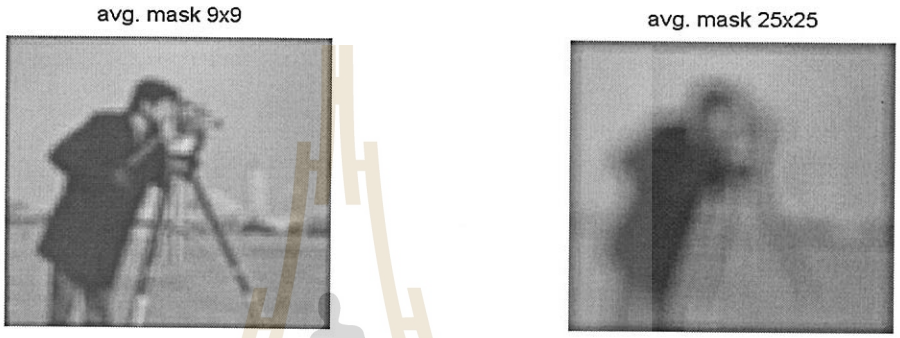
```
>> f = imread('pattern.tif');
>> f = im2double(f);
>> subplot(2,3,1); imshow(f), title('original');

>> w = fspecial('average', [3 3]);
>> gw3 = imfilter(f, w);
>> subplot(2,3,2); imshow(gw3), title('avg. mask 3x3');

>> w = fspecial('average', [5 5]);
>> gw5 = imfilter(f, w);
>> subplot(2,3,3); imshow(gw5), title('avg. mask 5x5');

>> w = fspecial('average', [9 9]);
>> gw9 = imfilter(f, w);
>> subplot(2,3,4); imshow(gw9), title('avg. mask 9x9');
```

```
>> w = fspecial('average', [15 15]);
>> gw15 = imfilter(f, w);
>> subplot(2,3,5); imshow(gw15), title('avg. mask 15x15');

>> w = fspecial('average', [35 35]);
>> gw35 = imfilter(f, w);
>> subplot(2,3,6); imshow(gw35), title('avg. mask 35x35');
```

Example 5: Blurring image with average masks to 'cameraman.tif'

```
>> f = imread('cameraman.tif');
>> f = im2double(f);
>> subplot(2,2,1); imshow(f), title('original');

>> w = fspecial('average');
>> gw3 = imfilter(f, w);
>> subplot(2,2,2); imshow(gw3), title('avg. mask 3x3');

>> w = fspecial('average', 9);
>> gw9 = imfilter(f, w);
>> subplot(2,2,3); imshow(gw9), title('avg. mask 9x9');

>> w = fspecial('average', 25);
>> gw25 = imfilter(f, w);
>> subplot(2,2,4); imshow(gw25), title('avg. mask 25x25');
```

Result of Example 4



Result of Example 5

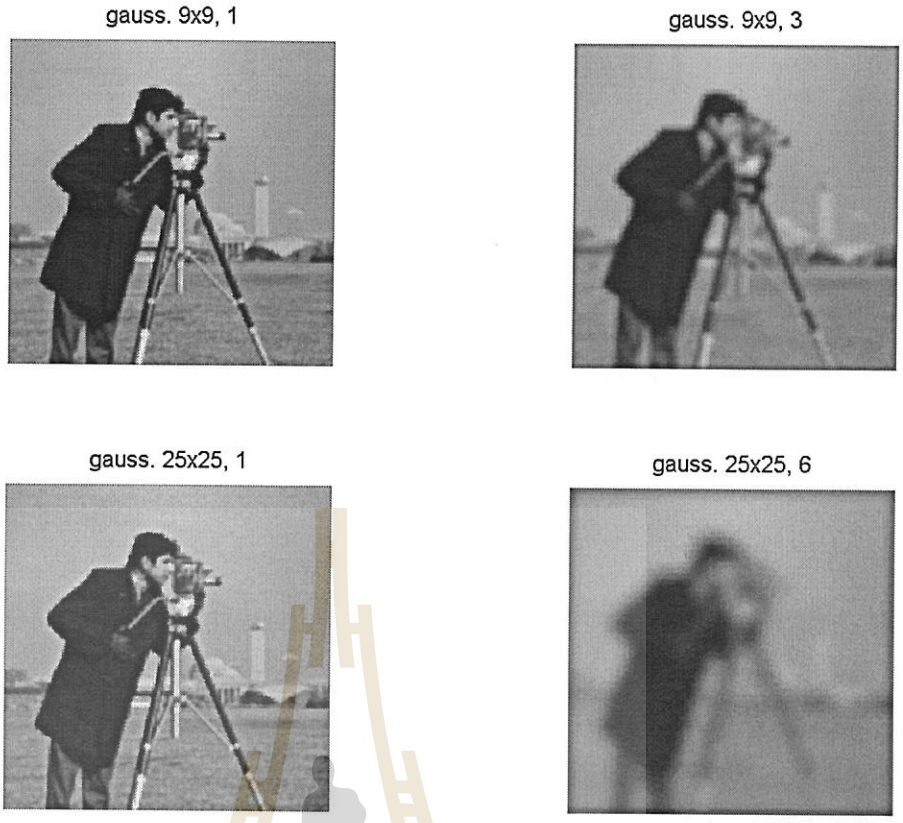
Example 6: Blurring image with gaussian masks to 'pattern.tif'

```
>> f = imread('pattern.tif');
>> f = im2double(f);
>> subplot(2,3,1); imshow(f,[]), title('original');

>> w = fspecial('gaussian', [9 9], 3);
>> gw1 = imfilter(f, w);
>> subplot(2,3,2); imshow(gw1,[]), title('gauss. 9x9, 3');

>> w = fspecial('gaussian', [15 15], 1);
>> gw2 = imfilter(f, w);
>> subplot(2,3,3); imshow(gw2,[]), title('gauss. 15x15, 1');

>> w = fspecial('gaussian', [15 15], 5);
>> gw3 = imfilter(f, w);
>> subplot(2,3,4); imshow(gw3,[]), title('gauss. 15x15, 5');

>> w = fspecial('gaussian', [35 35], 1);
>> gw4 = imfilter(f, w);
>> subplot(2,3,5); imshow(gw4,[]), title('gauss. 35x35, 1');

>> w = fspecial('gaussian', [35 35], 5);
>> gw5 = imfilter(f, w);
>> subplot(2,3,6); imshow(gw5,[]), title('gauss. 35x35, 5');
```

Result of Example 6

Example 7: Blurring image with gaussian masks to 'cameraman.tif'

```
>> f = imread('cameraman.tif');
>> f = im2double(f);

>> w = fspecial('gaussian', [9 9], 1);
>> gw1 = imfilter(f, w);
>> subplot(2,2,1); imshow(gw1), title('gauss. 9x9, 1');

>> w = fspecial('gaussian', [9 9], 3);
>> gw2 = imfilter(f, w);
>> subplot(2,2,2); imshow(gw2), title('gauss. 9x9, 3');

>> w = fspecial('gaussian', [25 25], 1);
>> gw3 = imfilter(f, w);
>> subplot(2,2,3); imshow(gw3), title('gauss. 25x25, 1');

>> w = fspecial('gaussian', [25 25], 6);
>> gw4 = imfilter(f, w);
>> subplot(2,2,4); imshow(gw4), title('gauss. 25x25, 6');
```

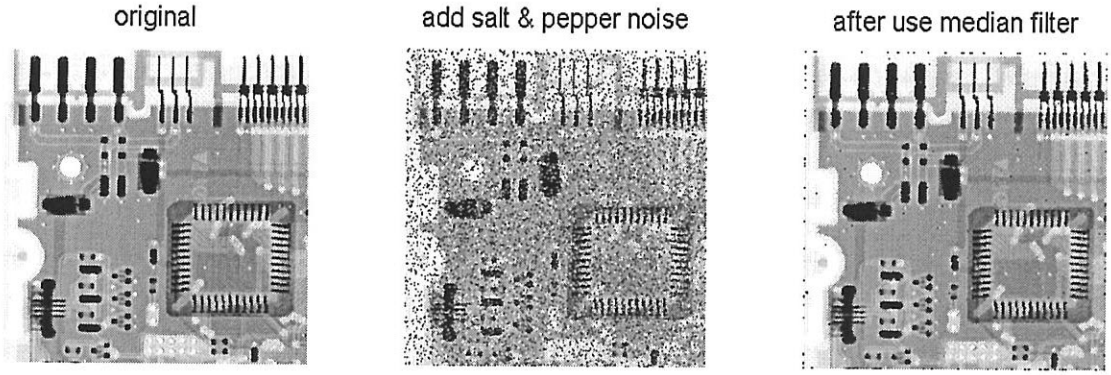


Result of Example 7

- **ตัวอย่างการทำให้ภาพเลือนและการกำจัด noise โดยใช้ Nonlinear Spatial Filter**

Example 8 : Blurring image with gaussian masks to 'cameraman.tif'

```
>> f = imread('board.tif');
>> subplot(1,3,1); imshow(f), title('original');

>> g = imread('st_pp_board.tif');
>> g = im2double(g);
>> subplot(1,3,2); imshow(g), title('add salt & pepper noise');

>> fn = medfilt2(g);
>> subplot(1,3,3); imshow(fn), title('after use median filter');
```



Result of Example 8

4.6 การทำให้ภาพคม

Sharpening Images

การทำให้ภาพคมนั้นเป็นการกระทำที่ตรงกันข้ามกับการทำให้ภาพเบลอ กล่าวคือ เป็นการทำให้บริเวณที่มีการเปลี่ยนแปลงระดับความเทาสูง (บริเวณขอบภาพของวัตถุและสัญญาณรบกวน) มีความเด่นชัดมากขึ้นเราสามารถนำเทคนิคการทำให้ภาพคมมาใช้ปรับปรุงภาพที่ลักษณะเบลอจากการถ่ายภาพหรือความผิดพลาดในขั้นตอนการรับภาพ (Image Acquisition) ให้มีความคมชัดมากขึ้น การทำให้ภาพคมจึงเทียบเท่ากับการกรองเอาความถี่ต่ำออกไป เหลือไว้แต่ส่วนที่มีองค์ประกอบความถี่สูง หรือเรียกได้ว่าเป็นตัวกรองแบบผ่านสูง (High pass Filter)

หลักการของการทำให้ภาพคมชัดมีการใช้อัตราการเปลี่ยนแปลงของระดับความเทาของพิกเซลภายในมาส์ก หรือ อนุพันธ์เชิงระยะทาง (Spatial Differentiation) ซึ่งจะเห็นได้ว่าเป็นหลักการที่ตรงกันข้ามกับการเบลอภาพซึ่งทำการหาค่าผลรวม (Integration) ของค่าระดับความเทาของพิกเซลภายในมาส์ก

การทำให้ภาพคมชัดทำได้โดยการกำหนดให้ทำการขยายค่าระดับความเทาให้มีค่ามากขึ้นสำหรับบริเวณที่มีค่าอนุพันธ์สูง และในทางตรงกันข้ามบริเวณที่มีค่าอนุพันธ์ต่ำจะถูกลดค่าระดับความเทาลง

พิจารณาค่าอนุพันธ์อันดับที่ 1 (First – order Derivation) ของฟังก์ชัน f(x) ดังสมการต่อไปนี้

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

สมการ 1

เราสามารถคำนวณหาค่าอนุพันธ์อันดับที่ 2 (Second – order Derivation) ได้ดังนี้

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

สมการ 2

เนื่องจากสัญญาณภาพเป็นฟังก์ชันของตัวแปร x และ y ดังนั้น สมการที่ 1 และ 2 ในกรณีที่มีตัวแปรมากกว่าหนึ่งตัวแปรจะมีค่าเท่ากับสมการที่ 3 และ 4 ตามลำดับ โดยฟังก์ชันของทั้งสองสมการในที่นี้เป็นฟังก์ชันเชิงเส้นแบบต่อเนื่อง

Gradient operator : $\nabla f = \frac{\partial f(x,y)}{\partial x \partial y} = \frac{\partial f(x,y)}{\partial x} + \frac{\partial f(x,y)}{\partial y}$

สมการ 3

Laplacian operation : $\nabla^2 f = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$

สมการ 4

ในกรณีที่เราพิจารณาภาพดิจิตอลซึ่งเป็นสัญญาณดีสครีต เราสามารถเขียนสมการ 4 ได้เป็น

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

สมการ 5

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

สมการ 6

เมื่อนำสมการ 5 3-37 และ 6 3-38 มารวมกัน จะได้

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$

สมการ 7

เมื่อนำค่าสัมประสิทธิ์ต่างๆของสมการ 7 ด้านขวามือมาใช้เป็นสัมประสิทธิ์ของมาส์กขนาด 3 x 3 เราจะได้มาส์กลาปลาเซียน (Laplacian Mask) ที่มีค่าดังแสดงในรูปที่ 1 รวมทั้งมาส์กที่ได้มีการปรับปรุงค่าสัมประสิทธิ์ของมาส์กลาปลาเซียน แสดงดังรูปที่ 2

0	1	0
1	-4	1
0	1	0

รูปที่ 1 มาส์กลาปลาเซียน

1	1	1
1	-8	1
1	1	1

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

รูปที่ 2 มาส์กลาปลาเซียนอื่นๆ

เนื่องจากภาพผลลัพธ์ที่ได้จากการใช้มาส์กลาปลาเซียนนั้นจะกำจัดส่วนที่เป็นพื้นภาพและส่วนที่มีการเปลี่ยนแปลงระดับความเทาต่ำ หากเรานำเอาภาพผลลัพธ์ที่ได้จากการใช้มาส์กลาปลาเซียนมารวมกับภาพตั้งต้นแล้วจะทำให้เราได้ภาพผลลัพธ์ที่มีรายละเอียดของภาพครบทุกส่วน อีกทั้งมีการทำให้บริเวณขอบภาพคมชัดมากยิ่งขึ้น การบวกผลลัพธ์จากการใช้มาส์กลาปลาเซียนกับภาพตั้งต้น เขียนเป็นสมการได้ดังนี้

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) \\ f(x,y) + \nabla^2 f(x,y) \end{cases}$$

สมการ 8

โดยสมการที่เป็นผลต่าง (-) นั้นจะใช้ในกรณีที่มาส์กลาปลาเซี่ยนมีค่าสัมประสิทธิ์ตรงกลางเป็นค่าลบและสมการที่เป็นผลบวก (+) นั้นจะใช้ในกรณีที่มาส์ก ลาปลาเซียนมีค่าสัมประสิทธิ์ตรงกลางเป็นค่าบวก เราสามารถกระจายสมการ 8 ได้เป็น

$$g(x,y) = f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) + 4f(x,y)]$$

สมการ 9

$$= 5f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(xy-1)]$$

และเมื่อนำค่าสัมประสิทธิ์ในสมการที่ 9 มาเขียนเป็นมาส์กจะได้ดังรูปที่ 3

0	-1	0
-1	5	-1
0	-1	0

-1	-1	-1
-1	9	-1
-1	-1	-1

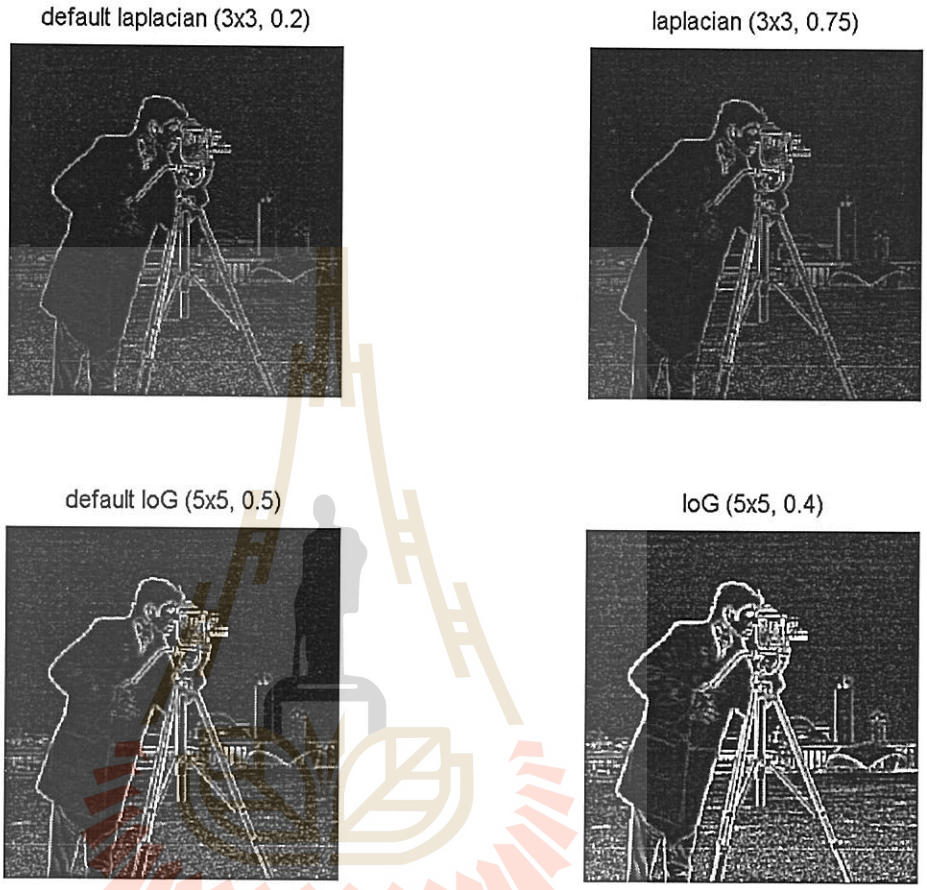
รูปที่ 3 มาส์กผลรวมค่าตั้งต้นกับลาปลาเซียน

- **ตัวอย่างการทำให้ภาพการทำให้ภาพคมโดยใช้ Linear Spatial Filter**

Example 9 : Sharpening image of 'cameraman.tif' by laplacian and log mask

```
>> f = imread('cameraman.tif');
>> f = im2double(f);

>> w = fspecial('laplacian');
>> fw1 = imfilter(f, w);
>> subplot(2,2,1); imshow(fw1), title('default laplacian (3x3, 0.2)');

>> w = fspecial('laplacian',0.75);
>> fw2 = imfilter(f, w);
>> subplot(2,2,2); imshow(fw2), title('laplacian (3x3, 0.75)');

>> w = fspecial('log');
```

```
>> fw3 = imfilter(f, w);
>> subplot(2,2,3); imshow(fw3), title('default loG (5x5, 0.5)');

>> w = fspecial('log', [5 5], 0.4);
>> fw4 = imfilter(f, w);
>> subplot(2,2,4); imshow(fw4), title('loG (5x5, 0.4)');
```



Result of Example 9

บทที่ 5: การประมวลผลทางสัญฐานของภาพ
Morphological Image Processing

Morphological processing comes from the word "morphing" in Biology which means "changing a shape".

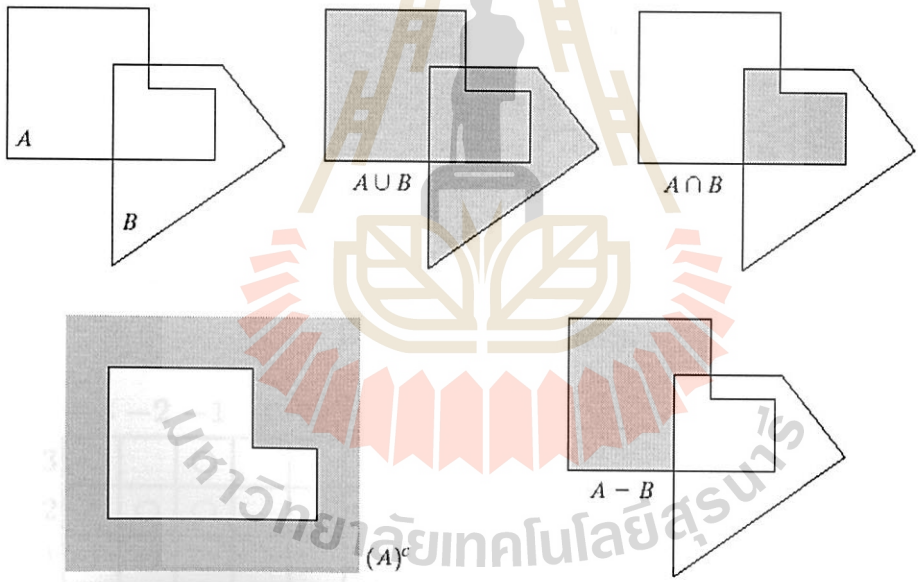
Image morphological processing is used to manipulate object shapes such as thinning, thickening, and filling.

5.1 ความรู้เบื้องต้นเกี่ยวกับการประมวลผลทางสัญฐานของภาพ

- **Some Basic Concepts from Set Theory**

  Concept of a set in binary image morphology:

  Each set may represent one object. Each pixel (x,y) has its status: belong to a set or not belong to a set.



a b c
d e

FIGURE 9.1
(a) Two sets A and B. (b) The union of A and B. (c) The intersection of A and B. (d) The complement of A. (e) The difference between A and B.

Set Operation	MATLAB Expression for Binary Images	Name
$A \cap B$	A & B	AND
$A \cup B$	A | B	OR
A^c	~A	NOT
$A - B$	A & ~B	DIFFERENCE

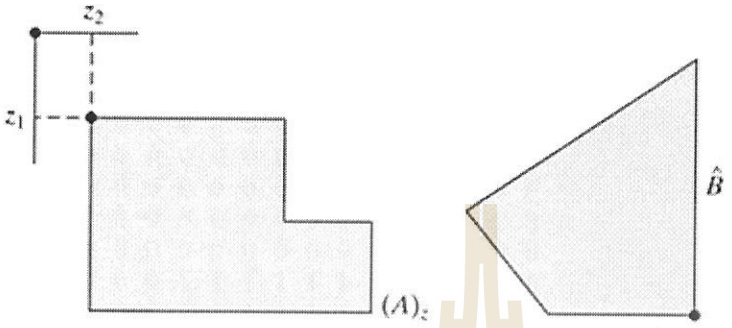
- **Translation and Refection**

Translation:

$$(A)_z = \{c \mid c = a + z, \quad \textbf{for } a \in A\}$$

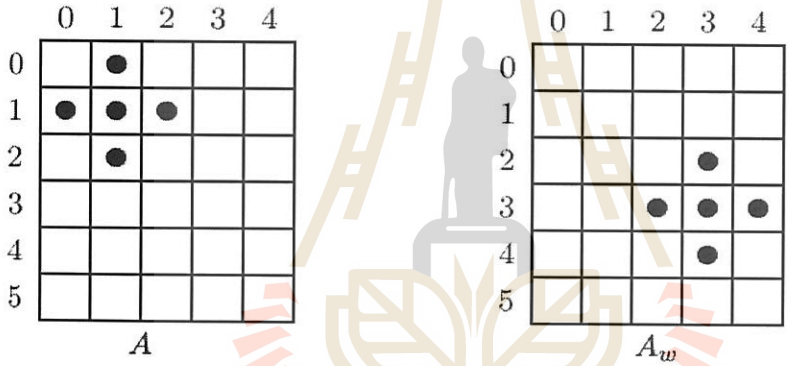
Reflection:

$$\hat{B} = \{w \mid w = -b, \quad \textbf{for } b \in B\}$$

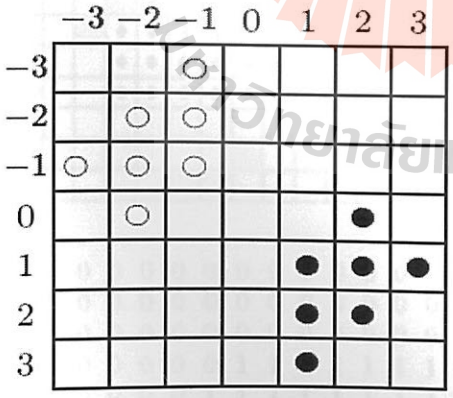


a b

FIGURE 9.2
(a) Translation of A by z.
(b) Reflection of B. The sets A and B are from Fig. 9.1.



Example of Translation



Example of Reflection

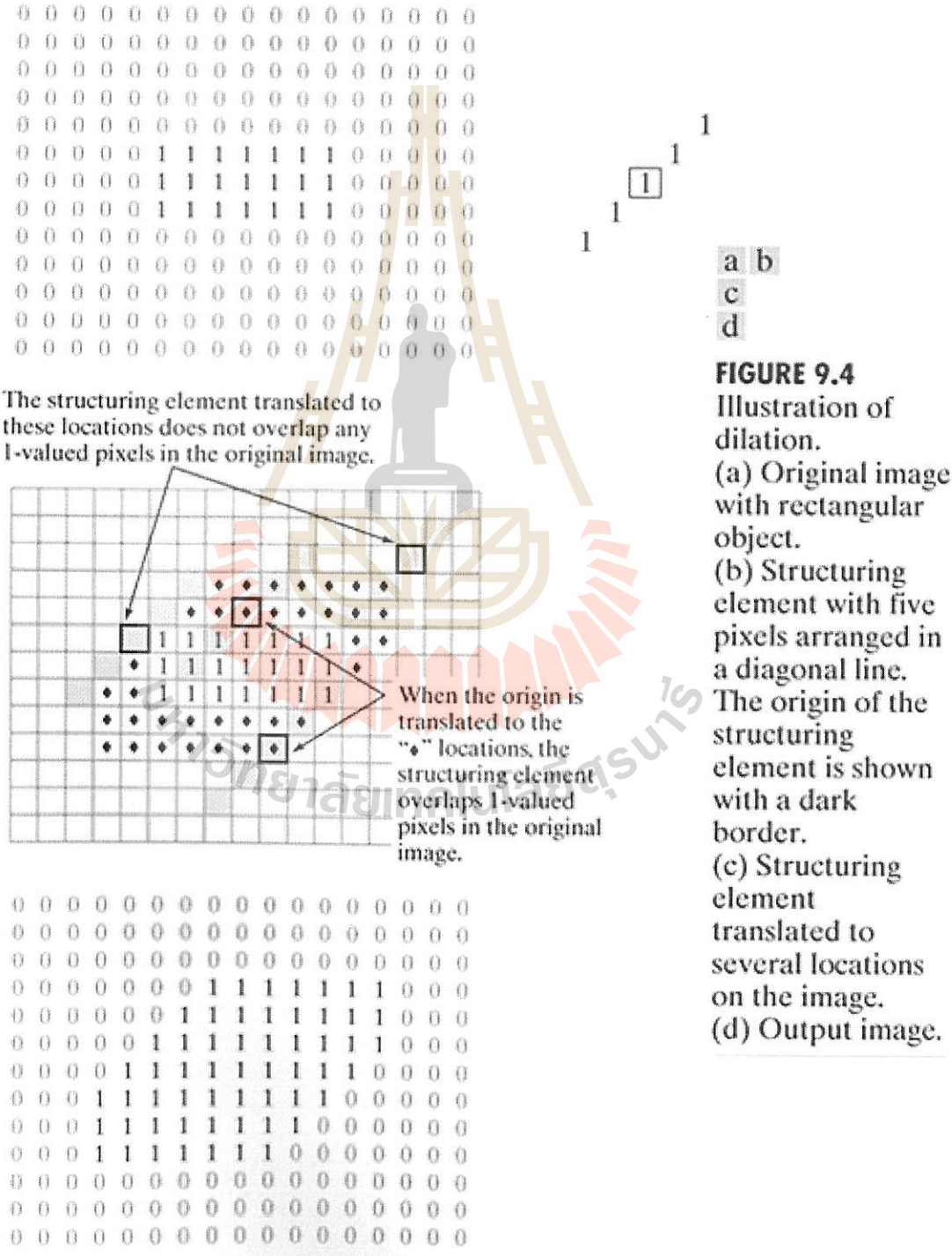
5.2 Dilation and Erosion

- **Dilation**

  Dilation is an operation that "grows" or "thickens" objects in a binary image. The specific manner and extent of this thickening is controlled by a shape referred to as a *structuring element.*

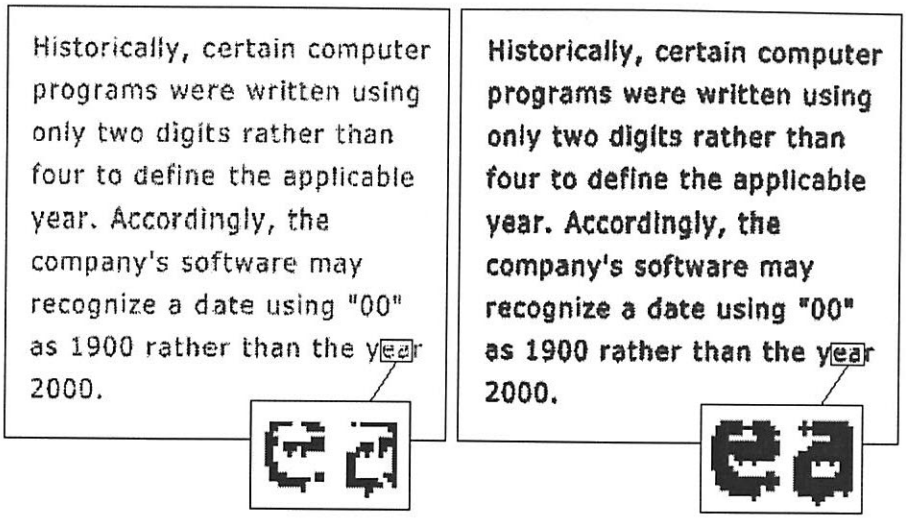
  Mathematically, dilation is defined in terms of set operations. The dilation of *A by B,* denoted $A \oplus B$ is defined as

$$A \oplus B = \left\{ z \middle| \left(\hat{B}\right)_z \cap A \neq \phi \right\}$$



a b
c
d

FIGURE 9.4 Illustration of dilation. (a) Original image with rectangular object. (b) Structuring element with five pixels arranged in a diagonal line. The origin of the structuring element is shown with a dark border. (c) Structuring element translated to several locations on the image. (d) Output image.



a c
b

FIGURE 9.5
(a) Sample text of poor resolution with broken characters (magnified view). (b) Structuring element. (c) Dilation of (a) by (b). Broken segments were joined.

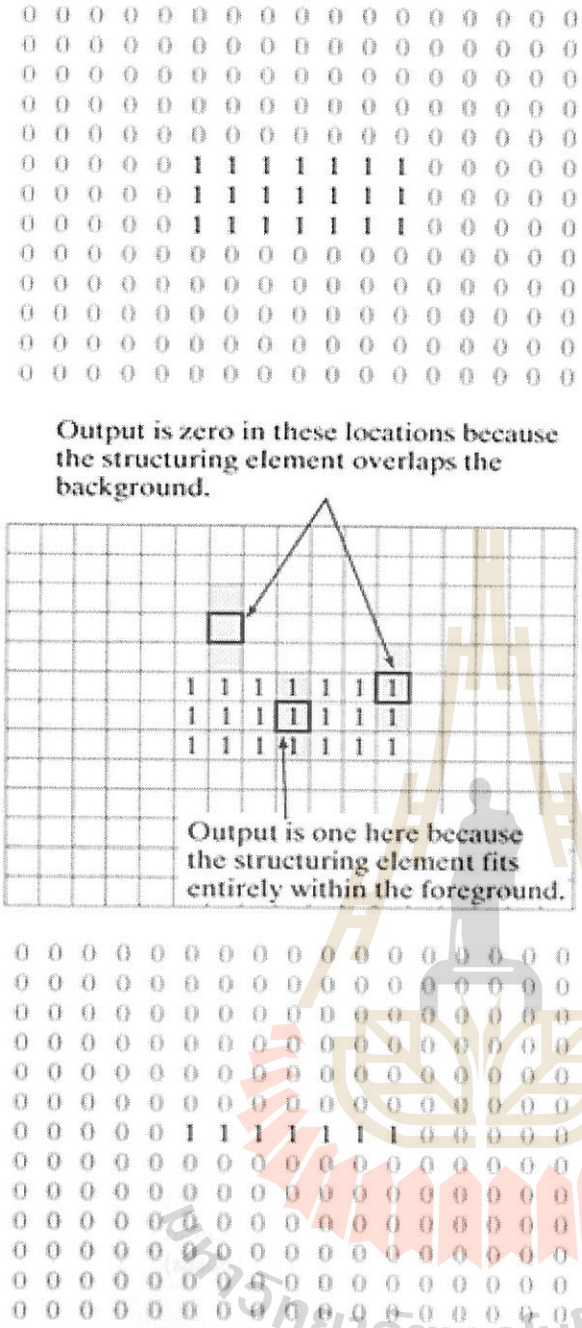
0	1	0
1	1	1
0	1	0

- **Erosion**

Erosion is an operation that "shrinks" or "thins" objects in a binary image. The specific manner and extent of shrinking is controlled by a shape referred to as a *structuring element* also.

Mathematically, dilation is defined in terms of set operations. The dilation of *A by B,* denoted $A \Theta B$ is defined as

$$A \Theta B = \left\{ z \middle| (B)_z \cap A^c \neq \phi \right\}$$



a b
c
d

FIGURE 9.7 Illustration of erosion. (a) Original image with rectangular object. (b) Structuring element with three pixels arranged in a vertical line. The origin of the structuring element is shown with a dark border. (c) Structuring element translated to several locations on the image. (d) Output image.

5.3 Combining Dilation and Erosion

- **Opening and Closing**
  - **Opening:** Erosion follows by Dilation

    The morphology opening of A by B, denoted by $A \circ B$, is simply erosion of A by B, followed by dilation of the result by B

    $$A \circ B = (A \ominus B) \oplus B$$

An alternative mathematical formulation of opening is

$$A \circ B = \bigcup \{(B)_z \,|\, (B)_z \subseteq A\}$$

That is $A \circ B$ is the union of all translations of B that fit entirely within A

- **Closing:** Dilation follows by Erosion

The morphology closing of A by B, denoted by $A \bullet B$, is simply dilation of A by B, followed by erosion of the result by B

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

That is $A \bullet B$ is the complement of the union of all translation of B that do not overlap A.

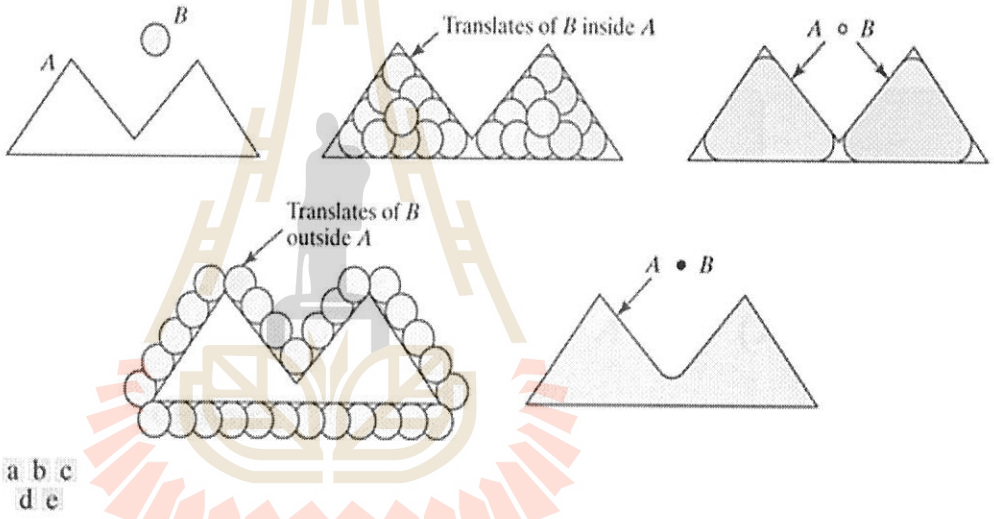
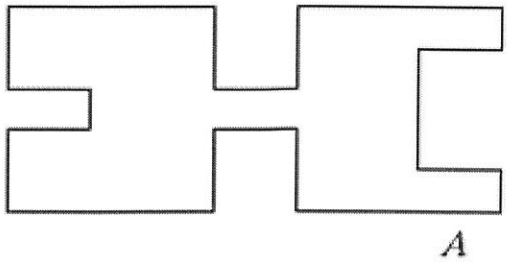
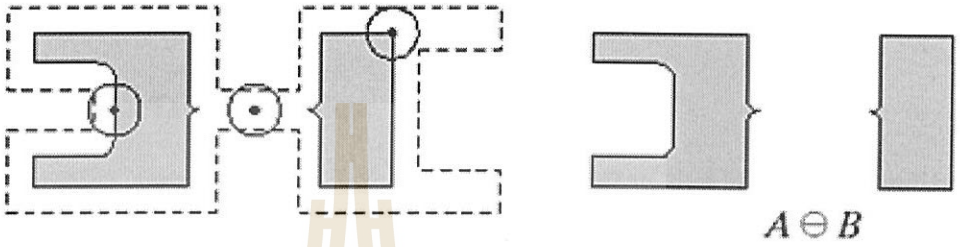
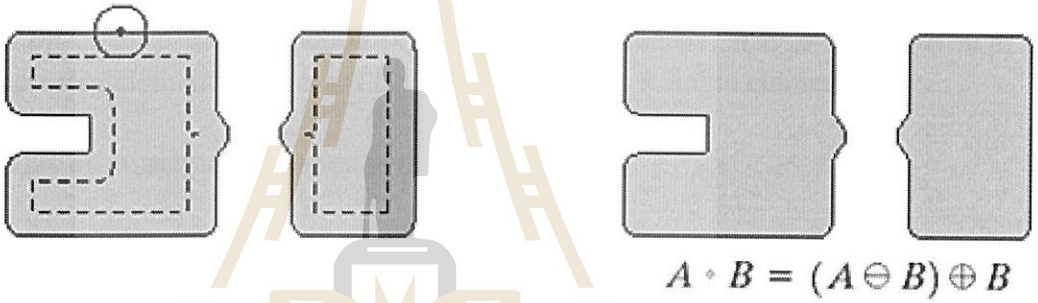


FIGURE 9.9 Opening and closing as unions of translated structuring elements. (a) Set A and structuring element B. (b) Translations of B that fit entirely within set A. (c) The complete opening (shaded). (d) Translations of B outside the border of A. (e) The complete closing (shaded).

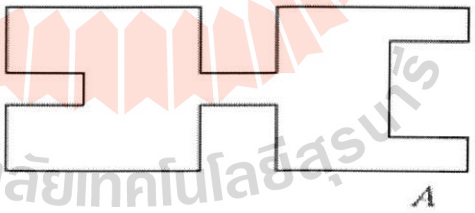
Example of Opening:

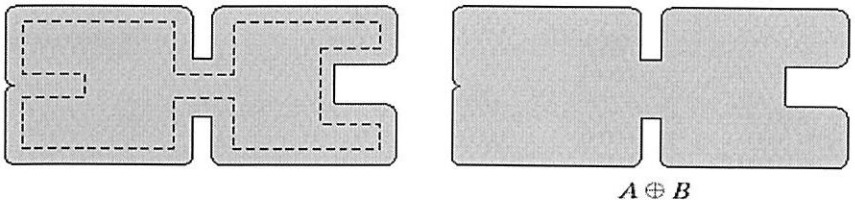


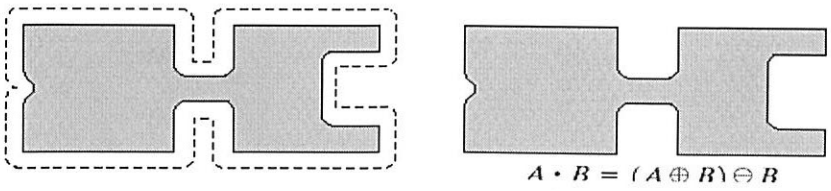




Example of Closing:







a b c

FIGURE 9.11 (a) Noisy fingerprint image. (b) Opening of image. (c) Opening followed by closing. (Original image courtesy of the National Institute of Standards and Technology.)

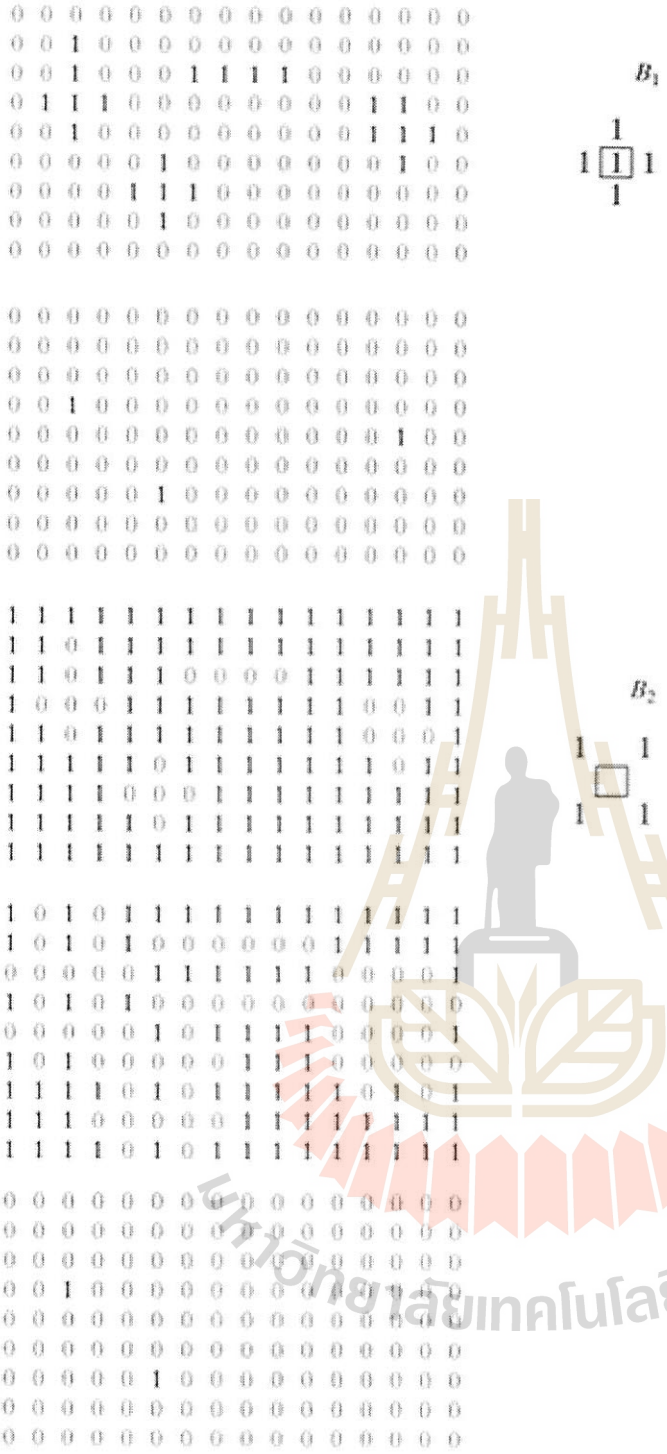
- **The Hit-or-Miss Transformation**

  The hit-or-miss transformation is useful for application that wants to identify specified configurations of pixels, such as isolated foreground pixels, or pixels that are end points of line segments

  The hit-or-miss transformation of A by B is denoted by $A \otimes B$ where B is a structuring element pair, $B = (B_1, B_2)$, rather than a single element,

  The hit-or-miss transformation is defined as

$$A \otimes B = (A \Theta B_1) \cap (A^c \Theta B_2)$$



a b
c
d e
f
g

FIGURE 9.12
(a) Original image A. (b) Structuring element B_1.
(c) Erosion of A by B_1.
(d) Complement of the original image, A^c. (e) Structuring element B_2.
(f) Erosion of A^c by B_2. (g) Output image.

5.4 Morphology Implement by Matlab

- **Logical expressions for binary images**

```
>> A = imread('utk.tif');
>> B = imread('gt.tif');
```

```
>> imshow(A)
>> figure, imshow(B)
>> C = ~A;
>> figure, imshow(C)
>> D = A|B;
>> figure, imshow(D)
>> E = A&B;
>> figure, imshow(E)
>> F = A&~B;
>> figure, imshow(F)
```

- **Dilation**

```
>> A = imread('broken_text.tif');
>> imshow(A)
>> B = [0 1 0; 1 1 1; 0 1 0]

B =

    0    1    0
    1    1    1
    0    1    0

>> A2 = imdilate(A, B);
>> figure, imshow(A2)
```

- **Erosion**

```
>> A = imread('wirebond.tif');
>> subplot(2,2,1); imshow(A);

>> se = strel('disk', 10);
>> A2 = imerode(A, se);
>> subplot(2,2,2); imshow(A2);

>> se = strel('disk', 5);
>> A3 = imerode(A, se);
>> subplot(2,2,3); imshow(A3);

>> A4 = imerode(A, strel('disk', 20));
>> subplot(2,2,4); imshow(A4);
```

- **The strel Function**

Syntax Forms	Description
`se = strel('diamond', R)`	Creates a flat, diamond-shaped structuring element, where `R` specifies the distance from the structuring element origin to the extreme points of the diamond.
`se = strel('disk', R)`	Creates a flat, disk-shaped structuring element with radius `R`. (Additional parameters may be specified for the disk; see the `strel` help page for details.)
`se = strel('line', LEN, DEG)`	Creates a flat, linear structuring element, where `LEN` specifies the length, and `DEG` specifies the angle (in degrees) of the line, as measured in a counterclockwise direction from the horizontal axis.
`se = strel('octagon', R)`	Creates a flat, octagonal structuring element, where `R` specifies the distance from the structuring element origin to the sides of the octagon, as measured along the horizontal and vertical axes. `R` must be a nonnegative multiple of 3.
`se = strel('pair', OFFSET)`	Creates a flat structuring element containing two members. One member is located at the origin. The second member's location is specified by the vector `OFFSET`, which must be a two-element vector of integers.
`se = strel('periodicline', P, V)`	Creates a flat structuring element containing `2*P + 1` members. `V` is a two-element vector containing integer-valued row and column offsets. One structuring element member is located at the origin. The other members are located at `1*V`, `-1*V`, `2*V`, `-2*V`, ..., `P*V`, and `-P*V`.
`se = strel('rectangle', MN)`	Creates a flat, rectangle-shaped structuring element, where `MN` specifies the size. `MN` must be a two-element vector of nonnegative integers. The first element of `MN` is the number rows in the structuring element; the second element is the number of columns.
`se = strel('square', W)`	Creates a square structuring element whose width is `W` pixels. `W` must be a nonnegative integer scalar.
`se = strel('arbitrary', NHOOD)` `se = strel(NHOOD)`	Creates a structuring element of arbitrary shape. NHOOD is a matrix of 0s and 1s that specifies the shape. The second, simpler syntax form shown performs the same operation.

TABLE 9.2 The various syntax forms of function `strel`. (The word *flat* means that the structuring element has zero height. This is meaningful only for gray-scale dilation and erosion. See Section 9.6.1.)

- **Dilation and Erosion**

```
>> rice = imread('rice.tif');      %gray scale image
>> r = rice>135;                   %change to binary image
>> subplot(2,2,1); imshow(rice); xlabel('rice.tif');

>> subplot(2,2,2); imshow(r); xlabel(' r = rice>135');

>> se = ones(3,3);

>> re=imerode(r,se);
>> subplot(2,2,3); imshow(re); xlabel('re=imerode(r,se)');


>> r_int = r&~re;
>> subplot(2,2,4); imshow(r_int); xlabel('rint = r&~re ');


>> rd=imdilate(r,se);
>> figure, subplot(2,2,1); imshow(rd); xlabel('rd=imdilate(r,se)');
>> subplot(2,2,2); imshow(~r); xlabel('~r');


>> r_ext=rd&~r;
>> subplot(2,2,3); imshow(r_ext); xlabel('rext=rd&~r');


>> r_grad=rd&~re;
>> subplot(2,2,4); imshow(r_grad); xlabel('rgrad=rd&~re');
```

- **Opening and Closing**

Example1:

```
>> f = imread('shapes.tif');
>> se = strel('square', 20);
>> fo = imopen(f, se);
>> subplot(2,2,1); imshow(f); xlabel('original');
>> subplot(2,2,2); imshow(fo); xlabel('opening');
>> fc = imclose(f,se);
>> subplot(2,2,3); imshow(fc); xlabel('closing');
>> foc = imclose(fo, se);
>> subplot(2,2,4); imshow(foc); xlabel('open then close');
```

Example2:

```
>> f = imread('fingerprint.tif');
>> subplot(1,3,1); imshow(f); xlabel('original');
>> se = strel('square', 3);
>> fo = imopen(f,se);
>> subplot(1,3,2); imshow(fo); xlabel('opening');
>> foc = imclose(fo, se);
>> subplot(1,3,3); imshow(foc); xlabel('open then close');
```

- **Hit-or-Miss Transformation**
  function: *bwhitmiss(A, B1, B2)*

```
>> f = imread('squares.tif');
>> subplot(1,2,1); imshow(f);
>> B1 = strel([0 0 0; 0 1 1; 0 1 0]);
>> B2 = strel([1 1 1; 1 0 0; 1 0 0]);
>> g = bwhitmiss(f, B1, B2);
>> subplot(1,2,2); imshow(g); pixval on;
```

แบบฝึกหัด ชุดที่ 1

1. จงอธิบายในแต่ละข้อต่อไปนี้

   (a). Digital Image Processing คือ

   (b). วัตถุประสงค์หลักๆ ของการประมวลผลภาพดิจิตอลคืออะไร

   (c). Image คือ

   (d). Quantization คือ

   (e). Sampling คือ

2. จงหาขนาดของไฟล์ (file size) ในแต่ละข้อต่อไปนี้ (อ้างอิง data class ของ Matlab) ออกมาเป็นหน่วยของ bytes เมื่อรูปมีขนาดเป็น M x N ใดๆ โดยกำหนดเงื่อนไขของรูปคือ

   (a). รูปขาวดำ (black-white image) class แบบ logical

   (b). รูประดับเทา (gray-scale image) class แบบ uint8

   (c). รูประดับเทา (gray-scale image) class แบบ uint16

   (d). รูปสีแบบ true color (RGB image) class แบบ uint8

   (e). รูปสีแบบ true color (RGB image) class แบบ uint16

   (f). รูปสีแบบ true color (RGB image) class แบบ double

   (g). รูปสีแบบ indexed color (indexed image) class ของ index เป็น uint8 และ table เป็น double

   (h). รูปสีแบบ indexed color (indexed image) class ของ index เป็น uint16 และ table เป็น double

3. จงหาค่าของ Z ในแต่ละข้อต่อไปนี้

   (a).

```
>> X = uint8([ 255 0 75; 44 225 100]);
>> Y = uint8([ 50 50 50; 50 50 50 ]);
>> Z = imadd(X,Y)
```

   (b).

```
>> X = uint16([ 255 0 75; 44 225 100]);
>> Y = uint16([ 50 50 50; 50 50 50 ]);
>> Z = imadd(X,Y)
```

   (c).

```
>> X = uint8([ 255 10 75; 44 225 100]);
>> Y = uint8([ 50 50 50; 50 50 50 ]);
```

```
>> Z = imsubtract(X,Y)
```

(d).

```
>> X = uint16([ 255 10 75; 44 225 100]);
>> Y = uint16([ 50 50 50; 50 50 50 ]);
>> Z = imsubtract(X,Y)
```

(e).

```
>> X = uint8([ 255 10 75; 44 225 100]);
>> Y = uint8([ 50 50 50; 50 50 50 ]);
>> Z = imabsdiff(X,Y)
```

(f).

```
>> X = uint16([ 50 10 75; 44 425 100]);
>> Y = uint16([ 355 50 50; 50 50 50 ]);
>> Z = imabsdiff(X,Y)
```

(g).

```
>> X = uint16([ 255 10 75; 44 225 100]);
>> Z = imcomplement(X)
```

4. ถ้าต้องการเอาภาพ gray scale 2 ภาพมาบวกกันด้วยคำสั่ง imadd โดยทั้งสองภาพมี class เป็น uint8 และเป็นภาพแบบ bright image มากๆ ทั้ง 2 ภาพ

   (a). หลังจากเอา 2 ภาพมาบวกกันแล้วผลลัพธ์ที่ได้จะเป็นอย่างไร

   (b). ถ้าต้องการแก้ไขผลลัพธ์ที่ได้จากข้อ (a) เพื่อต้องการให้เห็นรายละเอียดที่ชัดเจนของภาพเราจะต้องมีการ จัดการอย่างไร จงอธิบาย (อาจยกตัวอย่าง code ใน Matlab ด้วยก็ได้)

5. Midterm Take Home **(เป็นข้อนึงซึ่งมีแน่นอนในข้อสอบ midterm)**

   ถ้าต้องการเอาภาพ gray scale 2 ภาพมาลบกันด้วยคำสั่ง imsubtract โดยทั้งสองภาพมี class เป็น uint8 และเป็นภาพแบบ dark image มากๆ ทั้ง 2 ภาพ

   (a). หลังจากเอา 2 ภาพมาลบกันแล้วผลลัพธ์ที่ได้จะเป็นอย่างไร

   (b). ถ้าต้องการแก้ไขผลลัพธ์ที่ได้จากข้อ (a) เพื่อต้องการให้เห็นรายละเอียดที่ชัดเจนของภาพเราจะต้องมีการ จัดการอย่างไร จงอธิบาย (อาจยกตัวอย่าง code ใน Matlab ด้วยก็ได้)

6. จงอธิบายในแต่ละข้อต่อไปนี้

   (a). Contrast คือ

   (b). Histogram คือ

   (c). Histogram Equalization คือ

   (d). Histogram Matching คือ

   (e). Histogram Equalization มีข้อเสียอย่างไร

(f). Histogram Equalization ต่างจาก Histogram Matching อย่างไร

7. จากตารางข้อมูลด้านล่างในแต่ละข้อ จง plot histogram อย่างๆ คร่าว และจงหา Histogram Equalization จากข้อมูลดังกล่าว เมื่อข้อมูลบรรทัดบนคือ gray-level ซึ่งมี 0-15 ระดับ บรรทัดล่างคือจำนวน pixel ของในแต่ละระดับของ gray-level

(a).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	40	60	75	80	75	65	55	50	45	40	35	30	25	20	30

(b).

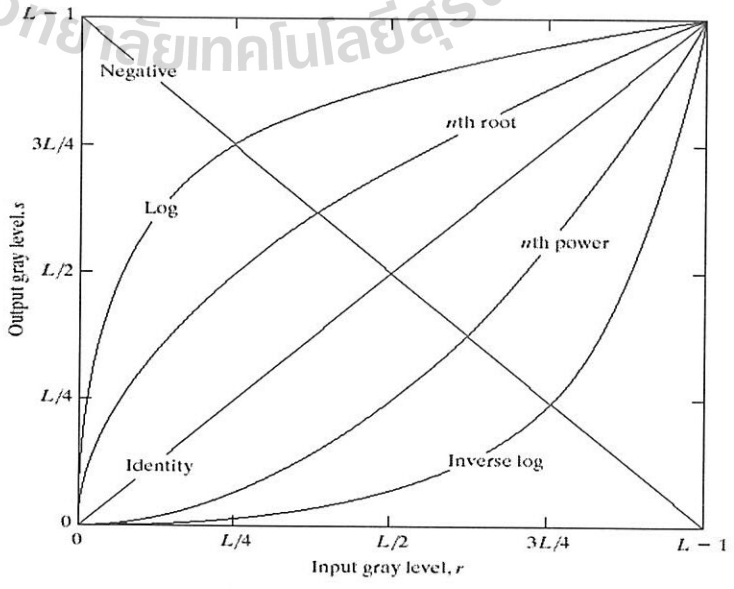
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	40	80	45	110	70	0	0	0	0	0	0	0	0	15

8. ในการปรับค่า intensity ในภาพแบบ gray scale เพื่อให้มีคุณภาพของภาพที่ดีขึ้น โดยใช้หลักการของ intensity transformation ซึ่งเลือกใช้ฟังก์ชันการแปลงแบบยกกำลัง (Power-law) จากลักษณะของภาพในแต่ละข้อต่อไปนี้ ค่าแกมมา (γ) ที่ควรเลือกใช้ในฟังก์ชันแบบยกกำลังดังกล่าวควรมีค่าเป็นอย่างไร เพราะอะไร จงอธิบาย

(a). ภาพ low contrast แบบ dark image

(b). ภาพ low contrast แบบ bright image

9. จากกราฟที่ให้มา เป็นการแสดงความสัมพันธ์ระหว่าง input intensity กับ output intensity ของฟังก์ชันในการ Transformation แบบต่างๆ จงใช้กราฟนี้ในการตอบคำถามข้อ 6.1-6.3 (ในแต่ละข้ออาจตอบได้มากกว่า 1 แบบ)



9.1 ฟังก์ชันในการ Transformation แบบใดบ้าง ที่สามารถช่วยให้ภาพที่เป็นแบบ bright image กลายเป็นภาพที่มืดขึ้นได้

9.2 ถ้าต้องการทำให้ pixel ที่มีระดับความเทาต่ำในภาพมีค่ายิ่งต่ำลงไปอีก ในขณะที่ pixel ที่สว่างถูกทำให้สว่างมากขึ้นไปอีกควรเลือกใช้ฟังก์ชันในการ Transformation แบบใด

9.3 ภาพที่มีองค์ประกอบสีดำจำนวนมาก ถ้าต้องการทำให้บริเวณที่มีสีขาวหรือเทาบนพื้นภาพสีดำมีความ เด่นชัดมากยิ่งขึ้นควรเลือกใช้กราฟใด

9.4 ถ้าข้อมูลภาพเดิมมีเฉพาะสีดำกับสีขาว สามารถใช้ฟังก์ชันในการ Transformation แบบใดในการเปลี่ยนจากสีขาวเป็นสีดำและสีดำเป็นสีขาว

9.5 Etc,.

10. จงหาผลลัพธ์ที่ได้จากการทำ mask processing ด้วยมือแบบ linear filter จากข้อมูลภาพที่ให้มา กับแต่ละ mask ในข้อ (a)-(h) โดยการทำแบบ correlaton ('corr') และใช้หลักการ zeros padding คือเติม 0 รอบๆ ภาพ (อาจลองตรวจสอบคำตอบโดยการใช้คำสั่งของ Matlab)

20	20	20	10	10	10	10	10	10
20	20	20	20	20	20	20	20	10
20	20	20	10	10	10	10	20	10
20	20	10	10	10	10	10	20	10
20	10	10	10	10	10	10	20	10
10	10	10	10	20	10	10	20	10
10	10	10	10	10	10	10	10	10
20	10	20	20	10	10	10	20	20
20	10	10	20	10	10	20	10	20

(a)
-1	-1	0
-1	0	1
0	1	1

(b)
0	-1	-1
1	0	-1
1	1	0

(c)
-1	-1	-1
2	2	2
-1	-1	-1

(d)
-1	2	-1
-1	2	-1
-1	2	-1

(e)
-1	-1	-1
-1	8	-1
-1	-1	-1

(f)
1	1	1
1	1	1
1	1	1

(g)
-1	0	1
-1	0	1
-1	0	1

(h)
0	-1	0
-1	4	-1
0	-1	0

11. จงหาผลลัพธ์ที่ได้จากการทำ mask processing ด้วยมือ แบบ linear filter จากข้อมูลภาพและ mask ที่ให้มา โดยการทำแบบ correlaton ('corr') และใช้หลักการ zeros padding คือเติม 0 รอบๆ ภาพ

1	1	5	5
10	10	1	5
5	20	10	5
5	0	20	5

Image

-1	0	-1
0	4	0
-1	0	-1

Mask

12. จากข้อมูลภาพที่ให้มาในข้อก่อนหน้า จงทำ mask processing ด้วยมือแบบ nonlinear filter ด้วย filter แบบ mean filter และ max filter เมื่อพิจารณา neighborhood เป็น 3x3 โดยใช้หลักการ zeros padding คือเติม 0 รอบๆ ภาพ

13. จงอธิบาย High pass filter และ Low pass filter ตามความเข้าใจของนักศึกษา (ในคำอธิบายต้องไปเกี่ยวข้องกับการทำให้ภาพเบรอขึ้นหรือคมชัดขึ้นด้วย)

แบบฝึกหัด ชุดที่ 2

1. จงเติมคำตอบในแต่ละข้อต่อไปนี้

   (a). ในการทำ $A \cap B$ (เอาภาพ A มา intersect กับภาพ B) ของภาพขาวดำ สามารถใช้ Matlab Expression คือ .....

   (b). ในการทำ $A \cup B$ (เอาภาพ A มา union กับภาพ B) ของภาพขาวดำ สามารถใช้ Matlab Expression คือ .....

   (c). ในการทำ A^c (Complement of A) ของภาพขาวดำ สามารถใช้ Matlab Expression คือ .....

   (d). ในการทำ $A - B$ (The difference of A and B) ของภาพขาวดำ สามารถใช้ Matlab Expression คือ .....

   (e). ในการหาจุด หรือหาเส้นตรง ในภาพนั้นจัดอยู่หลักการของ Image Segmentation ประเภท .....

   (f). ในการหาขอบของภาพนั้นจัดอยู่หลักการของ Image Segmentation ประเภท .....

   (g). การเติบโตของบริเวณ (Region Growing) นั้นจัดอยู่หลักการของ Image Segmentation ประเภท .....

   (h). การแบ่งแยกและการรวมกันของบริเวณ (Region Splitting and Merging) นั้นจัดอยู่หลักการของ Image Segmentation ประเภท .....

   (i). Model สีที่ใช้กับจอ computer แบบ CRT คือ Model สีแบบ .....

   (j). Model สีที่ใช้กับงานพิมพ์ (printing) คือ Model สีแบบ .....

2. จงตอบคำถามในแต่ละข้อต่อไปนี้

   (a). ถ้ามีภาพลายเส้นซึ่งเรารู้สึกว่าลายเส้นต่างๆ นั้นหนาเกินไป ถ้าต้องการเส้นบางลง เราสามารถใช้หลักการใดในเรื่อง image morphology มาช่วยได้

   (b). การที่เราประมวลผลภาพด้วยการเริ่มต้นทำ image dilation ก่อนแล้วตามด้วย image erosion โดยการใช้ structure element เดิม ถือว่าเป็นการทำ operation ใดในเรื่อง image morphology

(c). การที่เราประมวลผลภาพด้วยการเริ่มต้นทำ image erosion ก่อนแล้วตามด้วย image dilation โดยการใช้ structure element เดิม ถือว่าเป็นการทำ operation ใดในเรื่อง image morphology

(d). จงเขียน Matlab code ในการทำ image opening กับภาพ A ด้วย structure elements B1 โดยไม่ใช้ function imopen โดยตรง แต่สามารถเลือกใช้ functions หรือคำสั่งอื่นๆ ที่เกี่ยวข้องได้ไม่จำกัด

(e). จงเขียน Matlab code ในการทำ image closing กับภาพ A ด้วย structure elements B1 โดยไม่ใช้ function imclose โดยตรง แต่สามารถเลือกใช้ functions หรือคำสั่งอื่นๆ ที่เกี่ยวข้องได้ไม่จำกัด

(f). จงเขียน Matlab code ในการทำ hit-or-miss transformation กับภาพ A ด้วย structure elements B1 และ B2 โดยไม่ใช้ function bwhitmiss โดยตรง แต่สามารถเลือกใช้ functions หรือคำสั่งอื่นๆ ที่เกี่ยวข้องได้ไม่จำกัด

3. จงตอบคำถามในแต่ละข้อต่อไปนี้

(a). ถ้าต้องการหาเส้นตรงที่อยู่ในแนว -45 องศา ในภาพควรใช้ mask ขนาด 3X3 ที่มีรูปแบบอย่างไร

(b). ถ้าต้องการหาเส้นตรงที่อยู่ในแนวตั้ง ในภาพควรใช้ mask ขนาด 3X3 ที่มีรูปแบบอย่างไร

(c). ในการหาขอบภาพด้วยการใช้เกรเดียนต์โอเปอเรเตอร์ (gradient operator) นั้นใช้การพิจารณาค่าใดของระดับความเทาของภาพเพื่อการตัดสินว่า pixel ใดจะเป็นขอบของภาพ

(d). ในการหาขอบภาพด้วยการใช้ลาปลาเซียนโอเปอร์เรเตอร์ (laplacian operator) นั้นใช้การพิจารณาค่าใดของระดับความเทาของภาพเพื่อการตัดสินว่า pixel ใดจะเป็นขอบของภาพ

(e). วิธีการหาค่าแบ่ง (Threshold) แบบใดที่มีหลักการของความน่าจะเป็นเข้ามาเกี่ยวข้อง

(f). ยกตัวอย่างฟังก์ชันที่นิยมนำมาใช้ในการประมาณฟังก์ชันความหนาแน่นความน่าจะเป็นในการทำ Optimal Threshold มา 2 ฟังก์ชัน

(g). $P(R_j)$ ซึ่งกำหนดว่าเป็นเพรดิเคตทางตรรกะ (Logical Predicate) ที่ใช้ในเรื่อง Region-based segmentation นั้นคืออะไร

(h). การทำ Region Growing และ Region Splitting and Merging ในแต่ละ Application นั้น นักศึกษาคิดว่าการที่ผลลัพธ์ที่ได้จะออกมาดีหรือไม่ดีขึ้นอยู่กับอะไรเป็นประเด็นที่สำคัญที่สุด

4. จงเขียนขั้นตอนวิธี (algorithm) ในการทำ Global Thresholding ด้วยวิธีการค้นหาค่าที่เหมาะสม (Heuristic Approach) ซึ่งขั้นตอนวิธีดังกล่าวอาจเขียนอธิบายเป็นลำดับขั้นตอนตามความเข้าใจ หรือ เขียนโดยใช้ pseudo code หรือ เขียนด้วย Matlab code ก็ได้

5. จงเขียนขั้นตอนวิธี (algorithm) ในการทำ Local Thresholding ซึ่งขั้นตอนวิธีดังกล่าวอาจเขียนอธิบายเป็นลำดับขั้นตอนตามความเข้าใจ หรือ เขียนโดยใช้ pseudo code หรือ เขียนด้วย Matlab code ก็ได้

6. จงใช้วิธีการของ Hough Transform ในการ detect เส้นตรงที่ dominate ที่สุด (เด่นที่สุด) 2 เส้นแรกจากข้อมูลภาพที่ให้มาด้านล่าง โดยแสดงเฉพาะเมื่อค่ามุมเป็น -45 องศา, 0 องศา, 45 องศา, และ 90 องศา (ให้แสดงวิธีทำโดยละเอียดแบบที่เคยทำในชั้นเรียน)

y \ x	0	1	2	3	4	5	6
0	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
2	0	0	1	0	1	0	0
3	0	0	0	0	0	0	1
4	0	1	0	1	1	0	0
5	1	0	0	0	0	1	0
6	0	0	0	1	0	0	1

7. จงใช้วิธีการของ Hough Transform ในการ detect เส้นตรงที่ dominate ที่สุด (เด่นที่สุด) 2 เส้นแรกจากข้อมูลภาพที่ให้มาด้านล่าง โดยแสดงเฉพาะเมื่อค่ามุมเป็น -30 องศา, 0 องศา, 60 องศา, และ 90 องศา (ให้แสดงวิธีทำโดยละเอียดแบบที่เคยทำในชั้นเรียน)

y \ x	0	1	2	3	4	5	6
0	0	0	1	1	0	0	0
1	0	1	1	1	0	0	0
2	0	1	1	0	0	0	0
3	0	1	0	0	0	1	0
4	0	1	0	1	0	0	0
5	1	1	0	0	1	0	0
6	1	0	0	0	0	0	0

8. จงตอบคำถามในแต่ละข้อต่อไปนี้

   (a). สีหลักแบบบวก (Additive primary colors) และสีหลักแบบลบใช้หลักการในการผสมสีที่แตกต่างกันอย่างไร และในแต่ละแบบเกี่ยวข้องกับ model สีแบบใด

   (b). ยกตัวอย่าง model สีที่รู้จักมา 2 models พร้อมอธิบายคุณลักษณะของแต่ละ model อย่างคร่าวๆ

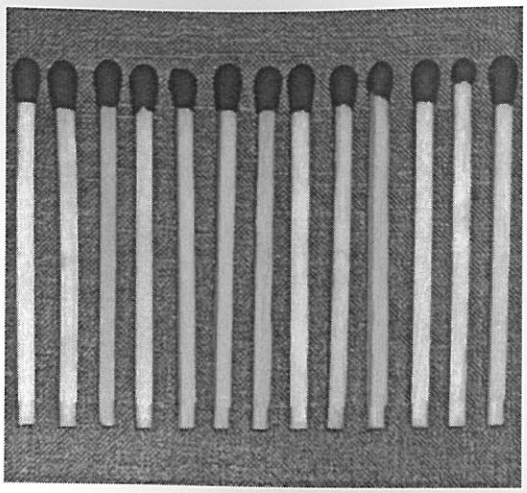
   (c). Pseudo color image processing คืออะไร และสามารถเอาไปใช้ในงานลักษณะอย่างไรบ้าง

   (d). ยกตัวอย่างงานที่มีการนำหลักการของ Pseudo color ไปใช้มาสองงาน

   (e). ในการทำ full-color image processing กับภาพสีโดยการพิจารณาแต่ละ component ของ model สี มีความเหมือนหรือแตกต่างจากการประมวลผลภาพแบบ gray scale อย่างไร จงอธิบาย

   (f). ถ้าต้องการนำภาพสีที่เรามีซึ่งเป็น model สีแบบ RGB ไปทำ histogram equalization จะต้องมีการจัดการอย่างไร

9. จากภาพที่ให้มาด้านล่าง ถ้าต้องการนับออกมาว่ามีก้อนไม่ขีดทั้งหมดกี่ก้าน นักศึกษาสามารถนำเอาวิธีการหรือหลักการใดในความรู้เรื่อง digital image processing มาใช้ในแต่ละขั้นตอนของการประมวลผลบ้างเพื่อคาดหวังว่าจะได้มาซึ่งข้อมูลที่เป็นจำนวนก้านดังกล่าว จงอธิบายพร้อมบอกด้วยว่าวิธีการหรือหลักการที่นำมาใช้นั้นเพื่อทำอะไรและคาดหวังว่าจะได้ผลลัพธ์ในแต่ละขั้นตอนอย่างไร



10. จากภาพที่ให้มาด้านล่าง ถ้าต้องการหาตำแหน่งของปุ่มกดออกมาทั้ง 12 ปุ่ม นักศึกษาสามารถนำเอาวิธีการหรือหลักการใดในความรู้เรื่อง digital image processing มาใช้ในแต่ละขั้นตอนของการประมวลผลบ้างเพื่อคาดหวังว่าจะได้ตำแหน่งของปุ่มดังกล่าว จงอธิบายพร้อมบอกด้วยว่าวิธีการหรือหลักการที่นำมาใช้นั้นเพื่อทำอะไรและคาดหวังว่าจะได้ผลลัพธ์ในแต่ละขั้นตอนอย่างไร