การแปลงใบหน้าคน

Real Face Transformation

นางสาวบุรินทรา	กณะโกมล	รหัส 47360136
นางสาวพิชชาภรณ์	บ่อน้อย	รหัส 47362058
นางสาววิชชุลดา	เสนานุช	รหัส 47362165

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2550

หัวข้อโครงงาน	การแปลงใบหน้าคน		
ผู้ดำเนินโครงงาน	นางสาวบุรินทรา	กณะโกมล	รหัส 47360136
	นางสาวพิชชาภรณ์	บ่อน้อย	รหัส 47362058
	นางสาววิชชุลดา	เสนานุช	รหัส 47362165
อาจารย์ที่ปรึกษา	ดร.พนมขวัญ	ริยะมงคล	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2550		

บทคัดย่อ

โครงงานนี้เป็นการศึกษาและพัฒนาโปรแกรมสำหรับการแปลงใบหน้าคนจากภาพนิ่ง 2 มิติ โดยใช้ทฤษฎีทางด้านการประมวลผลภาพ กล่าวคือ ใช้กระบวนการของสีและเทรสโฮลดิงเข้ามาใช้ในการตรวจหาใบหน้าจากรูปภาพสี 2 มิติ จากนั้นจะได้เทมเพลทที่ทับส่วนที่เป็นใบหน้า ซึ่งจะได้นำใบหน้านี้ไปทำการแปลงให้เปลี่ยนรูปหน้าเป็นลักษณะอื่นๆ โดยใช้ทฤษฎีการแปลงภาพ (Image Transformation) โดยโครงงานนี้ใช้โปรแกรม MATLAB ในการพัฒนา

ผลที่ได้รับจากโครงงานนี้ คือ ได้โปรแกรมที่สามารถจับเทมเพลทส่วนของใบหน้าจากรูปภาพสี 2 มิติ สามารถแยกใบหน้าคนออกจากส่วนประกอบอื่นที่อยู่ในภาพได้ สามารถนำรูปใบหน้าที่ได้ไปแปลงให้เปลี่ยนรูปหน้าเป็นลักษณะอื่นๆได้ อีกทั้งยังสามารถนำไปศึกษาและพัฒนาโปรแกรม เพื่อนำไปประยุกต์ใช้งานในด้านต่อๆ ไปได้อีก

Project Title	REAL FACE TRANSFORMATION		
Name	Miss Burintra	Kanakomoln	ID 47360136
	Miss Pitchaporn	Bornoi	ID 47362058
	Miss Witchulada	Senanuch	ID 47362165
Project Advisor	Panomkhawn	Riyamongkol, Ph.D.	
Major	Computer Engineering		
Department	Electrical and Computer Engineering		
Academic Year	2007		

ABSTRACT

This project is to study and develop the Real Human Face 2D Transformation program. Color image processing and thresholding techniques are used capture a face from a 2D color image. Then, we will get a template which is corresponding to the location of the face in the image. This template can be transformed to be another image by image transformation techniques. All above processes have been develop by MATLAB program.

The result of this project is a program that can get a face-template from a 2D image and transform a face-template. Also, this program can be studied and developed to used in other applications.

กิตติกรรมประกาศ

จากที่ข้าพเจ้าได้ศึกษาและปฏิบัติงานจากโครงงานวิศวกรรมนี้ ส่งผลให้ข้าพเจ้าได้รับความรู้ที่เกี่ยวกับ ทฤษฎีการประมวลผลภาพ(Digital Image Processing) ทฤษฎีการแปลงภาพ (Image Transformation) และการใช้งานโปรแกรม MATLAB การทำงานร่วมกันในกลุ่มนั้น ได้ฝึกความอดทน ความเอื้อเฟื้อเผื่อแผ่ ช่วยเหลือซึ่งกันและกันแก้ไขปัญหาต่างๆ ให้สำเร็จลุล่วงไปได้ด้วยดี และถ้าไม่มีผู้ที่คอยช่วยเหลือและให้คำปรึกษาที่ดีแล้ว โครงงานวิศวกรรมนี้ก็ไม่สามารถประสบความสำเร็จไปได้ ดังนั้นข้าพเจ้าขอกราบขอบพระคุณ ดร.พนมขวัญ ริยะมงคล อาจารย์ที่ปรึกษาโครงงานวิศวกรรมนี้เป็นอย่างยิ่งที่ได้ให้คำแนะนำที่ดีและมีประโยชน์ต่อข้าพเจ้าและโครงงานวิศวกรรมนี้

ขอขอบพระคุณ คุณพ่อ คุณแม่ พี่และน้องของข้าพเจ้าที่คอยอยู่เคียงข้างและเป็นกำลังใจที่ดีในการทำงานของข้าพเจ้า ขอบคุณเพื่อนๆ และน้องรหัสที่คอยสอบถาม ให้กำลังใจและให้คำปรึกษาในบางโอกาส และขอบคุณบุคคลในภาพทุกท่าน ที่ให้ความอนุเคราะห์ให้ข้าพเจ้านำภาพของท่านมาใช้ในโครงงานวิศวกรรมนี้

สุดท้ายข้าพเจ้าใคร่ขอขอบคุณ ผู้ที่มีส่วนเกี่ยวข้องทุกๆ ท่านที่ไม่ได้เอ่ยนาม ณ ที่นี้ ที่ท่านได้มีส่วนร่วมในการให้คำปรึกษา ข้อมูลและมีส่วนช่วยให้โครงงานวิศวกรรมนี้ประสบความสำเร็จลุล่วงไปได้ด้วยดี

นางสาวบุรินทรา กณะโมมล
นางสาวพิชชาภรณ์ บ่อน้อย
นางสาววิชชุลดา เสนานุช

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

เนื่องจากในปัจจุบันนี้ มีการนำรูปภาพมาดัดแปลงเพื่อนำมาประยุกต์ใช้งานในด้านต่างๆ อย่างแพร่หลาย ไม่ว่าจะเป็นในด้านโฆษณา ด้านบันเทิง หรือทางด้านอินเตอร์เน็ต ฯลฯ ซึ่งในด้านอินเตอร์เน็ตนั้น มีการใช้งานในการติดต่อสื่อสารกันอย่างแพร่หลาย โดยที่แต่ละคนนั้นก็จะสรรหาลูกเล่นและฟังก์ชั่นต่างๆ เพื่อมาใช้เป็นตัวแทนของตัวเองในการใช้งาน เช่น โปรแกรม MSN จะมีส่วนที่ให้แสดงรูปภาพ display ซึ่งเราสามารถเลือกรูปภาพต่างๆตามความพอใจของผู้ใช้ และในตอนนี้ก็ได้มีโปรแกรมมาช่วยในการสร้างรูปภาพ display เป็นรูปการ์ตูนที่เหมือนตัวเรา โดยการเลือกส่วนประกอบต่างๆบนใบหน้าและอื่นๆมาประกอบกันจนได้เป็นรูปการ์ตูนเหมือนตัวเรา จึงได้นำความคิดจากโปรแกรมดังกล่าวมาพัฒนาทำเป็นโครงงานนี้ ซึ่งโปรแกรมที่พัฒนาขึ้นมานี้จะให้ผลลัพธ์เป็นภาพถ่ายสีรูปคนหน้าล้อเลียน 2 มิติ

องค์ประกอบของภาพหน้าคนนั้น เมื่อนำมาวิเคราะห์และแยกองค์ประกอบ จะได้ส่วนประกอบของใบหน้าและโครงสร้างของหน้าคนที่แตกต่างกันออกไป เช่น รูปหน้า คิ้ว จมูก ปาก หนวด ฯลฯ โดยนำความสัมพันธ์ของโครงสร้างลักษณะใบหน้าและอัลกอริทึมที่เกี่ยวข้องมาใช้ในการทำโครงงานนี้

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อให้ได้ภาพถ่ายสีรูปคนหน้าล้อเลียนซึ่งแปลงมาจากภาพถ่ายสีรูปคน 1 คน โดยได้จากโปรแกรมที่พัฒนาขึ้นมา

1.2.2 เพื่อสามารถนำภาพที่ได้มาประยุกต์ใช้ในงานด้านต่างๆ เช่น เพื่อใช้เป็นรูปภาพ display ในโปรแกรม MSN เป็นต้น

1.2.3 เพื่อศึกษาทฤษฎีต่างๆและอัลกอริทึมที่ใช้ในโครงงานนี้แล้วนำไปประยุกต์ใช้ในโครงงาน

1.2.4 เพื่อศึกษาวิธีการใช้และการประยุกต์ใช้งานโปรแกรม MATLAB

1.2.5 เพื่อเป็นแนวทางในการพัฒนาโครงงานการแปลงใบหน้าคน ซึ่งอาจนำไปใช้งานในด้านต่างๆ ต่อไป

1.3 ขอบข่ายของโครงงาน

1.3.1 พัฒนาโปรแกรมการแปลงหน้าจริงเป็นหน้าล้อเลียนที่ต่างจากเดิม จากภาพถ่ายสีรูปคน 1 คน ให้เป็นภาพถ่ายสีรูปคนหน้าล้อเลียน 2 มิติ

1.3.2 ศึกษาทฤษฎีและออกแบบอัลกอริทึมที่นำมาใช้ในโครงงานนี้

1.3.3 ศึกษาการใช้งานและการประยุกต์ใช้งานโปรแกรม MATLAB

1.4 แผนการดำเนินงาน

ตารางที่ 1.1 ขั้นตอนการดำเนินงาน

กิจกรรม	ระยะเวลาการดำเนินงาน								
	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.
1.รวบรวมข้อมูล	←	→							
2.ศึกษาทฤษฎีและโปรแกรม		←	→	→	→				
3.ออกแบบอัลกอริทึม			←	→	→	→			
4.พัฒนาโปรแกรมการแปลงหน้าจริงเป็นหน้าการ์ตูน				←	→	→	→		
5.ทดสอบและปรับปรุงระบบ						←	→	→	
6.วิเคราะห์และสรุปผล								←	→

1.5 ผลที่คาดว่าจะได้รับ

1.5.1 สามารถสร้างและพัฒนาอัลกอริทึมสำหรับการแปลงหน้าคนจริงเป็นหน้าคนล้อเลียนให้มีประสิทธิภาพ และทำภาพให้ออกมาให้ต่างจากภาพเดิมได้

1.5.2 ได้รับความรู้เกี่ยวกับทฤษฎีพื้นฐานและอัลกอริทึมของการแปลงภาพ

1.5.3 ได้รับความรู้และวิธีการใช้งานโปรแกรม MATLAB

1.5.4 สามารถนำทฤษฎีและอัลกอริทึมที่ใช้ในโครงงานนี้ไปประยุกต์ใช้ในโครงงานอื่นๆ ได้

1.5.5 ผู้ที่สนใจสามารถนำไปศึกษาและพัฒนาไปใช้ในงานด้านๆต่อไปได้

1.6 งบประมาณที่ใช้

1.6.1 ค่าวัสดุสำนักงาน	เป็นจำนวนเงิน	1,000	บาท
1.6.2 อุปกรณ์คอมพิวเตอร์	เป็นจำนวนเงิน	1,000	บาท
1.6.3 ค่าถ่ายเอกสารและปริ้นท์งาน	เป็นจำนวนเงิน	1,000	บาท
	รวมเป็นเงินทั้งสิ้น	3,000	บาท

หมายเหตุ : ถัวเฉลี่ยทุกรายการ

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีทางด้าน Digital Image Processing

Image processing เป็นการประมวลผลสัญญาณโดยใช้ดิจิตอลคอมพิวเตอร์เพื่อบันทึกและจัดเก็บภาพ เพื่อปรับปรุงให้ดีขึ้นโดยใช้กระบวนการทางคณิตศาสตร์ เพื่อช่วยในการวิเคราะห์รูปภาพ เพื่อสังเคราะห์ภาพละเพื่อสร้างระบบการมองเห็นให้กับคอมพิวเตอร์ จะเห็นได้ว่า Image Processing มีประโยชน์และสำคัญมากที่จะใช้ในการจัดการเกี่ยวกับรูปภาพซึ่งในการทำโครงการนี้จะต้องใช้ความรู้ด้าน Image Processing ในการจัดการกับรูปภาพที่ได้จากกล้องดิจิตอลเช่นเดียวกัน การประมวลผลภาพเชิงดิจิตอล(Digital image processing) เป็นการนำคอมพิวเตอร์มาใช้ในการประมวลผลภาพในรูปดิจิตอลฟอร์แมต ดังนั้นขั้นตอนในการประมวลผลภาพที่จำเป็นนี้คือ

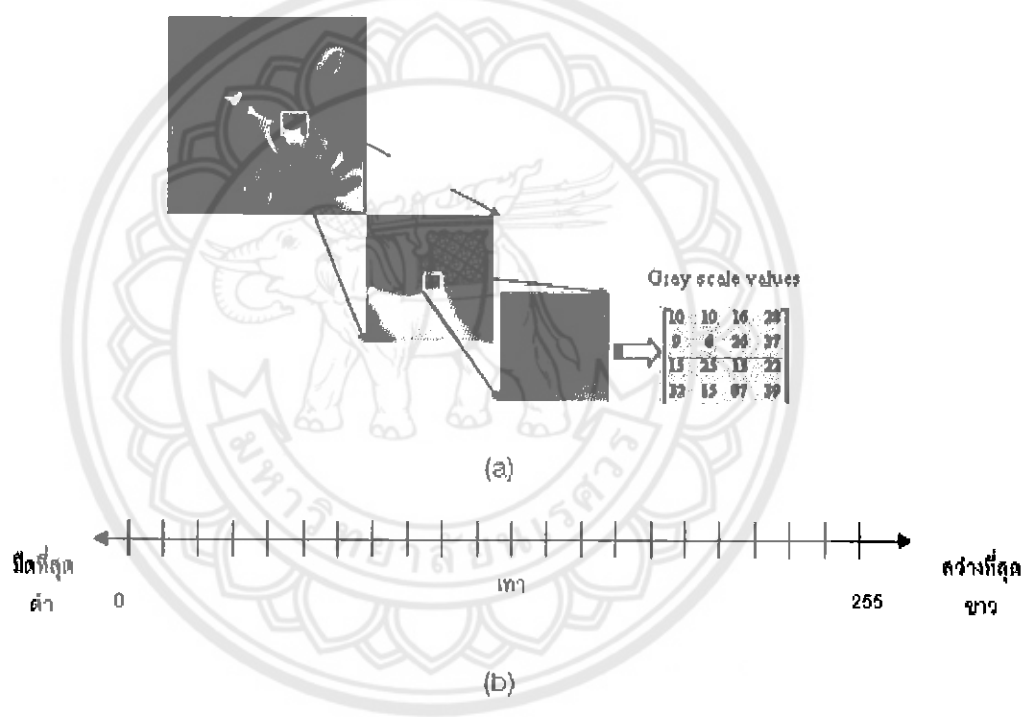
- **Image acquisition** เป็นขั้นตอนแรกสุดของการประมวลผลภาพ คือการรับภาพที่ต้องการประมวลผลเข้ามา
- **Image enhancement** เป็นการปรับปรุงภาพเพื่อให้ภาพมีคุณสมบัติที่เหมาะสมสำหรับการประมวลผลขั้นต่อไปโดยใช้กระบวนการทางคณิตศาสตร์ต่าง ๆ
- **Image restoration** เป็นกระบวนการปรับปรุงภาพออกมาให้เหมือนภาพตั้งต้น ข้อแตกต่างระหว่าง Image restoration กับ Image enhancement คือ Image restoration เป็นการทำให้ภาพเหมือนภาพเดิมมากที่สุด แต่ Image enhancement ไม่สนใจว่าภาพที่ออกมาจะเหมือนภาพเดิมหรือไม่ แต่ต้องการให้ภาพมีคุณภาพที่เหมาะสมที่จะนำไปประมวลผลในขั้นตอนถัดไป
- **Image compression** เป็นเทคนิคในการลดเนื้อที่ในการจัดเก็บรูปภาพ หรือลด Bandwidth ในการส่งข้อมูลภาพผ่านสื่อกลางต่างๆ
- **Morphological processing** เป็นขบวนการในการประมวลภาพตามรูปร่างของวัตถุในภาพ ส่วนมากเป็นการเปลี่ยนรูปร่างของวัตถุ
- **Image segmentation** เป็นกระบวนการในการแยกแยะส่วนประกอบของรูปภาพออกเป็นส่วนต่างๆเพื่อประโยชน์ในการวิเคราะห์ภาพ
- **Image representation and description** เป็นการอธิบายรูปภาพหรือแทนข้อมูลภาพด้วยข้อมูลที่สามารถสื่อความหมายได้
- **Image recognition** เป็นกระบวนการในการจดจำรูปภาพซึ่งเกี่ยวข้องกับความรู้ขั้นสูงในการตีความหมายของรูปภาพ

Digital Image เป็นแอเรย์หลายมิติของตัวเลข ได้แก่ ภาพระดับสีเทา(gray image) หรือแอเรย์หลายมิติของเวคเตอร์ ได้แก่ ภาพสี(color) แต่ละจุดบน digital image จะเรียกว่า pixel โดยที่ในแต่ละ pixel จะมีค่ากำกับเอาไว้ เรียกว่า pixel value

Digital Image สามารถแบ่งออกได้ 4 ประเภท ได้แก่

2.1.1 Intensity Image หรือ Monochrome Image หรือ Gray Image

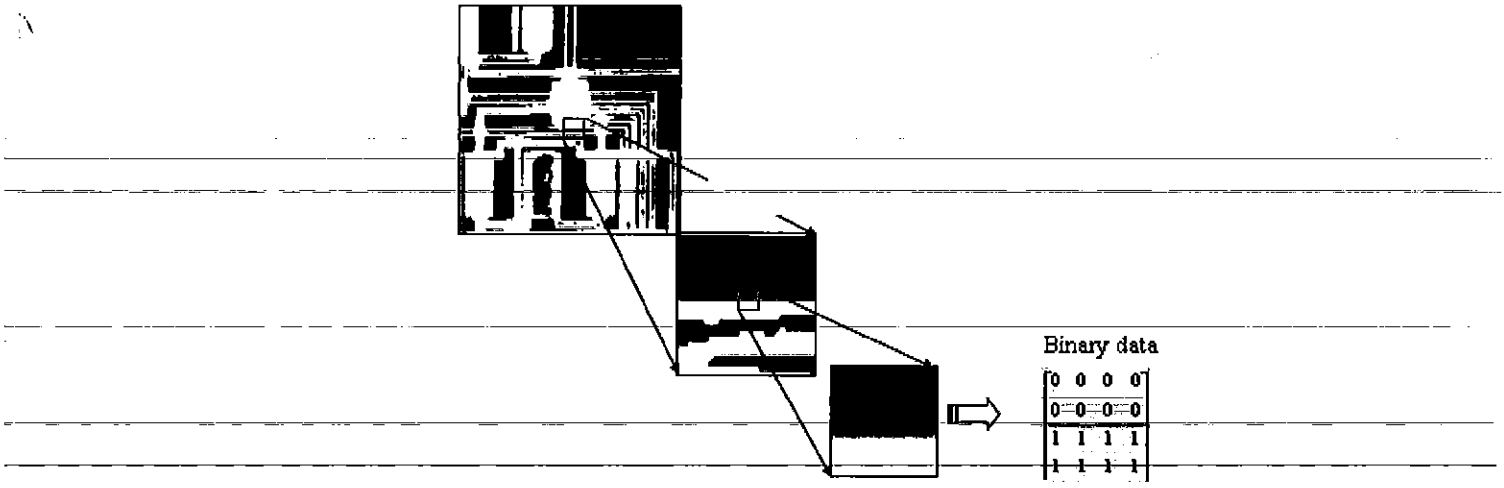
ค่าในแต่ละ pixel ของ gray image คือค่าความเข้มของแสง ณ แต่ละตำแหน่งของ pixel ซึ่งจะอยู่ในรูปของ gray scale(gray level) ดังรูปที่ 2.1 (a) ค่าที่เป็นไปได้ของ gray scale จะขึ้นอยู่กับจำนวน bit ที่ใช้ ตัวอย่างเช่น 8-bit monochrome จะมี gray scale ทั้งหมด 256 ระดับ ดังรูปที่ 2.1 (b)



รูปที่ 2.1 (a) แสดง gray image และ pixel value ของ gray image; (b) แสดง 8 bit gray scale

2.1.2 Binary Image หรือ Black and White Image

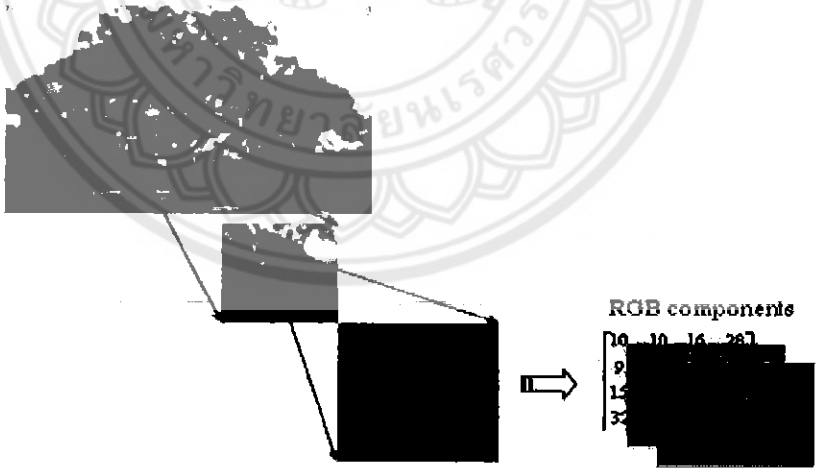
ค่าในแต่ละ pixel ของ binary image จะใช้แค่ 1 บิต ซึ่งจะมีค่าที่เป็น ไปได้คือ 0 (สีดำ) และ 1 (สีขาว) เท่านั้น ดังรูปที่ 2.2



รูปที่ 2.2 ภาพ binary และ pixel value ของภาพ binary

2.1.3 Color Image หรือ RGB Image

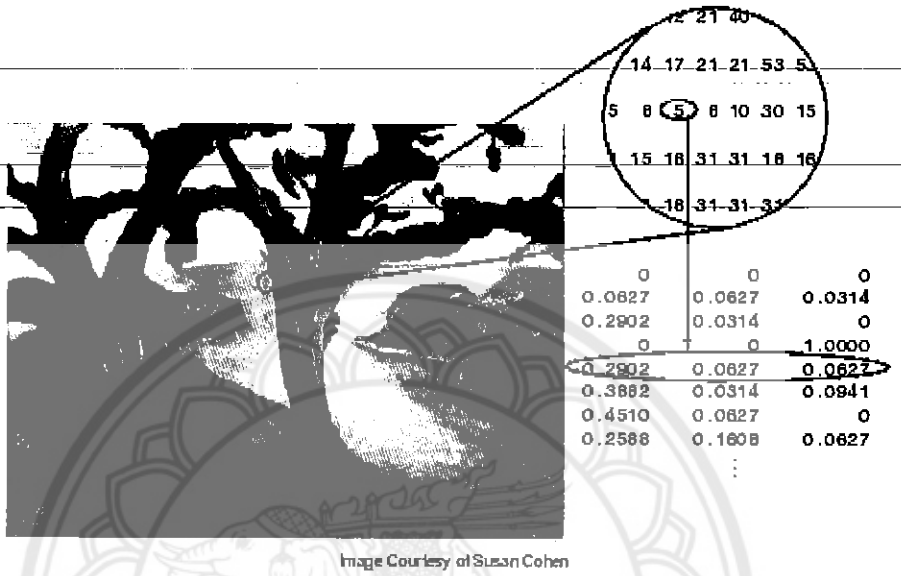
ค่าในแต่ละ pixel ของ color image จะประกอบไปด้วย vector ที่แสดงค่าของสีแดง สีเขียว และสีน้ำเงิน อย่างละ 8-bit ดังนั้น RGB image 1 pixel จะประกอบไปด้วยจำนวนบิตทั้งหมด 24 บิต ทำให้ RGB image มีจำนวนสีที่เป็นไปได้ทั้งหมด 2^{24} สีดังตัวอย่างรูปที่ 2.3



รูปที่ 2.3 ภาพ RGB และ pixel value ของภาพ RGB

2.1.4 Indexed Image

ค่าในแต่ละ pixel ของ indexed image จะประกอบไปด้วยค่าของ index number ขนาด 8 บิต ซึ่งจะชี้ไปยังค่าของสีในตารางสี ดังนั้น ถ้าเราต้องการจะทราบค่าสีในแต่ละ pixel เราจะต้องไปดูค่าในตารางสี index ตรงกับค่าใน pixel ดังรูปที่ 2.4



รูปที่ 2.4 ภาพ indexed image และ pixel value ของ indexed image

2.2 Image Transformation

การแปลงข้อมูลภาพพื้นฐานแบ่งออกเป็น 4 วิธีคือ

- การเลื่อนภาพ (Translation)
- การหมุนภาพ (Rotation)
- การย่อภาพ (Scaling)
- การบิดภาพ (Shearing)

2.2.1 การเลื่อนภาพ (Translation)

เป็นการเลื่อนตำแหน่งของภาพตามระยะการขจัดทางแนวแกน x และตามแนวแกน y เมื่อกำหนดให้พิกัดเดิม คือ (x, y) และพิกัดใหม่คือ (x', y') จะได้สมการของการเลื่อนภาพ

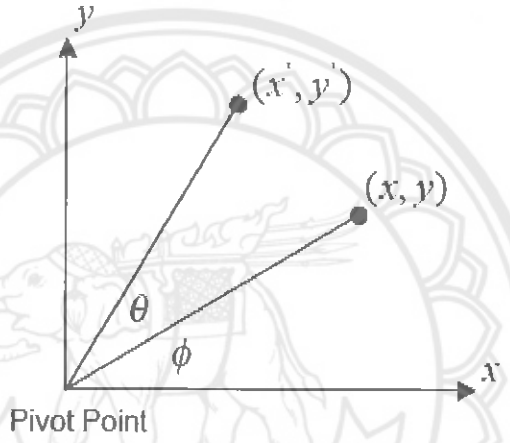
$$x' = x + Tx$$
$$y' = y + Ty \qquad (2.1)$$

ซึ่งสามารถเขียนให้อยู่ในรูปแบบของเมตริกได้ มีลักษณะดังนี้คือ $P' = P + T$ เมื่อ

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \end{bmatrix} \text{ และ } T = \begin{bmatrix} Tx \\ Ty \end{bmatrix} \tag{2.2}$$

2.2.2 การหมุนภาพ (Rotation)

เป็นการหมุนภาพในระนาบ x,y เมื่อจุดศูนย์กลางการหมุน (Pivot Point) อยู่ที่จุดเริ่มต้น (Origin) ดังรูปที่ 2.5



รูปที่ 2.5 การหมุนจุด (x,y) รอบจุดเริ่มต้น

จากรูปที่ 2.5 จะได้ว่า

$$x = r\cos(\phi)$$
$$y = r\sin(\phi) \tag{2.3}$$

และ

$$x' = r\cos(\phi + \theta) = r(\cos\phi\cos\theta - \sin\phi\sin\theta)$$
$$y' = r\sin(\phi + \theta) = r(\sin\phi\cos\theta - \cos\phi\sin\theta) \tag{2.4}$$

เพราะฉะนั้นจากสมการที่ (2.3) และ (2.4) จะได้สมการของการหมุนรอบจุด Pivot Point ดังนี้คือ

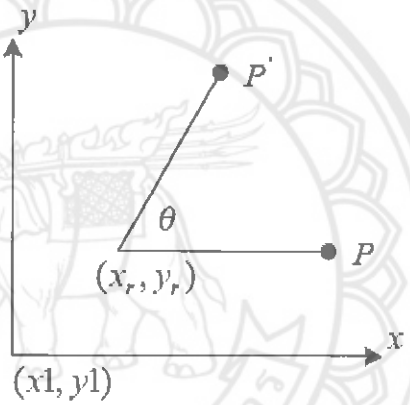
$$x' = x\cos(\theta) - y\sin\theta$$
$$y' = x\sin(\theta) - y\cos\theta \qquad (2.5)$$

ซึ่งสามารถเขียนให้อยู่ในรูปแบบของเมตริกได้ มีลักษณะดังนี้คือ $P' = R.P$ เมื่อ

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \end{bmatrix} \text{ และ } R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \qquad (2.6)$$

การหมุนภาพเมื่อจุดศูนย์กลางการหมุนไม่ได้อยู่ที่จุดเริ่มต้น

พิจารณาเมื่อจุดหมุนอยู่ที่ตำแหน่ง (x_r,y_r)



รูปที่ 2.6 การหมุนจุด P รอบจุด (x_r,y_r)

วิธีการในการหมุนภาพเมื่อจุดหมุนไม่ได้อยู่ที่จุด Origin สามารถทำได้ดังนี้คือ

1. ทำการเปลี่ยนจุด Pivot ไปยังจุด Origin

$$x1 = x - x_r$$
$$y1 = y - y_r \qquad (2.7)$$

2. ทำการหมุนรอบจุด Origin
3. ย้ายกลับไปยังจุดเดิม โดยการบวกด้วย x_r และ y_r
4. สมการการหมุนรอบจุด Pivot ใด ๆ ที่ไม่ใช้จุด Origin

$$x' = (x - x_r)\cos\theta - (y - y_r)\sin\theta + x_r$$
$$y' = (x - x_r)\sin\theta + (y - y_r)\cos\theta + y_r \qquad (2.8)$$

2.2.3 การย่อภาพและขยายภาพ (Scaling)

การย่อและการขยายภาพสามารถทำได้โดยการใช้ Scaling factor ได้แก่ Sx และ Sy ซึ่งใช้สำหรับการย่อและการขยายภาพในทางแกน x และ y ตามลำดับ

$0 < S_x, S_y < 1$	แสดงว่าเป็นการย่อภาพ
$S_x, S_y > 1$	แสดงว่าเป็นการขยายภาพ
$S_x = S_r$	แสดงว่าย่อและขยายจะเป็นไปตามสัดส่วน
$S_x \neq S_r$	แสดงว่าย่อและขยายจะไม่เป็นอัตราส่วน

สมการของการ Scaling จะมีลักษณะดังนี้

$$x' = x.S_x$$
$$y' = y.S_y \qquad (2.9)$$

ดังนั้นย่อและขยายภาพโดยใช้เมตริกจะมีลักษณะดังนี้คือ $P' = S.P$ เมื่อ

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \end{bmatrix} \text{ และ } S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \qquad (2.10)$$

การย่อและขยายภาพเมื่อจุด Fixed ไม่ได้อยู่ที่จุด Origin

วิธีการในการย่อและขยายภาพเมื่อจุด Fixed ของการย่อและขยายไม่ได้อยู่ที่จุด Origin สามารถทำได้ดังนี้คือ

1. ให้ย้ายตำแหน่งไปยังจุด Origin
2. ทำการย่อและขยายรอบจุด Origin
3. ย้ายไปยังจุด Fixed Point เหมือนเดิม

ซึ่งจะได้สมการของย่อและขยายภาพดังนี้คือ

$$x' = (x - x_f)S_x + x_f$$
$$y' = (y - y_f)S_x + y_f \qquad (2.11)$$

จะแปลงได้เป็นดังนี้คือ

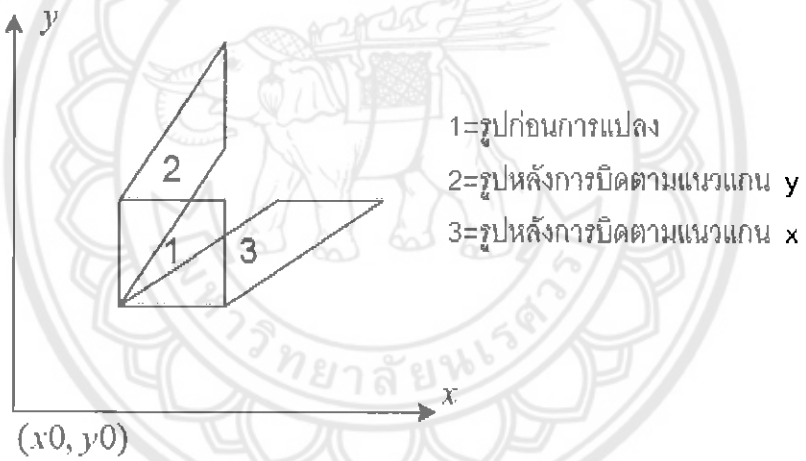
$$x' = xS_x + x_f(1 - S_x)$$
$$y' = yS_y + y_f(1 - S_y) \quad (2.12)$$

ดังนั้นการย่อและขยายภาพโดยใช้เมตริกจะมีลักษณะดังนี้คือ

$$P' = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_f(1 - S_x) \\ y_f(1 - S_y) \end{bmatrix} \quad (2.13)$$

2.2.4 การบิดภาพ (Shearing)

การบิดภาพสามารถบิดภาพได้ทั้งแนวแกน x และ y ดังรูปที่ 2.7



รูปที่ 2.7 ตัวอย่างการบิดภาพตามแนวแกน x และแกน y

โดยที่การบิดภาพตามแนวแกน x จะมีสมการดังนี้คือ

$$y' = y, x' = x + y.shX \quad (2.14)$$

และการบิดภาพตามแนวแกน y จะมีสมการดังนี้

$$x' = x, y' = y + x.shY \quad (2.15)$$

2.3 Image Thresholding

เป็นวิธีการแยกภาพในสิ่งที่สนใจ (Object) ออกจากพื้นหลัง (Background) ด้วยการเปลี่ยนคุณสมบัติของภาพจากภาพสีเทา(Gray Image) ไปเป็นภาพขาวดำ (Binary Image or Monochrome Image) ภาพขาวดำซึ่งแสดงระดับสีแต่ละระดับเป็นค่าสีเทา (gray value) โดยพื้นหน้าที่เป็นวัตถุที่เราสนใจเป็นสีดำที่มีค่าเท่ากับ 1 และส่วนที่เป็นพื้นหลังเป็นสีขาวมีค่าเท่ากับ 0

การกำหนดค่าสเกลสีเทาอ้างอิง (threshold) โดยสามารถกำหนดช่วงของค่าสเกลสีเทาได้ ซึ่งสามารถกำหนดให้ข้อมูลจุดภาพที่มีสเกลสีเทาในช่วง 0 ถึง 100 เป็นภาพสีดำ อาร์จีบี (0, 0, 0) จุดภาพที่มีสเกลสีเทาในช่วง 101 ถึง 255 เป็นภาพสีขาว อาร์จีบี (255, 255, 255) ผลจากกระบวนการตัดภาพทำให้สามารถแยกวัตถุและพื้นหลังได้อย่างอัตโนมัติ เพื่อลดความผิดพลาดจากการปรับค่าสเกลสีเทาด้วยมือ (Manual adjustment)

ขั้นตอนการประมวลผลภาพ ขั้นแรกจะเป็นการแปลงภาพสีเป็นภาพ Grayscale (256 สี) จากนั้นแปลงภาพให้เป็น Binary ด้วยวิธีการเลือกค่า Threshold ซึ่งจะเหลือข้อมูลภาพเพียงสองระดับ คือ 0 และ 1 โดย 0 จะเป็นพื้นหลัง (สีขาว) และ1 จะเป็นวัตถุในภาพ (สีดำ) จากนั้นทำการลดสัญญาณรบกวนหรือ Noise ที่เกิดขึ้น

Thresholding Technique คือการพิจารณาจุด pixel ในภาพว่าจุดใดควรจะเป็นจุดขาว หรือจุดใดควรจะเป็นจุดที่มีค่าเท่ากับ 1 โดยจะทำการเปรียบเทียบค่าของแต่ละ pixel (f(x, y)) กับค่าคงที่ที่เรียกว่า Threshold (Threshold Value) เทคนิคนี้นิยมใช้กันมากในกรณีที่ความแตกต่างระหว่างวัตถุ (Object) และพื้นหลัง (Background) ซึ่งเป็นลักษณะเดียวกับสีขาวของป้ายทะเบียนและสีดำของตัวอักษรบนป้ายทะเบียนรถยนต์ ค่า pixel ในภาพที่มีค่าน้อยกว่าค่า Threshold จะถูกกำหนดเป็น 1 (จุดดำ) และถ้าค่าของ pixel ใด ๆ ในภาพมีค่ามากกว่าหรือเท่ากับค่า Threshold จะถูกกำหนดให้เป็น 0 (จุดขาว)

ในการทำภาพ Binary โดยการทำ Thresholding ให้ได้ภาพดีและคมชัด ต้องเกิดจากการเลือกค่า Threshold ที่ถูกต้องและเหมาะสม ถ้าเลือกค่า Threshold ไม่เหมาะสม เช่น ค่า Threshold ที่มากหรือน้อยจนเกินไป ภาพที่ได้จะขาดความคมชัดหรืออาจทำให้รายละเอียดของภาพขาดหายไป หรือภาพที่ได้อาจจะมืดเกินไป หรือสว่างเกินไป หรืออาจจะเป็นภาพที่มีสิ่งรบกวน (Noise) เกิดขึ้น ทำให้ภาพผลลัพท์ที่ได้ไม่ชัดเจน

Thresholding สามารถใช้แปลงภาพ Gray Scale (มีระดับความเข้ม 256 ระดับ) ไปเป็น ภาพ Binary Image (มีระดับความเข้ม 2 ระดับ) ได้ วิธีการทำ Thresholding นี้ จะทำการ เลือกค่าความเข้มมา 1 ค่า (ค่า Thresholding Level) จากค่าความเข้มของภาพ Gray Scale (0 - 255) ซึ่งค่าความเข้มที่เลือก 16

ถ้า $f(x, y) \leq T$ แล้ว $T(x, y)$ = สีดำ(0) (Background)

ถ้า $f(x, y) \geq T$ แล้ว $T(x, y)$ = สีขาว(255) (Object)

มานี้จะถูกนำมาใช้เป็นจุดแบ่ง คือ จากแต่ละจุดของพิกเซลในภาพ ถ้าจุดใดมีค่าความเข้มน้อยกว่าค่า Thresholding Level จุดนั้นจะถูกกำหนดให้มีค่าความเข้มเป็น 0 (สีดำ) ถ้าจุดใดมีค่าความเข้มมากกว่า ค่า Thresholding Level จุดนั้นจะถูกกำหนดให้มีค่าความเข้มเป็น 255 (สีขาว) ซึ่งผลลัพธ์ที่ได้ก็คือภาพ binary image นั่นเอง

ถ้ากำหนดให้

$f(x, y)$ เป็นระดับความเข้มของภาพ Gray Scale ที่จุด x, y นั้นๆ

$T(x, y)$ เป็นผลลัพธ์หลังจากการทำ Threshold ของจุด x, y นั้นๆ

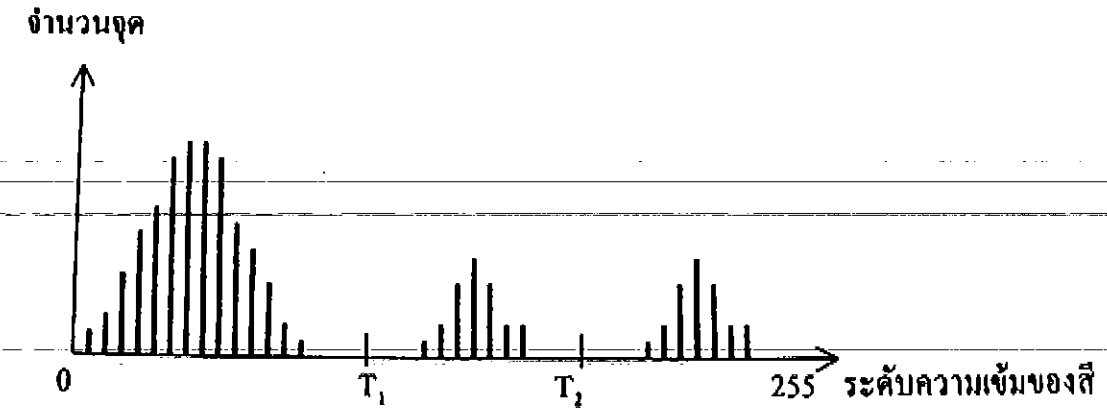
T เป็นค่า Thresholding Level ที่ใช้แบ่งภาพ

นอกจากนี้เรายังสามารถใช้หลักการเดียวกันในการแบ่งภาพออกเป็น n ระดับได้ ซึ่งวิธีแบ่งภาพเป็น n ระดับนี้เรียกว่า Multilevel Thresholding โดยจะต้องใช้ ค่า Thresholding Level n - 1 ค่า อย่างไรก็ตามแม้วิธี Multilevel Thresholding จะแบ่งภาพได้หลายระดับตามต้องการ แต่การเลือกค่า Thresholding Level ที่ดีนั้นนับเป็นเรื่องยาก

2.3.1 Adaptive Thresholding

โดยปกติการกำหนดค่า Threshold มีได้หลายค่าสำหรับทำ Intensity Thresholding มักจะทำด้วยมือซึ่งจะทำให้มีโอกาสที่ได้ค่าไม่เหมาะสมมาก Adaptive Threshold เป็นวิธีหนึ่งที่ช่วยแก้ปัญหาการเลือกปรับค่า Threshold ได้มีขั้นตอนดังนี้

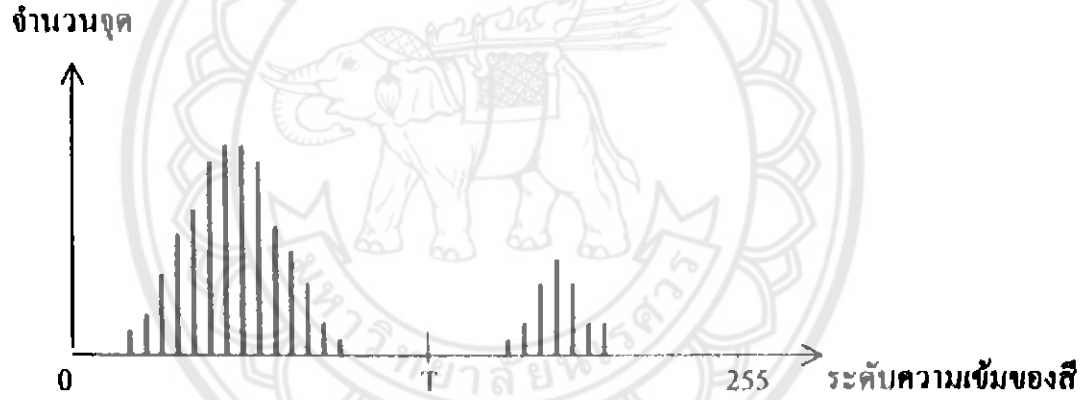
1. กำหนดค่า Intensity เริ่มต้นที่จะทำการ Threshold (T) โดยวิธีสุ่ม
2. หาค่าเฉลี่ยของพิกเซลที่มีค่าความเข้มของแสงมากกว่าค่า T และน้อยกว่าค่า T จะได้ค่า M1 และM2
3. คำนวณค่า Threshold Level ค่าใหม่จาก $T = (M1 + M2) / 2$
4. ทำซ้ำขั้นตอนที่ 2 จนกว่าค่า T จะไม่เปลี่ยน จะได้ค่า Intensity ของ Adaptive Threshold คือค่า T



รูปที่ 2.8 Histogram ของภาพซึ่งมีค่า threshold ได้หลายค่า

2.3.2 Global Thresholding

เป็นการให้ค่า threshold เป็นค่าคงที่เพียงค่าเดียว



รูปที่ 2.9 Histogram ของภาพซึ่งมีค่า threshold ได้ค่าเดียว

2.3.3 Bilevel Luminance Thresholding

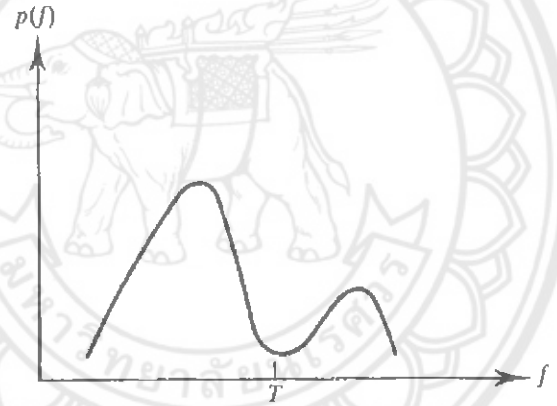
สำหรับภาพบางชนิดจะมีลักษณะวัตถุที่เราสนใจซึ่งมีความเข้มที่คงที่เมื่อเทียบกับพื้นหลัง ตัวอย่างได้แก่ ภาพของตัวอักษร (Text) เป็นต้น ซึ่งภาพเหล่านี้จะมีความเข้มของวัตถุที่เราสามารถแยกออกพื้นหลังได้อย่างชัดเจน (มีความเข้มข้นสองระดับได้แก่ความเข้มของวัตถุและความเข้มของพื้นหลัง)

การทำการ Segmentation สามารถทำได้โดยการกำหนดค่า Threshold ซึ่งเป็นค่าความเข้ม ให้มีค่าที่สามารถแยกความแตกต่างของวัตถุและพื้นหลังได้ตัวอย่างอย่างเช่น ภาพของตัวอักษรที่มีความความเข้มของตัวอักษรเป็น 0 (สีดำ) และมีความเข้มของพื้นหลังเป็น 255 (สีขาว) ดังนั้นค่า Threshold จึงควรจะมีค่าเท่ากับ 128 เพื่อที่จะให้สามารถแยกวัตถุออกจากพื้นหลังได้ โดยปกติแล้วการเลือกค่า Threshold จะขึ้นอยู่กับ Histogram ของภาพ ตามรูปที่ 8.1 แสดงการหาค่า Threshold โดยค่า Threshold ควรที่จะเลือกค่า histogram ที่อยู่ที่จุดต่ำสุดที่อยู่ระหว่างจุดสูงสุด (peaks)

$$g(x,y) = \begin{cases} 1 \_ if\,(x,y) > T \\ 0 \_ otherwise \end{cases} \qquad (2.16)$$

เมื่อ $g(x,y)$ เป็นข้อมูลภาพ ณ ตำแหน่งที่ x, y

T เป็นค่า Threshold



รูปที่ 2.10 Bimodal image histogram

2.3.5 Multilevel Luminance Thresholding

สำหรับภาพที่จะประกอบด้วยหลาย ๆ วัตถุสามารถทำการ Segmentation ได้โดยการใช้ค่า Threshold หลาย ๆ ค่า สำหรับภาพที่มี N วัตถุโดยที่แต่ละวัตถุจะมีช่วงกว้างของความเข้มเท่ากับ R_i (กำหนดได้ด้วยค่า Threshold 2 ค่าคือ T_{i-1}, T_i) สามารถทำการ Segment ได้ดังนี้

$$g(x,y) = R_i \qquad if\,(T_{i-1} <= f(x,y) <= T_i),\quad i = 1,...,N \qquad (2.17)$$

ค่า Threshold สามารถหาได้จาก histogram ของภาพ แต่ในหลาย ๆ กรณีที่การเปลี่ยนแปลงของ histogram ไม่สามารถบอกการเปลี่ยนแปลงระหว่างวัตถุได้อย่างชัดเจน วิธีการที่ง่ายที่สุดที่จะทำให้ histogram สามารถหาค่า Threshold ได้ง่ายขึ้นก็คือการใช้วิธี Edge Detection เพื่อพิจารณาพิกเซลต่าง ๆ ของภาพให้ว่าเป็นขอบของวัตถุ

2.4 Image Transformation(MATLAB)

รูปที่ 2.11 ภาพ Checkboard เริ่มต้น

1. **Apply Linear Conformal Transform to Checkerboard**

   รูปแบบนี้จะมีทั้งการหมุนภาพ การย่อภาพและขยายภาพ ขนาดและรูปร่างของภาพจะยังคงเหมือนเดิม

2. **Apply Affine Transform to Checkerboard**

   รูปแบบนี้ แกน x และแกน y จะย่อและขยายหรือบิดไปอย่างอิสระ ขนาดและรูปร่างของภาพจะยังคงเหมือนเดิม แบบ Linear conformal transformations เป็นสับเซตของแบบ affine transformations

3. **Apply Projective Transform to Checkerboard**

   รูปแบบนี้ จะยังคงมีโครงสร้างแบบเดิม แบบ affine transformations เป็นสับเซตของแบบ projective transformations.

4. **Apply Polynomial Transform to Checkerboard**

   รูปแบบนี้ ใช้ฟังก์ชัน polynomial ของการ mapping แกน x และแกน y

5. **Apply Piecewise Linear Transform to Checkerboard**

   รูปแบบนี้ เป็นการทำแต่ละส่วนของภาพให้มีความแตกต่างกัน

6. **Apply Sinusoidal Transform to Checkerboard**

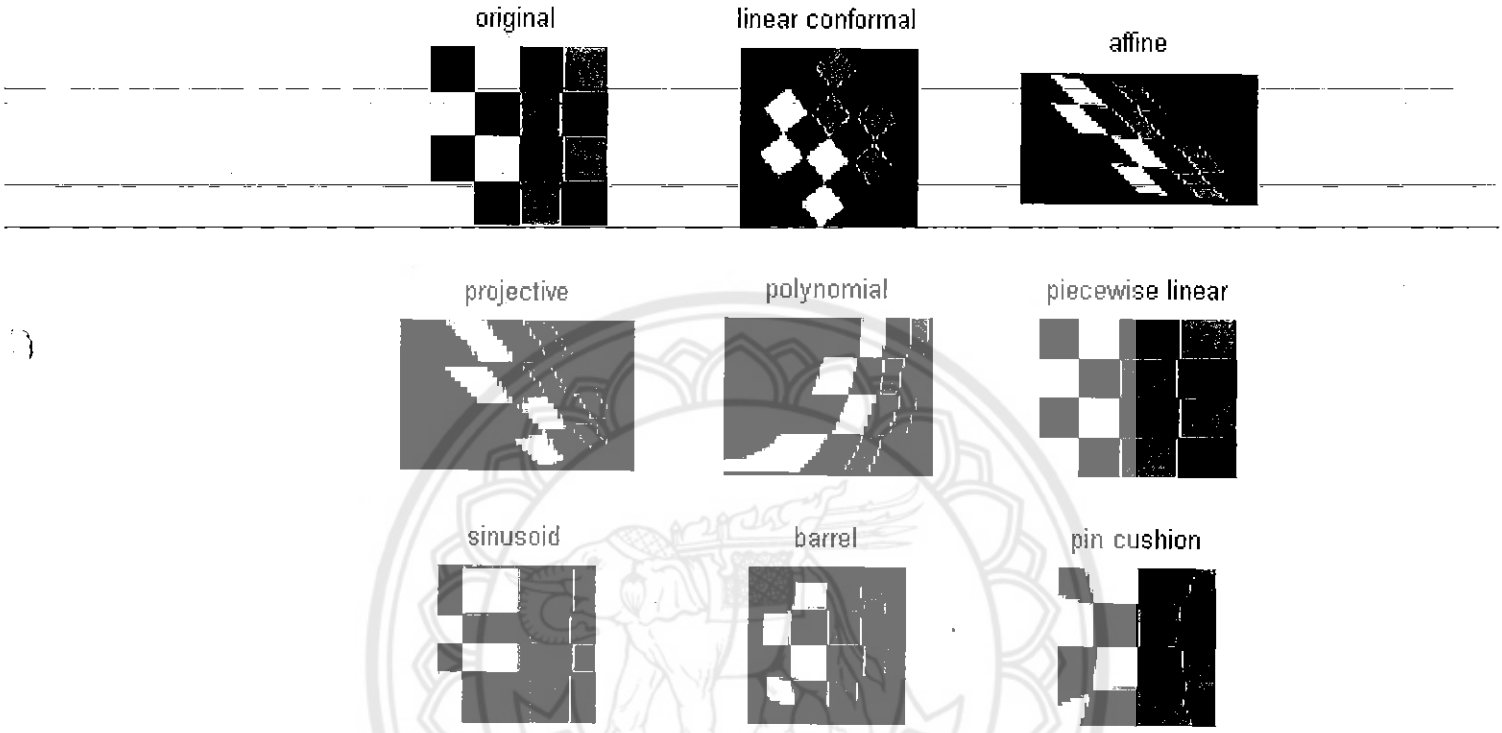
   รูปแบบนี้ใช้หลักการเดียวกับแบบ Piecewise Linear Transform

7. **Apply Barrel Transform to Checkerboard**

   รูปแบบนี้ เป็นการบิดภาพจากส่วนที่อยู่นอกจุดศูนย์กลาง ทำให้จุดศูนย์กลางของภาพนูนขึ้นและขอบของภาพโค้งลง

8. Apply Pin Cushion Transform to Checkerboard

รูปแบบนี้ ตรงข้ามกับแบบ barrel distortion ทำให้จุดศูนย์กลางของภาพนูนต่ำและขอบของภาพโค้งขึ้น



รูปที่ 2.12 ภาพรูปแบบการแปลงภาพแบบต่างๆ

2.5 การปรับปรุงรูปภาพ (Image Enhancement)

วัตถุประสงค์หลักในการปรับปรุงรูปภาพก็คือ การประมวลผลภาพให้ภาพที่ได้ออกมาใหม่นั้นมีความเหมาะสมต่อการนำไปใช้มากขึ้น ซึ่งการปรับปรุงรูปภาพนี้ไม่มีรูปแบบที่แน่นอนที่จะบอกว่าวิธีใดดีที่สุด ทั้งนี้การจะตัดสินว่าควรใช้วิธีใดจึงขึ้นอยู่กับว่าจะนำภาพไปใช้ทำอะไรในแอพพลิเคชั่นนั้นๆ วิธีการปรับปรุงรูปภาพจะแบ่งออกเป็น 2 ประเภทคือ

- **วิธีสปาเชียลโดเมน** (Spatial Domain Method) คือ การประมวลผลกับค่าที่อยู่ในแต่ละพิกเซลนั้นโดยตรง
- **วิธีฟรีเควนซีโดเมน** (Frequency Domain Method) คือ การประมวลผลกับภาพที่ถูกแปลงด้วยวิธี ฟูเรียร์ทรานส์ฟอร์ม (Fourier Transform) ก่อน

2.5.1 วิธีสปาเชียลโดเมน (Spatial Domain)

กระบวนการที่ทำกับ spatial domain สามารถแทนด้วยสมการคณิตศาสตร์ดังนี้

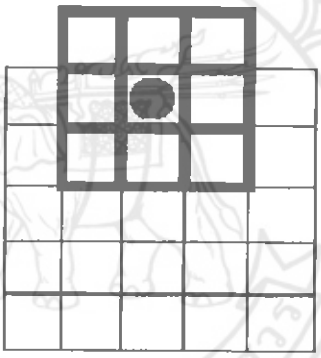
$$g(x,y) = T(f(x,y)) \quad (2.18)$$

โดยที่ f(x,y) คือ รูปภาพที่รับเข้ามา

g(x,y) คือ ภาพที่ได้ประมวลผลแล้ว

T คือ กระบวนการที่ทำกับ f โดยทำกับจุดข้างเคียงของ (x,y) ด้วย

การที่จะนำจุดข้างเคียงมาประมวลผลด้วย สามารถทำได้โดยใช้ภาพ มาครอบกับภาพที่รับเข้ามา และให้จุดกึ่งกลางของ mask อยู่ที่จุด (x,y) ดังรูป



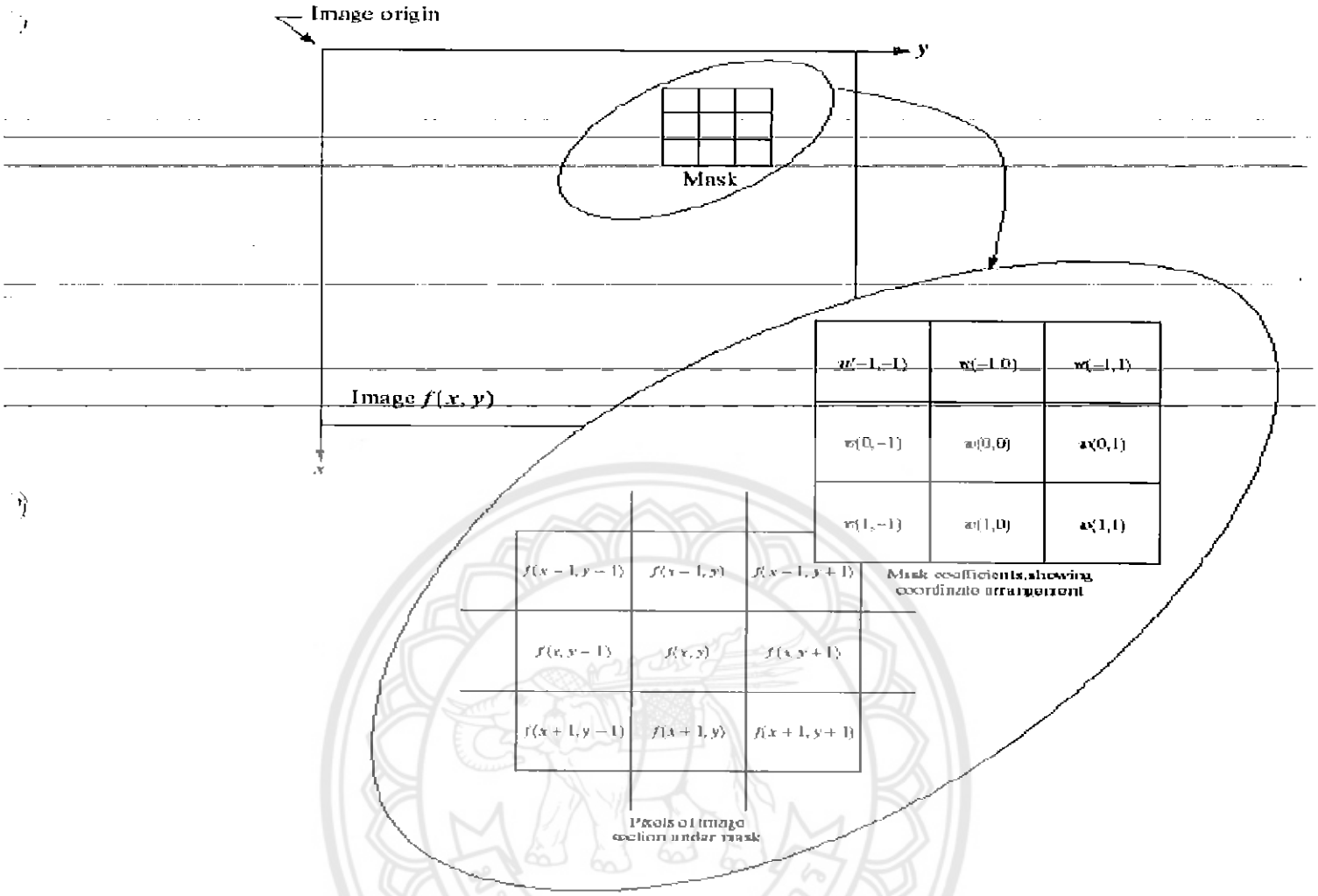
รูปที่ 2.13 แสดงการใช้หน้าต่างครอบภาพที่รับเข้ามา

Spatial Filtering

Filter คือ image ย่อย (subimage) ที่มีค่าของแต่ละพิกเซลเป็นค่าสัมประสิทธิ์ อาจเรียกได้เป็นชื่ออื่น เช่น Mask, Kernel ,Template หรือ Window โดยส่วนใหญ่แล้วจะมีขนาดเป็นเลขคี่ เช่น 3×3, 5×5,

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

รูปที่ 2.14 ตัวอย่าง mask ของ spatial filtering โดยมีขนาด 3×3



รูปที่ 2.15 วิธีการใช้ mask

จากรูปจะเป็นการย้าย mask แต่ละตัวไปที่แต่ละใน image จากนั้นหาผลรวมของผลคูณของ mask coefficient กับค่าของพิกเซลใน image ณ ตำแหน่งที่มีความสัมพันธ์กัน ตัวอย่างจากรูปเป็น mask ขนาด 3×3 ที่จุด w(0,0) จะตรงกับจุด f(x,y) ใน mask ผลลัพธ์ที่ได้ใน image จะได้

$$R = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \ldots + w(0,0)f(0,0) + \ldots + w(1,1)f(x+1,y+1)$$

รูปแบบทั่วไปการเขียน linear filtering ของ image f ขนาด m×n จะได้เป็น

$$g(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} \omega(s,t) f(x+s, y+t) \qquad (2.19)$$

เมื่อ a = (m-1)/2 และ b = (n-1)/2

เพื่อให้ได้ image g อย่างสมบูรณ์จะต้องหาค่าให้ครอบคลุม x = 0,1,2,... M-1 และ y = 0,1,2, … , N-1

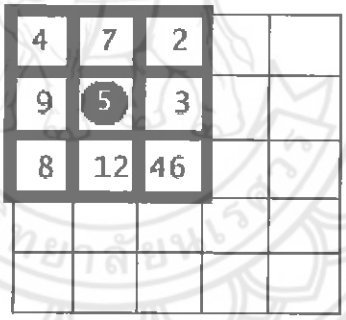
Smoothing Spatial Filters

Smoothing Spatial Filters ไว้ใช้สำหรับการทำให้ภาพเบลอโดยกำจัดรายละเอียดเล็กๆน้อยๆจากวัตถุที่มีขนาดใหญ่ใน image และทำการเชื่อมช่องว่างเล็กๆของเส้นตรงหรือเส้นโค้ง และสำหรับการลบสัญญาณรบกวนออก โดยลดรายละเอียดเล็กๆน้อยๆที่เป็นส่วนเกินที่เกิดขึ้นใน image

Smoothing linear Filters

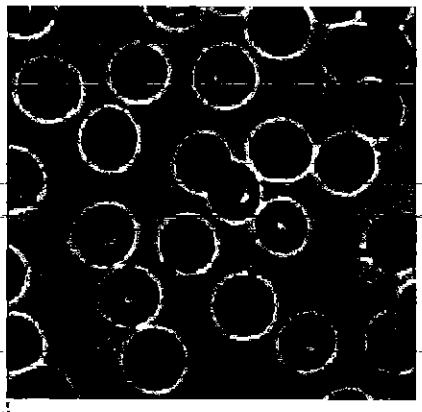
หรือจะเรียกอีกอย่างหนึ่งว่า "Averaging Filters" ผลลัพธ์ที่ได้จากการใช้ในแต่ละจุดก็คือค่าเฉลี่ยของพิกเซลทุกพิกเซลที่อยู่ใน mask วิธีการคือจะแทนที่ค่าของพิกเซลที่อยู่ใน image ด้วยค่าเฉลี่ยของ gray-level ในขอบเขตที่กำหนดโดย mask โดยที่ผลลัพธ์ที่ได้จะลดส่วนที่มีการเปลี่ยนแปลงกะทันหัน (sharp transition) นั่นคือลดสัญญาณรบกวนที่มีการเปลี่ยนแปลงกะทันหันมาก แต่การใช้วิธีนี้จะทำให้ลดความคมชัดของขอบของวัตถุ (edge) ลงไปด้วย

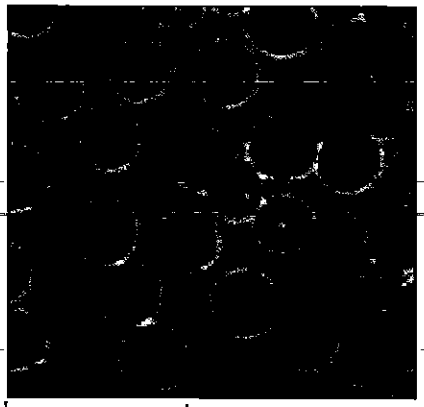
ยกตัวอย่างเช่น filter ขนาด 3×3 ดังรูป



รูปที่ 2.16 การทำ average filter

จากรูปแสดงการทำ average filter จะได้ค่าที่จุดกึ่งกลางเป็น (4+7+2+9+5+3+8+12+46)/9 = 10.67





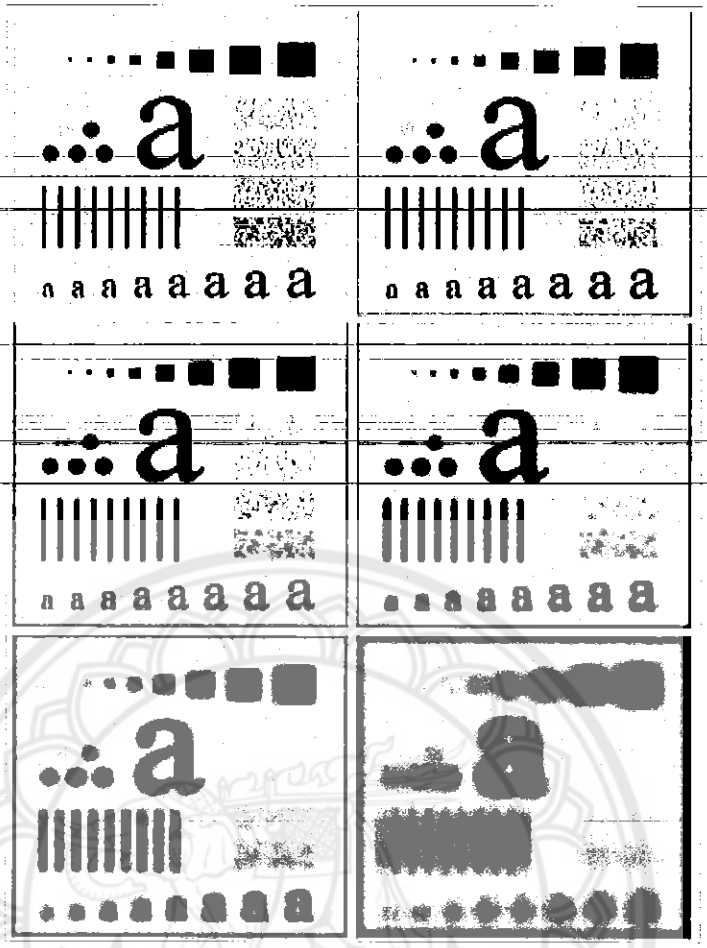
รูปที่ 2.17 ภาพก่อนการทำ average filtering **รูปที่ 2.18** แสดงรูปที่ผ่าน average filtering

Weighted average filter

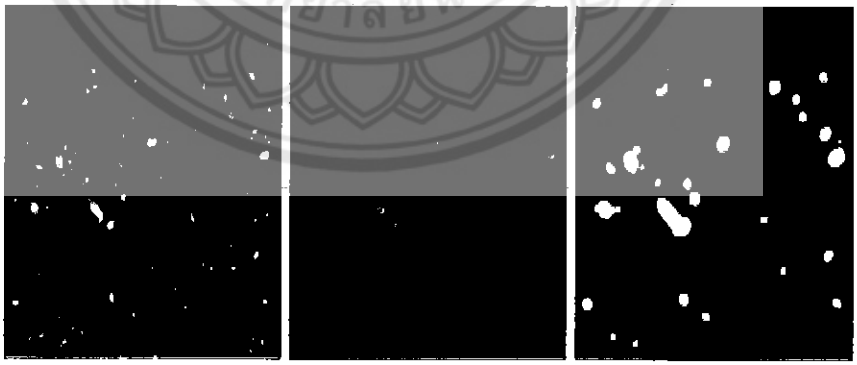
คือการที่พิกเซลใน image ถูกคูณด้วยค่าสัมประสิทธิ์ที่แตกต่างกัน โดยจะกำหนดความสำคัญของพิกเซลแตกต่างกัน สมการทั่วไป

$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b}\omega(s,t)f(x+s,y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b}\omega(s,t)} \qquad (2.20)$$

โดยที่ $\sum_{s=-a}^{a}\sum_{t=-b}^{b}\omega(s,t)$ เป็นผลรวมของค่าสัมประสิทธิ์ใน mask



รูปที่ 2.19 ภาพตัวอย่างการใช้ smoothing linear filter

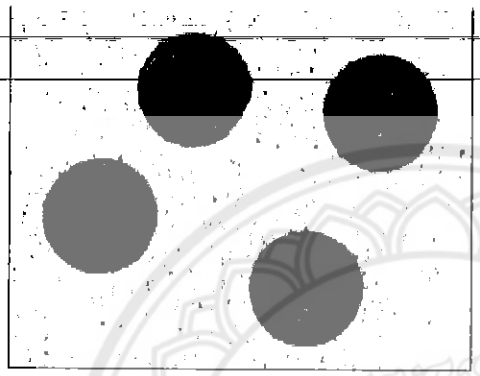


รูปที่ 2.20 ภาพต้นฉบับ ภาพที่ผ่านการทำ averaging mask และ thresholding

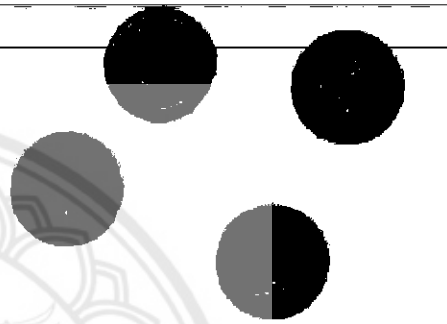
Order-Statistics filters

เป็น non-linear spatial filters โดยเป็นการจัดลำดับค่าของพิกเซลที่อยู่พื้นที่ของ mask นั้น แล้วแทนที่พิกเซลที่อยู่ตรงกลางด้วยค่าที่กำหนดในลำดับนั้น โดยมีดังนี้

- Median filter ผลลัพธ์ที่ได้จากการทำ median filter ก็คือ การแทนที่ค่าในพิกเซลทั้งหมดที่ mask ครอบอยู่ด้วยค่ามัธยฐาน จะใช้ได้ดีในการลด impulse noise หรือเรียกว่า salt-and-pepper noise นั่นก็คือ ภาพที่มีลักษณะเป็นจุดขาวและจุดดำ จากรูปถ้าจะประมวลผลกับภาพด้วย median filter จะทำได้โดย เรียงค่าจากน้อยไปมากนั่นก็คือ 2, 3, 4, 5, 7, 8, 9, 12 และ 46 เพราะฉะนั้นค่ามัธยฐานก็คือ 7 ภาพที่ใช้ median filter แสดงดังตัวอย่างข้างล่างดังนี้



รูปที่ 2.21 ภาพก่อนการทำ median filter

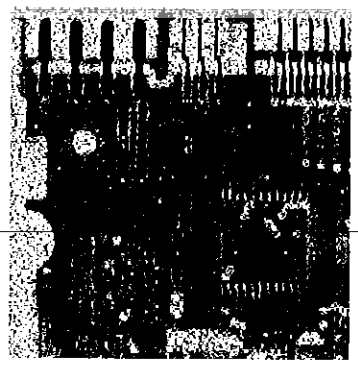


รูปที่ 2.22 ภาพที่ผ่านการทำ median filter

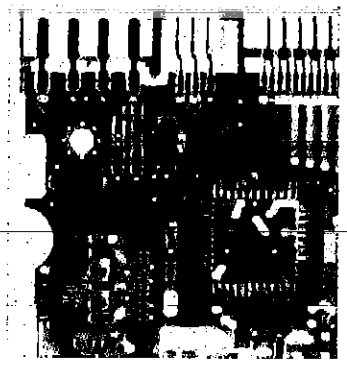
- Max filter ผลลัพธ์ที่ได้จากการ Max filter ก็คือ การแทนที่ค่าในพิกเซลทั้งหมดที่ mask ครอบอยู่ด้วยค่าสูงสุด

$$\hat{f}(x,y) = \max\{g(s,t)\} \tag{2.21}$$

เช่น ถ้าจะประมวลผลกับภาพ Max filter จะทำได้โดยหาค่ามากที่สุด เช่น 2, 3, 4, 5, 7, 8, 9, 12 และ 46 เพราะฉะนั้นค่าสูงสุดก็คือ 46



รูปที่ 2.23 ภาพก่อนการทำ Max filter

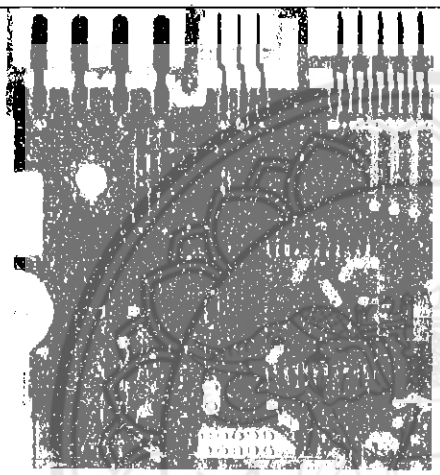


รูปที่ 2.24 ภาพที่ผ่านการทำ การ Max filter

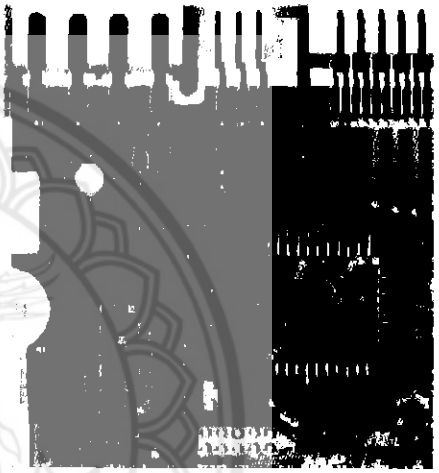
- Min filter ผลลัพธ์ที่ได้จากการ Min filter ก็คือ การแทนที่ค่าในพิกเซลทั้งหมดที่ mask ครอบอยู่ด้วยค่าต่ำสุด

$$\hat{f}(x,y) = \min\{g(s,t)\} \qquad (2.22)$$

เช่น ถ้าจะประมวลผลกับภาพ Min filter จะทำได้โดยหาค่าน้อยที่สุด เช่น 2, 3, 4, 5, 7, 8, 9, 12 และ 46 เพราะฉะนั้นค่าสูงสุดก็คือ 2



รูปที่ 2.25 ภาพก่อนการทำ Min filter



รูปที่ 2.26 ภาพที่ผ่านการทำ Min filter

2.5.2 วิธีฟรีเควนซีโดเมน (Frequency Domain Method)

Fourier transform หมายถึงการแปลงเชิงปริพันธ์ โดยเป็นการเขียนแทนฟังก์ชันใดๆ ในรูปผลบวก หรือปริพันธ์ ของฐาน ที่เป็นฟังก์ชันรูปคลื่น ไซน์หรือ โคไซน์ โดยในการหา Fourier transform 2 มิติ จะหาได้จาก

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy \qquad (2.23)$$

และInverse Fourier transform หาได้จาก

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}dudv \qquad (2.24)$$

Discrete Fourier transform (DFT)

สำหรับการคำนวณด้วยเครื่องคอมพิวเตอร์ ค่าสัญญาณในทั้งสองโดเมนจำเป็นต้องมีค่าเป็นดิจิตอล ซึ่งคือฟังก์ชันค่าไม่ต่อเนื่อง $x[n]$ บนโดเมนไม่ต่อเนื่อง แทนที่จะเป็นโดเมนต่อเนื่องในช่วงจำกัด หรือ เป็นคาบ ในกรณีนี้เราจะใช้ การแปลงฟูริเยร์ไม่ต่อเนื่อง (discrete Fourier transform - DFT) ซึ่งเขียนแทน $x[n]$ ด้วยผลบวกของฟังก์ชันคาบ

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{2\pi ink/N} \qquad n = 0, \ldots, N-1 \tag{2.25}$$

โดยที่ $X[k]$ คือ ค่าขนาดบนโดเมน

Discrete Fourier transform ในรูปดิจิตอลที่มีขนาด M×N เราจะได้

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{2.26}$$

สำหรับ u = 0, 1, …, M-1 และ v = 0, 1, …, N-1

และจะเขียน Inverse discrete Fourier transform ได้เป็น

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{2.27}$$

สำหรับ x = 0, 1, …, M-1 และ y = 0, 1, …, N-1

จากทฤษฎีบทของออยเลอร์ คือ $e^{j\theta} = \cos(\theta) + j\sin(\theta)$ จะทำให้เราสามารถเขียน Discrete Fourier transform (DFT) ได้ใหม่เป็น

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)[\cos(2\pi(\frac{ux}{M}+\frac{vy}{N})) - j\sin(2\pi(\frac{ux}{M}+\frac{vy}{N}))] \tag{2.28}$$

สำหรับ u = 0, 1, …, M-1 และ v = 0, 1, …, N-1

จากสมการข้างบนจะเห็นได้ว่า $f(x,y)$ ได้ถูกคูณด้วยไซน์และโคไซน์ที่ความถี่ต่างๆกัน จึงทำให้เราเรียกช่วงนี้ว่า “frequency domain” นอกจากนั้นเราสามารถหาค่า Fourier Spectrum ได้จาก

$$|F(u,v)| = [R^2(u,v) + I^2(u,v)]^{\frac{1}{2}} \quad (2.29)$$

โดยมุมของเฟสมีค่าเท่ากับ

$$\phi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right] \quad (2.30)$$

จากนั้นเราสามารถหาค่า power spectrum ได้จากสมการ

$$P(u,v) = [F(u,v)]^2 = R^2(u,v) + I^2(u,v) \quad (2.31)$$

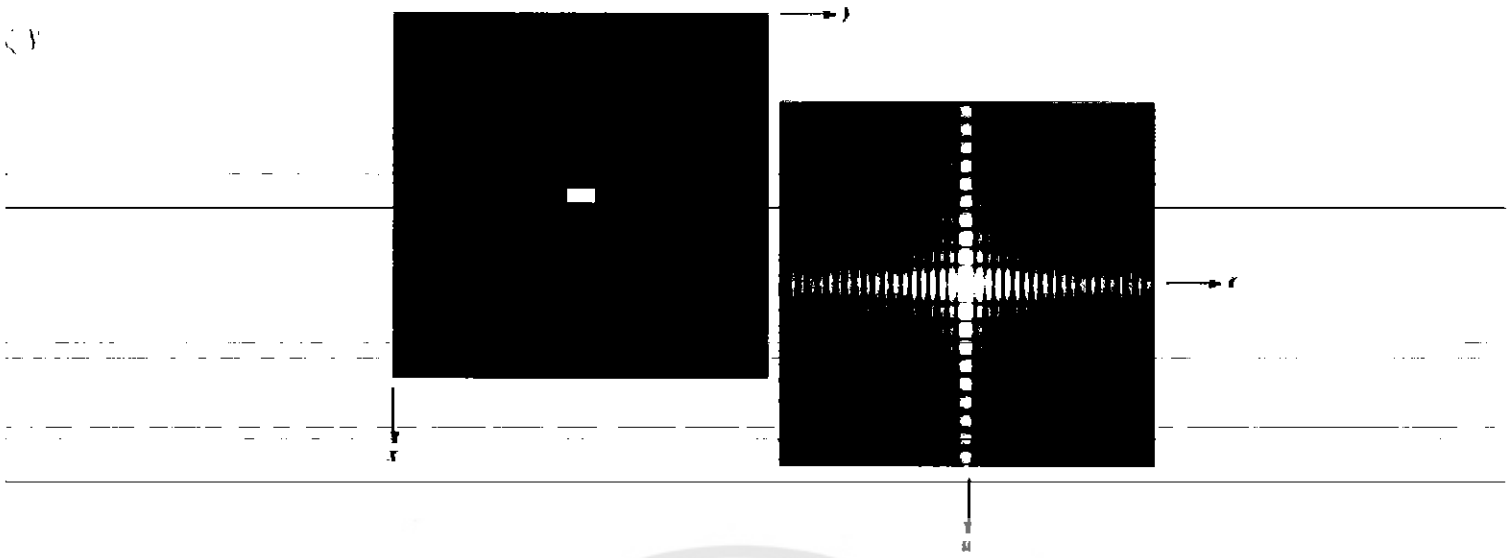
โดยที่ $R(u,v)$ คือส่วนจริง และ $I(u,v)$ คือส่วนจินตภาพของ F(u,v)
ถ้าเรากำหนด (u,v) ที่ (0,0) เราจะได้

$$F(0,0) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) \quad (2.32)$$

จากสมการก็คือค่าเฉลี่ยของ f(x, y) นั่นเอง และถ้าเรามองในลักษณะของ image ก็คือค่าเฉลี่ย Gray level ของ image

Low frequency คือการเปลี่ยนแปลงของ gray-level อย่างช้าๆ ซึ่งเราเรียกว่า smooth gray-level variation ส่วน High frequency คือการเปลี่ยนแปลงของ gray-level อย่างรวดเร็ว เราเรียกว่า fast gray-level variation

Low pass filter คือ filter ที่ทำการลดทอนความถี่สูงและให้ความถี่ต่ำผ่านได้ง่าย ส่วน high pass filter คือ filter ที่ทำการลดทอนความถี่ต่ำและให้ความถี่สูงผ่านได้ง่าย



รูปที่ 2.27 ภาพต้นฉบับและภาพที่ผ่านการทำ Fourier Spectrum

Smoothing Frequency-Domain Filter

frequency domain จะมีความสัมพันธ์กับ spatial domain ความแตกต่างของสองอย่างนี้คือ ภายใน spatial domain จะมีการใช้ mask ในแต่ละพิกเซล ส่วนภายใน frequency domain เราจะใช้ "convolution" ซึ่งจะเขียนได้เป็น $f(x,y) * h(x,y)$ ซึ่งผลลัพธ์ที่ได้จะสอดคล้องกับผลลัพธ์ของ $F(u,v)H(u,v)$ นั่นคือ

$$f(x,y) * h(x,y) \quad \Leftrightarrow \quad F(u,v)H(u,v) \tag{2.33}$$

โดยวิธีการทำ Smoothing Frequency-Domain Filter มี 3 วิธี ดังนี้

1. Ideal Low pass Filter (ILPF)
2. Butterworth Low pass Filter (BLPF)
3. Gaussian Low pass Filter (GLPF)

Ideal Low pass Filter (ILPF)

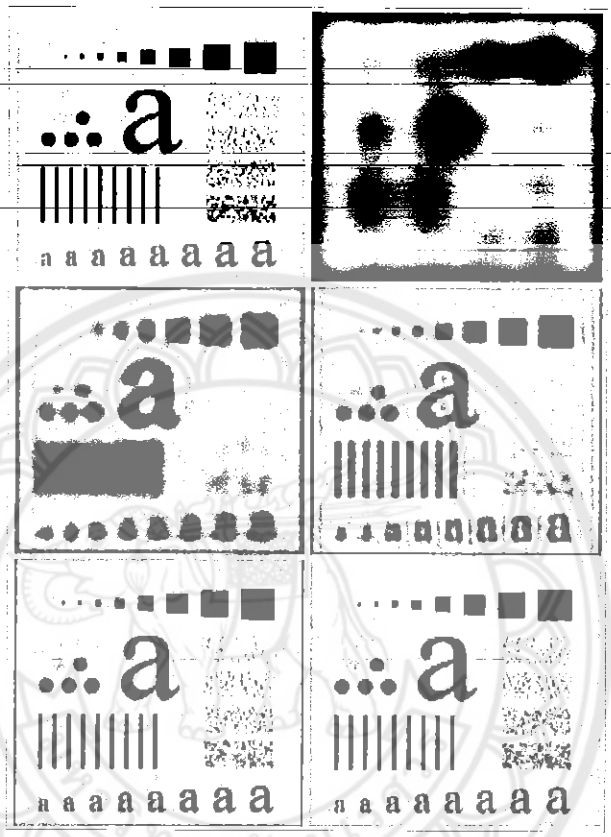
เป็น filter ที่ง่ายที่สุด โดยเราจะกำหนด Transfer function เป็น

$$\begin{aligned} H(u,v) &= 1 \quad if\ D(u,v) \le D_0 \\ &= 0 \quad if\ D(u,v) > D_0 \end{aligned} \tag{2.34}$$

โดยที่ D_0 คือ ค่าที่กำหนดให้ต้องไม่เป็นลบ โดยเราจะเรียกค่านี้ว่า cut off frequency

$D(u,v)$ คือ ระยะจากจุด (u,v) ไปยังจุดศูนย์กลาง โดยใช้สูตรดังนี้

$$D(u,v) = \left[(u-\frac{M}{2})^2 + (v-\frac{N}{2})^2\right]^{\frac{1}{2}} \quad (2.35)$$

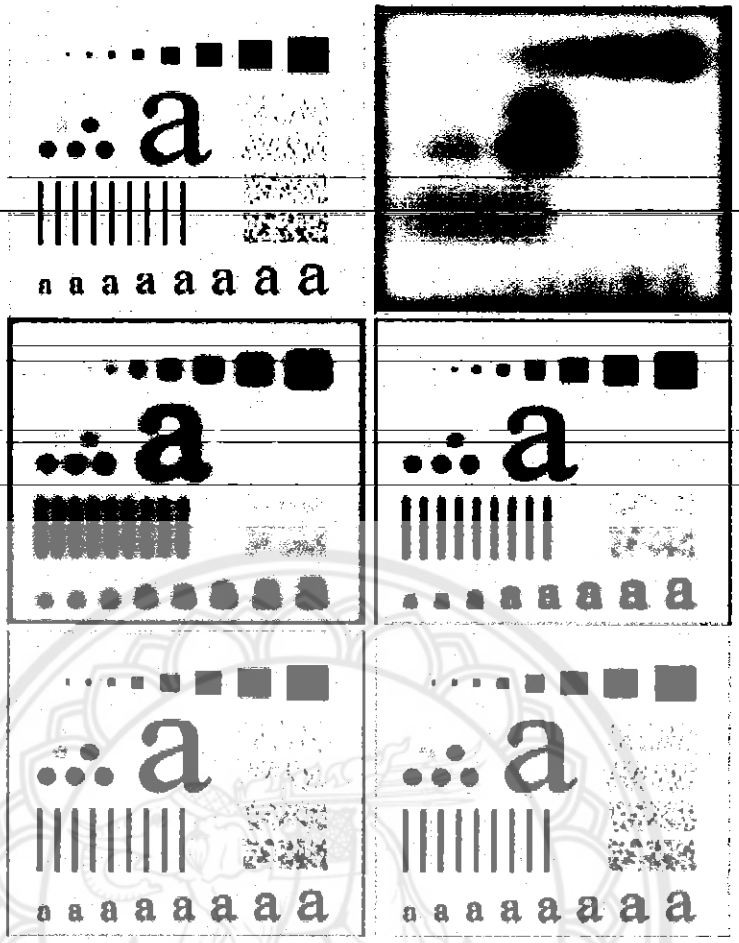


รูปที่ 2.28 ภาพต้นฉบับและภาพที่ผ่านการทำ ILPF โดยใช้ค่าแตกต่างกัน

Butterworth Low pass Filter (BLPF)

Transfer function ของ Butterworth low pass filter order n โดยมี cut off frequency ที่ D_0 ถูกกำหนดโดย

$$H(u,v) = \frac{1}{1+\left[\frac{D(u,v)}{D_0}\right]^{2n}} \quad (2.36)$$

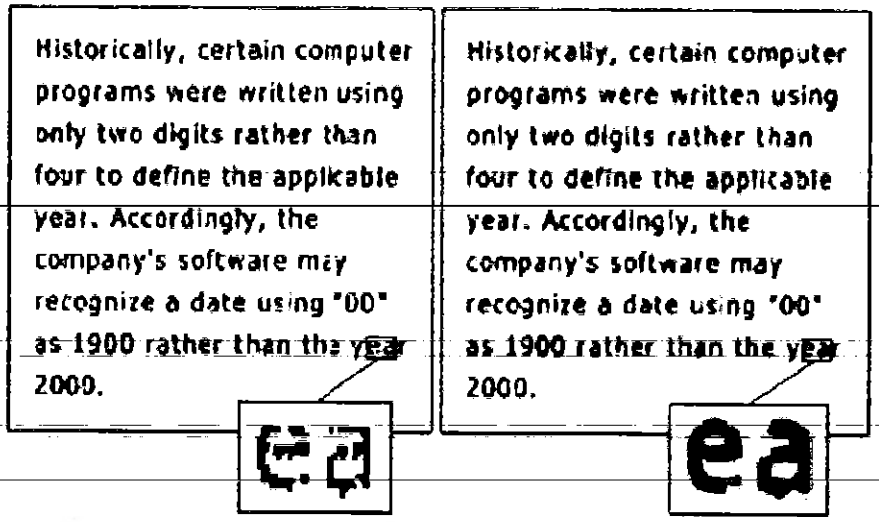


รูปที่ 2.29 ภาพต้นฉบับและภาพที่ผ่านการทำ BLPF โดยใช้ค่าแตกต่างกัน

Gaussian Low pass Filter (GLPF)

Transfer function ของ Gaussian low pass filter ถูกกำหนดโดย

$$H(u,v) = e^{\frac{-D^2(u,v)}{2\sigma^2}} \tag{2.37}$$



รูปที่ 2.30 ภาพต้นฉบับและภาพที่ผ่านการทำ GLPF

Sharpening Frequency-Domain Filter

กระบวนการ blurring คือการทำให้ภาพ smooth โดยจะทำการลดทอนส่วนประกอบที่มีความถี่สูง และกระบวนการ Sharpening จะลดทอนส่วนประกอบที่มีความถี่ต่ำ ดังนั้น low pass filter และ high pass filter จะมีความสัมพันธ์ดังสมการ

$$H_{hp}(u,v) = 1 - H_{lp}(u,v) \qquad (2.38)$$

โดยวิธีการทำ Smoothing Frequency-Domain Filter มี 3 วิธี ดังนี้

1. Ideal High pass Filter (IHPF)
2. Butterworth High pass Filter (BHPF)
3. Gaussian High pass Filter (GHPF)

Ideal High pass Filter (IHPF)

เป็น filter ที่ง่ายที่สุด โดยเราจะกำหนด Transfer function เป็น

$$\begin{aligned} H(u,v) &= 0 \quad if\ D(u,v) \le D_0 \\ &= 1 \quad if\ D(u,v) > D_0 \end{aligned} \qquad (2.39)$$

โดยที่ D_0 คือ ค่าที่กำหนดให้ต้องไม่เป็นลบ โดยเราจะเรียกค่านี้ว่า cut off frequency

$D(u,v)$ คือ ระยะจากจุด (u,v) ไปยังจุดศูนย์กลาง โดยใช้สูตรดังนี้

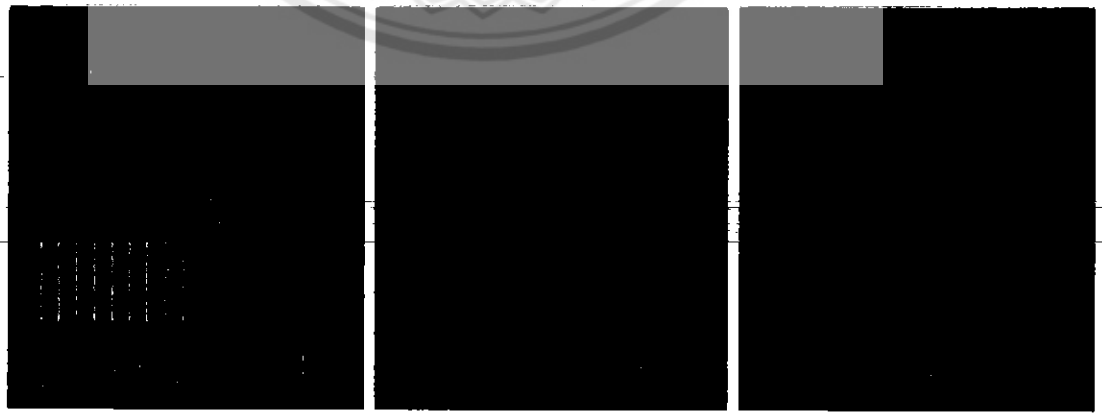
$$D(u,v) = \left[(u-\frac{M}{2})^2 + (v-\frac{N}{2})^2\right]^{\frac{1}{2}} \quad (2.40)$$

รูปที่ 2.31 ภาพต้นฉบับและภาพที่ผ่านการทำ IHPF โดยใช้ค่าแตกต่างกัน

Butterworth high pass Filter (BHPF)

Transfer function ของ Butterworth high pass filter order n โดยมี cut off frequency ที่ D_0 ถูกกำหนดโดย

$$H(u,v) = \frac{1}{1+\left[\frac{D_0}{D(u,v)}\right]^{2n}} \quad (2.41)$$



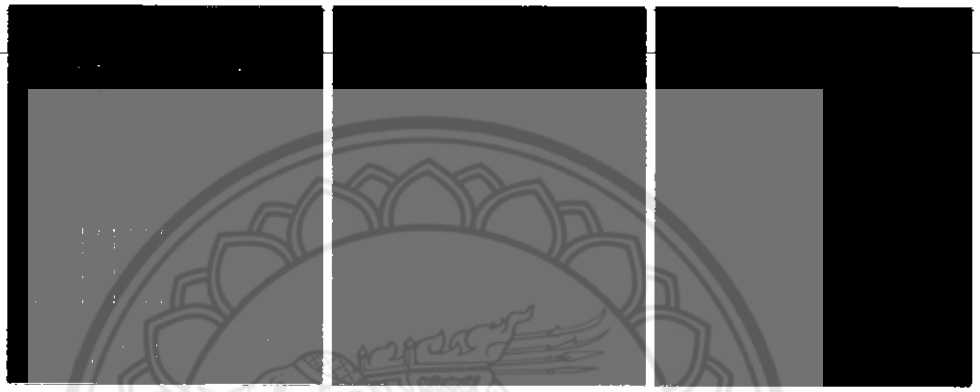
รูปที่ 2.32 ภาพต้นฉบับและภาพที่ผ่านการทำ BHPF โดยใช้ค่าแตกต่างกัน

Gaussian High pass Filter (GHPF)

Transfer function ของ Gaussian high pass filter ถูกกำหนดโดย

$$H(u,v) = 1 - e^{\frac{-D^2(u,v)}{2\sigma^2}} \qquad (2.42)$$

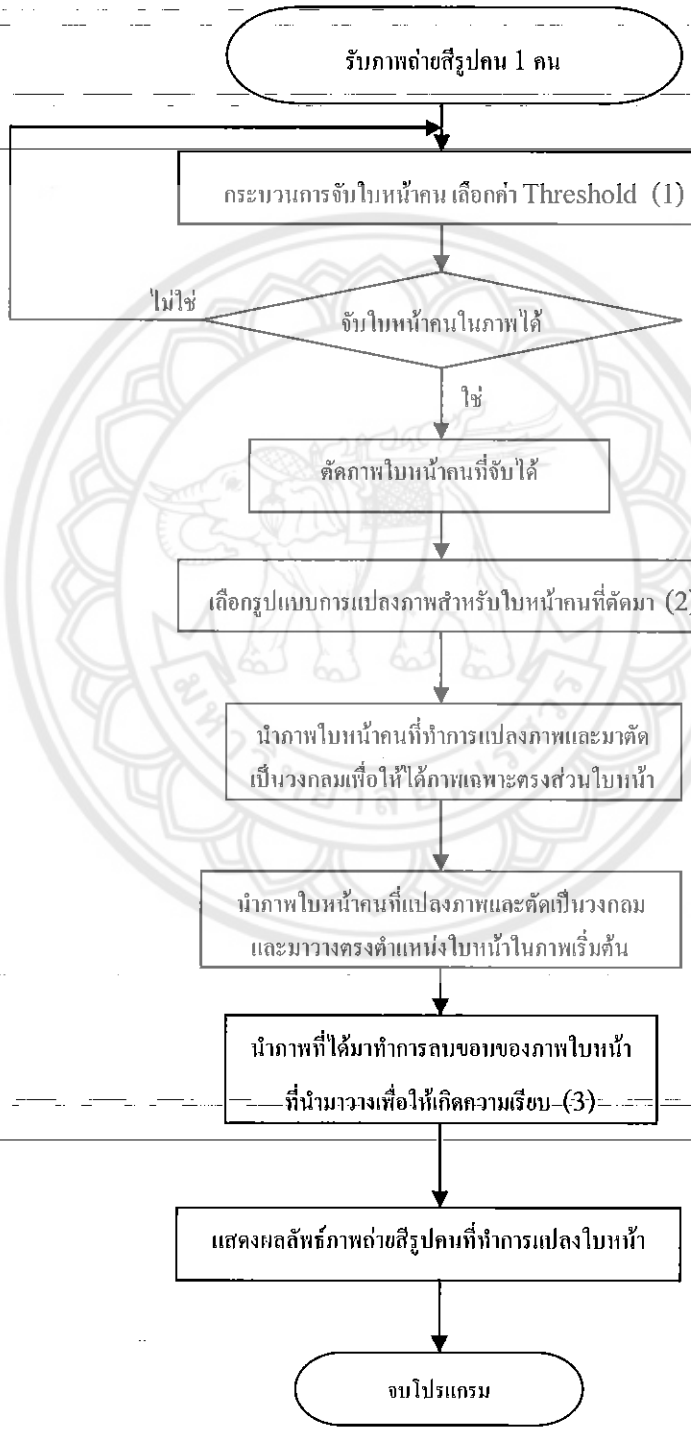
ตัวอย่างการใช้ Gaussian high pass filter (GHPF)



รูปที่ 2.33 ภาพต้นฉบับและภาพที่ผ่านการทำ GHPF โดยใช้ค่าแตก**ต่างกัน**

บทที่ 3
วิธีการดำเนินโครงงาน

3.1 แผนภาพขั้นตอนการทำงานการแปลงใบหน้าคน



รูปที่ 3.1 แผนภาพขั้นตอนการทำงาน

3.2 กระบวนการจับใบหน้าคน(Face detection)

ในกระบวนการนี้ ได้นำโปรแกรมการตรวจจับใบหน้ามาจาก *W. Kienzle, G. Bakir, M. Franz and B. Scholkopf: Face Detection - Efficient and Rank Deficient. In: Advances in Neural Information Processing Systems 17, pg. 673-680, 2005.* ซึ่งมีให้ดาวน์โหลดได้ที่ website http://www.kyb.mpg.de/bs/people/kienzlc/facedemo/facedemo.htm#fdlib จากการศึกษาและทำการทดลองเกี่ยวกับรูปภาพตัวอย่าง พบว่าภาพที่ใช้ในการทดลองเป็นภาพดิจิตอล ซึ่งภาพดิจิตอลนี้เป็นแอเรย์หลายมิติของตัวเลข ได้แก่ ภาพระดับสีเทา(gray image) หรือแอเรย์หลายมิติของเวคเตอร์ ได้แก่ ภาพสี(color) แต่ละจุดบนภาพดิจิตอล จะเรียกว่า pixel โดยที่ในแต่ละ pixel จะมีค่ากำกับเอาไว้ เรียกว่า pixel value

ค่าในแต่ละ pixel ของ gray image คือค่าความเข้มของแสง ณ แต่ละตำแหน่งของ pixel ขึ้นอยู่กับจำนวน bit ที่ใช้ ตัวอย่างเช่น 8-bit monochrome จะมี gray scale ทั้งหมด 256 ระดับ

ค่าในแต่ละ pixel ของ color image จะประกอบไปด้วย vector ที่แสดงค่าของสีแดง สีเขียว และสีน้ำเงิน อย่างละ 8-bit ดังนั้น RGB image 1 pixel จะประกอบไปด้วยจำนวนบิตทั้งหมด 24 บิต ทำให้ RGB image มีจำนวนสีที่เป็นไปได้ทั้งหมด 2^{24} สี

และสามารถนำมาหากรอบบนใบหน้าโดยผ่านกระบวนการ Threshoding โดยที่ค่าพิกเซลนั้นจะถูกแปลงจาก ภาพสี RGB เป็นภาพระดับสีเทา(gray Image) โดยกำหนดค่า Threshold ซึ่งมีได้หลายค่าสำหรับทำ Adaptive Thresholding จะไม่มีการกำหนดค่าแบบตายตัว เนื่องจากภาพแต่ละภาพมีความแตกต่างกันทั้งในด้านความสว่างและความเข้มสีขององค์ประกอบของแต่ละภาพ ดังนั้นจึงใช้เทรสโฮลดิงที่มีการปรับค่าได้(Adaptive Thresholding) เพื่อให้เหมาะสมกับแต่ละภาพ โดยภาพที่ได้หลังจากการทำเทรสโฮลดิงแล้วนั้นจะเป็นภาพขาว-ดำ

รูปที่ 3.2 ตัวอย่างภาพต้นฉบับ

รูปที่ 3.3 ภาพที่ได้รับการตรวจจับใบหน้าคน(ภาพขาว-ดำ)

และทำการแปลงภาพกลับจากภาพสีขาว-ดำที่ได้จากการตกรอบบนใบหน้า เป็นภาพสี RGB เพื่อที่จะนำภาพที่ได้จากการตีกรอบนั้นไปทำการแปลงภาพต่อไป โดยภาพที่ได้จากการตีกรอบจะเป็นภาพสี RGB

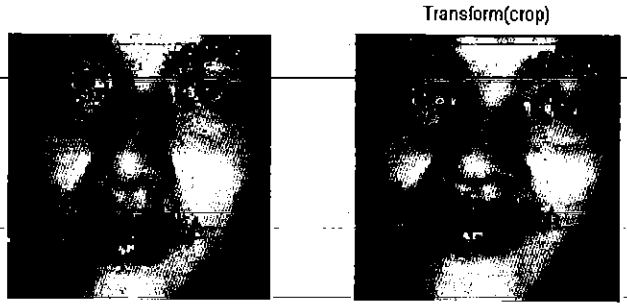
รูปที่ 3.4 ภาพสีที่ได้รับการตัดเฉพาะใบหน้า

3.3 การแปลงภาพ(Image transformation)

3.3.1 Apply Sinusoidal Transform to Checkerboard

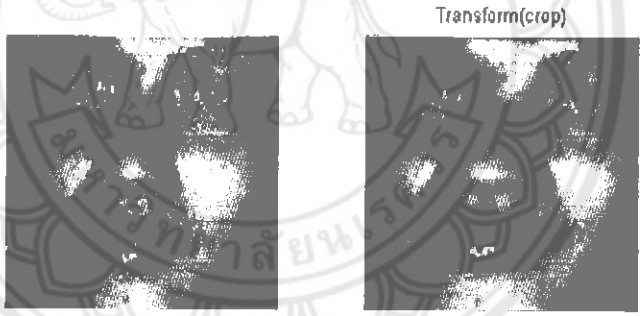
เป็นกระบวนการการย่อและการขยายภาพสามารถทำได้โดยการใช้ Scaling factor ได้แก่ Sx และ Sy ซึ่งใช้สำหรับการย่อและการขยายภาพในทางแกน x และ y ตามสมการ (2.9) และสมการ (2.10)

เมื่อผ่านกระบวนการแปลงภาพโดยการย่อภาพที่แปลงตามแนวแกน y แล้ว จะได้ภาพที่ดูแปลกไป คือ บริเวณใบหน้าจะหดเข้าหากันตามแกน y



รูปที่ 3.5 ภาพที่ได้รับการแปลงใบหน้าโดยวิธี Sinusoidal Transform(แบบที่ 1)

เมื่อผ่านกระบวนการแปลงภาพโดยการขยายภาพที่แปลงตามแนวแกน x แล้ว จะได้ภาพที่ดูแปลกไป คือ บริเวณใบหน้าจะขยายออกจากกันแกน x

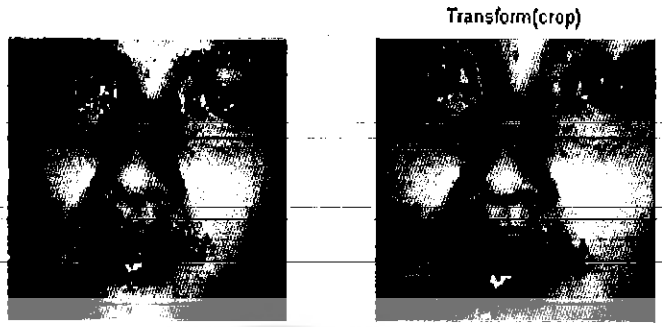


รูปที่ 3.6 ภาพที่ได้รับการแปลงใบหน้าโดยวิธี Sinusoidal Transform(แบบที่ 2)

3.3.2 Apply Pin Cushion Transform to Checkerboard

เป็นกระบวนการเป็นการหมุนภาพในระนาบ x, y เมื่อจุดศูนย์กลางการหมุน (Pivot Point) อยู่ที่จุดเริ่มต้น(Origin) จุดเริ่มต้น คือ จุดตรงกลางภาพแล้วขยายภาพออกไปด้านข้างตามมุมของภาพ ตามสมการ (2.5) และสมการ (2.6)

เมื่อผ่านกระบวนการแปลงภาพโดยการหมุนภาพในระนาบ x, y แล้ว จะได้ภาพที่ดูแปลกไป คือ บริเวณตรงกลางใบหน้าจะมีขนาดเท่าเดิมแต่บริเวณด้านข้างมุมทั้ง 4 จะถูกดึงให้มีขยายไปตามมุมทั้ง 4

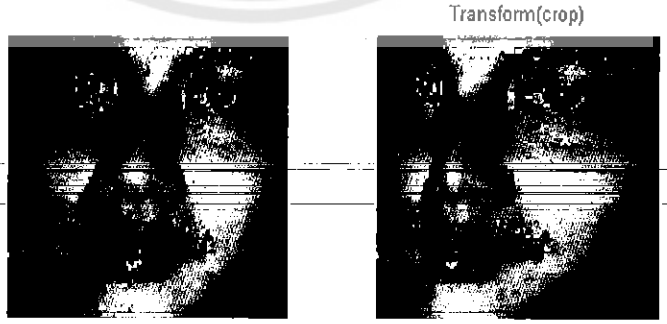


รูปที่ 3.7 ภาพที่ได้รับการแปลงใบหน้าโดยวิธี Pin Cushion Transform

3.3.3 Apply Piecewise Linear Transform to Checkerboard

เป็นกระบวนการย่อและขยายภาพในรูปเดียวกัน โดยแบ่งภาพออกเป็น 2 ส่วน เพื่อที่จะทำการย่อและขยายภาพนั้นกันคนละส่วน สามารถทำได้โดยการใช้ Scaling factor ได้แก่ Sx และ Sy ซึ่งใช้สำหรับการย่อและการขยายภาพในทางแกน x และ y ตามสมการ (2.9) และสมการ (2.10)

เมื่อผ่านกระบวนการแปลงภาพโดยการย่อภาพที่แปลงตามแนวแกน x และขยายภาพตามแนวแกน x แล้ว จะได้ภาพที่ดูแปลกไป คือ บริเวณใบหน้าจะหดเข้าหากันด้านซ้ายและขยายออกด้านขวา

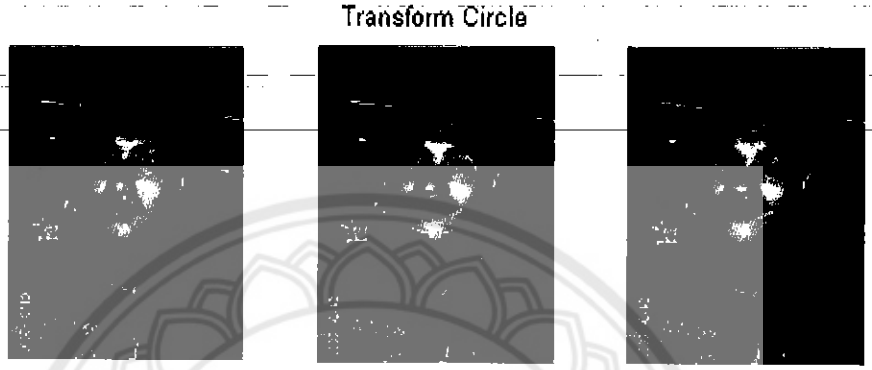


รูปที่ 3.8 ภาพที่ได้รับการแปลงใบหน้าโดยวิธี Piecewise Lincar Transform

3.4 การปรับปรุงภาพ(Image smoothing)

3.4.1 การปรับปรุงขอบวงกลมของภาพ

เมื่อได้ภาพใบหน้าคนที่ได้จากการแปลงภาพแล้วนำมาวางที่ตำแหน่งใบหน้าคนในภาพเริ่มต้น จากนั้นทำการลบส่วนเกิน(สีดำ)และปรับปรุงขอบวงกลมของภาพที่นำมาวางโดยใช้วิธี linear filtering ตามสมการ (2.19)



รูปที่ 3.9 ภาพแสดงการปรับปรุงขอบวงกลมของภาพโดยวิธี Sinusoidal Transform(แบบที่ 1)

บทที่ 4

ผลการทดลอง

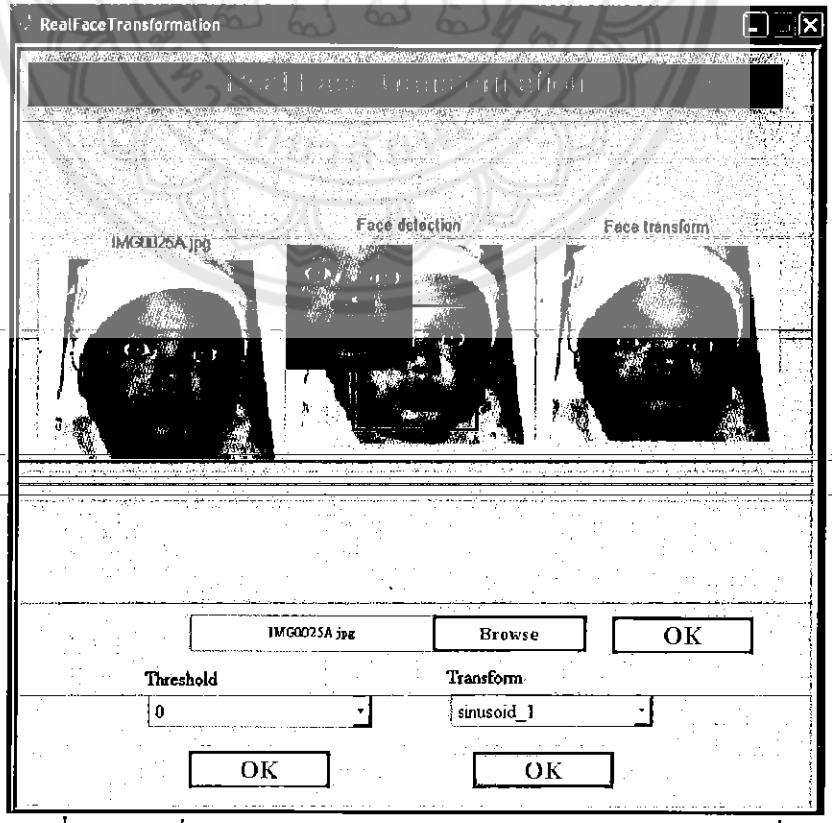
4.1 ผลการทดลอง

การแปลงใบหน้าคนจากภาพถ่ายสีรูปคน 2 มิติ เริ่มแรกนำภาพถ่ายสีรูปคน 2 มิติ ที่มีส่วนประกอบในหน้าครบทุกส่วนที่จะใช้ในการทดลอง มาทำการประมวลผลตรวจจับใบหน้า เมื่อได้ภาพใบหน้าที่ได้จากการตรวจจับแล้วนำมาทำการแปลงภาพในรูปแบบต่างๆ จากนั้นนำภาพใบหน้าที่แปลงแล้วมาวางที่ตำแหน่งใบหน้าในภาพเริ่มต้นของการทดลอง และทำการปรับปรุงภาพในขั้นตอนสุดท้าย เพื่อให้เกิดความสวยงาม ก็จะได้ผลลัพธ์เป็นภาพถ่ายสีรูปคนหน้าล้อเลียน

ต่อไปเป็นผลการทดลองของภาพที่นำมาใช้ทดสอบในโครงงานวิศวกรรมนี้ ซึ่งจะแยกเป็น 4 ประเภท ดังนี้

4.1.1 การแปลงภาพโดยวิธี Sinusoidal Transform(แบบที่ 1)

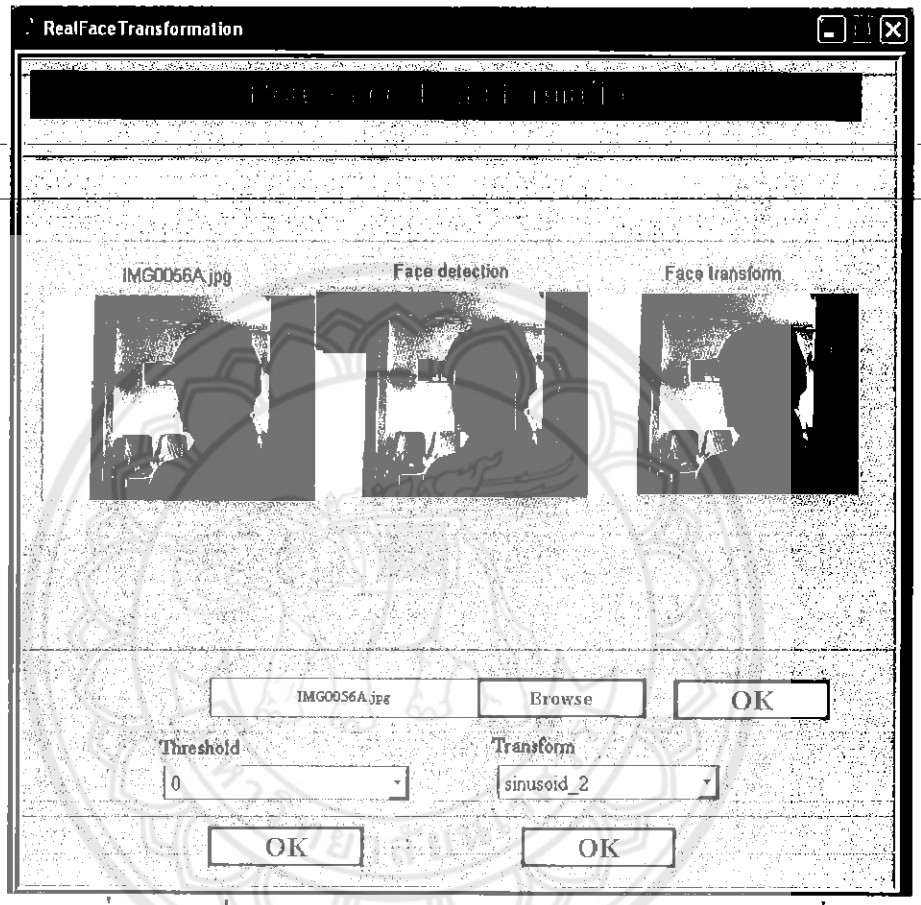
นำภาพเริ่มต้นมาผ่านกระบวนการตรวจจับใบหน้า การแปลงภาพโดยการย่อภาพที่แปลงตามแนวแกน y แล้ว จะได้ภาพที่ดูแปลกไป คือ บริเวณใบหน้าจะหดเข้าหากันตามแกน y และนำมาปรับปรุงภาพ ซึ่งโปรแกรมทำการแปลงใบหน้าคนได้ถูกต้อง ดังรูปที่ 4.1



รูปที่ 4.1 ภาพที่ได้จากการแปลงโดยวิธี Sinusoidal Transform(แบบที่ 1)

4.1.2 การแปลงภาพโดยวิธี Sinusoidal Transform(แบบที่ 2)

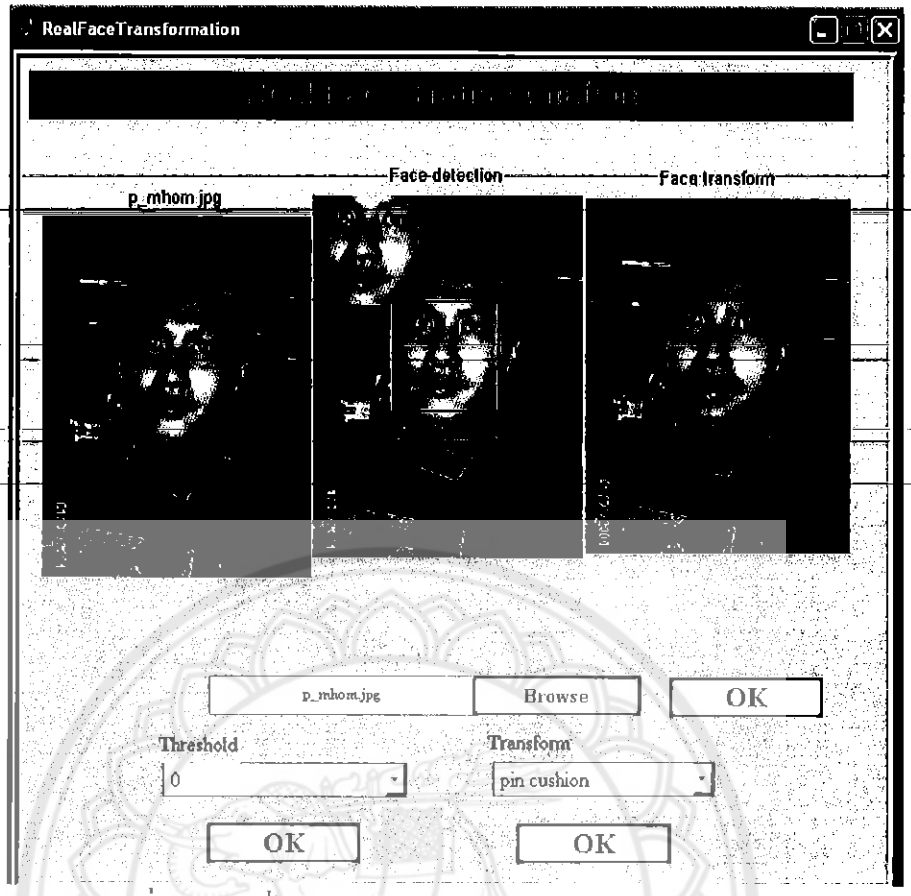
นำภาพเริ่มต้นมาผ่านกระบวนการตรวจจับใบหน้า การแปลงภาพโดยการขยายภาพที่แปลงตามแนวแกน x แล้ว จะได้ภาพที่ดูแปลกไป คือ บริเวณใบหน้าจะขยายออกจากกันแกน x และนำมาปรับปรุงภาพ ซึ่งโปรแกรมทำการแปลงใบหน้าคนได้ถูกต้อง ดังรูปที่ 4.2



รูปที่ 4.2 ภาพที่ได้จากการแปลงโดยวิธี Sinusoidal Transform(แบบที่ 2)

4.1.3 การแปลงภาพโดยวิธี Pin Cushion Transform

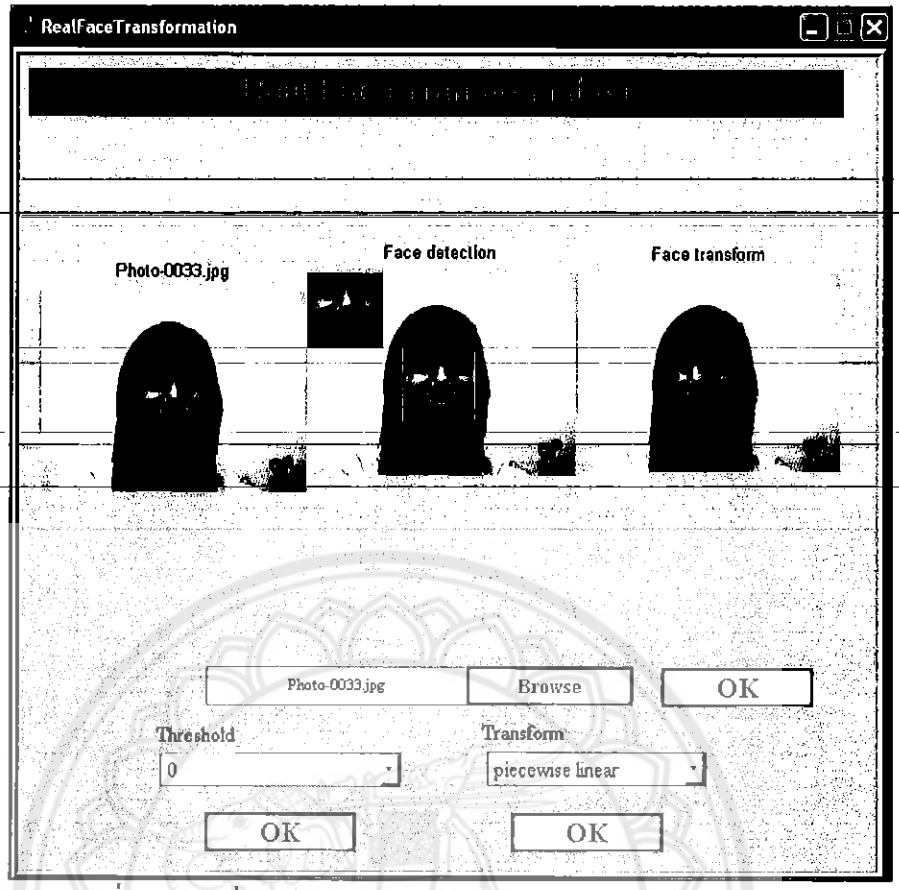
นำภาพเริ่มต้นมาผ่านกระบวนการตรวจจับใบหน้า การแปลงภาพโดยการหมุนภาพในระนาบ x, y แล้ว จะได้ภาพที่ดูแปลกไป คือ บริเวณตรงกลางใบหน้าจะมีขนาดเท่าเดิมแต่บริเวณด้านข้างมุมทั้ง 4 จะถูกดึงให้มีขยายไปตามมุมทั้ง 4 และนำมาปรับปรุงภาพ ซึ่งโปรแกรมทำการแปลงใบหน้าคนได้ถูกต้อง ดังรูปที่ 4.3



รูปที่ 4.3 ภาพที่ได้จากการแปลงโดยวิธี Pin Cushion Transform

4.1.4 การแปลงภาพโดยวิธี Piecewise Linear Transform

นำภาพเริ่มต้นมาผ่านกระบวนการตรวจจับใบหน้า การแปลงภาพโดยการย่อภาพที่แปลงตามแนวแกน x และขยายภาพตามแนวแกน x แล้ว จะได้ภาพที่ดูแปลกไป คือ บริเวณใบหน้าจะหดเข้าหากันด้านซ้ายและขยายออกด้านขวา และนำมาปรับปรุงภาพ ซึ่งโปรแกรมทำการแปลงใบหน้าคนได้ถูกต้อง ดังรูปที่ 4.4



รูปที่ 4.4 ภาพที่ได้จากการแปลงโดยวิธี Piecewise Linear Transform

4.2 วิเคราะห์ผลการทดลอง

จากการทดลองพบว่าในการตรวจจับใบหน้าคนในภาพนั้น ถ้าภาพมีองค์ประกอบของหน้าไม่ครบทุกส่วนทั้งตา จมูก ปาก เป็นต้น ภาพไม่มีความชัด มีขนาดใหญ่และเล็กเกินไป ก็จะไม่สามารถจับใบหน้าคนในภาพได้ ส่วนของการแปลงใบหน้าคนนั้นสามารถแปลงได้กับทุกใบหน้าที่มีองค์ระกอบของหน้าครบทุกส่วน แต่การแปลงใบหน้าที่ทำให้ยากต่อการปรับปรุงภาพคือ การแปลงใบหน้าโดยใช้วิธี Pin Cushion Transform ในการปรับปรุงภาพนั้น ภาพที่ทำการปรับปรุงแล้วจะมีความเรียบเนียนขึ้น จากขอบของภาพที่ตรวจจับใบหน้าแล้วแปลงภาพ ภาพที่ปรับปรุงขอบสี่เหลี่ยมจะทำได้ไม่เรียบเนียนเท่ากับภาพที่ปรับปรุงขอบวงกลม

บทที่ 5
บทสรุป

5.1 สรุปผลการทดลอง

จากการทดลองพบว่าในการตรวจจับใบหน้าคนในภาพนั้น ถ้าภาพมีองค์ประกอบของหน้าครบทุกส่วนทั้งตา จมูก ปาก เป็นต้น ภาพมีความชัด มีขนาดไม่ใหญ่และเล็กเกินไป ก็จะสามารถจับใบหน้าคนในภาพได้ ในการแปลงภาพก็สามารถแปลงภาพใบหน้าในรูปแบบต่างๆได้และสามารถปรับปรุงภาพเพื่อให้เกิดความเรียบเนียนในภาพ

5.2 ปัญหาและแนวทางแก้ไข

ตารางที่ 5.1 ตารางแสดงปัญหาและแนวทางในการแก้ปัญหา

ปัญหา	แนวทางในการแก้ปัญหา
1. ภาษาที่ใช้ในการเขียนโปรแกรม(C++,JAVA) มีความยากต่อการทำโครงงานวิศวกรรมนี้และยากต่อการทำ application	1. เลือกใช้โปรแกม MATLAB ในการเขียนโปรแกรมและง่ายต่อการทำ GUI
2. ในการปรับปรุงภาพที่นำภาพสี่เหลี่ยมมาวางที่ตำแหน่งใบหน้านั้นเกิดความไม่เรียบเนียน	2. ใช้การปรับปรุงภาพจากการนำภาพสี่เหลี่ยมมาเป็นการนำภาพวงกลมมาวางบนตำแหน่งใบหน้า

5.3 ข้อเสนอแนะสำหรับการพัฒนา

5.3.1 ปรับปรุงโปรแกรมการตรวจจับใบหน้าให้มีประสิทธิภาพที่ดีขึ้นเพื่อตรวจจับใบหน้าได้แม่นยำยิ่งขึ้น

5.3.2 สามารถนำทฤษฎีการแปลงภาพอื่นๆมาประยุกต์ใช้ในการแปลงภาพได้หลากหลาย

5.3.3 สามารถนำทฤษฎีการปรับปรุงภาพอื่นๆมาประยุกต์ใช้ในการปรับปรุงภาพเพื่อให้ได้ภาพที่มีความเรียบเนียนยิ่งขึ้น

5.3.4 ทำโปรแกรมให้เป็นแอพพลิเคชั่นเพื่อให้ผู้ใช้สามารถใช้งานได้โดยไม่ต้องมีโปรแกรม MATLAB

เอกสารอ้างอิง

[1] Rafael C. Gonzalez and Richard E. Woods. **Digital Image Processing.** 2nd Edition, New Jersey : Prentice-Hall, Inc. 2002.

[2] Adrian Low. **Introductory Computer Vision And Image Processing.** International Edition. McGraw-Hill Book Company. 1991.

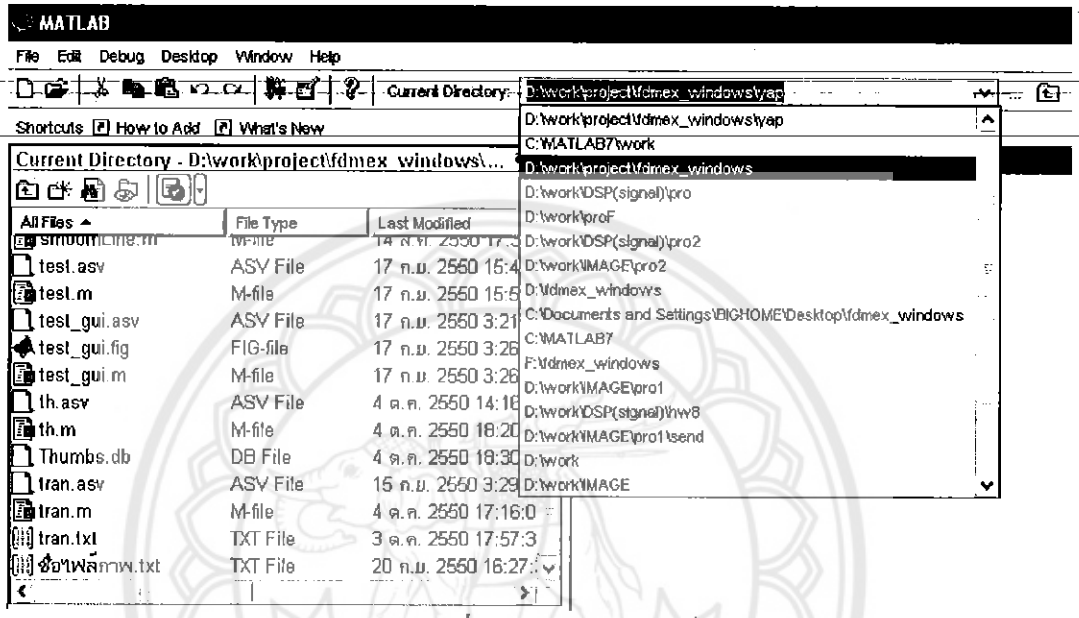
[3] W. Kienzle, G. Bakir, M. Franz and B. Scholkopf. **Face Detection - Efficient and Rank Deficient.** page. 673-680. 2005.

[4] W. Kienzle, G. Bakir, M. Franz and B. Scholkopf. "Face Detection." [Online]. Available : http://www.kyb.mpg.de/bs/people/kienzle/facedemo/facedemo.htm#fdlib. 2005

ภาคผนวก ก.

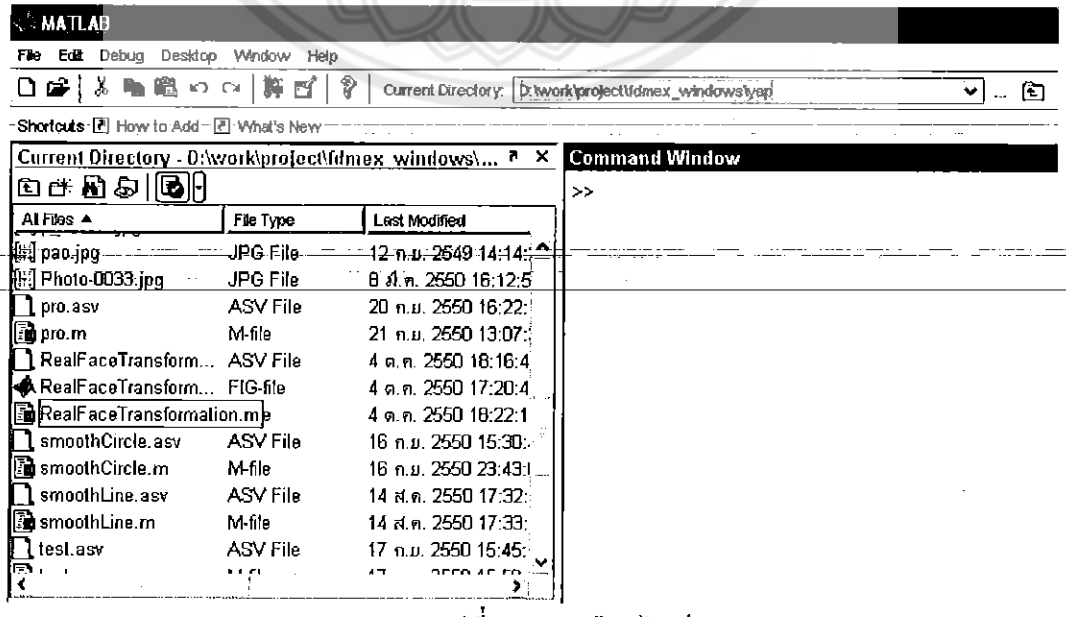
ตัวอย่างโปรแกรม

1. เลือกไดเรคทอรี่ที่บันทึกไฟล์ GUI



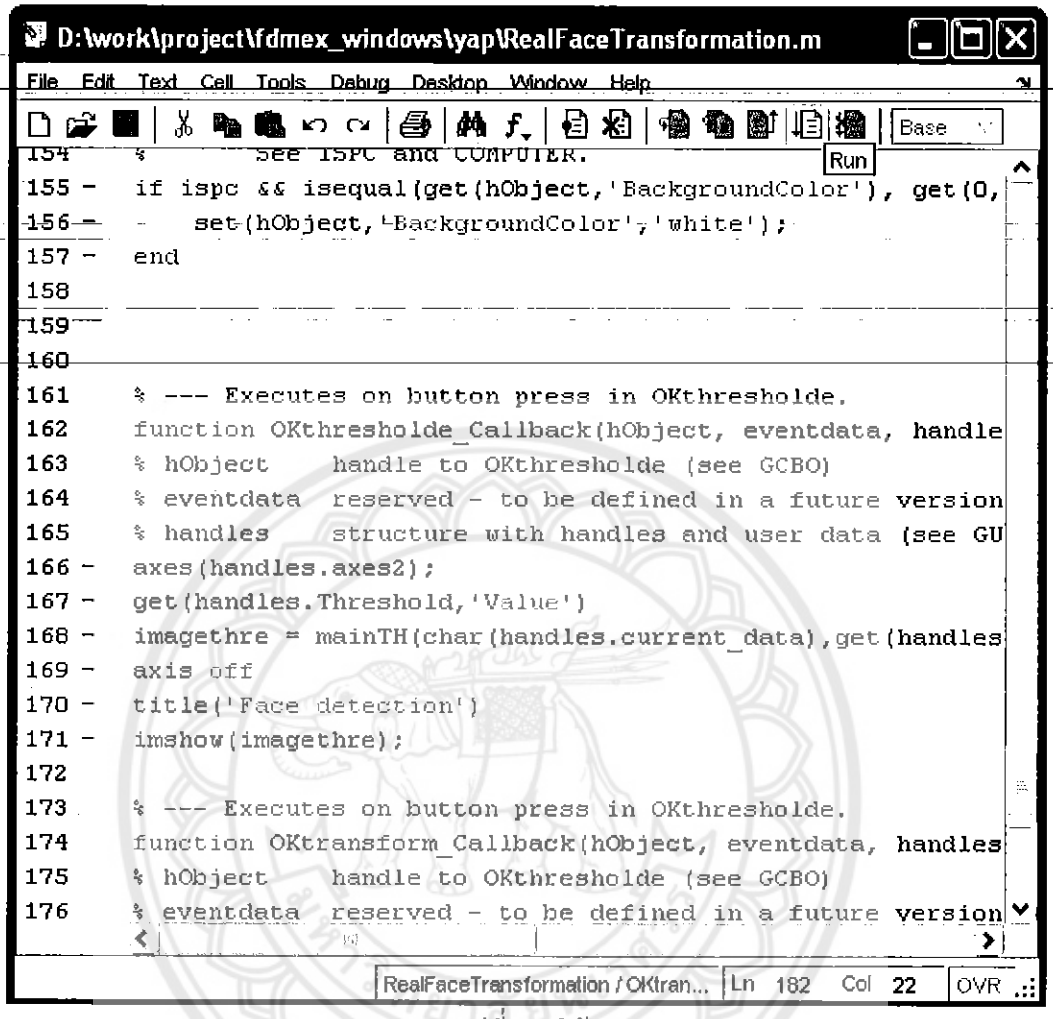
รูปที่ 1.1 เลือกไดเรคทอรี่

2. เลือกไฟล์ GUI ขึ้นมาเพื่อทำการรันโปรแกรม



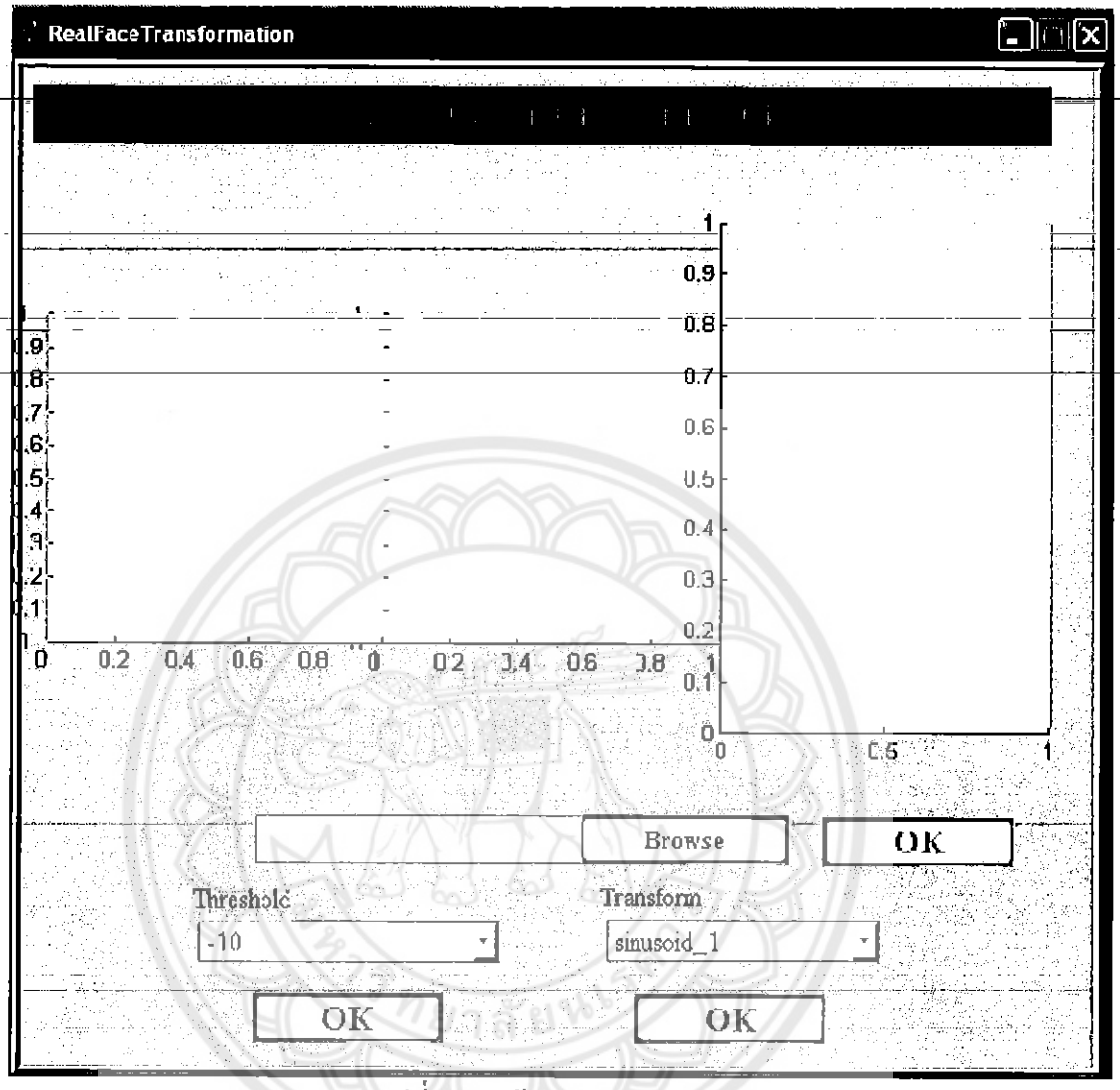
รูปที่ 1.2 การเลือกไฟล์

3. เมื่อเปิดไฟล์ออกมา ให้กด (run) เพื่อทำการรัน GUI



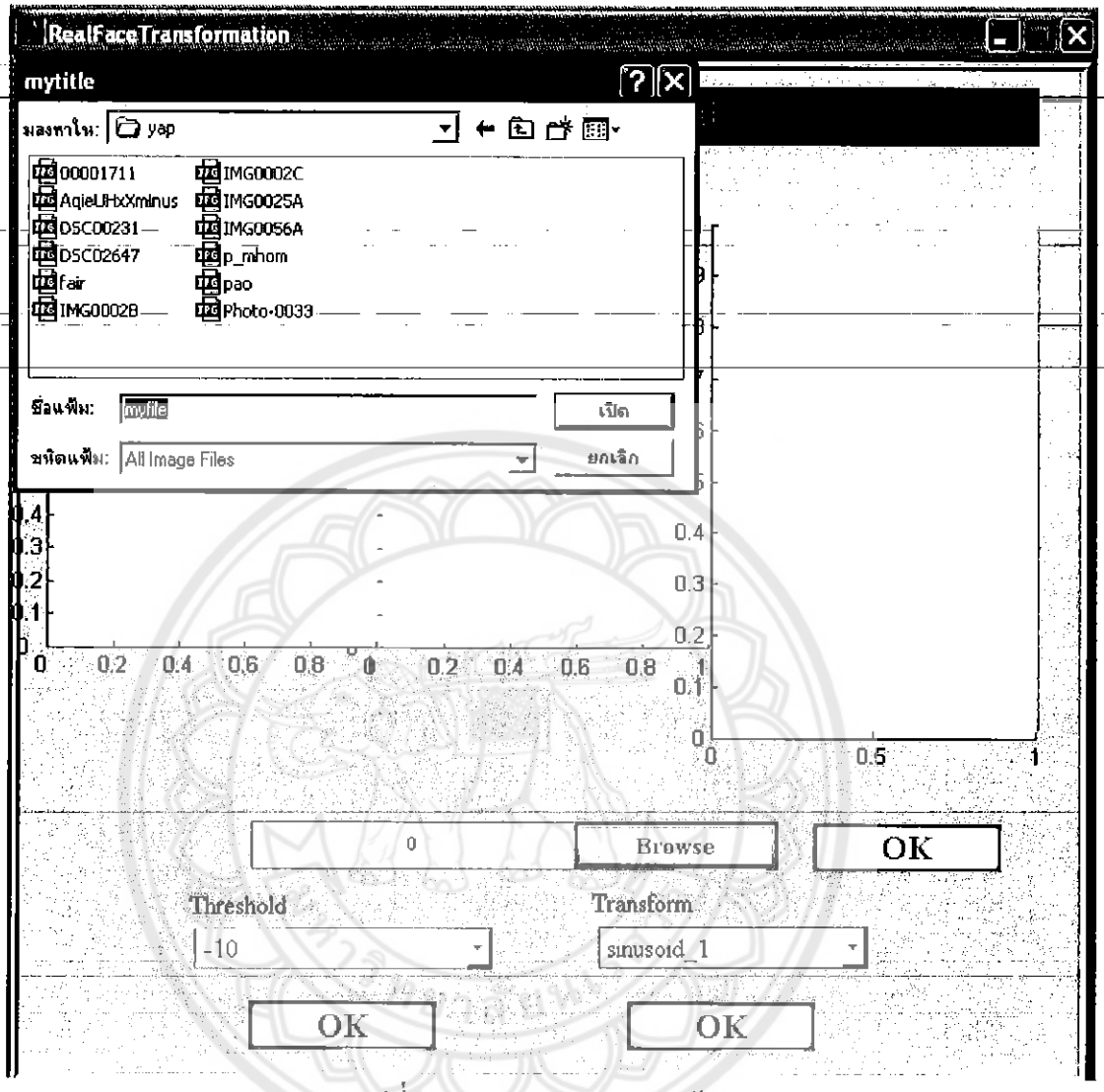
รูปที่ 1.3 โค้ด GUI

4. หน้าแรกของโปรแกรม GUI



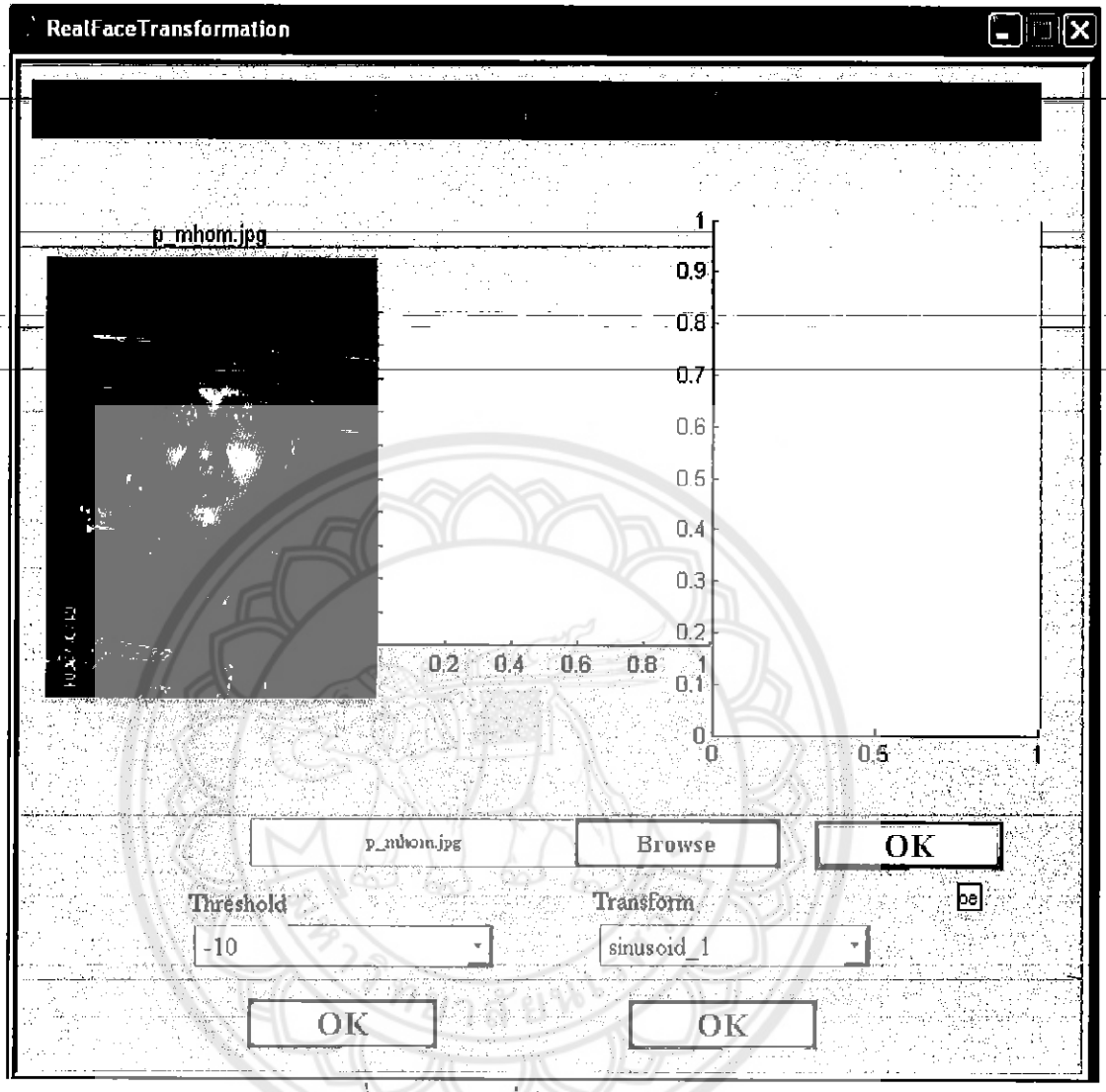
รูปที่ 1.4 หน้าแรกของโปรแกรม

5. คลิก browse เพื่อเลือกภาพที่ต้องการทำการแปลงภาพ คลิกเปิด จากนั้นคลิก ok



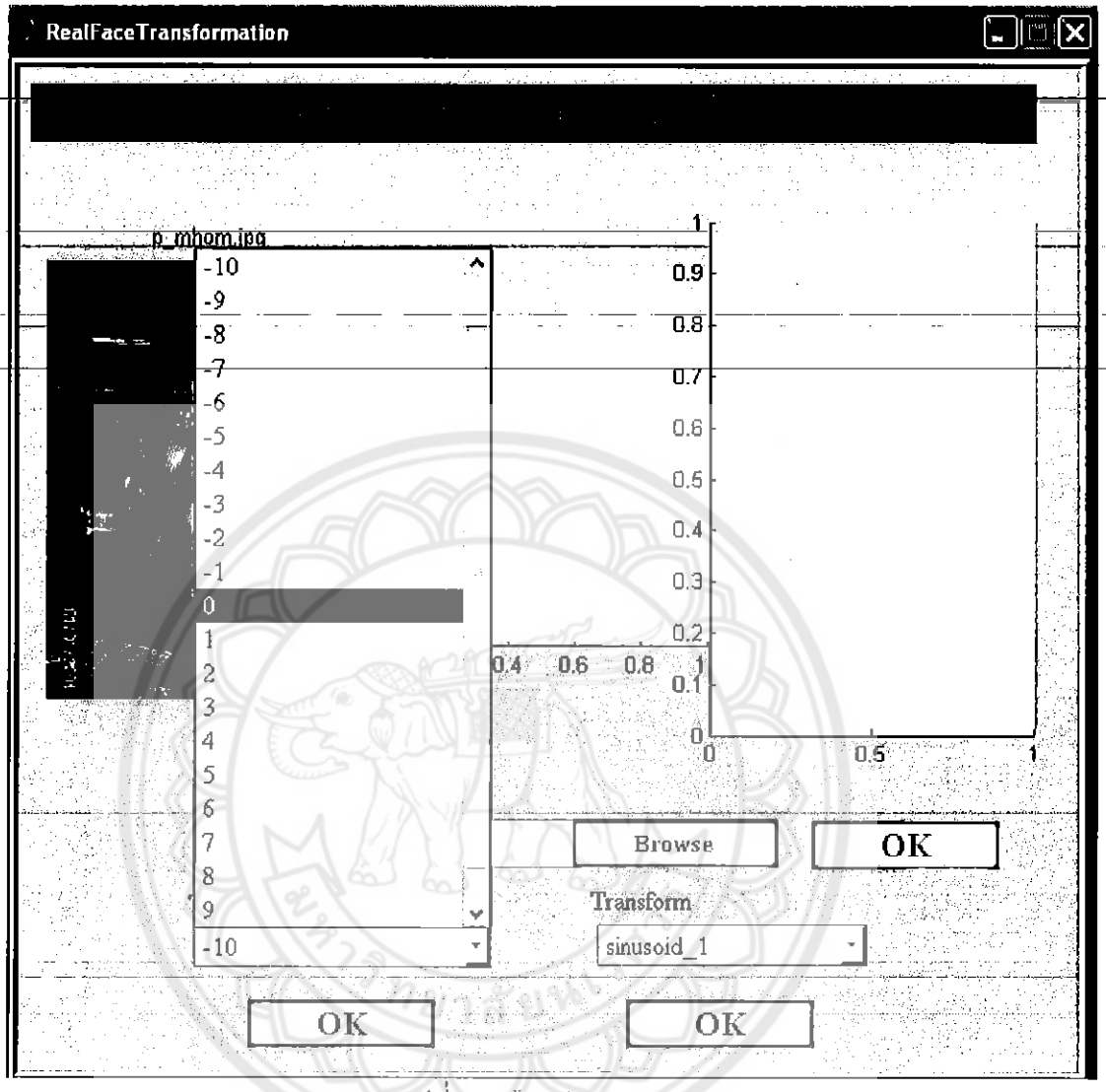
รูปที่ 1.5 การ browse ภาพมาใช้งาน

6. แสดงรูปภาพที่ต้องการแปลงภาพ



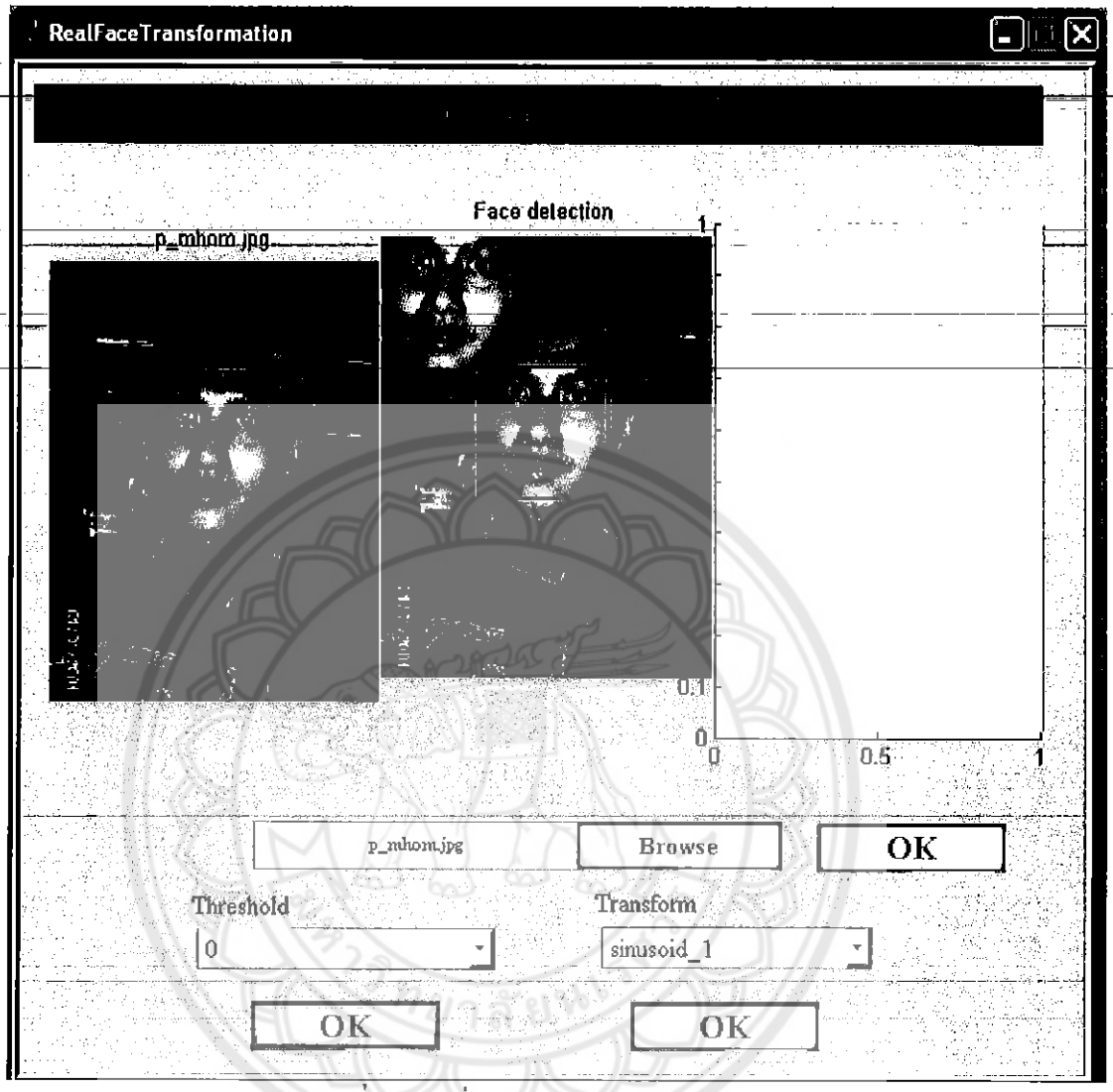
รูปที่ 1.6 รูปภาพที่ต้องการนำมาแปลงภาพ

7. เลือกค่า Threshold ระหว่าง -10 ถึง 10 เพื่อให้จับใบหน้าได้ จากนั้นคลิก OK



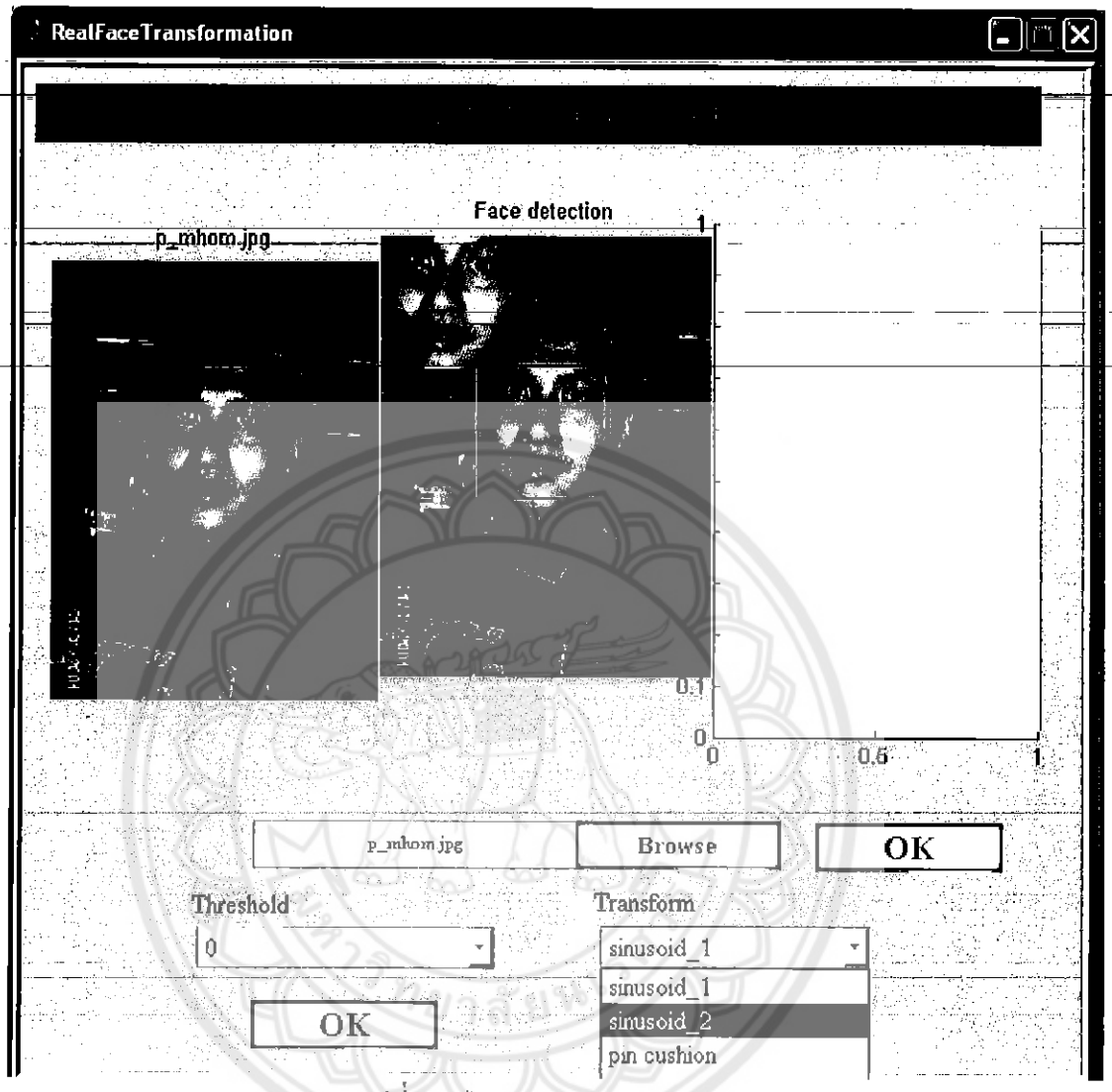
รูปที่ 1.7 เลือกค่า Thershold

8. แสดงผลที่ได้จากการเลือกค่า Threshold



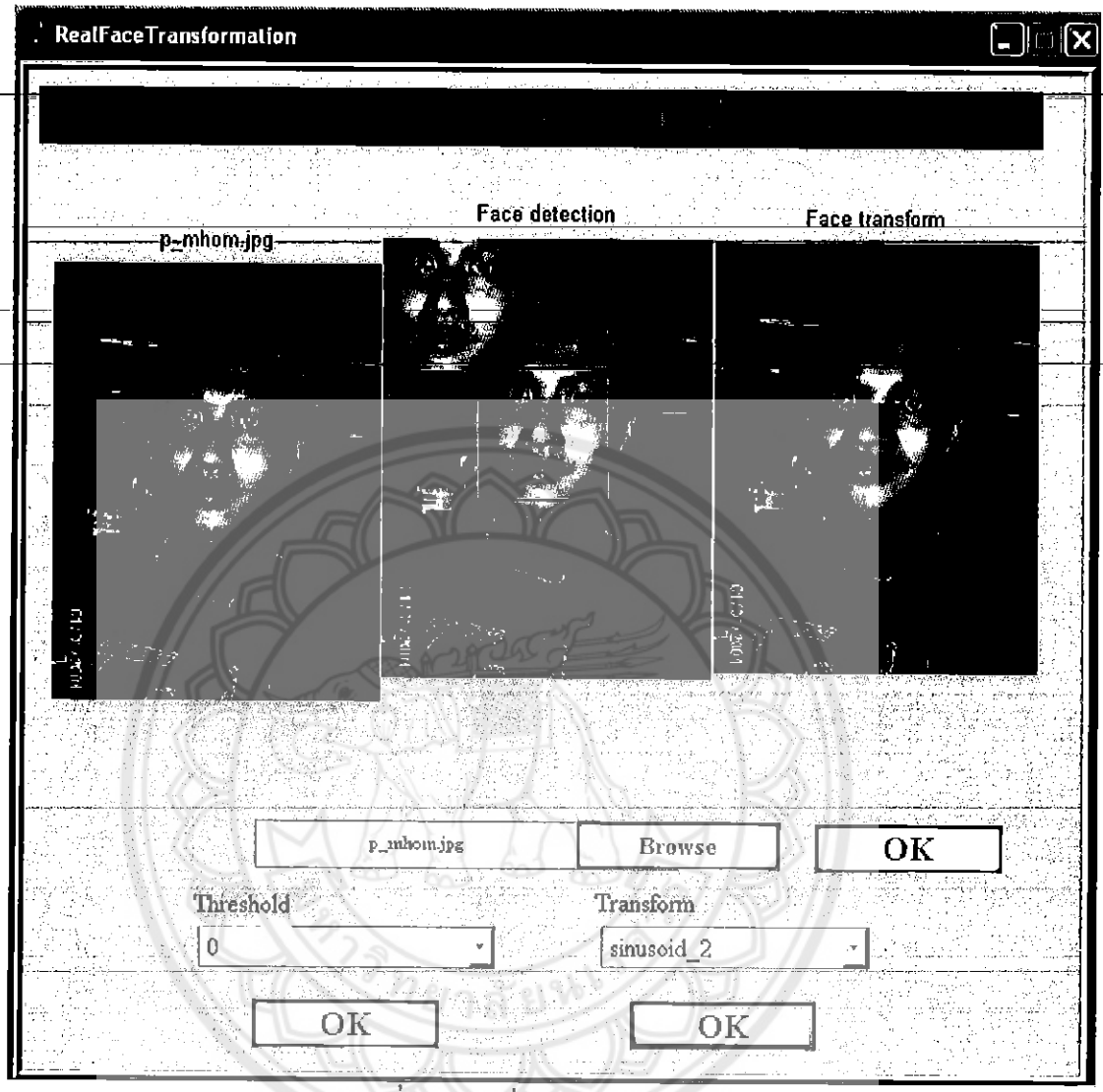
รูปที่ 1.8 รูปที่ได้จากการเลือกค่า Threshold

9. เลือกรูปแบบการแปลงภาพ จากนั้นคลิก OK



รูปที่ 1.9เลือกรูปแบบการแปลงภาพ

10. แสดงผลลัพธ์ที่ได้จากการแปลงภาพ



รูปที่ 1.10 รูปที่ได้จากการแปลงภาพ

ภาคผนวก ข.

ตัวอย่างโค้ดโปรแกรม

ไฟล์ mainTR. m

```
function S2 = mainTR(ml, imagethre, intr, inte)

% circle face
I = imread(ml) ;
x = rgb2gray(I) ;
threshold = inte – 11 ;
imagesc(x) ; hold on ; colormap gray ;
s = fdmex(x', threshold) ;
o = tran(intr, imagethre) ;

C = 0 ;
[mm nn oo] = size(C) ;
F = zeros(mm, nn, oo) ;
[fm fn fo] = size(F) ;
center_row = floor(mm/2) + 1 ;
center_col = floor(nn/2) + 1 ;

sz = size(s) ;
i = sz(1) ;
d = s(i, 3) ;
r = d/2 ;
for k = 1: mm,
        for l = 1: nn,
                a = k - center_row ;
                b = l - center_col ;
                if sqrt(a*a + b*b) <= r ;
                        F(k, l, : ) = C(k, l, : ) ;
```

```
                end
            end
        end
```

```
        left = s(i, 1) - s(i, 3)/2 ;
        bottom = s(i, 2) - s(i, 3)/2 ;
        width = s(i,3);
        height = s(i,3);

        rmin = round(s(i, 2) - s(i, 3)/2) ;
        rmax = round(s(i, 2) - s(i, 3)/2 + s(i, 3)) ;
        cmin = round(s(i, 1) - s(i, 3)/2) ;
        cmax = round(s(i, 1) - s(i, 3)/2 + s(i, 3)) ;

        R = I ;
        RF = I ;
        [rfm rfn rfo] = size(RF) ;

        % Transform(cirle)
        RF([rmin : rmax], [cmin : cmax], : ) = F ;

        % TransformCir(delete black)
        RCF = RF ;
        for k = 1: rfm,
            for l = 1: rfn,
                if RF(k, l) <= 0 ;
                    RCF(k, l, : ) = I(k, l, : ) ;
                end
            end
        end
```

```
% Smooth(circle)
center = floor([(bottom + height + bottom)/2 (left + left + width)/2]) ;
S2 = smoothCircle(RCF, 3, center(1), center(2), r) ;
S2 = smoothCircle(S2, 5, center(1), center(2), r) ;
S2 = smoothCircle(S2, 3, center(1), center(2), r) ;
S2 = smoothCircle(S2, 3, center(1), center(2), r) ;
```

ไฟล์ mainTH. m

```
function imagethre = mainTH(image, int)

%input image
I = imread(image) ;
x = rgb2gray(I) ;
imagethre  = th(I, x, int - 1 1) ;
```

ไฟล์ TH. m

```
function I2 = th(I, x, a) ;
threshold = a ;
imagesc(x) ; hold on ; colormap gray ;
s = fdmex(x', threshold) ;

for I = 1 : size(s, 1)
        h = rectangle('Position', [s(i, 1) - s(i, 3)/2, s(i, 2) - s(i, 3)/2, s(i, 3), s(i, 3)], ...
        'EdgeColor', [1, 0, 0], 'linewidth', 2) ;
end
```

```
axis equal ;
axis off ;

%crop image
I2 = imcrop(I, [s(i, 1) - s(i, 3)/2, s(i, 2) - s(i, 3)/2, s(i, 3), s(i, 3)]) ;
```

ไฟล์ Tran. m

```
function o = tran(d, I2)
ansTD = lower(d) ;
switch ansTD

        case 1

        % locally varying with sinusoid_1(หดแนวตั้ง)
        BW1 = I2 ;
        imid = round(size(BW1, 2)/2) ;
        % Find index of middle element[nrows, ncols] = size(BW1) ;

        [nrows, ncols, ncolor] = size(I2*3) ;
        [xi, yi] = meshgrid(1 : ncols, 1: nrows) ;
        a1 = 3 ;          % Try varying the amplitude of the sinusoids. (หน้ากว้าง)
        a2 = 6 ;          % (หน้าสั้น)
        u = xi + a1*sin(pi*xi/imid) ;
        v = yi - a2*sin(pi*yi/imid) ;
        tmap_B = cat(3, u, v) ;
        resamp = makeresampler('linear', 'fill') ;
        BW1_sinusoid_1 = tformarray(BW1, [ ], resamp, [2 1], [1 2], [ ], tmap_B, .3) ;
        o =  BW1_sinusoid_1 ;

        case 2

        % locally varying with sinusoid_2(ยืดบริเวณตรงกลาง)
        BW2 = I2 ;
        imid = round(size(BW2, 2)/2) ;
        % Find index of middle element[nrows, ncols] = size(BW2) ;

        [nrows, ncols, ncolor] = size(I2*3) ;
        [xi, yi] = meshgrid(1 : ncols, 1 : nrows) ;
```

```
a1 = 6 ;            % Try varying the amplitude of the sinusoids. (หน้ากว้าง)
a2 = 3 ; % (หน้าสั้น)
u = xi + a1*sin(pi*xi/imid) ;
v = yi - a2*sin(pi*yi/imid) ;
tmap_B = cat(3, u, v) ;
resamp = makeresampler('linear', 'fill') ;
BW2_sinusoid_2 = tformarray(BW2, [ ], resamp, [2 1], [1 2], [ ], tmap_B, .3) ;
o = BW2_sinusoid_2 ;
```

```
case 3

% radial pin cushion distortion
BW3 = I2 ;
imid = round(size(BW3, 2)/2) ;
% Find index of middle element[nrows, ncols] = size(BW3) ;

[nrows, ncols, ncolor] = size(I2*3) ;
[xi, yi] = meshgrid(1 : ncols, 1 : nrows) ;
xt = xi( : ) - imid ;
yt = yi( : ) - imid ;
[theta, r] = cart2pol(xt, yt) ;
a = - .0005 ;       % Try varying the amplitude of the cubic term.
s = r + a*r.^2.5 ;
[ut, vt] = pol2cart(theta, s) ;
u = reshape(ut, size(xi)) + imid ;
v = reshape(vt, size(yi)) + imid ;
tmap_B = cat(3, u, v) ;
resamp = makeresampler('linear', 'fill') ;
BW3_pin = tformarray(BW3, [ ], resamp, [2 1], [1 2], [ ], tmap_B, .3) ;
o = BW3_pin ;
```

```
case 4

% piecewise linear
BW4 = I2 ;
imid = round(size(BW4, 2)/2) ;    % Find index of middle element
st_l = 0.6 ;                      % หดด้านซ้าย
st_r = 1.4 ;                      % ขยายด้านขวา
size_left = [size(BW4, 1) round(st_l*imid)] ;
BW4_left = BW4( : , 1 : imid, : ) ;
size_right = [size(BW4, 1) round(st_r*imid)] ;
BW4_right = BW4( : , imid + 1 : end, : ) ;
BW4_left_stretched = imresize(BW4_left, size_left) ;
BW4_right_stretched = imresize(BW4_right, size_right) ;
BW4_piecewiselinear = [BW4_left_stretched BW4_right_stretched] ;
o = BW4_piecewiselinear ;

otherwise
end
```

ไฟล์ SmootCircle.m

```
function S = smoothCircle(R, maskSize, centerRow, centerCol, radius)
R = double(R) ;
T = R ;
[centerRow centerCol] ;

Mask = 1/(maskSize^2)*ones(maskSize) ;
[sr sc flat] = size(T) ;
a = (maskSize - 1)/2 ;
for k = 1 : 1 : flat
        for i = 1 : 1 : sr
                for j = 1 : 1 : sc
                if abs(floor(sqrt((centerRow - i)^2 + (centerCol - j)^2)) - radius) <= 3 ;
```

```
                R(i, j, k) = sum(sum(mask. *T(i – a : i + a, j – a : j + a, k))) ;
                end
                end
            end
        end
        S = uint8(R) ;
```

ประวัติผู้เขียนโครงงาน

ชื่อ นางสาวบุรินทรา กณะโกมล

ภูมิลำเนา 310 ม.2 ต.บ้านคลอง อ.เมือง จ.พิษณุโลก

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนเฉลิมขวัญสตรี
- ปัจจุบันกำลังศึกษาอยู่ในระดับปริญญาตรีชั้นปีที่ 4 สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail : **panda_mimo@hotmail.com**

ชื่อ นางสาวพิชชาภรณ์ บ่อน้อย

ภูมิลำเนา 632 ถ.ราชดำเนิน1 ต.ในเมือง อ.เมือง จ.กำแพงเพชร

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนกำแพงเพชรพิทยาคม
- ปัจจุบันกำลังศึกษาอยู่ในระดับปริญญาตรีชั้นปีที่ 4 สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail : **pink_berry_23@hotmail.com**

ชื่อ นางสาววิชชุลดา เสนานุช

ภูมิลำเนา 78 ม. 3 ต.นาหอ อ.ด่านซ้าย จ.เลย

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนหล่มเก่าพิทยาคม
- ปัจจุบันกำลังศึกษาอยู่ในระดับปริญญาตรีชั้นปีที่ 4 สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail : **i_witchy19@hotmail.com**