การปรับปรุงรายละเอียดของขอบในภาพสี ด้วยการแปลงภาพผลลบที่ถูกทำให้เรียบ

Edge Enhancement In Colour Images Using Transform Of Subtracted Smoothing Image

อาโมทย์ สมบูรณ์แก้ว

Armote Somboonkaew

อาจารย์ที่ปรึกษา
รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์

Advisor
Assoc. Prof. Dr. Fusak Cheevasuvit

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2536

ISBN 974 - 621 - 094 - 7

EDGE ENHANCEMENT IN COLOUR IMAGES USING TRANSFORM OF SUBTRACTED SMOOTHING IMAGE

ARMOTE SOMBOONKAEW

ADVISOR

ASSOC. PROF. FUSAK CHEEVASUVIT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

GRADUATE SCHOOL

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

1993

ISBN 974 - 621 - 094 - 7

บทคัดย่อ

หัวข้อวิทยานิพนธ์	การปรับปรุงรายละเอียดของขอบในภาพสี ด้วยการแปลงภาพผลลบที่ถูกทำให้เรียบ
ชื่อนักศึกษา	นาย อาโมทย์ สมบูรณ์แก้ว
อาจารย์ที่ปรึกษา	รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์
ระดับการศึกษา	วิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า
ปีการศึกษา	2536

ภาพสีที่ได้จากการผสมภาพถ่ายดาวเทียมสามแบนด์ ถูกนำมาใช้อย่างกว้างขวาง ตัวอย่างเช่นใช้ในการตรวจสอบการลักลอบตัดไม้ทำลายป่า ดังนั้นการแปลความหมายภาพและการจำแนกข้อมูลภาพดังกล่าวจะมีประสิทธิภาพสูงก็ต่อเมื่อภาพนั้นได้รับการปรับปรุงรายละเอียดของขอบหรือการเน้นขอบต่าง ๆ ภายในภาพ วิธีการปรับปรุงรายละเอียดของขอบหรือการเน้นขอบภายในภาพสามารถทำได้โดยการนำเอาภาพที่ถูกทำให้เรียบไปลบออกจากภาพเดิม ก็จะได้ภาพผลลบที่เป็นภาพของขอบ ภาพที่มีการเน้นขอบจะได้จากการนำภาพผลลบนี้บวกกลับไปในภาพเดิม แต่เนื่องจากภายในภาพผลลบที่ได้นั้นจะประกอบไปด้วยขอบจริงที่มีค่าผลลบสูงและขอบเทียมซึ่งเกิดขึ้นในพื้นที่เอกพันธ์ที่มีค่าผลลบต่ำ โดยขอบเทียมเหล่านี้จะทำให้เกิดการเพิ่มค่าความสว่างตลอดทั้งภาพ ทำให้เกิดความสับสนและยุ่งยากในการแปลภาพ ดังนั้นในวิทยานิพนธ์นี้จึงได้เสนอวิธีการกำจัดขอบเทียมโดยการใช้ฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธ เพื่อทำการกำจัดขอบเทียมและขยายขอบจริง โดยขบวนการเน้นขอบภาพที่เสนอนี้จะถูกนำมาประยุกต์ใช้กับภาพถ่ายดาวเทียมทั้งสามแบนด์ ภาพทั้งสามหลังการประมวลผลด้วยวิธีการดังกล่าวจะถูกนำมาผสมกันเป็นภาพสีที่มีการเน้นขอบ

ABSTRACT

THESIS TITLE	edge enhancement in colour images using transform of subtracted smoothing image
STUDENT	Armote Somboonkaew
THESIS ADVISOR	Assoc. Prof. Dr. Fusak Cheevasuvit
LEVEL OF STUDY	Master of Engineering in Electrical Engineering
ACADEMIC YEAR	1993

A colour image derived by three multispectral image bands is used in wide varity of applications, such as detection of illegal deforestration. For high efficiency of image interpretation and classification, the colour image should be enhanced before using. The edge enhancement in image will be performed by subtracting the smoothed image from the original image. The result of subtracted image will consist of both real edges and also spurious edges which appear in the homogeneous regions. This effect appears since the spurious edge will increase the brightness in all image. The spurious edges will cause some confusions in the image interpretation. Therefore, to improve the contrast in the edge enhancement, so the inverse Butterworth transform function will be used for depressing these spurious edges and magnified the real edges. The proposed edge enhancement procedure is applied to the three different bands of satellite image. Thus, the edge enhancement in colour image can be performed by these three processed images.

กิตติกรรมประกาศ

ขอขอบพระคุณ รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์ เป็นอย่างสูงที่ได้ให้การประสิทธิ์ประสาทวิชา ให้คำแนะนำปรึกษาในเรื่องต่าง ๆ แก่ผู้เขียน ขอขอบพระคุณ รศ.ดร. กอบชัย เดชหาญ ที่ได้ช่วยเหลือ และให้คำแนะนำต่าง ๆ ขอขอบพระคุณท่านกรรมการสอบทุกท่านที่ได้สละเวลาอันมีค่า ขอขอบพระคุณมูลนิธิเพื่อการศึกษาคอมพิวเตอร์และการสื่อสารที่มอบทุนการศึกษาให้สามารถทำการศึกษาวิจัยได้เต็มเวลา ขอขอบพระคุณคุณพ่อคุณแม่และเพื่อน ๆ ที่ได้ช่วยเหลือและเป็นกำลังใจ ในการทำวิทยานิพนธ์นี้จนสำเร็จลุล่วงได้ด้วยดี

สารบัญ

หน้า

หน้า

สารบัญรูป

รูปที่ **หน้า**

รูปที่ หน้า

รูปที่ **หน้า**

สารบัญตาราง

ตารางที่ **หน้า**

บทที่ 1
บทนำ

1.1 คำนำ

ในปัจจุบันนี้ไมโครคอมพิวเตอร์ส่วนบุคคลได้มีการพัฒนาฮาร์ดแวร์เพื่อตอบสนองความต้องการต่าง ๆ ของซอฟต์แวร์ทั้งในด้านเพิ่มความเร็ว ประสิทธิภาพและความสามารถ ซึ่งนับวันความสามารถและความเข้ากันได้ดีของทั้งคู่ก็จะยิ่งสูงขึ้นไปเรื่อย ๆ ในการแสดงภาพสีก็สามารถแสดงได้อย่างง่ายดาย ทำให้ได้ประโยชน์อย่างมากสำหรับการประมวลผลภาพทางด้านการสำรวจข้อมูลระยะไกล ซึ่งซอฟต์แวร์ในปัจจุบันได้มุ่งไปที่โปรแกรมประยุกต์ต่าง ๆ ที่ทำงานอยู่บนวินโดว์กันมาก เพราะการทำงานบนวินโดว์มีการจัดการกับระบบฮาร์ดแวร์และเตรียมอัตถประโยชน์ต่าง ๆ ไว้ให้ใช้อย่างครบครัน ทำให้เรียกใช้ฮาร์ดแวร์ได้อย่างง่ายดายและมีประสิทธิภาพ ตัวอย่างเช่น มีการจัดการบริหารหน่วยความจำให้ จึงสามารถเรียกใช้หน่วยความจำได้ที่มีได้ทั้งหมด และมีการใช้หน่วยความจำเวอร์ชวล (virtual memory) ที่ใช้ฮาร์ดดิสก์มาทำเป็นหน่วยความจำเสริมได้อีก ทำให้สามารถจองแอร์เรย์ขนาดใหญ่สำหรับใช้ในการเก็บข้อมูลภาพต่าง ๆ ได้อย่างต่อเนื่องเป็นแอร์เรย์เดียวซึ่งเป็นความสะดวกอย่างมาก ทั้งนี้เพราะหน่วยความจำนี้เป็นสิ่งสำคัญสำหรับการประมวลผลภาพถ่ายดาวเทียมที่ต้องใช้เนื้อที่เป็นจำนวนมาก สำหรับการประมวลผลบางอย่างที่ต้องการความเร็ว และสำหรับวิธีการบางอย่างที่ขบวนการคำนวณไม่สามารถใช้การอ่านข้อมูลออกมาครั้งละเส้นภาพหรือครั้งละจุดภาพได้ ต้องให้ข้อมูลทั้งหมดอยู่ในหน่วยความจำ

ในวิทยานิพนธ์นี้ได้เสนอวิธีการประยุกต์การใช้ไมโครคอมพิวเตอร์ในการปรับปรุงคุณภาพของภาพสี โดยเฉพาะขอบของภาพและครอบคลุมถึงคอนทราสของภาพด้วย ทั้งนี้เพราะภาพสีของภาพถ่ายดาวเทียมนั้นได้ถูกนำมาใช้งานกันอย่างกว้างขวาง ในการแปลความหมายของภาพ (image interpretation) และการจำแนกข้อมูลของภาพ (image classification) สำหรับการสำรวจทรัพยากรธรรมชาติ ซึ่งความถูกต้องแม่นยำในการแปลความหมายและการจำแนกข้อมูลภาพนั้นจะขึ้นอยู่กับรายละเอียดของข้อมูลต่าง ๆ ภายในภาพ

แต่โดยทั่วไปแล้วภาพถ่ายดาวเทียมมักจะมีปัญหาเรื่องรายละเอียดของภาพและคอนทราสของภาพ ทั้งนี้เพราะตัวตรวจจับ (sensor) ระบบภาพของดาวเทียมมักถูกรบกวนจากการแตกกระจายของคลื่นแม่เหล็กไฟฟ้าในชั้นบรรยากาศ (Atmospheric scatterring) โดยชั้นบรรยากาศแอทโมส์เฟียทำตัวคล้ายตัวกรองความถี่ต่ำผ่าน ทำให้ภาพที่ได้มีความแตกต่าง (contrast) ของข้อมูลในภาพต่ำ และขาดความคมชัดเนื่องจากการทำลายส่วนประกอบที่เป็นความถี่สูง (high spatial frequency) ทำให้ส่วนที่เป็นลายเส้น ขอบภาพ และรายละเอียดต่าง ๆ ถูกทำลายลงไป การแปลความหมายและการจำแนกข้อมูลภาพดังกล่าว ขาดความถูกต้องแม่นยำ ก่อนนำไปใช้ในการประมวลผลอื่น ๆ จึงต้องทำการปรับปรุงแก้ไขรายละเอียด

ต่าง ๆ ให้ดีขึ้นเช่นเดียวกับกรณีภาพถ่ายทั่ว ๆ ไป อีกทั้งภาพถ่ายดาวเทียมก็ยังมีความละเอียดและลายเส้นในลักษณะต่าง ๆ ที่ซับซ้อนมากกว่าภาพทั่วไปมาก ดังนั้นการแก้ปรากฏการณ์ตัวกรองความถี่ต่ำผ่านจึงมีความจำเป็นอย่างมาก

ในการปรับปรุงคุณภาพของภาพถ่ายดาวเทียมสามารถทำได้โดยการเพิ่มรายละเอียดของขอบภายในภาพ อย่างไรก็ตามรายละเอียดของขอบที่ได้บางครั้งอาจจะมีขอบเทียมที่เกิดขึ้นในพื้นที่เอกพันธ์ (homogeneous region) ปรากฏเข้ามาปะปนอยู่ด้วย ดังในวิทยานิพนธ์นี้จึงได้นำเสนอแนวความคิดใหม่ในการปรับปรุงรายละเอียดของขอบในภาพ โดยมีการกำจัดขอบเทียมด้วยฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธ ภาพที่ได้จากวิธีดังกล่าวเมื่อนำมาใช้ในการแปลความหมายของภาพและการจำแนกข้อมูลภาพแล้ว จะส่งผลให้การแปลความหมายและการจำแนกข้อมูลมีความถูกต้องแม่นยำสูง

1.2 ขอบเขตการวิจัย

ในส่วนของการวิจัยได้เสนอเทคนิคการปรับปรุงภาพวิธีต่าง ๆ เพื่อที่จะสามารถปรับปรุงคุณภาพของภาพให้มีคุณภาพดีมากที่สุด เพื่อที่จะสามารถนำไปใช้ในการแปลความหมายภาพและการจำแนกข้อมูลภาพได้อย่างมีประสิทธิภาพสูง ลดภาระหนักต่าง ๆ ที่จะนำภาพไปใช้ประโยชน์โดยตรง รายละเอียดของการวิจัยวิทยานิพนธ์ฉบับนี้ จะแบ่งย่อยออกเป็น 6 บท โดยที่แต่ละบทมีหัวข้อและเนื้อหาดังนี้

บทที่ 1 เป็นบทนำ

บทที่ 2 เป็นการแสดงภาพถ่ายดาวเทียมสีบนไมโครคอมพิวเตอร์ รูปแบบข้อมูลภาพถ่ายดาวเทียมในการสำรวจข้อมูลระยะไกล การแปลงรูปแบบข้อมูลภาพในรูปแบบแอสกี้ไปเป็นรูปแบบบิทแมพเพื่อให้สามารถเข้ากับมาตรฐานของวินโดว์จะได้สามารถนำไปใช้กับโปรแกรมประยุกต์ต่าง ๆ ได้ และความรู้ทั่วไปที่ใช้ในการปรับปรุงภาพให้ดีขึ้น (image enhancement) เช่น วิธีการทำคอนโวลูชันในสปาร์เชียนโดเมนต์ การใช้วิธีการเปิดตารางในการเพิ่มความเร็วสำหรับการแปลง และ การเตรียมภาพโดยปรับปรุงคอนทราส ของภาพเพื่อนำไปใช้เป็นภาพต้นแบบในบทอื่น ๆ ต่อไป

บทที่ 3 เป็นการปรับปรุงรายละเอียดของขอบภาพ ได้แก่ การเพิ่มความคมชัดของขอบภาพด้วยการเสริมข้อมูลเกรเดียนท์โดยวิธีของ Shettigara และ Odins การเพิ่มความคมชัดของขอบภาพด้วยการเสริมข้อมูลเกรเดียนท์โดยใช้เทมเพล็ทแบบ 12 ทิศทาง

บทที่ 4 เป็นการปรับปรุงรายละเอียดของขอบภาพโดยใช้ภาพผลลบที่เกิดจากการทำให้เรียบ และวิธีการปรับปรุงที่ใช้ตัวกรองแทนขบวนการทั้งหมด

บทที่ 5 เป็นการปรับปรุงรายละเอียดของขอบภาพโดยใช้ฟังก์ชันการแปลงในการกำจัดขอบเทียมเพื่อเพิ่มความคมชัดให้กับขอบภาพ

บทที่ 6 เป็นบทสรุปเปรียบเทียบผลของการปรับปรุงขอบของภาพที่ได้จากการปรับปรุงรายละเอียดของขอบภาพโดยใช้การแปลงภาพผลลบที่เกิดจากการทำให้เรียบกับวิธีการต่าง ๆและวิจารณ์ถึงข้อดีข้อเสียพร้อมทั้งเสนอแนะแนวทางการวิจัยที่สามารถจะพัฒนาต่อไปได้

บทที่ 2

การแสดงภาพสี รูปแบบข้อมูลภาพ และความรู้ทั่วไป ในการปรับปรุงภาพ

Display of color image, images files format and basic knowledge of contrast enhancement

2.1 คำนำ

การประมวลผลภาพสีในการสำรวจข้อมูลระยะไกล (remote sensing) นั้นมีข้อได้เปรียบกว่าการใช้ภาพขาวดำ เพราะในการวิเคราะห์ภาพอัตโนมัตินั้น สีจะเป็นตัวอธิบายที่มีความสามารถสูงในการชี้ (identification) และแยกแยะวัตถุ (extraction) ที่สนใจออกมาจากฉากเบื้องหลังได้ง่าย และอีกทั้งในการวิเคราะห์ภาพนั้นถ้าไม่ได้ทำเป็นแบบอัตโนมัติ แต่มนุษย์เป็นผู้กระทำการตัดสินใจเอง การกระตุ้นของสีที่มนุษย์สามารถมองเห็นได้นั้น มนุษย์สามารถมองเห็นและแยกแยะเชดสี (shade) ได้เป็นพัน ๆ ระดับ ในขณะที่เมื่อเปรียบเทียบกับภาพระดับสีเทา (gray scale) ที่มนุษย์สามารถมองเห็นและแยกแยะได้เพียงยี่สิบกว่าเชดสีเท่านั้น ดังนั้นในการวิเคราะห์โดยใช้ภาพสีจึงมีประสิทธิภาพอย่างมาก

2.2 รูปจำลองของสี (color model)

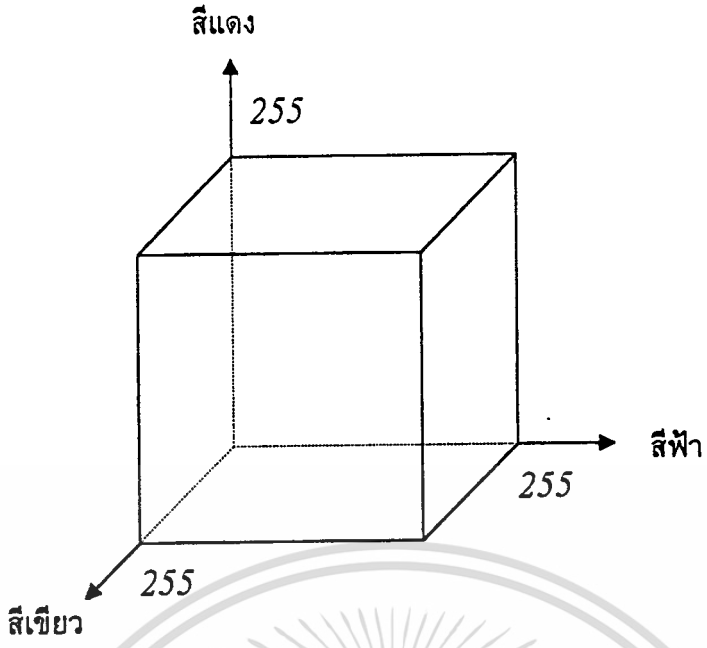
รูปจำลองของสีมีลักษณะเป็นรูประบบพิกัดแบบสามมิติ แต่ละสีเกิดจากการผสมระหว่างแกนสีทั้งสาม โดยรูปแบบจำลองของสีในปัจจุบันได้หันเหไปตามฮาร์ดแวร์ เช่น จอภาพสี เครื่องพิมพ์ และการประยุกต์ใช้งานต่าง ๆ รูปแบบจำลองของสีที่นิยมใช้กันมากที่สุดสำหรับมอนิเตอร์สีและกล้องวิดีโอ คือ RGB model (RED , GREEN , BLUE) สำหรับรูปจำลองของสีแบบ CMY model (CYAN , MAGENTA , YELLOW) นิยมใช้กับเครื่องพิมพ์สี ส่วนรูปแบบการส่งแบบ YIQ model เป็นมาตรฐานสำหรับการส่งกระจายภาพโทรทัศน์สี โดย Y คือ luminance และ I กับ Q คือ ส่วนของ chromatic ที่เป็น inphase และ quadrature ตามลำดับ ส่วนรูปจำลองกลางที่ใช้ในการเปลี่ยนไปเปลี่ยนมาระหว่างรูปจำลองสีแบบต่าง ๆ คือระบบรูปจำลองแบบ HSI (hue , saturation , intensity) และรูปจำลองแบบ HSV (hue , saturation , value)

2.2.1 รูปแบบจำลองสีแบบอาร์จีบี (RGB color model)

ลำของแสงเมื่อผ่านแก้วปริซึมแล้ว ลำของแสงที่ออกมาไม่ได้มีสีขาว แต่ประกอบไปด้วยสเปกตรัมที่ต่อเนื่องของสี มีย่านจากสีม่วง (violet) ไปจนถึงแดง (red) โดยพื้นฐานแล้วสีที่มนุษย์สามารถมองเห็นวัตถุเกิดมาจากธรรมชาติของแสงในการสะท้อนมาจากวัตถุ วัตถุที่สะท้อนแสงทั้งหมดทุกแสงจะมองเห็นเป็นสีขาว วัตถุที่สะท้อนบางส่วนของสเปคตรัมจะให้บางเชดของสี ตัวอย่างเช่น วัตถุสีเขียวสะท้อนช่วงที่มีความยาวคลื่น 500 ถึง 570 nm โดยดูดซับเอาความยาวคลื่นที่เหลือนอกนั้นไว้ทั้งหมด ทำให้เราสามารถมองเห็นวัตถุชิ้นนั้นเป็นสีเขียวนั่นเอง ในย่านที่มนุษย์สามารถมองเห็นได้นั้นความยาวคลื่นของแสงจะเป็นตัวกำหนดสี ที่อยู่ในหน่วยนาโนเมตร ในระบบรูปแบบจำลองแบบอาร์จีบี สีแดง สีเขียว และสีฟ้าจะถูกเรียกว่าสีปฐมภูมิ (primary color) แสงที่มีความยาวคลื่นประมาณ 430 นาโนเมตรคือสีฟ้า ความยาวคลื่นประมาณ 550 นาโนเมตรคือสีเขียว และที่มีความยาวคลื่นประมาณ 700 นาโนเมตรคือสีแดง ในการผสมของแสงสีแดง เขียว และฟ้า เข้าด้วยกันในหลาย ๆ รูปแบบ ทำให้สามารถได้สีต่าง ๆ อื่น ๆ ออกมามากมาย ถึงแม้จะมีสีอีกมากที่ไม่สามารถสร้างโดยวิธีการนี้ได้ก็ตาม แต่วิธีการนี้ก็มีประสิทธิภาพดีพอเพียงที่จะใช้ในการแสดงผลภาพ และมีความนิยมใช้กันอย่างมาก เช่น การแสดงผลในมอนิเตอร์สีสำหรับคอมพิวเตอร์ทั่วไป เป็นต้น

2.2.2 การแสดงภาพสีบนเครื่องไมโครคอมพิวเตอร์ สำหรับการสำรวจข้อมูลระยะไกล (Display of color image for remote sensing on microcomputer)

ข้อมูลภาพสีของภาพถ่ายดาวเทียมแบบหลายความถี่ (multispectrum image) มีลักษณะเป็นข้อมูลภาพสามแบนด์ แต่ละจุดภาพที่แสดงผลเกิดจากการรวมกันระหว่างข้อมูลภาพทั้งสามแบนด์ แต่ละแบนด์เป็นข้อมูลขนาด 8 บิท ดังนั้นข้อมูลสีหนึ่งจุดจะมีขนาด 24 บิท หรือเรียกว่าเป็นข้อมูล 24 บิทต่อหนึ่งจุดภาพ นั่นคือแต่ละจุดภาพมีความเป็นไปได้ที่จะเกิดข้อมูลที่แตกต่างถึง 2^{24} ระดับคือได้เท่ากับ 16.7 ล้านระดับที่แตกต่างกัน ซึ่งในการแสดงผลภาพสีถ้านำข้อมูลภาพทั้งสามแบนด์มาทำการพล็อตเป็นฮีสโตแกรมสามมิติ โดยที่แต่ละแกนคือค่าระดับสีเทาของข้อมูลภาพที่มีค่าได้ตั้งแต่ 0-255 ของแต่ละแบนด์ ทำให้เกิดบล็อกสี่เหลี่ยมเล็ก ๆ ภายในฮีสโตแกรมดังกล่าวจำนวน 256×256×256 บล็อก โดยที่แต่ละบล็อกก็คือข้อมูลของสีของเชดสีหนึ่งสีที่เกิดขึ้นได้ในแต่ละจุดภาพนั่นเอง ฮีสโตแกรมแสดงดังรูปที่ 2.1

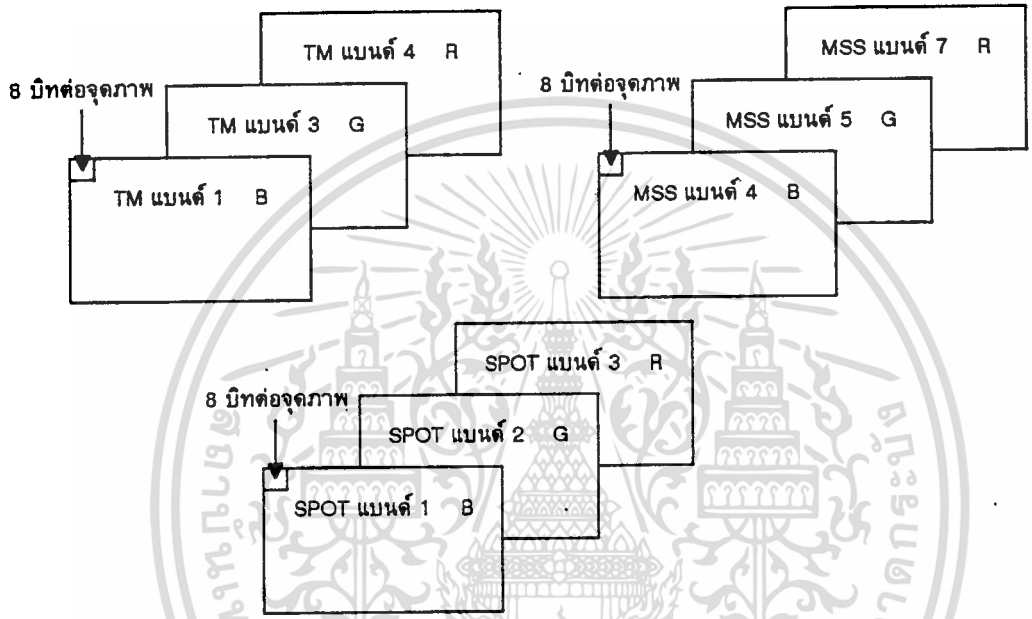


รูปที่ 2.1 แสดงไดอะแกรมของฮีสโตแกรมสามมิติของการแสดงภาพสีโดยใช้รูปแบบจำลองสีแบบอาร์จีบีซึ่งมีขนาด 24 บิทต่อหนึ่งจุดภาพ

การแสดงผลภาพสีนั้นในส่วนของฮาร์ดแวร์ระบบแสดงผล (graphic adapter) เป็นตัวกำหนดประสิทธิภาพในการแสดงจำนวนสี โดยถ้าเป็นระบบ CGA (color graphic adapter) จะมีประสิทธิภาพต่ำเกินไป ทั้งในแง่ของความละเอียดของภาพ และจำนวนของสีที่นำมาใช้งาน คือจะมีความละเอียด 640X200 จุดสำหรับการแสดง 2 สีพร้อม ๆ กัน และจะมีความละเอียด 320X200 สำหรับการแสดง 4 สีพร้อม ๆ กัน จึงไม่เหมาะสมที่จะนำมาใช้ในการประมวลผลภาพถ่ายดาวเทียม ถ้าเป็นระบบ EGA (enhance graphic adapter) สามารถแสดงภาพโดยความละเอียดสูงสุด 640X350 โดยแสดงได้ 16 สีพร้อม ๆ กันจากจำนวนสีทั้งหมด 64 สี เป็นระบบที่พอจะนำมาประยุกต์ใช้ในการประมวลผลภาพถ่ายดาวเทียมได้บ้างในบางกรณี แต่ถ้าต้องการความละเอียดหรือจำนวนสีมากขึ้นต้องข้ามไปใช้ระบบการแสดงผลแบบ VGA (video graphic adapter) ซึ่งสามารถแสดงผลได้ 256 สี โดยมีความละเอียดสูงสุด 1024X 768 จุดภาพ และโดยการใช้การแสดงสีแบบสีเท็จ (false color) เพื่อลดจำนวนสีให้กับระบบ EGA 16 สีหรือ VGA 256 สีก็จะทำให้สามารถแสดงภาพสีในรูปแบบจำลองแบบอาร์จีบีได้ ดังเอกสารอ้างอิง [8] แต่สีสันที่ได้ก็ยังไม่ได้สีสันที่เป็นจริง จึงยังไม่สามารถประยุกต์ใช้กับงานทุกอย่างได้ ทำให้การนำมาใช้ประมวลผลภาพสีตกเป็นภาระกับเครื่องที่สามารถแสดงผลแบบ RGB 24 บิท ซึ่งยังไม่เป็นที่แพร่หลาย เช่นเครื่องไมโครคอมพิวเตอร์ NEC 9801 ที่ใช้ฮาร์ดแวร์ของระบบแสดงผลที่ชื่อว่า superframe เป็นต้น แต่ก็มีราคาแพง

ในปัจจุบัน เครื่องไมโครคอมพิวเตอร์มีความสามารถเพิ่มมากขึ้น ทั้งในด้านความเร็วและความสามารถในการแสดงสี จอภาพที่เป็นมาตรฐานที่ใช้งานกันแพร่หลายต่อ ๆ ไปในอนาคตจะเป็นจอภาพสี และฮาร์ดแวร์ของระบบแสดงผล 24 บิท ทำให้สามารถแสดงสีในรูปจำลองแบบอาร์จีบี 24 บิทได้ทันที ซึ่งจำนวนสีที่สามารถแสดงได้พร้อม ๆ กันมีมากถึง 16.7 ล้านสี ดังที่ได้กล่าวมาแล้ว ดังนั้นการนำไมโครคอมพิวเตอร์ส่วนบุคคลมาใช้ในทางด้านการสำรวจข้อมูลระยะไกลจึงไม่ใช่เรื่องฝันอีกต่อไป

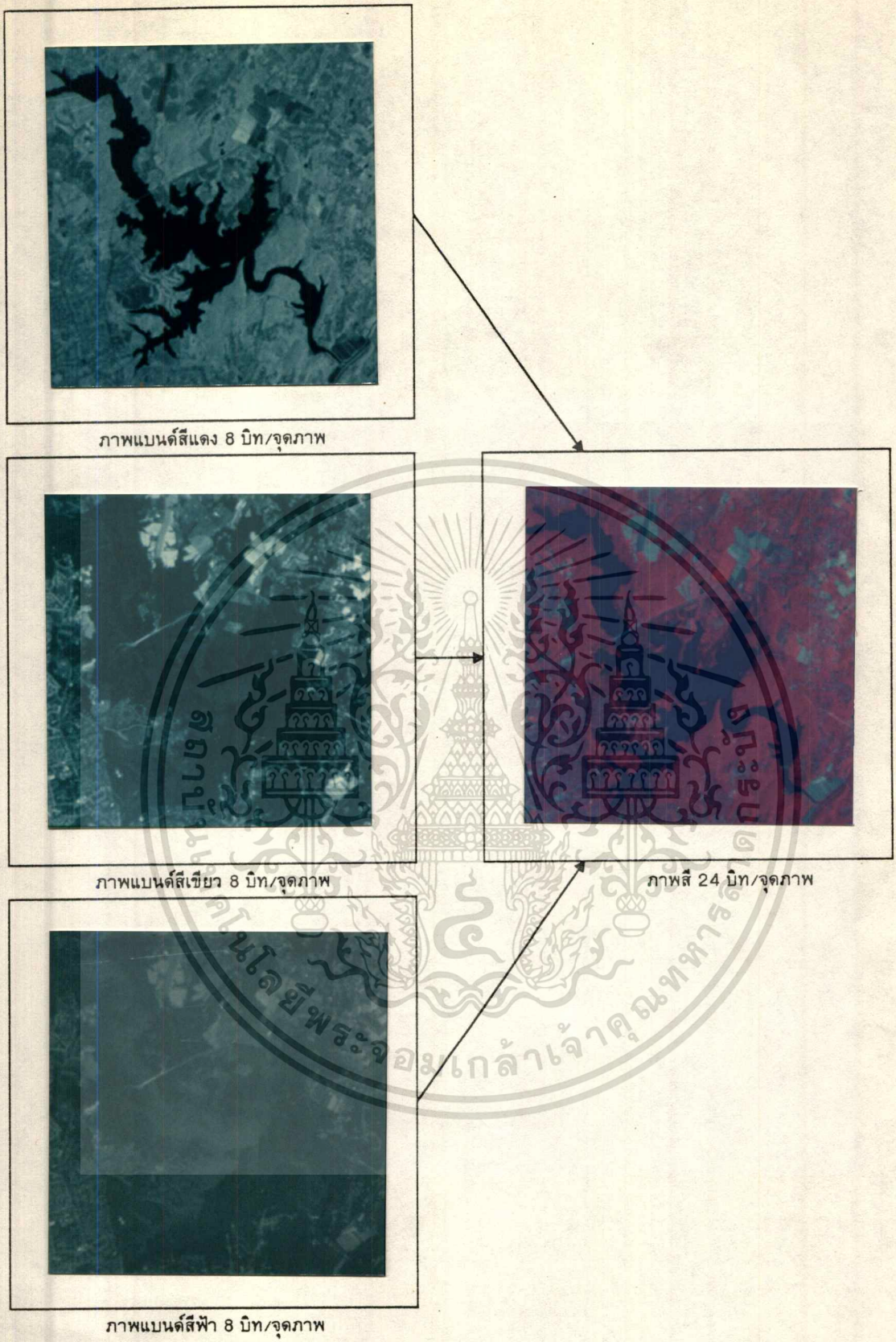
ในการผสมสี ข้อมูลสีแต่ละสีจะถูกเลือกมาจากข้อมูลภาพแบนด์ต่าง ๆ ของข้อมูลภาพแบบหลายช่วงคลื่น (multispectral image) ดังแสดงในรูป 2.2



รูปที่ 2.2 แสดงการเลือกข้อมูลภาพแบนด์ต่าง ๆ ของข้อมูลภาพแบบหลายช่วงความยาวคลื่นมาเป็นองค์ประกอบสีในการแสดงภาพสีผสมแบบอาร์จีบี

จากรูปที่ 2.2 แสดงให้เห็นถึงลักษณะของข้อมูลภาพแบบหลายความยาวคลื่นที่เกิดจากคุณสมบัติในการสะท้อนคลื่นความร้อนในย่านความยาวคลื่นแตกต่างกันของทรัพยากรต่าง ๆ บนผิวโลก ซึ่งในการแสดงภาพถ่ายดาวเทียมสีนั้นสิ่งสำคัญประการแรกคือการเลือกข้อมูลภาพที่มีการสะท้อนคลื่นความร้อนในช่วงความยาวคลื่นต่าง ๆ ของข้อมูลในแต่ละระบบมาเป็นองค์ประกอบข้อมูลสีแดง สีเขียว และสีน้ำเงิน เพื่อให้ได้ภาพสีผลลัพธ์ที่สามารถแสดงรายละเอียดของทรัพยากรต่าง ๆ บนผิวโลกได้อย่างครบถ้วน จึงทำการกำหนดให้องค์ประกอบของระบบ MSS ประกอบด้วยข้อมูลภาพแบนด์ 7 สำหรับสีแดง ข้อมูลภาพแบนด์ 5 สำหรับสีเขียว และข้อมูลภาพแบนด์ 4 สำหรับสีฟ้า องค์ประกอบของระบบ TM ประกอบด้วยข้อมูลภาพแบนด์ 4 สำหรับสีแดง ข้อมูลภาพแบนด์ 3 สำหรับสีเขียว และ ข้อมูลภาพแบนด์ 1 สำหรับสีฟ้า และองค์ประกอบของระบบ SPOT ประกอบด้วยข้อมูลภาพแบนด์ 3 สำหรับสีแดง ข้อมูลภาพแบนด์ 2 สำหรับสีเขียว และข้อมูลภาพแบนด์ 1 สำหรับสีฟ้า

ตัวอย่างภาพสี 24 บิทของระบบ TM สามารถแสดงได้ดังรูปที่ 2.3



รูปที่ 2.3 แสดงการผสมภาพสามแบนด์ออกมาเป็นภาพสี 24 บิทของภาพขนาด 256×256 ในระบบ TM

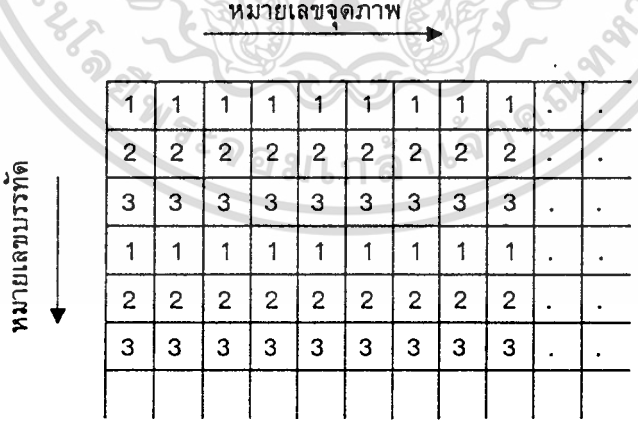
2.3 รูปแบบข้อมูลภาพ

สำหรับรูปแบบข้อมูลภาพถ่ายนอกจากที่มีในภาพถ่ายดาวเทียมในการสำรวจข้อมูลระยะไกลแล้วก็ยังมีรูปแบบมาตรฐานในภาพที่ได้จากสแกนเนอร์ หรือกล้องดิจิไตซ์เซอร์ของโปรแกรมประยุกต์ที่ทำงานบนไมโครซอฟท์วินโดว์ บนเครื่องไมโครคอมพิวเตอร์ส่วนบุคคลทั่ว ๆ ไปด้วย ดังมีรายละเอียดตามลำดับดังนี้

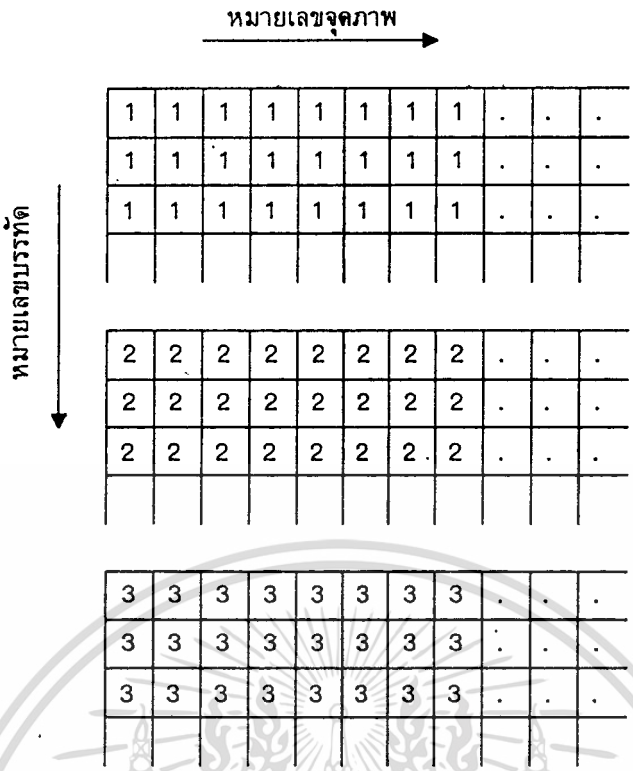
2.3.1 รูปแบบข้อมูลภาพถ่ายดาวเทียมในการสำรวจระยะไกล

ภาพถ่ายดาวเทียมแบบหลายความถี่ (multispectral image) นั้นในการบันทึกภาพหลาย ๆ แบนด์เป็นภาพภาพเดียวในรูปแบบที่ใช้กันในการสำรวจข้อมูลระยะไกลนั้นจะใช้การแทรกภาพกันในลักษณะต่าง ๆ 3 รูปแบบ คือ band-interleaved by pixel (BIP) band-inleaved by line (BIL) และ band sequential (BSQ) ซึ่งรูปแบบทั้งสามสามารถแสดงได้ดังรูปที่ 2.4

(a) Band interleaved by pixel (BIP)



(b) Band interleaved by line (BIL)



(c) Band Sequential (BSQ)

รูปที่ 2.4 แสดงรูปแบบข้อมูลภาพโดยทั่วไปของภาพถ่ายดาวเทียมแบบหลายความถี่ที่ประกอบด้วยภาพ 3 แบนด์ ค่าตัวเลข 1,2,3 หมายถึงแบนด์ 1, แบนด์ 2 และแบนด์ 3 ตามลำดับ

โดยแต่ละรูปแบบก็เหมาะสมกับการประมวลผลเฉพาะเรื่องไป ถ้าการประมวลผลเป็นแบบจุดต่อจุดในการจำแนกข้อมูลภาพ (image classification) หลาย ๆ แบนด์ ตัวอย่างเช่น ถ้าใช้รูปแบบ BIP จะเหมาะสมในการใช้และมีความสะดวกมาก เพราะว่าค่าระดับสีเทาของจุดภาพในแต่ละแบนด์ถูกเก็บอยู่อย่างต่อเนื่องกันในส่วนบันทึกข้อมูล ถ้าหากต้องการใช้งานภาพเพียงแบนด์เดียวจากภาพหลายแบนด์ รูปแบบ BSQ จะมีความเหมาะสมมากที่สุด เพราะใช้เวลาในการอ่านข้อมูลภาพแบนด์เดียวน้อยที่สุด รูปแบบ BIL เป็นรูปแบบที่ประนีประนอมมากที่สุดในประสิทธิภาพและความเหมาะสมสำหรับการประยุกต์โดยทั่ว ๆ ไป และน่าจะเป็นรูปแบบที่มีการใช้อย่างกว้างขวางกว่ารูปแบบอื่นทั้ง 2 แบบ

สำหรับภาพถ่ายดาวเทียมแลนด์แซท โดยปกติมักจะมีการบันทึกหัว (header) หรือ (trailer) ที่จุดเริ่มต้น หรือเมื่อสิ้นสุดข้อมูลภาพตามลำดับ โดยจะเป็นตัวอธิบายภาพเช่น หมายเลขภาพ จำนวนจุดภาพต่อเส้น (หรือจำนวนไบต์ต่อเรคคอร์ด) จำนวนเส้น จำนวนแบนด์ ข้อมูลในการปรับ radiometric calibration เป็นต้น โดยข้อมูลเหล่านี้มักจะถูกบันทึกในรูปแบบต่าง ๆ กันตามลักษณะการใช้ข้อมูลภาพ และได้อธิบายตามเอกสารประกอบที่ติดมากับเทปข้อมูลนั้น

2.3.2 รูปแบบข้อมูลภาพถ่ายโดยทั่วไป

รูปแบบที่มีใช้อยู่โดยทั่วไปมีรูปแบบดังนี้คือ GIF IFF/LBM WPG BMP PIC TGA PCX และ TIFF สำหรับรูปแบบใหญ่ ๆ ที่นิยมใช้กันมากคือ PCX GIF BMP และ TIFF แต่รูปแบบหนึ่งที่ได้รับความนิยมมากที่สุด คือ BMP หรือบิตแมพ สาเหตุก็เพราะว่าข้อมูลแบบบิตแมพนั้นไม่มีการบีบข้อมูล (compression) จึงมีความรวดเร็วและสะดวกในการอ่านข้อมูลอย่างมากและเป็นรูปแบบข้อมูลกลางที่ใช้ในการแลกเปลี่ยนระหว่างกันของโปรแกรมประยุกต์ต่าง ๆ ใช้ในการส่งข้อมูลไปให้อุปกรณ์ต่าง ๆ เช่นอุปกรณ์การแสดงผลเป็นต้น แต่ในรูปแบบบิตแมพนี้ต้องเปลืองเนื้อที่ฮาร์ดดิสในการจัดเก็บข้อมูลมากกว่ารูปแบบอื่นที่ใช้การเข้ารหัสบีบข้อมูล

2.3.2.1 รูปแบบข้อมูลแบบ BMP

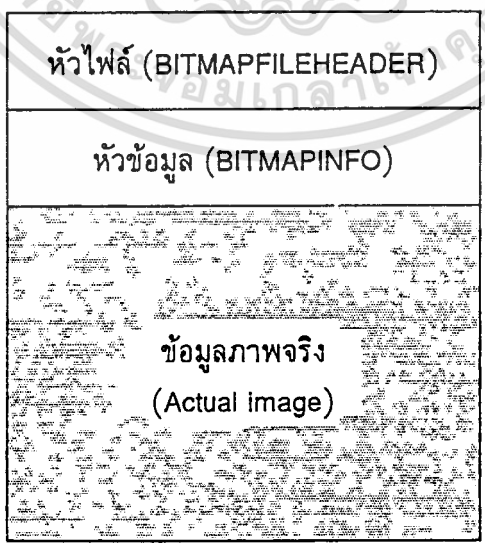
รูปแบบสำหรับใช้กับโปรแกรมระบายสีในวินโดว์ version 3.0 เป็นต้นมา โดยรูปแบบข้อมูลแบบนี้สามารถเป็นข้อมูลสีได้ ตั้งแต่ 1 ถึง 24 บิท มีส่วนหัว (header) ในการบอกรายละเอียดต่าง ๆ ของภาพ โดยกำหนดในลักษณะโครงสร้าง (structure) ในภาษาระดับสูง ในที่นี้ใช้ภาษาซี ข้อมูลภาพในรูปแบบบิทแมพประกอบด้วยสามส่วนคือ

-หัวไฟล์ (BITMAPFILEHEADER)

-หัวข้อมูล (BITMAPINFO) ซึ่งรวมทั้งข้อมูลต่าง ๆ และพาเลตของสี (color palette)

-ข้อมูลภาพจริง

รูปที่ 2.5 แสดงส่วนประกอบกันของโครงสร้างที่ใหญ่ที่สุดของภาพถ่ายในรูปแบบบิทแมพ ซึ่งข้อมูลพาเลตสีและข้อมูลภาพจะเปลี่ยนโครงสร้างไปตามรูปแบบของจำนวนสีของภาพและวิธีการถอดรหัส ที่ใช้ในการบีบข้อมูลภาพ ดังจะได้กล่าวต่อไป



รูปที่ 2.5 แสดงโครงสร้างที่ใหญ่ที่สุดของภาพถ่ายในรูปแบบบิทแมพ

2.3.2.1.1 หัวไฟล์ (BITMAPFILEHEADER)

หัวของไฟล์ (BITMAPFILEHEADER) จะประกอบด้วยข้อมูลเกี่ยวกับชนิดของรูปแบบข้อมูลภาพ ขนาดของภาพและโครงร่าง ดังนี้

BITMAPFILEHEADER ประกอบด้วย	
-WORD	bfType;
-DWORD	bfSize;
-WORD	bfReserved1;
-WORD	bfReserved2;
-DWORD	bfOffBits;

โดย **bfType** เป็นชนิดของรูปแบบข้อมูลภาพในที่นี้สำหรับบิทแมพคือ "BM"

bfSize เป็นขนาดของไฟล์ทั้งหมด

bfReserved1,2 ไม่ได้ใช้งาน ปกติกำหนดให้มีค่าเท่ากับศูนย์

bfOffBits เป็นขนาดของหัวไฟล์ทั้งหมด ใช้เพื่อข้ามไปจุดเริ่มต้นของข้อมูลภาพจริง

2.3.2.1.2 หัวข้อมูล (BITMAPINFO)

หัวข้อมูลจะเป็นตัวกำหนดขนาดต่าง ๆ (dimensions) และข้อมูลสี (color information) ดังนี้

BITMAPINFO ประกอบด้วย	
-BITMAPINFOHEADER	bmiHeader;
-RGBQUAD	bmiColors[1];

โดยที่ **bmiHeader** เป็นข้อมูลเกี่ยวกับขนาดต่าง ๆ จำนวนสี และรูปแบบของสี

bmiColors เป็นอาร์เรย์ข้อมูลสีอาร์จีบีที่เป็นตัวกำหนดสีในการแสดงผลของภาพ

ที่เป็นรายละเอียดต่าง ๆ จะอยู่ใน BITMAPINFOHEADER ดังนี้

STRUCTURE BITMAPINFOHEADER ประกอบด้วย

```
-DWORD    biSize;
-DWORD    biWidth;
-DWORD    biHeight;
-WORD     biPlanes;
-WORD     biBitCount;
-DWORD    biCompression;
-DWORD    biSizeImage;
-DWORD    biXPelsPerMeter;
-DWORD    biYPelsPerMeter;
-DWORD    biClrUsed;
-DWORD    biClrImportant;
```

โดยที่ **biSize** เป็นขนาดของส่วนหัวข้อมูล (BITMAPINFOHEADER) มีหน่วยเป็นไบต์

biWidth เป็นความกว้างของภาพ (มีหน่วยเป็น จุดภาพ/เส้น)

biHeight เป็นความสูงของภาพ (มีหน่วยเป็น เส้น)

biPlanes เป็นจำนวนหน้าของสี (color plane) สำหรับอุปกรณ์เป้าหมาย (targetdevice) ปกติกำหนดให้เป็น 1 เสมอ

biBitCount จำนวนบิทต่อจุดภาพ (1, 4, 8, 24) ดูรายละเอียดได้ในตารางที่ 2.1

biCompressionชนิดของการบีบอัด(compression)ดูรายละเอียดได้ในตารางที่ 2.2

biSizeImage เป็นขนาดของภาพ (มีหน่วยเป็นไบต์)

biXPelsPerMeter ความละเอียดสำหรับอุปกรณ์เป้าหมายในแนวนอนต่อหนึ่งเมตร

biYPelsPerMeter ความละเอียดสำหรับอุปกรณ์เป้าหมายในแนวตั้งต่อหนึ่งเมตร

biClrUsed จำนวนดัชนีสี (color index) ในตารางสี (color table) มีค่าได้ 3 กรณี โดยดูรายละเอียดได้ในตารางที่ 2.3

biClrImportant จำนวนความสำคัญของดัชนีสี (color index) ในการแสดงผล ถ้าเป็นศูนย์ทุกสีจะมีความสำคัญเท่ากันหมด

ค่า	อธิบาย	จำนวนแอร์เรย์สี (bmiColors)
1	บิทแมพ 2 สี (monochrome bitmap)	2 แอร์เรย์
4	บิทแมพมี 16 สีสูงสุด	16 แอร์เรย์
8	บิทแมพมี 256 สีสูงสุด	256 แอร์เรย์
24	บิทแมพมี 16.7 ล้านสีสูงสุด	NULL (ไม่มีแอร์เรย์)

ตารางที่ 2.1 แสดง จำนวนบิทต่อจุดภาพ ของ biBitCount

ชนิด	อธิบาย
BI_RGB	แสดงว่าข้อมูลไม่มีการบีบอัด
BI_RLE4	ทำการเข้ารหัสรันเลนส์ (run-length) ด้วยขนาด 4 บิทต่อจุดภาพ โดยในสองไบต์จะประกอบด้วย ไบต์นับ (count byte) และ ดัชนีสีขนาด 1 ไบต์ที่แสดงผลได้ 2 จุดภาพ
BI_RLE8	ทำการเข้ารหัสรันเลนส์ (run-length) ด้วยขนาด 8 บิทต่อจุดภาพ โดยในสองไบต์จะประกอบด้วย ไบต์นับ (count byte) และ ดัชนีสีขนาด 1 ไบต์ในการแสดงผล 1 จุดภาพ

ตารางที่ 2.2 แสดงรูปแบบการบีบอัด (compression formats) ของ biCompression

ค่า	อธิบาย	
0	ใช้จำนวนของสีสูงสุดเท่ากับ $2^{biBitCount}$	
ไม่เป็นศูนย์	ถ้า biBitsCount<24 biClrUsed จะระบุจำนวนสีที่ใช้ในการแสดงผล เช่น เมื่อ biBitsCount เท่ากับ 8 จะมีค่า biClrUsed เท่ากับ 256 สี เป็นต้น	ถ้าbiBitsCount=24 biClrUsed จะระบุขนาดของตารางอ้างอิง (reference table) เพื่อให้วินโดว์ได้รับรู้และใช้ในการแสดงผลพาเลตส์สีให้เหมาะสมที่สุด

ตารางที่ 2.3 แสดงจำนวนดัชนีสี (color index) ในตารางสี (color table) ของ biClrUsed

สำหรับส่วน RGBQUAD ที่เก็บรายละเอียดของ palette สีประกอบด้วยส่วนประกอบดังนี้

STRUCTURE RGBQUAD ประกอบด้วย
BYTE rgbBlue;
BYTE rgbGreen;
BYTE rgbRed;
BYTE rgbReserved; ไม่ใช้งาน (ปกติตั้งให้เป็นศูนย์)

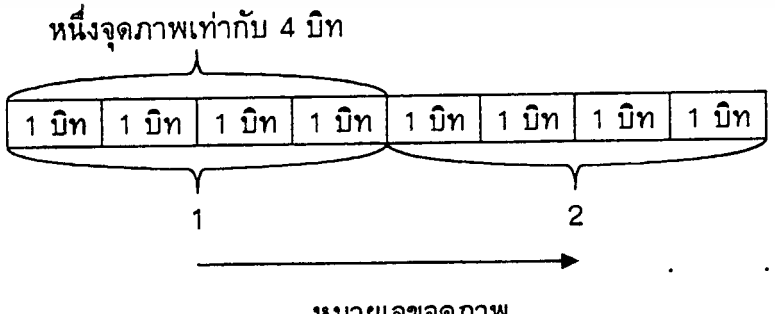
2.3.2.1.3 ข้อมูลภาพจริง (actual image information)

ส่วนของข้อมูลบิตแมพจริงจะมีการเก็บที่แตกต่างกันตามจำนวนบิทข้อมูลภาพ (biBitCount) และลักษณะการบีบอัด biCompression แต่โดยทั่วไปแล้วข้อมูลในรูปแบบบิตแมพปกติที่นิยมในปัจจุบันในการส่งข้ามระหว่างโปรแกรมประยุกต์ต่าง ๆ นั้น จะไม่ใช้การบีบอัดข้อมูล ดังนั้นจึงแบ่งรูปแบบข้อมูลในลักษณะที่นิยมใช้กันในปัจจุบันได้เพียงสามกรณีดังนี้

กรณีที่ 1 ถ้าเป็น 1 บิทต่อจุดภาพ หมายความว่ามีดัชนีสี bmiColors เป็นแอร์เรย์จำนวน 2 แอร์เรย์ คือมี 2 สี โดยข้อมูลแต่ละบิทจะแทนหนึ่งจุดภาพ ดังที่ 2.6

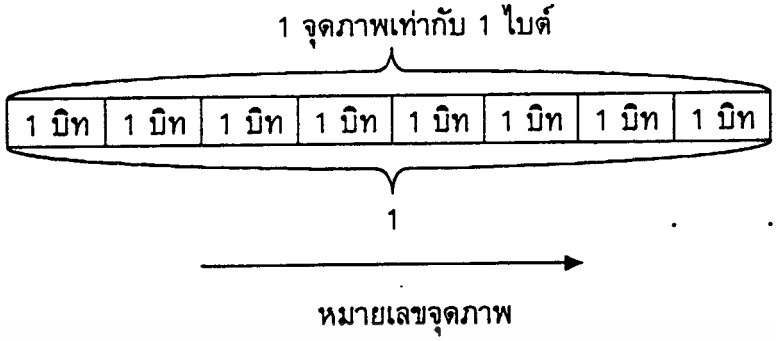
รูปที่ 2.6 แสดงข้อมูลภาพบิตแมพ 2 สี (monochrome Bitmap)

กรณีที่ 2 ถ้าเป็น 4 บิทต่อจุดภาพ หมายความว่ามีดัชนีสี bmiColors ที่เป็นแอร์เรย์จำนวน 16 แอร์เรย์ คือมี 16 สี โดยข้อมูลแต่ละ 4 บิทจะแทนหนึ่งจุดภาพ ดังรูปที่ 2.7



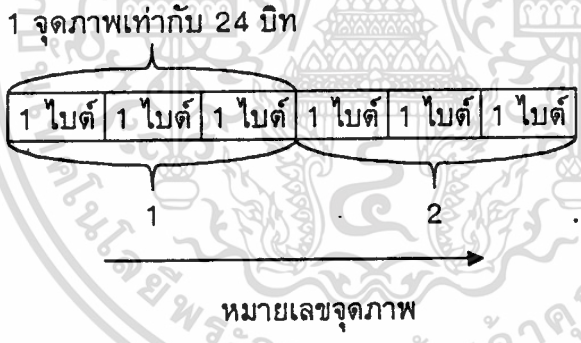
รูปที่ 2.7 แสดงข้อมูลภาพบิตแมพ 16 สี

<u>กรณีที่ 3</u> ถ้าเป็น 8 บิทต่อจุดภาพ หมายความว่ามีดัชนีสี bmiColors ที่เป็นแอร์เรย์จำนวน 256 แอร์เรย์ คือมี 256 สี โดยข้อมูลแต่ละ 8 บิทจะแทนหนึ่งจุดภาพ ดังรูปที่ 2.8



รูปที่ 2.8 แสดงข้อมูลภาพบิตแมพ 256 สี

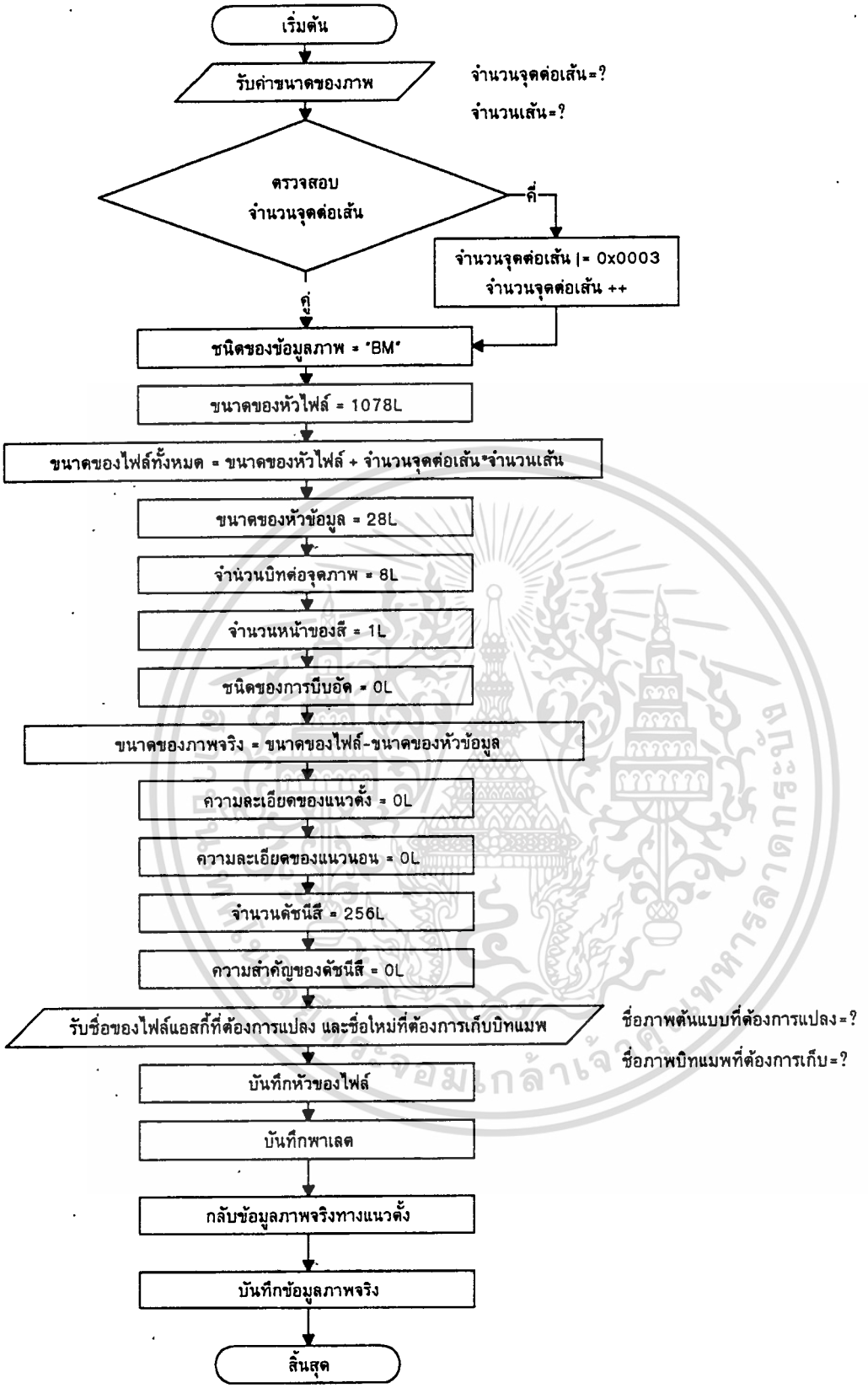
<u>กรณีที่ 4</u> ถ้าเป็น 24 บิทต่อจุดภาพ หมายความว่าจะสามารถแสดงสีได้พร้อม ๆ กันเท่ากับ 2^{24} คือ 16.7 ล้านสี คือสามารถผสมสีได้โดยตรง ไม่ต้องมีการใช้ดัชนีสี ดังนั้นในกรณีนี้ bmiColors = NULL ข้อมูลแต่ละ 3 ไบต์ที่เรียงกันจะแสดง relative intensity ของสีฟ้า สีเขียว และสีแดง (BGR) ตามลำดับ โดยข้อมูลแต่ละ 24 บิทจะแทนหนึ่งจุดภาพ ดังรูปที่ 2.9



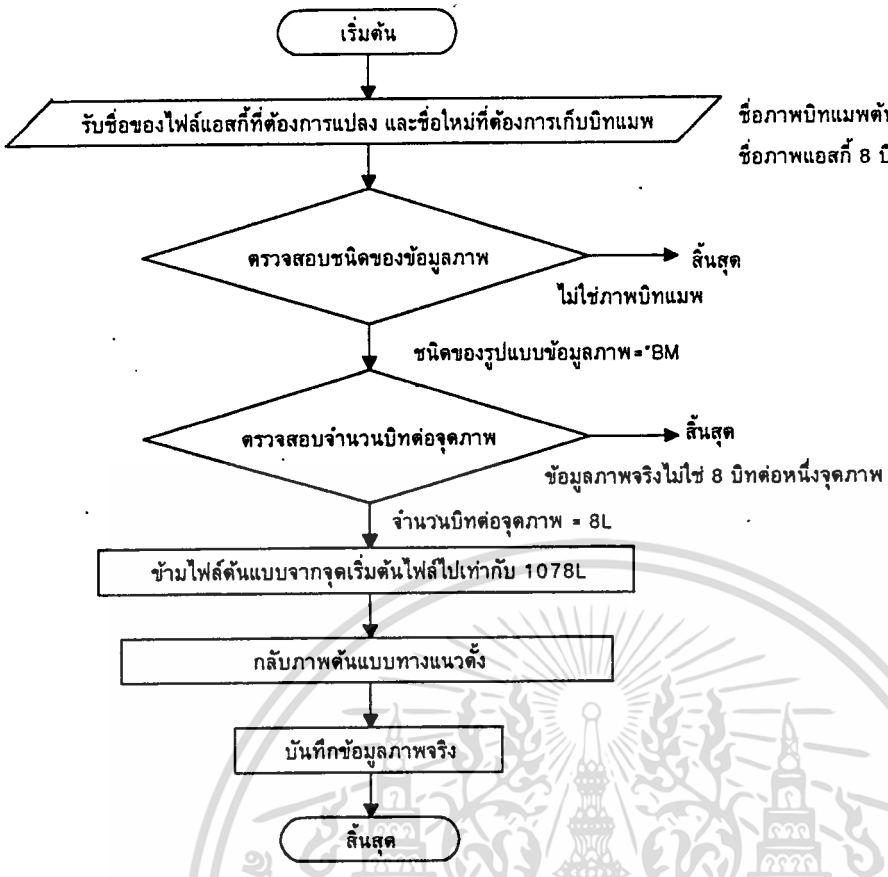
รูปที่ 2.9 แสดงข้อมูลภาพบิตแมพ 16.7 ล้านสี

2.3.3 การแปลงรูปแบบข้อมูลภาพถ่ายดาวเทียมเป็นภาพถ่ายโดยทั่วไป

เพื่อให้สามารถติดต่อสื่อสารกับโปรแกรมประยุกต์อื่น ๆ ได้นั้นจำเป็นต้องมีการแปลงภาพถ่ายดาวเทียมแอสกี้ 8 บิทให้เป็นรูปแบบบิทแมพที่มีหัวไฟล์เพิ่มเติมเข้ามาเพื่อบอกรายละเอียดต่าง ๆ ที่ให้ความสะดวกแก่การใช้งาน โดยมีอัลกอริทึมดังรูปที่ 2.10 และในการแปลงกลับรูปแบบบิทแมพให้กลับมาเป็นภาพถ่ายดาวเทียมแอสกี้ 8 บิท ก็สามารถทำได้ดังอัลกอริทึมดังรูปที่ 2.11 โดยมีโปรแกรมการทำงานทั้งหมดดังแสดงในภาคผนวก ข.



รูปที่ 2.10 แสดงลำดับขั้นตอนการแปลงข้อมูลภาพแอสกี้ 8 บิทเป็นรูปแบบบิทแมพ



รูปที่ 2.11 แสดงลำดับขั้นตอนการแปลงภาพในรูปแบบบิทแมพเป็นข้อมูลภาพแอสกี้ 8 บิท

2.4 ความรู้ทั่วไปในการปรับปรุงภาพ

2.4.1 การทำคอนโวลูชันในสปาเชียนโดเมน

การทำคอนโวลูชันเป็นการทำงานในรูปแบบที่เรียกว่า Neightbourhood operation ซึ่งเป็นการทำงานที่ต้องใช้จุดข้างเคียงมาพิจารณาในการคำนวณ การทำ Neighbourhood operation นั้นผลลัพธ์ได้ค่าออกมา ณ ตำแหน่งเดียวกันกับอินพุตและใช้จุดข้างเคียงที่อยู่รอบ ๆ ซึ่งการทำคอนโวลูชันเป็นการคำนวณผลรวมของผลคูณ (sum of product) รูปที่ 2.12 แสดงหน้ากากของคอนโวลูชัน (convolution mask) และจุดข้างเคียงจากภาพ สมาชิกแต่ละตัวในหน้ากากเรียกว่าน้ำหนัก (weight) ค่าน้ำหนักในหน้ากากเป็นตัวคำนวณผลของการทำคอนโวลูชัน ซึ่งจะกำหนดลักษณะการกรองตามการประยุกต์ใช้ในงานต่าง ๆ

$$\begin{matrix} M(-1,-1) & M(0,-1) & M(1,-1) \\ M(-1,\ 0) & M(0,\ 0) & M(1,\ 0) \\ M(-1,\ 1) & M(0,\ 1) & M(1,\ 1) \end{matrix}$$

(a) หน้ากากของการทำคอนโวลูชันขนาด 3×3

$$
\begin{array}{ccccc}
\cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & F(i-1,j-1) & F(i,j-1) & F(i+1,j-1) & \cdots \\
\cdots & F(i-1,j\ \ ) & F(i,j\ \ ) & F(i+1,j\ \ ) & \cdots \\
\cdots & F(i-1,j+1) & F(i,j+1) & F(i+1,j+1) & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots
\end{array}
$$

(b) จุดข้างเคียง (neighbourhood) ต่าง ๆ ในบริเวณเมตริกซ์ขนาด 3×3 ของภาพต้นแบบ

รูปที่ 2.12 แสดงคู่ของการทำคอนโวลูชั่นในสปาเชียลโดเมน

ตัวชี้ในหน้ากากมีจุดเริ่มต้นที่จุดศูนย์กลาง โดยเริ่มที่ตำแหน่งมุมบนด้านซ้ายของภาพ สามารถเขียนสมการคอนโวลูชันได้ดังนี้

$$C(i,j)=\sum_{k=i-1}^{i+1}\sum_{l=j-1}^{j+1}F(k,l)M(i-k,k-l) \qquad (2.1)$$

แต่ละจุดภาพจะเป็นการวนเอามาจากจุดรอบข้างและคูณด้วยค่าที่ตรงตำแหน่งเดียวกันกับหน้ากาก นำผลคูณของแต่ละตัวนั้นมารวมกันก็จะได้ ค่าจุดภาพผลลัพธ์ ซึ่งการคำนวณที่ทำนั้นสามารถกระจายแสดงออกมาได้ดังนี้

$$
\begin{aligned}
C(i,j) = {} & F(i-1,j-1)M(1,\ 1)+F(i,j-1)M(0,\ 1)+F(i+1,j-1)M(-1,\ 1)+ \\
& F(i-1,j\ \ )M(1,\ 0)+F(i,j\ \ )M(0,\ 0)+F(i+1,j\ \ )M(-1,\ 0)+ \\
& F(i-1,i+1)M(1,-1)+F(i,j+1)M(0,-1)+F(i+1,j+1)M(-1,-1)
\end{aligned}
\qquad (2.2)
$$

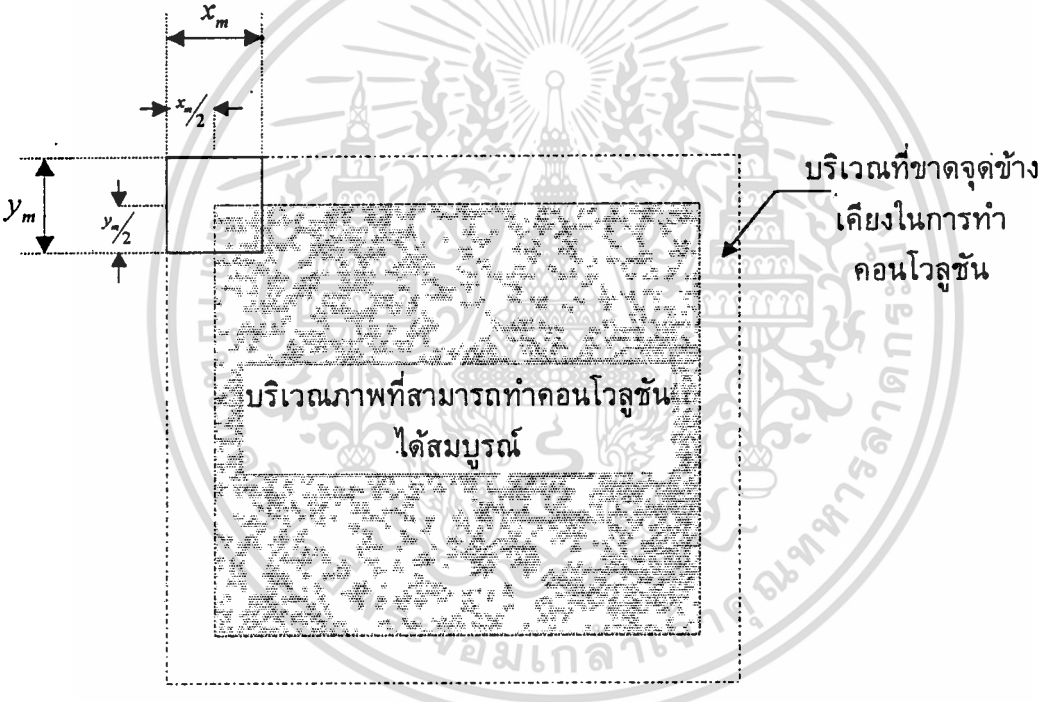
จากสมการข้างบนแสดงการจัดเรียงเทอมต่าง ๆ โดยลำดับในหน้ากากมีลำดับจากบนซ้ายไปยังด้านล่างขวา สำหรับค่าตำแหน่งที่ตรงกันในหน้ากากจะมีลำดับตรงกันข้ามคือเริ่มจากด้านล่างขวาขึ้นไปยังด้านบนซ้าย สำหรับสิ่งที่น่าสนใจของการตรงกันข้ามของลำดับนี้ก็คือว่าถ้าหมุนหน้ากาก 180° ลำดับของเทอมภาพกับเทอมหน้ากากจะมีลำดับในทิศทางเดียวกัน โดยได้การกระจายดังนี้

$$
\begin{aligned}
C(i,j) = {} & F(i-1,j-1)M(-1,-1)+F(i,j-1)M(0,-1)+F(i+1,j-1)M(1,-1)+ \\
& F(i-1,j\ \ )M(-1,\ 0)+F(i,j\ \ )M(0,\ 0)+F(i+1,j\ \ )M(1,\ 0)+ \\
& F(i-1,i+1)M(-1,-1)+F(i,j+1)M(0,\ 1)+F(i+1,j+1)M(1,\ 1)
\end{aligned}
\qquad (2.3)
$$

ในการประยุกต์ใช้งานบางอย่างนิยมใช้การหมุนหน้ากาก 180° นี้เพื่อให้ง่ายต่อการกำหนดตัวชี้ให้ภาพและหน้ากากมีตัวชี้ที่เหมือนกัน ในการประยุกต์บางอย่างจะมีการจัดเรียงสมาชิกของหน้ากากอย่างอัตโนมัติ โดยมีตำแหน่งของหน้ากากและภาพที่สอดคล้องกัน อีกทั้งหน้ากากก็ไม่จำเป็นต้องมีจำนวนแนวตั้งเท่ากับจำนวนแนวนอนเสมอไปด้วย แต่โดยส่วนใหญ่แล้วหน้ากากมักจะสมมาตรกัน (symmetric) และโดยทั่วไปความกว้างและความสูงจะเป็นจำนวนคี่ เพื่อให้จุดศูนย์กลางของหน้ากากเป็นจุดศูนย์กลางได้

- **การขาดจุดข้างเคียงในบริเวณของการทำคอนโวลูชันในบริเวณกรอบภาพ**

ในขบวนการคำนวณแบบ Neightbourhood operation นี้เป็นไปไม่ได้ที่จะทำคอนโวลูชันในบริเวณขอบรอบนอกของภาพ (บริเวณกรอบภาพ) เพราะในการคำนวณขาดจุดภาพที่อยู่ภายนอกอาณาบริเวณของภาพ ดังแสดงในรูปที่ 2.13



รูปที่ 2.13 การขาดจุดข้างเคียงในบริเวณของการทำคอนโวลูชันในบริเวณกรอบภาพ

ดังนั้นโดยทั่วไปแล้ววิธีการที่จะแก้ปัญหาเหล่านี้ได้ก็มักใช้วิธีการกำหนดแถบจุดภาพในบริเวณกรอบของภาพทั้ง 4 ด้านให้เป็นค่าเดิมหรือเป็นศูนย์แล้วแต่กรณี ความกว้างของแถบด้านบนและด้านล่างของภาพมีค่าเท่ากับ $y_m/2$ และด้านข้างซ้ายและขวาเท่ากับ $x_m/2$ เมื่อ x_m คือความกว้างของหน้ากาก และ y_m คือความสูงของหน้ากาก ตัวอย่างเช่นหน้ากากขนาด 3×3 จะมีแถบที่มีความกว้าง 1 จุดภาพ ซึ่งไม่สามารถคำนวณได้รอบ ๆ ภาพ

ในการใช้งานจะต้องรู้ว่าจะนำภาพที่ได้ไปทำอะไรต่อไป เพราะอาจจะเป็นไปได้ที่ภาพผลลัพธ์อาจจะมีขนาดเล็กกว่าภาพต้นแบบตามจำนวนโรว์และคอลัมน์ที่ไม่สามารถทำการคำนวณได้ ถ้าเงื่อนไขไม่เป็นเช่นนั้นและต้องการรักษาขนาดของภาพผลลัพธ์ให้เท่ากับภาพต้นแบบก็ใช้การใส่ค่าเดิมหรือศูนย์แล้วแต่กรณีลงไปในกรอบที่ไม่สามารถคำนวณได้ ซึ่งวิธีนี้เป็นที่นิยมใช้กันอย่างมากเพราะภาพผลลัพธ์สามารถนำไปเปรียบเทียบกับภาพต้นแบบได้โดยตรงและในอนาคตก็สามารถนำไปคำนวณกันได้ต่อไปในขณะที่ยังรักษาแนวของจุดภาพให้ตรงกันอยู่อย่างนั้น

บางระบบต้องการการคำนวณที่สมบูรณ์แบบอย่างแท้จริง โดยการใช้ข้อมูลในบริเวณอื่น ๆ มาใช้แทนบริเวณโรว์และคอลัมน์ของจุดภาพที่อยู่เลยออกไปจากภาพต้นแบบ วิธีการหนึ่งนั้นทำได้โดยการใช้โรว์ของทางด้านล่างของภาพมาใช้เป็นข้อมูลที่ต้องการของการคำนวณทางด้านบน และในทำนองเดียวกันโดยการใช้โรว์ของทางด้านบนของภาพมาใช้เป็นข้อมูลที่ต้องการของการคำนวณทางด้านล่าง ในลักษณะที่คล้ายกันของคอลัมน์ทางด้านขวาของภาพก็นำมาใช้เป็นข้อมูลที่ต้องการของการคำนวณทางด้านซ้ายเสร็จสมบูรณ์ และคอลัมน์ทางด้านซ้ายของภาพก็นำมาใช้เป็นข้อมูลที่ต้องการของการคำนวณทางด้านขวา เช่นกัน ถึงแม้ว่าวิธีการเช่นนี้ไม่เป็นการเหมาะสมนักก็ตาม แต่เพื่อให้ผลลัพธ์มีขนาดภาพเท่ากับภาพต้นแบบ และในการประยุกต์ใช้งานบางอย่างต้องหลีกเลี่ยงการเกิดการเพี้ยนอย่างรุนแรงของวิธีการคำนวณจากการแทนกรอบที่คำนวณไม่ได้ด้วยศูนย์ เช่นในกรณีทำเกรเดียนท์ เป็นต้น

บางระบบสามารถใช้วิธีแทนบริเวณกรอบที่คำนวณไม่ได้นั้นด้วยกรอบของภาพต้นแบบในตำแหน่งที่ตรงกันได้อย่างทันที ถ้าหากเป็นการประยุกต์ที่ภาพผลลัพธ์ค่อนข้างคล้ายกับภาพต้นแบบอย่างมากเช่นการทำให้เรียบ (smoothing) เป็นต้น ถ้าไม่มีสัญญาณรบกวนอยู่ในภาพต้นแบบในส่วนของกรอบก็จะไม่สามารถสังเกตุข้อแตกต่างได้เลย

- **การนอร์มอลไลซ์ (Normalization)**

ในกรณีที่ต้องการทำการประมวลผลให้ข้อมูลมีความราบเรียบ (smoothing) เพื่อกำจัดสัญญานรบกวนนั้น คุณสมบัติบางประการของข้อมูลภาพเดิมจำเป็นจะต้องรักษาเอาไว้ ในที่นี้คือคุณสมบัติเอกพันธ์หรือความเป็นเนื้อเดียวกันของกลุ่มจุดภาพข้างเคียง ถ้าหากพื้นที่ใด ๆ มีความเป็นเนื้อเดียวกันอยู่แล้ว หลังจากการประมวลผลในการทำข้อมูลให้ราบเรียบจะต้องยังคงไว้ซึ่งความเป็นเนื้อเดียวเช่นเดิม ดังนั้นหน้ากากที่ใช้ในการประมวลผลความราบเรียบของข้อมูลจึงจำเป็นต้องมีการนอร์มอลไลซ์ ซึ่งค่าที่ใช้นอร์มอลไลซ์ในกรณีนี้คือผลรวมของน้ำหนักแต่ละจุดในหน้ากากนั้นเอง

<u>ตัวอย่าง</u> สมมุติค่าน้ำหนักของหน้ากากอันหนึ่งเป็น {1, 2, 1, 2, 4, 2, 1, 2, 1} เมื่อทำการรวมค่าน้ำหนักเราจะได้ 1+2+1+2+4+2+1+2+1 = 16 ก็ทำการนอร์มอลไลซ์ด้วย 16 ดังนี้

$$M(i,j)=\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \qquad (2.4)$$

- **เวลาที่ใช้**

การทำคอนโวลูชันค่อนข้างจะใช้การคำนวณมากครั้ง เช่นถ้าเป็นหน้ากาก 3×3 ในการที่จะได้ผลลัพธ์ 1 จุดภาพจะต้องใช้การคูณ 9 ครั้ง การบวก 9 ครั้ง และการหารอีก 1 ครั้ง ถ้ามีการทำนอร์มอลไลซ์พิจารณาการทำคอนโวลูชันของหน้ากากที่มีโรว์เท่ากับคอลัมน์แล้ว จำนวนการบวกจะเท่ากับจำนวนการคูณ ตารางที่ 2.4 แสดงจำนวนครั้งที่ต้องใช้ในการคำนวณ โดยทดลองใช้หน้ากากขนาดต่าง ๆ โดยใช้ภาพต้นแบบขนาด 512×512 โดยสมมุติให้สามารถทำคอนโวลูชันที่บริเวณกรอบของภาพได้ ตัวอย่างเมื่อใช้หน้ากากขนาด 3×3 ต้องใช้จำนวนครั้งในการคำนวณเท่ากับ 2,359,296×2+262,144 = 4,980,736 ครั้ง เป็นต้น

ขนาดหน้ากาก	จำนวนสมาชิกในหน้ากาก	จำนวนครั้งในการบวก และคูณ	จำนวนครั้งในการหาร	รวมจำนวนครั้งทั้งหมด
3	9	2,359,296	262,144	4,980,736
5	25	6,553,600	262,144	13,369,344
7	49	12,845,056	262,144	25,952,256
9	81	21,233,664	262,144	42,729,472
11	121	31,719,424	262,144	63,700,992
15	225	58,982,400	262,144	118,226,944
25	625	163,840,000	262,144	327,942,144

ตารางที่ 2.4 ภาระในการคำนวณ โดยการทดลองใช้ตารางหน้ากากจตุรัสขนาดต่าง ๆ กับภาพต้นแบบขนาด 512×512 จุดภาพ

เมื่อขนาดของหน้ากากใหญ่ขึ้นจะเห็นว่าจำนวนครั้งในการใช้คำนวณจะเพิ่มขึ้นมากอย่างมาก ซึ่งในตารางนั้นจะเป็นการคิดจำนวนครั้งในการคำนวณเท่านั้น มันไม่ได้นับการทำงานที่แฝงอยู่อื่น ๆ ที่จำเป็นต้องใช้เลย ได้แก่ การแยกจุดภาพที่ถูกต้องจากภาพและหน้ากาก เขียนเก็บข้อมูลของภาพผลลัพธ์ รักษาดัชนีตัวชี้และตัวนับ และควบคุมการทำงานทั้งหมด ดังนั้นจำนวนครั้งในการทำงานทั้งหมดในโครงสร้างการประมวลผลจึงใช้เวลามากกว่าที่แสดงมาในตารางข้างต้นมาก ดังนั้นเมื่อต้องมีการทำคอนโวลูชันจึงมักนิยมใช้ขนาดของหน้ากากที่เล็กที่สุดเท่าที่จะทำได้

2.4.2 การเปิดตาราง (look-up table)

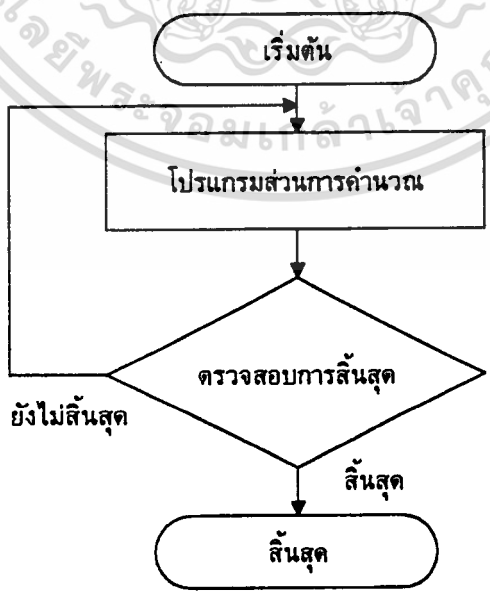
ในขบวนการเพิ่มประสิทธิภาพของการประมวลผลภาพนั้นมีการใช้การเปิดตารางกันอย่างกว้างขวางในการประยุกต์ใช้งานต่าง ๆ ที่ต้องการความเร็วสูง เช่น การปรับปรุงคอนทราสของภาพที่เป็นการแปลง (transformation) ของค่าระดับสีเทาแต่ละจุดภาพไปยังค่าระดับสีเทาใหม่ สำหรับการปรับปรุงภาพอย่างง่ายแล้ว ฟังก์ชันการแปลงจะเป็นฟังก์ชันเดียวกันสำหรับ ทุก ๆ จุดในภาพ ดังนั้นอาจเขียนได้ว่า

$$G'_L = T(G_L) \qquad (2.5)$$

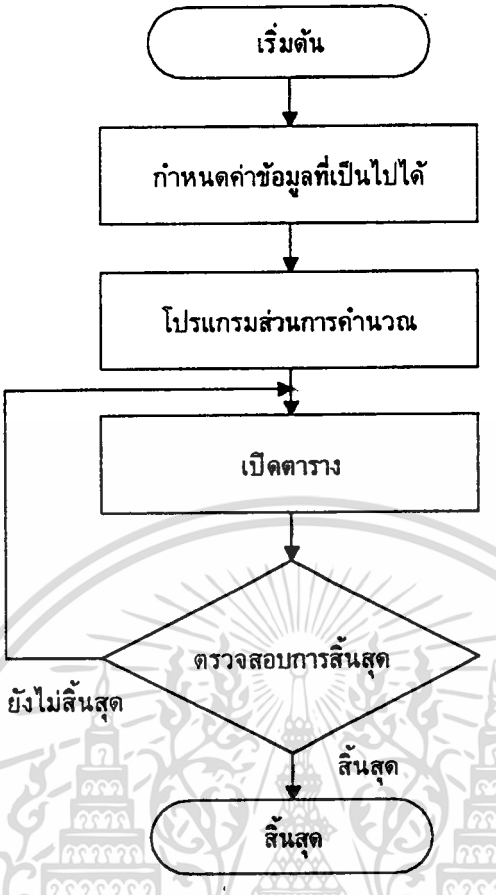
โดยที่ G_L และ G'_L เป็นค่าระดับสีเทาอินพุทและเอาต์พุทตามลำดับ และมี T คือ ฟังก์ชันการแปลง

ขั้นตอนในการทำการประมวลผลโดยตรงแสดงดังรูปที่ 2.14(a) ส่วนในการใช้วิธีการเปิดตารางแสดงดังรูปที่ 2.14(b) ซึ่งในการเขียนโดยภาษาระดับสูงต่าง ๆ ฟังก์ชันการแปลง $T(G_L)$ อาจจะกำหนดให้เป็นแอร์เรย์ (array) ที่มีจำนวนเต็ม (integer) G_L เป็นตัวชี้อยู่ ถ้ามีการใช้ฟังก์ชัน T เดียวกันในการแปลงทุก ๆ จุดภายในภาพ จึงสามารถใช้การคำนวณสำหรับทุก ๆ ค่าที่เป็นไปได้ของตัวชี้ G_L ก่อนการประมวลผล และทำการเก็บค่าระหว่างการคำนวณค่าต่าง ๆ ไว้ในแอร์เรย์ ภาพต้นแบบที่มีขนาด 8 บิทต่อหนึ่งจุดภาพ จะต้องการตารางแอร์เรย์จำนวน 256 ตาราง ในการประมวลผลถัดมาก็เป็นการนำค่าระดับสีเทาของจุดภาพต่าง ๆ มาเป็นตัวชี้ตาราง T ค่าที่สอดคล้องของ T จะเป็นค่าระดับสีเทาเอาต์พุท จะไม่มีการคำนวณระหว่างการเปิดตารางอีก ใช้เพียงตัวชี้ตำแหน่งที่อยู่ในตำแหน่งตารางที่ต้องการเท่านั้น

การเปิดตารางสามารถใช้สำหรับจุดประสงค์อื่น ๆ ได้อีกมากมาย ประหยัดจำนวนครั้งในการคำนวณซึ่งหมายถึงเวลาที่ใช้ก็ลดลงนั้นเอง เห็นได้ชัดเจนในการคำนวณเลขทศนิยมที่ใช้เวลาในการคำนวณสูง



(a) วิธีการคำนวณโดยตรง



(b) วิธีการเพิ่มความเร็วโดยการเปิดตาราง

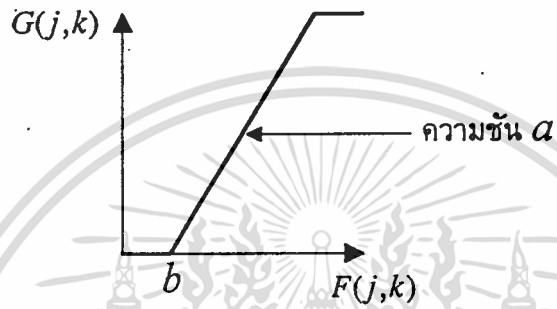
รูปที่ 2.14 แสดงลำดับขั้นตอนวิธีการคำนวณโดยตรงเทียบกับการใช้วิธีการเปิดตาราง

2.4.3 การปรับปรุงคอนทราสของภาพ (contrast enhancement)

เนื่องจากภาพถ่ายดาวเทียมโดยมากมักจะมีคอนทราสต่ำ และมีย่านไดนามิคแคบ กล่าวคือ ภาพมีระดับสีเทาของจุดภาพใกล้เคียงกันตลอดทั้งภาพ ถ้ามองจากฮีสโตแกรมของภาพจะพบว่าการกระจายของระดับสีเทาของจุดภาพจะรวมกันเป็นช่วงแคบ ๆ ทำให้ไม่สามารถมองเห็นรายละเอียดของภาพ ในการปรับปรุงคอนทราสของภาพโดยการเปลี่ยนลักษณะการกระจายฮีสโตแกรมของภาพใหม่ ให้มีย่านของข้อมูลที่กว้างและมีความต่างของค่าระดับสีเทาสูงขึ้น ซึ่งจะทำให้สามารถมองเห็นรายละเอียดที่อยู่ในภาพเดิมที่ไม่สามารถมองเห็นได้อย่างชัดเจน ในการปรับปรุงคอนทราสของภาพสามารถใช้ฟังก์ชันเชิงเส้น เช่น ฟังก์ชันเส้นตรงมาทำการกระจาย ฮีสโตแกรมเสียใหม่ ในการประมวลผลภาพถ่ายดาวเทียมส่วนมากมักใช้เพื่อการปรับปรุงคอนทราสของภาพ และแปลงข้อมูลที่เลยออกนอกย่านหลังการประมวลผลใด ๆ ให้กลับอยู่ในย่านที่ต้องการ

จากสมการที่ 2.6 ในเทอมสมการเส้นตรง $y=aF(j,k)+b$ นั้นจะมี a เป็นพารามิเตอร์เกณฑ์ (gain) ที่มีผลต่อความชัน (slope) ในการ mapping และมี b เป็นพารามิเตอร์ไบอัส (bias) เป็นค่าคงที่ที่บวกเข้าไปทุก ๆ จุดภาพ เมื่อนำมาใช้ในการปรับปรุงคอนทราสของภาพ ต้องคำนึงถึงค่าที่ต่ำกว่าจุดตัดแกนเดิมที่ต้องกำหนดให้เป็นศูนย์ และค่าสูงที่เลยออกนอกย่านที่ต้องกำหนดให้เป็น 255 ดังแสดงในรูปที่ 2.15

$$G(j,k)=\begin{cases} 0 & \leftarrow if \ldots 0 \le F(j,k) \le lower \\ aF(j,k)+b & \leftarrow if \ldots lower < F(j,k) < upper \\ 255 & \leftarrow if \ldots upper \le F(j,k) \le 255 \end{cases} \qquad (2.6)$$



รูปที่ 2.15 แสดงฟังก์ชันในการดึงอย่างเชิงเส้นอย่างง่าย

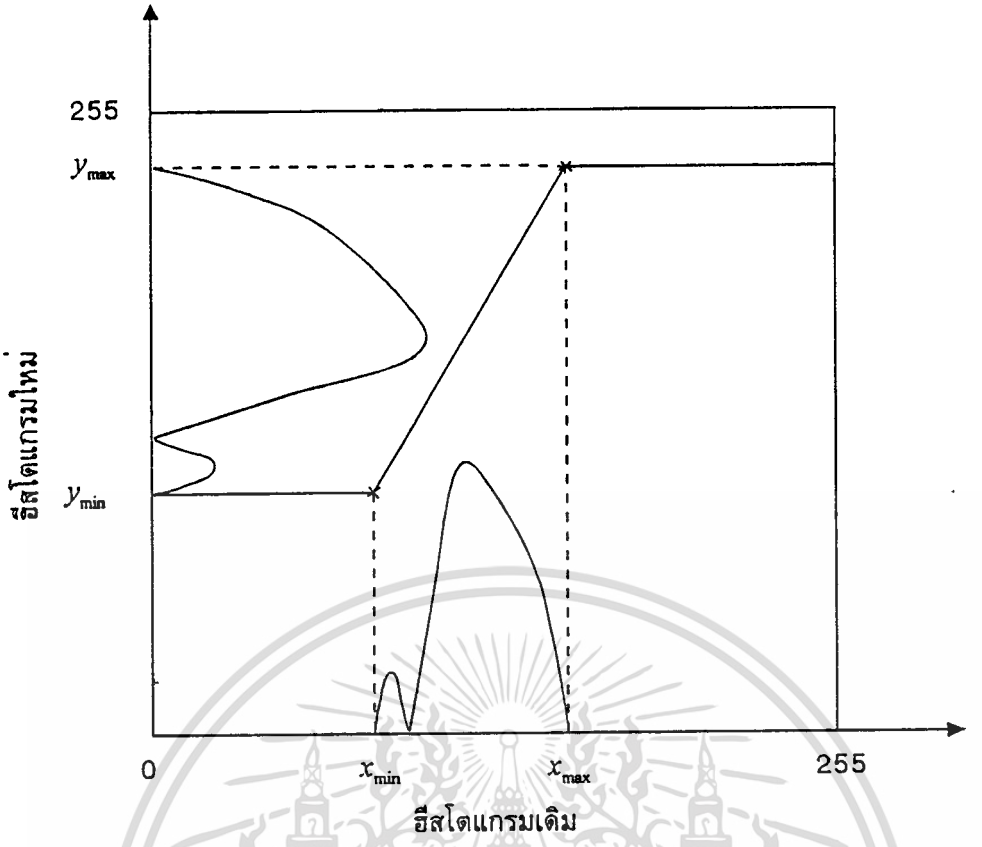
จะเห็นว่าการกำหนดตัวแปรเกณฑ์และไบอัสเพื่อให้ได้ฟังก์ชันเส้นตรงในการแปลงที่ต้องการนั้นกำหนดในลักษณะที่สื่อความหมายให้เข้าใจได้ยาก ดังนั้นในรูปที่กำหนดค่าได้ง่ายเป็นรูปแบบที่นิยมใช้มากที่สุด ดังนี้

$$G(j,k)=\begin{cases} y_{\min} & \leftarrow if \quad 0 \le F(j,k) \le x_{\min} \\ F'(j,k) & \leftarrow if \quad x_{\min} < F(j,k) < x_{\max} \\ y_{\max} & \leftarrow if \quad x_{\max} \le F(j,k) \le 255 \end{cases} \qquad (2.7)$$

เมื่อมีการเพิ่มการกำหนดสมการเส้นตรงในรูปแบบที่ง่ายขึ้น ดังนี้

$$F'(j,k)=\left\{\frac{(F(j,k)-x_{\min})(y_{\max}-y_{\min})}{(x_{\max}-x_{\min})}\right\}+y_{\min} \qquad (2.8)$$

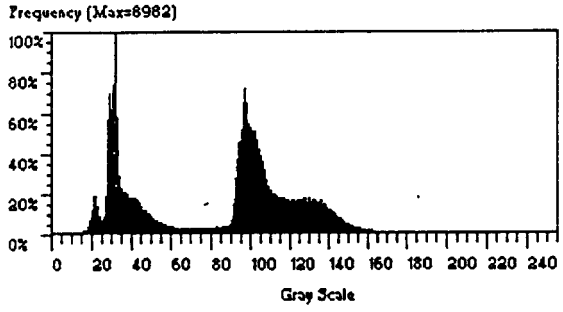
ในขณะที่วิธีการปกติในสมการที่ (2.6) ไม่สามารถทำได้นั้นสมการที่ (2.7) และ (2.8) มีข้อได้เปรียบอีกอย่างหนึ่งคือสามารถแปลงการกระจายไปสู่ทุก ๆ ตำแหน่งบนฮีสโตแกรมใหม่ได้ทั้งหมดดังแสดงในรูปที่ 2.16 โดยสามารถกำหนดย่านบนแกนใหม่ได้ตามต้องการ



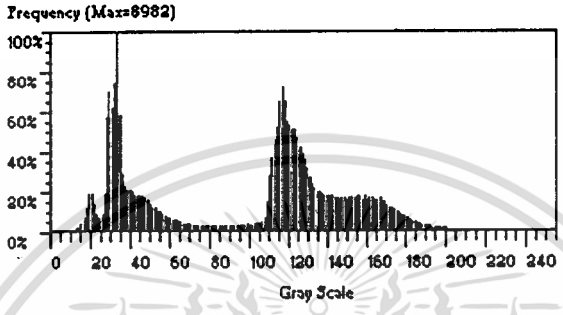
รูปที่ 2.16 แสดงฮิสโตแกรมการดึงอย่างเชิงเส้น โดยมีแนวนอนเป็นฮิสโตแกรมเดิมในแนวตั้งเป็นฮิสโตแกรมใหม่ของผลลัพธ์หลังการดึง

ในการประมวลผลสามารถนำวิธีการเปิดตารางดังที่ได้กล่าวมาแล้วในหัวข้อ 2.3.2 เพื่อช่วยเพิ่มความเร็วในการคำนวณได้ และนอกจากการให้ค่าสูงสุดและต่ำสุดโดยตรงแล้วยังสามารถใช้การคำนวณหาค่าสูงสุดและต่ำสุดของค่าระดับสีเทาแล้วทำการดึงอย่างอัตโนมัติ (automatic linear contrast stretching) ได้เช่นกัน จึงมีประโยชน์อย่างมากในการนำไปใช้ดึงค่าที่เลยออกนอกย่านให้กลับเข้ามาอยู่ในย่าน 0-255

นอกจากนี้ในทางปฏิบัติจะพบภาพที่มีค่าระดับสีเทาเด่นออกมาจากกลุ่มค่าระดับสีเทาส่วนใหญ่ของภาพ ซึ่งค่าระดับสีเทานี้เมื่อเทียบกันแล้วจะเป็นเพียงจำนวนน้อย ๆ เท่านั้น ดังนั้นในการดึงอย่างอัตโนมัติจึงไม่อาจให้ค่าคอนทราสที่ดีออกมาได้ ตัวอย่างเช่นเมื่อมีสัญญาณรบกวนที่มีค่าระดับสีเทาสูงอยู่ 1 จุดภาพที่เป็นค่าระดับสีเทาสูงสุดในภาพเมื่อทำการดึงอย่างอัตโนมัติค่าระดับสีเทาของสัญญาณรบกวนนี้จะเป็นค่า x_{max} ทำให้เกิดเกณฑ์ของการดึงกลุ่มข้อมูลหลักผิดไปโดยได้คอนทราสที่น้อยกว่าที่ควรจะเป็น รูปที่ 2.17(a) แสดงฮิสโตแกรมรวมของภาพต้นแบบจากรูปที่ 2.2 ส่วนรูปที่ 2.17(b) แสดงฮิสโตแกรมของภาพผลลัพธ์ที่ได้จากการดึงอย่างอัตโนมัติด้วยค่าสูงสุดและต่ำสุดที่หาได้



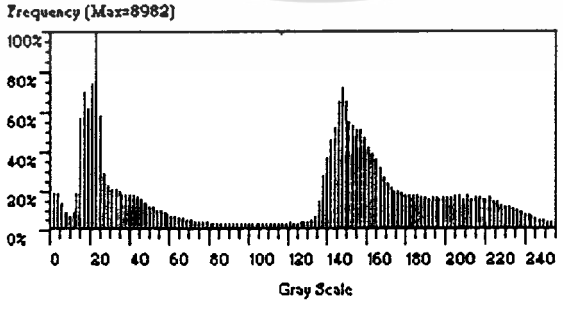
(a)



(b)

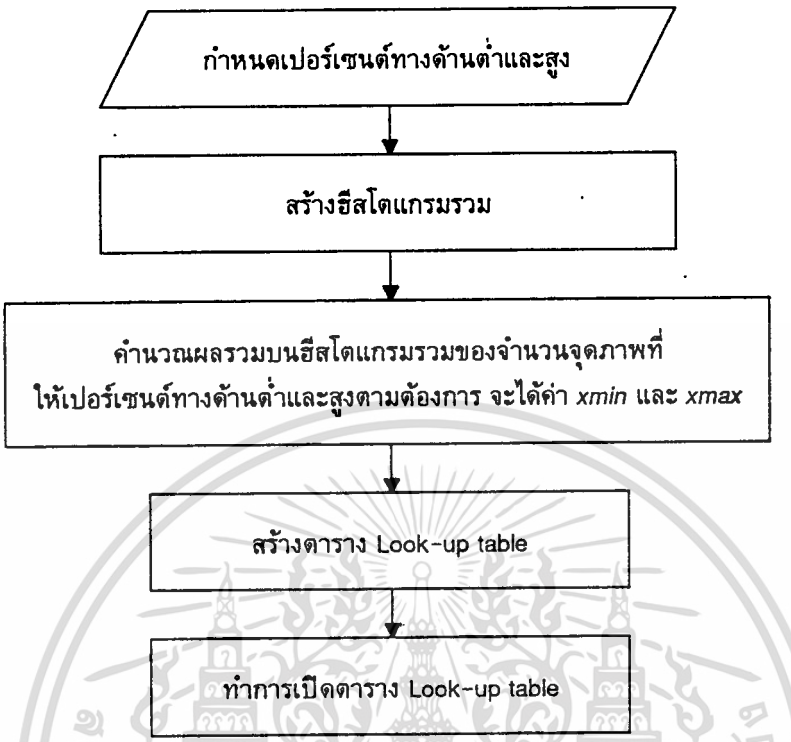
รูปที่ 2.17 แสดง (a) ฮีสโตแกรมรวมของภาพต้นแบบจากรูปที่ 2.2 และ (b) ฮีสโตแกรมของภาพผลลัพธ์ที่ได้จากการดึงอย่างอัตโนมัติด้วยค่าสูงสุดและต่ำสุดที่หาได้

จากรูปที่ 2.17(b) ซึ่งจะเห็นว่าไม่สามารถให้ค่าคอนทราสที่ดีออกมาได้ ดังนั้นในทางปฏิบัติวิธีการแก้ปัญหาทั้งหมดนี้ให้หมดไปทำได้โดยการนำเปอร์เซนต์ของจำนวนจุดภาพมาช่วยแก้ข้อบกพร่องของจุดภาพจำนวนน้อย ๆ เหล่านี้ทั้งทางด้านต่ำและทางด้านสูงของฮีสโตแกรม ภาพผลลัพธ์ที่ได้จะมีคอนทราสของกลุ่มข้อมูลส่วนใหญ่สูงตามที่ต้องการ สำหรับรูปที่ 2.18 แสดงฮีสโตแกรมที่ทำการดึงอย่างอัตโนมัติด้วยค่าสูงสุดและต่ำสุดที่นำค่าเปอร์เซนต์ของจำนวนจุดภาพทั้งทางด้านต่ำและทางด้านสูงมาใช้ โดยข้อมูลทางด้านต่ำและทางด้านสูงใช้จำนวนจุดเท่ากับ 1 เปอร์เซนต์ของจำนวนจุดภาพทั้งหมด



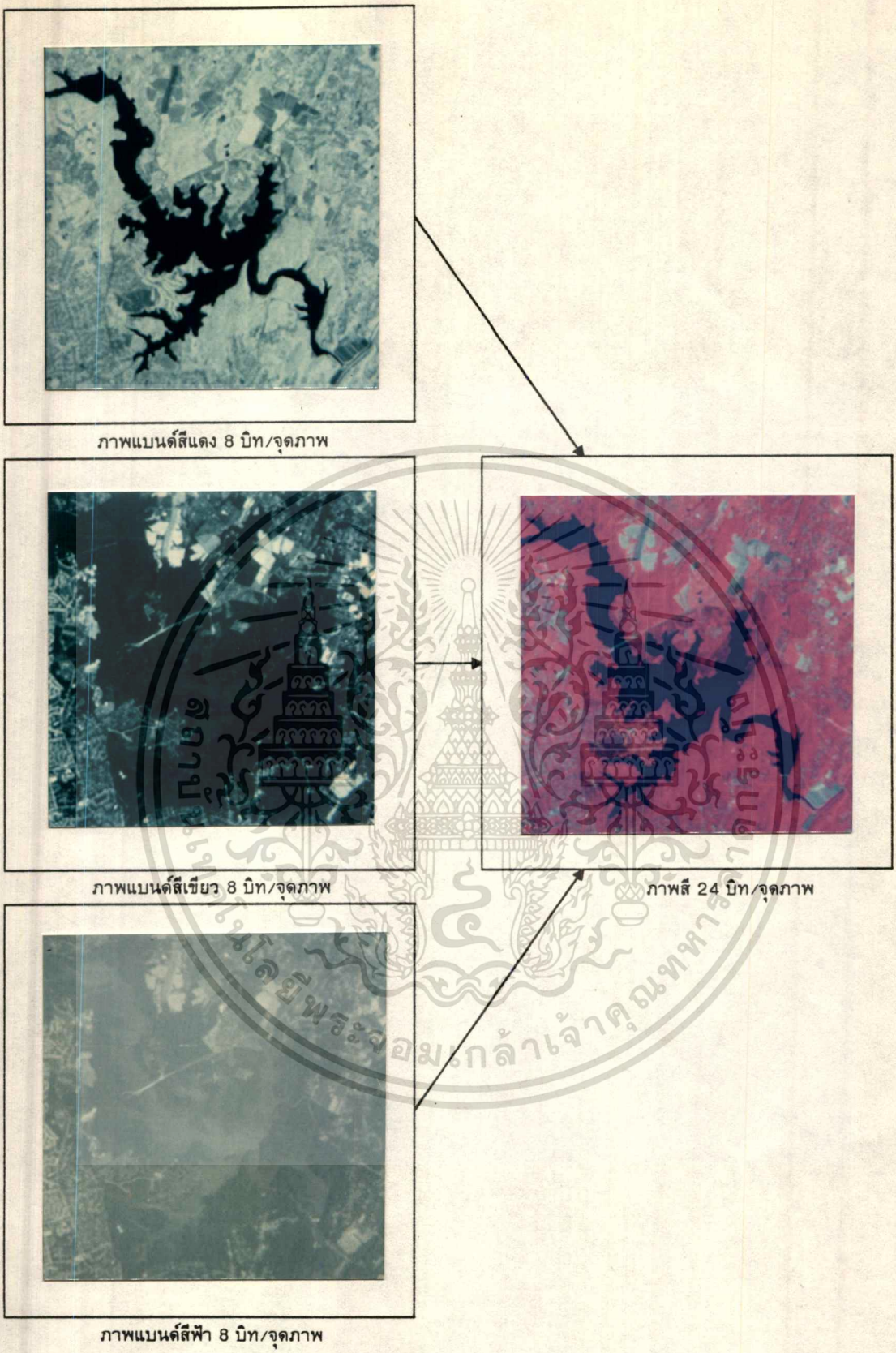
รูปที่ 2.18 แสดงฮีสโตแกรมรวมของภาพผลลัพธ์ที่ได้จากการดึงอย่างอัตโนมัติด้วยค่าสูงสุดและต่ำสุดจากการกำหนดเปอร์เซนต์ทางด้านต่ำและสูงเท่ากับ 1 เปอร์เซนต์ของจำนวนจุดภาพทั้งหมด

จากรูปที่ 2.18 จะให้ค่าคอนทราสของกลุ่มข้อมูลส่วนใหญ่ได้อย่างถูกต้องโดยมีลำดับขั้นตอนการประมวลผลดังรูปที่ 2.19



รูปที่ 2.19 โฟลว์ชาร์ตลำดับขั้นตอนการดึงเชิงเส้นอย่างอัตโนมัติด้วยค่าสูงสุดและต่ำสุดที่หาได้จากการกำหนดเปอร์เซนต์ทางด้านต่ำและสูง

จากโฟลว์ชาร์ตลำดับขั้นตอนการดึงอย่างเชิงเส้นรูปที่ 2.19 เริ่มจากการกำหนดค่าเปอร์เซนต์จำนวนจุดทางด้านต่ำและสูงที่ต้องการจากนั้นทำการสร้างฮีสโตแกรมรวม ความหมายของฮีสโตแกรมรวมคือ จะทำการตรวจสอบและนับจำนวนของจุดค่าระดับสีเทาต่าง ๆ ในภาพทั้งสามแบนด์ ผลจากการนับทั้งหมดจะนำมาไว้ที่ตัวเก็บการนับที่เป็นแอร์เรย์เดียวกัน เมื่อได้ฮีสโตแกรมรวมก็ทำการคำนวณจำนวนจุดทางด้านต่ำและสูงตามค่าเปอร์เซนต์ที่กำหนดไว้ ตัวอย่างเช่น 5% ของทางด้านต่ำเท่ากับ 9830.4 จุด เมื่อจำนวนจุดทั้งหมดเท่ากับ 256×256×3 สำหรับภาพขนาด 256×256 ของภาพ 3 แบนด์ เป็นต้น จากนั้นทำการหาผลรวมของจำนวนจุดภาพทางด้านต่ำและสูงในฮีสโตแกรมรวมจนได้จำนวนจุดที่ต้องการจะได้ค่า x_{min} และค่า x_{max} ที่ด้านต่ำและสูงตามลำดับ ต่อไปทำการสร้างตาราง lookup table ของฟังก์ชันการดึงเชิงเส้นตามสมการที่ (2.7) และ (2.8) โดยมีค่า x_{min} , x_{max} จากที่หามาได้และค่า y_{min}=0 , y_{max}=255 เพื่อให้มีการดึงสู่แกนใหม่อย่างสูงสุด และขั้นตอนสุดท้ายเป็นการเปิดตารางสำหรับทุก ๆ จุดภาพของภาพทั้งสามแบนด์ก็จะได้ภาพที่ได้รับการปรับปรุงคอนทราสที่มีคอนทราสสูงตามต้องการ ตัวอย่างผลลัพธ์ภาพถ่ายดาวเทียมของระบบ TM แสดงได้ในรูปที่ 2.20



รูปที่ 2.20 แสดงตัวอย่างผลลัพธ์ภาพในระบบ TM ของการปรับปรุงคอนทราสด้วยการดึงอย่างเชิงเส้นที่มีการใช้ค่าเปอร์เซนต์ของจำนวนจุดภาพทางด้านต่ำและสูงมาช่วย

2.5 บทสรุป

จากการปรับปรุงคอนทราสของภาพโดยวิธีการดึงอย่างเชิงเส้นที่นำค่าเปอร์เซนต์จำนวนจุดภาพบนฮีสโตแกรมมาใช้ในรูปแบบกึ่งอัตโนมัตินั้น ช่วยให้ได้ภาพที่มีคอนทราสของภาพสูงตามต้องการ ดังในรูปที่ 2.20 และจากการใช้การเปิดตารางมาช่วยในการแปลงก็ทำให้ได้ความเร็วที่สูงมากขึ้น ภาพที่ได้นี้จะได้นำไปใช้เป็นภาพต้นแบบในบทต่อ ๆ ไป โดยรูปแบบข้อมูลภาพที่ใช้เป็นรูปแบบบิทแมพ ที่ได้นำภาพถ่ายดาวเทียมและภาพถ่ายทั่ว ๆ ไปในรูปแบบแอสกี้มาทำการจัดรูปแบบและใส่หัวไฟล์ หัวข้อมูลต่าง ๆ เพื่อใช้ในการแสดงผลภาพ อ่านภาพ เก็บผลภาพ ในลักษณะมาตรฐานได้อย่างสะดวกรวดเร็วและมีความเร็วสูง และอีกทั้งเพื่อที่จะให้เป็นมาตรฐานในการส่งเข้าโปรแกรมประยุกต์อื่น ๆ ได้ด้วย สำหรับโปรแกรมต่าง ๆ ที่เขียนแสดงดังภาพผนวกโดยใช้ภาษาซีและตัวแปลภาษาเป็นบอร์แลนด์บนวินโดว์เวอร์ชัน 3.1

บทที่ 3

การปรับปรุงรายละเอียดขอบภาพด้วยการเสริมข้อมูลเกรเดียนท์

Edge enhancement by adding gradient informations

3.1 คำนำ

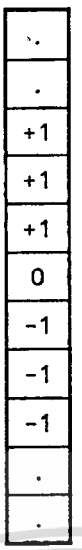
การปรับเปลี่ยนความคมชัดของขอบภาพนั้นเป็นวิธีที่นิยมใช้กันมากและมีประสิทธิภาพสูงจะช่วยทำให้เพิ่มรายละเอียดของข้อมูลภาพถ่ายดาวเทียมให้มีความคมชัดมากยิ่งขึ้น ซึ่งสามารถทำได้โดยการเพิ่มหรือเสริมข้อมูลเกรเดียนท์เข้าไปในภาพต้นแบบทำให้ภาพมีความเด่นชัดมากยิ่งขึ้น หรือแสดงให้เห็นถึงผลต่างของจุดภาพที่อยู่ใกล้เคียงกัน การหาข้อมูลเกรเดียนท์หรือการตรวจจับขอบภาพใด ๆ สามารถจะกระทำได้ง่ายในสปาเชียนโดเมน โดยการทำคอนโวลูชันระหว่างเทมเพล็ทของการหาขอบภาพกับภาพต้นแบบ ซึ่งมีอยู่หลายวิธีด้วยกัน สามารถหารายละเอียดเพิ่มเติมได้จากเอกสารอ้างอิง [5,6] ซึ่งจะมีประโยชน์ในการตีความภาพโดยทั่วไป หรือเพื่อใช้ประโยชน์ในการเลือกจุดควบคุมภาคพื้นดิน (ground control point) ในการแก้ไขความผิดปกติทางรูปทรงเรขาคณิต (geometric correction) ของภาพถ่ายดาวเทียม เนื่องจากปัญหาที่เกิดขึ้นคือจุดควบคุมภาคพื้นดินบนภาพถ่ายดาวเทียมมักจะไม่เด่นชัด ดังนั้นการทำเกรเดียนท์จึงเป็นทางออกที่ดีที่สามารถระบุตำแหน่งเพื่อใช้ในการเปรียบเทียบตำแหน่งพิกัดระหว่างภาพถ่ายดาวเทียมกับตำแหน่งในแผนที่ที่สอดคล้องได้อย่างเที่ยงตรง และอย่างง่ายดาย

3.2 การเพิ่มความคมชัดของขอบภาพโดยการเสริมข้อมูลเกรเดียนท์ โดยวิธีของ Shettigara และ Odins

วิธีการของ Shettigara และ Odins [10] ก็เป็นอีกวิธีการหนึ่งในการเพิ่มความคมชัดให้กับขอบภาพที่เป็นการใช้เทมเพล็ททางแนวนอน และแนวตั้งดังแสดงในรูปที่ 3.1

.	.	+1	+1	+1	0	-1	-1	-1	.	.

(a) เทมเพล็ททางแนวนอน



(b) เท็มเพล็ททางแนวตั้ง

รูปที่ 3.1 แสดงเท็มเพล็ททางแนวนอนและแนวตั้งในวิธีการของ Shettigara และ Odins

จากทฤษฎี เกรเดียนท์ ณ จุดภาพใด ๆ สามารถคำนวณได้ดังนี้

สำหรับการตรวจจับทางแนวนอน ค่าเฉลี่ยความเข้มสำหรับจำนวน n จุด บนด้านลบ (negative side) ของจุดภาพใด ๆ คือ

$$I_{h-}(i,j) = \frac{1}{n}\{I(i-n,j) + I(i-n+1,j)+...+I(i-1,j)\}$$
$$= \frac{1}{n}\sum_{k=1}^{n} I(i-k,j) \qquad (3.1)$$

และทางด้านบวก (positive side) ของจุดภาพใด ๆ คือ

$$I_{h+}(i,j) = \frac{1}{n}\{I(i+1,j) + I(i+2,j)+...+I(i+n,j)\}$$
$$= \frac{1}{n}\sum_{k=1}^{n} I(i+k,j) \qquad (3.2)$$

เมื่อ $I(i,j)$ คือ ค่าความเข้มของจุดภาพใด ๆ

ดังนั้นผลต่างของสมการที่ (3.1) กับ (3.2) จะได้

$$\Delta_h(i,j) = \left|I_{h+}(i,j) - I_{h-}(i,j)\right| \qquad (3.3)$$

เพื่อชดเชยสำหรับเทมเพล็ทที่มีขนาดใหญ่มาก ๆ ต้องทำการหาค่าเฉลี่ยสำหรับจุดภาพใด ๆ ดังนั้นในทิศทางตามแนวนอนค่าเฉลี่ยของเกรเดียนท์จากสมการที่ (3.3) คือ

$$\Delta_{h'}(i,j) = \frac{1}{N}\sum_{n=1}^{N} \Delta_h(i,j) \qquad (3.4)$$

เมื่อ N คือจำนวนจุดที่ต้องการเฉลี่ย และในลักษณะทำนองเดียวกัน สำหรับทิศทางตามแนวตั้ง ค่าเฉลี่ยเกรเดียนท์ คือ

$$\Delta_{v'}(i,j) = \frac{1}{N}\sum_{n=1}^{N} \Delta_v(i,j) \qquad (3.5)$$

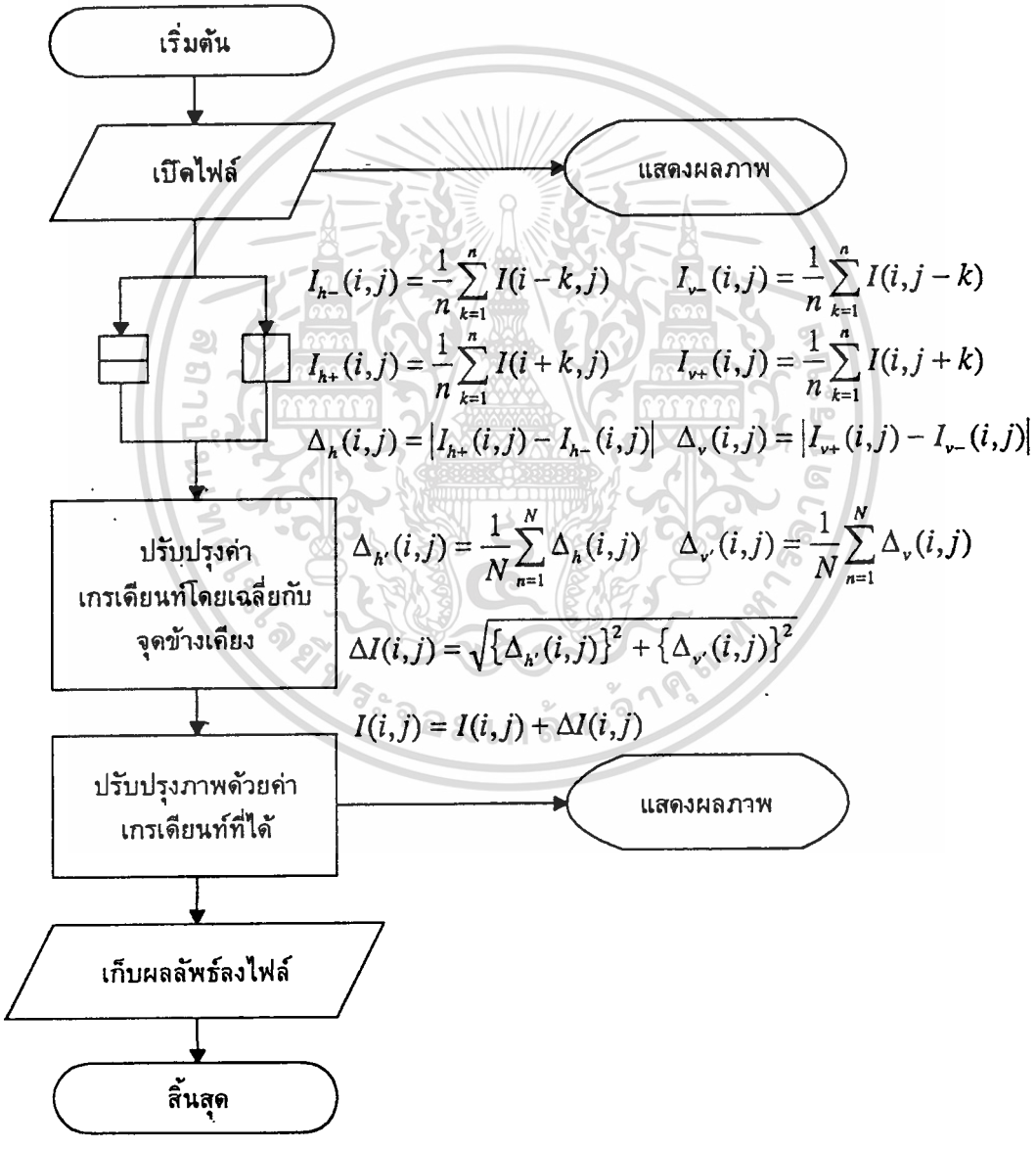
ดังนั้นค่าเฉลี่ยของเกรเดียนท์ ใน 2 มิติ สามารถคำนวณได้ดังนี้

$$\Delta I(i,j) = \sqrt{\{\Delta_{h'}(i,j)\}^2 + \{\Delta_{v'}(i,j)\}^2} \qquad (3.6)$$

เมื่อปรับปรุงแก้ไขค่าระดับสีเทา ณ จุดภาพใด ๆ I_i ในรูปแบบเชิงเส้นก็จะเป็นสมการดังนี้

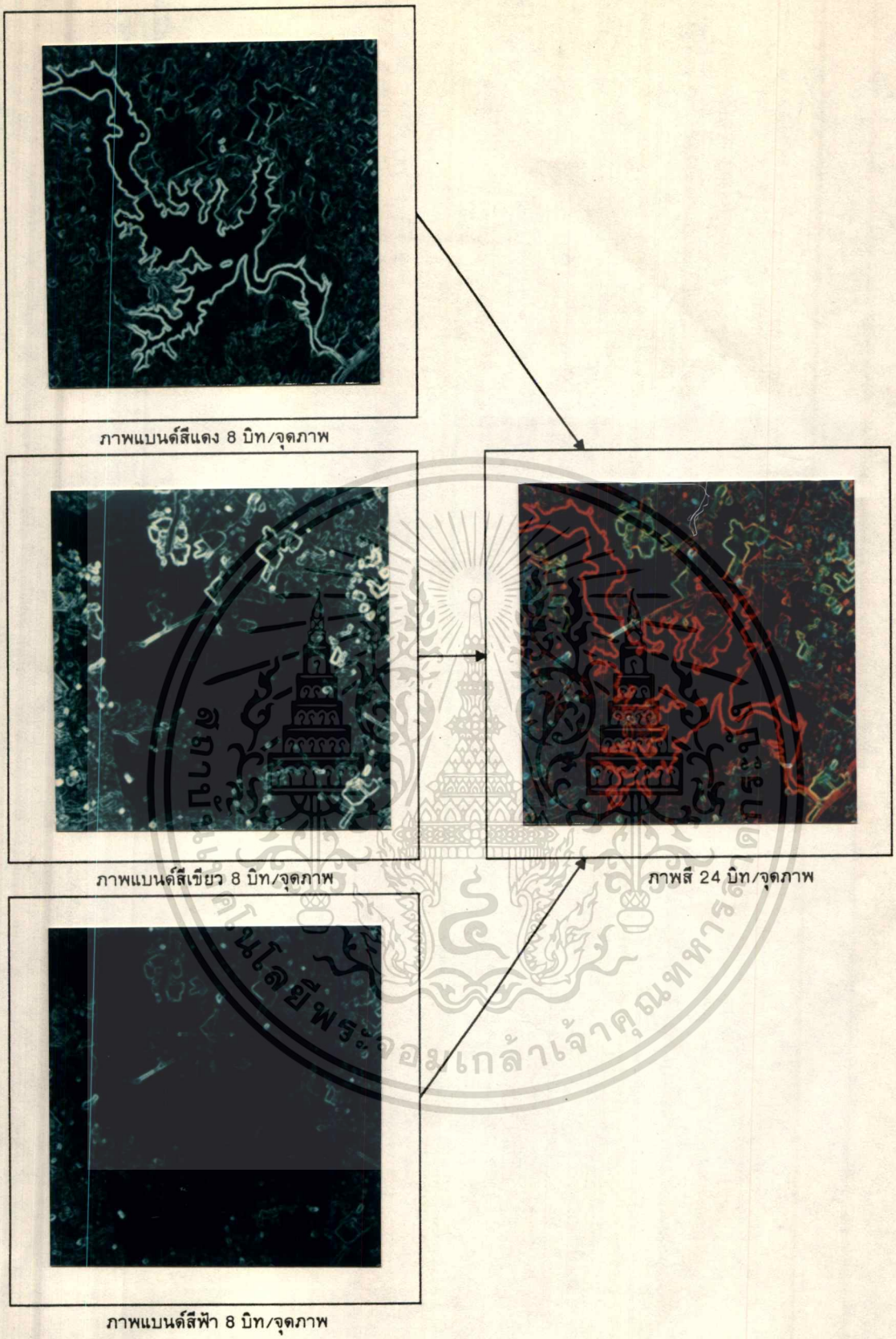
$$I(i,j) = I(i,j) + \Delta I(i,j) \qquad (3.7)$$

ลำดับขั้นตอนการปรับปรุงรายละเอียดของขอบภาพโดยวิธีของ Shettigara และ Odins สรุปได้ดังรูปที่ 3.2

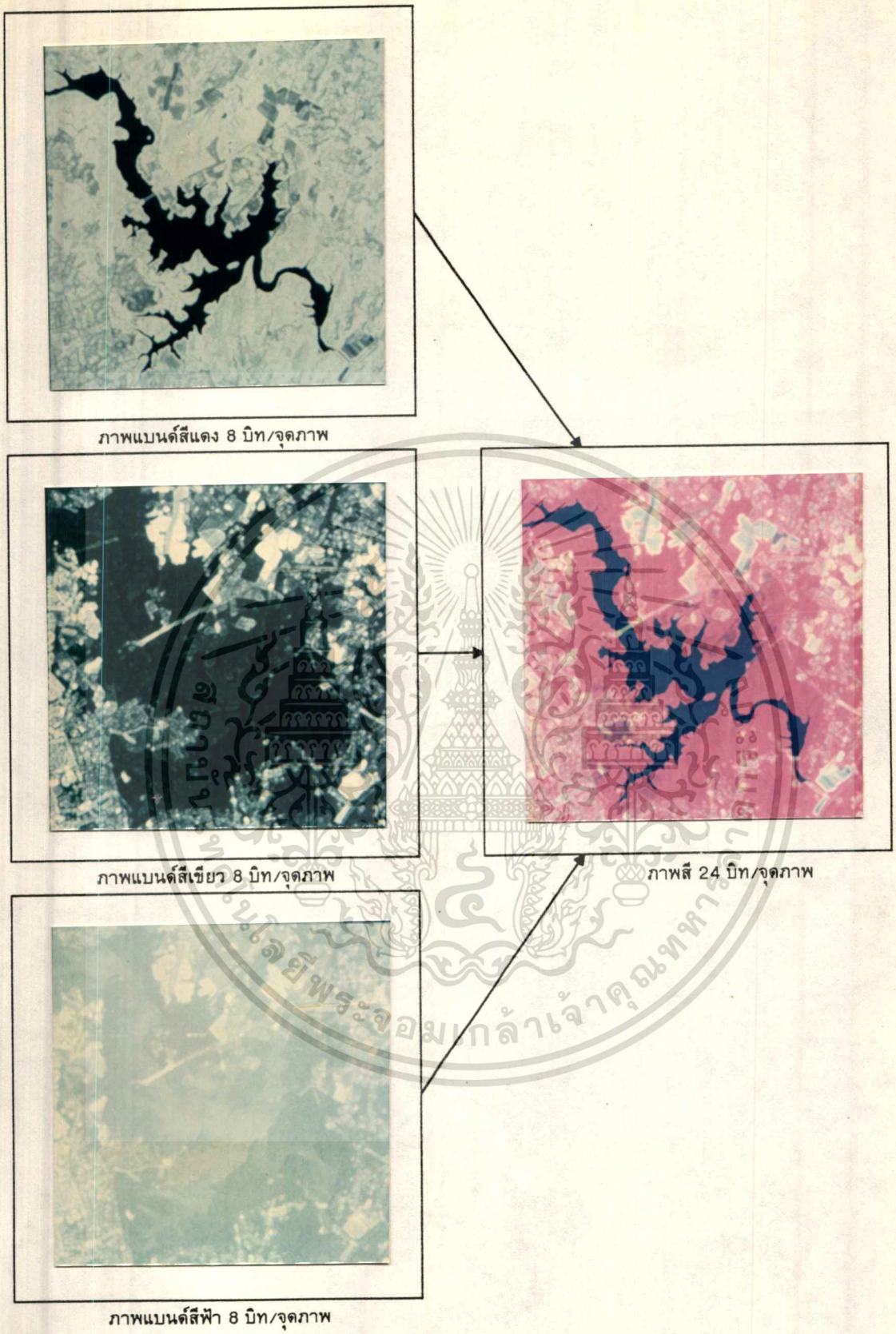


รูปที่ 3.2 แสดงโฟล์วชาร์ตลำดับขั้นตอนการปรับปรุงรายละเอียดของขอบภาพ โดยวิธีการเสริม ข้อมูลเกรเดียนท์ของ Shettigara และ Odins

จากรูปที่ 3.2 ในการหาผลตอบสนองของขอบภาพนั้น Shettigara และ Odins ได้หลีกเลี่ยงวิธีการนำเทมเพล็ทมาทำคอนโวลูชันเพื่อหาขอบของภาพ ทั้งนี้เพื่อให้มีความเร็วในการประมวลผลสูงถึงแม้เทมเพล็ทจะมีขนาดใหญ่ ในขั้นตอนการประมวลผลเริ่มจากการหาผลตอบสนองทางแนวนอนและแนวตั้ง โดยแบ่งช่วงบวกและช่วงลบออกจากกันแล้วทำการหาผลรวมของแต่ละส่วน ผลต่างที่ได้เมื่อนำมาลบกันจะกลายเป็นผลตอบสนองของขอบ เมื่อนำมาหาค่าสัมบูรณ์ (absolute) จะทำให้ได้ขอบภาพที่ไม่มีทิศทาง ทำเช่นนี้ทั้งทางแนวนอนและแนวตั้ง จากนั้นทำการปรับปรุงค่าเกรเดียนท์ที่ได้ด้วยการหาค่าเฉลี่ยเพื่อเกลี่ยความรุนแรงของขอบเกรเดียนท์ให้ลดลงเล็กน้อย นำเกรเดียนท์ทางแนวนอนและแนวตั้งที่ได้มาหาขนาดของเกรเดียนท์รวมก็จะได้เกรเดียนท์ที่สมบูรณ์ดังแสดงได้ดังรูปที่ 3.3 ซึ่งเมื่อนำมาทำการปรับปรุงรายละเอียดของขอบภาพแบบเชิงเส้นแสดงได้ดังรูปที่ 3.4



รูปที่ 3.3 แสดงตัวอย่างขอบภาพจากวิธีการเสริมข้อมูลเกรเดียนท์ของ Shettigara และ Odins เมื่อใช้ค่า n = 1



รูปที่ 3.4 แสดงตัวอย่างภาพผลลัพธ์จากวิธีการเสริมข้อมูลเกรเดียนท์ของ Shettigara และ Odins เมื่อใช้ค่า n = 1

3.3 การเพิ่มความคมชัดของขอบภาพโดยวิธีการเสริมข้อมูลเกรเดียนท์ โดยใช้เทมเพล็ทแบบ 12 ทิศทาง

เนื่องจากการใช้วิธีการของ Shettigara และ Odins นั้นมีการตอบสนองต่อลักษณะของขอบแบบต่าง ๆ ที่น้อยเกินไปไม่เหมาะสำหรับภาพถ่ายดาวเทียมที่มีความละเอียดและลักษณะของขอบแบบต่าง ๆ มากมายหลายรูปแบบดังนั้นเพื่อเพิ่มประสิทธิภาพการตอบสนองต่อภาพถ่ายดาวเทียมที่มีความละเอียด และลักษณะลายเส้นที่หลากหลายมากยิ่งขึ้นจึงมีการใช้ 12 เทมเพล็ทสำหรับ 12 ทิศทาง [9] ในการทดสอบขอบของภาพในลักษณะต่าง ๆ ที่จะเป็นไปได้ โดยมีขนาดหน้าต่างเป็น 7x7 ซึ่งเทมเพล็ทเหล่านี้ แสดงให้เห็นได้ในรูปที่ 3.5

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 +1 +1 +1 0 -1 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 +1 0 0 0 0 0 0 +1 0 0 0 0 0 0 +1 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 -1 0 0 0 0 0 0 -1 0 0 0
+1 0 0 0 0 0 0 0 +1 0 0 0 0 0 0 0 +1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 -1	0 0 0 0 0 0 +1 0 0 0 0 0 +1 0 0 0 0 0 +1 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 -1 0 0 0 0 0 -1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 +1 +1 0 0 0 0 0 0 0 +1 0 -1 0 0 0 0 0 0 0 -1 -1 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 0 0 0 0 0 0 0 0 +1 +1 0 0 -1 0 +1 0 0 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 +1 0 0 0 0 0 0 +1 0 0 0 0 0 0 0 +1 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 -1 0 0	0 0 0 0 +1 0 0 0 0 0 0 +1 0 0 0 0 0 +1 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 -1 0 0 0 0 0 0 -1 0 0 0 0
0 +1 0 0 0 0 0 0 +1 0 0 0 0 0 0 0 +1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 -1 0	0 0 0 0 0 +1 0 0 0 0 0 0 +1 0 0 0 0 0 +1 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 -1 0 0 0 0 0 0 -1 0 0 0 0 0

0	0	0	0	0	0	0		0	0	0	0	0	0	0
+1	+1	0	0	0	0	0		0	0	0	0	0	+1	+1
0	0	+1	0	0	0	0		0	0	0	0	+1	0	0
0	0	0	0	0	0	0		0	0	0	0	0	0	0
0	0	0	0	-1	0	0		0	0	-1	0	0	0	0
0	0	0	0	0	-1	-1		-1	-1	0	0	0	0	0
0	0	0	0	0	0	0		0	0	0	0	0	0	0

รูปที่ 3.5 แสดงเท็มเพล็ท 12 แบบ ที่ใช้ในการหาค่าเกรเดียนท์

ดังนั้น จากสมการที่ (3.4) และ (3.5) ก็จะกลายเป็น

$$\Delta I^m(i,j) = \frac{1}{N}\sum_{n=1}^{3}\Delta I^m(i,j) \qquad (3.8)$$

เมื่อ N มีค่าเท่ากับ 3

m คือ ตารางในแต่ละเท็มเพล็ท ซึ่งแปรเปลี่ยนได้จากตาราง 1-12

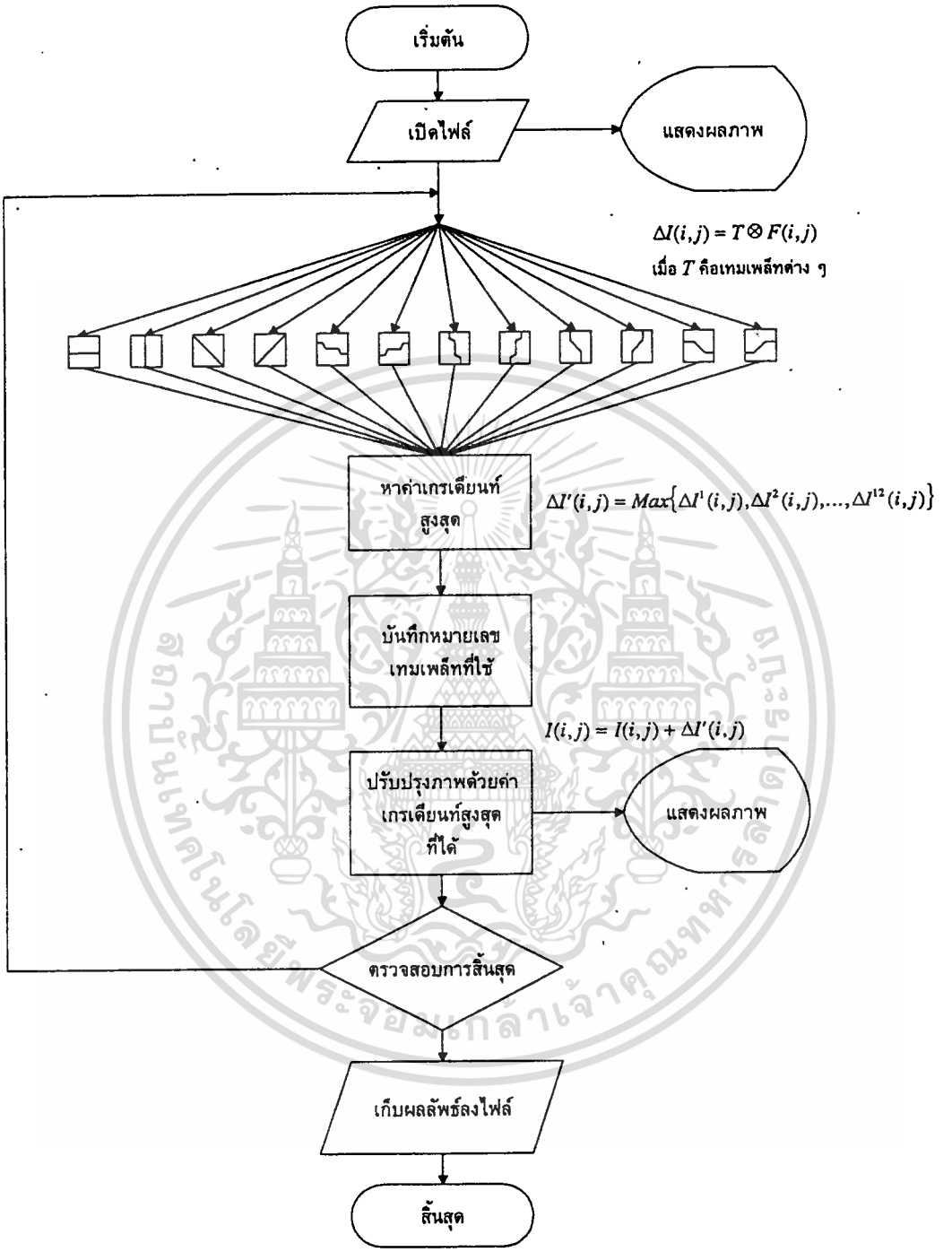
ค่าเกรเดียนท์ของแต่ละจุดภาพที่คำนวณได้นั้น จะเลือกเอาเฉพาะค่าสูงสุด จากตารางหน้าต่างทั้ง 12 แบบ ดังนั้นเลือกได้จาก

$$\Delta I(i,j) = \max\{\Delta I^1(i,j), \Delta I^2(i,j), .., \Delta I^{12}(i,j)\} \qquad (3.9)$$

การปรับปรุงภาพอย่างเชิงเส้นให้กับจุดภาพใด ๆ $I(i,j)$ ก็จะได้สมการดังนี้

$$I(i,j) = I(i,j) + \Delta I(i,j) \qquad (3.10)$$

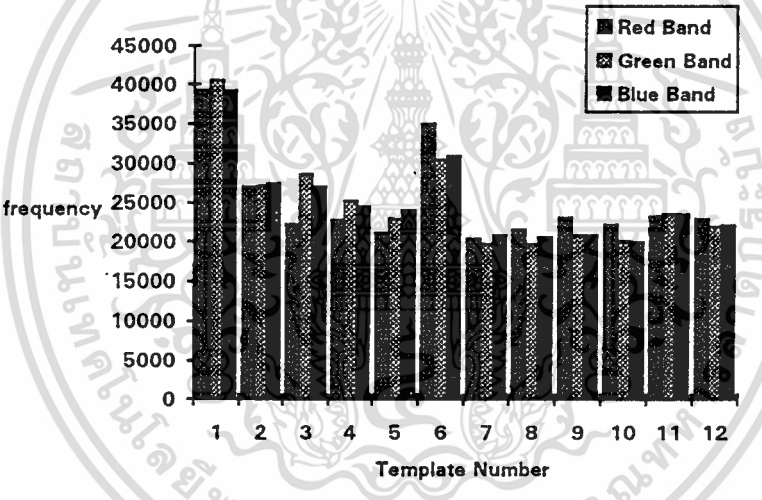
ลำดับขั้นตอนการปรับปรุงรายละเอียดของขอบภาพโดยวิธี 12 เทมเพล็ทสามารถสรุปได้ดังรูปที่ 3.6



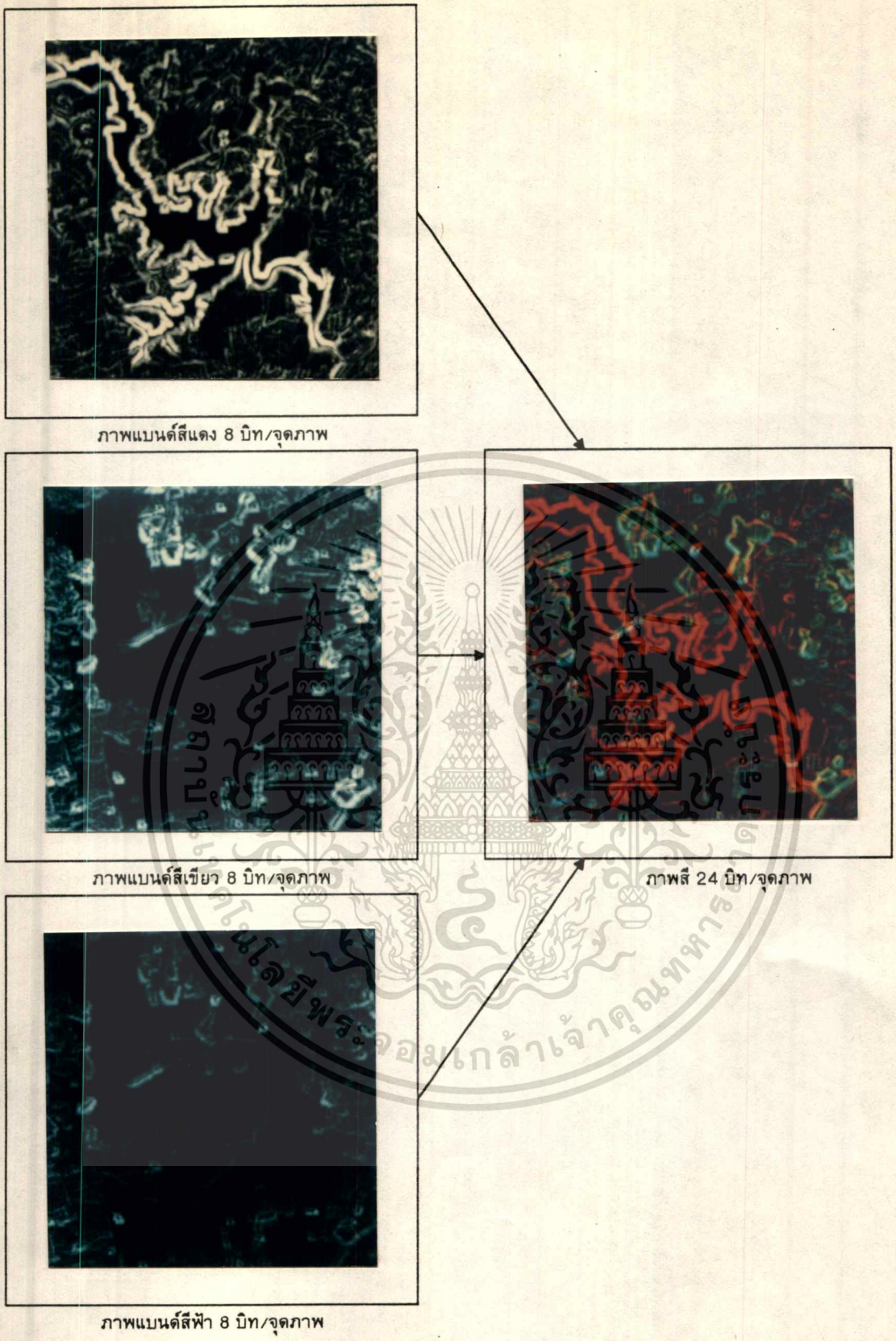
รูปที่ 3.6 แสดงโฟล์วชาร์ตลำดับขั้นตอนการปรับปรุงรายละเอียดของขอบภาพ โดยใช้เทมเพล็ทแบบ 12 ทิศทาง

จากรูปที่ 3.6 ในการหาผลตอบสนองของขอบภาพนั้นขบวนการประมวลผลเริ่มจากการใช้ 12 เทมเพล็ทมาทดลองทำคอนโวลูชัน เมื่อได้ผลตอบสนองครบทั้ง 12 ทิศทางก็นำมาหาค่าสัมบูรณ์ (absolute) จะทำให้ได้ขอบภาพที่ไม่มีทิศทางและมีการตอบสนองได้ถึง 24 เทมเพล็ท จากนั้นนำมาเปรียบเทียบกันเพื่อเลือกหาค่าเกรเดียนท์ที่มีผลตอบสนองสูงสุด จะได้หมายเลขเทมเพล็ทที่ต้องการ ทำการนับความถี่ในการถูกเรียกใช้และบันทึกหมายเลขไว้เพื่อเป็นข้อมูลในการพล็อตกราฟตอบสนองต่อขอบในทิศทางต่าง ๆ ของภาพ สำหรับกราฟที่แสดงให้เห็นถึงจำนวนเปอร์เซนต์ความถี่ของแต่ละเท็มเพล็ท สามารถแสดงได้ดังรูปที่ 3.7 ซึ่งจากกราฟทำให้สามารถเรียนรู้ถึงทิศทางต่าง ๆ ของขอบภาพใด ๆ ได้ และแสดงความจริงให้เห็นถึงลักษณะของขอบที่มีมากมายกว่าที่จะเป็นเพียงแนวตั้งและแนวนอนเท่านั้น สำหรับภาพขอบการตอบสนองเกรเดียนท์สูงสุดแสดงได้ดังรูปที่ 3.8

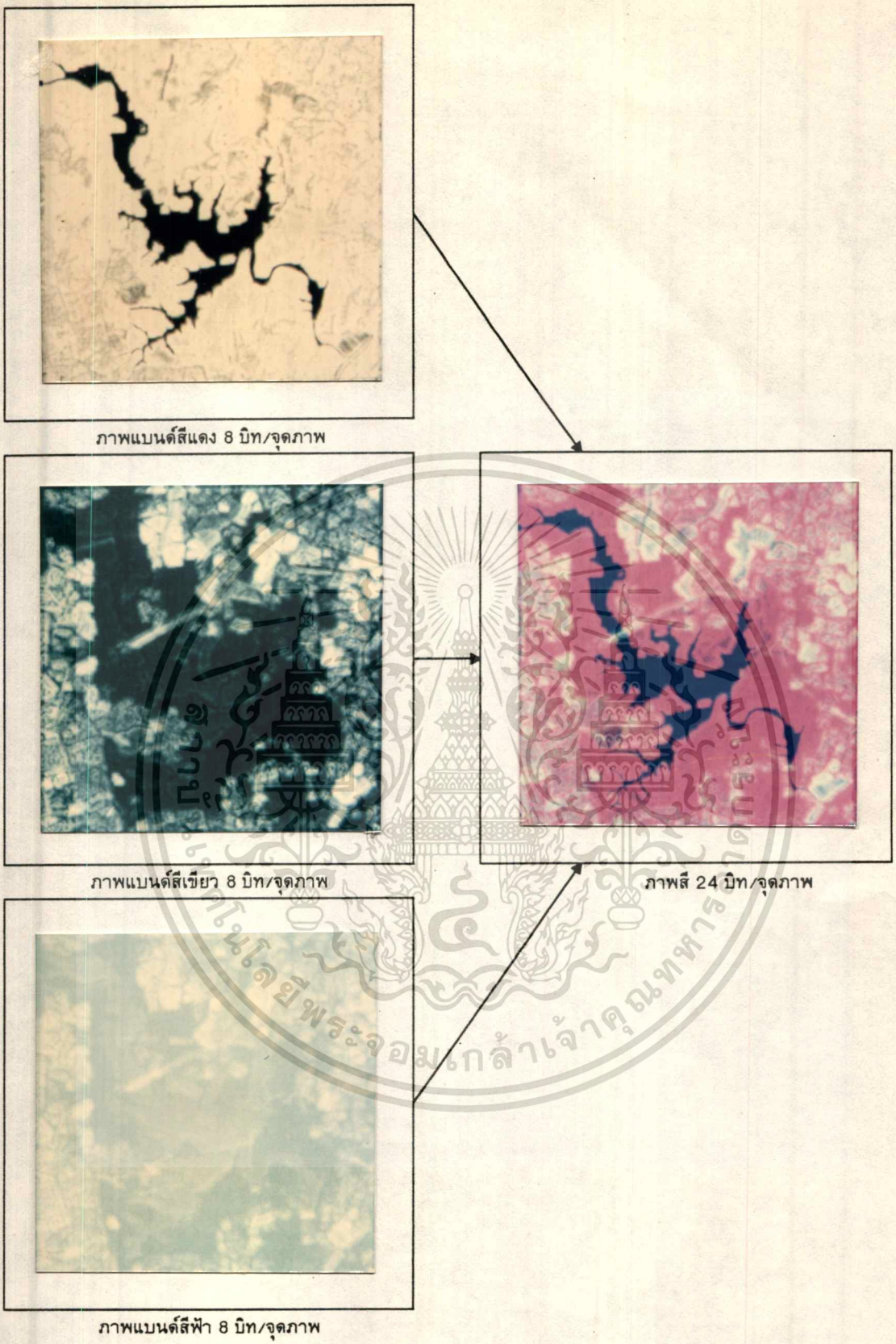
เมื่อนำค่าเกรเดียนสูงสุดที่ได้มาทำการปรับปรุงรายละเอียดของขอบภาพแบบเชิงเส้นแสดงผลลัพธ์ได้ดังรูปที่ 3.9



รูปที่ 3.7 แสดงจำนวนความถี่ในการเรียกใช้เทมเพล็ทหมายเลขต่าง ๆ



รูปที่ 3.8 แสดงตัวอย่างภาพขอบจากวิธีการเสริมข้อมูลเกรเดียนท์ โดยใช้เทมเพล็ทแบบ 12 ทิศทาง



รูปที่ 3.9 แสดงตัวอย่างภาพผลลัพธ์จากวิธีการเสริมข้อมูลเกรเดียนท์ โดยใช้เทมเพล็ทแบบ 12 ทิศทาง

3.4 บทสรุป

ในการปรับปรุงรายละเอียดของขอบภาพนั้น ภาพผลลัพธ์จะมีการเกิดโอเวอร์ชูตตรงบริเวณขอบต่าง ๆ ภายในภาพอย่างรุนแรง ดังนั้นในปรับปรุงความคมชัดของภาพที่แท้จริงแล้วไม่สามารถใช้การเน้นขอบที่รุนแรงเช่นนี้ได้ ดังนั้นต้องมีการหาขบวนการในการหาเกรเดียนท์ที่ให้การตอบสนองที่รุนแรงน้อยกว่านี้ หรือวิธีการที่สามารถควบคุมขนาดของเกรเดียนท์ได้ตามต้องการ

จากการทดลองโดยวิธีการของ Shettigara และ Odins ในเรื่องของความหนาของขอบภาพนั้น พบว่าขึ้นอยู่กับขนาดของเท็มเพล็ทที่นำมาใช้ หากเป็นเทมเพล็ทที่มีขนาดใหญ่ ขอบของเกรเดียนท์ก็จะยิ่งมีความหนามากขึ้น ดังนั้นในการเลือกใช้จึงพยายามใช้ขนาดที่เล็กที่สุดเท่าที่จะเป็นไปได้ สำหรับวิธีการ 12 เทมเพล็ทนั้นเพื่อตอบสนองขอบในหลายรูปแบบ จึงไม่สามารถลดขนาดของเทมเพล็ทได้อีก ขอบที่ได้จะมีความหนามากเนื่องจากใช้เทมเพล็ทขนาดใหญ่ถึง 7×7 ทำให้เกิดข้อเสียที่สำคัญอีกอย่างหนึ่ง คือใช้เวลาในการประมวลผลมากตามมา ดังนั้นต้องหาขบวนการที่หากมีการใช้วิธีการคอนโวลูชัน ในการหาผลตอบสนองของขอบภาพ ก็จะต้องเป็นขบวนการที่ใช้เทมเพล็ทขนาดเล็ก เพื่อให้การตอบสนองของขอบที่มีความบางกว่า และเหตุผลทางด้านความเร็วในการประมวลผล

และจากการทดลองด้วยวิธีการทั้งสองนี้ ทำให้ได้พบกับคำว่าขอบเทียมที่เกิดขึ้นในพื้นที่เอกพันธ์ทำให้ข้อมูลภาพทั่ว ๆ ไปเกิดการยกตัวเพิ่มค่าความสว่างขึ้นไป ทำให้ภาพมีความจ้าไปทั้งภาพ เมื่อทำการทดลองโดยใช้ฟังก์ชันอย่างง่ายคือฟังก์ชันสเต็ปหนึ่งหน่วยมาทำการตัดเทรสโฮลด์ตัดขอบเทียมเหล่านั้น ผลปรากฏว่าภาพที่ได้ก็ไม่มีการยกค่าความสว่างของข้อมูลทั่วไป มีการเสริมเฉพาะขอบจริงที่ถูกต้องเท่านั้น วิธีการ 12 เทมเพล็ทก็สามารถใช้วิธีการนี้ในการลดความหนาของขอบได้ แต่ภาพขอบจากการตัดเทรสโฮลด์อย่างง่ายนี้เมื่อนำมาใช้ในการปรับปรุงรายละเอียดของขอบภาพ โดยนำมารวมกับภาพต้นแบบแล้วจะทำให้ภาพขาดความต่อเนื่องระหว่างข้อมูลทั่วไป กับข้อมูลของขอบภาพไป ดังนั้นในการกำจัดขอบเทียมต้องมีการใช้ฟังก์ชันอื่น ๆ ที่ให้ผลของความราบเรียบของรอยต่อได้มากกว่านี้

บทที่ 4

การปรับปรุงรายละเอียดของขอบภาพด้วยภาพผลลบที่เกิดจากการทำให้เรียบ

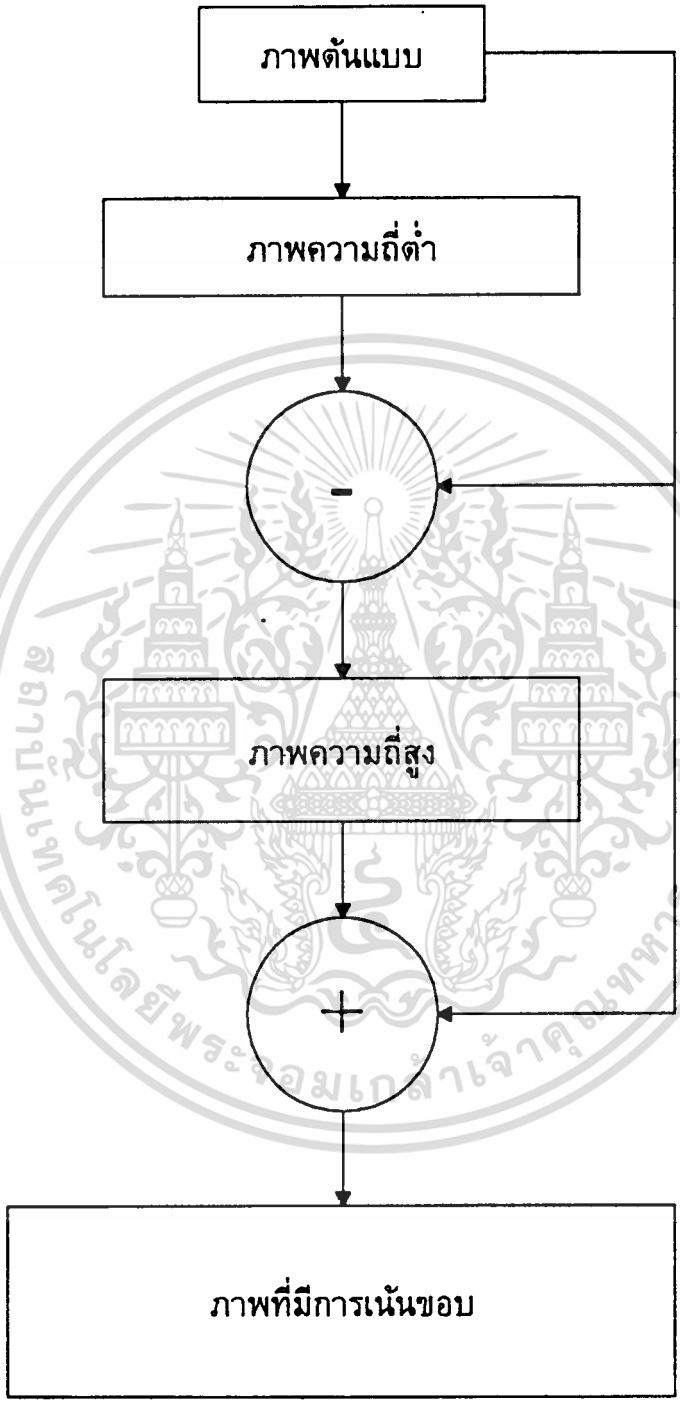
Edge enhancement by using subtracted smoothing images

4.1 คำนำ

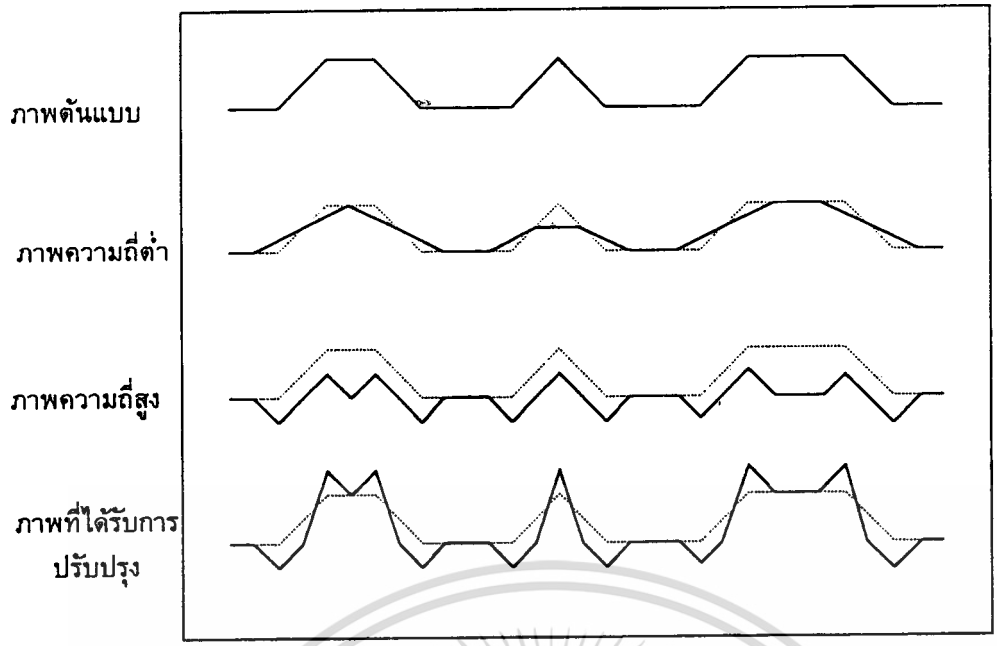
จากบทที่ผ่านมาในการปรับปรุงขอบภาพนั้น วิธีการของ Shettigara และ Odins นั้นใช้เกรเดียนท์เทมเพล็ทแบบแนวตั้งและแนวนอน ทำให้ผลลัพธ์ที่ได้ไม่เหมาะสมกับภาพถ่ายดาวเทียมที่มีรายละเอียดมาก ในลักษณะของลายเส้นในทิศทางต่าง ๆ มากมาย จึงได้หันมาทดลองเพิ่มเทมเพล็ทโดยกลายเป็นเทมเพล็ทแบบ 12 ทิศทาง เพื่อดอบสนองขอบได้หลาย ๆ ลักษณะขึ้น แต่นอกจากผลลัพธ์ที่ได้มาแล้วนั้นสิ่งที่เป็นจุดอ่อนคือความเร็ว ในการประมวลผลภาพถ่ายดาวเทียมในทางปฏิบัติจริงเป็นภาพขนาดใหญ่มาก ดังนั้นต้องทำการเปลี่ยนแนวทางโดยหาวิธีการอื่นที่นอกเหนือไปอีก วิธีการนั้นคือการใช้ภาพผลลบที่เกิดจากการทำให้เรียบ (subtracted smoothing image) [13] ซึ่งเป็นวิธีการที่สามารถรวมวิธีการทั้งหมดให้เป็นตัวกรองในสปาเชียนโดเมนได้เพียงตัวเดียว ซึ่งจะมีความเร็วสูงมาก ดังที่จะได้กล่าวต่อไป

4.2 การปรับปรุงรายละเอียดของภาพโดยการใช้ภาพผลลบที่ได้ทำให้เรียบ

ในการปรับปรุงรายละเอียดของขอบภาพโดยวิธีนี้ทำได้โดยการนำภาพต้นแบบมาทำให้เรียบ (smoothing) จะได้ภาพความถี่ต่ำ จากนั้นนำไปลบออกจากภาพต้นแบบ ผลต่างหรือผลลบที่ได้เรียกว่าภาพผลลบที่เกิดจากการทำให้เรียบ (subtracted smoothing image) นั่นเอง ภาพนี้จะเป็นส่วนของความถี่สูง (high spatial frequency) ซึ่งเป็นข้อมูลของรายละเอียดทั้งหมดของลายเส้นขอบต่าง ๆ ภายในภาพ ดังนั้นเพื่อปรับปรุงรายละเอียดของขอบภายในภาพสามารถทำได้โดยการเพิ่มข้อมูลความถี่สูงให้กับภาพ โดยนำภาพผลลบที่เกิดจากการทำให้เรียบไปรวมกับภาพต้นแบบ จะได้ภาพผลลัพธ์ที่มีการเน้นขอบทำให้มีความคมชัดมากขึ้นได้ ไดอะแกรมขั้นตอนการประมวลผลแสดงได้ดังรูปที่ 4.1 และตัวอย่างแนวความคิดในรูปแบบมิติเดียวจากการปรับปรุงรายละเอียดของขอบภาพแสดงได้ดังรูปที่ 4.2 โดยมีการแสดงลำดับตามขั้นตอนของไดอะแกรมการทำงานทุกขั้นตอน และให้ผลลัพธ์ของการปรับปรุงรายละเอียดของขอบที่มีการเน้นให้เกิดโอเวอร์ชูตตรงขอบของการเปลี่ยนแปลงทำให้ภาพมีความคมชัดมากยิ่งขึ้นได้ตรงตามวัตถุประสงค์ที่ต้องการ



รูปที่ 4.1 แสดงกระบวนการในการเน้นขอบภาพโดยการใช้ภาพผลลบที่ถูกทำให้เรียบ



รูปที่ 4.2 แสดงตัวอย่างแนวความคิดในการเน้นขอบภาพโดยการใช้ภาพผลลบที่ถูกทำให้เรียบในรูปแบบมิติเดียว

ในรายละเอียดของขั้นตอนต่าง ๆ อธิบายได้ดังนี้

ขั้นตอนที่ 1 สร้างภาพความถี่ต่ำ

ในการสร้างภาพความถี่ต่ำทำได้โดยการนำภาพถ่ายดาวเทียมแถบความยาวคลื่นใด ๆ มาผ่านกระบวนการทำให้เรียบ (smoothing) ซึ่งเป็นการหาค่าเฉลี่ยของจุดภาพต่าง ๆ ภายในเทมเพล็ท เพื่อนำมาลงแทนจุดภาพที่อยู่ ณ ตำแหน่งกึ่งกลางของเทมเพล็ท หรือหมายความว่าภาพความถี่ต่ำจะได้มาจากการทำคอนโวลูชันระหว่างเทมเพล็ทกับข้อมูลภาพเดิมนั่นเอง เทมเพล็ทดังกล่าวเป็นเมทริกซ์ขนาด 3X3 ดังนี้

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

(a) (b)

รูปที่ 4.3 แสดงตัวอย่างเท็มเพล็ทต่าง ๆ ที่ใช้ในการทำให้เรียบ

ภาพที่ได้จากกระบวนการทำให้เรียบนี้จะเหลือเฉพาะข้อมูลความถี่ต่ำ ขอบต่าง ๆ ซึ่งเป็นข้อมูลความถี่สูง (high spatial frequency component) จะถูกทำลายไป สมการที่ใช้ในการสร้างภาพความถี่ต่ำแสดงได้ดังนี้

$$S(x,y) = \frac{1}{9}\sum_{j=1}^{3}\sum_{k=1}^{3} T(j,k)F(x+j-2,y+k-2) \qquad (4.1)$$

เมื่อ $x = 1,2,3,...,M$ โดย M เป็นจำนวนจุดภาพต่อหนึ่งเส้นภาพ

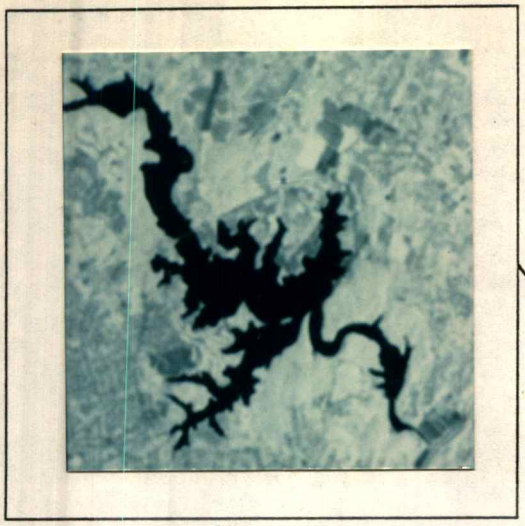
และ $y = 1,2,3,...,N$ โดย N เป็นจำนวนเส้นภาพ

$S(x,y)$ คือ ข้อมูลความถี่ต่ำของผลลัพธ์ที่ได้

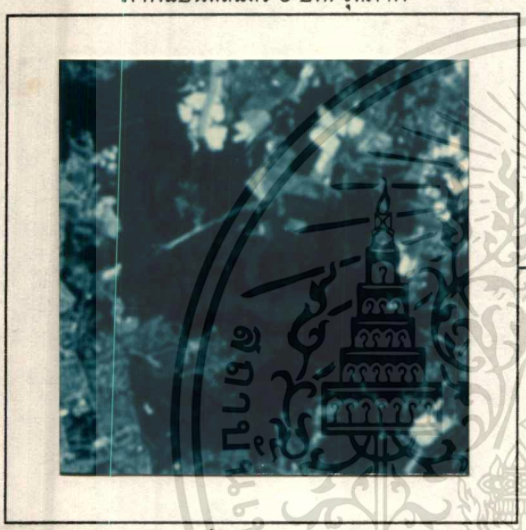
$T(j,k)$ คือ เทมเพล็ทในการทำให้เรียบ

$F(x,y)$ คือ ข้อมูลภาพต้นแบบที่ต้องการประมวลผล

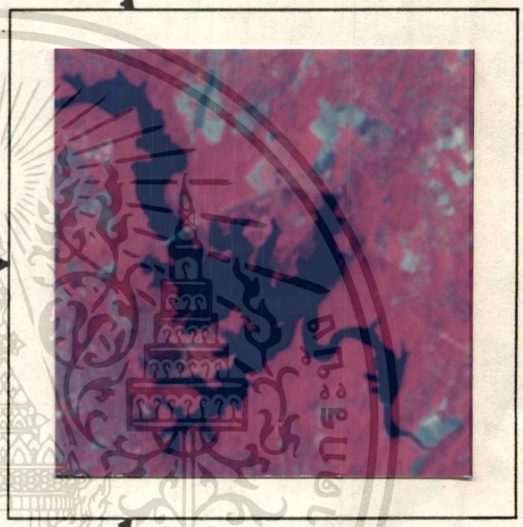
ตัวอย่างผลลัพธ์ของข้อมูลภาพความถี่ต่ำ แสดงได้ดังรูปที่ 4.4



ภาพแบนด์สีแดง 8 บิท/จุดภาพ



ภาพแบนด์สีเขียว 8 บิท/จุดภาพ



ภาพสี 24 บิท/จุดภาพ

ภาพแบนด์สีฟ้า 8 บิท/จุดภาพ

รูปที่ 4.4 แสดงผลลัพธ์จากขบวนการทำให้เรียบที่ทำให้เกิดภาพความถี่ต่ำ

ขั้นตอนที่ 2 สร้างภาพความถี่สูง

ในการสร้างภาพความถี่สูงทำได้โดยการนำภาพความถี่ต่ำที่ได้มาลบออกจากภาพต้นแบบดังสมการข้างล่างนี้

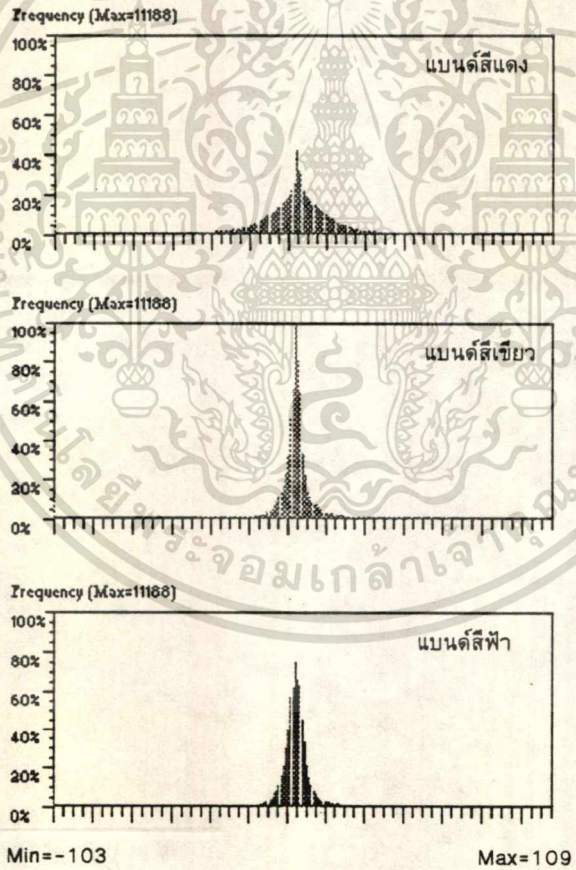
$$H(x,y) = F(x,y) - S(x,y) \qquad (4.2)$$

เมื่อ $H(x,y)$ คือ ภาพความถี่สูงที่ได้

$F(x,y)$ คือ ภาพต้นแบบ

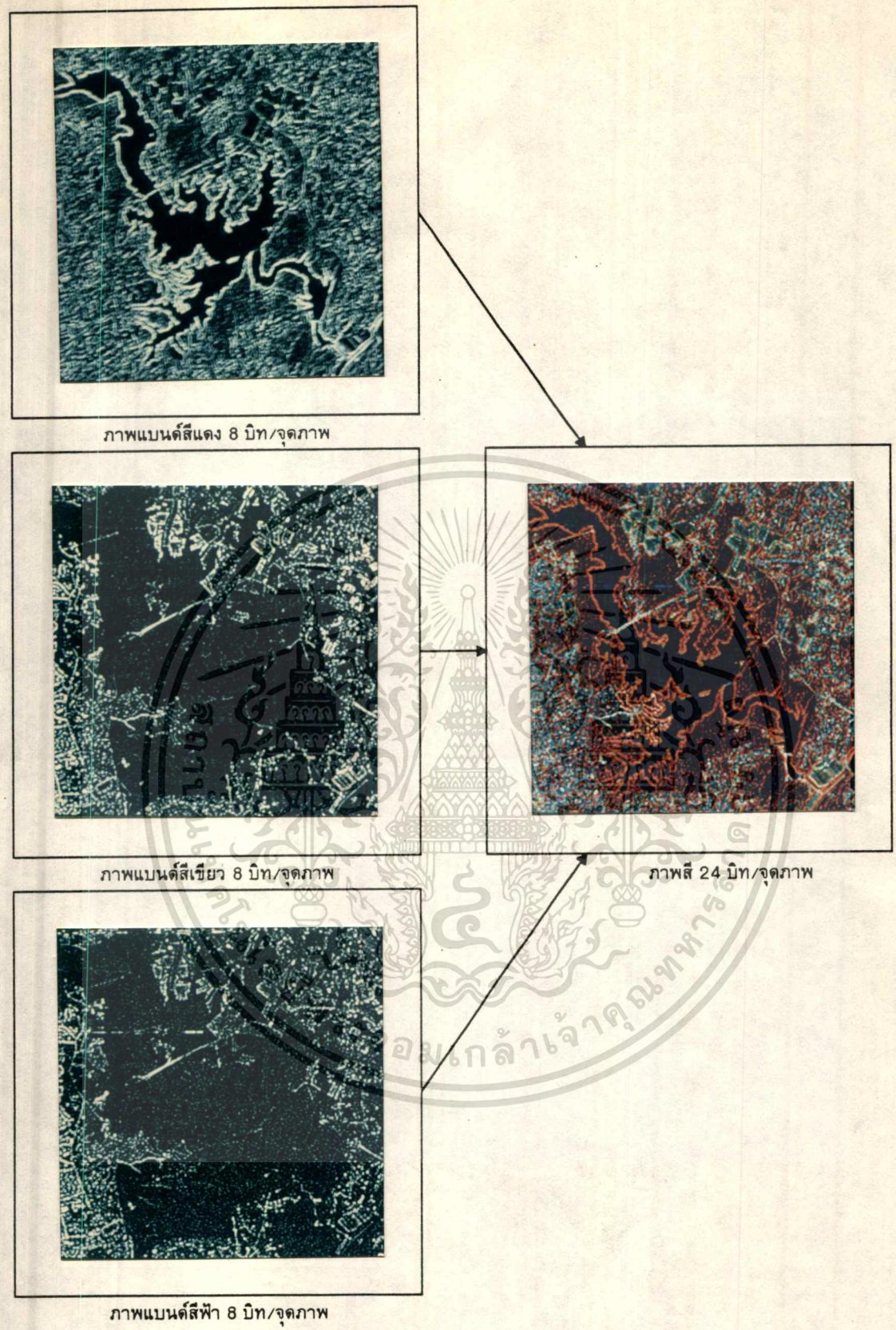
$S(x,y)$ คือ ภาพความถี่ต่ำที่ได้จากขั้นตอนที่ 1

ภาพความถี่สูงที่ได้จะเป็นรายละเอียดทั้งหมดของขอบ ลายเส้นต่าง ๆ ภายในภาพ ผลต่างที่ได้นี้จะมีค่าเลยออกนอกย่านปกติโดยจะมีค่าที่เป็นไปได้อยู่ระหว่าง -255 ถึง +255 ตัวอย่างฮีสโตแกรมของภาพความถี่สูงที่ได้แสดงดังรูปที่ 4.5



รูปที่ 4.5 แสดงตัวอย่างฮีสโตแกรมของภาพความถี่สูงที่มีค่าเลยออกนอกย่านปกติ

ฮีสโตแกรมที่ได้จะมีความสมมาตรรอบแกนศูนย์ ซึ่งเมื่อทำการดึงอย่างเชิงเส้นให้มีค่ากลับเข้ามาอยู่ในย่านปกติที่อยู่ในย่าน 0 ถึง 255 เพื่อการแสดงผลภาพ แสดงได้ดังรูปที่ 4.6



รูปที่ 4.6 แสดงตัวอย่างผลลัพธ์ของภาพความถี่สูงที่ผ่านการดึงอย่างเชิงเส้นเพื่อให้ค่ากลับเข้ามาอยู่ในย่านปกติ

ขั้นตอนที่ 3 การเน้นขอบ

ในการเน้นขอบทำได้โดยนำภาพความถี่สูงที่ได้รวมเข้ากับภาพต้นแบบ ดังนี้

$$E(x,y) = F(x,y) + H(x,y) \qquad (4.3)$$

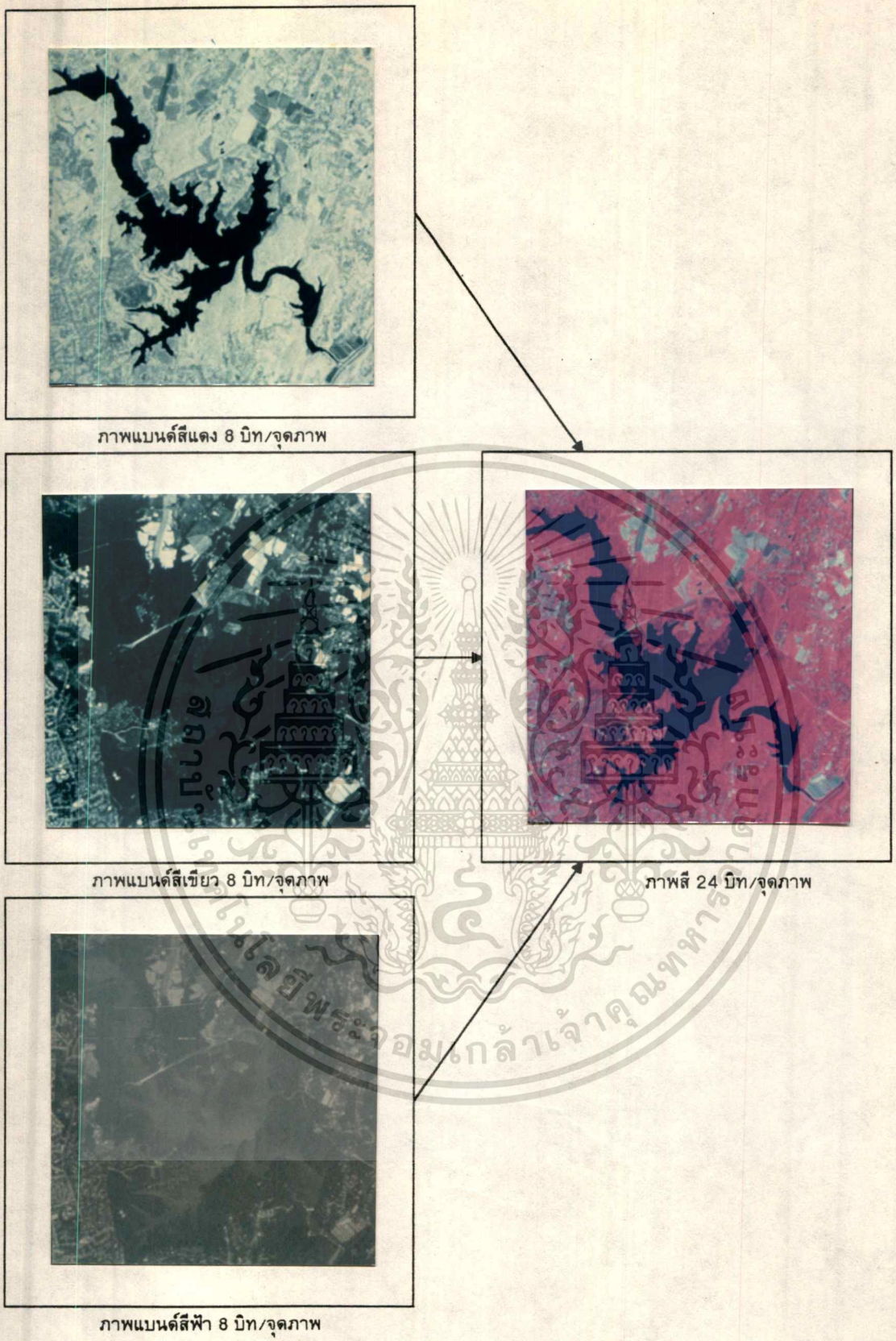
เมื่อ $E(x,y)$ คือ ภาพที่ได้รับการปรับปรุงขอบภาพ

$F(x,y)$ คือ ภาพต้นแบบ

$H(x,y)$ คือ ภาพความถี่สูงที่ได้จากขั้นตอนที่ 2

สำหรับความหมายของความสมมาตรรอบแกนศูนย์ของภาพความถี่สูงหมายความว่าในลายเส้นบนภาพ 1 เส้น ส่วนประกอบของความถี่สูงจะเกิดขึ้นทั้งส่วนที่เป็นขอบด้านนอกและด้านใน ทำให้เราสามารถทำการเน้นขอบได้เฉพาะขอบนอกหรือขอบใน หรือทำการเน้นทั้งหมดได้ตามต้องการ ซึ่งในที่นี้เลือกการเน้นทั้งหมด เพราะว่าถ้าเป็นขอบด้านนอก เมื่อนำมารวมกับภาพต้นแบบจะทำให้มีค่าลดลง ถ้าเป็นขอบด้านใน เมื่อนำมารวมกับภาพต้นแบบทำให้มีค่าเพิ่มขึ้น คือทำให้เส้นที่ต้องการเน้นมีความคมชัดมากยิ่งขึ้น

ตัวอย่างภาพผลลัพธ์ที่ได้รับการปรับปรุงขอบภาพ แสดงได้ดังรูปที่ 4.7



รูปที่ 4.7 แสดงตัวอย่างภาพผลลัพธ์ที่ได้รับการปรับปรุงขอบภาพ

ในการประมวลผลจะทำการคำนวณไปตามลำดับขั้นตอนดังที่กล่าวมา และอีกทั้งยังสามารถปรับปรุงขบวนการเพื่อเพิ่มความเร็วในการประมวลผลโดยใช้รูปแบบการกรอง โดยวิธีการทำคอนโวลูชันในสปาเชียลโดเมนท์ ดังที่จะกล่าวในหัวข้อต่อไป

4.3 การปรับปรุงรายละเอียดของขอบภาพโดยใช้ภาพผลลบที่ถูกทำให้เรียบด้วยตัวกรองเพียงตัวเดียว

สมมุติให้ x_4 เป็นจุดภาพที่กำลังต้องการประมวลผล เมื่อผ่านขบวนการทำให้เรียบโดยใช้เทมเพล็ทดังรูปที่ 4.3(a) ผลการทำคอนโวลูชันคือ

$$S_4 = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \otimes \begin{bmatrix}x_0 & x_1 & x_2\\ x_3 & x_4 & x_5\\ x_6 & x_7 & x_8\end{bmatrix}$$

$$= \frac{x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8}{9}$$

การหาความถี่สูงในแต่ละตำแหน่งจุดภาพ x_4 ได้จาก

$$H_4 = F_4 - S_4 = x_4 - \left\{\frac{x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8}{9}\right\}$$

$$= \frac{9x_4 - x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 - x_7 - x_8}{9}$$

เมื่อทำการเน้นขอบจะได้ว่า

$$E_4 = F_4 + H_4 = x_4 + \left\{\frac{9x_4 - x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 - x_7 - x_8}{9}\right\}$$

$$= \frac{9x_4 + 9x_4 - x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 - x_7 - x_8}{9}$$

$$= \frac{18x_4 - x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 - x_7 - x_8}{9}$$

$$= \frac{-x_0 - x_1 - x_2 - x_3 + 17x_4 - x_5 - x_6 - x_7 - x_8}{9}$$

นั่นคือเมื่อนำมาเขียนในรูปเมตริกซ์ใหม่จะได้ว่า

$$E_4 = \frac{1}{9}\begin{bmatrix}-1 & -1 & -1\\ -1 & 17 & -1\\ -1 & -1 & -1\end{bmatrix} \otimes \begin{bmatrix}x_0 & x_1 & x_2\\ x_3 & x_4 & x_5\\ x_6 & x_7 & x_8\end{bmatrix}$$

ในทำนองเดียวกันสำหรับเทมเพล็ทที่ใช้ในการสร้างภาพความถี่ต่ำในรูปที่ 4.3(b) ก็จะได้เทมเพล็ท สำหรับทำคอนโวลูชันในการปรับปรุงขอบภาพ ดังนี้

$$E_4 = \frac{1}{16}\begin{bmatrix} -1 & -2 & -1 \\ -2 & 28 & -2 \\ -1 & -2 & -1 \end{bmatrix} \otimes \begin{bmatrix} x_0 & x_1 & x_2 \\ x_3 & x_4 & x_5 \\ x_6 & x_7 & x_8 \end{bmatrix}$$

ดังนั้นในการปรับปรุงภาพโดยการใช้การทำคอนโวลูชันในสปาเชียนโดเมนท์สามารถเขียนสมการรูปทั่วไปได้ดังนี้

$$E(x,y) = \sum_{j=1}^{3}\sum_{k=1}^{3} T(k,l) \otimes F(x+j-2, y+k-2) \qquad (4.4)$$

เมื่อ $F(x,y)$ คือ ภาพต้นแบบ

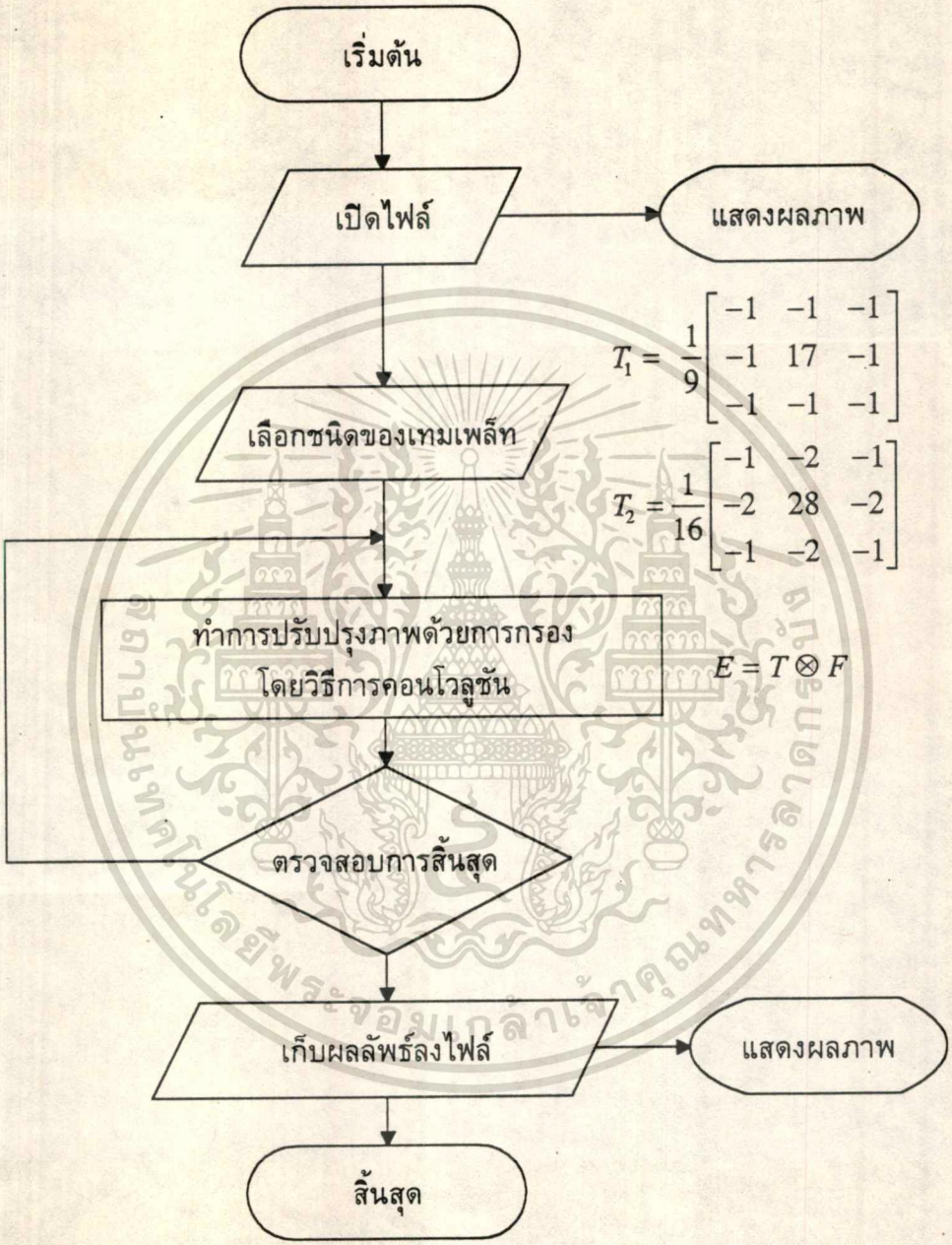
$E(x,y)$ คือ ภาพผลลัพธ์ที่ได้รับการปรับปรุงขอบภาพ

และ $T(k,l)$ คือเทมเพล็ทของตัวกรองที่ใช้ในการปรับปรุงขอบภาพ ดังสมการที่ (4.5)

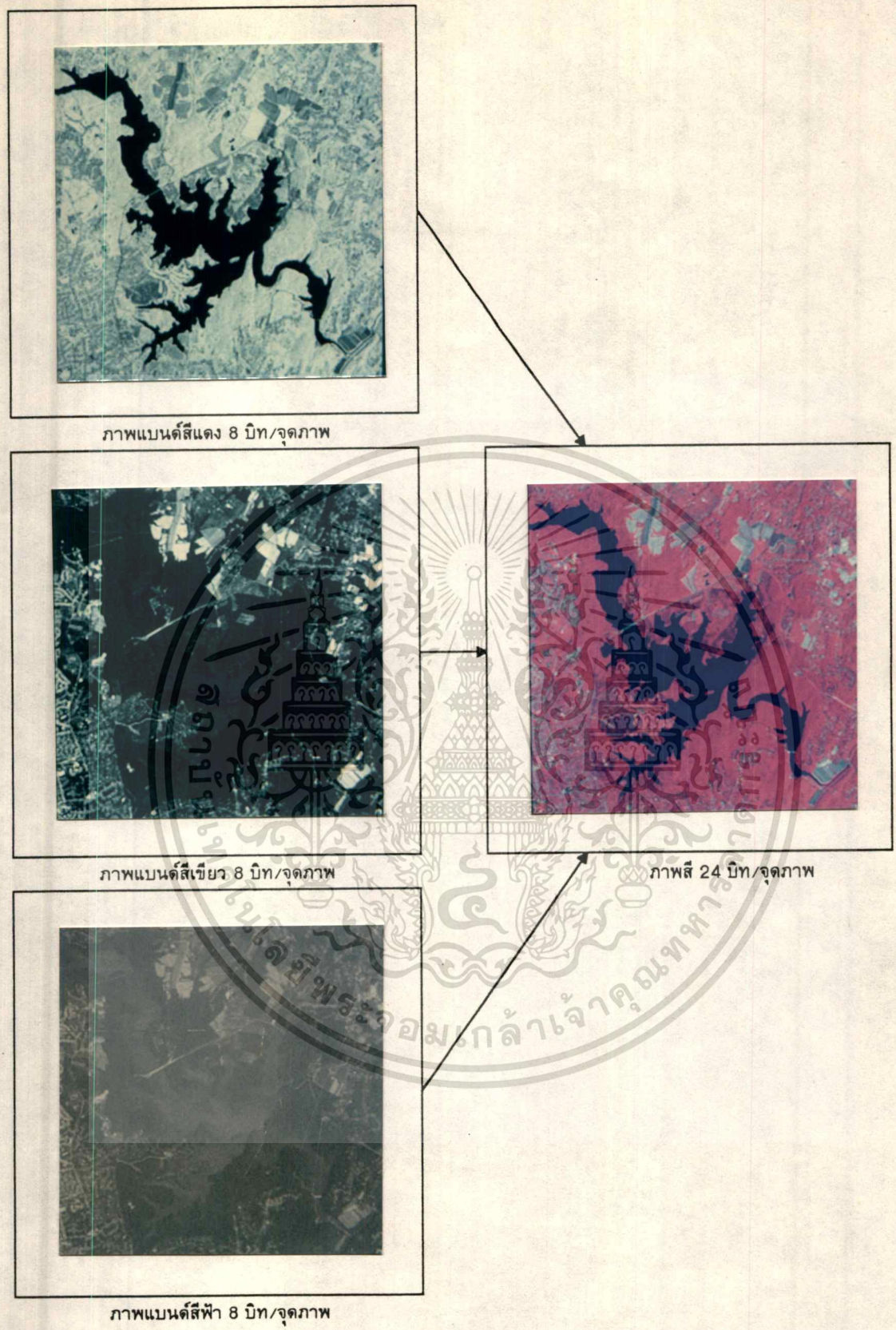
$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 17 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \text{หรือ} \quad \frac{1}{16}\begin{bmatrix} -1 & -2 & -1 \\ -2 & 28 & -2 \\ -1 & -2 & -1 \end{bmatrix} \qquad (4.5)$$

เนื่องจากผลรวมภายในเทมเพล็ทเมตริกซ์มีค่าเท่ากับหนึ่งดังที่ได้กล่าวมาในบทที่ 2 ดังนั้นพื้นที่ใดที่มีคุณสมบัติเป็นเอกพันธ์ จะยังคงถูกรักษาค่าระดับสีเทาของพื้นที่นั้นไว้

สำหรับขบวนการที่สมบูรณ์ในการปรับปรุงรายละเอียดด้วยภาพที่เกิดจากการทำให้เรียบที่ใช้ตัวกรอง สามารถแสดงได้ดังโฟลว์ชาร์ตดังรูปที่ 4.8 โดยใช้วิธีการคอนโวลูชันด้วยสมการที่ (4.4) ระหว่างเทมเพล็ทในสมการที่ (4.5) กับข้อมูลภาพทุก ๆ จุดภาพ ตัวอย่างภาพผลลัพธ์ในการปรับปรุงขอบภาพโดยวิธีนี้แสดงตัวอย่างดังรูปที่ 4.9 ซึ่งจะเห็นว่าก็ได้เหมือนผลลัพธ์จากการทำโดยวิธีการปกติจากรูปที่ (4.7) ที่กล่าวมาแล้วนั่นเอง



รูปที่ 4.8 แสดงโฟลว์ชาร์ตลำดับขั้นตอนการปรับปรุงรายละเอียดของขอบภาพโดยการใช้ตัวกรองเพียงตัวเดียวด้วยวิธีการคอนโวลูชัน



รูปที่ 4.9 แสดงตัวอย่างภาพผลลัพธ์ที่ได้รับการปรับปรุงขอบภาพโดยใช้ภาพผลลบที่ถูกทำให้เรียบด้วยการใช้ตัวกรองเพียงตัวเดียว

4.4 บทสรุป

ในการปรับปรุงรายละเอียดของขอบภาพด้วยการแปลงภาพผลลบที่เกิดจากการทำให้เรียบนี้ขอบภาพที่ได้จะมีความบางตามจุดมุ่งหมายที่ต้องการ และมีความเร็วสูงเหมาะสำหรับการทำภาพขนาดใหญ่ในทางปฏิบัติจริงอย่างมาก ดังนั้นปัญหาที่ยังเหลือก็มีเพียงเรื่องของขอบเทียมในพื้นที่เอกพันธ์ที่ทำให้มีการยกตัวเลื่อนความสว่างขึ้นไป ทำให้คอนทราสน้อยกว่าที่ควรจะเป็น ดังนั้นในบทต่อไปจะกล่าวถึงฟังก์ชันในการแปลงที่จะใช้เพื่อการกำจัดขอบเทียม เพื่อกำหนดความรุนแรงของขอบได้ตามที่ต้องการ เพื่อช่วยทำการเกลี่ยให้รอยต่อระหว่างขอบเกรเดียนท์กับข้อมูลภาพทั่วไปเพื่อที่สามารถให้ผลลัพธ์ได้อย่างสมบูรณ์

บทที่ 5

การกำจัดขอบเทียมโดยใช้ฟังก์ชันการแปลง

Spurious Edges Elimination by using transform functions

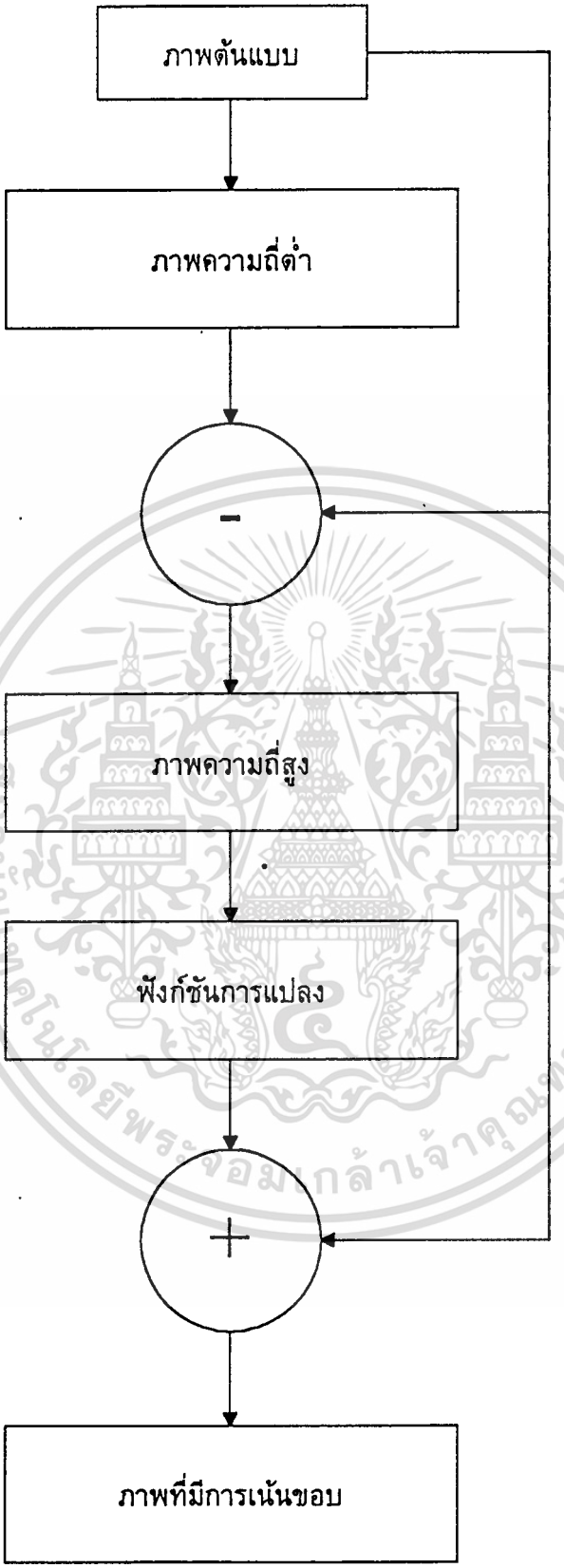
5.1 คำนำ

จากบทที่ผ่านมาในการปรับปรุงรายละเอียดของขอบภาพ โดยการใช้ภาพผลลบที่เกิดจากการทำให้เรียบนั้น ภาพผลลบที่ได้จะประกอบไปด้วยขอบจริงที่มีค่าผลลบสูง และขอบเทียมซึ่งเกิดในพื้นที่เอกพันธ์ที่มีค่าผลลบต่ำ ๆ ดังนั้นในวิทยานิพนธ์นี้ได้เสนอวิธีการกำจัดขอบเทียมเหล่านั้นให้หมดไป โดยการใช้ฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธ [1,2,3,4] ขบวนการเน้นขอบภาพนี้นำมาใช้กับภาพถ่ายดาวเทียมทั้งสามแบนด์ ภาพทั้งสามหลังการประมวลผลจะถูกนำมาผสมเป็นภาพสีที่มีการเน้นขอบอย่างสมบูรณ์ ภาพสีที่ได้เมื่อนำไปใช้ในการแปลความหมายของภาพ (image interpretation) และการจำแนกข้อมูลภาพ (image classification) แล้วส่งผลให้การแปลความหมายและการจำแนกข้อมูลภาพมีความถูกต้องแม่นยำสูง รายละเอียดของกระบวนการดังกล่าว จะได้กล่าวในหัวข้อต่อไป

5.2 การแปลงภาพผลลบที่ได้ทำให้เรียบ

จากการปรับปรุงขอบภาพโดยใช้ภาพผลลบที่ถูกทำให้เรียบนั้น จะมีปัญหาในเรื่องขอบเทียม (spurious edge) ขอบเทียมจะเกิดในบริเวณพื้นที่เอกพันธ์ซึ่งเป็นพื้นที่ที่มีค่าระดับสีเทาของจุดภาพใกล้เคียงกัน โดยให้ภาพความถี่สูงเกิดมีค่าต่ำ ๆ ออกมา ดังแสดงในบทที่ผ่านมาในรูปที่ 5.5 บริเวณดังกล่าวคือพื้นที่ที่ไม่ใช่ขอบจริงของภาพ และเป็นส่วนที่ไม่พึงปราถนาในการนำมาใช้ปรับปรุงขอบภาพ ถึงแม้จะมีค่าไม่สูงมากแต่ก็เป็นพื้นที่ที่เป็นบริเวณส่วนใหญ่ของภาพ และมีผลให้คอนทราสรวมของภาพจากการปรับปรุงขอบมีต่ำลง

เพื่อทำการกำจัดขอบเทียมในภาพความถี่สูง ดังนั้นจึงต้องทำการแปลงภาพความถี่สูงด้วยฟังก์ชันการแปลงก่อนการนำไปทำการปรับปรุงขอบภาพ ดังแสดงลำดับการประมวลผลที่มีการเพิ่มส่วนของการแปลงดังไดอะแกรมต่อไปนี้



รูปที่ 5.1 แสดงไดอะแกรมลำดับขั้นตอนการปรับปรุงขอบภาพโดยใช้ภาพผลลบที่เกิดจากการทำให้เรียบที่มีการกำจัดขอบเทียมด้วยฟังก์ชันการแปลง

ฟังก์ชันการแปลงที่เหมาะสมนั้นเป็นส่วนสำคัญที่มีส่วนตัดสินประสิทธิภาพในการปรับปรุงขอบของภาพได้มากที่สุด ฟังก์ชันที่เหมาะสมต้องประกอบทั้งความง่ายในการใช้งาน ง่ายในการกำหนดค่าเพื่อให้ได้ลักษณะฟังก์ชันที่ต้องการ และมีความยืดหยุ่นในการกำหนดตำแหน่งในการแปลงได้มากที่สุด ฟังก์ชันที่นำมาใช้คือฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธ ซึ่งมีรายละเอียดดังนี้

- **ฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธ (Inverse Butterworth)**

จากลักษณะของฟังก์ชันบัตเตอร์เวิร์ธที่มีค่าสูงทางด้านต่ำ และมีค่าต่ำทางด้านสูง ซึ่งเป็นลักษณะตรงข้ามกับจุดประสงค์ที่ต้องการขยายขอบจริงและขยายขอบเทียมดังสมการที่ (5.1) และรูปที่ 5.2 จึงต้องมีการใช้ฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธดังสมการที่ (5.2) และรูปที่ 5.3 ในการกำหนดลักษณะฟังก์ชันใช้ค่าอันดับ (order) และค่าจุดตัด (cutoff) ซึ่งมีความยืดหยุ่นที่สุดเพราะจากการกำหนดค่าตัวแปรอันดับกับจุดตัดเพียงสองตัวก็สามารถกำหนดฟังก์ชันให้เป็นลักษณะต่าง ๆ ได้มากมาย ตั้งแต่ฟังก์ชันส่วนคล้ายกับฟังก์ชันอย่างง่ายที่สุดคือ Unit Step จนถึง ฟังก์ชัน Trapezoidal และแม้กระทั่ง Logarithm ก็ตาม จึงเป็นฟังก์ชันที่ดูเหมาะสมที่สุดในการนำมาใช้กำจัดขอบเทียม

$$B(f) = \frac{1}{1 - 0.414\left(\frac{f}{c}\right)^{2k}} \qquad (5.1)$$

$$I(f) = 1 - B(f) \qquad (5.2)$$

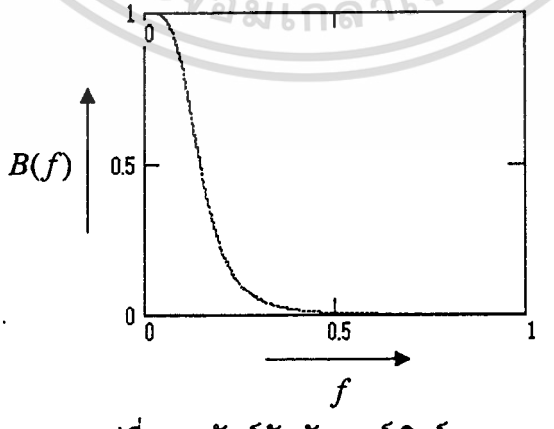
เมื่อ f คือ ค่าระดับสีเทาต้นแบบ

c คือ ตำแหน่งในการตัด (cutoff)

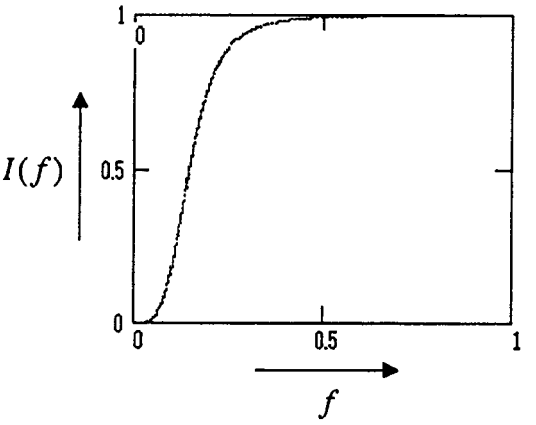
k คือ อันดับความลาดชัน (order)

$B(f)$ คือ ผลลัพธ์ของการแปลงของฟังก์ชันของบัตเตอร์เวิร์ธ

$I(f)$ คือ ผลลัพธ์ของการแปลงของฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธ



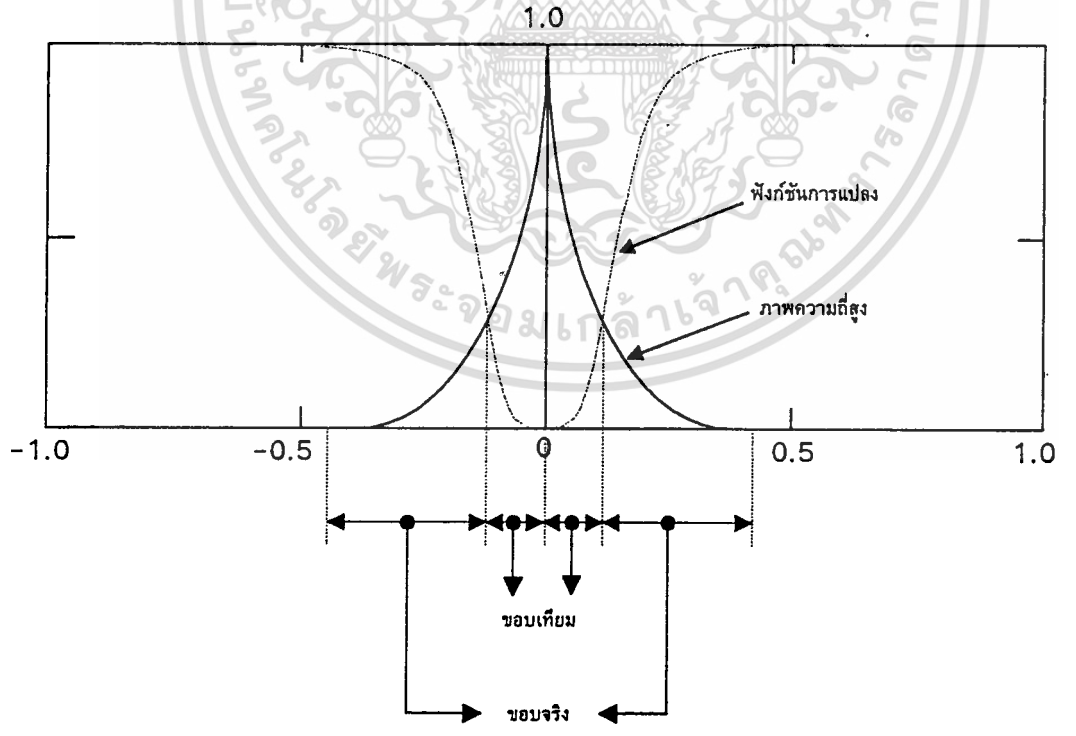
รูปที่ 5.2 ฟังก์ชันบัตเตอร์เวิร์ธ



รูปที่ 5.3 ฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธ

จากฟังก์ชันที่กล่าวมาสามารถสรุปได้ว่าฟังก์ชันที่มีความเหมาะสมที่สุดคือ ฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธ ทั้งนี้เนื่องมาจากความยืดหยุ่นของตัวฟังก์ชันเองดังที่ได้กล่าวมาแล้ว เพื่อให้สามารถใช้ได้กับลักษณะความเป็นไปได้ต่าง ๆ ของฮีสโตแกรมของภาพความถี่สูงในลักษณะของการกระจายตัวของขอบเทียมและขอบจริงในลักษณะต่าง ๆ

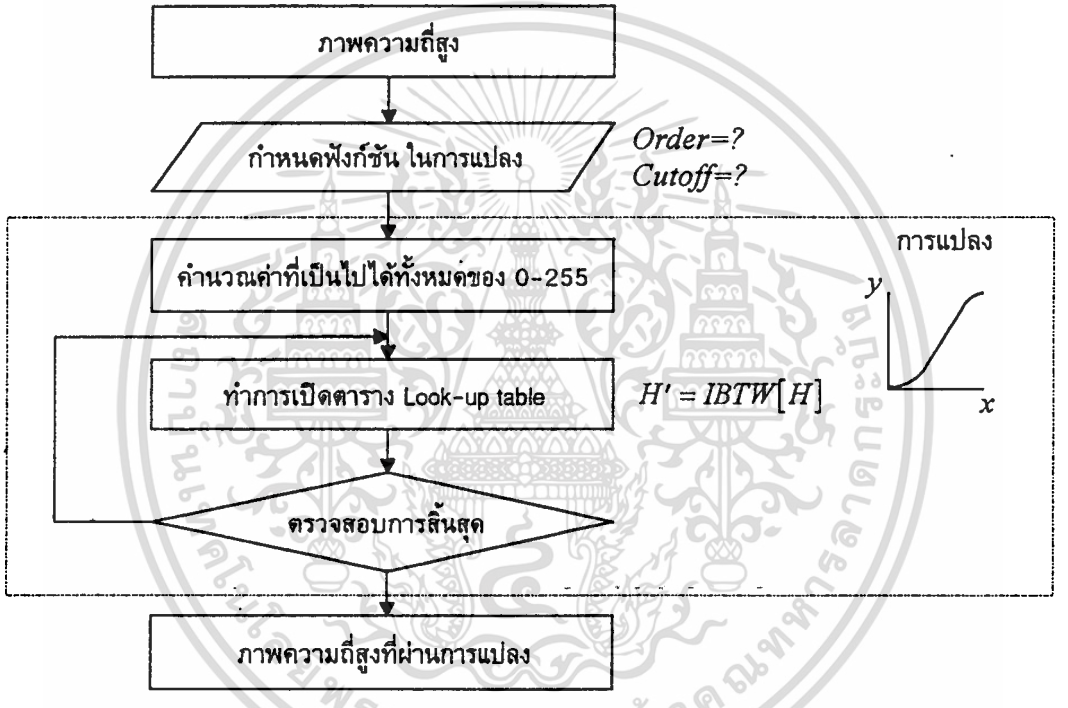
จากการทดลองหาภาพความถี่สูงของภาพถ่ายดาวเทียมต่าง ๆ และจากบทที่ผ่านมาในรูปที่ 4.6 และฮีสโตแกรมในรูปที่ 4.5 สามารถสรุปเขียนลักษณะการกระจายตัวของขอบจริงและขอบเทียม ได้ดังฮีสโตแกรมในรูปที่ 5.4



รูปที่ 5.4 แสดงลักษณะการกระจายตัวของขอบจริงและขอบเทียม

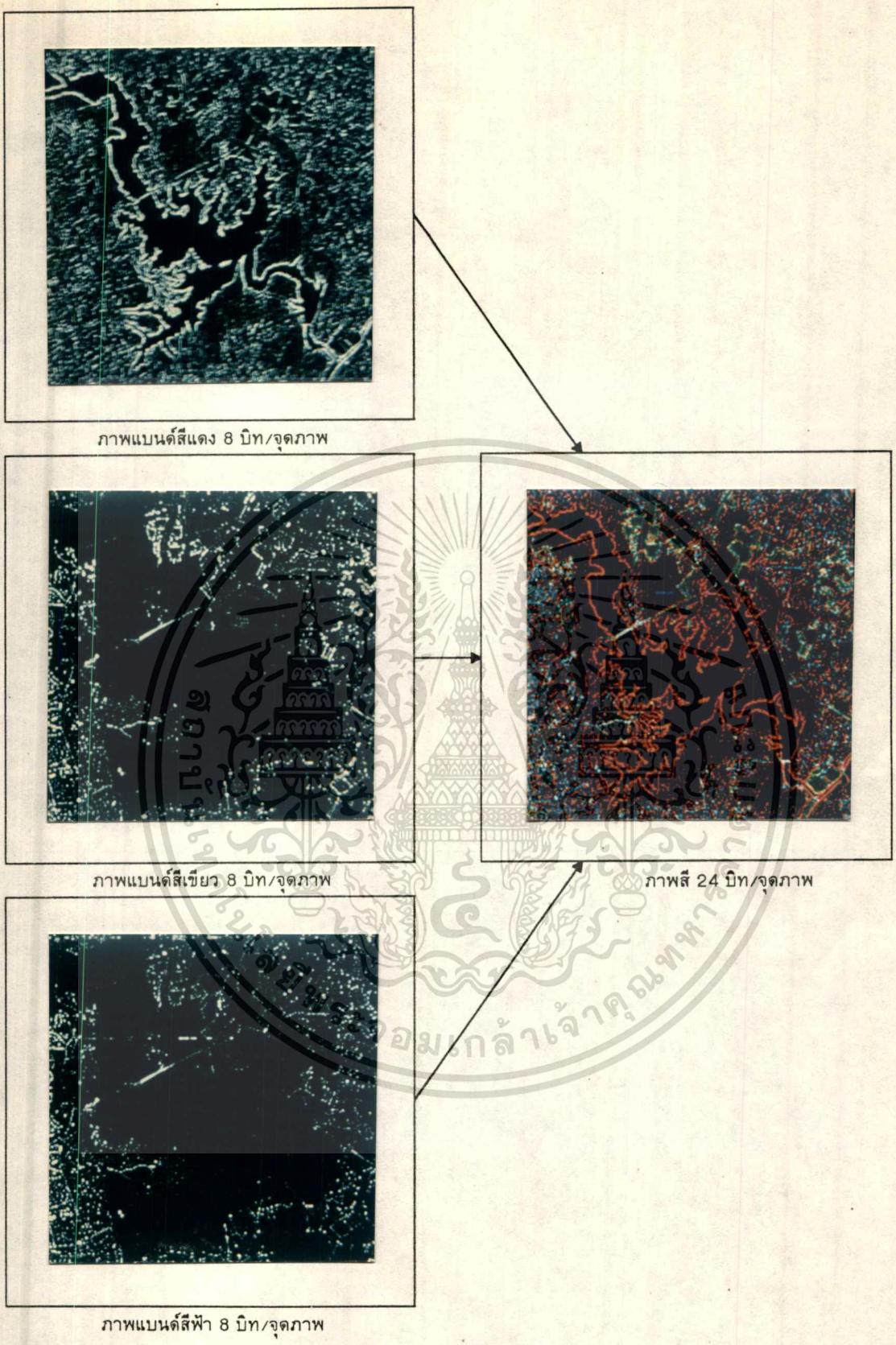
จากรูปที่ 5.4 การกระจายของภาพความถี่สูงจะกระจายอยู่รอบแกนศูนย์ส่วนที่เป็นค่าลบคือส่วนของขอบด้านนอกวัตถุ ค่าบวกคือส่วนที่เป็นขอบด้านในของวัตถุ ปริมาณของทั้งสองจะใกล้เคียงกัน เนื่องจากจะเกิดขอบขึ้นทั้งสองอย่างในวัตถุใด ๆ ในภาพ ภาพกราฟฮีสโตแกรมที่ได้จึงมีความสมมาตรกันรอบ ๆ แกนศูนย์ โดยส่วนของขอบเทียมคือส่วนที่มีค่าใกล้แกนศูนย์ ส่วนของขอบจริงคือส่วนที่อยู่ห่างออกไป

ในการกำจัดขอบเทียมจะใช้ฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธมาทำการกดค่าขอบเทียมให้มีการฉายสู่แกนใหม่ที่มีค่าต่ำจนถึงศูนย์ และทำการขยายค่าของขอบจริงให้มีค่าสูงขึ้นโดยให้มีค่าความรุนแรงได้ตามต้องการ ขบวนการในการกำจัดขอบเทียมโดยใช้ฟังก์ชันการแปลงแสดงได้ดังโฟลว์ชาร์ดดังรูปที่ 5.5



รูปที่ 5.5 แสดงโฟลว์ชาร์ดลำดับขั้นตอนในการกำจัดขอบเทียมโดยใช้ฟังก์ชันการแปลง

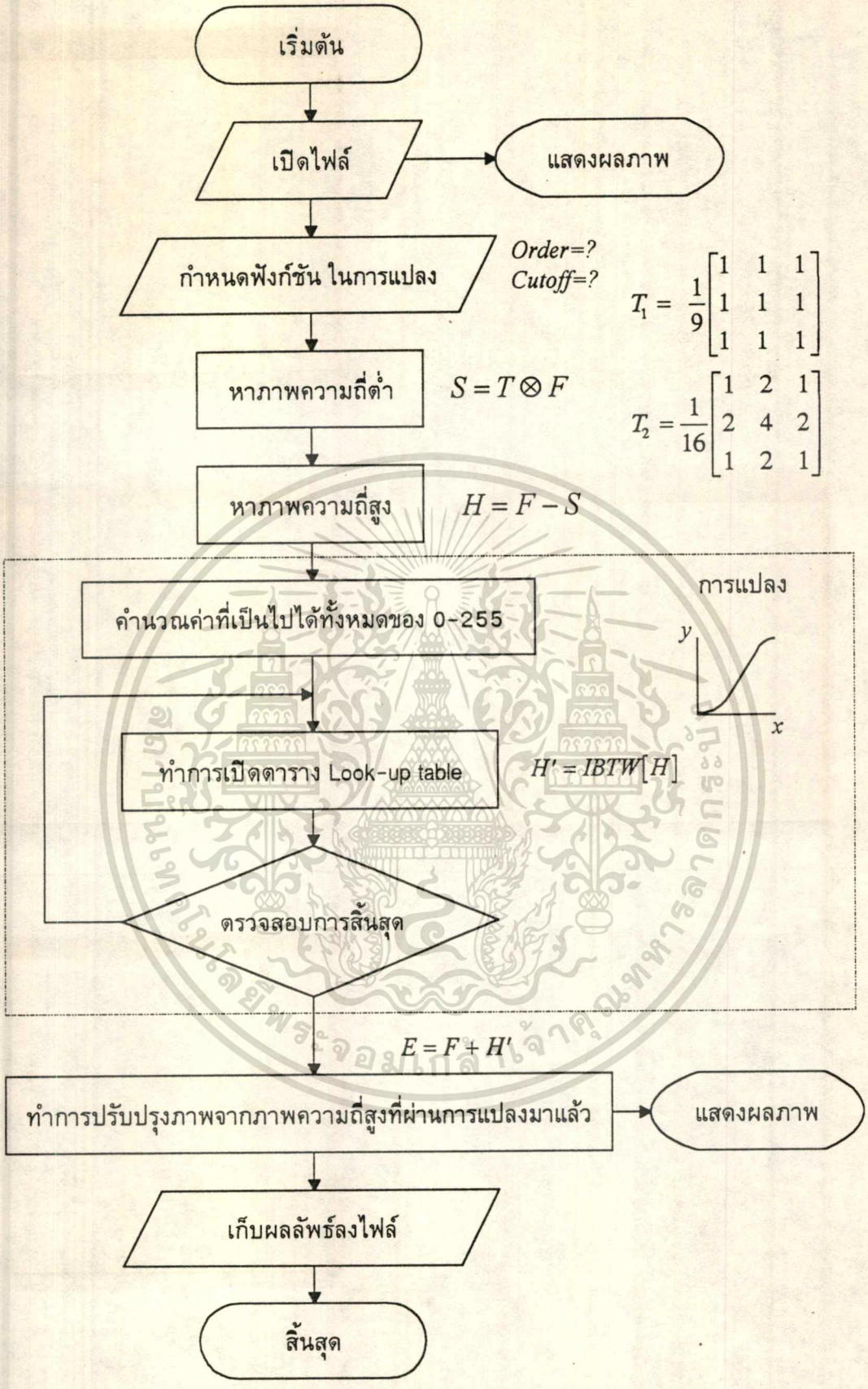
จากรูปที่ 5.5 ในการแปลงจะทำการแปลงในลักษณะการเปิดตาราง โดยเริ่มจากการกำหนดค่าอันดับและจุดตัดเพื่อกำหนดลักษณะของฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ต จากนั้นทำการสร้างตารางโดยการป้อนค่าตั้งแต่ 0 ถึง 255 ให้ฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ตดังสมการที่ (5.2) ผลที่ได้นำมาเก็บไว้ในตารางรอการเปิด จากนั้นนำภาพความถี่สูงมาเป็นตัวชี้ทำการเปิดตาราง ค่าความถี่ที่เป็นขอบเทียมจะโดนกด และค่าขอบจริงจะโดนขยายตามลักษณะของฟังก์ชันที่กำหนดมาจากค่าอันดับ และจุดตัด เมื่อนำภาพความถี่สูงจากรูปที่ 4.6 มาผ่านการกำจัดขอบเทียมโดยใช้ค่าอันดับเท่ากับ 2 และค่าจุดตัดเท่ากับ 70 จะได้ภาพความถี่สูงที่ผ่านการแปลงดังรูปที่ 5.6



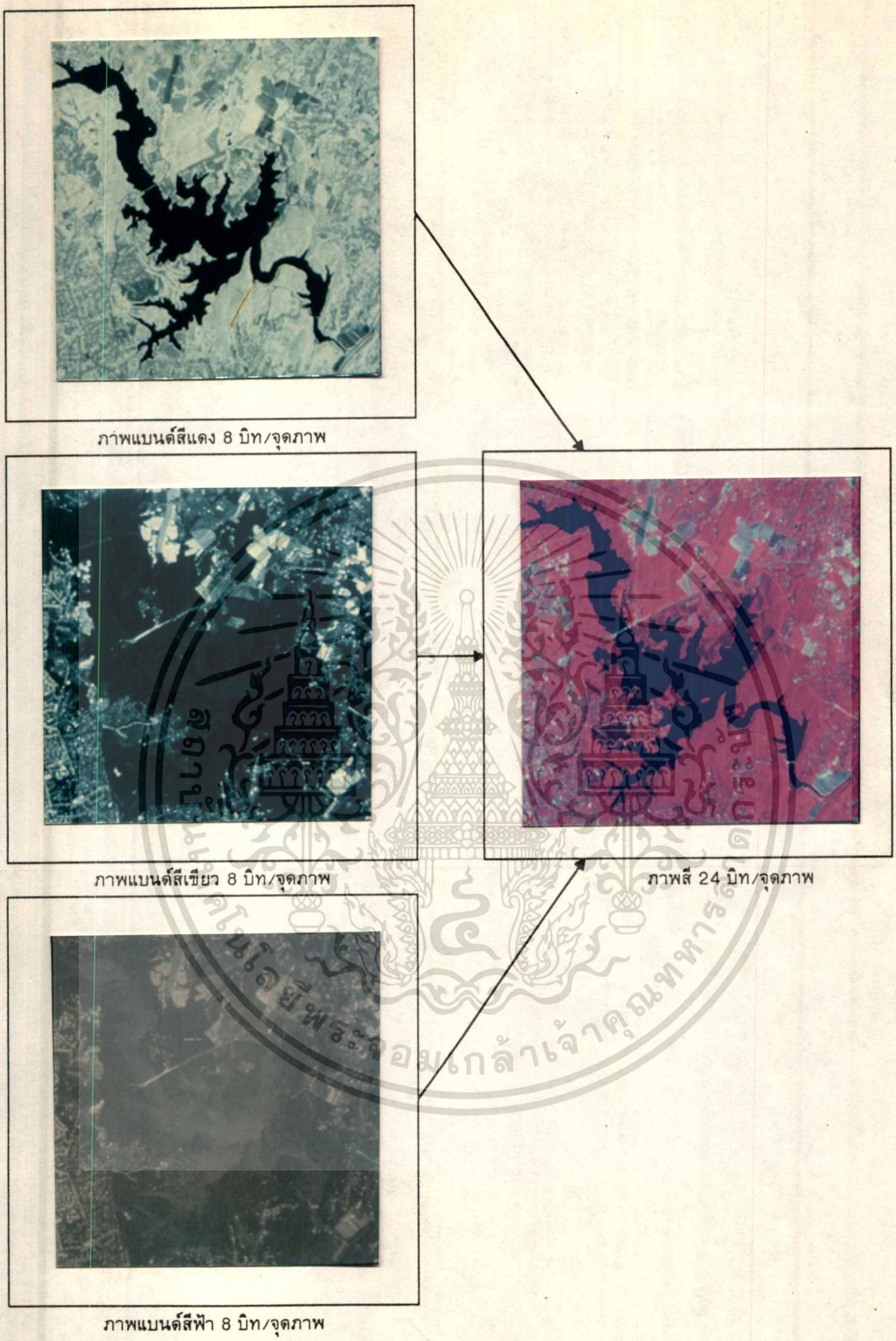
รูปที่ 5.6 แสดงตัวอย่างภาพความถี่สูงที่ผ่านการแปลงจากการใช้ค่าอันดับเท่ากับ 2 และค่าจุดตัดเท่ากับ 70

5.3 การปรับปรุงรายละเอียดของขอบด้วยการแปลงภาพผลลบที่ถูกทำให้เรียบ

เมื่อได้ภาพความถี่สูงที่ได้รับการกำจัดขอบเทียมแล้ว ต่อไปก็นำมาทำการปรับปรุงรายละเอียดของขอบภาพ โดยนำมารวมกลับเข้าไปยังภาพต้นแบบ ขบวนการในการปรับปรุงภาพที่สมบูรณ์แสดงได้ดังโฟลว์ชาร์ตดังรูปที่ 5.7 และตัวอย่างภาพความถี่สูงที่ได้รับการกำจัดขอบเทียมแล้วจากรูปที่ 5.6 เมื่อนำมาทำการปรับปรุงรายละเอียดของขอบภาพสามารถแสดงได้ดังรูปที่ 5.8



รูปที่ 5.7 แสดงโฟลว์ชาร์ตลำดับขั้นตอนที่สมบูรณ์ในการปรับปรุงภาพด้วยการแปลงภาพผลลบที่เกิดจากการทำให้เรียบ



รูปที่ 5.8 แสดงตัวอย่างผลลัพธ์จากการปรับปรุงภาพโดยการใช้ฟังก์ชันการแปลงกลับของบัตเตอร์เวิร์ธ ในการกำจัดขอบเทียมด้วยค่าอันดับเท่ากับ 2 และจุดตัดเท่ากับ 70

5.4 บทสรุป

โดยการใช้ภาพผลลบที่เกิดจากการทำให้เรียบในการปรับปรุงรายละเอียดของขอบภาพนั้น เนื่องจากขอบเทียมภาพผลลัพธ์จากการปรับปรุงจะโดนยกค่าความสว่างในพื้นที่เอกพันธ์ขึ้นไป เมื่อใช้ฟังก์ชันการแปลงมาทำการกำจัดขอบเทียมเหล่านั้นจะทำให้บริเวณพื้นที่เอกพันธ์มีความเป็นเนื้อเดียวกับภาพต้นแบบทุกประการ ไม่มีการยกค่าความสว่างขึ้นไป จะมีก็เพียงขอบเท่านั้นที่มีการเน้นรายละเอียด ภาพที่ได้จึงมีคอนทราสที่ดี สำหรับความต่อเนื่องของรอยต่อจะมีความต่อเนื่องเป็นอย่างดี ทั้งนี้เพราะจากการเกลี่ยของฟังก์ชันบัตเตอร์เวิร์ธ ดังนั้นภาพที่ได้จะเป็นภาพที่ได้รับการปรับปรุงอย่างสมบูรณ์แบบ

บทที่ 6

บทสรุปและแนวทางในการพัฒนาต่อไป

6.1 สรุปผลการวิจัย

ในการประมวลผลภาพถ่ายดาวเทียม หน่วยความจำเป็นสิ่งจำเป็นสำหรับใช้ในการจองพื้นที่ เพื่อใช้ในการเก็บภาพขนาดใหญ่ การทำงานบนไมโครซอร์ฟวินโดว์ สามารถให้สิ่งที่ต้องการนั้นได้ และได้อัตถประโยชน์อย่างอื่นอีกมากมาย ทำให้ไม่ต้องเสียเวลากับการจัดเตรียมส่วนประกอบของโปรแกรม โดยสามารถใช้เวลาทั้งหมดไปกับวิธีการในการประมวลผลได้อย่างเต็มที่ การเขียนโปรแกรมที่ทำงานบนวินโดว์จึงมีประสิทธิภาพอย่างมาก การใช้ข้อมูลภาพในรูปแบบบิทแมพที่เป็นมาตรฐานบนไมโครซอร์ฟวินโดว์ เอื้ออำนวยต่อการแสดงผลภาพ การอ่านและเก็บข้อมูลภาพ ช่วยให้สามารถส่งผ่านไปยังโปรแกรมประยุกต์ต่าง ๆ บนไมโครซอร์ฟวินโดว์ได้ อีกทั้งในการแสดงภาพสีบนไมโครซอร์ฟวินโดว์ ที่ใช้กับภาพถ่ายก็สามารถช่วยบุกเบิกในการแสดงภาพสี ของการประมวลผลภาพถ่ายดาวเทียม ในการสำรวจระยะไกลได้เป็นอย่างดี

ภาพถ่ายดาวเทียมโดยมากมักจะมีคอนทราสต่ำ กล่าวคือ ภาพมีระดับสีเทาของจุดภาพใกล้เคียงกันตลอดทั้งภาพ ทำให้มองไม่เห็นรายละเอียดของภาพ ในการปรับปรุงคอนทราสของภาพทำได้โดยการเปลี่ยนแปลงลักษณะการกระจายระดับสีเทาของจุดภาพ หรือเป็นการเปลี่ยนแปลงฮีสโตแกรมของภาพนั่นเอง วิธีที่ใช้ในการดึงอย่างเชิงเส้นของภาพสี ทำได้โดยการดึงภาพถ่ายดาวเทียมทุกแบนด์บนฮีสโตแกรมรวม ด้วยฟังก์ชันการแปลงเดียวกัน เพื่อรักษาสัดส่วนของสีแต่ละสีให้มีค่าคงเดิมไว้ หากภาพถ่ายดาวเทียมมีสัญญาณรบกวน หรือจุดภาพที่มีค่าแยกออกมาจากกลุ่มข้อมูลภาพส่วนรวมเป็นจำนวนเล็กน้อย ในการดึงด้วยค่าสูงสุดและต่ำสุดอย่างอัตโนมัติไม่สามารถให้คอนทราสที่ดีออกมาได้ ดังนั้นการนำค่าเปอร์เซนต์จำนวนจุดภาพบนฮีสโตแกรมมาใช้ จะช่วยให้ได้ภาพที่มีคอนทราสของกลุ่มข้อมูลจริงที่สนใจสูงขึ้นได้ ดังแสดงไว้ในบทที่ 2 เมื่อใช้ภาพต้นแบบจากรูปที่ 2.3 จะได้ภาพที่ได้รับการปรับปรุงคอนทราสดังรูปที่ 2.20

นอกจากปัญหาในด้านคอนทราสของภาพแล้ว ในบทที่ 3 กล่าวถึงปัญหาอีกอย่างหนึ่งคือการขาดความคมชัดของภาพ เนื่องจากปรากฏการณ์กรองความถี่ต่ำในชั้นบรรยากาศแอสโมสเฟียร์ ดังนั้นการเพิ่มความคมชัดของขอบภาพ จะช่วยเพิ่มรายละเอียดของข้อมูลภาพถ่ายดาวเทียมให้มีความคมชัดมากยิ่งขึ้น ซึ่งสามารถทำได้โดยการเสริมข้อมูลเกรเดียนท์เข้าไปในภาพต้นแบบ

จากการปรับปรุงความคมชัดของขอบภาพด้วยการใช้เกรเดียนท์โดยวิธีการของ Shettigara และ Odins นั้นใช้เกรเดียนท์เทมเพล็ทแบบแนวตั้งและแนวนอน ทำให้ผลลัพธ์ที่ได้ไม่เหมาะสมกับภาพถ่ายดาวเทียมที่มีรายละเอียดมาก เมื่อใช้ภาพต้นแบบจากรูปที่ 2.20 จะได้ภาพผลลัพธ์ของวิธีการ Shettigara และ Odins ดังรูปที่ 3.4

ดังนั้นเพื่อผลตอบสนองของขอบในทิศทางต่าง ๆ ในภาพถ่ายดาวเทียม จึงได้หันมาทดลองเพิ่มเทมเพล็ทโดยกลายเป็นเทมเพล็ทแบบ 12 ทิศทาง เพื่อตอบสนองขอบได้หลาย ๆ ลักษณะขึ้น เมื่อใช้ภาพต้นแบบจากรูปที่ 2.20 จะได้ภาพผลลัพธ์ของวิธีเทมเพล็ทแบบ 12 ทิศทางดังรูปที่ 3.9

แต่การใช้จำนวนเทมเพล็ทที่ตอบสนองหลายทิศทางมากขึ้น จึงใช้เทมเพล็ทขนาดใหญ่ ทำให้ต้องใช้เวลาในการประมวลผลมากขึ้น อีกทั้งภาพถ่ายดาวเทียมในทางปฏิบัติจริงเป็นภาพขนาดใหญ่มาก ตัวอย่างเช่น ภาพขนาด 640×480 จากการทดลองต้องใช้เวลาในการประมวลผลมากกว่า 10 นาที ดังนั้นต้องทำการเปลี่ยนแนวทางและหาวิธีการอื่นที่นอกเหนือออกไป

สรุปปัญหาที่เกิดขึ้น จากการทดลองใช้วิธีการทั้งสองนี้ในทางปฏิบัติจริง กับภาพถ่ายดาวเทียมได้ 3 ข้อคือ

- ขอบมีความหนาตามขนาดของเทมเพล็ท ในการทำคอนโวลูชันด้วยเทมเพล็ทขนาดใหญ่นั้น ใช้เวลาในการคำนวณมากตามมาถึงแม้การใช้ถึง 12 เทมเพล็ท มีวัตถุประสงค์เพื่อให้ตอบสนองได้มากทิศทางของขอบภาพก็ตาม การหาวิธีการใหม่ที่ใช้เทมเพล็ทขนาดเล็กก็น่าจะเป็นทางที่ดีกว่า
- เกรเดียนท์ที่มีความรุนแรงสูงมากดังนั้นภาพผลลัพธ์ที่ได้จะเกิดโอเวอร์ชูตอย่างรุนแรง
- มีขอบเทียมในพื้นที่เอกพันธ์ ผลของเกรเดียนท์ในพื้นที่เอกพันธ์ที่เมื่อนำมาใช้ในการปรับปรุงภาพ โดยรวมกับภาพต้นแบบทำให้มีการยกค่าของความสว่าง (brightness) ขึ้นไปทั้งภาพ พิจารณาได้จากภาพผลลัพธ์จากวิธีการของ Shettigara และ Odins ในรูปที่ 3.4 และภาพผลลัพธ์จากวิธีการ 12 เทมเพล็ทดังในรูปที่ 3.9 ที่บริเวณพื้นที่เอกพันธ์ในภาพมีความสว่างกว่าภาพต้นแบบจนเห็นได้ชัด

แนวความคิดอย่างง่ายในการกำจัดขอบเทียมในพื้นที่เอกพันธ์เหล่านั้น โดยการใช้ฟังก์ชันอย่างง่ายคือฟังก์ชันสเต็พหนึ่งหน่วย (unit step) มาทำการตัดเทรสโฮลด์ผลการทดลองที่ได้ทำให้รู้ว่าสามารถแก้การยกค่าความสว่างในพื้นที่เอกพันธ์ได้ โดยกดค่าขอบเทียมที่ไม่ต้องการทิ้งไป ซึ่งในการตัดนั้นถึงแม้ภาพผลลัพธ์ที่ได้จะขาดความต่อเนื่องของขอบภาพและภาพข้อมูลทั่วไปก็ตามแต่ก็เป็นแนวทางในการหาฟังก์ชันที่สามารถให้ความต่อเนื่องของขอบที่ดีต่อไปได้

ดังนั้นในความประสงค์ที่ต้องการปรับปรุงรายละเอียดของขอบภาพเพื่อแก้ปัญหาการกรองความถี่ต่ำในภาพถ่ายดาวเทียมและเพื่อเพิ่มความคมชัดให้กับขอบภาพ จึงต้องทำการหาวิธีการใหม่ที่ให้ผลได้ตามวัตถุประสงค์ต่าง ๆ ที่ต้องการ โดยการใช้ภาพผลลบที่เกิดจากการทำให้เรียบ (subtracted smoothing image) ที่ให้ผลของเกรเดียนท์ที่มีความรุนแรงน้อยกว่า ในลักษณะของภาพความถี่สูง โดยสามารถตอบสนองได้ในลักษณะทุกทิศทุกทาง และให้ภาพขอบที่มีความบางมากจึงตรงตามวัตถุประสงค์ที่ต้องการอย่างมาก และในวิธีการของทำให้เรียบ การบวกและการลบทำให้สามารถปรับปรุงขบวนการ โดยย่อขบวนการลงบนตัวกรอง

ความถี่ ได้เป็นการทำคอนโวลูชันในสปาเชียนโดเมนของเทมเพล็ทกรองความถี่สูงผ่านได้เพียงตัวเดียว ดังนั้นจะมีความเร็วสูงมาก เมื่อใช้ภาพต้นแบบจากรูปที่ 2.20 จะได้ภาพผลลัพธ์จากวิธีการปรับปรุงรายละเอียดของขอบภาพด้วยการใช้ภาพผลลบที่ถูกทำให้เรียบดังรูปที่ 4.9

ในการใช้ฟังก์ชันการแปลงเพื่อกำจัดขอบเทียม ในที่นี้เลือกใช้ฟังก์ชันการแปลงกลับของบัตเตอร์เวิอร์ธ ทั้งนี้เนื่องมาจากว่าเป็นฟังก์ชันที่ยืดหยุ่นที่สุด เพราะสามารถปรับค่าอันดับ (order) และจุดตัด (cutoff) ให้สามารถเป็นได้ตั้งแต่ฟังก์ชันในการตัดอย่างง่ายคือฟังก์ชันสเต็บหนึ่งหน่วย (unit step) ไปจนถึงฟังก์ชันที่มีความลาดชันคือฟังก์ชันแรมพ์ (ramp)

ในลำดับขั้นตอนการตัดขอบเทียม จะเริ่มจากการใช้ฟังก์ชันบัตเตอร์เวิร์ธที่มีค่าอันดับสูง (high order) ที่ให้ลักษณะของฟังก์ชันใกล้เคียงฟังก์ชันสเต็บหนึ่งหน่วยก่อนเป็นอันดับแรก เพื่อทำการทดลองตัดหาช่วงตำแหน่งที่สามารถแยกขอบจริงออกจากขอบเทียมได้อย่างชัดเจนที่สุด จะได้ค่าจุดตัดออกมา จากนั้นขั้นตอนต่อไปก็ทำการลดมาใช้ค่าอันดับค่าต่ำ (lower order) โดยใช้ค่าจุดตัดค่าเดิมที่ให้ลักษณะของฟังก์ชันใกล้เคียงฟังก์ชันฟังก์ชันแรมพ์ เพื่อเป็นการสร้างความต่อเนื่องให้กับขอบของภาพความถี่สูง เพื่อไม่ทำให้ภาพผลลัพธ์จากการปรับปรุงขาดความต่อเนื่องของภาพต้นแบบและภาพความถี่สูงที่นำมารวมเข้าด้วยกัน เมื่อใช้ภาพต้นแบบจากรูปที่ 2.20 จะได้ภาพผลลัพธ์การปรับปรุงรายละเอียดของขอบภาพ ที่ได้รับการกำจัดขอบเทียมดังรูปที่ 5.8

ภาพที่ได้รับปรับปรุงและมีการกำจัดขอบเทียมนั้น ทำให้มีพื้นที่เอกพันธ์ที่เหมือนกับภาพต้นแบบ ในขณะที่มีเส้นขอบภาพที่คมชัด จึงเป็นภาพที่ได้รับการปรับปรุงอย่างสมบูรณ์แบบ ภาพสีที่ได้เมื่อนำไปใช้ในการแปลความหมายของภาพ (image interpretation) และในการจำแนกข้อมูลภาพ (image classification) แล้วจะส่งผลให้มีความถูกต้องแม่นยำสูง

6.2 ปัญหาที่เกิดขึ้นและข้อเสนอแนะ

ในการทำการกำจัดขอบเทียมนั้นสามารถเลือกใช้ฟังก์ชันต่าง ๆ มาทำการแปลงได้มากมายตามความง่ายแก่การกำหนดค่า ฟังก์ชันหนึ่งที่มีความยืดหยุ่นมากคือฟังก์ชันบัตเตอร์เวิอร์ธ ที่สามารถทำการกำหนดความชัดและตำแหน่งในการตัดได้มากหลายรูปแบบ ในการตัดส่วนของขอบเทียมนั้นต้องทำการทดสอบผลของผลลัพธ์ทุกครั้งที่ทำการตัด จึงต้องเขียนโปรแกรมในลักษณะแสดงผลทันที (interactive) และทำการลดหรือเพิ่มค่าความชันและตำแหน่งในการตัดอย่างลองผิดลองถูกจนเข้าใกล้ค่าที่ให้ผลของการปรับปรุงขอบที่ดีที่สุดต่อไป ดังนั้นจึงไม่เป็นการทำงานในรูปแบบอัตโนมัติ ต้องอาศัยการตัดสินใจจากมนุษย์อยู่

สำหรับประโยชน์จากการนำภาพผลลัพธ์สีที่ได้ไปประยุกต์ใช้งานอื่น ๆ เช่นในกรณีที่ต้องการทำการแก้ความผิดปกติทางรูปทรงเรขาคณิต ของภาพถ่ายดาวเทียม ก็สามารถทำได้เช่นกัน โดยการกำหนดค่าฟังก์ชันให้มีค่าอันดับ (order) ที่สูงขึ้น จนขยายขอบจริงให้การขยายด้วยความรุนแรงมากขึ้น จึงสามารถกำหนดจุดควบคุมภาคพื้นดิน (ground control point) ได้อย่างถูกต้องแม่นยำ

ภาคผนวก ก

เอกสารอ้างอิง

[1] F. Cheevasuvit, K. Dejhan and A. Somboonkaew, ***"Edge enhancement using transform of subtracted smoothing image"***, Proceedings of the 13th Asian Conference on Remote Sensing (ACRS), Ulaanbaatar, Mongolia, October 7 to 11, 1992.

[2] อาโมทย์ สมบูรณ์แก้ว และ ฟูศักดิ์ ชีวสุวิทย์, ***"การปรับปรุงรายละเอียดของขอบในภาพด้วยการแปลงภาพผลลบที่ถูกทำให้เรียบ"***, เอกสารการประชุมทางวิชาการของมหาวิทยาลัยเกษตรศาสตร์ ครั้งที่ 31, มหาวิทยาลัยเกษตรศาสตร์, 3 ถึง 6 กุมภาพันธ์ 2536.

[3] F. Cheevasuvit, K. Dejhan and A. Somboonkaew, ***"Edge enhancement in colour images using transform of subtracted smoothing image"***, Proceedings of the South East Asian Regional Conference on Education and Research in Remote Sensing, Johor Bahru, Malaysia, session 6, June, 1993.

[4] อาโมทย์ สมบูรณ์แก้ว บัณฑิต สุมนวัฒนเดช และ ฟูศักดิ์ ชีวสุวิทย์, ***"การปรับปรุงรายละเอียดของขอบในภาพสีด้วยการแปลงภาพผลลบที่ถูกทำให้เรียบ"***, เอกสารการประชุมใหญ่วิชาการทางวิศวกรรม ประจำปี 2536, วิศวกรรมสถานแห่งประเทศไทยในพระบรมราชูปถัมภ์, หน้า 601 ถึง 606, 27 ถึง 30 พฤศจิกายน 2536.

[5] W.K. Pratt, ***"Digital Image Processing"*** *(second edition)*, Reading, John Wiley & Sons Inc, 1991.

[6] R.C. Gonzalez, and R.E. Woods, ***"Digital Image Processing"*** *(second edition)*, Reading, Addison-Wesley Publishing company Inc, 1992.

[7] จิตรเกษม งามนิล และ ฟูศักดิ์ ชีวสุวิทย์, ***"เทคนิคการกำหนดสีในการแสดงภาพสีบนระบบ VGA โดยการสร้างบล็อกสามมิติบนแกนสีทั้งสาม"***, เอกสารการประชุมทางวิชาการวิศวกรรมไฟฟ้าประจำปี 2534, วิศวกรรมสถานแห่งประเทศไทย, หน้า 54 ถึง 62, พ.ศ. 2534.

[8] จิตรเกษม งามนิล, *"เทคนิคการแสดงภาพถ่ายดาวเทียมแบบภาพสีเท็จบนไมโครคอมพิวเตอร์"*, วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า, บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ. 2535.

[9] F. Cheevasuvit, C. Peanvijarnpong and S. Murai, ***"Edge Enhancement by adding gradient information"***, Proceedings of the 9th Asian Conference on Remote Sensing (ACRS), Bangkok, Thailand , pp p-2-1 to p-2-8, Nov. 23-29, 1998.

[10] K.V. Shettigara and J.A. Odins, ***"Application of NOAA-AVHRR Images for Structural Studies Related to Groundwater flow in the Murray Basin"***, Proceedings of the 4th Australasian Remote Sensing Conference, September 14-18, 1987.

[11] R.A. Schowengerdt, ***"Techniques for IMage Processing and Classification in Remote Sensing"***, Reading, Academic Press Inc. 1983.

[12] P.J. Ahern and J. Sirois ,***"Reflectance Enhancements for the Thematic Mapper: An Efficient Way to Produce Images of Consistently High Quality"***, Photogrammetric Engineering and Remote Sensing, Vol. 55, No. 1, pp. 61-67, January 1989.

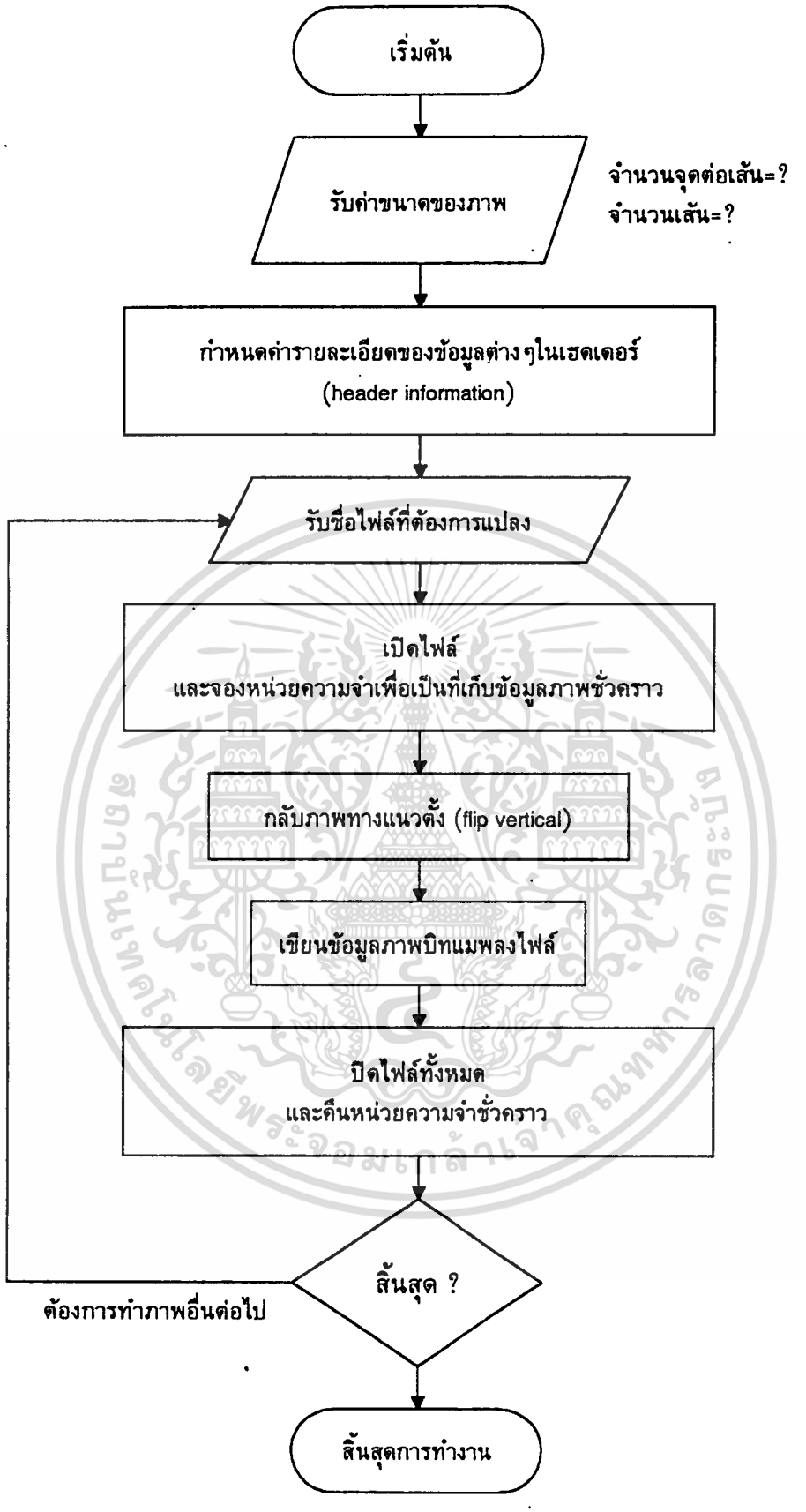
[13] J.A. Richards, ***"Remote Sensing Digital Image Analysis"***, Reading, Springer-Verlag Berlin Heidelberg, 1986.

ภาคผนวก ข.

การแปลงรูปแบบข้อมูลภาพ

การแปลงภาพแอสกี้เป็นรูปแบบบิทแมพ

ภาพถ่ายที่ต้องการแปลงเป็นภาพสีเทาระหัสแอสกี้ (ASCII) ขนาด 8 บิทต่อหนึ่งจุดภาพ เป็นข้อมูลภาพทั้งหมดโดยไม่มีเฮดเดอร์ ในการแปลงจะใส่เฮดเดอร์ให้ โดยจะรับขนาดจุดภาพต่อหนึ่งเส้น (ขนาดทางแกน x) จำนวนเส้นของภาพ (ขนาดทางแกน y) และรับชื่อไฟล์ที่ต้องการแปลง โดยมีลำดับอัลกอริทึมในการทำงานของโปรแกรมดังนี้



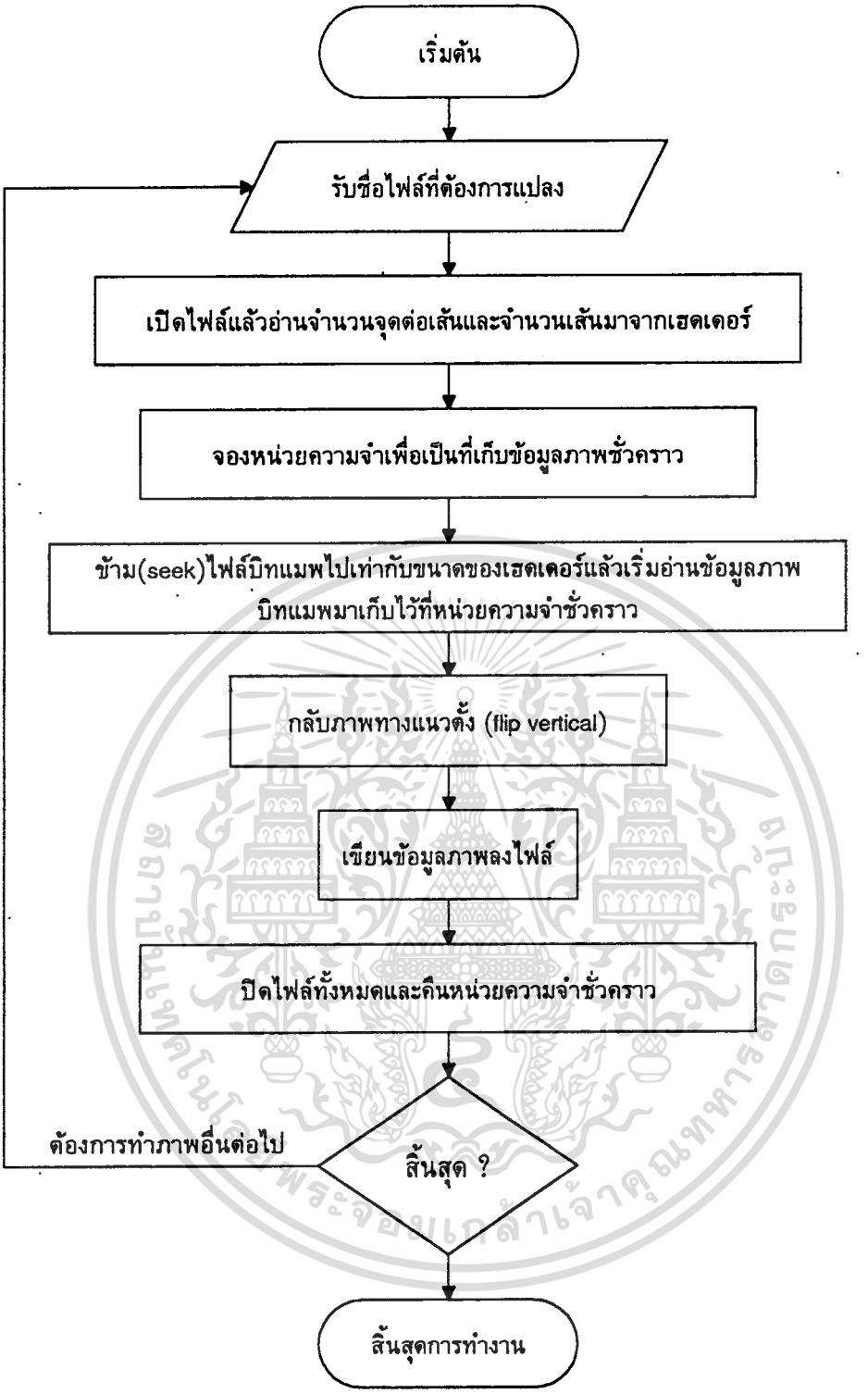
โฟล์วชาร์ตการทำงานของโปรแกรมแปลงภาพแอสกี้เป็นรูปแบบบิทแมพ

```
/*--------------------------------------

PROGRAM         ->CHANGE IMAGE TO BMP FILE...
VERSION         ->3.0...(on Windows)
BY              ->ARMOTE SOMBOONKAEW...
START WRITTEN   ->7 JUNE 1993...
LAST UPDATE     ->18 MARCH 1994...
INSTITUTE       ->KMIT'L...
COMMENT         ->...have output (*.bmp) that add
                    to original file...

--------------------------------------*/
#include  <stdio.h>
#include  <dos.h>
#include  <string.h>
#include  <conio.h>
#include  <stdlib.h>
#include  <malloc.h>

#define   pixel2bytes(n)  ((n+7)/8)

int       XPIX,YPIX;


typedef struct  {
                char id[2];
                long filesize;
                int reserved[2];
                long headersize;
                long infoSize;
                long width;
                long depth;
                int biPlanes;
                int bits;
                long biCompression;
                long biSizeImage;
                long biXPelsPerMeter;
                long biYPelsPerMeter;
                long biClrUsed;
                long biClrImportant;
                }BMPHEAD;



main()
{
int             x,y,bytes;
char            *n;
FILE            *infile, *outfile;
static char             name[80];
static unsigned char    image[1024];
BMPHEAD                 bmp;
unsigned char huge *data;
unsigned long   index,imagesize;

clrscr();
printf("\nCONVERT...IMAGE TO BMP...VERSION 3.0
                                (on Windows)");
printf("\nBY..ARMOTE SOMBOONKAEW...");
printf("\nxpix -> ");scanf("%d",&XPIX);
printf("ypix -> "); scanf("%d",&YPIX);

/*for THE odd XPIX*/
bytes=XPIX;
if(bytes & 0x0003) {
        bytes |= 0x0003;
        ++bytes;
}

memset((char *)&bmp,0,(long)sizeof(BMPHEAD));
memcpy(bmp.id,"BM",2);

bmp.headersize=1078L;

bmp.width=(long)XPIX;
bmp.depth=(long)YPIX;
bmp.filesize=bmp.headersize+(long)bytes*
                        (long)bmp.depth;

bmp.infoSize=0x28L;
bmp.bits=8;
bmp.biPlanes=1;
bmp.biCompression=0L;
bmp.biSizeImage=bmp.filesize-bmp.headersize;

bmp.biXPelsPerMeter=0L;
bmp.biYPelsPerMeter=0L;
bmp.biClrUsed=0L;
bmp.biClrImportant=0L;

printf("\nMaking Header Complete!");

do {
  printf("\nInput from  -> "); scanf("%s",&name);
  printf("Output  to  -> (*.bmp)...");

  printf("\nReading IMG from file [%s]",name);
  infile = fopen(name,"rb");
  n=strchr(name,'.');
  if(n) strcpy(n,".bmp");
  else  strcat(name,".bmp");

  printf("\nSaving BMP to file [%s]\n",name);
  outfile = fopen(name,"wb");
  if((infile&&outfile) == NULL) {
          printf(" Open file error..\n");
          exit(1);
  }
```

```
printf("\nWrite header");
if(fwrite((char *)&bmp,1,
        (long)sizeof(BMPHEAD),outfile)
        != sizeof(BMPHEAD))
{
          printf("fwrite HEADER error");
          exit(1);
}


printf("\nWrite pallette");
for(x=0;x<256;++x){
          fputc(x,outfile);/*R*/
          fputc(x,outfile);/*G*/
          fputc(x,outfile);/*B*/
          fputc(0,outfile);
}

/*BODY SECTION*/
memset((unsigned char *)&image,0,bytes);
imagesize=XPIX*(long)YPIX;
printf("\nAllocated memory = %ld bytes",imagesize);
data=(unsigned char huge*)farcalloc(imagesize,
                          sizeof(unsigned char));
if(data==NULL)
  {
   printf("\nAllocated error");
   exit(1);
  }
printf("\nMemory allocation complete");

printf("\nReading input file");
for(y=0;y<bmp.depth;++y){
          if(y%40==0) printf("\n");
          printf(".");
          fread(image,sizeof(char),XPIX,infile);
          for(x=0;x<XPIX;x++)
            {
         index=XPIX*(long)y+x;
             data[index]=image[x];
        }
}

printf("\nWriting output file and flip vertical");
for(y=0;y<bmp.depth;++y){
          if(y%40==0) printf("\n");
          printf(".");
          for(x=0;x<XPIX;x++)
            {
         index=XPIX*(long)((YPIX-1)-y)+x;
             image[x]=data[index];
        }
          fwrite(image,sizeof(char),bytes,outfile);
}

printf("\nClosing allocated memory");
farfree(data);
   printf("\nClosing both file");
   fclose(infile);
   fclose(outfile);
 } while(1);
}
```

การแปลงภาพในรูปแบบบิทแมพเป็นภาพแอสกี้

ภาพถ่ายที่ต้องการแปลงเป็นภาพสีเทาในรูปแบบบิทแมพขนาด 8 บิทต่อหนึ่งจุดภาพ เป็นข้อมูลภาพที่มีเฮดเดอร์ ในการแปลงจะปลดเฮดเดอร์ออก โดยจะรับชื่อไฟล์ที่ต้องการแปลงเท่านั้น สำหรับขนาดจุดภาพต่อหนึ่งเส้น (ขนาดทางแกน x) จำนวนเส้นของภาพ (ขนาดทางแกน y) โปรแกรมจะอ่านออกมาจากเฮดเดอร์เอง ลำดับอัลกอริทึมในการทำงานของโปรแกรมดังนี้



โฟล์วชาร์ตการทำงานของโปรแกรมแปลงภาพในรูปแบบบิทแมพเป็นภาพแอสกี้

```
/*---------------------------------------
PROGRAM         ->CHANGE BMP TO IMAGE FILE..
VERSION         ->3.0... (on Windows)
BY              ->ARMOTE SOMBOONKAEW...
START WRITTEN   ->7 JUNE 1993...
LAST UPDATE     ->18 MARCH 1994...
INSTITUTE       ->KMIT'L...
COMMENT         ->...have output (*.img) that add
                    to original file...
---------------------------------------*/

#include  <stdio.h>
#include  <dos.h>
#include  <string.h>
#include  <conio.h>
#include  <stdlib.h>
#include  <alloc.h>

#define   pixel2bytes(n)  ((n+7)/8)

int       XPIX,YPIX;


typedef struct  {
                char id[2];
                long filesize;
                int reserved[2];
                long headersize;
                long infoSize;
                long width;
                long depth;
                int  biPlanes;
                int  bits;
                long biCompression;
                long biSizeImage;
                long biXPelsPerMeter;
                long biYPelsPerMeter;
                long biClrUsed;
                long biClrImportant;
                }BMPHEAD;

main()
{
int                     x,y,bytes;
char                    *n;
FILE                    *infile, *outfile;
static char             name[80];
static unsigned char    image[1024];
BMPHEAD                 bmp;
unsigned char           huge *data;
unsigned long           index,imagesize;

clrscr();
printf("\nCONVERT...BMP TO IMAGE...VERSION 3.0
                                (on Windows)");
printf("\nBY..ARMOTE SOMBOONKAEW...");

do {
   printf("\nInput from  -> "); scanf("%s",&name);
   printf("\nOutput  to  -> (*.img)...");

   printf("\nReading BMP file [%s]",name);
   infile = fopen(name,"rb");
   n=strchr(name,'.');
   if(n) strcpy(n,".img");
   else  strcat(name,".img");

   printf("\nSaving IMG file [%s]",name);
   outfile = fopen(name,"wb");
   if((infile&&outfile) == NULL) {
        printf(" Open file error..\n");
        exit(1);
   }

   printf("\nRead header information");
   memset((char *)&bmp,0,(long)sizeof(BMPHEAD));
   if(fread((char *)&bmp,1,
        (long)sizeof(BMPHEAD),infile)
        != sizeof(BMPHEAD))
   {
        printf("fread HEADER error");
        exit(1);
   }
   printf("\nRead Header Complete!
            (xsize=%ld,ysize=%ld)\n",
                bmp.width,bmp.depth);
```

```
/*for THE odd XPIX*/
YPIX=bmp.depth;
XPIX=bytes=bmp.width;
if(bytes & 0x0003) {
        bytes |= 0x0003;
        ++bytes;
}
/*BODY SECTION*/
imagesize=XPIX*(long)YPIX;
printf("\nAllocated memory = %ld bytes",imagesize);
data=(unsigned char huge*)farcalloc(imagesize,
                sizeof(unsigned char));
if(data==NULL)
 {
  printf("\nAllocated error");
  exit(1);
 }
printf("\nMemory allocation complete");
fseek(infile,bmp.headersize,SEEK_SET);
memset((unsigned char *)&image,0,bytes);

printf("\nReading input file");
for(y=0;y<bmp.depth;++y)
  {
        if(y%40==0) printf("\n");
        printf(".");
        fread(image,sizeof(char),bytes,infile);
        for(x=0;x<XPIX;x++)
          {
                index=XPIX*(long)y+x;
                data[index]=image[x];
          }
  }

printf("\nWriting output file and flip vertical");
for(y=0;y<bmp.depth;++y)
  {
      if(y%40==0) printf("\n");
      printf(".");
      for(x=0;x<XPIX;x++)
        {
         index=XPIX*(long)((YPIX-1)-y)+x;
         image[x]=data[index];
        }
      fwrite(image,sizeof(char),XPIX,outfile);
}
      printf("\nClosing allocated memory");
      farfree(data);
      printf("\nClosing both file");
      fclose(infile);
      fclose(outfile);
    }while(1);
}
```

ภาคผนวก ค.

โปรแกรมการประมวลผล

ภาพถ่ายที่นำมาประมวลผลเป็นภาพสี 24 บิทต่อจุดภาพ เป็นข้อมูลภาพในรูปแบบบิทแมพที่มีเฮดเดอร์บอกรายละเอียดต่าง ๆ ของภาพทั้งหมด โปรแกรมการประมวลผลเขียนด้วยตัวแปลภาษาซี จาก Borland C++ ของบริษัทบอร์แลนด์ รุ่น 3.1 บนวินโดว์ โดยใช้ตัวทำเมนู ไดอะล็อก และไอคอน ด้วย App Studio จาก Microsoft Visual C++ รุ่น 1.0 ของบริษัทไมโครซอร์ฟ ประกอบไปด้วย 6 โปรแกรม คือ

- โปรแกรมปรับปรุงคอนทราสของภาพ โดยใช้ *การดึงเชิงเส้นอัตโนมัติ ด้วยค่าสูงสุดและต่ำสุด ที่หามาได้จากการกำหนดเปอร์เซนต์ทางด้านต่ำและสูง* ดังแสดงโฟล์วชาร์ตการทำงานดังรูปที่ 2.19 โดยโปรแกรมอยู่ในหน้าที่ ค-18
- โปรแกรมปรับปรุงรายละเอียดของขอบภาพโดยวิธีการของ *Shettigara และ Odins* ดังแสดงโฟล์วชาร์ตการทำงานดังรูปที่ 3.2 โดยโปรแกรมอยู่ในหน้าที่ ค-15
- โปรแกรมปรับปรุงรายละเอียดของขอบภาพโดยวิธีการใช้ *เทมเพล็ทแบบ 12 ทิศทาง* ดังแสดงโฟล์วชาร์ตการทำงานดังรูปที่ 3.6 โดยโปรแกรมอยู่ในหน้าที่ ค-10
- โปรแกรมปรับปรุงรายละเอียดของขอบภาพโดยวิธีการใช้ *ภาพผลลบที่เกิดจากการทำให้เรียบ* ดังแสดงโฟล์วชาร์ตการทำงานดังรูปที่ 4.1 โดยโปรแกรมอยู่ในหน้าที่ ค-6
- โปรแกรมปรับปรุงรายละเอียดของขอบภาพโดยวิธีการใช้ *ภาพผลลบที่เกิดจากการทำให้เรียบ ที่ใช้ตัวกรองเพียงตัวเดียว* ดังแสดงโฟล์วชาร์ตการทำงานดังรูปที่ 4.8 โดยโปรแกรมอยู่ในหน้าที่ ค-8
- โปรแกรมปรับปรุงรายละเอียดของขอบภาพด้วย *การแปลงภาพผลลบที่เกิดจากการทำให้เรียบ* ดังแสดงโฟล์วชาร์ตการทำงานดังรูปที่ 5.7 โดยโปรแกรมอยู่ในหน้าที่ ค-2

File name -> bmpdisp.c

```
/*---------------------------------------

PROGRAM         ->Image Enhancement...
VERSION         ->1.0...(on Windows)...
BY              ->ARMOTE
                SOMBOONKAEW...
WRITTEN         ->19 MARCH 1994...
LAST UPDATE     ->19 MARCH 1994...
INSTITUTE       ->KMIT'L...
COMMENT         ->Input (*.bmp)...
COMPLIER        ->Borland C++ Ver. 3.1
                  (on windows)...
---------------------------------------*/
#include  'windows.h'
#include  'bmpdisp.h'
#include  'commdlg.h'
#include  'math.h'
#include  'stdio.h'

HANDLE          hInst;
char            achFileName[128];
BITMAPFILEHEADER bf;
WORD            wOperation=IDM_TODEV;
BOOL            bDIBLoaded = FALSE;
WORD            offBits;
HANDLE          hDIBInfo = NULL;
HBITMAP         hDDBitmap = NULL;
HBITMAP         hOldBitmap;
HDC             hMemDC;
HDC             hDC;
DWORD           bmWidth, bmHeight,
                bmImgSize, bmBand,
                bmScanWidth;
LOGFONT         LogFont;
static HANDLE   hHelv;
int             MinScroll,
                MaxScroll,
                StartScroll=TRUE;
float           n[2],OrderCutoff[2],Step;
unsigned char   huge *UndoR,
                huge *UndoG,
                huge *UndoB;
int             Template=0,deno=9;
int             Style=0;
int             TxESMT_C[2][3][3]
                = {  -1,  -1,  -1,
                     -1,  17,  -1,
                     -1,  -1,  -1,

                     -1,  -2,  -1,
                     -2,  28,  -2,
                     -1,  -2,  -1 };
int             TxESMT_S[2][3][3]
                = {  1,  1,  1,
                     1,  1,  1,
                     1,  1,  1,

                     1,  2,  1,
                     2,  4,  2,
                     1,  2,  1 };
#include  'dibs16k.c'

/************************************

IMAGE PROCESSING
PROCEDURE:
Edge enhancement by Smoothing image
with Spurious edges elimination
with Butterworth function transform

************************************/

void DoESMT_B(HWND hWnd)
{
LPBITMAPINFOHEADER lpbi;
unsigned char   huge *DataR,
                huge *DataG,
                huge *DataB;
unsigned char   huge *OutR ,
                huge *OutG ,
                huge *OutB;
int             huge *BufR ,
                huge *BufG ,
                huge *BufB;
BYTE            huge *lpData;
unsigned long   BandSize;
unsigned long   pos,pos1,
                pos00,pos01,
                pos10,pos11;
int             x,y,i,j,f;
long            buf1,buf2,buf3;
static float    IBTW[256],
                fbuf1,fbuf2,
                fbuf3,fbuf4;
//----------var for LCS--------------
float           RatioS,fBuf;
int             xMaxS,xMinS,
                yMax,yMin;
static unsigned char LTBS[2048];
//---------------------------------

SetCursor( LoadCursor( NULL, IDC_WAIT ) );
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
lpData=lpbi;
```

File name -> bmpdisp.c

```
//saparate band allocated memory size
BandSize=bmWidth*(long)bmHeight;

//input data allocate
DataR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(DataR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataR", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }
DataG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(DataG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataG", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }
DataB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(DataB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataB", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//output data allocate
OutR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(OutR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutR",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(OutG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutG",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
BandSize) );
if(OutB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutB",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//split image and flip vertical for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos=bmWidth*(long)y+x;
   pos1=((bmScanWidth*(long)(bmHeight-1-
         y)+x)*bmBand)+offBits;
   DataB[pos]=*(lpData+pos1);
   DataG[pos]=*(lpData+pos1+1);
   DataR[pos]=*(lpData+pos1+2);
  }

//make Lookup table of IBTW function
for(x=0;x<256;x++)
 {
  if(x==0) f=1;
  else     f=x;
  IBTW[x]=255.0*((1.0/(1.0+(0.414*
           pow(((float)OrderCutoff[1]/(float)f),
           (float)OrderCutoff[0])))));
 }

//Enhance by Smoothing Image
for( y=1; y<bmHeight-1; y++)
 for( x=1; x<bmWidth-1 ; x++)
 {
  buf1=buf2=buf3=0;
  //low pass
  for(j=-1;j<=1;j++)
   for(i=-1;i<=1;i++)
    {
     pos  = bmWidth*(long)(y+j)+x+i;
     buf1 += TxESMT_S[Template][j+1][i+1] *
             DataR[pos];
     buf2 += TxESMT_S[Template][j+1][i+1] *
             DataG[pos];
     buf3 += TxESMT_S[Template][j+1][i+1] *
             DataB[pos];
    }
  buf1/=deno;
  buf2/=deno;
  buf3/=deno;

  pos  =bmWidth*(long)y+x;
```

File name -> bmpdisp.c

```
switch(Style)
        {
        case 0://low-pass component
                break;

        case 1://high-pass component
                buf1=DataR[pos]-buf1;
                if(buf1<0) buf1=0;
                buf2=DataG[pos]-buf2;
                if(buf2<0) buf2=0;
                buf3=DataB[pos]-buf3;
                if(buf3<0) buf3=0;
                break;

        case 2://IBTW transform
                buf1=DataR[pos]-buf1;
                if(buf1<0) buf1=0;
                buf2=DataG[pos]-buf2;
                if(buf2<0) buf2=0;
                buf3=DataB[pos]-buf3;
                if(buf3<0) buf3=0;

                buf1=IBTW[buf1];
                buf2=IBTW[buf2];
                buf3=IBTW[buf3];
                break;

        case 3://linear adding enhancement
                buf1=DataR[pos]-buf1;
                if(buf1<0) buf1=0;

                buf2=DataG[pos]-buf2;
                if(buf2<0) buf2=0;

                buf3=DataB[pos]-buf3;
                if(buf3<0) buf3=0;

                buf1=IBTW[buf1];
                buf2=IBTW[buf2];
                buf3=IBTW[buf3];

                buf1+=DataR[pos];
                buf2+=DataG[pos];
                buf3+=DataB[pos];
                break;
        }
//protect out of range
if(buf1<0)      buf1=0;
else if(buf1>255) buf1=255;
if(buf2<0)      buf2=0;
else if(buf2>255) buf2=255;
if(buf3<0)      buf3=0;
else if(buf3>255) buf3=255;
  //resultance
  OutR[pos]=buf1;
  OutG[pos]=buf2;
  OutB[pos]=buf3;
 }

//Linear Contrast Stretching
//clear LTB
/*for(x=0;x<2048;x++)
  LTBS[x]=0;
yMax=255;
yMin=0;
//find maximum & minimum gray
xMaxS=-1023;xMinS=1023;
for(y=1;y<bmHeight-1;y++)
 for(x=1;x<bmWidth-1;x++)
  {
   pos = bmWidth*(long)y+x;
   xMaxS = max(xMaxS,BufR[pos]);
   xMaxS = max(xMaxS,BufG[pos]);
   xMaxS = max(xMaxS,BufB[pos]);

   xMinS = min(xMinS,BufR[pos]);
   xMinS = min(xMinS,BufG[pos]);
   xMinS = min(xMinS,BufB[pos]);
  }

 //find ratio
 RatioS = (yMax-yMin)/(float)(xMaxS-xMinS);

 //make stretching lookup table
 for(x=xMinS;x<xMaxS;x++)
  {
   fBuf=((x-xMinS)*RatioS+yMin)+0.5;
   //rounding with 0.5
   if(fBuf>255.0) fBuf=255.0;//maximize
   if(fBuf<0)     fBuf=0.0;  //minimize
   LTBS[x-xMinS]=(BYTE)(int)fBuf;
  }

 //transform image
 for(y=1;y<bmHeight-1;y++)
  for(x=1;x<bmWidth-1;x++)
  {
   pos=bmWidth*(long)y+x;
   OutR[pos]=LTBS[BufR[pos]-xMinS];
   OutG[pos]=LTBS[BufG[pos]-xMinS];
   OutB[pos]=LTBS[BufB[pos]-xMinS];
  }*/
```

File name -> bmpdisp.c

```
//merge image RGB through original one
//and anti-vertical flip for standard bmp format
for(  y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
    pos  = bmWidth*(long)y+x;
    pos1=((bmScanWidth*(long)(bmHeight-1-
          y)+x)*bmBand)+offBits;
    *(lpData+pos1)    =OutB[pos];
    *(lpData+pos1+1)=OutG[pos];
    *(lpData+pos1+2)=OutR[pos];
  }
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataB)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutB)));
GlobalUnlock(hDIBInfo);
SetCursor( LoadCursor( NULL, IDC_ARROW ) );
}
/*-------------------------------------*/
BOOL FAR PASCAL BTWDlg( HWND hDlg,
                        WORD msg,
                        WORD wParam,
                        LONG lParam )

{
static     strbuf[10];
char       szAspect[10];
float      nl;
int        h,i,j;
switch( msg )
 {
 case WM_INITDIALOG:
          for(h=1;h<3;h++)
           for(i=0;i<3;i++)
            for(j=0;j<3;j++)
             {
             sprintf( szAspect, '%d',
                    TxESMT_S[h-1][i][j]);
             SendDlgItemMessage( hDlg,
                                 1000*h+i*3+j ,
                                 EM_LIMITTEXT,
                                 10, 0L );
             SetDlgItemText( hDlg, 1000*h+i*3+j ,
                             szAspect );
             }
          if(Template>1) Template=1;
          if(Template<0) Template=0;
          CheckRadioButton( hDlg, 3000, 3001,
                            3000+Template);
          if(Style>3) Style=3;
          if(Style<0) Style=0;
          CheckRadioButton( hDlg, 5000, 5003,
                            5000+Style);
          for(i=0;i<2;i++)
 {
           sprintf( szAspect,'%3.2f', OrderCutoff[i]);
           SendDlgItemMessage( hDlg, 4000+i ,
                               EM_LIMITTEXT,
                               10, 0L );
           SetDlgItemText( hDlg, 4000+i , szAspect );
 }
          return(TRUE);

case WM_HSCROLL:
          i = GetWindowWord(HIWORD(lParam),
                            GWW_ID);
          switch(i)
           {
           case 101:j=0;break;
           case 102:j=1;break;
           }
          if(StartScroll)
           {
           GetScrollRange(GetFocus(),SB_CTL,
                          &MinScroll,&MaxScroll);
           Stepl=(MaxScroll-MinScroll)/255.0;
           StartScroll=FALSE;
           OrderCutoff[0]=0;
           OrderCutoff[1]=0;
           }
          switch(wParam)
           {
           case SB_LINEUP:      nl = -1; break;
           case SB_LINEDOWN: nl = 1;  break;
           case SB_PAGEUP:      nl = -10;break;
           case SB_PAGEDOWN:nl = 10; break;
           case SB_THUMBPOSITION:
                nl = LOWORD(lParam)-OrderCutoff[j];
                break;
           default: nl=0;;
           }
          if(nl == max(-OrderCutoff[j],
                   min(nl,255.0-OrderCutoff[j])))
           {
           OrderCutoff[j] += nl;
           SetScrollPos(GetFocus(),SB_CTL,
                  (int)(OrderCutoff[j]*Stepl),TRUE);
           sprintf( szAspect,'%3.2f', OrderCutoff[j]);
           SendDlgItemMessage( hDlg, 4000+j,
                               EM_LIMITTEXT, 10,
                               0L );
           SetDlgItemText( hDlg, 4000+j, szAspect );
           }
        return(FALSE);
```

File name -> bmpdisp.c

```
case WM_COMMAND:
        switch( wParam )
        {
        case 3000:Template=0;deno=9;  break;
        case 3001:Template=1;deno=16;break;
        case 5000:Style=0;break;
        case 5001:Style=1;break;
        case 5002:Style=2;break;
        case 5003:Style=3;break;
        case IDOK:
        case IDCANCEL: EndDialog( hDlg, TRUE );
                       return( TRUE );
        default: return( TRUE );
        }
}
return( FALSE );
}
```

```
/*************************************************

IMAGE PROCESSING PROCEDURE:
edge enhancement
by Smoothing image
used step by step Method

*************************************************/

void DoESMT_S(HWND hWnd)
{
LPBITMAPINFOHEADER      lpbi;
unsigned char           huge *DataR,
                        huge *DataG,
                        huge *DataB;
unsigned char           huge *OutR ,
                        huge *OutG ,
                        huge *OutB;
BYTE                    huge *lpData;
unsigned long           BandSize;
unsigned long           pos,pos1,
                        pos00,pos01,
                        pos10,pos11;
int                     x,y,buf1,buf2,buf3,i,j;

SetCursor( LoadCursor( NULL, IDC_WAIT ) );
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
lpData=lpbi;

//saparate band allocated memory size
BandSize=bmWidth*(long)bmHeight;

//input data allocate
DataR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(DataR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
      DataR", "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
DataG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(DataG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
      DataG", "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
```

File name -> bmpdisp.c

```
DataB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(DataB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
      DataB", "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//output data allocate
OutR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(OutR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutR",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(OutG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutG",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                        BandSize) );
if(OutB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutB",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//split image and flip vertical for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos=bmWidth*(long)y+x;
   pos1=((bmScanWidth*
        (long)(bmHeight-1-y)+x)*bmBand)+offBits;
   DataB[pos]=*(lpData+pos1);
   DataG[pos]=*(lpData+pos1+1);
   DataR[pos]=*(lpData+pos1+2);
  }

//Enhance by Smoothing image
for( y=1; y<bmHeight-1; y++)
 for( x=1; x<bmWidth-1 ; x++)
  {
   buf1=buf2=buf3=0;
   //low pass
   for(j=-1;j<=1;j++)
    for(i=-1;i<=1;i++)
     {
      pos  = bmWidth*(long)(y+j)+x+i;
      buf1 += TxESMT_S[Template][j+1][i+1] *
              DataR[pos];
      buf2 += TxESMT_S[Template][j+1][i+1] *
              DataG[pos];
      buf3 += TxESMT_S[Template][j+1][i+1] *
              DataB[pos];
     }
   buf1/=deno;
   buf2/=deno;
   buf3/=deno;
   pos =bmWidth*(long)y+x;

   //high pass
   buf1=DataR[pos]-buf1;
   buf2=DataG[pos]-buf2;
   buf3=DataB[pos]-buf3;

   //used single inner edge only
   if(buf1<0)     buf1=0;
   if(buf2<0)     buf2=0;
   if(buf3<0)     buf3=0;

   //enhance
   buf1+=DataR[pos];
   buf2+=DataG[pos];
   buf3+=DataB[pos];

   //protect out of range
   if(buf1>255) buf1=255;
   if(buf2>255) buf2=255;
   if(buf3>255) buf3=255;
   //resultance
   OutR[pos]=buf1;
   OutG[pos]=buf2;
   OutB[pos]=buf3;
  }
//merge image RGB through original one
//and anti-vertical flip for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos = bmWidth*(long)y+x;
   pos1=((bmScanWidth*
        (long)(bmHeight-1-y)+x)*bmBand)+offBits;
   *(lpData+pos1)   =OutB[pos];
   *(lpData+pos1+1)=OutG[pos];
   *(lpData+pos1+2)=OutR[pos];
  }
```

File name -> bmpdisp.c

```
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataB)));

GlobalFree((HANDLE)GlobalHandle(HIWORD(OutR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutB)));
GlobalUnlock(hDIBInfo);
SetCursor( LoadCursor(.NULL, IDC_ARROW ) );
}
/*-------------------------------------*/
BOOL FAR PASCAL ESMT_S_Dlg( HWND hDlg,
                            WORD msg,
                            WORD wParam,
                            LONG lParam )
{
static    strbuf[10];
char      szAspect[10];
int       nHInc,h,i,j;
//WORD status;
switch( msg )
 {
 case WM_INITDIALOG:
        for(h=1;h<3;h++)
         for(i=0;i<3;i++)
          for(j=0;j<3;j++)
           {
           sprintf( szAspect, "%d",
                  TxESMT_S[h-1][i][j]);
           SendDlgItemMessage( hDlg,
                               1000*h+i*3+j,
                               EM_LIMITTEXT,
                               10, 0L );
           SetDlgItemText( hDlg, 1000*h+i*3+j,
                           szAspect );
           }
        if(Template>1) Template=1;
        CheckRadioButton( hDlg, 3000, 3001,
                          3000+Template);
        return(TRUE);

 case WM_COMMAND:
        switch( wParam )
         {
         case 3000:Template=0;deno=9;  break;
         case 3001:Template=1;deno=16;break;
         case IDOK:EndDialog( hDlg, TRUE );
                   return( TRUE );
         default: return( TRUE );
         }
 }
return( FALSE );
}
```

```
/*********************************************

IMAGE PROCESSING PROCEDURE:
edge enhancement by Smoothing Image
used convolution Method

*********************************************/

void DoESMT_C(HWND hWnd)
{
LPBITMAPINFOHEADER      lpbi;
unsigned char           huge *DataR,
                        huge *DataG,
                        huge *DataB;
unsigned char           huge *OutR ,
                        huge *OutG ,
                        huge *OutB;
BYTE                    huge *lpData;
unsigned long           BandSize;
unsigned long           pos,pos1,
                        pos00,pos01,
                        pos10,pos11;
int                     x,y,buf1,buf2,buf3,j,i;

SetCursor( LoadCursor( NULL, IDC_WAIT ) );
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
lpData=lpbi;

//saparate band allocated memory size
BandSize=bmWidth*(long)bmHeight;

//input data allocate
DataR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(DataR==NULL)
 {
 MessageBox(hWnd, "Can't create Mem of the
            DataR", "Error",
            MB_ICONSTOP | MB_OK);
 return FALSE;
 }
DataG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(DataG==NULL)
 {
 MessageBox(hWnd, "Can't create Mem of the
            DataG", "Error",
            MB_ICONSTOP | MB_OK);
 return FALSE;
 }
```

File name -> bmpdisp.c

```
DataB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(DataB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataB", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//output data allocate
OutR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(OutR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutR",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(OutG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutG",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(OutB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutB",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//split image and flip vertical for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos  =bmWidth*(long)y+x;
   pos1=((bmScanWidth*
         (long)(bmHeight-1-y)+x)*bmBand)+offBits;
   DataB[pos]=*(lpData+pos1);
   DataG[pos]=*(lpData+pos1+1);
   DataR[pos]=*(lpData+pos1+2);
  }
//Edge Enhance Using Subtracted Smoothing Image
//by Convolution Filter
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   if(x==0||y==0||x==(bmWidth-1)||y==
                    (bmHeight-1))
   {
        pos=bmWidth*(long)y+x;
        OutR[pos]=DataR[pos];
        OutG[pos]=DataG[pos];
        OutB[pos]=DataB[pos];
   }
   else
   {
        buf1=buf2=buf3=0;
        for(j=-1;j<=1;j++)
         for(i=-1;i<=1;i++)
          {
           pos  = bmWidth*(long)(y+j)+x+i;
           buf1 += TxESMT_C[Template][j+1][i+1] *
                   DataR[pos];
           buf2 += TxESMT_C[Template][j+1][i+1] *
                   DataG[pos];
           buf3 += TxESMT_C[Template][j+1][i+1] *
                   DataB[pos];
          }
        buf1/=deno;
        buf2/=deno;
        buf3/=deno;

        if(buf1<0)      buf1=0;
        else if(buf1>255) buf1=255;
        if(buf2<0)      buf2=0;
        else if(buf2>255) buf2=255;
        if(buf3<0)      buf3=0;
        else if(buf3>255) buf3=255;
        pos  =bmWidth*(long)y+x;
        OutR[pos]=buf1;
        OutG[pos]=buf2;
        OutB[pos]=buf3;
   }
  }

//merge image RGB through original one
//and anti-vertical flip for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos  = bmWidth*(long)y+x;
   pos1=((bmScanWidth*
         (long)(bmHeight-1-y)+x)*bmBand)+offBits;
   *(lpData+pos1)   =OutB[pos];
   *(lpData+pos1+1) =OutG[pos];
   *(lpData+pos1+2)=OutR[pos];
  }
```

File name -> bmpdisp.c

```
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataB)));

GlobalFree((HANDLE)GlobalHandle(HIWORD(OutR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutB)));
GlobalUnlock(hDIBInfo);
SetCursor( LoadCursor( NULL, IDC_ARROW ) );
}
/*--------------------------------------*/
BOOL FAR PASCAL ESMT_C_Dlg( HWND hDlg,
                            WORD msg,
                            WORD wParam,
                            LONG lParam )
{
static    strbuf[10];
char      szAspect[10];
int       nHInc,h,i,j;
//WORD status;
switch( msg )
 {
 case WM_INITDIALOG:
        for(h=1;h<3;h++)
         for(i=0;i<3;i++)
          for(j=0;j<3;j++)
           {
           sprintf( szAspect, "%d",
                  TxESMT_C[h-1][i][j]);
           SendDlgItemMessage( hDlg,
                               1000*h+i*3+j,
                               EM_LIMITTEXT,
                               10, 0L );
           SetDlgItemText( hDlg, 1000*h+i*3+j,
                           szAspect );
           }
        if(Template>1) Template=1;
        CheckRadioButton( hDlg, 3000, 3001,
                          3000+Template);
        return(TRUE);
 case WM_COMMAND:
        switch( wParam )
        {
         case 3000:Template=0;deno=9;  break;
         case 3001:Template=1;deno=16;break;
         case IDOK:EndDialog( hDlg, TRUE );
                   return( TRUE );
         default: return( TRUE );
        }
}
return( FALSE );
}
```

```
/*********************************************

IMAGE PROCESSING PROCEDURE:
edge enhancement by 12 Template Method

*********************************************/


void Do12Template(HWND hWnd)
{
LPBITMAPINFOHEADER      lpbi;
unsigned char           huge *DataR,
                        huge *DataG,
                        huge *DataB;
unsigned char           huge *OutR ,
                        huge *OutG ,
                        huge *OutB;
BYTE                    huge *lpData;
unsigned long           BandSize;
unsigned long           pos,pos1,
                        pos00,pos01,
                        pos10,pos11;
int                     x,y,buf1,buf2,buf3;
long                    LongSumR[12],
                        LongSumG[12],
                        LongSumB[12];
unsigned long           TempCountR[12],
                        TempCountG[12],
                        TempCountB[12];
long                    GradMaxR, GradMaxG,
                        GradMaxB;
int                     TempNoR,TempNoG,
                        TempNoB;
int                     i,j,k;
FILE                    *outfile;
static char             name[80];
float                   RatioS,fBuf;
int                     xMaxS,xMinS,yMax,yMin;
static unsigned char    LTBS[2048];
char                    *pname;

int             TxT12[12][7][7] = {

0,0,0,0,0,0,0,//T#1
0,0,0,0,0,0,0,
0,0,0,0,0,0,0,
1,1,1,0,-1,-1,-1,
0,0,0,0,0,0,0,
0,0,0,0,0,0,0,
0,0,0,0,0,0,0,
```

File name -> bmpdisp.c

```
0,0,0, 1,0,0,0,//T#2
0,0,0, 1,0,0,0,
0,0,0, 1,0,0,0,
0,0,0, 0,0,0,0,
0,0,0,-1,0,0,0,
0,0,0,-1,0,0,0,
0,0,0,-1,0,0,0,

1,0,0,0,0,0,0,//T#3
0,1,0,0,0,0,0,
0,0,1,0,0,0,0,
0,0,0,0,0,0,0,
0,0,0,0,-1,0,0,
0,0,0,0,0,-1,0,
0,0,0,0,0,0,-1,

0,0,0,0,0,0,1,//T#4
0,0,0,0,0,1,0,
0,0,0,0,1,0,0,
0,0,0,0,0,0,0,
0,0,-1,0,0,0,0,
0,-1,0,0,0,0,0,
-1,0,0,0,0,0,0,

0,0,0,0,0,0,0,//T#5
0,0,0,0,0,0,0,
1,1,0,0,0,0,0,
0,0,1,0,-1,0,0,
0,0,0,0,0,-1,-1,
0,0,0,0,0,0,0,
0,0,0,0,0,0,0,

0,0,0,0,0,0,0,//T#6
0,0,0,0,0,0,0,
0,0,0,0,0,1,1,
0,0,-1,0,1,0,0,
-1,-1,0,0,0,0,0,
0,0,0,0,0,0,0,
0,0,0,0,0,0,0,

0,0,1,0,0,0,0,//T#7
0,0,1,0,0,0,0,
0,0,0,1,0,0,0,
0,0,0,0,0,0,0,
0,0,0,-1,0,0,0,
0,0,0,0,-1,0,0,
0,0,0,0,-1,0,0,

0,0,0,0,1,0,0,//T#8
0,0,0,0,1,0,0,
0,0,0,1,0,0,0,
0,0,0,0,0,0,0,
0,0,0,-1,0,0,0,
0,0,-1,0,0,0,0,
0,0,-1,0,0,0,0,

0,1,0,0,0,0,0,//T#9
0,1,0,0,0,0,0,
0,0,1,0,0,0,0,
0,0,0,0,0,0,0,
0,0,0,0,-1,0,0,
0,0,0,0,0,-1,0,
0,0,0,0,0,-1,0,

0,0,0,0,0,1,0,//T#10
0,0,0,0,0,1,0,
0,0,0,0,1,0,0,
0,0,0,0,0,0,0,
0,0,-1,0,0,0,0,
0,-1,0,0,0,0,0,
0,-1,0,0,0,0,0,

0,0,0,0,0,0,0,//T#11
1,1,0,0,0,0,0,
0,0,1,0,0,0,0,
0,0,0,0,0,0,0,
0,0,0,0,-1,0,0,
0,0,0,0,0,-1,-1,
0,0,0,0,0,0,0,

0,0,0,0,0,0,0,//T#12
0,0,0,0,0,1,1,
0,0,0,0,1,0,0,
0,0,0,0,0,0,0,
0,0,-1,0,0,0,0,
-1,-1,0,0,0,0,0,
0,0,0,0,0,0,0 };

SetCursor( LoadCursor( NULL, IDC_WAIT ) );
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
lpData=lpbi;

//saparate band allocated memory size
BandSize=bmWidth*(long)bmHeight;

//input data allocate
DataR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
```

File name -> bmpdisp.c

```
if(DataR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataR", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }
DataG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(DataG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataG", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }
DataB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(DataB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataB", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//output data allocate
OutR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(OutR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutR",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(OutG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutG",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(OutB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutB",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//split image and flip vertical for standard bmp format
for(  y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos=bmWidth*(long)y+x;
   pos1=((bmScanWidth*
         (long)(bmHeight-1-y)+x)*bmBand)+offBits;
   DataB[pos]=*(lpData+pos1);
   DataG[pos]=*(lpData+pos1+1);
   DataR[pos]=*(lpData+pos1+2);
  }

//12 template edge adding
//clear all array
for(k=0;k<12;k++)
  {
  TempCountR[k]=TempCountG[k]=TempCountB[k]=0;
  LongSumR[k]=LongSumG[k]=LongSumB[k]=0;
  }
//find all grad
for(  y=3; y<bmHeight-3; y++)
 for( x=3; x<bmWidth-3 ; x++)
  {//start big for
   for(k=0;k<12;k++)
   {//start for#1
    for(j=-3;j<=3;j++)
     for(i=-3;i<=3;i++)
      if((TxT12[k][j+3][i+3])!=0)
      {
       pos = bmWidth*(long)(y+j)+x+i;
       LongSumR[k] += (TxT12[k][j+3][i+3] *
                       (int)DataR[pos]);
       LongSumG[k] += (TxT12[k][j+3][i+3] *
                       (int)DataG[pos]);
       LongSumB[k] += (TxT12[k][j+3][i+3] *
                       (int)DataB[pos]);
      }
    LongSumR[k]=abs(LongSumR[k]/3);
    LongSumG[k]=abs(LongSumG[k]/3);
    LongSumB[k]=abs(LongSumB[k]/3);
   }//end for#1

   //find max grad
   GradMaxR=GradMaxG=GradMaxB=0;
   TempNoR=TempNoG=TempNoB=0;
   for(k=0;k<12;k++)
   {//start for#2
        if(LongSumR[k]>GradMaxR)
         {
          GradMaxR=LongSumR[k];
          TempNoR=k;
         }
```

File name -> bmpdisp.c

```
        if(LongSumG[k]>GradMaxG)
          {
           GradMaxG=LongSumG[k];
           TempNoG=k;
          }
        if(LongSumB[k]>GradMaxB)
          {
           GradMaxB=LongSumB[k];
           TempNoB=k;
          }
    }//end for#2
    //count frequency used of template
    TempCountR[TempNoR]++;
    TempCountG[TempNoG]++;
    TempCountB[TempNoB]++;

    pos  =bmWidth*(long)y+x;
    switch(Template)
    {
    case 0://gradient information
          if(GradMaxR>n[0])  buf1=GradMaxR;
          else               buf1=0;
          if(GradMaxG>n[0])  buf2=GradMaxG;
          else               buf2=0;
          if(GradMaxB>n[0])  buf3=GradMaxB;
          else               buf3=0;
          break;
    case 1://adding gradiant with maximum grad
          if(GradMaxR>n[0])
            buf1=DataR[pos]+(GradMaxR);
          else       buf1=DataR[pos];
          if(GradMaxG>n[0])
            buf2=DataG[pos]+(GradMaxG);
          else       buf2=DataG[pos];
          if(GradMaxB>n[0])
            buf3=DataB[pos]+(GradMaxB);
          else       buf3=DataB[pos];
          break;
    }

    //limitted grayscale 0-255
    if(buf1<0) buf1=0; else if(buf1>255) buf1=255;
    if(buf2<0) buf2=0; else if(buf2>255) buf2=255;
    if(buf3<0) buf3=0; else if(buf3>255) buf3=255;

    OutR[pos]=buf1;
    OutG[pos]=buf2;
    OutB[pos]=buf3;
  }//end big for

/*-------------------------------------*/
```

```
//write frequency of used template
 strcpy(name,achFileName);
 pname=strchr(name,'.');
 if(pname)
   strcpy(pname,".his");
 else
   strcat(name,".his");
 if((outfile = fopen(name,"wb"))==NULL)
   {
    printf(" Write file error..\n");
    exit(0);
   }

 //write body data
 for(k=0;k<12;k++)
  {
   fwrite(&TempCountR[k],1,sizeof(long),outfile);
   fwrite(&TempCountG[k],1,sizeof(long),outfile);
   fwrite(&TempCountB[k],1,sizeof(long),outfile);
  }
 fclose(outfile);
 /*-------------------------------------*/

 //merge image RGB through original one
 //and anti-vertical flip for standard bmp format
 for(  y=3; y<bmHeight-3; y++)
  for( x=3; x<bmWidth-3 ; x++)
   {
    pos  = bmWidth*(long)y+x;
    pos1=((bmScanWidth*
          (long)(bmHeight-1-y)+x)*bmBand)+offBits;
    *(lpData+pos1)   =OutB[pos];
    *(lpData+pos1+1)=OutG[pos];
    *(lpData+pos1+2)=OutR[pos];
   }

 GlobalFree((HANDLE)GlobalHandle(HIWORD(DataR)));
 GlobalFree((HANDLE)GlobalHandle(HIWORD(DataG)));
 GlobalFree((HANDLE)GlobalHandle(HIWORD(DataB)));

 GlobalFree((HANDLE)GlobalHandle(HIWORD(OutR)));
 GlobalFree((HANDLE)GlobalHandle(HIWORD(OutG)));
 GlobalFree((HANDLE)GlobalHandle(HIWORD(OutB)));
 GlobalUnlock(hDIBInfo);
 SetCursor( LoadCursor( NULL, IDC_ARROW ) );
}
```

File name -> bmpdisp.c

```
/*--------------------------------------*/
BOOL FAR PASCAL T12Dlg( HWND hDlg,
                        WORD msg,
                        WORD wParam,
                        LONG lParam )
{
char    strbuf[10];
float   nl;
int     i,j;
switch(msg)
 {
 case WM_INITDIALOG:
    if(Template>1) Template=1;
    if(Template<0) Template=0;
    CheckRadioButton( hDlg, 2000, 2001,
                      2000+Template);
    sprintf(strbuf,"%3.0f", n[0]);
    SendDlgItemMessage(hDlg, 1000 ,
                       EM_LIMITTEXT, 10, 0L);
    SetDlgItemText(hDlg, 1000 , strbuf );
    return(TRUE);

 case WM_HSCROLL:
       if(StartScroll)
        {

GetScrollRange(GetFocus(),SB_CTL,&MinScroll,
               &MaxScroll);
          Stepl=(MaxScroll-MinScroll)/255.0;
          StartScroll=FALSE;
          n[0]=0;
        }
       switch(wParam)
        {
          case SB_LINEUP:     nl = -1; break;
          case SB_LINEDOWN:   nl =  1; break;
          case SB_PAGEUP:     nl = -10;break;
          case SB_PAGEDOWN:   nl = 10; break;
          case SB_THUMBPOSITION:
               nl = LOWORD(lParam)-n[0]; break;
          default: nl=0;;
        }
       if(nl == max(-n[0],min(nl,255-n[0])))
        {
          n[0] += nl;
          SetScrollPos(GetFocus(),SB_CTL,
                       (int)(n[0]*Stepl),TRUE);
          sprintf(strbuf,"%3.0f",n[0]);
          SendDlgItemMessage(hDlg, 1000,
                             EM_LIMITTEXT,
                             10, 0L );
          SetDlgItemText(hDlg, 1000, strbuf );
        }
          return(FALSE);

 case WM_COMMAND:
          switch( wParam )
          {
          case 2000:Template=0;break;
          case 2001:Template=1;break;
          case IDOK:
          case IDCANCEL: EndDialog( hDlg, TRUE );
                         return( TRUE );
          default: return( TRUE );
          }
 }
 return( FALSE );
 }
```

File name -> bmpdisp.c

```
/*************************************************

IMAGE PROCESSING PROCEDURE:
edge enhancement by Shettigara and Odins method

*************************************************/
void DoShetAndOdins(HWND hWnd)
{
LPBITMAPINFOHEADER      lpbi;
unsigned char           huge *DataR,
                        huge *DataG,
                        huge *DataB;
unsigned char           huge *OutR ,
                        huge *OutG ,
                        huge *OutB;
BYTE                    huge *lpData;
unsigned long           BandSize;
unsigned long           pos,pos1,
                        pos00,pos01,
                        pos10,pos11,pos2;
int                     x,y,buf1,buf2,buf3,i,j;
DWORD                   n1,n2,n3,p1,p2,p3;
int                     tn,N;
DWORD                   VSumR,VSumG,
                        VSumB,HSumR,
                        HSumG,HSumB;
DWORD                   SumR,SumG,SumB;
SetCursor( LoadCursor( NULL, IDC_WAIT ) );
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
lpData=lpbi;

//saparate band allocated memory size
BandSize=bmWidth*(long)bmHeight;

//input data allocate
DataR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(DataR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataR", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }
DataG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(DataG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataG", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }
DataB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(DataB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataB", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//output data allocate
OutR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(OutR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutR",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(OutG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutG",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                BandSize) );
if(OutB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutB",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//split image and flip vertical for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos=bmWidth*(long)y+x;
   pos1=((bmScanWidth*
         (long)(bmHeight-1-y)+x)*bmBand)+offBits;
   DataB[pos]=*(lpData+pos1);
   DataG[pos]=*(lpData+pos1+1);
   DataR[pos]=*(lpData+pos1+2);
  }
```

File name -> bmpdisp.c

```
//Shettigara and Odins in image enhancement
//#1 find ver and hor gradient
tn=n[0];//number of half template
//hor mean gradient
for( y=tn; y<bmHeight-tn; y++)
 for( x=tn; x<bmWidth-tn ; x++)
 {//start loop
 VSumR=VSumG=VSumB=0;
 HSumR=HSumG=HSumB=0;
 //hor gradient
 n1=n2=n3=p1=p2=p3=0;
 for(i=1;i<=tn;i++)
  {
    pos1 =bmWidth*(long)y+x-tn;
    pos2 =bmWidth*(long)y+x+tn;
    n1+=DataR[pos1]; p1+=DataR[pos2];
    n2+=DataG[pos1]; p2+=DataG[pos2];
    n3+=DataB[pos1]; p3+=DataB[pos2];
  }
 n1/=tn; n2/=tn; n3/=tn;
 p1/=tn; p2/=tn; p3/=tn;
 HSumR+=abs(n1-p1);
 HSumG+=abs(n2-p2);
 HSumB+=abs(n3-p3);
 //ver
 n1=n2=n3=p1=p2=p3=0;
 for(i=1;i<=tn;i++)
  {
    pos1 =bmWidth*(long)(y-tn)+x;
    pos2 =bmWidth*(long)(y+tn)+x;
    n1+=DataR[pos1]; p1+=DataR[pos2];
    n2+=DataG[pos1]; p2+=DataG[pos2];
    n3+=DataB[pos1]; p3+=DataB[pos2];
  }
 n1/=tn; n2/=tn; n3/=tn;
 p1/=tn; p2/=tn; p3/=tn;

 VSumR+=abs(n1-p1);
 VSumG+=abs(n2-p2);
 VSumB+=abs(n3-p3);

 SumR=sqrt(VSumR*VSumR+HSumR*HSumR);
 SumG=sqrt(VSumG*VSumG+HSumG*HSumG);
 SumB=sqrt(VSumB*VSumB+HSumB*HSumB);

 pos =bmWidth*(long)y+x;
 switch(Template)
    {
    case 0://gradient
          if(SumR>n[1])   buf1=SumR;
          else            buf1=0;
          if(SumG>n[1])   buf2=SumG;
          else            buf2=0;
          if(SumB>n[1])   buf3=SumB;
          else            buf3=0;
          break;
    case 1://adding gragient to original
          if(SumR>n[1])   buf1=DataR[pos]+SumR;
          else            buf1=DataR[pos];

          if(SumG>n[1])   buf2=DataG[pos]+SumG;
          else            buf2=DataG[pos];

          if(SumB>n[1])   buf3=DataB[pos]+SumB;
          else            buf3=DataB[pos];
          break;
    }

 if(buf1<0)        buf1=0;
 else if(buf1>255) buf1=255;

 if(buf2<0)        buf2=0;
 else if(buf2>255) buf2=255;

 if(buf3<0)        buf3=0;
 else if(buf3>255) buf3=255;

 OutR[pos]=buf1;
 OutG[pos]=buf2;
 OutB[pos]=buf3;
 }

 //#2 MEAN GRAD
 /*for( y=n; y<bmHeight-n; y++)
 for( x=n; x<bmWidth-n ; x++)
  {
   SumR=SumG=SumB=0;
   for(j=-1;j<=1;j++)
    for(i=-1;i<=1;i++)
      {
        pos=bmWidth*(long)(y+j)+x+i;
        SumR+=GradR[pos];
        SumG+=GradG[pos];
        SumB+=GradB[pos];
      }
   pos =bmWidth*(long)y+x;
   OutR[pos]=SumR/9;
   OutG[pos]=SumG/9;
   OutB[pos]=SumB/9;
  }
*/
```

File name -> bmpdisp.c

```
//merge image RGB through original one
//and anti-vertical flip for standard bmp format
for(  y=tn; y<bmHeight-tn; y++)
 for( x=tn; x<bmWidth-tn ; x++)
  {
    pos = bmWidth*(long)y+x;
    pos1=((bmScanWidth*
          (long)(bmHeight-1-y)+x)*bmBand)+offBits;
    *(lpData+pos1)   =OutB[pos];
    *(lpData+pos1+1)=OutG[pos];
    *(lpData+pos1+2)=OutR[pos];
  }

GlobalFree((HANDLE)GlobalHandle(HIWORD(DataR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataB)));

GlobalFree((HANDLE)GlobalHandle(HIWORD(OutR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutB)));
GlobalUnlock(hDIBInfo);
SetCursor( LoadCursor( NULL, IDC_ARROW ) );
}
/*-------------------------------------*/
BOOL FAR PASCAL ShetAndOdinsDlg(HWND hDlg,
                                WORD msg,
                                WORD wParam,
                                LONG lParam )

{
char      strbuf[10];
float     nl;
int       i,j;
switch(msg)
 {
 case WM_INITDIALOG:
    if(Template>1) Template=1;
    if(Template<0) Template=0;
    CheckRadioButton( hDlg, 2000, 2001,
                      2000+Template);
    for(j=0;j<2;j++)
    {
    sprintf(strbuf,"%3.0f", n[j]);
    SendDlgItemMessage(hDlg, 1000+j ,
                       EM_LIMITTEXT, 10, 0L);
    SetDlgItemText(hDlg, 1000+j , strbuf );
    }
    return(TRUE);
case WM_HSCROLL:
        i = GetWindowWord(HIWORD(lParam),
                          GWW_ID);
        switch(i)
        {
        case 100:j=0;break;
        case 101:j=1;break;
        }

        if(StartScroll)
        {
          GetScrollRange(GetFocus(),SB_CTL,
                         &MinScroll,&MaxScroll);
          Step=(MaxScroll-MinScroll)/255.0;
          StartScroll=FALSE;
          n[0]=n[1]=0;
        }
        switch(wParam)
        {
          case SB_LINEUP:      nl = -1;   break;
          case SB_LINEDOWN:    nl =  1;   break;
          case SB_PAGEUP:      nl = -10;  break;
          case SB_PAGEDOWN:    nl =  10;  break;
          case SB_THUMBPOSITION:
               nl = LOWORD(lParam)-n[0];break;
          default: nl=0;;
        }
        if(nl == max(-n[j],min(nl,255-n[j])))
        {
          n[j] += nl;
          SetScrollPos(GetFocus(),SB_CTL,
                       (int)(n[j]*Step),TRUE);
          sprintf(strbuf,"%3.0f",n[j]);
          SendDlgItemMessage(hDlg, 1000+j,
                             EM_LIMITTEXT,
                             10, 0L );
          SetDlgItemText(hDlg, 1000+j, strbuf );
        }
      return(FALSE);

    case WM_COMMAND:
        switch( wParam )
        {
        case 2000:Template=0;break;
        case 2001:Template=1;break;
        case IDOK:
        case IDCANCEL: EndDialog( hDlg, TRUE );
                       return( TRUE );
        default: return( TRUE );
        }
 }
return( FALSE );
}
```

File name -> bmpdisp.c

```
/*************************************************

IMAGE PROCESSING PROCEDURE:
Image enhancement by Linear contrast Stretching

*************************************************/
BOOL FAR PASCAL LCSDlg( HWND hDlg,
                        WORD msg,
                        WORD wParam,
                        LONG lParam )
{
char    strbuf[10];
float   nl;
int     i,j;
switch(msg)
{
case WM_INITDIALOG:
    for(i=0;i<2;i++)
    {
    sprintf(strbuf,"%f", n[i]);
    SendDlgItemMessage(hDlg, 1000+i ,
                       EM_LIMITTEXT, 10, 0L);
    SetDlgItemText(hDlg, 1000+i , strbuf );
    }
    return(TRUE);

case WM_HSCROLL:
        i = GetWindowWord(HIWORD(lParam),
                          GWW_ID);
        switch(i)
        {
        case 100:j=0;break;
        case 101:j=1;break;
        default:   j=0;break;
        }
        if(StartScroll)
        {
          GetScrollRange(GetFocus(),SB_CTL,
                         &MinScroll,&MaxScroll);
          Step=(MaxScroll-MinScroll)/100;
          StartScroll=FALSE;
          n[0]=0;   n[1]=0;
        }
        switch(wParam)
        {
          case SB_LINEUP:       nl = -0.1;  break;
          case SB_LINEDOWN:     nl =  0.1;  break;
          case SB_PAGEUP:       nl = -1;    break;
          case SB_PAGEDOWN:     nl =  1;    break;
          case SB_THUMBPOSITION:
                nl = LOWORD(lParam)-n[j];  break;
          default: nl=0;;
        }
        if(nl == max(-n[j],min(nl,100-n[j])))
        {
          n[j] += nl;
          SetScrollPos(GetFocus(),SB_CTL,
                       n[j]*Step,TRUE);
          sprintf(strbuf,"%f",n[j]);
          SendDlgItemMessage(hDlg, 1000+j,
                             EM_LIMITTEXT,
                             10, 0L );
          SetDlgItemText(hDlg, 1000+j, strbuf );
        }
      return(FALSE);
 case WM_COMMAND:
        switch( wParam )
        {
        case IDOK:
        case IDCANCEL: EndDialog( hDlg, TRUE );
                       return( TRUE );
        default: return( TRUE );
        }
 }
 return( FALSE );
 }
 /*--------------------------------------*/
 void DoLinearStretch(HWND hWnd)
 {
 LPBITMAPINFOHEADER      lpbi;
 unsigned char           huge *DataR,
                         huge *DataG,
                         huge *DataB;
 unsigned char           huge *OutR ,
                         huge *OutG ,
                         huge *OutB;
 BYTE                    huge *lpData;
 unsigned long           BandSize;
 unsigned long           pos,pos1,NL,NU,sum;
 int                     x,y,buf1,buf2,buf3;
 static BYTE             LTBS[256];
 static unsigned long    HISS[256];
 int                     yMax=255,yMin=0;
 int                     xMaxS,xMinS;
 float                   RatioS,fBuf;

 SetCursor( LoadCursor( NULL, IDC_WAIT ) );
 lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
 lpData=lpbi;

 //saparate band allocated memory size
 BandSize=bmWidth*(long)bmHeight;
```

File name -> bmpdisp.c

```
//input data allocate
DataR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(DataR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataR", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }
DataG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(DataG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataG", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }
DataB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(DataB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the
              DataB", "Error",
              MB_ICONSTOP | MB_OK);
  return FALSE;
 }
//output data allocate
OutR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(OutR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutR",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(OutG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutG",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
OutB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(OutB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the OutB",
              "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
```

```
//split image and flip vertical for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos=bmWidth*(long)y+x;
   pos1=((bmScanWidth*
         (long)(bmHeight-1-y)+x)*bmBand)+offBits;
   DataB[pos]=*(lpData+pos1);
   DataG[pos]=*(lpData+pos1+1);
   DataR[pos]=*(lpData+pos1+2);
  }

//Linear Contrast Stretching
//clear LTB
 for(x=0;x<256;x++)
  {
   LTBS[x]=0;
   HISS[x]=0;
  }

 //find maximum & minimum gray
 /*xMaxS=0;xMinS=255;
 for(y=0;y<bmHeight;y++)
  for(x=0;x<bmWidth;x++)
   {
    pos = bmWidth*(long)y+x;
    xMaxS = max(xMaxS,DataR[pos]);
    xMaxS = max(xMaxS,DataG[pos]);
    xMaxS = max(xMaxS,DataB[pos]);

    xMinS = min(xMinS,DataR[pos]);
    xMinS = min(xMinS,DataG[pos]);
    xMinS = min(xMinS,DataB[pos]);
   }*/

 //make Sum HISTOGRAM
 for(y=0;y<bmHeight;y++)
  for(x=0;x<bmWidth;x++)
   {
    pos = bmWidth*(long)y+x;
    HISS[DataR[pos]]++;
    HISS[DataG[pos]]++;
    HISS[DataB[pos]]++;
   }

 //find amount of pixel for n percent
 NL=(bmWidth*(long)bmHeight*(long)3)*
     (n[0]/100.0);
 NU=(bmWidth*(long)bmHeight*(long)3)*
     (n[1]/100.0);
```

File name -> bmpdisp.c

```
//count n percent to find min and max
sum=0;
for(xMinS=0;xMinS<256;xMinS++)
  {
   sum+=HISS[xMinS];
   if(sum>NL)
     break;
  }
sum=0;
for(xMaxS=255;xMaxS>0;xMaxS--)
  {
   sum+=HISS[xMaxS];
   if(sum>NU)
     break;
  }

//find ratio
RatioS = (yMax-yMin)/(float)(xMaxS-xMinS);

//initial lookup table
for(x=0;x<256;x++)
 {
  LTBS[x]=x;
 }

//make stretching lookup table
for(x=0;x<256;x++)
 {
  fBuf=((LTBS[x]-xMinS)*
       RatioS+yMin)+0.5;//rounding with 0.5
  if(fBuf>255.0) fBuf=255.0;//maximize
  if(fBuf<0)  fBuf=0.0;  //minimize
  LTBS[x]=(BYTE)(int)fBuf;
 }

//transform image
for(y=0;y<bmHeight;y++)
 for(x=0;x<bmWidth;x++)
 {
  pos=bmWidth*(long)y+x;
  OutR[pos]=LTBS[DataR[pos]];
  OutG[pos]=LTBS[DataG[pos]];
  OutB[pos]=LTBS[DataB[pos]];
 }

//merge image RGB through original one
//and anti-vertical flip for standard bmp format
for(  y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos = bmWidth*(long)y+x;
   pos1=((bmScanWidth*
        (long)(bmHeight-1-y)+x)*bmBand)+offBits;
   *(lpData+pos1)   =OutB[pos];
   *(lpData+pos1+1)=OutG[pos];
   *(lpData+pos1+2)=OutR[pos];
  }

GlobalFree((HANDLE)GlobalHandle(HIWORD(DataR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(DataB)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(OutB)));
GlobalUnlock(hDIBInfo);
SetCursor( LoadCursor( NULL, IDC_ARROW ) );
}
```

File name -> bmpdisp.c

```
/*****************************************************

   FUNCTION: About(HWND, unsigned, WORD, LONG)
   PURPOSE:  Processes messages for "About" dialog
             box
   MESSAGES:
        WM_INITDIALOG - initialize dialog box
        WM_COMMAND    - input received

*****************************************************/
BOOL FAR PASCAL __export About(hDlg, message,
                               wParam, lParam)
HWND     hDlg;
unsigned message;
WORD     wParam;
LONG     lParam;
{
switch (message)
 {
  case WM_INITDIALOG:return (TRUE);
  case WM_COMMAND:
        if (wParam == IDOK || wParam == IDCANCEL)
          {
          EndDialog(hDlg, TRUE);
          return (TRUE);
          }
        break;
 }
return (FALSE);
}
/*****************************************************

FUNCTION: InitApplication(HANDLE)
PURPOSE: Initializes window data and registers window
         class

*****************************************************/
BOOL InitApplication(hInstance)
HANDLE          hInstance;
{
WNDCLASS        wc;
wc.style        = NULL;
wc.lpfnWndProc  = SubMain;
wc.cbClsExtra   = 0;
wc.cbWndExtra   = 0;
wc.hInstance    = hInstance;
wc.hIcon        = LoadIcon(hInstance, IDI_ICON1);
wc.hCursor      = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground  = COLOR_WINDOW+1;
wc.lpszMenuName  =  "TRANSFORM";
wc.lpszClassName  ="dibitWClass";
return (RegisterClass(&wc));
}
```

```
/*****************************************************

FUNCTION:  InitInstance(HANDLE, int)
PURPOSE:  Saves instance handle and creates main
          window

*****************************************************/
BOOL InitInstance(hInstance, nCmdShow)
HANDLE          hInstance;
int             nCmdShow;
{
HWND            hWnd;
int             i;
short           loop;
FARPROC         lpfnKeybdProc;
RECT            rc;
hInst = hInstance;
hWnd  = CreateWindow(
                "dibitWClass",
                "Image Processing Application",
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,
                _USEDEFAULT,
                _USEDEFAULT,
                _USEDEFAULT,
                LL,
                LL,
                nstance,
                LL    );
GetWindowRect(hWnd,&rc);
if(!hWnd)
 return (FALSE);
ShowWindow(hWnd, nCmdShow);
UpdateWindow(hWnd);
return (TRUE);
}
/*****************************************************

FUNCTION: SubMain(HWND, UINT, WPARAM, LPARAM)
PURPOSE:  Processes messages
MESSAGES:
 WM_COMMAND- application menu (About dialog box)
 WM_DESTROY  - destroy window

*****************************************************/
long FAR PASCAL __export SubMain(hWnd, message,
                                 wParam, lParam)
HWND            hWnd;
UINT            message;
WPARAM          wParam;
LPARAM          lParam;
{
OPENFILENAME    ofn;
```

File name -> bmpdisp.c

```
FARPROC         lpProcAbout;
FARPROC         lpProc;

switch(message)
 {
 case WM_CREATE:
   hDIBInfo = GlobalAlloc(GMEM_MOVEABLE,
         (DWORD)(sizeof(BITMAPINFOHEADER)+
         256 * sizeof(RGBQUAD)));
   memset(&LogFont,0,sizeof(LOGFONT));
   LogFont.lfHeight=10;
   LogFont.lfWidth=14;
   LogFont.lfWeight=900;
   LogFont.lfUnderline=0;
   LogFont.lfItalic=0;
   lstrcpy(LogFont.lfFaceName,"Swiss");
   hHelv=CreateFontIndirect(&LogFont);
   break;

 case WM_INITMENU:
   CheckMenuItem(wParam, wOperation,
MF_CHECKED);
   break;

 case WM_PAINT:
   if(!bDIBLoaded)//if no DIB loaded , nothing to draw
    return (DefWindowProc(hWnd, message, wParam,
lParam));
   else
    PaintBMP(hWnd);
 break;

 case WM_SIZE://if doing , resize the images
   //GetWindowRect(hWnd,&rc);
   if(wOperation==IDM_STRETCH && bDIBLoaded)
    InvalidateRect(hWnd,NULL,TRUE);
   break;

case WM_COMMAND:
switch(wParam)
{
 case IDM_OPEN:
   ofn.lStructSize    = sizeof(OPENFILENAME);
   ofn.hwndOwner  = hWnd;
   ofn.lpstrFilter    = NULL;
   ofn.lpstrFilter    = "Bitmaps (*.BMP)\0*.BMP\0";
   ofn.lpstrCustomFilter= NULL;
   ofn.nFilterIndex   = 1;
   achFileName[0]  = 0;      /* pass in NULL */
   ofn.lpstrFile      = (LPSTR)achFileName;
   ofn.nMaxFile       = 128;
   ofn.lpstrInitialDir  = NULL;
   ofn.lpstrTitle     = NULL;
   ofn.lpstrFileTitle   = NULL;
   ofn.lpstrDefExt    = NULL;
   ofn.Flags          = 0;
   if(GetOpenFileName((LPOPENFILENAME)&ofn))
    if(InitBMP(hWnd))
     InvalidateRect (hWnd, NULL, TRUE);
   break;

case IDM_SAVE:
   ofn.lStructSize    = sizeof(OPENFILENAME);
   ofn.hwndOwner  = hWnd;
   ofn.lpstrFilter    = NULL;
   ofn.lpstrFilter    = "Bitmaps (*.BMP)\0*.BMP\0";
   ofn.lpstrCustomFilter = NULL;
   ofn.nFilterIndex   = 1;
   achFileName[0]  = 0;
   ofn.lpstrFile      = (LPSTR)achFileName;
   ofn.nMaxFile       = 128;
   ofn.lpstrInitialDir  = NULL;
   ofn.lpstrTitle     = NULL;
   ofn.lpstrFileTitle   = NULL;
   ofn.lpstrDefExt    = NULL;
   ofn.Flags          = 0;
   if(GetOpenFileName((LPOPENFILENAME)&ofn))
    if(SaveBMP(hWnd))
     InvalidateRect (hWnd, NULL, TRUE);
   break;

 case IDM_ABOUT:
   lpProcAbout = MakeProcInstance(About, hInst);
   DialogBox(hInst,
             "AboutBox",
             hWnd,
             lpProcAbout);
   FreeProcInstance(lpProcAbout);
   break;

 case IDM_TODEV:
 case IDM_STRETCH:
 case IDM_SUMHIS:
 case IDM_REDHIS:
 case IDM_GREENHIS:
 case IDM_BLUEHIS:
 case IDM_ALLHIS:
   CheckMenuItem(GetMenu(hWnd), wOperation,
                 MF_UNCHECKED);
   wOperation = wParam;
   CheckMenuItem(GetMenu(hWnd), wOperation,
                 MF_CHECKED);
   InvalidateRect(hWnd, NULL, TRUE);
   break;
```

File name -> bmpdisp.c

```
//image processing procedure
 case IDM_LCS:
   StartScroll=TRUE;
   n[0]=n[1]=0;
   lpProc = MakeProcInstance( LCSDlg, hInst );
   DialogBox( hInst, "LCSBOX", hWnd, lpProc );
   FreeProcInstance( lpProc );
   //Linear Contrast Stretching image enhancement
   DoLinearStretch(hWnd);
   //display result again
   InvalidateRect (hWnd, NULL, TRUE);
   break;

//edge enhance by adding gradient information
 case IDM_EAGI_SO:
   StartScroll=TRUE;//Shetttigara and Odins method
   n[0]=n[1]=0;
   lpProc = MakeProcInstance( ShetAndOdinsDlg, hInst );
   DialogBox( hInst, "SAOBOX", hWnd, lpProc );
   FreeProcInstance( lpProc );
   DoShetAndOdins(hWnd);
   //display result again
   InvalidateRect (hWnd, NULL, TRUE);
   break;

//edge enhance by adding gradient information
 case IDM_EAGI_12T:
   StartScroll=TRUE;
   n[0]=n[1]=0;
   lpProc = MakeProcInstance( T12Dlg, hInst );
   DialogBox( hInst, "T12BOX", hWnd, lpProc );
   FreeProcInstance( lpProc );
   //12 Template method
   Do12Template(hWnd);
   //display result again
   InvalidateRect (hWnd, NULL, TRUE);
   break;

//edge enhancement by smoothing
//used convolution method
 case IDM_ESMT_C:
   lpProc = MakeProcInstance( ESMT_C_Dlg, hInst );
   DialogBox( hInst, "SMTBOX", hWnd, lpProc );
   FreeProcInstance( lpProc );
   DoESMT_C(hWnd);
   //display result again
   InvalidateRect (hWnd, NULL, TRUE);
   break;

 case IDM_ESMT_S://edge enhancement by smoothing
                 //step by step method
   lpProc = MakeProcInstance( ESMT_S_Dlg, hInst );
   DialogBox( hInst, "SMTBOX", hWnd, lpProc );
   FreeProcInstance( lpProc );
   DoESMT_S(hWnd);
   //display result again
   InvalidateRect (hWnd, NULL, TRUE);
   break;

case IDM_ESMT_B://edge enhancement by smoothing
                //step by step method and
                //Spurious edges elimination
                //by Buttrworth function transform
   StartScroll=TRUE;
   OrderCutoff[0]=OrderCutoff[1]=0;
   Style=0;
   lpProc = MakeProcInstance( BTWDlg, hInst );
   DialogBox( hInst, "BTWBOX", hWnd, lpProc );
   FreeProcInstance( lpProc );
   DoESMT_B(hWnd);
   //display result again
   InvalidateRect (hWnd, NULL, TRUE);
   break;

 //case IDM_UNDO://for copy original image back
// to active picture
   // DoUndo();
   //InvalidateRect (hWnd, NULL, TRUE);
   //display result again
   // break;
 }
 break;

 //message: window being destroyed
  case WM_DESTROY:
//free mem allocate for Undo procedure
   //FreeUndo();
   DeleteObject(hHelv);
   DeleteObject(hDDBitmap);
   PostQuitMessage(0);
 break;

 default:
    return (DefWindowProc(hWnd, message, wParam,
                          lParam));
 }
return (NULL);
}
```

File name -> bmpdisp.c

```
/*******************************************************

FUNCTION: WinMain(HANDLE, HANDLE, LPSTR, int)
PURPOSE: calls initialization function, processes message
         loop

*******************************************************/
int PASCAL WinMain(hInstance, hPrevInstance,
                   lpCmdLine,nCmdShow)
HANDLE  hInstance;
HANDLE  hPrevInstance;
LPSTR   lpCmdLine;
int     nCmdShow;
{
MSG     msg;
if(!hPrevInstance)
 if(!InitApplication(hInstance))
  return (FALSE);
if(!InitInstance(hInstance, nCmdShow))
 return (FALSE);
while (GetMessage(&msg,    NULL, NULL, NULL))
{
  TranslateMessage(&msg);
  DispatchMessage(&msg);
}
return (msg.wParam);
}
□
```

File name -> DIBS16k.c

```
#include "math.h"

PreUndo(HWND hWnd)
{
unsigned long           BandSize,pos,pos1;
int                     x,y;
LPBITMAPINFOHEADER      lpbi;
BYTE                    huge *lpData;

SetCursor( LoadCursor( NULL, IDC_WAIT ) );
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
lpData=lpbi;
BandSize=bmWidth*(long)bmHeight;
//Undo data allocate
UndoR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(UndoR==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the DataR",
             "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
UndoG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(UndoG==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the DataG",
             "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }
UndoB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(UndoB==NULL)
 {
  MessageBox(hWnd, "Can't create Mem of the DataB",
             "Error", MB_ICONSTOP | MB_OK);
  return FALSE;
 }

//split image and flip vertical for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos=bmWidth*(long)y+x;
   pos1=((bmScanWidth*
         (long)(bmHeight-1-y)+x)*bmBand)+offBits;
   UndoB[pos]=*(lpData+pos1);
   UndoG[pos]=*(lpData+pos1+1);
   UndoR[pos]=*(lpData+pos1+2);
  }
GlobalUnlock(hDIBInfo);
SetCursor( LoadCursor( NULL, IDC_ARROW ) );
}

FreeUndo()
{
GlobalFree((HANDLE)GlobalHandle(HIWORD(UndoR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(UndoG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(UndoB )));
}

DoUndo()
{
unsigned long           BandSize,pos,pos1;
int                     x,y;
LPBITMAPINFOHEADER      lpbi;
BYTE                    huge *lpData;
SetCursor( LoadCursor( NULL, IDC_WAIT ) );
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
lpData=lpbi;
//merge image RGB through original one
//and anti-vertical flip for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
   pos = bmWidth*(long)y+x;
   pos1=((bmScanWidth*
         (long)(bmHeight-1-y)+x)*bmBand)+offBits;
   *(lpData+pos1)   =UndoB[pos];
   *(lpData+pos1+1)=UndoG[pos];
   *(lpData+pos1+2)=UndoR[pos];
  }
GlobalUnlock(hDIBInfo);
SetCursor( LoadCursor( NULL, IDC_ARROW ) );
}
/*************************************************/
/** PRIVATE ROUTINE TO WRITE MORE THAN 64K **
* FUNCTION   : lwrite(int fh, VOID FAR *pv, DWORD ul)
* PURPOSE    : Writes data in steps of 32k till all the
               data has been write.
*************************************************/
DWORD PASCAL lwrite (int fh, unsigned char huge *pv,
                     DWORD ul)
{
DWORD    ulT = ul;
BYTE huge *hp = pv;
  while (ul > (DWORD)MAXREAD)
  {
        if (_lwrite(fh, (LPSTR)hp,
                (WORD)MAXREAD) != MAXREAD)
                return 0;
        ul -= MAXREAD;   hp += MAXREAD;
  }
  if (_lwrite(fh, (LPSTR)hp, (WORD)ul) != (WORD)ul)
        return 0;
  return ulT;
}
```

File name -> DIBS16k.c

```
#include 'math.h'

PreUndo(HWND hWnd)
{
unsigned long           BandSize,pos,pos1;
int                     x,y;
LPBITMAPINFOHEADER      lpbi;
BYTE                    huge *lpData;

SetCursor( LoadCursor( NULL, IDC_WAIT ) );
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
lpData=lpbi;
BandSize=bmWidth*(long)bmHeight;
//Undo data allocate
UndoR = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(UndoR==NULL)
  {
   MessageBox(hWnd, 'Can't create Mem of the DataR',
              'Error', MB_ICONSTOP | MB_OK);
   return FALSE;
  }
UndoG = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(UndoG==NULL)
  {
   MessageBox(hWnd, 'Can't create Mem of the DataG',
              'Error', MB_ICONSTOP | MB_OK);
   return FALSE;
  }
UndoB = (BYTE huge*)GlobalLock(GlobalAlloc( GHND,
                                    BandSize) );
if(UndoB==NULL)
  {
   MessageBox(hWnd, 'Can't create Mem of the DataB',
              'Error', MB_ICONSTOP | MB_OK);
   return FALSE;
  }

//split image and flip vertical for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
    pos=bmWidth*(long)y+x;
    pos1=((bmScanWidth*
          (long)(bmHeight-1-y)+x)*bmBand)+offBits;
    UndoB[pos]=*(lpData+pos1);
    UndoG[pos]=*(lpData+pos1+1);
    UndoR[pos]=*(lpData+pos1+2);
  }
 GlobalUnlock(hDIBInfo);
 SetCursor( LoadCursor( NULL, IDC_ARROW ) );
}

FreeUndo()
{
GlobalFree((HANDLE)GlobalHandle(HIWORD(UndoR)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(UndoG)));
GlobalFree((HANDLE)GlobalHandle(HIWORD(UndoB )));
}

DoUndo()
{
unsigned long           BandSize,pos,pos1;
int                     x,y;
LPBITMAPINFOHEADER      lpbi;
BYTE                    huge *lpData;
SetCursor( LoadCursor( NULL, IDC_WAIT ) );
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
lpData=lpbi;
//merge image RGB through original one
//and anti-vertical flip for standard bmp format
for( y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
  {
    pos = bmWidth*(long)y+x;
    pos1=((bmScanWidth*
          (long)(bmHeight-1-y)+x)*bmBand)+offBits;
    *(lpData+pos1)   =UndoB[pos];
    *(lpData+pos1+1)=UndoG[pos];
    *(lpData+pos1+2)=UndoR[pos];
  }
GlobalUnlock(hDIBInfo);
SetCursor( LoadCursor( NULL, IDC_ARROW ) );
}
/*************************************************/
/** PRIVATE ROUTINE TO WRITE MORE THAN 64K **
*  FUNCTION   : lwrite(int fh, VOID FAR *pv, DWORD ul)
*  PURPOSE    : Writes data in steps of 32k till all the
                data has been write.
*************************************************/
DWORD PASCAL lwrite (int fh, unsigned char huge *pv,
                     DWORD ul)
{
DWORD    ulT = ul;
BYTE huge *hp = pv;
   while (ul > (DWORD)MAXREAD)
   {
        if (_lwrite(fh, (LPSTR)hp,
                (WORD)MAXREAD) != MAXREAD)
                return 0;
        ul -= MAXREAD;    hp += MAXREAD;
   }
   if (_lwrite(fh, (LPSTR)hp, (WORD)ul) != (WORD)ul)
        return 0;
   return ulT;
}
```

File name -> DIBS16k.c

```
/*****************************************************
 * FUNCTION   : SaveBMP
 * PURPOSE    : Saves data *.bmp in all size.
*****************************************************/
int SaveBMP(HWND hWnd)
{
unsigned int                fh;
LPBITMAPINFOHEADER          lpbi;
OFSTRUCT                    of;
char                        str[255];
WORD                        result = FALSE;
BYTE huge                   *lpData;

// Open the file and get a handle to it's BITMAPINFO
fh = OpenFile (achFileName, &of, OF_CREATE);
if(fh == -1){
  wsprintf(str,"Can't saving file '%ls'",
                    (LPSTR)achFileName);
  MessageBox(hWnd, str, "Error",
                    MB_ICONSTOP | MB_OK);
  return (FALSE);
}

//write file header
_lwrite (fh, (LPSTR)&bf, sizeof(BITMAPFILEHEADER));

//pointer for all data
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);

//write file info header
_lwrite (fh, (LPSTR)lpbi, sizeof(BITMAPINFO));

_llseek(fh,bf.bfOffBits,SEEK_SET);

lwrite(fh, (LPSTR)lpbi+offBits, lpbi->biSizeImage);

ErrExit:
   _lclose(fh);
   GlobalUnlock(hDIBInfo);
ErrExit2:
   return(result);
}

/*****************************************************/
/** PRIVATE ROUTINE TO READ MORE THAN 64K **
 * FUNCTION   : lread(int fh, VOID FAR *pv, DWORD ul)
 * PURPOSE    : Reads data in steps of 32k till all the
                data has been read.
*****************************************************/
DWORD PASCAL lread (int fh, unsigned char huge *pv,
DWORD ul)
{
   DWORD    ulT = ul;
   BYTE huge *hp = pv;
   while (ul > (DWORD)MAXREAD) {
        if (_lread(fh, (LPSTR)hp,
                (WORD)MAXREAD) != MAXREAD)
                return 0;
        ul -= MAXREAD;   hp += MAXREAD;
   }
   if (_lread(fh, (LPSTR)hp, (WORD)ul) != (WORD)ul)
        return 0;
   return ulT;
}

/*****************************************************
 * FUNCTION   : ReadBMP
 * PURPOSE    : Reads data *.bmp in all sizes.
*****************************************************/
int ReadBMP(HWND hWnd)
{
unsigned int                fh;
LPBITMAPINFOHEADER          lpbi;
OFSTRUCT                    of;
char                        str[255];
WORD                        result = FALSE;
BYTE huge                   *lpData;

// Open the file and get a handle to it's BITMAPINFO
fh = OpenFile (achFileName, &of, OF_READ);
if(fh == -1){
  wsprintf(str,"Can't open file '%ls'",
(LPSTR)achFileName);
  MessageBox(hWnd, str, "Error", MB_ICONSTOP |
MB_OK);
  return (FALSE);
}

//pointer for all data
lpbi=(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);

// read the BITMAPFILEHEADER
if (sizeof (bf) != _lread (fh, (LPSTR)&bf, sizeof (bf)))
  goto ErrExit;

//check bmp type
if (bf.bfType != 0x4d42)  /* 'BM' */
  goto ErrExit;

//read all header details
if (sizeof(BITMAPINFOHEADER) != _lread (fh,
(LPSTR)lpbi, sizeof(BITMAPINFOHEADER)))
  goto ErrExit;
```

File name -> DIBS16k.c

```
// !!!!! for now, don't even deal with CORE headers
if (lpbi->biSize == sizeof(BITMAPCOREHEADER))
   goto ErrExit;

//check bmp file is true color
 if(lpbi->biBitCount != 24)//does not 24 bit true color ,
exit
   goto ErrExit;

//must be an even WORD size
bmScanWidth= lpbi->biWidth;
while(bmScanWidth%sizeof(WORD))
   bmScanWidth++;

bmBand = 3;//24 bits true color
//check size of sizeimage and fill in some default values if
they are zero
lpbi->biSizeImage =
bmScanWidth*(long)bmBand*(long)lpbi->biHeight;

// get a proper-sized buffer for header, color table and
bits
GlobalUnlock(hDIBInfo);
hDIBInfo = GlobalReAlloc(hDIBInfo, lpbi->biSize +
                                   lpbi-
>biSizeImage, 0);
if (!hDIBInfo)        /* can't resize buffer for loading */
   goto ErrExit2;

//global definition variable
lpbi      =
(LPBITMAPINFOHEADER)GlobalLock(hDIBInfo);
bmWidth   = lpbi->biWidth;
bmHeight  = lpbi->biHeight;
bmImgSize = lpbi->biSizeImage;

// offset to the bits from start of DIB header
offBits = (WORD)lpbi->biSize;
if(bf.bfOffBits != 0L)
  _llseek(fh,bf.bfOffBits,SEEK_SET);

//check complete of body bmp read y/n
if(lpbi->biSizeImage == lread(fh, (LPSTR)lpbi + offBits,
lpbi->biSizeImage))
   result = TRUE;

ErrExit:
   _lclose(fh);
   GlobalUnlock(hDIBInfo);
ErrExit2:
   return(result);
}
```

```
/*****************************************************
* FUNCTION   : InitBMP
* PURPOSE    : Initial for Reads data *.bmp and open
               window
*****************************************************/
int InitBMP(HWND hWnd)
{
LPBITMAPINFOHEADER  lpbi;
RECT                Rectangle;
int                 bufx,bufy;
/* if there was an old DIB, free it up */
if(bDIBLoaded)
{
  //clear handle bitmap
  if(hDDBitmap)
  {
    //FreeUndo();//for free mem of Undo procedure
    SelectObject(hMemDC, hOldBitmap);
    DeleteDC(hMemDC);
    DeleteObject(hDDBitmap);
    hDDBitmap = NULL;
  }
  //clear load flag
  bDIBLoaded = FALSE;
}

// load the BMP IMAGE from the file
if(!ReadBMP(hWnd))
{
  MessageBox(hWnd, "Error attempting to read DIB",
             "Error", MB_ICONSTOP | MB_OK);
  return(FALSE);
}

//open window for bmp
bDIBLoaded = TRUE; // there is a DIB loaded now
SetWindowText(hWnd, achFileName);
Rectangle.left      = 0;
Rectangle.top       = 0;
Rectangle.right  = (bmWidth<256)?
                   256:bmWidth;//(WORD)bufx;
Rectangle.bottom = (bmHeight<256)?
                   256:(bmWidth+20);//(WORD)bufy;
GlobalUnlock(hDIBInfo);
/* Compute the size of the window rectangle
*  based on the given client rectangle size
*  and the window style, then size the window.
*  Do not deal with possibility of more than one menu line.
*/

AdjustWindowRect (&Rectangle,
WS_OVERLAPPEDWINDOW, TRUE);
```

File name -> DIBS16k.c

```
SetWindowPos (hWnd, (HWND)NULL, 0, 0,
              Rectangle.right  - Rectangle.left,
              Rectangle.bottom - Rectangle.top + 1,
              SWP_NOMOVE | SWP_NOZORDER);
GetClientRect(hWnd, &Rectangle);
//PreUndo(hWnd);//allocate for Undo procedure
return(TRUE);
}

/*************************************************
* FUNCTION   : PaintBMP
* PURPOSE    : display BMP file
*************************************************/
int PASCAL NEAR PaintBMP(HWND hWnd)
{
BYTE huge               *lpData;
PAINTSTRUCT             ps;
RECT                    Rect;
LPBITMAPINFOHEADER      lpInfo;
LPBITMAPINFOHEADER      lpHeader;

//histogram var
unsigned char     huge *DataR,
                  huge *DataG,
                  huge *DataB;
int               x,y,b;
static LOGPEN     lpBlue  = {PS_SOLID, 1, 1,
                             RGB(0,   0, 255)},
                  lpGreen = {PS_SOLID, 1, 1,
                             RGB(0, 255,   0)},
                  lpRed   = {PS_SOLID, 1, 1,
                             RGB(255, 0,   0)},
                  lpBlack = {PS_SOLID, 1, 1,
                             RGB(0,   0,   0)};
HPEN                        hPen[4];
unsigned long               Max;
int                         Height,tem;
float                       Ratio;
HBRUSH                      hBrush;
unsigned long               His[3][256];
unsigned long               pos,pos1;
int                         hleft=25,htop=20;
static char                 strbuf[20];
unsigned long               BandSize;

//start paint mode
hDC      = BeginPaint(hWnd, &ps);
SelectObject(hDC,hHelv);
lpInfo = (LPBITMAPINFOHEADER) GlobalLock(hDIBInfo);
lpData   = lpInfo;
lpHeader = lpInfo;

switch (wOperation)
{
case IDM_TODEV:
   SetDIBitsToDevice(hDC,0,0,(WORD)lpInfo->biWidth,
                     (WORD)lpInfo->biHeight,0,0,0,
                     (WORD)lpInfo->biHeight,
                     (LPSTR)lpInfo + offBits,
                     (LPBITMAPINFO)lpHeader,
                     DIB_RGB_COLORS);
   break;

case IDM_STRETCH:
   /* get dimensions of window and stretch to fit */
   GetClientRect(hWnd, (LPRECT)&Rect);
   StretchDIBits(hDC,0,0,Rect.right,Rect.bottom,0,0,
                     (WORD)lpInfo->biWidth,
                     (WORD)lpInfo->biHeight,
                     (LPSTR)lpInfo + offBits,
                     (LPBITMAPINFO)lpHeader,
                     DIB_RGB_COLORS,SRCCOPY);
   /* NOTE: because driver does not do the StretchDIB
   **       itself, this is not a fast operation.
   **       internally it converts to CreateDIBitmap
   **       followed by a StretchBlt */
   break;

case IDM_SUMHIS:
case IDM_REDHIS:
case IDM_GREENHIS:
case IDM_BLUEHIS:
case IDM_ALLHIS:
   //saparate band allocated memory size
   BandSize=bmWidth*(long)bmHeight;

   //input data allocate
   DataR = (BYTE huge*)GlobalLock(GlobalAlloc
                        ( GHND, BandSize) );
   if(DataR==NULL)
      {
      MessageBox(hWnd, "Can't create Mem of the
                        DataR", "Error",
                        MB_ICONSTOP | MB_OK);
      return FALSE;
      }
   DataG = (BYTE huge*)GlobalLock(GlobalAlloc
                        ( GHND, BandSize) );
   if(DataG==NULL)
      {
      MessageBox(hWnd, "Can't create Mem of the
                        DataG", "Error",
                        MB_ICONSTOP | MB_OK);
      return FALSE;
      }
```

File name -> DIBS16k.c

```
DataB = (BYTE huge*)GlobalLock(GlobalAlloc
                        ( GHND, BandSize) );
if(DataB==NULL)
     {
      MessageBox(hWnd, "Can't create Mem of the
                  DataB", "Error",
                  MB_ICONSTOP | MB_OK);
      return FALSE;
     }

//split image and flip vertical for standard bmp format
for(  y=0; y<bmHeight; y++)
 for( x=0; x<bmWidth ; x++)
      {
       pos=bmWidth*(long)y+x;
       pos1=((bmScanWidth*(long)
             (bmHeight-1-y)+x)*
              bmBand)+offBits;
       DataB[pos]=*(lpData+pos1);
       DataG[pos]=*(lpData+pos1+1);
       DataR[pos]=*(lpData+pos1+2);
      }

/*plot histogram*/
hPen[3] = CreatePenIndirect(&lpBlack);
hPen[2] = CreatePenIndirect(&lpBlue);
hPen[1] = CreatePenIndirect(&lpGreen);
hPen[0] = CreatePenIndirect(&lpRed);
hBrush  = GetStockObject(HOLLOW_BRUSH);

SelectObject(hDC,hPen[3]);
Rectangle(hDC,0,0,300,160);
TextOut(hDC,(hleft+256)/2,htop+125,
                  "Gray Scale",10);
Rectangle(hDC,hleft,htop,hleft+256,htop+100);

//label y axis
for(y=100;y>=0;y--)
{
      if(y%20==0)
       {
       MoveTo(hDC,hleft-5,htop+y);
       LineTo(hDC,hleft  ,htop+y);
       sprintf(strbuf,"%d%%",100-y);
       TextOut(hDC,hleft-22,htop+y,
                        strbuf,strlen(strbuf));
       }
      else if(y%5==0)
       {
       MoveTo(hDC,hleft-2,htop+y);
       LineTo(hDC,hleft  ,htop+y);
       }
}

//label x axis
for(x=0;x<256;x++)
{
      if(x%20==0)
       {
       MoveTo(hDC,hleft+x,htop+100);
       LineTo(hDC,hleft+x,htop+108);
       sprintf(strbuf,"%d",x);
       TextOut(hDC,x+hleft,htop+110,
              strbuf,strlen(strbuf));
       }
      else if(x%5==0)
       {
       MoveTo(hDC,hleft+x,htop+100);
       LineTo(hDC,hleft+x,htop+105);
       }
}

for(y=0;y<256;y++)
 for(x=0;x<bmBand;x++)
      His[x][y]=0; //clear histogram

for(y=0;y<bmHeight;y++)
 for(x=0;x<bmWidth;x++)
  {//s-xy
   pos=bmWidth*(long)y+x;
   switch(wOperation)
   {//s-sw
    case IDM_REDHIS:
    case IDM_GREENHIS:
    case IDM_BLUEHIS:
    case IDM_ALLHIS:
          His[0][DataR[pos]]++;
          His[1][DataG[pos]]++;
          His[2][DataB[pos]]++;
          break;
    case IDM_SUMHIS:
          His[0][DataR[pos]]++;
          His[0][DataG[pos]]++;
          His[0][DataB[pos]]++;
    break;
   }//e-sw
  }//e-xy

//find max of all band
switch(wOperation)
{//s-sw
 case IDM_REDHIS:
 case IDM_GREENHIS:
 case IDM_BLUEHIS:
 case IDM_ALLHIS:
       Max=0;
```

File name -> DIBS16k.c

```
      for(y=0;y<256;y++)
     {
          if(His[0][y]>Max) Max=His[0][y];
          if(His[1][y]>Max) Max=His[1][y];
          if(His[2][y]>Max) Max=His[2][y];
     }
      break;
 case IDM_SUMHIS:
        Max=0;
        for(y=0;y<256;y++)
          if(His[0][y]>Max) Max=His[0][y];
        break;
}//e-sw

Height=100;
Ratio=(float)Height/Max;
sprintf(strbuf,"Frequency (Max=%ld)",Max);
TextOut(hDC,hleft-22,5,strbuf,strlen(strbuf));

//plot his
switch(wOperation)
{//s-sw
 case IDM_REDHIS:
        SelectObject(hDC,hPen[0]);
        for(y=0;y<256;y++)
        {
          MoveTo(hDC,hleft+y,
               htop+Height-His[0][y]*Ratio);
          LineTo(hDC,hleft+y,htop+100);
        }
    break;
 case IDM_GREENHIS:
        SelectObject(hDC,hPen[1]);
        for(y=0;y<256;y++)
        {
          MoveTo(hDC,hleft+y,
               htop+Height-His[1][y]*Ratio);
          LineTo(hDC,hleft+y,htop+100);
        }
    break;
 case IDM_BLUEHIS:
        SelectObject(hDC,hPen[2]);
        for(y=0;y<256;y++)
        {
          MoveTo(hDC,hleft+y,
               htop+Height-His[2][y]*Ratio);
          LineTo(hDC,hleft+y,htop+100);
        }
    break;
 case IDM_ALLHIS:
        for(y=0;y<256;y++)
         for(x=0;x<3;x++)
         {
          SelectObject(hDC,hPen[x]);
          MoveTo(hDC,hleft+y,
               htop+Height-His[x][y]*Ratio);
          LineTo(hDC,hleft+y,htop+100);
         }
        break;
 case IDM_SUMHIS:
        SelectObject(hDC,hPen[3]);
        for(y=0;y<256;y++)
         {
          MoveTo(hDC,hleft+y,
               htop+Height-His[0][y]*Ratio);
          LineTo(hDC,hleft+y,htop+100);
         }
        break;
     }//e-sw
 GlobalFree((HANDLE)GlobalHandle(HIWORD(DataR)));
 GlobalFree((HANDLE)GlobalHandle(HIWORD(DataG)));
 GlobalFree((HANDLE)GlobalHandle(HIWORD(DataB)));
 break;
 }//end switch

 ExitTime:
    GlobalUnlock(hDIBInfo);
    DeleteObject(hPen[0]);
    DeleteObject(hPen[1]);
    DeleteObject(hPen[2]);
    EndPaint(hWnd, &ps);
    return TRUE;
 }
```

File name -> bmpdisp.rc

```
//Microsoft App Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////
////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#define APSTUDIO_HIDDEN_SYMBOLS
#include "windows.h"
#undef APSTUDIO_HIDDEN_SYMBOLS

////////////////////////////////////////////////
////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS


////////////////////////////////////////////////
////////////////////////////////////////////////
//
// Menu
//

TRANSFORM MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open",
IDM_OPEN
        MENUITEM "&Save", IDM_SAVE
        MENUITEM "&Exit",   IDM_EXIT
    END
    MENUITEM "&Undo",     IDM_UNDO
    POPUP "&Display-mode"
    BEGIN
        MENUITEM "&Normal size image",
IDM_TODEV
        MENUITEM "&Full size image",
IDM_STRETCH
        MENUITEM "&Sum Histogram",
IDM_SUMHIS
        MENUITEM "&Red Histogram",
IDM_REDHIS
        MENUITEM "&Green Histogram",
IDM_GREENHIS
        MENUITEM "&Blue Histogram",
IDM_BLUEHIS
        MENUITEM "&All Histogram",
IDM_ALLHIS
    END
    POPUP "&Process"
    BEGIN
        MENUITEM "&LCS", IDM_LCS
        MENUITEM "EAGI by Shet and &Odins",
                 IDM_EAGI_SO
        MENUITEM "EAGI by 12&Template",
IDM_EAGI_12T
        MENUITEM "ESMT by &Step",
IDM_ESMT_S
        MENUITEM "ESMT by &Convolution",
IDM_ESMT_C
        MENUITEM "ESMT by &Butterworth",
IDM_ESMT_B
    END
    MENUITEM "&About",
IDM_ABOUT
END

#ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////
////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define APSTUDIO_HIDDEN_SYMBOLS\r\n"
    "#include ""windows.h""\r\n"
    "#undef APSTUDIO_HIDDEN_SYMBOLS\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "\r\n"
    "\0"
END

////////////////////////////////////////////////
////////////////////////////////////////////////
#endif    // APSTUDIO_INVOKED


////////////////////////////////////////////////
////////////////////////////////////////////////
//
// Icon
//

IDI_ICON1           ICON    DISCARDABLE
"ICON1.ICO"
```

File name -> bmpdisp.rc

```
//////////////////////////////////////////////
//////////////////////////////////////////////
//
// Dialog
//
ABOUTBOX DIALOG DISCARDABLE  22, 17, 144,
75
STYLE DS_MODALFRAME | WS_CAPTION |
WS_SYSMENU
CAPTION "About "
FONT 8, "System"
BEGIN
    CTEXT           "Microsoft Windows",-1,0,6,144,8
    CTEXT           "Image Processing Application",-
1,0,17,144,8
    CTEXT           "Version 1.0 (Testing)",-
1,0,42,144,8
    DEFPUSHBUTTON
"OK",IDOK,40,54,65,17,WS_GROUP
    CTEXT           "by Armote Somboonkaew",-
1,0,29,144,8
END

LCSBOX DIALOG DISCARDABLE  0, 0, 229, 155
STYLE DS_MODALFRAME | WS_POPUP |
WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Linear Contrast Stretching"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,25,115,81,25
    PUSHBUTTON-
"Cancel",IDCANCEL,119,115,84,25
    EDITTEXT
1000,169,34,22,16,ES_AUTOHSCROLL |
WS_DISABLED
    GROUPBOX        "Lower Stretching
Percent",IDC_STATIC,30,19,169,39,
                WS_DISABLED
    SCROLLBAR       100,39,38,115,11,WS_GROUP
| WS_TABSTOP
    EDITTEXT
1001,169,82,22,16,ES_AUTOHSCROLL |
WS_DISABLED
    GROUPBOX        "Upper Stretching
Percent",IDC_STATIC,30,67,169,39,
                WS_DISABLED
    SCROLLBAR       101,40,85,115,11,WS_GROUP
| WS_TABSTOP
    CTEXT           "Please select stretching
percent",IDC_STATIC,0,7,228,11
END

SMTBOX DIALOG DISCARDABLE  0, 0, 185, 94
STYLE DS_MODALFRAME | WS_POPUP |
WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Smooth Templete"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,37,75,111,14
    CONTROL         "templete
1/9",3000,"Button",BS_AUTORADIOBUTTON |
                WS_TABSTOP,34,60,60,10
    CONTROL         "templete
1/16",3001,"Button",BS_AUTORADIOBUTTON |
                WS_TABSTOP,100,60,60,10
    EDITTEXT
2000,99,15,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2001,114,15,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2002,129,15,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2003,99,30,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2004,114,30,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2005,129,30,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2006,99,45,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2007,114,45,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2008,129,45,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1000,43,15,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1001,58,15,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1002,73,15,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1003,43,30,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1004,58,30,12,11,ES_AUTOHSCROLL |
WS_DISABLED
```

File name -> bmpdisp.rc

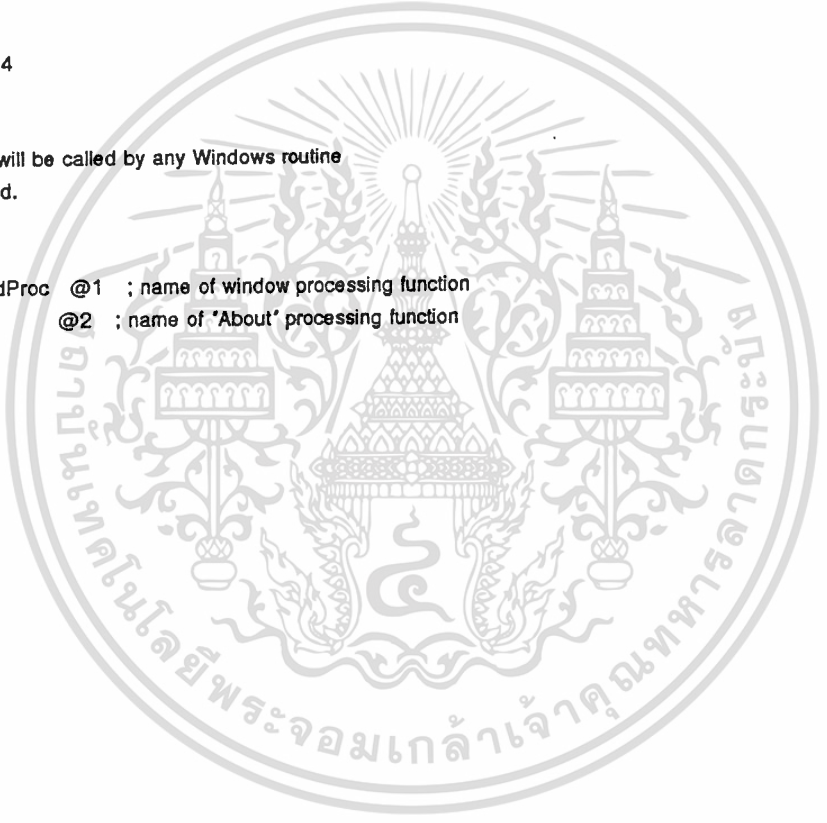
```
    EDITTEXT
1005,73,30,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1006,43,45,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1007,58,45,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1008,73,45,12,11,ES_AUTOHSCROLL |
WS_DISABLED
END
BTWBOX DIALOG DISCARDABLE  0, 0, 339, 182
STYLE DS_MODALFRAME | WS_POPUP |
WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Edge Enhancement using Transform of
BTW"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,11,149,169,27
    PUSHBUTTON
"Cancel",IDCANCEL,184,149,141,27
    EDITTEXT
4000,146,81,29,16,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
4001,146,118,30,16,ES_AUTOHSCROLL |
WS_DISABLED
    GROUPBOX
"Order",IDC_STATIC,11,72,169,31,WS_DISABLED
    GROUPBOX
"Cutoff",IDC_STATIC,11,108,170,33,WS_DISABLED
    SCROLLBAR       101,22,85,115,11,WS_GROUP
| WS_TABSTOP
    SCROLLBAR
102,22,122,115,11,WS_GROUP | WS_TABSTOP
    CONTROL         "templete
1/9",3000,"Button",BS_AUTORADIOBUTTON |
                WS_TABSTOP,34,55,60,10
    CONTROL         "templete
1/16",3001,"Button",BS_AUTORADIOBUTTON |
                WS_TABSTOP,101,55,60,10
    EDITTEXT
2000,99,10,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2001,114,10,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2002,130,10,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2003,99,26,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2004,114,26,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2005,130,26,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2006,99,41,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2007,114,41,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
2008,130,41,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1000,43,10,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1001,58,10,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1002,74,10,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1003,43,26,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1004,58,26,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1005,74,26,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1006,43,41,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1007,58,41,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    EDITTEXT
1008,74,41,12,11,ES_AUTOHSCROLL |
WS_DISABLED
    CONTROL         "Low frequency image
(Smooth)",5000,"Button",
                BS_AUTORADIOBUTTON |
WS_TABSTOP,193,24,114,9
    CONTROL         "High frequency image
(Edge)",5001,"Button",
                BS_AUTORADIOBUTTON |
WS_TABSTOP,193,55,127,10
    GROUPBOX        "Consideration
Result",IDC_STATIC,184,6,141,135,
                WS_DISABLED
    CONTROL         "Edge when pass IBTW
transform",5002,"Button",
                BS_AUTORADIOBUTTON |
WS_TABSTOP,193,85,127,10
```

File name -> bmpdisp.rc

```
    CONTROL         "Final Edge Adding with
Original",5003,"Button",
                    BS_AUTORADIOBUTTON |
WS_TABSTOP,193,116,127,10
END


T12BOX DIALOG DISCARDABLE  0, 0, 229, 169
STYLE DS_MODALFRAME | WS_POPUP |
WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Edge enhancement by adding 12 template
Gradient"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,26,133,81,25
    PUSHBUTTON
"Cancel",IDCANCEL,119,133,84,25
    EDITTEXT
1000,169,98,22,16,ES_AUTOHSCROLL |
WS_DISABLED
    GROUPBOX        "Edge Lowest
Cutoff",IDC_STATIC,30,83,169,37,WS_DISABLED
    SCROLLBAR       100,38,98,115,16,WS_GROUP
| WS_TABSTOP
    CONTROL         "Pure
Edge",2000,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,51,32,48,9
    CONTROL         "Final Edge Adding with
Original",2001,"Button",
                    BS_AUTORADIOBUTTON |
WS_TABSTOP,51,46,127,10
    GROUPBOX        "Consideration
Result",IDC_STATIC,44,12,141,58,
                    WS_DISABLED
END

SAOBOX DIALOG DISCARDABLE  0, 0, 229, 191
STYLE DS_MODALFRAME | WS_POPUP |
WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Edge enhancement by Shettigara and
Odins"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,26,159,81,25
    PUSHBUTTON
"Cancel",IDCANCEL,119,159,84,25
    EDITTEXT
1000,169,82,22,16,ES_AUTOHSCROLL |
WS_DISABLED
    GROUPBOX        "Number of Half Template
(n)",IDC_STATIC,30,70,169,32,
                    WS_DISABLED
    SCROLLBAR       100,38,83,115,15,WS_GROUP
| WS_TABSTOP
    CONTROL         "Pure
Edge",2000,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,51,24,48,9
    CONTROL         "Final Edge Adding with
Original",2001,"Button",
                    BS_AUTORADIOBUTTON |
WS_TABSTOP,51,38,127,10
    GROUPBOX        "Consideration
Result",IDC_STATIC,44,4,141,58,
                    WS_DISABLED
    EDITTEXT
1001,169,124,22,16,ES_AUTOHSCROLL |
WS_DISABLED
    GROUPBOX        "Edge Lowest
Cutoff",IDC_STATIC,30,112,169,32,
                    WS_DISABLED
    SCROLLBAR
101,38,125,115,15,WS_GROUP | WS_TABSTOP
END


#ifndef APSTUDIO_INVOKED
////////////////////////////////////////////////
////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
////////////////////////////////////////////////
////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

File name -> bmpdisp.def

```
; module-definition file for generic -- used by LINK.EXE

NAME         dibit     ; application's module name

DESCRIPTION  'Manipulates Device Independent Bitmap'

EXETYPE      WINDOWS         ; required for all Windows applications

STUB         'WINSTUB.EXE' ; Generates error message if application
                           ; is run without Windows

;CODE can be moved in memory and discarded/reloaded
CODE  PRELOAD MOVEABLE DISCARDABLE

;DATA must be MULTIPLE if program can be invoked more than once
DATA  PRELOAD MOVEABLE MULTIPLE


HEAPSIZE     1024


; All functions that will be called by any Windows routine
; MUST be exported.

EXPORTS
        MainWndProc   @1   ; name of window processing function
        About         @2   ; name of 'About' processing function
```



File name -> bmpdisp.h

```
#include "resource.h"

/* flags for _lseek */
#define  SEEK_CUR 1
#define  SEEK_END 2
#define  SEEK_SET 0

#define MAXREAD  32768                  /* Number of bytes to be read during */
                                        /* each read operation.              */
int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
BOOL InitApplication(HANDLE);
BOOL InitInstance(HANDLE, int);
long FAR PASCAL __export SubMain(HWND, UINT, WPARAM, LPARAM);
BOOL FAR PASCAL __export About(HWND, unsigned, WORD, LONG);

/* OpenFile functions */
HANDLE FAR PASCAL OpenDlg(HWND, unsigned, WORD, LONG);
void SeparateFile(HWND, LPSTR, LPSTR, LPSTR);
void UpdateListBox(HWND);
void AddExt(PSTR, PSTR);
void ChangeDefExt(PSTR, PSTR);
```

File name -> bmpdisp.prj

```
#include "bmpdisp.def"
#include "bmpdisp.rc"
#include "bmpdisp.c"
```

ประวัติผู้เขียน

นายอาโมทย์ สมบูรณ์แก้ว เกิดวันที่ 28 ธันวาคม 2510 ที่จังหวัดปัตตานี จบการศึกษาระดับปริญญาตรีในหลักสูตรอุตสาหกรรมศาสตรบัณฑิต สาขาวิชาเทคโนโลยีโทรคมนาคม จากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เมื่อปี พ.ศ. 2534 หลังจากนั้นได้ศึกษาต่อและได้รับทุนการศึกษาจากมูลนิธิเพื่อการศึกษาคอมพิวเตอร์และการสื่อสาร ประจำปีการศึกษา 2535

ผลงานวิจัยที่ได้รับการตีพิมพ์

[1] อาโมทย์ สมบูรณ์แก้ว, วิทยา ทิพย์สุวรรณพร และ ฟูศักดิ์ ชีวสุวิทย์ ,"***การตรวจสอบหาชนิดวัตถุของหุ่นยนต์โดยอาศัยการเปรียบเทียบแกนหลัก***", การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 14, มหาวิทยาลัยสงขลานครินทร์ หาดใหญ่, หน้า 4-73 ถึง 4-76, 7 ถึง 8 พฤศจิกายน 2534.

[2] F. Cheevasuvit, K. Dejhan and A. Somboonkaew, ***"Edge enhancement using transform of subtracted smoothing image"***, Proc. of the 13th Asian Conference on Remote Sensing (ACRS), Ulaanbaatar, Mongolia, October 7 to 11, 1992.

[3] อาโมทย์ สมบูรณ์แก้ว และ ฟูศักดิ์ ชีวสุวิทย์, ***"การปรับปรุงรายละเอียดของขอบในภาพด้วยการแปลงภาพผลลบที่ถูกทำให้เรียบ"***, การประชุมทางวิชาการของมหาวิทยาลัยเกษตรศาสตร์ ครั้งที่ 31, มหาวิทยาลัยเกษตรศาสตร์, 3 ถึง 6 กุมภาพันธ์ 2536.

[4] F. Cheevasuvit, K. Dejhan and Armote Somboonkaew, ***"Edge enhancement in colour images using transform of subtracted smoothing image"***, Proc. of the South East Asian Regional Conference on Education and Research in Remote Sensing, Johor Bahru, Malaysia, session 6, June, 1993.

[5] อาโมทย์ สมบูรณ์แก้ว บัณฑิต สุมนวัฒนเดช และ ฟูศักดิ์ ชีวสุวิทย์, ***"การปรับปรุงรายละเอียดของขอบในภาพสีด้วยการแปลงภาพผลลบที่ถูกทำให้เรียบ"***, การประชุมใหญ่วิชาการทางวิศวกรรม ประจำปี 2536, วิศวกรรมสถานแห่งประเทศไทยในพระบรมราชูปถัมภ์, หน้า 601 ถึง 606, 27 ถึง 30 พฤศจิกายน 2536.