การกำจัดรอยตะเข็บออกจากการต่อภาพสีของภาพถ่ายดาวเทียม MOS-I MESSR

SEAM REMOVAL FROM COLOUR MOSAICKING OF MOS-I MESSR IMAGES

สักรียา ชิตวงค์
SAKREYA CHITWONG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2536

ISBN 974-621-088-2

SEAM REMOVAL FROM COLOUR MOSAICKING OF MOS-I MESSR IMAGES

SAKREYA CHITWONG

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 974-621-088-2

หัวข้อวิทยานิพนธ์ การกำจัดรอยตะเข็บออกจากการต่อภาพสีของภาพถ่ายดาวเทียม MOS-I MESSR

นักศึกษา นายสักรียา ชิตวงค์

อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์

ระดับการศึกษา วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

ภาควิชา วิศวกรรมไฟฟ้า สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2536

บทคัดย่อ

การวิเคราะห์ความหมายของภาพถ่ายในระยะไกลจากข้อมูลภาพของดาวเทียมสำรวจทรัพยากรธรรมชาติสามารถกระทำการวิเคราะห์ความหมายของภาพได้เฉพาะบางส่วนของบริเวณพื้นที่ทั้งหมดที่ต้องการทำให้ไม่สามารถวิเคราะห์ความหมายของภาพพื้นที่บริเวณกว้างๆ ที่ปรากฏอยู่คนละภาพ ดังนั้นวิทยานิพนธ์ฉบับนี้จะกล่าวถึงเทคนิคการต่อภาพสีโดยใช้ข้อมูลภาพถ่ายระยะไกลของดาวเทียมสำรวจทรัพยากรธรรมชาติเพื่อให้การวิเคราะห์ความหมายภาพกระทำได้อย่างมีประสิทธิภาพยิ่งขึ้น ในการต่อภาพนั้นจะต้องใช้ภาพถ่ายอย่างน้อย 2 ภาพขึ้นไปมาต่อกันโดยภาพทั้งสองได้จากการถ่ายภาพในบริเวณใกล้เคียงกัน เนื่องจากการถ่ายภาพแต่ละครั้งจะถ่ายได้เฉพาะบางส่วนเท่านั้น สำหรับบริเวณที่เหลือก็จะต้องถ่ายในครั้งต่อไป สังเกตได้ว่าในการถ่ายภาพแต่ละครั้งจะต่างเวลากันเสมอทำให้สภาพแวดล้อมและบรรยากาศต่างๆเปลี่ยนแปลงไปตามกาลเวลา ซึ่งทำให้ภาพที่ได้แม้จะเป็นภาพบริเวณใกล้เคียงกันแต่มีคุณสมบัติแตกต่างกันเช่นค่าระดับสีเทา จำนวนก้อนเมฆ เป็นต้น เมื่อนำภาพถ่ายดังกล่าวมาต่อกันจะทำให้เกิดรอยตะเข็บขึ้นตรงบริเวณรอยต่อ ซึ่งจะเป็นสาเหตุหนึ่งที่จะนำไปสู่การวิเคราะห์ความหมายภาพผิดพลาดได้งานวิจัยผ่านมาการลบรอยตะเข็บตรงบริเวณรอยต่อด้วยการปรับความสว่างของภาพที่นำมาต่อให้มีค่าระดับสีเทาใกล้เคียงกันก่อน จากนั้นไบอัสด้วยฟังก์ชั่นเชิงเส้นภาพใหม่ที่ได้หลังต่อภาพจะปรากฏแถบริ้วรอยในบริเวณไม่ใช่รอยต่อ ดังนั้นเพื่อแก้ปัญหาดังกล่าววิทยานิพนธ์ฉบับนี้ได้เสนอวิธีการไบอัสใหม่ด้วยการประยุกต์ใช้ฟังก์ชั่นไม่เป็นเชิงเส้นแบบบัทเตอร์เวิร์ธ ซึ่งทำให้ได้ภาพใหม่ที่ผ่านการต่อปรากฏจากรอยตะเข็บตรงบริเวณ

รอยต่อและแถบริ้วรอยที่ปรากฏขึ้นจะหายไป

ในส่วนที่สองของวิทยานิพนธ์จะเสนอวิธีการตรวจสอบการเปลี่ยนแปลงบนพื้นดินด้วยเทคนิคการวิเคราะห์องค์ประกอบหลัก ทำได้โดยการนำข้อมูลภาพถ่ายดาวเทียม 2 ภาพที่ได้จากบริเวณเดียวกันแต่ถูกบันทึกในเวลาที่แตกต่างกันหรือต่างฤดูกันจะพบว่าองค์ประกอบหลักลำดับสูงๆ ซึ่งมีค่าสหสัมพันธ์ต่ำจะถูกนำมาใช้ในการตรวจสอบการเปลี่ยนแปลงของพื้นดิน

THESIS TITLE Seam Removal from Colour Mosaicking of MOS-I MESSR Images
STUDENT MR.SAKREYA CHITWONG
THESIS ADVISOR ASSOC.PROF.DR.FUSAK CHEEVASUVIT
LEVEL OF STUDY MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING
DEPARTMENT ELECTRICAL ENGINEERING KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
YEAR 1993

Abstract

Remote sensing imagery data is used for interpretation to perform a subset of total area. Therefore, each separating land-cover images can not perform precisely for interpretation. This thesis proposes color mosaicking technique for remote sensing imagery data. Two or more contiguos images scan from near area, which will be jointed into a mosaic. Generally, each image has different characteristic such as gray level, number of cloud, and etc., while all image scan from different time. The existed seam or spurious artificial edge in the mosaiced image is caused by the variation of brightness value across the join, consequently the interpretation of image can not accomplish with the high efficientcy. Previously a gray scale matching method has been applied to minimize this effect and then the linear function bias will be used. This method makes some streak in outside of seam point region, for solving this problem. The Butterworth bias function has been used over the joint region, thus the seamless mosaiced image will be obtained as show in the results.

The second part of this thesis presents a land-cover change detection method by principal component analysis technique based-on two different times or season images of the same area. For the proposal land-cover change detection method, the high order of principal component which has low correlation will be used as a changed decision factor.

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงได้ดีด้วยความอนุเคราะห์และช่วยเหลือในการทำวิจัยจากอาจารย์ดังนี้

รองศาสตราจารย์ ดร.ฟูศักดิ์ ชีวสุวิทย์ อาจารย์ที่ปรึกษาที่ได้กรุณาให้คำปรึกษาและชี้แนะแนวทางในการทำงานวิจัยและการแก้ไขปัญหาต่าง ๆ อย่างทุ่มเทด้วยดีตลอดมา จนทำให้ผู้เขียนมีความสามารถในการทำงานวิจัยและมีการพัฒนาได้อย่างมีประสิทธิภาพ

ขอกราบขอบพระคุณ รองศาสตราจารย์ ดร.กอบชัย เดชหาญ ที่ได้ให้คำแนะนำและแนวทางในการทำวิทยานิพนธ์นี้

ในท้ายนี้ขอกราบขอบพระคุณคุณพ่อและคุณแม่ที่คอยเป็นกำลังใจแก่ผู้เขียนตลอดเวลา และขอขอบคุณเพื่อน ๆ และ น้อง ๆ ของผู้เขียนที่มีส่วนช่วยเหลือในวิทยานิพนธ์นี้

สารบัญ

บทที่ 1
บทนำ

ปัจจุบันประเทศไทยได้มีสถานีรับสัญญาณภาพถ่ายดาวเทียมต่างๆ เช่น LANDSAT , SPOT และ MOS-I รวมทั้งภาพถ่ายจากเรดาร์ (SAR) คือ JERS-1 เพื่อใช้สำหรับการสำรวจทรัพยากรธรรมชาติต่างๆและการตรวจสอบสภาพพื้นผิวของโลกและการใช้งานอื่นๆ ซึ่งมีการเปลี่ยนแปลงไปตามกาลเวลา และปัจจุบันยิ่งทวีความรุนแรงเพิ่มขึ้นทุกวัน อันเนื่องมาจากการกระทำของมนุษย์โดยตรงเช่น การตัดไม้ทำลายป่า การบุกรุกพื้นที่ป่าสงวน การเติบโตของสังคมเมือง รวมไปถึงสิ่งที่เกิดขึ้นเองจากธรรมชาติเช่น การเกิดน้ำท่วม การตื้นเขินของชายฝั่งทะเล เป็นต้น ด้วยสาเหตุดังกล่าวทำให้ประเทศไทยได้หันมาให้ความสำคัญกับการใช้ข้อมูลภาพถ่ายดาวเทียมเพื่อการพัฒนาประเทศ โดยมอบหมายให้หน่วยราชการต่างๆรับผิดชอบตามลักษณะของงานที่กระทำ เช่น กรมป่าไม้ใช้สำหรับตรวจจับการตัดไม้ทำลายป่า กรมวิชาการเกษตรใช้สำหรับการสำรวจพื้นที่เพาะปลูก สำนักงานป้องกันและปราบปรามยาเสพติดใช้สำหรับตรวจจับพื้นที่เพาะปลูกกัญชา ฝิ่น เป็นต้น จะเห็นว่าประโยชน์จากการประยุกต์ใช้งานข้อมูลภาพถ่ายดาวเทียมมีมากมาย แต่การที่จะใช้ข้อมูลภาพให้บรรลุวัตถุประสงค์และมีประสิทธิภาพนั้นจะต้องอาศัยเทคโนโลยีการประมวลผลข้อมูล (information processing) ผสมผสานกันตามความเหมาะสม โดยเทคโนโลยีดังกล่าวหน่วยงานราชการภายในประเทศได้สั่งนำเข้าจากต่างประเทศแทบทั้งสิ้น ทั้งนี้เพราะว่าประเทศไทยยังไม่มีศักยภาพพอที่จะพัฒนาเทคโนโลยีดังกล่าวใช้งานภายในประเทศได้ รวมไปถึงระบบการศึกษาของประเทศในสาขาวิชาดังกล่าวไม่เจริญก้าวหน้าพอ ทำให้ประเทศขาดบุคลากรเฉพาะสาขาที่จะพัฒนางานวิจัยเพื่อให้ได้เทคโนโลยีใหม่ๆสำหรับใช้งานภายในประเทศ ดังนั้นวิทยานิพนธ์นี้ได้นำเสนอการพัฒนาเทคโนโลยีสำหรับวิธีการประมวลผลข้อมูลภาพถ่ายดาวเทียม ด้วยการนำเสนอในลักษณะของอัลกอริทึม (algorithm) สำหรับการประมวลผลข้อมูลกับภาพถ่ายดาวเทียม MOS-I ในระบบ MESSR ซึ่งวิธีการที่นำเสนอสามารถนำไปใช้กับข้อมูลภาพของดาวเทียมดวงอื่นๆ ได้เช่น LANDSAT ระบบ MSS และ TM เป็นต้น

1.1 วัตถุประสงค์ของวิทยานิพนธ์

ปัญหาหนึ่งที่เกิดขึ้นกับผู้ใช้ข้อมูลภาพในการแปลความหมายภาพก็คือ การแปลความหมายของวัตถุที่ได้จากการถ่ายภาพต่างเวลา โดยปกติวงโคจรของดาวเทียมขณะทำการสแกนภาพจะมีวงโคจรคงที่ ดังนั้นการสแกนภาพในแต่ละครั้งจะครอบคลุมเฉพาะบางบริเวณของพื้นที่เท่านั้น ซึ่งจะมีพื้นที่กว้างเท่าไรนั้นขึ้นอยู่กับคุณสมบัติของดาวเทียมนั้นๆด้วยและในการสแกนภาพแต่ละครั้งจะกระทำ ณ เวลาต่างกันเสมอ จากเหตุผลดังกล่าวส่งผลให้มีบางส่วนของวัตถุในภาพอยู่คนละภาพอย่างน้อยสองภาพ แต่เนื่องจากภาพทั้งสองได้จากการสแกนภาพต่างเวลาทำให้มีระดับค่าความสว่างของภาพทั้งสองแตกต่างกันตามกาลเวลาและสภาพแวดล้อมขณะนั้นด้วย ซึ่งจะเห็นว่าวัตถุชิ้นเดียวกันแต่มีค่าระดับสีเทาต่างกันทำให้การแปลความหมายของวัตถุดังกล่าวกระทำไม่ได้หรือก่อให้เกิดการแปลความหมายของภาพผิดพลาดไปจากความเป็นจริง เพื่อให้การแปลความหมายภาพของวัตถุดังกล่าวกระทำได้ถูกต้อง ซึ่งจะส่งผลให้การใช้งานข้อมูลภาพถ่ายดาวเทียมมีประสิทธิภาพยิ่งขึ้น วิธีการอย่างหนึ่งคือ การนำข้อมูลภาพนั้นมาเชื่อมต่อเข้าด้วยกันให้เป็นหนึ่งเดียว อย่างไรก็ตามภาพที่จะนำมาเชื่อมต่อมักมีระดับค่าความสว่างแตกต่างกัน ดังนั้นภาพที่ได้หลังนำมาเชื่อมต่อกันตรงบริเวณรอยต่อของภาพจะเกิดเป็นรอยตะเข็บขึ้น ทำให้ภาพที่ได้หลังผ่านการเชื่อมต่อแล้วมีค่าความสว่างไม่เป็นเนื้อเดียวกัน ปัญหาดังกล่าวได้มีนักวิจัยพยายามค้นหาวิธีการกำจัดรอยตะเข็บเพื่อให้ได้ภาพใหม่มีค่าความสว่างราบรื่นตลอดภาพเช่น การต่อภาพโดยวิธีการเฉลี่ย การต่อภาพโดยใช้ฟังก์ชั่นเชิงเส้น [8] เป็นต้น เนื่องจากวิธีการดังกล่าวยังไม่ดีพอที่จะใช้เชื่อมต่อภาพในบางกรณี ดังนั้นในส่วนแรกของวิทยานิพนธ์ฉบับนี้จึงนำเสนอวิธีการเชื่อมต่อภาพด้วยการกำจัดรอยตะเข็บโดยใช้ฟังก์ชั่นไม่เป็นเชิงเส้นแบบบัทเตอร์เวิร์ธ ซึ่งจะทำให้ภาพที่ได้หลังจากการเชื่อมต่อมีค่าความสว่างต่อเนื่องราบรื่นตลอดทั้งภาพ ส่วนที่สองของวิทยานิพนธ์ฉบับนี้เป็นการนำข้อมูลภาพถ่ายดาวเทียมต่างเวลาสองภาพจากพื้นที่เดียวกันสำหรับตรวจสอบหาการเปลี่ยนแปลงทางภาคพื้นดิน เนื่องจากข้อมูลภาพถ่ายดาวเทียมประกอบด้วยข้อมูลหลายแบนด์แต่ละแบนด์จะให้ผลตอบสนองของคลื่นแม่เหล็กไฟฟ้าต่อวัตถุแตกต่างกัน ดังนั้นวิธีการที่ง่ายที่สุดในการตรวจสอบการเปลี่ยนแปลงก็คือ ทำการเปรียบเทียบข้อมูลแบบแบนด์ต่อแบนด์ แต่จากการใช้ข้อมูลเพียงหนึ่งแบนด์ในการเปรียบเทียบแต่ละครั้งไม่สามารถตรวจสอบการเปลี่ยนแปลงของวัตถุได้ทุกชนิดพร้อมๆกัน อีกทั้งดาวเทียมแต่ละดวงจะมีจำนวนแบนด์ของภาพที่แตกต่างกันออกไปขึ้นอยู่กับดาวเทียมเช่น LANDSAT ระบบ TM มี 7 แบนด์ MOS-I ในระบบ MESSR มี 4 แบนด์ ดังนั้นจะเห็นว่าหากใช้วิธีการดังกล่าวจะต้องกระทำการเปรียบเทียบ 7 และ 4 ครั้ง ตามลำดับ จึงเป็นวิธีการที่ไม่สะดวกและยากในการรวบรวมข้อมูลและกินเวลาในการประมวลผล ในการแก้ปัญหาดังกล่าวและการเพิ่มประสิทธิภาพให้มีความแม่นยำในการตรวจสอบการเปลี่ยนแปลงของพื้นดินนั้นทำได้ด้วยการปรับฐานของข้อมูลภาพทั้งหมดทุกแบนด์ให้อยู่ในฐานเดียวกันก่อนแล้วจึงนำมาเปรียบเทียบกันโดยอาศัยคุณสมบัติของทฤษฎีการวิเคราะห์องค์

ประกอบหลัก ทำให้ได้ภาพภายหลังผ่านการประมวลผลแสดงให้เห็นบริเวณที่มีการเปลี่ยนแปลงเกิดขึ้นจากภาพทั้งสอง

1.2 ขอบเขตการวิจัย

ในส่วนของการวิจัยนั้นเป็นการเสนอเทคนิคของกำจัดรอยตะเข็บด้วยวิธีการต่างๆเพื่อจะให้ได้ภาพใหม่หลังการเชื่อมต่อที่มีค่าความสว่างต่อเนื่องราบรื่นตลอดภาพและวิธีการตรวจสอบการเปลี่ยนแปลงของพื้นดิน โดยมีรายละเอียดของวิทยานิพนธ์นี้ แบ่งออกได้เป็น 6 บทดังนี้

บทที่ 1 เป็นบทนำ กล่าวถึงวัตถุประสงค์ และขอบเขตของวิทยานิพนธ์

บทที่ 2 เป็นวิธีการต่อภาพแบบดั้งเดิม โดยอาศัยการเฉลี่ยค่าระดับสีเทาของภาพทั้งสองเฉพาะบริเวณที่ซ้อนทับกันเท่านั้น ในบทนี้ได้กล่าวถึงเทคนิคการหาตำแหน่งซ้อนทับของภาพและการปรับค่าความสว่างของภาพทั้งสองให้มีค่าสอดคล้อง ซึ่งต่อไปจะนำไปใช้สำหรับการเชื่อมต่อภาพในบทที่ 3 และ 4 ด้วย

บทที่ 3 เป็นการต่อภาพโดยใช้ฟังก์ชั่นเชิงเส้นสำหรับการกำจัดรอยตะเข็บ เนื่องจากวิธีการเฉลี่ยค่าระดับสีเทาภาพใหม่หลังผ่านการเชื่อมต่อจะได้ค่าความสว่างใหม่ 3 บริเวณและบริเวณซ้อนทับภาพทั้งสองจะสูญเสียความคมชัด

บทที่ 4 เป็นการต่อภาพโดยใช้ฟังก์ชั่นไม่เป็นเชิงเส้นแบบบัทเตอร์เวิร์ธสำหรับการกำจัดรอยตะเข็บ เนื่องจากวิธีการเดิม (การกำจัดรอยตะเข็บโดยใช้ฟังก์ชันเชิงเส้น) ถ้าหากใช้กับข้อมูลภาพที่มีจุดสว่างเป็นเนื้อเดียว จะก่อให้เกิดความสว่างใหม่ที่ไม่ต่อเนื่องเป็นแถบริ้วรอย ทำให้ภาพใหม่ที่ได้หลังผ่านการกำจัดรอยตะเข็บมีค่าความสว่างไม่ต่อเนื่อง ซึ่งสามารถแก้ปัญหาดังกล่าวได้โดยการใช้ฟังก์ชั่นไม่เป็นเชิงเส้นแบบบัทเตอร์เวิร์ธแทนฟังก์ชั่นเชิงเส้น

บทที่ 5 การตรวจสอบการเปลี่ยนแปลงพื้นดิน เป็นการประยุกต์ใช้เทคนิคการวิเคราะห์องค์ประกอบหลักเพื่อใช้สำหรับการตรวจสอบการเปลี่ยนแปลงของพื้นดิน โดยอาศัยข้อมูลภาพถ่ายดาวเทียมอย่างน้อย 2 ภาพที่ได้จากการสแกนบริเวณเดียวกันแต่ต่างเวลากัน จะได้ข้อมูลภาพใหม่แสดงให้เห็นบริเวณที่เกิดการเปลี่ยนแปลงจากสาเหตุต่าง ๆ เช่น การตัดไม้ทำลายป่า การเกิดน้ำท่วม เป็นต้น

บทที่ 6 บทสรุป เป็นการสรุปผลการวิจัยและข้อเสนอแนะต่าง ๆ

และในส่วนสุดท้ายซึ่งเป็นภาคผนวก จะเป็นรายละเอียดของโปรแกรมที่ใช้ในแต่ละบทเอาไว้เพื่อความสะดวกของผู้ที่จะค้นคว้าวิจัยต่อไป

บทที่ 2
การต่อภาพโดยวิธีการเฉลี่ย

2.1 คำนำ

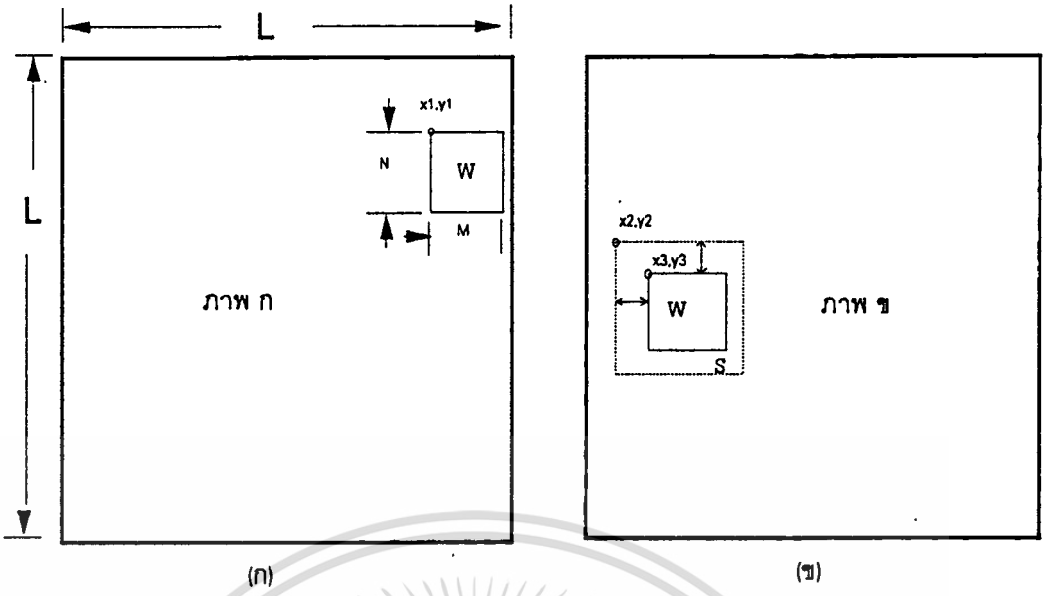
ในบทนี้จะกล่าวถึงแนวความคิดเบื้องต้นเกี่ยวกับวิธีการต่อภาพอย่างง่ายคือ การต่อภาพโดยวิธีการเฉลี่ยค่าระดับสีเทาของภาพเฉพาะบริเวณที่ซ้อนทับกันข้อมูลภาพที่จะนำมาต่อกันปกติแล้วจะมีบางบริเวณของภาพซ้อนกันเสมอ ดังนั้นก่อนที่จะนำข้อมูลภาพต่างๆมาต่อกันจึงจำเป็นต้องทำการคำนวณหาตำแหน่งของภาพที่ซ้อนทับกันเสียก่อน โดยการค้นหาพื้นที่บางส่วนของข้อมูลภาพที่มีลักษณะเหมือนกันมากที่สุด วิธีการที่ได้รับความนิยมมากคือการทำสหสัมพันธ์ (Correlation) ของข้อมูลภาพทั้งสองเพื่อที่จะคำนวณหาตำแหน่งต่างๆของภาพตรงบริเวณที่ซ้อนทับกัน จากนั้นทำการปรับความสว่างของภาพที่จะนำมาเชื่อมต่อให้สอดคล้องก่อนแล้วจึงทำการเชื่อมต่อภาพโดยอาศัยตำแหน่งดังกล่าว ดังมีรายละเอียดต่อไปนี้

2.2 เทคนิคการหาตำแหน่งซ้อนทับของภาพ

ก่อนที่จะนำภาพมาเชื่อมต่อกันสิ่งแรกที่จะต้องกระทำคือ การหาตำแหน่งซ้อนทับของภาพก่อน โดยปกติแล้วภาพถ่ายดาวเทียมแต่ละชีนที่ได้จากบริเวณใกล้เคียงกันจะมีพื้นที่บางส่วนซ้อนทับกันอยู่ประมาณ 40 เปอร์เซนต์ ดังนั้นในการหาตำแหน่งรอยต่อของภาพจึงอาศัยความเป็นพื้นที่เดียวกันหรือความเหมือนมาเป็นเงื่อนไขเพราะพื้นที่ซ้อนทับกันย่อมมีความเหมือนสูงสุด ถึงแม้ว่าจุดภาพตำแหน่งเดียวกันจากภาพซ้อนทับทั้งสองจะมีค่าระดับสีเทาต่างกันก็ตาม แต่จะเป็นความแตกต่างที่เกือบจะคงที่ตลอดพื้นที่ซ้อนทับ

ในการหาตำแหน่งซ้อนทับของภาพสามารถกระทำได้ด้วยการตัดภาพจากภาพใหญ่ให้มีขนาดเล็กลงเพื่อนำไปเป็นภาพย่อยของภาพใหญ่ จากนั้นนำภาพย่อยชิ้นดังกล่าวไปเปรียบเทียบกับอีกภาพที่ต้องการนำมาต่อด้วยดังแสดงในรูปที่ 2.1 วิธีการเปรียบเทียบข้อมูลภาพมีอยู่ 3 แบบคือ

1. ความเหมือน
2. ความเหมือนแบบปรับตามค่าเฉลี่ย
3. การทำสหสัมพันธ์



รูปที่ 2.1 การหาตำแหน่งซ้อนทับของภาพ

จากรูปที่ 2.1 (ก) เลือกภาพย่อย W ณ ตำแหน่งเริ่มต้น (x_1, y_1) ขนาด M*N จุดภาพจากภาพใหญ่ขนาด L*L จุดภาพ โดยเลือกจากตำแหน่งส่วนที่มีลักษณะเด่นชัดอย่างเช่น มีแม่น้ำหรือทางแยกถนน เป็นต้น

2.2.1 วิธีการหาตำแหน่งซ้อนทับภาพวัดค่าความเหมือน

นำภาพย่อย W ไปเปรียบเทียบกับอีกภาพดังรูปที่ 2.1 (ข) ซึ่งกำหนดให้ตำแหน่งเริ่มต้นที่ (0,0) หรือโดยการกำหนดตำแหน่งเริ่มต้นตามความเหมาะสมขึ้นอยู่กับลักษณะของข้อมูลทั้งสองเมื่อเทียบกับภาพย่อย W เช่น ณ ตำแหน่ง (x_2, y_2) ทั้งนี้เพื่อที่จะเป็นการเพิ่มความเร็วสำหรับการคำนวณหาตำแหน่งซ้อนทับของภาพ สมมติว่านำภาพย่อย W วางที่ตำแหน่ง (x_2, y_2) จากนั้นคำนวณหาค่าความแตกต่าง $E_0(i,j)$ โดยใช้สมการที่ (2.1) ถ้าผลการคำนวณ $E_0(i,j)$ มีค่าใกล้เคียงศูนย์ที่ตำแหน่ง $i = x_3$ และ $j = x_3$ หมายความว่า ภาพย่อย W กับ ภาพ S เหมือนกันทุกประการ ณ ตำแหน่งเริ่มต้น (x_3, y_3) ถ้าหากไม่สามารถคำนวณหาตำแหน่งที่ $E_0(i,j)$ มีค่าใกล้เคียงศูนย์ตลอดภาพได้ก็แสดงว่าข้อมูลภาพดังกล่าวไม่เหมือนกัน

$$E_0(i,j) = \sum_{n=1}^{N}\sum_{m=1}^{M}|w(n,m) - s(i+n-1, j+m-1)| \qquad (2.1)$$

เมื่อ w คือ ข้อมูลภาพย่อย w ขนาด M*N

s คือ ข้อมูลภาพนำมาเปรียบเทียบกับภาพย่อย w

i,j คือ ตำแหน่งของข้อมูลภาพ s เปรียบเทียบกับภาพย่อย w

2.2.2 วิธีการหาตำแหน่งซ้อนทับภาพวัดค่าความเหมือนแบบปรับตามค่าเฉลี่ย

เป็นวิธีการเปรียบเทียบโดยการคำนวณหาค่าความเหมือนเช่นเดียวกับหัวข้อ 2.2.1 แต่จะให้ประสิทธิภาพที่สูงกว่าทั้งนี้เพราะจะทำการปรับข้อมูลภาพให้อยู่ในฐานเดียวกัน (รอบๆค่าเฉลี่ยของแต่ละภาพ) สมการที่ใช้ในการคำนวณความเหมือนคือ

$$E_1(i,j) = \sum_{n=1}^{N}\sum_{m=1}^{M}|\{w(n,m)-\overline{w}\}-\{s(i+n-1,j+m-1)-\overline{s}\}| \qquad (2.2)$$

เมื่อ $\overline{w} = \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}w(n,m)$ (เป็นค่าเฉลี่ยของภาพย่อย)

และ $\overline{s}_{ij} = \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}s(i+n-1,j+m-1)$ (เป็นค่าเฉลี่ยของภาพใหญ่)

ถ้าหาก $E_1(i,j)$ เท่ากับศูนย์หมายความว่า ข้อมูลภาพที่นำมาเปรียบเทียบมีความเหมือน แต่ถ้า $E_1(i,j)$ มีค่าเข้าใกล้ศูนย์ ($E_1(i,j) > 0$ แต่ไม่เท่ากับ 0) ก็แสดงว่า ข้อมูลภาพที่นำมาเปรียบเทียบแค่มีความคล้ายกันเท่านั้น ทั้งนี้การที่จะตัดสินใจได้ว่าข้อมูลภาพจะมีความคล้ายกันมากแค่ไหนจะขึ้นอยู่กับ $E_1(i,j)$ ว่าเข้าใกล้เคียงศูนย์มากเพียงใด

วิธีการ 2.2.2 นี้ก็จะก่อให้เกิดปัญหาได้เช่นกัน ถ้าภาพที่นำมาต่อเป็นภาพที่เก็บในระยะเวลาที่ต่างกันมาก ทั้งนี้เพราะเกิดการเปลี่ยนแปลงการใช้พื้นดิน ทำให้อัตราการสะท้อนของแสงจากพื้นดินในภาพทั้งสองแตกต่างกัน ซึ่งไม่ได้เป็นความแตกต่างคงที่ของความสว่างในจุดภาพเพียงอย่างเดียว

2.2.3 การทำสหสัมพันธ์ (Correlation)

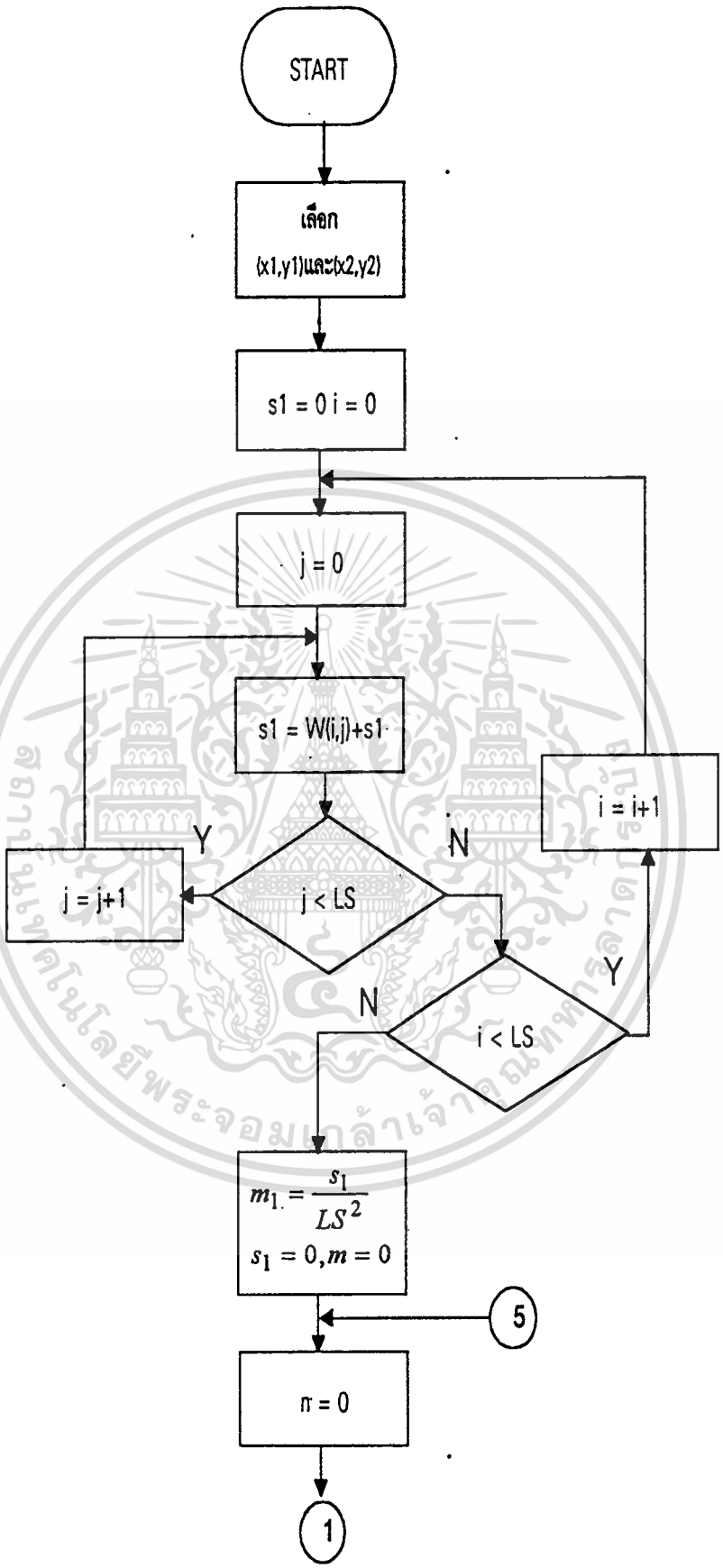
วิธีการคำนวณหาความเหมือนของข้อมูลภาพโดยการทำสหสัมพันธ์เป็นวิธีการซึ่งได้รับความนิยมใช้สำหรับหาค่าความเหมือนของข้อมูลภาพมากที่สุดเพราะสามารถใช้แก้จุดบกพร่องทั้งจากวิธีการ 2.2.1 และ 2.2.2 โดยการคำนวณหาความเหมือนจะใช้สมการที่ (2.3)

$$E_2(i,j) = \frac{\sum_{n=1}^{N}\sum_{m=1}^{M}\{w(n,m)-\overline{w}\}\{s(i+n-1,j+m-1)-\overline{s}\}}{\sqrt{\sum_{n=1}^{N}\sum_{m=1}^{M}\{w(n,m)-\overline{w}\}^2}\ \sqrt{\sum_{n=1}^{N}\sum_{m=1}^{M}\{s(i+n-1,j+m-1)-\overline{s}\}^2}} \quad (2.3)$$

ในการคำนวณถ้า $E_2(i,j)$ มีค่าเท่ากับ 1 หมายความว่า ข้อมูลภาพที่นำมาเปรียบเทียบคล้ายกันทุกประการ แต่ถ้า $E_2(i,j)$ มีค่าเข้าใกล้ 1 ($E_2(i,j) < 1$ แต่ไม่เท่ากับ 1) ก็หมายความว่า ข้อมูลภาพที่นำมาเปรียบเทียบคล้ายกันมากๆ ทั้งนี้การที่จะตัดสินใจได้ว่าข้อมูลภาพจะมีความคล้ายกันมากแค่ไหนจะขึ้นอยู่กับ $E_2(i,j)$ ว่าเข้าใกล้เคียง 1 มากเพียงใด ในทางตรงข้ามถ้าค่า $E_2(i,j)$ มีค่าต่ำมาก ๆ เช่น น้อยกว่า 0.5 ซึ่งถือได้ว่า ข้อมูลภาพที่นำมาเปรียบเทียบกันนั้นมีความเหมือนที่ไม่อาจยอมรับได้ นั่นก็คือ ตำแหน่งของข้อมูลภาพ (x_3,y_3) ที่ได้จะมีความคลาดเคลื่อนไปจากค่าที่ควรจะเป็นจนไม่อาจยอมรับได้

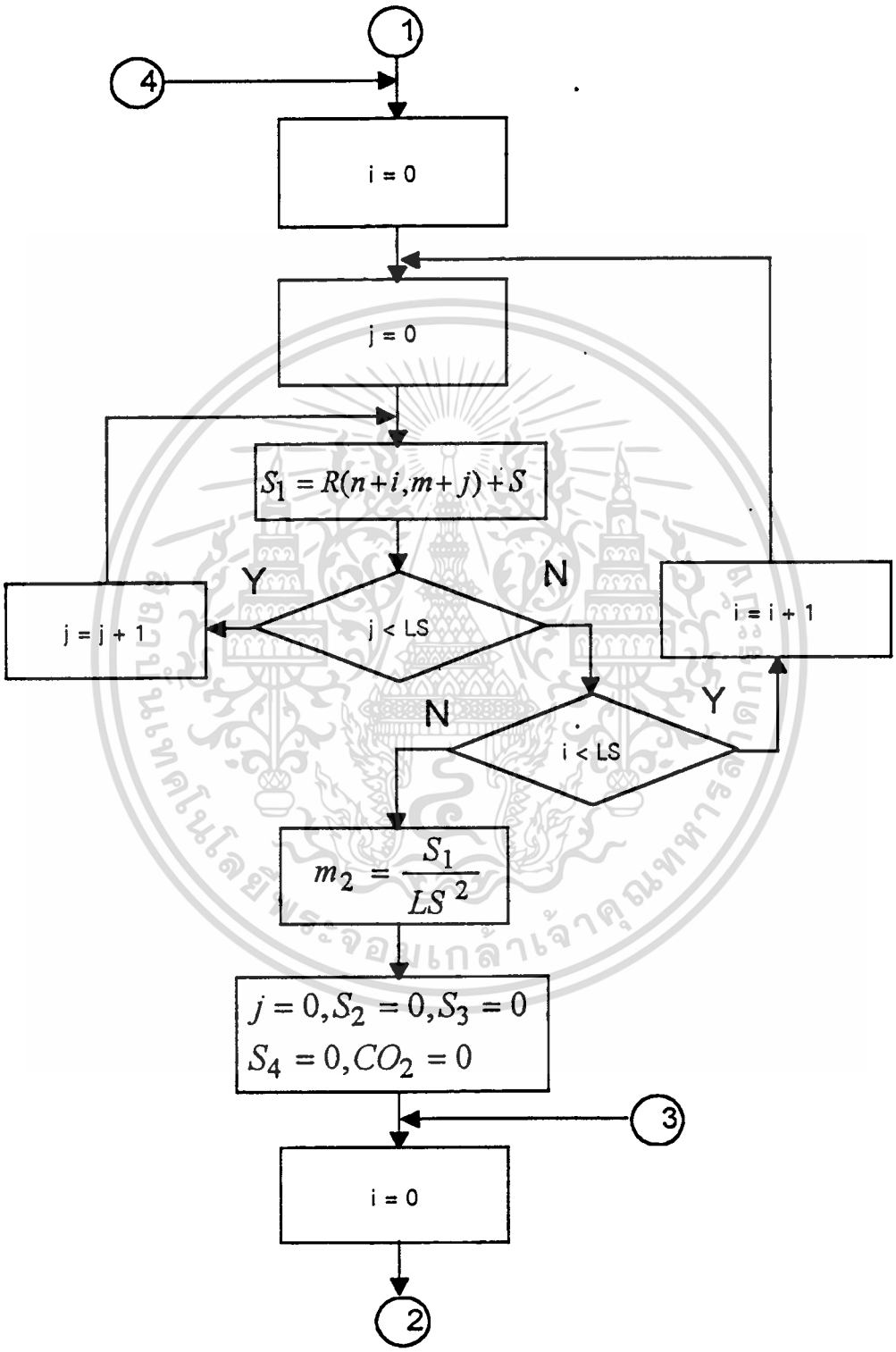
จากวิธีการคำนวณหาค่าความเหมือนหัวข้อที่ 2.2.1., 2.2.2 และ 2.2.3 ผลการทดลองวิธีที่ 3 ให้ผลลัพธ์ดีที่สุดคือ **วิธีการทำสหสัมพันธ์** ดังนั้นในวิทยานิพนธ์ฉบับนี้ใช้สมการที่ (2.3) สำหรับคำนวณหาตำแหน่งซ้อนทับของข้อมูลภาพ ซึ่งค่า $E_2(i,j)$ จากการคำนวณได้จะให้ตำแหน่งถูกต้องและแม่นยำสุง สมมติว่าภาพย่อย w คล้ายกับภาพ s มากที่สุดที่ตำแหน่ง (x_3,y_3) ดังนั้นสามารถคำนวณหาตำแหน่งซ้อนทับได้คือ $ncox = x_2 + x_3$ และ $ncoy = y_2 + y_3$ ซึ่งค่า $ncox$ และ $ncoy$ ต่อไปจะนำไปใช้เป็นตำแหน่งอ้างอิงสำหรับการต่อภาพ จะกล่าวรายละเอียดในหัวข้อที่ 2.4 วิธีการดังที่ได้กล่าวมาแล้วสามารถเขียนเป็นโพล์ชาร์ทได้ดังรูปที่ 2.2

จากการทดลองและประสบพการณ์ที่ผ่านมาสามารถสรุปสาเหตุที่ทำให้การหาตำแหน่งซ้อนทับของภาพกระทำไม่ได้ก็เนื่องมาจากคุณภาพของข้อมูลภาพของดาวเทียมบางดวงมีค่าตัวอย่างเช่น (1) ดาวเทียม MOS-I ในระบบ MESSR จะมีค่า dynamic range ค่อยข้างแคบซึ่งจะทำให้การแบ่งแยกความแตกต่างของข้อมูลภาพบางภาพไม่สามารถกระทำได้และ (2) เมฆได้ทับตรงบริเวณจุดเด่นของข้อมูลภาพบางส่วนเช่น จุดตัดถนน แม่น้ำ สนามบินเป็นต้น วิธีการแก้ไขขั้นต้นทำได้โดยการพยายามเลือกภาพย่อยที่คุณลักษณะเด่นมากที่สุดที่จะนำมาเชื่อมต่อกันเฉพาะภาพที่เหมาะสมที่สุด

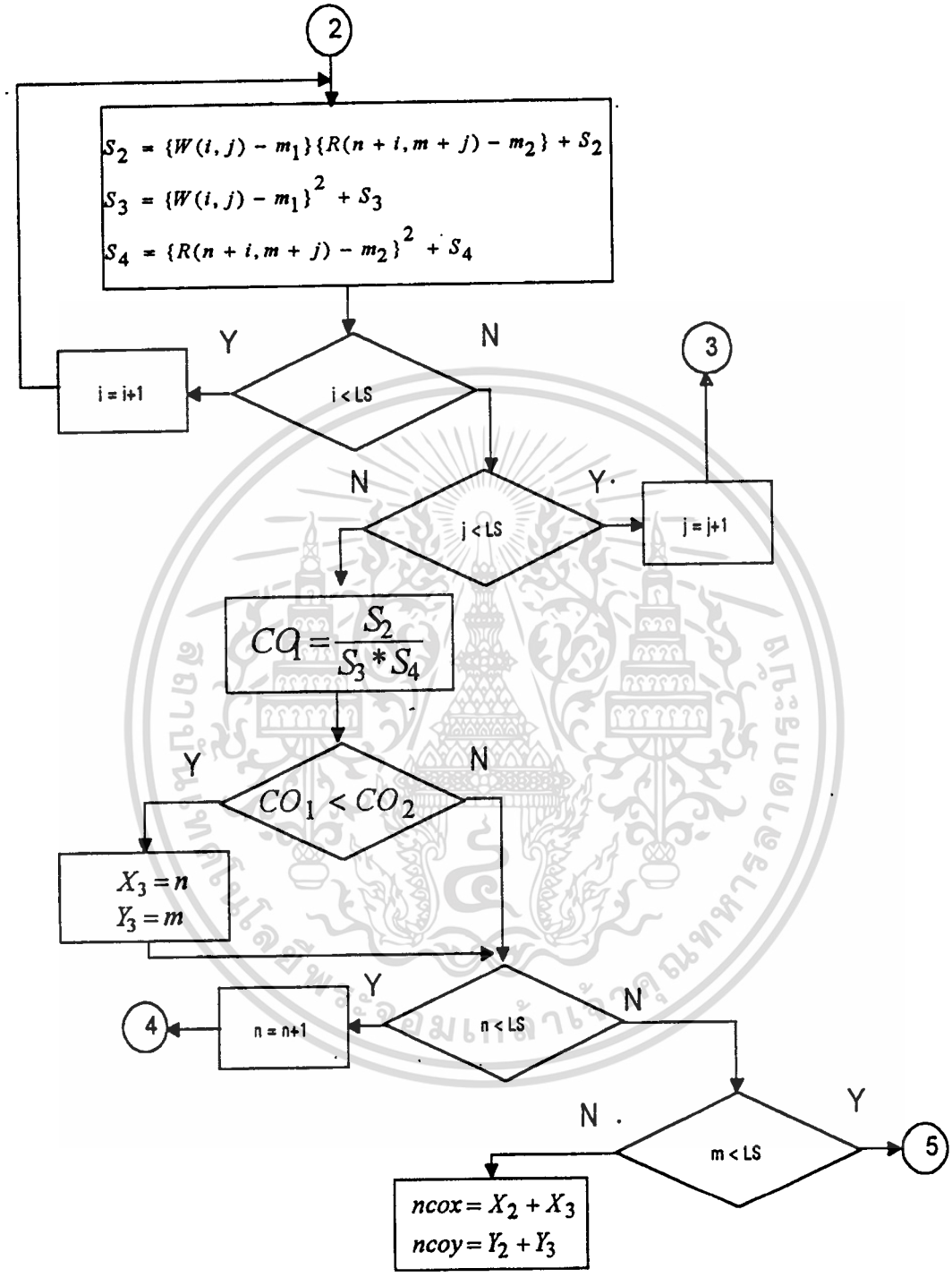


รูปที่ 2.2 แสดงโพล์ชาร์ท การคำนวณหาตำแหน่งซ้อนทับ

รูปที่ 2.2 ต่อ



รูปที่ 2.2 ต่อ



2.3 วิธีการปรับความสว่างให้สอดคล้องกัน (Brightness matching)

โดยปกติภาพถ่ายดาวเทียมที่จะนำมาต่อเข้าด้วยกันนั้นได้รับจากการถ่ายภาพด้วยดาวเทียม ณ บริเวณใกล้เคียงกันและจะมีบางส่วนของภาพซ้อนทับกัน เนื่องจากภาพทั้งสองที่จะนำมาต่อโดยปกติมักจะถ่ายในเวลาที่แตกต่างกัน จึงเป็นสาเหตุทำให้ภาพถ่ายที่ได้แต่ละภาพมีค่าความสว่างแตกต่างกันไปตามสภาพแวดล้อมขณะเวลานั้น โดยภาพที่ได้ ณ เวลานั้น ๆ จะแตกต่างกันมากน้อยเพียงใดจะขึ้นอยู่กับปัจจัยต่าง ๆ เช่น จำนวนก้อนเมฆ ฤดูกาล เป็นต้น

เมื่อนำภาพถ่ายที่ได้จากการถ่ายภาพต่างเวลามาเชื่อมต่อเข้าด้วยกัน ทำให้ภาพผลลัพธ์ที่ได้หลังเชื่อมต่อเกิดความไม่ต่อเนื่องของความสว่าง ซึ่งจะทำให้เกิดรอยตะเข็บตรงบริเวณรอยต่อของภาพอย่างเห็นได้ชัดเจนและยากที่จะลบรอยตะเข็บดังกล่าวให้หมดไปได้ ดังนั้นวิธีการปรับความสว่างให้สอดคล้องกันก่อนที่จะทำการต่อภาพเป็นวิธีการหนึ่งที่ช่วยลดความแตกต่างของความสว่างในภาพทั้งสองให้มีค่าใกล้เคียงกันยิ่งขึ้นก่อนที่จะนำไปทำการลบรอยตะเข็บด้วยวิธีการเฉลี่ยค่าระดับสีเทา วิธีการไบอัสด้วยฟังก์ชั่นเชิงเส้นและวิธีการไบอัสด้วยฟังก์ชั่นบัทเตอร์เวิร์ธ ซึ่งจะกล่าวรายละเอียดในบทที่ 3 และ 4 ตามลำดับต่อไป วิธีการปรับความสว่างมีรายละเอียดต่างๆดังนี้

คำนวณหาค่าเฉลี่ย m_1 , m_2 และค่าเบี่ยงเบน p_1, p_2 ของข้อมูลภาพ ดังสมการที่ (2.4)

$$m_1 = \frac{1}{T}\sum_{i=0}^{T-1} g_1(i),\ p_1 = \sqrt{\frac{1}{T}\sum_{i=0}^{T-1}(g_1(i)-m_1)^2}$$

$$m_2 = \frac{1}{T}\sum_{i=0}^{T-1} g_2(i),\ p_2 = \sqrt{\frac{1}{T}\sum_{i=0}^{T-1}(g_2(i)-m_2)^2} \qquad (2.4)$$

จากนั้นทำการปรับค่าความสว่างของภาพโดยใช้สมการที่ (2.5)

$$P = \frac{p_1}{p_2}Q + m_1 - \frac{p_1}{p_2}m_2 \qquad (2.5)$$

เมื่อ Q คือ ระดับค่าสีเทาของจุดภาพเดิม

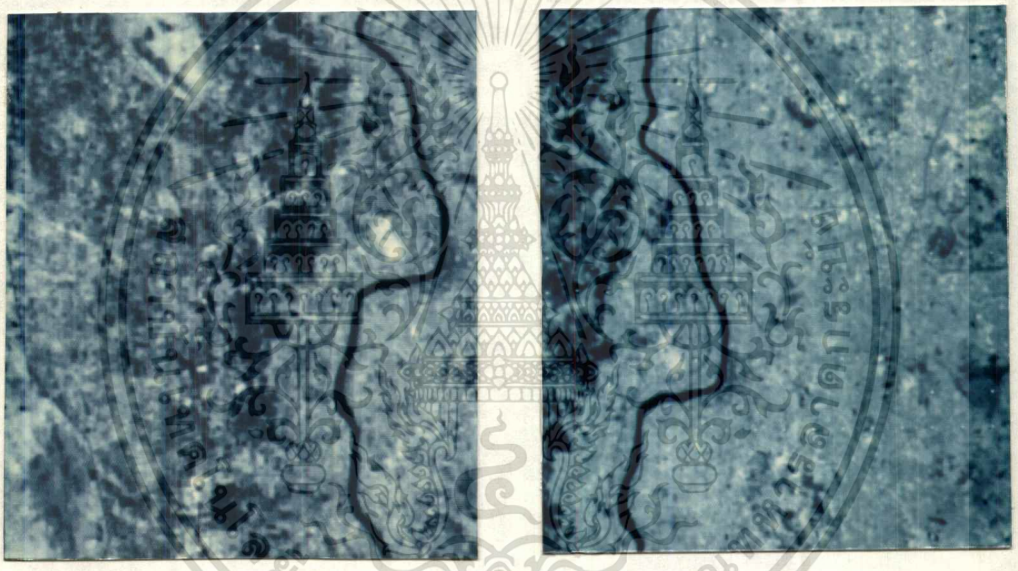
P คือ ระดับค่าสีเทาของจุดภาพใหม่

m_1,p_1 คือ ค่าเฉลี่ยและค่าเบี่ยงเบนมาตราฐานของข้อมูลภาพอ้างอิงจากพื้นที่ส่วนบริเวณข้อบทับ

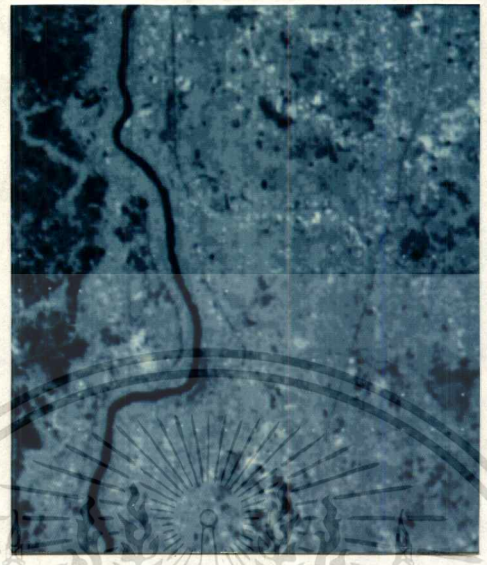
m_2, p_2 คือ ค่าเฉลี่ยและค่าเบี่ยงเบนมาตราฐานของข้อมูลภาพที่จะทำการปรับความสว่างจากพื้นที่ส่วนบริเวณซ้อนทับ

รูปที่ 2.3 (ก) เป็นตัวอย่างภาพสองภาพที่มีความสว่างแตกต่างกัน ซึ่งต้องการนำมาเชื่อมต่อเข้าด้วยกัน จากการใช้สมการที่ (2.5) เพื่อปรับค่าความสว่างของภาพขวามือของรูปที่ 2.3 (ก) ให้เข้าใกล้ค่าความสว่างของภาพรูปทางซ้ายมือของรูปที่ 2.3 (ก) (ภาพอ้างอิง) จะได้ภาพผลลัพธ์ดังแสดงในรูปที่ 2.3 (ข)

จากภาพอ้างอิงและภาพที่ผ่านการปรับความสว่างจะถูกนำมาคำนวณหาตำแหน่งซ้อนทับด้วยวิธีการที่กล่าวไว้ในหัวข้อที่ 2.2.3



(ก) ภาพต้นแบบก่อนปรับความสว่าง โดยทำการปรับค่าความสว่างในภาพขวามือเข้าหาภาพทางซ้ายมือ



(ข) ภาพผลลัพธ์หลังปรับค่าความสว่าง

รูปที่ 2.3 การปรับความสว่างในภาพก่อนทำการคำนวณหาตำแหน่งการซ้อนทับ

2.4 การต่อภาพโดยวิธีการเฉลี่ยค่าระดับสีเทา

หลังจากคำนวณหาตำแหน่งซ้อนทับได้แล้ว ขั้นตอนต่อไปจะเป็นการต่อภาพในการต่อภาพแบบวิธีการดั้งเดิมทำได้ด้วยการนำภาพที่จะนำมาเชื่อมต่อกันวางซ้อนทับโดยอาศัยตำแหน่งต่างๆ ซึ่งหาได้จากวิธีการในหัวข้อที่ 2.2 ดังรูปที่ 2.4 (ก) จากนั้นประยุกต์ใช้ฟังก์ชั่นน้ำหนักแบบเชิงเส้นจำนวน 2 ฟังก์ชั่นดังรูปที่ 2.4 (ข) ซึ่งจะเห็นว่าผลบวกของน้ำหนักของฟังก์ชั่นทั้งสองแต่ละจุดภาพมีค่าเท่ากับ 1 เสมอ วิธีนี้เป็นการเฉลี่ยค่าระดับสีเทาโดยใช้ฟังก์ชั่นดังกล่าว โดยจะแบ่งภาพออกได้ 3 ช่วงคือ ช่วงของข้อมูลภาพ ก อยู่ระหว่าง y_1 ถึง y_2 ช่วงของข้อมูลภาพซ้อนทับกันอยู่ระหว่าง y_2 ถึง y_3 และช่วงของข้อมูลภาพ ข อยู่ระหว่าง y_3 ถึง y_4 กำหนดให้ช่วงของข้อมูลภาพที่ไม่ซ้อนทับกันฟังก์ชั่นทั้งสองมีค่าเท่ากับ 1 เสมอ ส่วนช่วงที่ข้อมูลภาพซ้อนทับกันกำหนดให้ฟังก์ชั่นทั้งสองมีค่าลดลงและเพิ่มขึ้น ซึ่งมีผลรวมเท่ากับ 1 $[f_a(y) + f_b(y) = 1]$ ดังแสดงรายละเอียดในรูปที่ 2.4 (ข)

วิธีการดังกล่าวสามารถเขียนเป็นสมการคณิตศาสตร์ได้ดังสมการที่ (2.6)

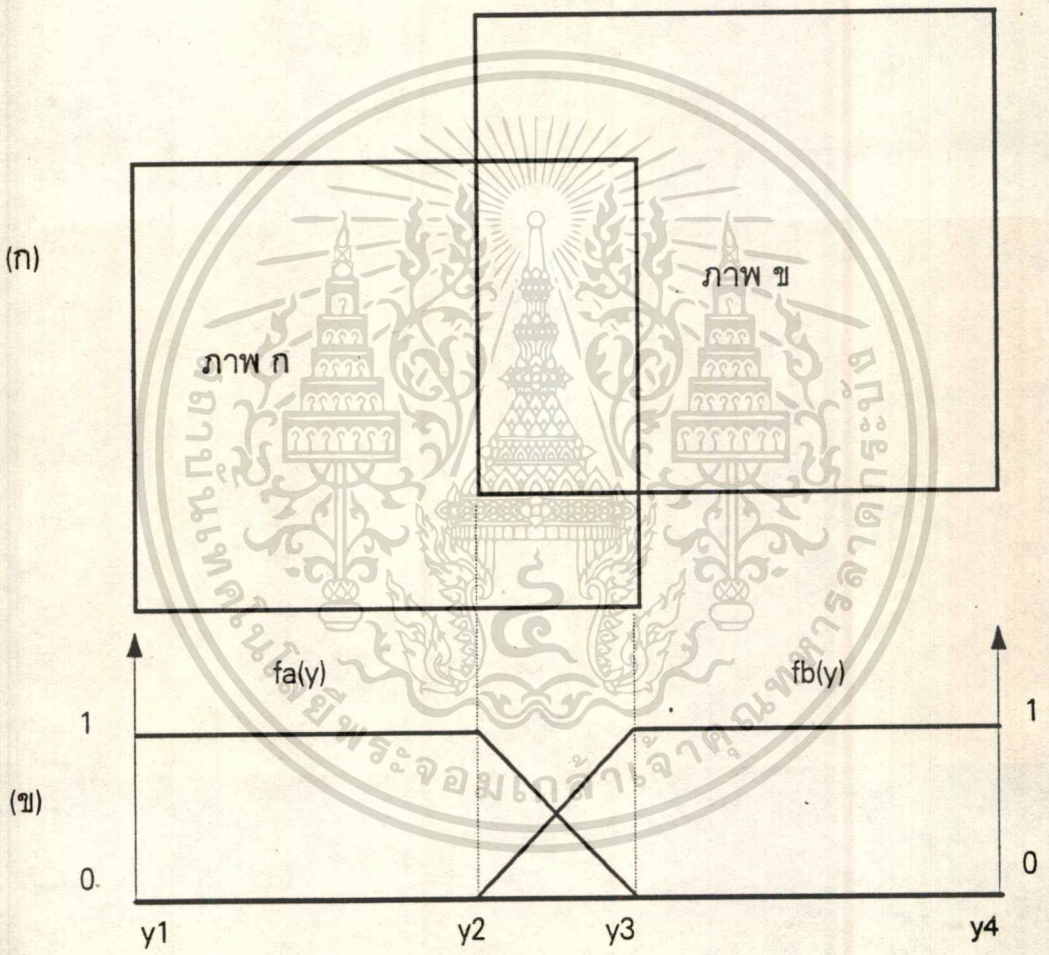
$$g(x,y)=\begin{cases} g_a(x,y) & ;y_1 < y \le y_2 \\ f_a(y)g_a(x,y)+f_b(y)g_b(x,y) & ;y_2 < y \le y_3 \\ g_b(x,y) & ;y_3 < y \le y_4 \end{cases} \quad (2.6)$$

เมื่อ $g(x,y)$ คือ ค่าระดับสีเทาของข้อมูลที่ได้จากการเฉลี่ย

$g_a(x,y)$ คือ ค่าระดับสีเทาของข้อมูลภาพ ก

$g_b(x,y)$ คือ ค่าระดับสีเทาของข้อมูลภาพ ข

$f_a(y), f_b(y)$ คือ ฟังก์ชั่นน้ำหนักแบบเชิงเส้น สำหรับภาพ ก และภาพ ข ตามลำดับ



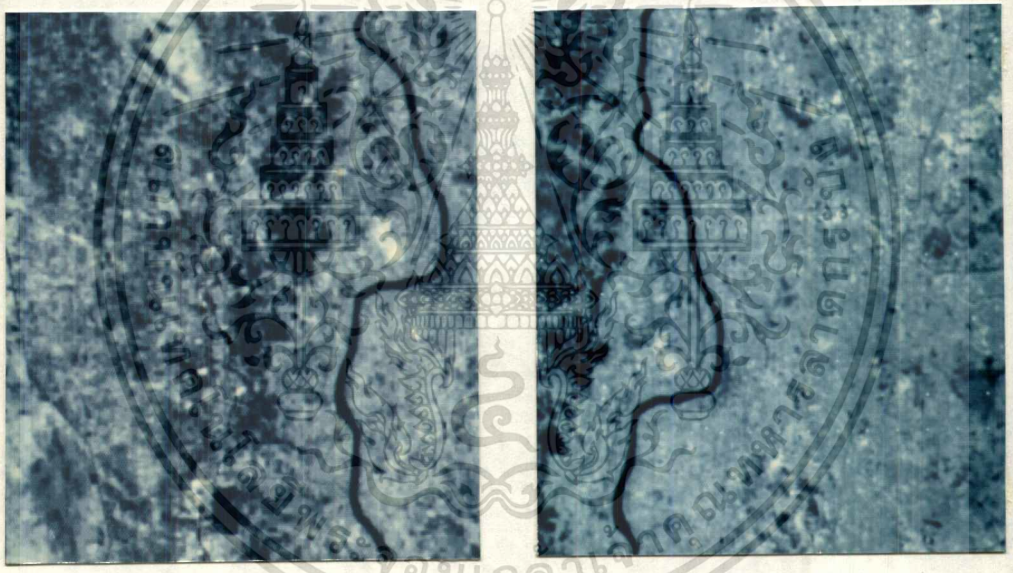
รูปที่ 2.4 ก) ลักษณะการวางซ้อนทับของข้อมูลภาพ ก และ ข

ข) ฟังก์ชั่นน้ำหนักแบบเชิงเส้น $f_a(y)$ และ $f_b(y)$ สำหรับภาพ ก และ ข

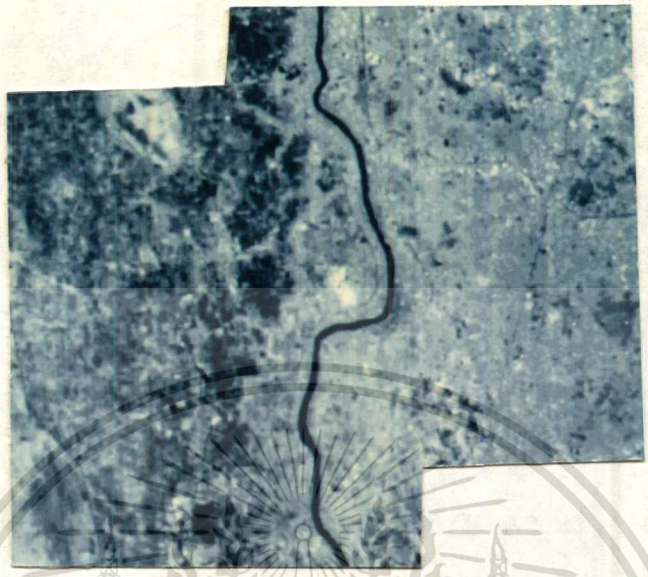
จากวิธีการต่อภาพซึ่งได้กล่าวมานี้สังเกตุจะพบว่าค่าความสว่างของข้อมูลภาพบริเวณที่ไม่ใช่ส่วนที่ซ้อนทับกันจะยังคงมีค่าระดับสีเทาเหมือนเดิมไม่เปลี่ยนแปลง เฉพาะบริเวณข้อมูลภาพที่ซ้อนทับกันเท่านั้นจะนำมาทำการเฉลี่ยให้ค่าความสว่างของข้อมูลภาพปรับเข้าหากัน

2.5 ผลการทดลอง

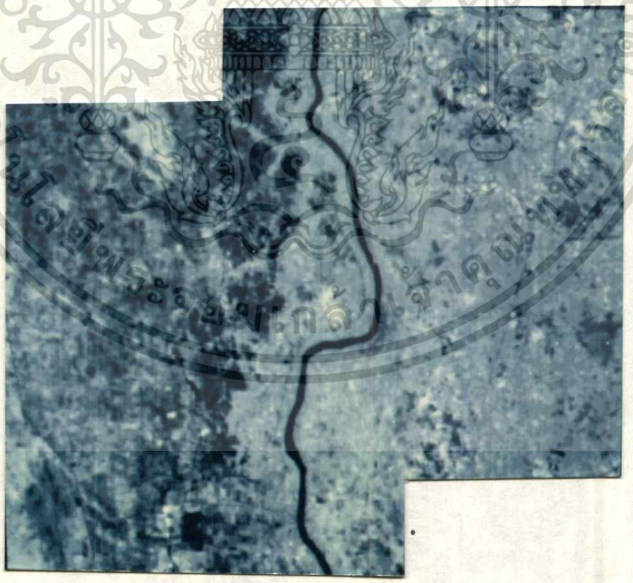
วิธีการต่อภาพโดยวิธีการเฉลี่ยค่าระดับสีเทาในหัวข้อที่ 2.4 ทำการทดลองกับข้อมูลภาพของดาวเทียม MOS-I ระบบ MESSR จำนวน 2 ภาพดังรูปที่ 2.5 (ก) มาทำการหาตำแหน่งซ้อนทับและปรับความสว่างของภาพให้สอดคล้องกัน จากนั้นเชื่อมต่อภาพจะได้ดังรูปที่ 2.5 (ข) วิธีการเฉลี่ยค่าระดับสีเทาบริเวณซ้อนทับจะได้ข้อมูลใหม่ดังรูปที่ 2.5 (ค)



(ก) ข้อมูลก่อนต่อภาพ



(ข) ข้อมูลหลังจากผ่านการต่อภาพ



(ค) ข้อมูลภาพหลังจากผ่านการเฉลี่ยค่าระดับสีเทา

รูปที่ 2.5 ผลลัพธ์การต่อภาพ

2.6 สรุป

อย่างไรก็ตามวิธีการต่อภาพโดยวิธีการเฉลี่ยค่าระดับสีเทายังมีขีดจำกัดในการประยุกต์ใช้งาน คือ ภาพที่ต้องการนำมาเชื่อมต่อกันจะต้องมีค่าความสว่างที่มีลักษณะใกล้เคียงกัน มิฉะนั้นแล้วภาพใหม่ที่ได้หลังผ่านการต่อจะทำให้เกิดเป็นความสว่างใหม่ 3 บริเวณคือ ความสว่างของภาพเดิมทั้งสองและความสว่างใหม่บริเวณที่ได้ทำการฉลีย ดังนั้นช่วงบริเวณรอยต่อระหว่างความสว่างที่ได้จากการเฉลี่ยกับความสว่างของภาพเดิมทั้งสองยังคงเห็นเป็นรอยตะเข็บ ทำให้การเชื่อมต่อภาพไม่ได้ภาพใหม่ที่มีค่าสว่างต่อเนื่องตลอดทั้งภาพ วิธีการนี้ยังมีข้อเสียอีกประการคือ บริเวณซ้อนทับภาพทั้งสองจะสูญเสียความคมชัด (Blur) ทั้งนี้เป็นไปตามคุณลักษณะในการเฉลี่ยค่าเช่น ถ้าภาพคนละวันภาพหนึ่งแม่น้ำชัดอีกภาพไม่ชัดมาเฉลี่ยทำให้ภาพไม่ชัด

บทที่ 3

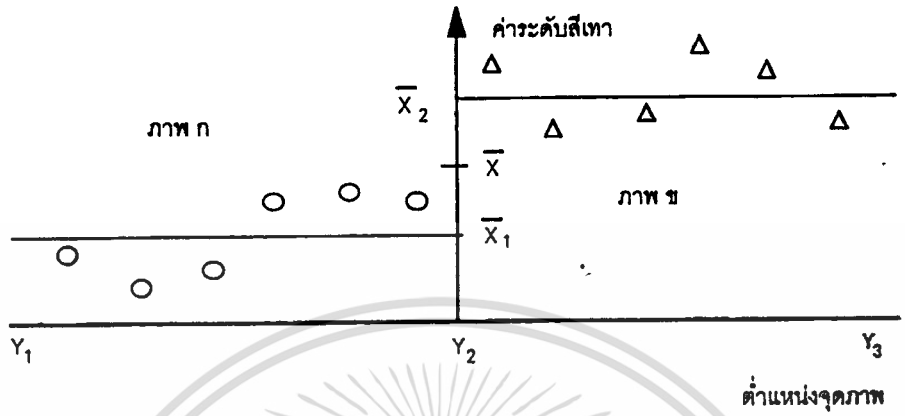
การเชื่อมต่อภาพโดยใช้ฟังก์ชั่นเชิงเส้น

3.1 คำนำ

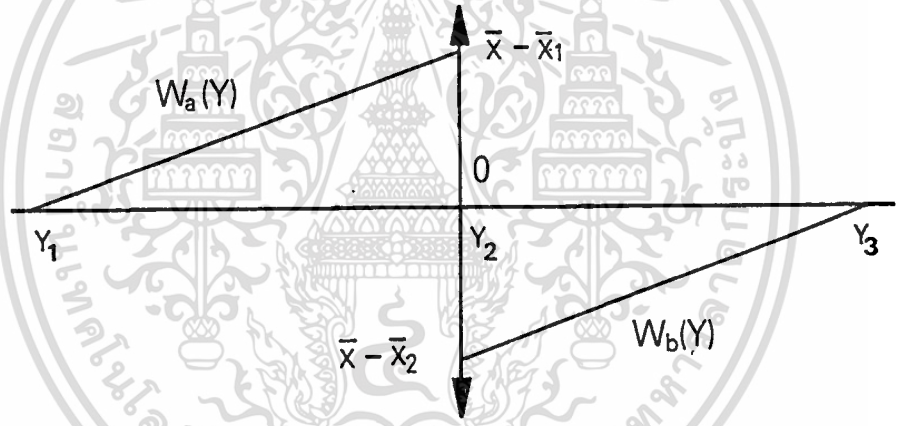
วิธีการต่อภาพโดยใช้ฟังก์ชั่นเชิงเส้นเป็นวิธีการหนึ่งสามารถนำมาช่วยแก้ไขปัญหาที่เกิดจากวิธีการแบบดั้งเดิมในหัวข้อ 2.4 เป็นวิธีการเฉลี่ย ซึ่งไม่อาจกระทำการต่อภาพบางกรณีให้ได้ความสว่างของภาพต่อเนื่องตลอดทั้งภาพ การเชื่อมต่อภาพโดยใช้ฟังก์ชั่นเชิงเส้นมีลำดับขั้นตอนหลักๆ คือ คำนวณหาตำแหน่งบริเวณซ้อนทับของภาพเช่นเดียวกับหัวข้อที่ 2.2 จากนั้นทำการปรับความสว่างของภาพให้สอดคล้องกัน (Brightness matching) และสุดท้ายทำการไบอัสตรงบริเวณรอยตะเข็บด้วยฟังก์ชั่นเชิงเส้น ซึ่งรายละเอียดของขั้นตอนต่าง ๆ จะกล่าวในหัวข้อต่อไปดังนี้

3.2 วิธีการลบรอยตะเข็บโดยใช้ฟังก์ชั่นเชิงเส้น

หลังจากขั้นตอนการหาตำแหน่งซ้อนทับเรียบร้อยแล้ว พื้นที่ส่วนที่ซ้อนทับในภาพหนึ่งจะถูกตัดทิ้ง จากนั้นจึงต่อภาพทั้งสองเข้าด้วยกัน ถึงแม้ว่าภาพหนึ่งที่จะนำมาต่อนั้นได้ผ่านกระบวนการปรับความสว่างในภาพแล้วก็ตาม หลังการต่อภาพจะยังคงปรากฏตะเข็บรอยต่อให้เห็นดังแสดงในรูปที่ 2.5 (ข) ซึ่งรอยตะเข็บควรจะถูกกำจัดทิ้งไปเพื่อช่วยให้การตีความหมายภาพ (โดยเฉพาะภาพสีแบบ false colour) สามารถกระทำได้อย่างมีประสิทธิภาพ วิธีการกำจัดรอยตะเข็บที่เสนอในบทความ [18] เป็นการใช้การไบอัส (bias) ค่าความสว่างบริเวณรอยตะเข็บด้วยฟังก์ชั่นเชิงเส้นดังแสดงในรูปที่ 3.1 โดยรูปที่ 3.1(ก) เป็นการพิจารณาถึงแต่ละเส้นภาพที่นำมาต่อเชื่อมเข้าด้วยกัน โดยจะปรากฏว่าค่าเฉลี่ยของบริเวณรอยต่อทั้งสองด้านคือจากตำแหน่งจุดภาพ y_1 ถึง y_2 ในภาพ ก และจากตำแหน่งจุดภาพ y_2 ถึง y_3 ในภาพ ข จะไม่เท่ากัน ดังนั้นจึงต้องทำการปรับค่าความสว่างรอบๆบริเวณรอยต่อของแต่ละเส้นภาพให้มีค่าใกล้เคียงกันโดยใช้ฟังก์ชั่นเชิงเส้นในรูปที่ 3.1 (ข) เป็นตัวไบอัส



(ก) ค่าระดับสีเทาของภาพ ก ที่มาต่อกับภาพ ข



(ข) ฟังก์ชั่นเชิงเส้นที่ใช้ในการไบอัส

รูปที่ 3.1 การไบอัสด้วยฟังก์ชั่นเชิงเส้น

วิธีการไบอัสนี้จะทำการประมวลผลที่ละเส้นภาพ สมมติว่าการนำภาพ ก มาต่อกับภาพ ข จะมีขั้นตอนดังนี้

ขั้นที่ 1 คำนวณค่าเฉลี่ยของจุดภาพ $(\overline{X}_1)$ ของเส้นภาพที่ x ในภาพรวม โดยคิดจากจุดภาพในตำแหน่ง y_1 ถึงตำแหน่งรอยต่อ y_2 ในส่วนของภาพ ก ดังแสดงในรูปที่ 3.1 (ก)

ในทำนองเดียวกันคำนวณค่าเฉลี่ยของจุดภาพ $(\overline{X}_2)$ ของเส้นภาพที่ x ในภาพรวม โดยคิดจากจุดภาพตำแหน่งรอยต่อ $y_2 + 1$ ถึงตำแหน่ง y_3 ในส่วนของภาพ ข ดังแสดงในรูปที่ 3.1 (ก)

ขั้นที่ 2 คำนวณค่าเฉลี่ย $(\overline{X})$ ของค่าเฉลี่ยทั้งสอง

$$\overline{X} = \frac{\overline{X}_1 + \overline{X}_2}{2} \tag{3.1}$$

ขั้นที่ 3 สร้างฟังก์ชั่นเชิงเส้น $W_a(x,y)$ เมื่อ $y_1 < y \le y_2$ ในส่วนของภาพ ก โดยที่

$$W_a(x,y) = \frac{(\overline{X} - \overline{X}_1)}{(y_2 - y_1)} y - \frac{(\overline{X} - \overline{X}_1) y_1}{(y_2 - y_1)} \qquad ; y_1 < y \le y_2 \tag{3.2}$$

ฟังก์ชั่น $W_a(y)$ นี้จะเป็นเชิงเส้นดังแสดงในรูปที่ 3.1 (ข) ดังนั้นค่าของ $W_a(y)$ ที่ y แปรจาก y_1 ถึง y_2 จะอยู่ในลักษณะของค่าจากสมการเส้นตรงนั้นเอง

ในทำนองเดียวกันหาฟังก์ชั่นเชิงเส้น $W_b(y)$ เมื่อ $y_2 < y \le y_3$ ในส่วนของภาพ ข โดยที่

$$W_b(x,y) = \frac{(\overline{X} - \overline{X}_2)}{(y_2 - y_3)} y - \frac{(\overline{X} - \overline{X}_2) y_3}{(y_2 - y_3)} \qquad ; y_2 < y \le y_3 \tag{3.3}$$

ขั้นที่ 4 ทำการไบอัสให้กับจุดภาพใกล้บริเวณรอยต่อในส่วนของภาพ ก ด้วยฟังก์ชั่น $W_a(y)$ และในส่วนของภาพ ข ด้วยฟังก์ชั่น $W_b(y)$ สำหรับภาพรวมหลังการเชื่อมต่อ ซึ่งจุดภาพค่าระดับสีเทา ในภาพเชื่อมต่อจะกลายเป็น

$$g(x,y) = \begin{cases} g_a(x,y) & ; y \le y_1 \\ g_a(x,y) + W_a(x,y) & ; y_1 < y \le y_2 \\ g_b(x,y) + W_b(x,y) & ; y_2 < y \le y_3 \\ g_b(x,y) & ; y_3 < y \end{cases} \tag{3.4}$$

โดยที่ $g(x,y)$ คือ ค่าระดับสีเทาที่ตำแหน่ง (x,y) ของจุดภาพใหม่หลังการเชื่อมต่อ

$g_a(x,y)$ และ $g_b(x,y)$ เป็นค่าระดับสีเทาของจุดภาพในภาพ ก และภาพ ข ตามลำดับ

$W_a(x,y)$ และ $W_b(x,y)$ เป็นค่าฟังก์ชั่นเชิงเส้นที่นำมาใช้ในการไบอัสสำหรับภาพ ก และภาพ ข ตามลำดับ

วิธีการไบอัสเชิงเส้นนี้จะเป็นการปรับค่าความสว่างให้เข้าหากันของจุดภาพในภาพ ก และภาพ ข ณ บริเวณรอบๆรอยต่อ ภาพผลลัพธ์จากการไบอัสนี้จะช่วยขจัดปัญหาเรื่องรอยตะเข็บและจะให้ความต่อเนื่องของค่าความสว่างปรากฏในภาพหลังการเชื่อมต่อ

3.3 ผลการทดลอง

จากการนำภาพรูปที่ 2.5 (ก) มาเชื่อมต่อกัน จะได้รูปที่ 2.5 (ข) ซึ่งปราศจากการไบอัส แต่ถ้านำเอาขั้นตอนการไบอัสเชิงเส้นมาใช้กับภาพในรูปที่ 2.5 (ข) จะได้ภาพในรูปที่ 3.2 ซึ่งพบว่ารอยตะเข็บจะถูกกำจัดไป

รูปที่ 3.2 ข้อมูลภาพผ่านการลบรอยตะเข็บ

3.4 สรุป

จากภาพเชื่อมต่อผลลัพธ์ในรูปที่ 3.2 พบว่ารอยตะเข็บถูกกำจัดทิ้งไป แต่วิธีการนี้จะมีข้อบกพร่องตรงที่ถ้าหากปรากฏว่ากลุ่มจุดภาพที่อยู่ใกล้จุดรอยต่อเป็นวัตถุชนิดเดียวกัน ซึ่งเดิมจะมีค่าระดับสีเทาเท่ากัน แต่ปรากฏว่าหลังการไบอัสด้วยฟังก์ชั่นเชิงเส้น จะพบว่ากลุ่มจุดภาพของวัตถุดังกล่าวจะมีค่าระดับสีเทาไม่เท่ากันเนื่องจากความลาด (slope) ของฟังก์ชั่นเชิงเส้นนั้นเอง ทำให้วัตถุเดิมนั้นอาจถูกจำแนกเป็นวัตถุหลายๆชนิดว่างติดกัน ซึ่งมีผลทำให้การตีความภาพหรือการจำแนกภาพขาดประสิทธิภาพไป

บทที่ 4.

การต่อภาพโดยใช้ฟังก์ชั่นบัทเตอร์เวิร์ธ

4.1 คำนำ

ดังได้สรุปไว้ในบทที่ 3 แล้วว่าการต่อภาพโดยใช้สมการเชิงเส้นในการไบอัส สามารถกำจัดรอยตะเข็บในการต่อภาพได้ แต่จะมีข้อเสียคือ ฟังก์ชั่นที่ใช้ในการไบอัสมีผลทำให้ค่าระดับสีเทาของกลุ่มจุดภาพเนื้อเดียวกันเปลี่ยนค่าไป การแก้ไขปัญหาดังกล่าวสามารถทำได้ด้วยการใช้ฟังก์ชั่นบัทเตอร์เวิร์ธในการไบอัสเพื่อลบรอยตะเข็บ ในขณะเดียวกันก็จะพยายามรักษาค่าระดับสีเทาของกลุ่มจุดภาพในวัตถุเนื้อเดียวกันที่อยู่ใกล้บริเวณรอยต่อจะยังคงมีค่าเท่ากัน ซึ่งวิธีการนี้จะนำไปประมวลผลกับภาพถ่ายดาวเทียมสามแบนด์จากภาพสองกลุ่มที่นำมาต่อเชื่อมในลักษณะของภาพสี

4.2 ฟังก์ชั่นบัทเตอร์เวิร์ธ

ในการรักษาให้กลุ่มจุดภาพของวัตถุใกล้บริเวณรอยต่อยังคงมีค่าระดับสีเทาเท่ากันทำได้ถ้าหากฟังก์ชั่นในการไบอัสมีค่าคงที่บริเวณใกล้ๆรอยต่อแล้วค่อยๆลาดลงถ้าระยะห่างจากรอยต่อเพิ่มขึ้น พบว่าฟังก์ชั่นที่มีคุณสมบัติดังกล่าวนี้คือ ฟังก์ชั่นบัทเตอร์เวิร์ธ [18] โดยสมการทางคณิตศาสตร์ของบัทเตอร์เวิร์ธคือ

$$B(x,y)=\frac{1}{1+0.414\left(\frac{y}{y_0}\right)^n} \qquad ;0\le y\le y_e \qquad (4.1)$$

เมื่อ n เป็นอันดับความราบเรียบ y_0 เป็นค่าความถี่ตัด ซึ่งเป็นตัวเปลี่ยนแปลงความลาดชันของฟังก์ชั่นนั้นเอง y_e เป็นตำแหน่งจุดปลายของกราฟบัทเตอร์เวิร์ธและ y เป็นตำแหน่งใดๆของจุดภาพ

4.3 วิธีการลบรอยตะเข็บโดยใช้ฟังก์ชั่นบัทเตอร์เวิร์ธ

ขั้นตอนในการลบรอยตะเข็บโดยใช้ฟังก์ชั่นบัทเตอร์เวิร์ธ จะคล้ายกับการใช้ฟังก์ชั่นเชิงเส้น จากภาพหลังการคำนวณหาตำแหน่งซ้อนทับและทำการปรับค่าความสว่างทั้งสองแล้ว ให้ตัด

พื้นที่ซ้อนทับของภาพใดภาพหนึ่งทิ้งไปแล้วนำภาพทั้งสองมาเชื่อมต่อเข้าด้วยกัน สมมติว่าเส้นภาพที่ x จากการเชื่อมต่อภาพ ก และภาพ ข เข้าด้วยกันมีค่าระดับสีเทาของจุดภาพใกล้ๆรอยต่อ ดังแสดงในรูปที่ 4.1 (ก) ดังนั้นการลบรอยต่อโดยใช้ฟังก์ชั่นบัทเตอร์เวิร์ธ มีขั้นตอนดังนี้

ขั้นที่ 1 คำนวณค่าเฉลี่ย $(\overline{X}_1)$ จากตำแหน่งจุดภาพที่ y_1 ถึงจุดภาพ ณ ตำแหน่งรอยต่อ y_2 ในส่วนของภาพ ก ดังแสดงในรูปที่ 4.1 (ก)

ในทำนองเดียวกัน ทำการคำนวณค่าเฉลี่ย $(\overline{X}_2)$ จากตำแหน่งรอยต่อ y_2+1 ถึงตำแหน่งจุดภาพ y_3 ในส่วนของภาพ ข ดังแสดงในรูปที่ 4.1 (ก)

ขั้นที่ 2 คำนวณค่าเฉลี่ยของค่าเฉลี่ยทั้งสอง

$$\overline{X} = \frac{\overline{X}_1 + \overline{X}_2}{2} \tag{4.2}$$

ขั้นที่ 3 สร้างฟังก์ชั่นบัทเตอร์เวิร์ธ $B_a(x,y)$ ในส่วนของภาพ ก โดยที่

$$B_a(x,y) = (\overline{X} - \overline{X}_1)B(x, y_2 - y) \qquad ;y_1 < y \le y_2 \tag{4.3}$$

ในทำนองเดียวกันสร้างฟังก์ชั่นบัทเตอร์เวิร์ธ $B_b(y)$ ในส่วนของภาพ ข โดยที่

$$B_b(x,y) = (\overline{X} - \overline{X}_2)B(x, y - y_2) \qquad ;y_2 < y \le y_3 \tag{4.4}$$

ขั้นที่ 4 ทำการไบอัสให้กับจุดภาพใกล้บริเวณรอยต่อ ในส่วนของภาพ ก ด้วยฟังก์ชั่น $B_a(x,y)$ และในส่วนของภาพ ข ด้วยฟังก์ชั่น $B_b(x,y)$ สำหรับภาพรวมหลังการเชื่อมต่อ ซึ่งค่าระดับสีเทาในภาพเชื่อมต่อ จะกลายเป็น

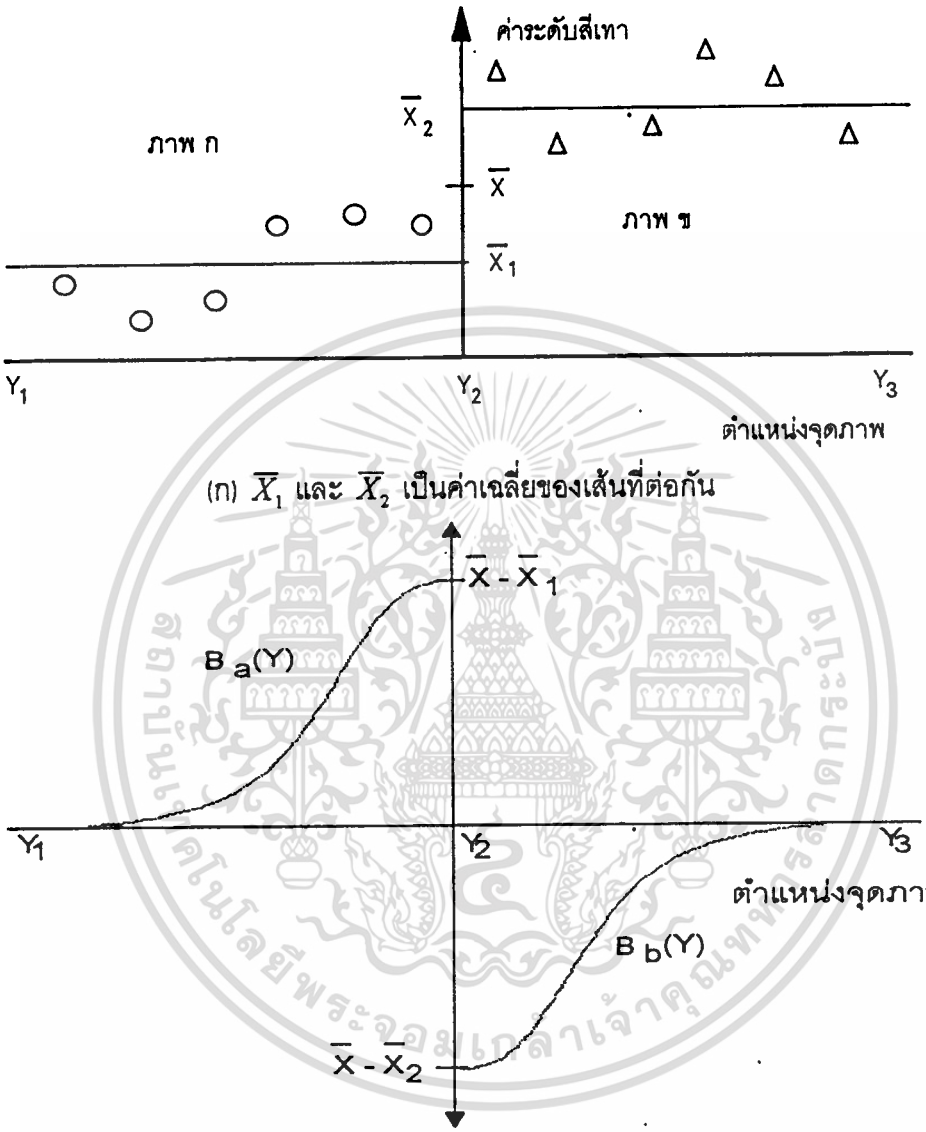
$$g(x,y) = \begin{cases} g_a(x,y) & ;y \le y_1 \\ g_a(x,y) + B_a(x,y) & ;y_1 < y \le y_2 \\ g_b(x,y) + B_b(x,y) & ;y_2 < y \le y_3 \\ g_b(x,y) & ;y_3 < y \end{cases} \tag{4.4}$$

โดยที่ $g(x,y)$ คือค่าระดับสีเทาที่ตำแหน่ง (x,y) ของจุดภาพใหม่หลังการเชื่อมต่อ

$g_a(x,y)$ และ $g_b(x,y)$ เป็นค่าระดับสีเทาของจุดภาพในภาพ ก และ ข ตามลำดับ

$B_a(x,y)$ และ $B_b(x,y)$ เป็นค่าฟังก์ชั่นบัทเตอร์เวิร์ธที่นำมาใช้ในการไบอัสสำหรับภาพ ก และภาพ ข ตามลำดับ

วิธีการไบอัสด้วยฟังก์ชั่นบัทเตอร์เวิอร์ธเป็นการปรับค่าความสว่างให้เข้าหากันระหว่างจุดภาพในภาพ ก และ ภาพ ข ณ บริเวณรอบๆรอยต่อ ซึ่งจะช่วยขจัดปัญหาเรื่องรอยตะเข็บและให้ความสว่างของภาพเชื่อมต่อมีความต่อเนื่อง



(ก) $\overline{X}_1$ และ $\overline{X}_2$ เป็นค่าเฉลี่ยของเส้นที่ต่อกัน

(ข) การไบอัสด้วยฟังก์ชั่นไม่เป็นเชิงเส้นแบบบัทเตอร์เวิอร์ธ

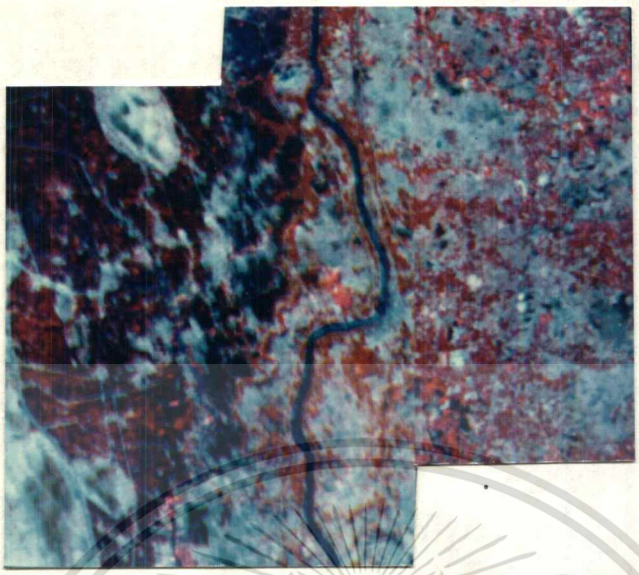
รูปที่ 4.1 การใช้ฟังก์ชั่นไม่เป็นเชิงเส้นแบบบัทเตอร์เวิอร์ธกับข้อมูลภาพ

4.4 ผลการทดลอง

จากการนำภาพรูปที่ 2.5 (ข) มาผ่านกระบวนการที่กล่าวไว้ในหัวข้อ 4.3 ซึ่งเป็นการลบรอยตะเข็บด้วยการใช้ฟังก์ชั่นบัทเตอร์เวิร์ธในการไบอัส จะได้ภาพเชื่อมต่อผลลัพธ์ดังแสดงในรูปที่ 4.2

ถ้าหากว่านำภาพถ่ายดาวเทียมจำนวนสามแบนด์จำนวนสองภาพที่ถ่ายในเวลาต่างกันมาเชื่อมต่อเข้าด้วยกัน อย่างเช่น ภาพถ่ายดาวเทียม MOS-I ระบบ MESSR โดยใช้ภาพแบนด์ที่ 4,2 และ 1 ภายหลังการต่อภาพตามกระบวนการที่กล่าวไว้ในหัวข้อ 4.3 โดยกระทำที่ละแบนด์ จากนั้นทำการกำหนดสีให้แบนด์ที่ 4,2 และ 1 เป็นสีแดง สีเขียว และสีน้ำเงิน ตามลำดับ เมื่อนำภาพเชื่อมต่อมาผสมสีกันจะได้ภาพ False colour ดังแสดงในรูปที่ 4.3

รูปที่ 4.2 การเชื่อมต่อภาพขาว-ดำโดยปราศจากรอยตะเข็บ



รูปที่ 4.3 การเชื่อมต่อภาพสีที่ปราศจากรอยตะเข็บของข้อมูลภาพระบบ MESSR ของดาวเทียม MOS-I

4.5 สรุป

จากวิธีการการกำจัดรอยตะเข็บตรงบริเวณรอยต่อสำหรับการเชื่อมต่อภาพสีของภาพถ่ายดาวเทียมที่ได้นำเสนอนั้น จะพบว่าภาพผลลัพธ์จะปราศจากรอยตะเข็บทำให้ผู้ใช้ข้อมูลภาพถ่ายดาวเทียมภาพสีสามารถนำภาพมาต่อขยายให้ได้พื้นที่บริเวณกว้างๆเพื่อการแปลความหมายภาพสีจะสามารถกระทำได้ครอบคลุมพื้นที่กว้างๆ ดังนั้นการแปลความหมายของวัตถุชนิดเดียวกันแต่อยู่คนละภาพจะไม่เกิดความผิดพลาดคลาดเคลื่อนเลย ซึ่งจะส่งผลต่อการเพิ่มประสิทธิภาพในการแปลความหมายของภาพให้สูงขึ้น

บทที่ 5

การตรวจสอบการเปลี่ยนแปลง

5.1 คำนำ

ในการตรวจสอบการเปลี่ยนแปลงบนภาคพื้นดินที่ครอบคลุมพื้นที่กว้างๆนั้นจำเป็นต้องอาศัยภาพถ่ายดาวเทียมที่ครอบคลุมพื้นที่เดียวกัน แต่ถ่ายบันทึกไว้คนละวัน แล้วนำภาพคู่กรณีนั้นมาเปรียบเทียบ แต่เนื่องจากภาพถ่ายดาวเทียมประกอบด้วยหลายแบนด์ ทั้งนี้เพราะวัตถุต่างชนิดบนพื้นดินให้อัตราการสะท้อนของคลื่นแม่เหล็กไฟฟ้าได้ดีที่แถบความยาวคลื่นแตกต่างกันออกไป ดังนั้นถ้าต้องการความถูกต้องในการตรวจสอบการเปลี่ยนแปลงที่เกิดขึ้นจากภาพถ่ายดาวเทียมทั้งสองวันจึงต้องทำการเปรียบเทียบแบบแบนด์ต่อแบนด์ ทำให้ยากต่อการรวบรวมข้อมูลที่มีการเปลี่ยนแปลงจากแถบความยาวคลื่นทั้งหมดของภาพถ่ายดาวเทียม ดังนั้นในวิทยานิพนธ์นี้ จึงเสนอวิธีการตรวจสอบการเปลี่ยนแปลงบนภาคพื้นดิน ด้วยการใช้เทคนิคการวิเคราะห์แบบพหุตัวแปรจากข้อมูลภาพถ่ายดาวเทียมต่างเวลา เทคนิคการวิเคราะห์แบบพหุตัวแปรที่ใช้นี้รู้จักกันในชื่อของ **การวิเคราะห์องค์ประกอบหลัก** (Principal Component Analysis) เทคนิคดังกล่าวจะทำการปรับข้อมูลจากแถบความยาวคลื่นต่างๆให้อยู่ในฐานเดียวกันเสียก่อน ด้วยการคำนวณหา eigenvector ที่ได้จากแมทริกซ์ covariance ของข้อมูลภาพถ่ายดาวเทียมทุกแถบความยาวคลื่นทั้งสองวัน จะพบว่าที่อันดับสูงๆของ eigenvector จะให้สมาชิกของเวคเตอร์ที่ได้จากภาพถ่ายดาวเทียมคนละวันมีเครื่องหมายตรงกันข้ามกัน ซึ่งถ้าทำการโปรเจคชั่นข้อมูลภาพเดิมลงบน eigenvector ดังกล่าวก็จะได้ภาพความแตกต่างที่เกิดขึ้นในทุกแถบความยาวคลื่นของภาพถ่ายดาวเทียมทั้งสองวัน ภาพดังกล่าวจะถูกนำมาใช้ในการตรวจหาการเปลี่ยนแปลงต่างๆ ของทรัพยากรบนภาคพื้นดิน

5.2 ทฤษฎีการวิเคราะห์องค์ประกอบหลัก

การวิเคราะห์องค์ประกอบหลักเป็นวิธีการวิเคราะห์แบบ linear combination ของข้อมูลภาพที่จะรักษาไว้ซึ่งค่าการเปลี่ยนแปลงหรือความแปรปรวน (variance) ของแกนเดิม (Original axis) แกนใหม่จะถูกสร้างขึ้นมาโดยจะได้มีการจัดสร้างความสัมพันธ์ของข้อมูลภาพจากแกนเดิมต่างๆการวิเคราะห์องค์ประกอบหลักของตัวแปร X จะเป็นการแปลงเชิงเส้น (Linear transformation) ของค่าความแปร

ปรวนจากข้อมูลเดิมสู่ตัวแปรใหม่ Y โดยตัวแปรใหม่ Y นั้นจะไม่มีค่าสหสัมพันธ์ถึงแม้ว่าข้อมูลในตัวแปร X จะมีค่าสหสัมพันธ์สูงก็ตาม การวิเคราะห์องค์ประกอบหลักได้ถูกนำมาใช้เป็นเวลานานหลายปีแล้ว คณิตศาสตร์ของการวิเคราะห์องค์ประกอบหลักนี้ได้ถูกเขียนขึ้นในปี ค.ศ 1923 โดย Hoteling และในปี ค.ศ 1964 โดย Searle [18] การวิเคราะห์องค์ประกอบหลักนี้เป็นเทคนิคทางสถิติที่อยู่บนพื้นฐานของ variance และ covariance ของกลุ่มของข้อมูล ค่า variance นี้เป็นการแตกกระจาย (Scatter) ที่ปรากฏอยู่ภายในหนึ่งตัวแปรของกลุ่มข้อมูล ส่วน covariance เป็นการวัดการแตกกระจายในระหว่างกลุ่มตัวแปร

การวิเคราะห์องค์ประกอบหลักของตัวแปร X ทั้งหมด n ตัว เป็นการกำหนดการแปลงเชิงเส้นของทุกๆการแปรเปลี่ยนในตัวแปรเดิมสู่ตัวแปรใหม่ Y ทั้งหมด n ตัว ในการแปลงนี้จะกำหนดให้ตัวแปรลำดับแรก (คือ องค์ประกอบแรก) ของกลุ่มตัวแปร Y มีค่าความแปรปรวนสูงสุดจากความแปรปรวนทั้งหมด ส่วนตัวแปรลำดับที่สองของกลุ่มของตัวแปร Y จะมีความแปรปรวนสูงสุดจากความแปรปรวนที่เหลือและไล่ลงไปเรื่อยๆสำหรับตัวแปรถัดไป จุดประสงค์หลักของทฤษฎีในการแปลงก็คือ การได้คืนมาของการแปรเปลี่ยนทั้งหมดเพื่อที่จะรักษาเอาไว้ซึ่งการแปรเปลี่ยนทุกๆ อย่างที่ต้องการ ดังนั้นจำนวนการแปรเปลี่ยนที่เกิดขึ้นเพียงเล็กน้อย ในตัวแปรหลังๆนั้นจะมีค่าความแปรปรวนน้อยจนอาจจะพิจารณาได้ว่ามีค่าน้อยมากและสามารถตัดทิ้งได้ ถ้าหากมีการตัดเอาตัวแปรหลังๆดังกล่าวทิ้งไปก็จะเป็นการลดขนาดมิติของภาพ ซึ่งจะทำให้ค่าความแปรปรวนทั้งหมดของข้อมูลหลังผ่านการแปลงมีค่าน้อยกว่าค่าความแปรปรวนของข้อมูลเดิม

ตัวแปรที่ได้หลังผ่านการแปลงคือกลุ่มตัวแปร Y ได้จากการวิเคราะห์องค์ประกอบหลักจะไม่ให้ค่าสหสัมพันธ์ ซึ่งเป็นคุณสมบัติอย่างหนึ่งของการแปลงเรียกว่า orthogonality หรือการขาดหายไปของค่าสหสัมพันธ์ของแกนภายหลังจากการแปลงเกิดจากผลของการแปลงนั่นเอง แม้ว่าในกลุ่มตัวแปร X เดิมจะมีค่าสหสัมพันธ์สูงก็ตาม โดยปกติแล้วจำนวนตัวแปรหลังจากการแปลงจะน้อยลงเมื่อตัวแปร X เดิมมีค่าสหสัมพันธ์สูงขึ้น ข้อมูลที่มีค่าสหสัมพันธ์สูงเป็นข้อมูลสำหรับการใช้ค่าของตัวแปรหนึ่งในการทำนายค่าที่สอดคล้องในตัวแปรอื่นๆ ส่วนเกินของข้อมูลดังกล่าวนี้จะไม่ปรากฏในกลุ่มตัวแปรที่ขาดความสัมพันธ์กัน ดังนั้นข้อมูลที่มีความสัมพันธ์กันสามารถแสดงได้ด้วย กลุ่มของตัวแปรไม่สัมพันธ์กันได้กระชับกว่ากลุ่มตัวแปรที่สัมพันธ์กัน

การวิเคราะห์องค์ประกอบหลักเป็นการแปลงเชิงเส้นคือ

$$Y = CX + B \tag{5.1}$$

เมื่อ

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_q \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_p \end{bmatrix}$$

$$C = \begin{bmatrix} c_{11} & c_{12} & . & . & c_{1p} \\ c_{21} & c_{22} & . & . & c_{2p} \\ \vdots & & & & \\ . & & & & \\ c_{q1} & c_{q2} & . & . & c_{qp} \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_q \end{bmatrix}$$

โดยที่ $q \le p$ มีเงื่อนไขกำหนดว่าการแปลงนั้นต้องได้ซึ่ง geometric orthogonality นั่นคือ $CC^t = I$ โดย C^t เป็นทรานสโพสของ C และ I เป็นแมทริกซ์เอกลักษณ์และที่เพิ่มขึ้นมาคือ ตัวแปรหลังการแปลง Y ต้องตั้งฉากกันนั้นคือ ไม่มีความสัมพันธ์กัน การกำหนดเงื่อนไขนี้ให้พิจารณาจาก

$$E(X) = m_x \text{ , เมื่อ } m_x \text{ เป็นเวคเตอร์ของค่าเฉลี่ยของตัวแปร } X$$

จะได้ว่า

$$\begin{aligned} E(Y) &= E(CX + B) \\ &= CE(X) + B \\ &= Cm_x + B = m_y \end{aligned} \tag{5.2}$$

เนื่องจากไม่มีข้อกำหนดจำกัดในการเลื่อนของ Y ไปยังจุดกำเนิดใหม่ ดังนั้นจึงสามารถเลือกให้ $m_y = 0$

จาก (5.2) จะกลายเป็น

$$Cm_x + B = 0$$

หรือ

$$-Cm_x = B \tag{5.3}$$

แทน (5.3) ลงใน (5.1) จะได้ว่า

$$\begin{aligned} Y &= CX + (-Cm_x) \\ Y &= C(X - m_x) \end{aligned} \tag{5.4}$$

ปัญหาคือ ให้คำนวณหา C ภายใต้เงื่อนไขของ $CC^t = I$

เนื่องจาก $m_x = 0$ ดังนั้นแมทริกซ์ covariance Y หาได้จาก

$$\begin{aligned} E(YY^t) &= E\{[C(X - m_x)][C(X - m_x)]^t\} \\ &= CE[(X - m_x)(X - m_x)^t]C^t \\ &= C\Sigma C^t \end{aligned} \tag{5.5}$$

เมื่อ Σ เป็นแมทริกซ์ covariance ขนาด p*p ของตัวแปร X

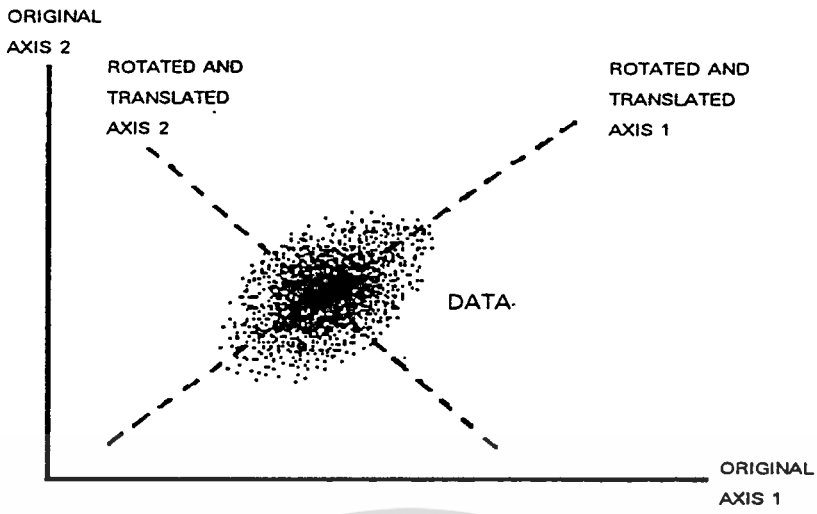
ในการบังคับให้ได้เงื่อนไขที่ว่าตัวแปร Y ไม่มีความสัมพันธ์กัน ดังนั้นความสัมพันธ์ที่ได้มาคือ

$$C\Sigma C^t = \Lambda \tag{5.6}$$

Λ คือ แมทริกซ์ covariance ของตัวแปร Y ซึ่งเป็นแมทริกซ์ทะแยงที่ $\lambda_{11}, \lambda_{22}, ..., \lambda_{qq}$ เป็นเทอมในแนวทะแยง โดยสมาชิกที่ไม่อยู่ในแนวทะแยงจะเป็นศูนย์ทั้งหมด ดังนั้นค่า covariance ต่างก็เป็นศูนย์ ซึ่งหมายความว่า Y ต่าง ๆ ไม่มีความสัมพันธ์กัน λ_{11} เป็นค่าความแปรปรวนของตัวแปร Y หรือองค์ประกอบหลักแรกของตัวแปร Y ส่วน λ_{22} เป็นค่าความแปรปรวนของตัวแปร Y หรือองค์ประกอบหลักที่ 2 และอื่น ๆ

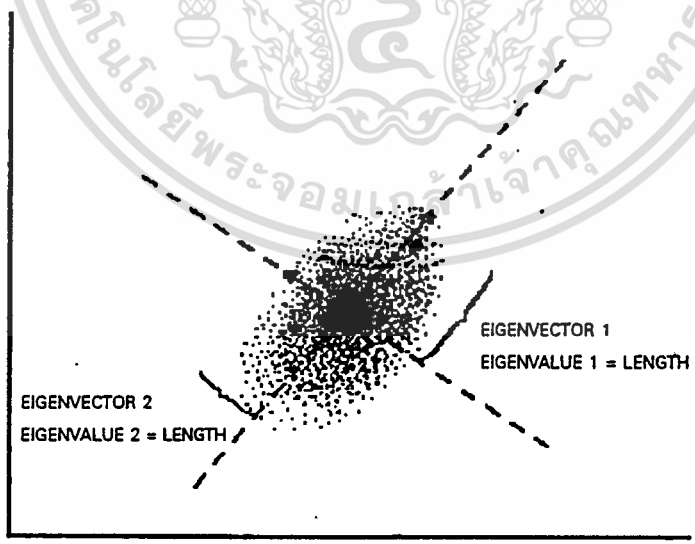
จาก (5.6) จะเห็นว่า Λ คือแมทริกซ์ eigenvalue สำหรับ Σ และ C เป็นแมทริกซ์ eigenvector ที่สอดคล้อง ดังนั้นจึงต้องทำการคำนวณค่า eigenvalue ต่าง ๆ โดยจะพบว่า $\lambda_{11}, \lambda_{22}, ..., \lambda_{qq}$ ค่าเหล่านี้ใช้แสดงแทนจำนวนค่าความแปรปรวนทั้งหมดที่อยู่บนแกนของตัวแปร Y เงื่อนไขของการวิเคราะห์องค์ประกอบหลักคือค่าความแปรปรวนทั้งหมดของกลุ่มข้อมูลเดิม (ผลบวกของสมาชิกในแนวทะแยง Σ) เท่ากับค่าความแปรปรวนทั้งหมดของข้อมูลหลังผ่านการแปลง(ผลบวกของสมาชิกในแนวทะแยง Λ)

การวิเคราะห์องค์ประกอบหลักที่นำมาใช้กับข้อมูลรีโมทเซนซิ่ง สามารถอธิบายให้เข้าใจได้ง่ายเมื่อใช้เรขาคณิต กระบวนการของการแปลงแบบเชิงเส้นเมื่อตัวแปรเดิมมีการเลื่อนและการหมุนดังรูปที่ 1 ตำแหน่งของแกนต่างๆ เมื่อเปรียบเทียบกับข้อมูลแล้วถูกเปลี่ยนไป ตำแหน่งของกลุ่มข้อมูลที่มีสัมพันธ์กับกลุ่มข้อมูลอื่นจะไม่เปลี่ยน ตัวอย่างของการวิเคราะห์องค์ประกอบหลักนั้นเปรียบเทียบได้กับหลอดที่บรรจุลูกปิงปองหลายๆลูก ถ้ามองจากปลายข้างหนึ่งของหลอดปรากฏว่าจะเห็นลูกปิงปองริมสุดเพียงลูกเดียว ถ้าพลิกดูด้านข้างของหลอดจะเห็นปิงปองทุกลูก ซึ่งก็เหมือนกับการมองภาพขององค์ประกอบหลักต่างๆ



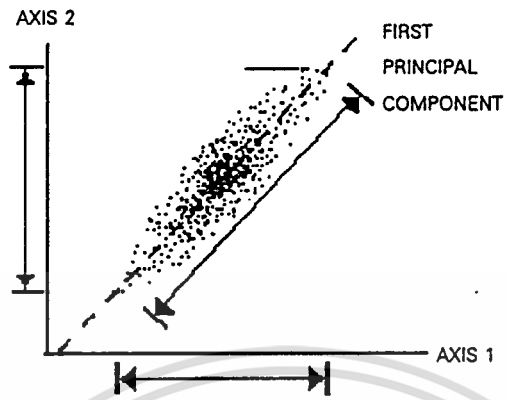
รูปที่ 5.1 การหมุนของแกนใน 2 มิติ

ค่า eigenvalue และ eigenvector มีความสัมพันธ์ดังรูปการแตกกระจายของข้อมูลในรูปที่ 2 eigenvector นั้นจะอยู่บนแกนที่แปลงไปแล้ว และความยาวของ eigenvector ต่างๆ คือ ค่า eigenvalue ดังนั้น eigenvector ที่ยาวที่สุดถูกเลือกเป็นแกนแรกของการแปลงหรือองค์ประกอบหลักลำดับที่ 1 ส่วนความยาวที่ 2 ก็ถูกเลือกเป็นองค์ประกอบหลักลำดับที่ 2 และอื่นๆถัดไป สำหรับกรณีที่มีตัวแปรเพียง 2 ตัวนั้น คู่ของ eigenvector-eigenvalue จะสอดคล้องกับแกนยาวสุดของวงรีที่ล้อมรอบข้อมูลทั้งหมด

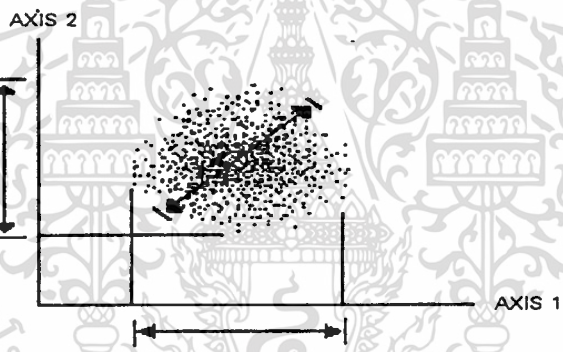


รูปที่ 5.2 Eigenvector และ eigenvalue ใน 2 มิติ

ลักษณะของสเก็ตเตอร์แกรมแสดงให้เห็นถึงค่าองศาของสหสัมพันธ์ที่ปรากฏในข้อมูล ถ้าสเก็ตเตอร์แกรมที่ยาวและแคบโดยไม่ขนานกับแกนเดิมแสดงว่าข้อมูลมีค่าสหสัมพันธ์สูงดังรูปที่ 5.3



รูปที่ 5.3 การวิเคราะห์องค์ประกอบหลักของข้อมูลที่มีค่าสหสัมพันธ์สูง การโปรเจคชั่นลงบนองค์ประกอบหลักลำดับแรกจะยาวกว่าการโปรเจคลงบนแกนเดิมทั้งสอง



รูปที่ 5.4 การวิเคราะห์องค์ประกอบหลักของข้อมูลที่มีค่าสหสัมพันธ์ต่ำ การโปรเจคชั่นลงบนองค์ประกอบหลักลำดับแรกจะเท่ากับการโปรเจคชั่นลงบนแกนเดิมทั้งสอง

สเก็ตเตอร์แกรมที่มีลักษณะเป็นวงกลมดังรูปที่ 5.4 แสดงให้เห็นถึงข้อมูลที่มีค่าสหสัมพันธ์ต่ำ องค์ประกอบหลักต่างๆของข้อมูลที่ไม่สัมพันธ์กันนี้จะไม่มีแกนใดที่ลากผ่านวงกลมแล้วได้ข้อมูลมากกว่าการโปรเจคชั่นข้อมูลลงบนแกนเดิม สมาชิกต่างๆในแมทริกซ์ C เรียกว่า **โหลดดิ้ง** ในตารางที่ 5.1 เป็นการแสดงให้เห็นถึงตัวอย่างตารางของโหลดดิ่งสำหรับ 4 ตัวแปร โดยที่คอลัมน์แรกของโหลดดิ้งในตารางที่ 5.1 แสดงให้เห็นถึงตัวแปรหลังจากการแปลงเป็นตัวแปรหนึ่งมีค่าความแปรปรวนที่ได้จากตัวแปรทั้ง 4 ของข้อมูลเดิมในจำนวนค่าเกือบเท่ากับค่าความแปรปรวนทั้งหมดของตัวแปรเดิม ในกรณีที่ค่าโหลดดิ้งเป็นลบนั้นใช้แสดงให้เห็นถึงการรวมในทางตรงกันข้าม. คอลัมน์ที่ 2 ของตารางที่ 5.1 จึงแสดงให้เห็นถึงความแตกต่างระหว่างตัวแปรเดิมลำดับที่ 1 (ค่าโหลดดิ่ง = 0.8) กับตัวแปรเดิมลำดับที่ 2 (ค่าโหลดดิ้ง = -0.9) เป็นต้น

		ตัวแปรหลังการแปลง			
		1	2	3	4
ตัวแปรเดิม	1	0.7	0.8	-0.1	0.5
	2	0.5	-0.9	0.3	-0.6
	3	0.6	0.2	0.5	-0.4
	4	0.4	0.2	-0.2	0.4

ตารางที่ 5.1

5.3 วิธีการประยุกต์ทฤษฎีการวิเคราะห์องค์ประกอบหลักกับงานรีโมทเซนซิ่ง

5.3.1 นิยามของตัวแปร

จากทฤษฎีของการวิเคราะห์องค์ประกอบหลักในหัวข้อที่ 5.2 เมื่อ X เป็นข้อมูลเดิมนำมาแปลงเพื่อให้ได้ข้อมูลใหม่คือ Y สามารถเขียนบล็อกไดอะแกรมกว้าง ๆ ได้ดังรูปที่ 5.5

รูปที่ 5.5 บล็อกไดอะแกรมของระบบ

ในทางรีโมทเซนซิ่งข้อมูลเดิม X ที่จะทำการวิเคราะห์คือข้อมูลภาพถ่ายดาวเทียมประกอบด้วย แบนด์ที่ 1 , แบนด์ที่ 2 , แบนด์ที่ 3 และแบนด์ที่ 4 (สมมติว่าดาวเทียมดวงนั้นมีข้อมูล 4 แบนด์) นั้นหมายความว่า ข้อมูล X ประกอบด้วยตัวแปร 4 ตัว คือ x_1, x_2, x_3 และ x_4 โดยกำหนดให้เป็นข้อมูลแบนด์ที่ 1,2,3 และ 4 ตามลำดับ เมื่อ X ผ่านการแปลงแล้วจะได้ข้อมูลใหม่คือ Y ซึ่งประกอบด้วย y_1, y_2, y_3 และ y_4 เช่นกัน จากทฤษฎีจะได้ว่า y_1 คือ องค์ประกอบหลักที่ 1 ให้ค่าความแปรปรวนสูงสุด y_2 คือ องค์ประกอบหลักที่ 2 ให้ค่าความแปรปรวนต่ำลงมาและเรื่อยไปจนถึง y_4 คือ องค์ประกอบหลักที่ 4 ให้ค่าความแปรปรวนต่ำสุด

5.3.2 วิธีการตรวจสอบการเปลี่ยนแปลง

ในการที่จะทำการตรวจจับการเปลี่ยนแปลงของพื้นดินโดยอาศัยภาพถ่ายดาวเทียมด้วยเทคนิคการวิเคราะห์องค์ประกอบหลักนั้นจำเป็นจะต้องใช้ข้อมูลภาพถ่ายดาวเทียมอย่างน้อยสองภาพ โดยมีเงื่อนไขว่าจะต้องเป็นข้อมูลที่ได้จากพื้นที่เดียวกันแต่เวลาแตกต่างกันออกไป ดังนั้นสำหรับการตรวจจับการเปลี่ยนแปลงจะต้องใช้ข้อมูลภาพถ่ายดาวเทียมสองวัน โดย 4 แบนด์แรกเป็นภาพถ่ายดาวเทียมวันหนึ่ง ส่วนอีก 4 แบนด์หลังเป็นภาพถ่ายดาวเทียมของอีกวันหนึ่งรวมทั้งหมดจะเป็นข้อมูล 8 แบนด์หรือ 8 ภาพ ในการแปลงจะได้ข้อมูลใหม่ 8 ภาพหรือ 8 องค์ประกอบหลักนั้นเอง จะเห็นว่าจำนวนแบนด์ของข้อมูลภาพเป็นปัจจัยกำหนดจำนวนองค์ประกอบทั้งหมดที่จะได้หลังผ่านการแปลงแล้ว

5.3.3 วิธีการแปลง

ในการแปลงด้วยเทคนิคการวิเคราะห์องค์ประกอบหลักจะมีขั้นตอนต่างๆดังต่อไปนี้

ขั้นที่ 1. คำนวณหาค่าเฉลี่ยของข้อมูลแต่ละแบนด์
ขั้นที่ 2. คำนวณหาแมทริกซ์ covariance
ขั้นที่ 3. คำนวณหา eigenvalue
ขั้นที่ 4. คำนวณหา eigenvector
ขั้นที่ 5 ทำการโปรเจคชั่น
ขั้นที่ 6. ทำการปรับระดับข้อมูลใหม่ให้อยู่ในช่วง 0-255 ระดับ

เขียนเป็นโพล์ชาร์ทได้ดังรูปที่ 5.6

5.3.4 วิธีการคำนวณหาแมทริกซ์ covariance

สูตรที่ใช้ในการคำนวณ คือ

$$\Sigma = (X - \overline{X})(X - \overline{X})^t \qquad (5.7)$$

เมื่อ $\overline{X}$ คือ ค่าเฉลี่ยของข้อมูลเดิม

$(X - \overline{X})'$ คือ ทรานสโพสของ $(X - \overline{X})$

จากนิยามในหัวข้อที่ 5.3.1 จะได้

$$\overline{X} = \overline{x}_1, \overline{x}_2, \overline{x}_3, \overline{x}_4, \overline{x}_5, \overline{x}_6, \overline{x}_7, \overline{x}_8$$

= ค่าเฉลี่ยของข้อมูลเดิมในแต่ละแบนด์

$$(X - \overline{X}) = [(x_1 - \overline{x}_1)(x_2 - \overline{x}_2)(x_3 - \overline{x}_3)(x_4 - \overline{x}_4)(x_5 - \overline{x}_5)(x_6 - \overline{x}_6)(x_7 - \overline{x}_7)(x_8 - \overline{x}_8)]$$

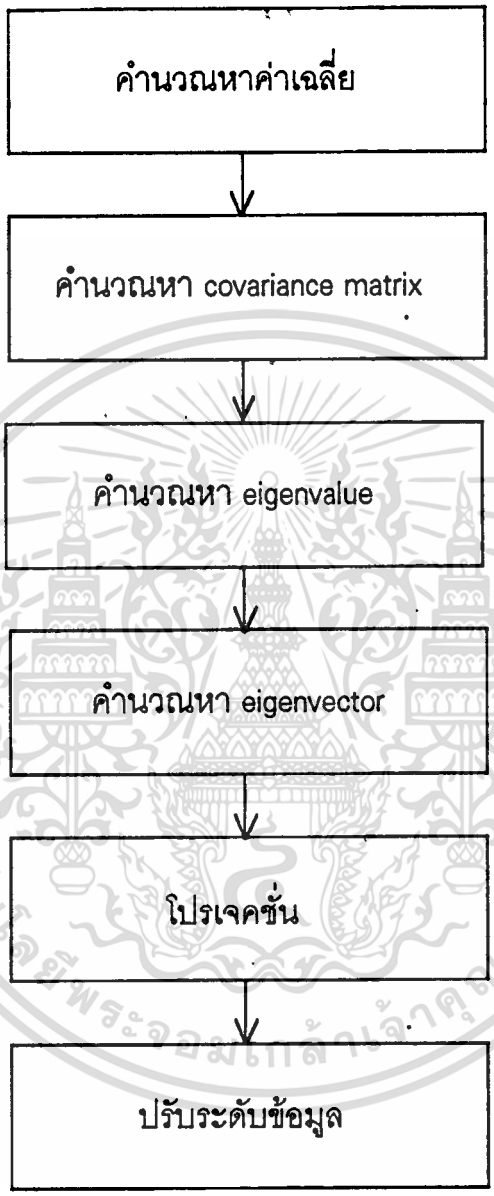
เขียนใหม่

$$(x_i - \overline{x_i}) = \begin{bmatrix} x_{i,1} - \overline{x_i} \\ x_{i,2} - \overline{x_i} \\ x_{i,3} - \overline{x_i} \\ \cdot \\ \cdot \\ \cdot \\ x_{i,j} - \overline{x_i} \end{bmatrix}$$

เมื่อ $x_{i,j}$ = ข้อมูลภาพตำแหน่งที่ j ของภาพแบนด์ที่ i

โดยข้อมูลภาพทั้ง 2 ภาพทุกแบนด์จะต้องมีขนาดเท่ากัน ผู้ใช้จะกำหนดขนาดของภาพเพื่อความสะดวกในการใช้งานกับภาพขนาดต่าง ๆ ได้ ในที่นี่สมมติภาพมีขนาด 256*256 จุดภาพ มีลำดับตำแหน่งของจุดภาพดังนี้

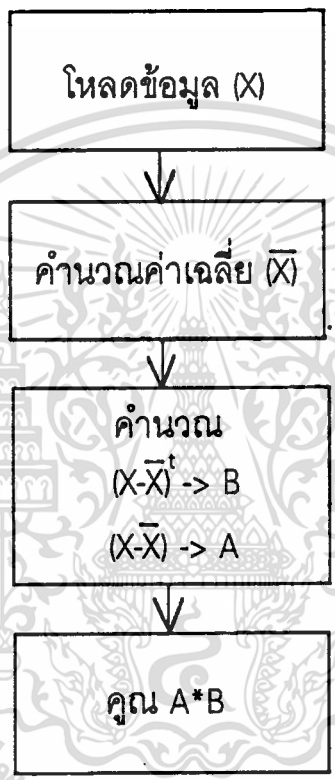
$$x_i = \begin{bmatrix} x_{i,1} & x_{i,2} & \cdot & \cdot & \cdot & x_{i,256} \\ x_{i,257} & x_{i,258} & \cdot & \cdot & \cdot & x_{i,512} \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ x_{i,65281} & x_{i,65282} & \cdot & \cdot & \cdot & x_{i,65536} \end{bmatrix}$$



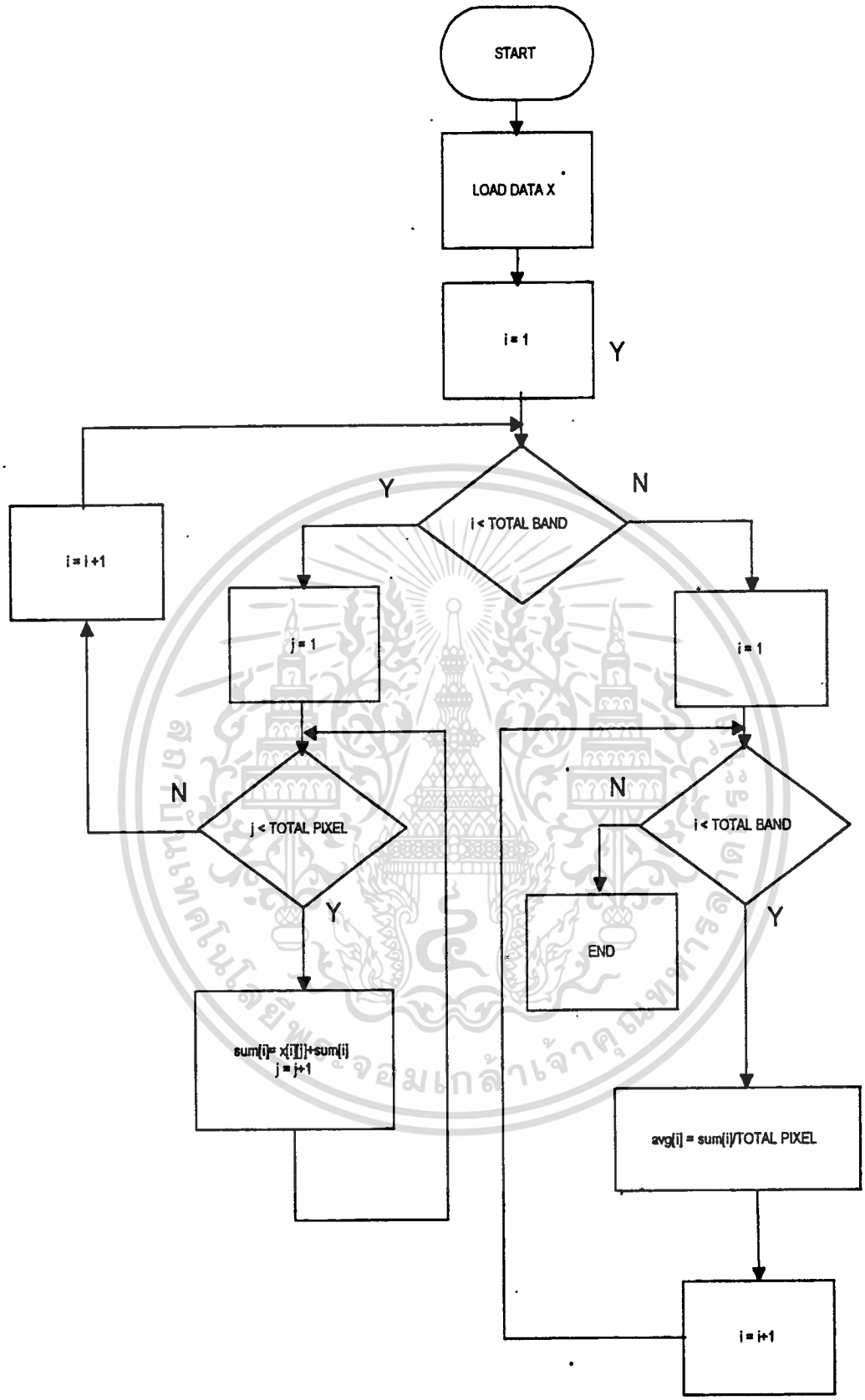
รูปที่ 5.6 ขั้นตอนของการแปลง

5.3.5 วิธีการคำนวณหา eigenvalue, eigenvector

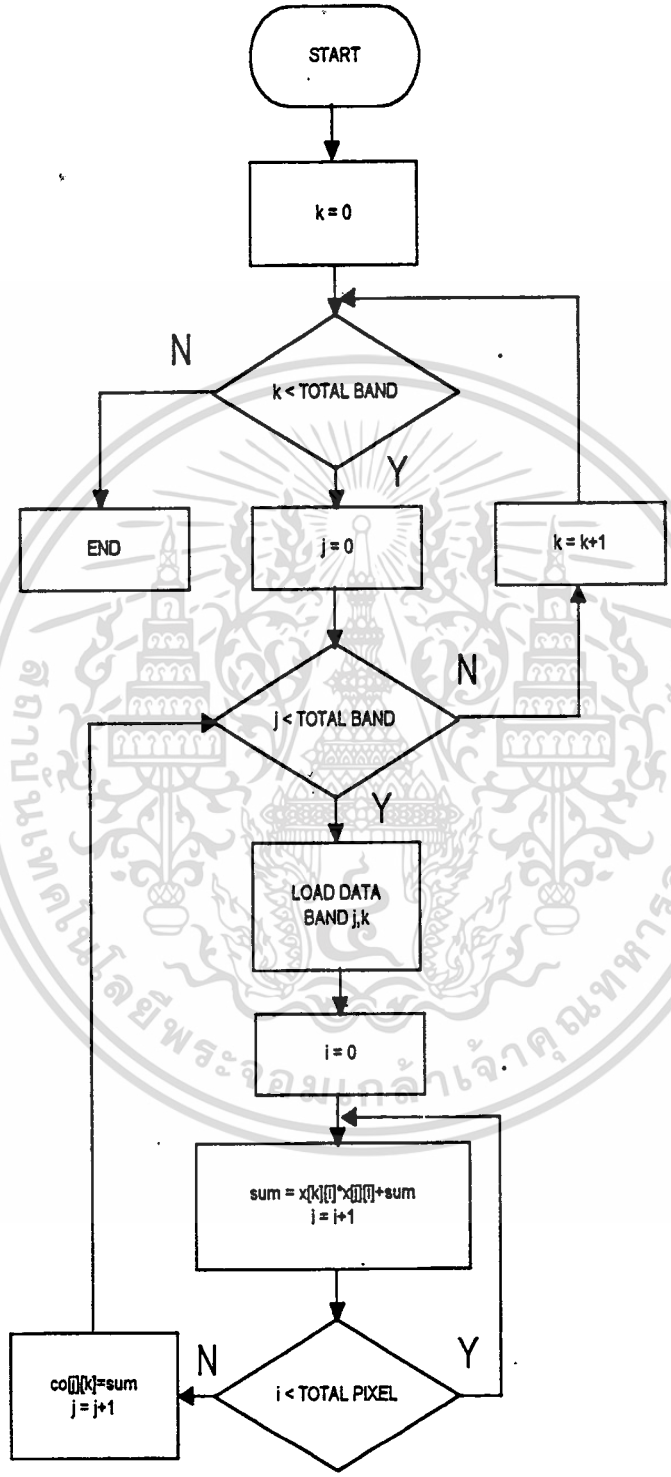
ค่า eigenvalue และ eigenvector ขององค์ประกอบหลักต่าง ๆ คำนวณหาได้จากแมทริกซ์ covariance ที่ได้จากข้อมูลเดิม โดยใช้วิธีการของ Jacobi [17] ซึ่งมีโพล์ชาร์ทแสดงการทำงานดังรูปที่ 5.10 จะได้ค่า eigenvalue เท่ากับจำนวนแบนด์ของข้อมูลเดิมและแต่ละค่า eigenvalue จะได้ค่า eigenvector เท่ากับจำนวนแบนด์ของข้อมูลเดิมเช่นกัน



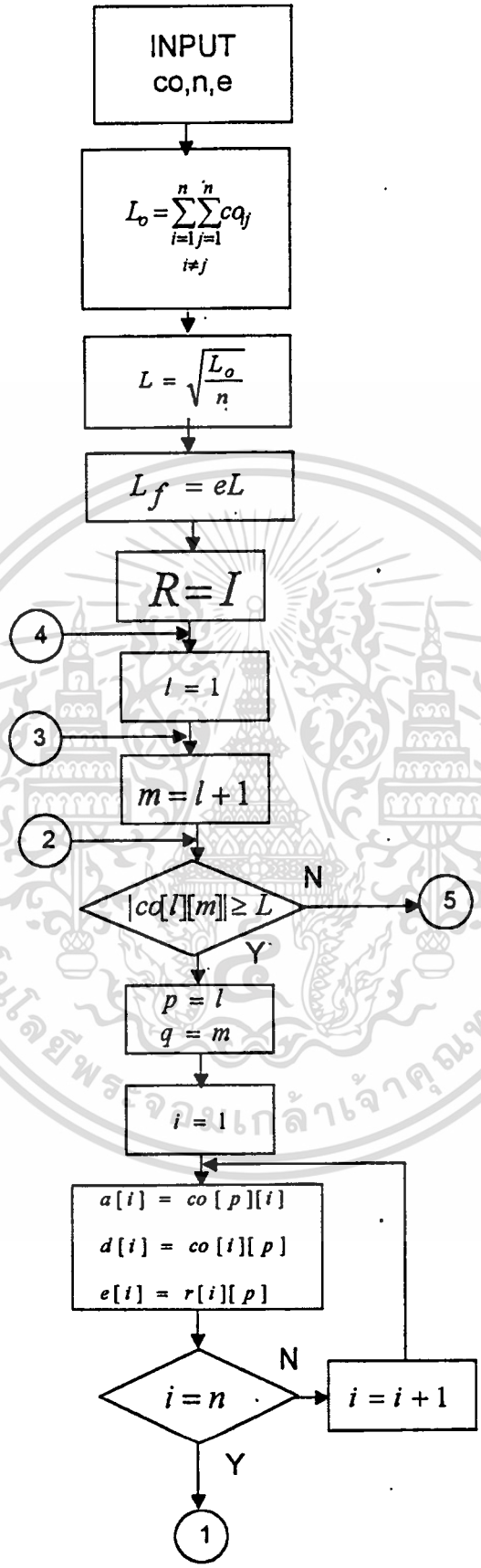
รูปที่ 5.7 ขั้นตอนของการคำนวณแมทริกซ์ covariance



รูปที่ 5.8 โพล์ชาร์ทสำหรับหาค่าเฉลี่ย ($\overline{X}$)

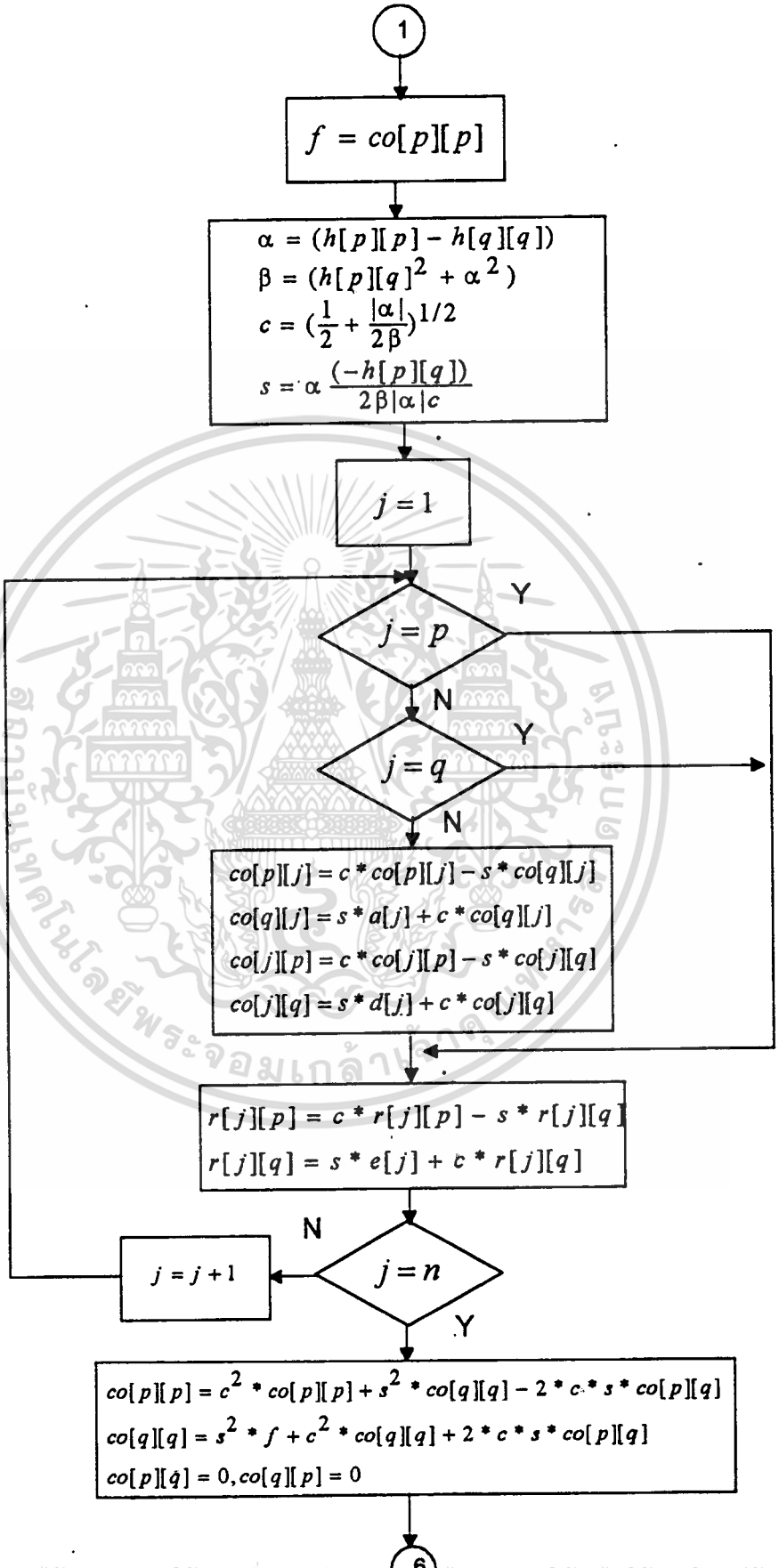


รูปที่ 5.9 โฟล์ชาร์ทในการสร้างแมทริกซ์ covariance

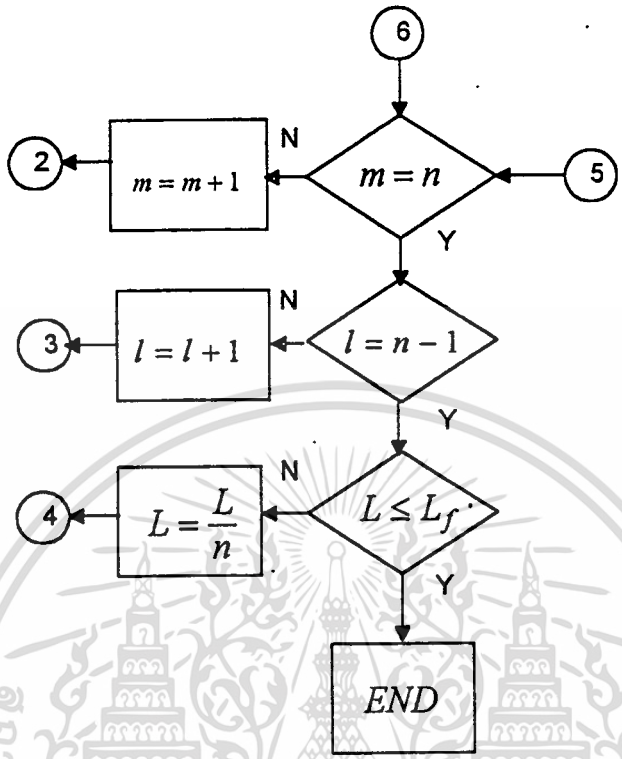


รูปที่ 5.10 โฟล์ชาร์ทการคำนวณหาค่า eigenvalue และ eigenvector

รูปที่ 5.10 ต่อ



รูปที่ 5.10 ต่อ



จากตัวอย่างของแมทริกซ์ covariance ใช้วิธีการ Jacobi จะได้ eigenvalue และ eigenvector ของแต่ละองค์ประกอบดังตารางที่ 5.2

ตารางที่ 5.2

องค์ประกอบที่ i	1	2	3	4
eigenvalue	35,127,092.175	9,920,275.969	438,945.185	354,293.191
eigenvector				
Ai	0.136	0.505	0.851	0.042
Bi	0.208	0.810	-0.522	0.167
Ci	0.585	-0.013	-0.046	-0.809
Di	0.772	-0.297	0.025	0.561

5.3.6 วิธีการโปรเจคชั่น

โดยทั่วไปการใช้ค่า eigenvector ของแต่ละองค์ประกอบหลักมาคูณกับข้อมูลเดิม (X) จะได้ข้อมูลใหม่ (Y) ดังนี้

สูตรทั่วไปของการคำนวณข้อมูลใหม่ (Y_i)

$$Y_i = [X - \overline{X}]^* \begin{bmatrix} A_i \\ B_i \\ C_i \\ D_i \end{bmatrix}$$

ข้อมูลใหม่องค์ประกอบหลักที่ (Y_1)

$$Y_1 = [X - \overline{X}]^* \begin{bmatrix} A_1 \\ B_1 \\ C_1 \\ D_1 \end{bmatrix}$$

ข้อมูลใหม่องค์ประกอบหลักที่ (Y_2)

$$Y_2 = [X - \overline{X}]^* \begin{bmatrix} A_2 \\ B_2 \\ C_2 \\ D_2 \end{bmatrix}$$

ข้อมูลใหม่องค์ประกอบหลักที่ (Y_3)

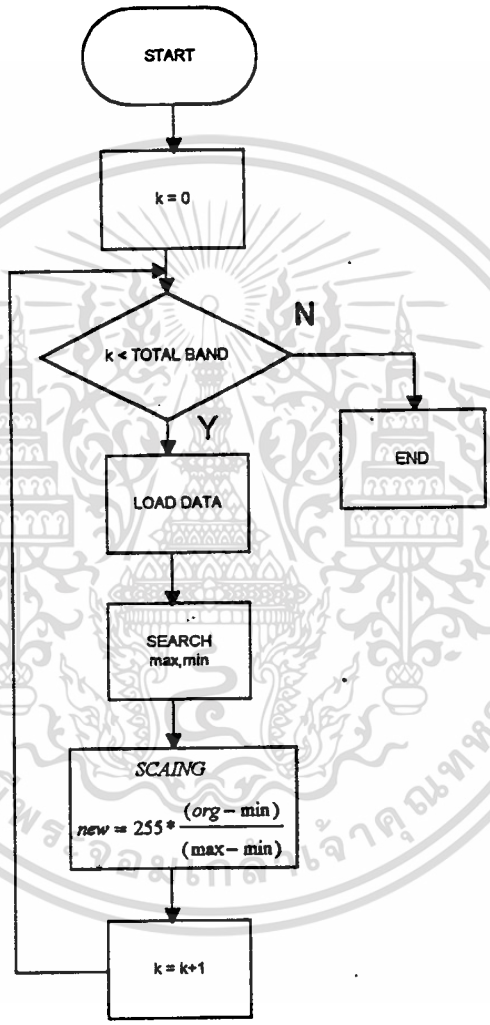
$$Y_3 = [X - \overline{X}]^* \begin{bmatrix} A_3 \\ B_3 \\ C_3 \\ D_3 \end{bmatrix}$$

ข้อมูลใหม่องค์ประกอบหลักที่ (Y_4)

$$Y_4 = [X - \overline{X}]^* \begin{bmatrix} A_4 \\ B_4 \\ C_4 \\ D_4 \end{bmatrix}$$

5.3.7 วิธีการสเกล

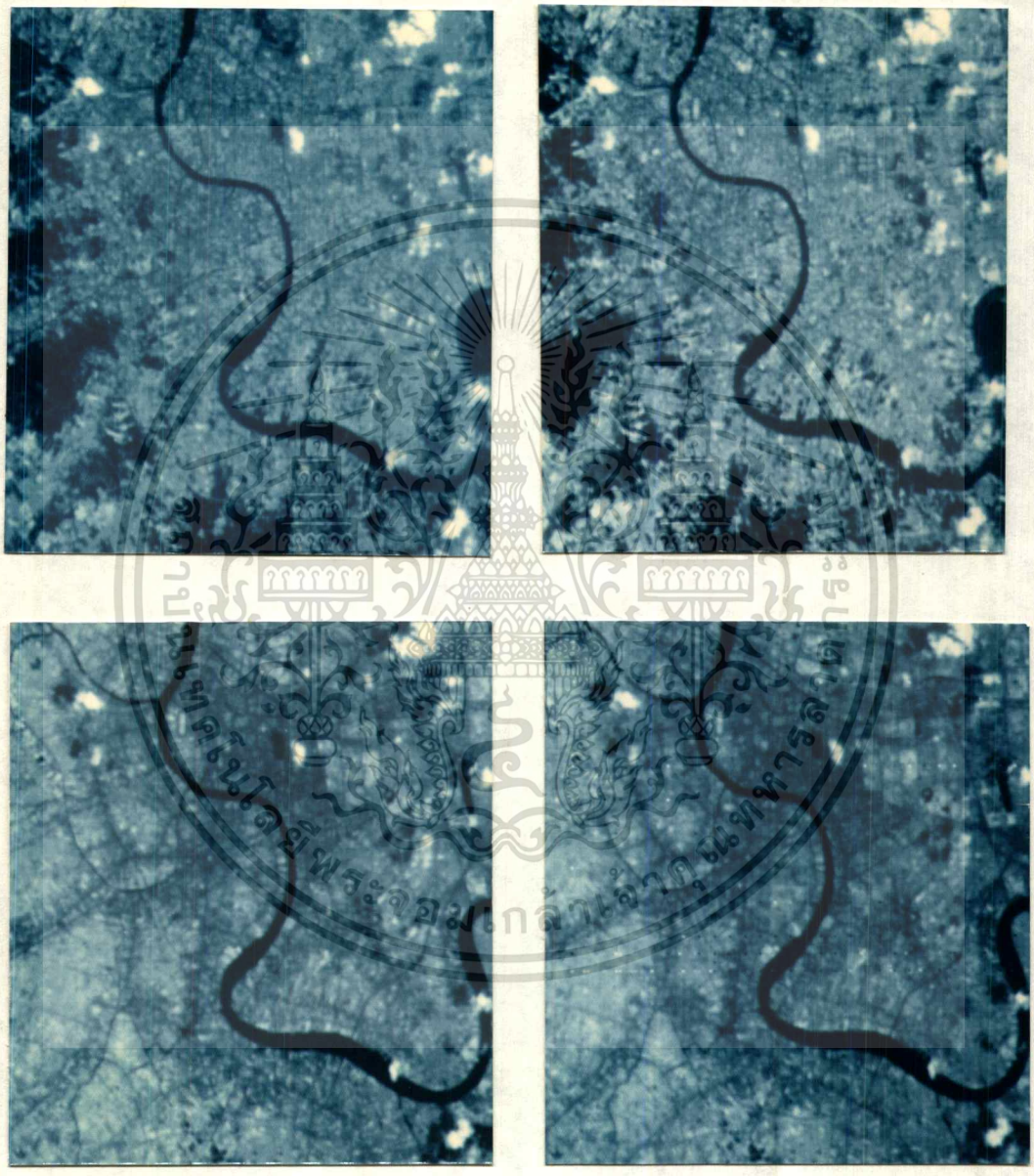
ข้อมูลที่ได้จากการโปรเจคชั่น (Y_i) มีค่าระดับสีเทาไม่ได้อยู่ในช่วง 0-255 ระดับ ซึ่งจะทำให้การใช้ข้อมูลไม่มีความหมาย ดังนั้นจึงจำเป็นต้องปรับค่าระดับสีเทาของแต่ละองค์ประกอบหลักใหม่ทั้งหมดให้อยู่ในช่วง 0-255 ระดับ มีวิธีการดังแสดงในโฟล์ชาร์ทรูปที่ 5.11



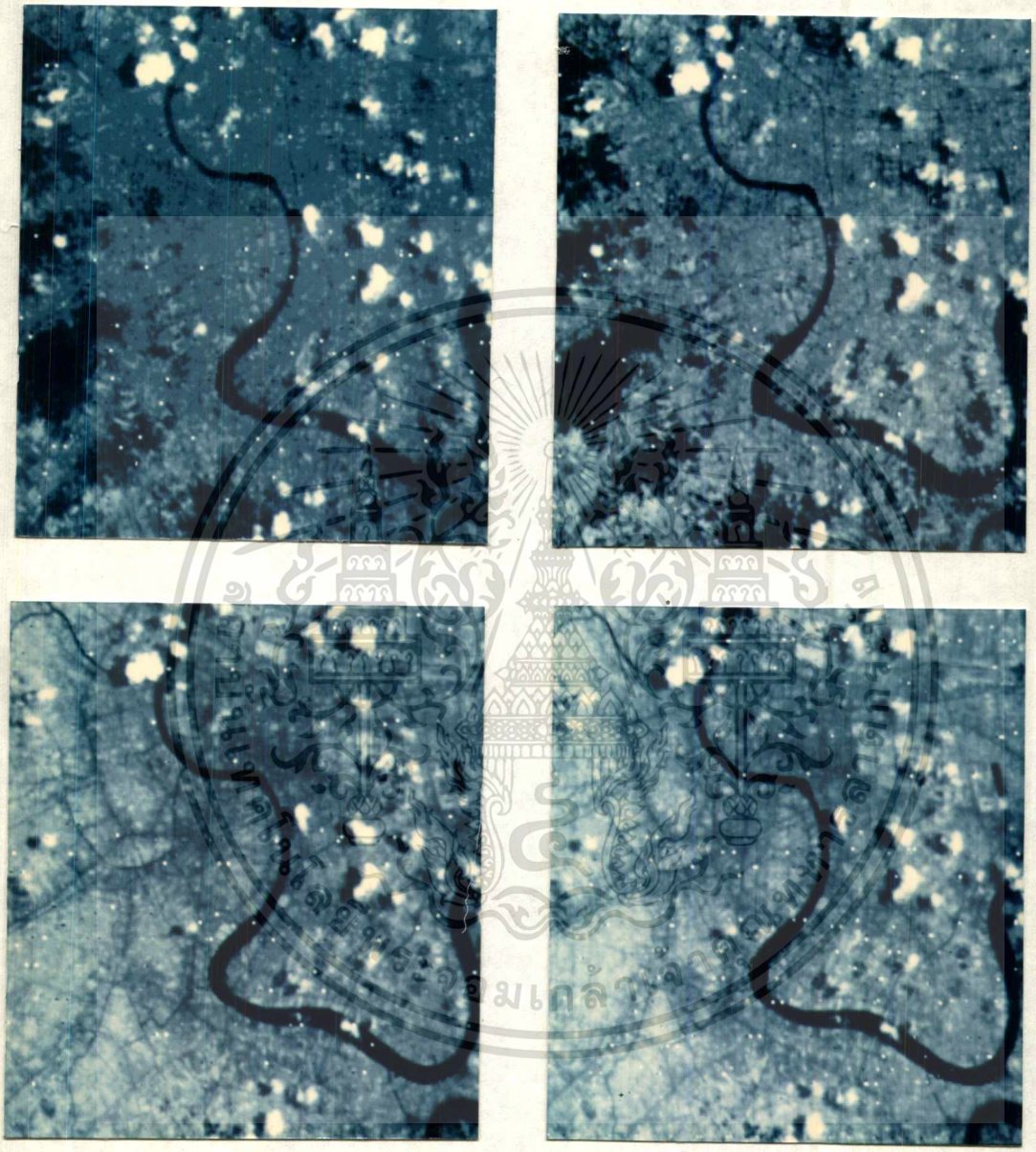
รูปที่ 5.11 โฟลชาร์ท์การสเกล

5.4 ผลการทดลอง

จากวิธีการที่ได้กล่าวในข้อหัวที่ 5.3 ได้ทำการทดลองกับข้อมูลภาพถ่ายดาวเทียม MOS-I ระบบ MESSR 2 วัน (วันละ 4 แบนด์) ทั้งหมด 8 แบนด์ ดังแสดงในรูปที่ 5.12



(ก) กลุ่มของภาพถ่ายดาวเทียมวันแรกจำนวน 4 แบนด์



(ข) กลุ่มของภาพถ่ายดาวเทียมวันที่สองจำนวน 4 แบนด์

รูปที่ 5.12

นำข้อมูลทั้ง 8 แบนด์มาทำการประมวลผลพร้อม ๆ กัน เพื่อคำนวณหาองค์ประกอบหลัก ซึ่งจะแสดงให้เห็นบริเวณที่เกิดการเปลี่ยนแปลงขึ้นอยู่ใน 8 องค์ประกอบหลักเช่นกัน ผลจากการคำนวณจะได้ค่า eigenvalues และ variance ดังตารางที่ 5.3 ส่วนค่า eigenvectors ที่สอดคล้องกันดังตารางที่ 5.4 จากนั้นทำการโปรเจคชั่นข้อมูลเดิมทั้ง 8 แบนด์ด้วยค่า eigenvectors และขณะเดียวกันทำการสเกลข้อมูลที่ผ่านการโปรเจคชั่นแล้วให้มีค่าระดับสีเทาอยู่ในช่วง 0-255 ระดับ จะได้ภาพใหม่ 8 ภาพแสดงให้เห็นบริเวณที่เกิดการเปลี่ยนแปลงอันเนื่องมาจากสาเหตุต่าง ๆ ดังรูปที่ 5.13

ตารางที่ 5.3

องค์ประกอบหลัก	1	2	3	4	5	6	7	8
Eigenvalue	4,053,621.25	2,552,036.50	1,573,298.12	192.521.00	177,305.11	135,819.23	65,609.37	54,224.72
% variance	46.04	28.99	17.87	2.19	2.01	1.54	0.74	061

ตารางที่ 5.4

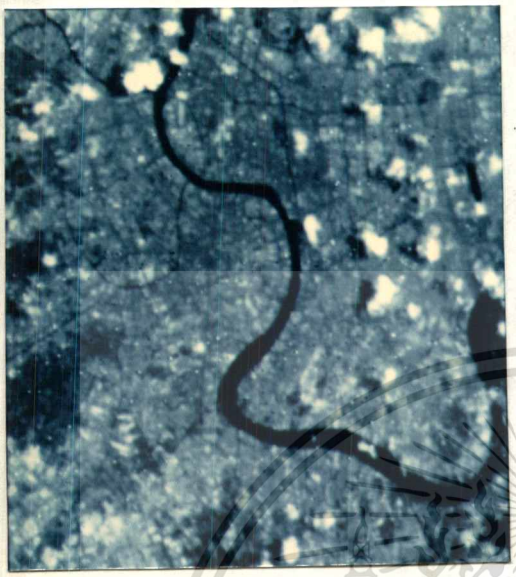
Eigenvector							
องค์ประกอบที่ 1	องค์ประกอบที่ 2	องค์ประกอบที่ 3	องค์ประกอบที่ 4	องค์ประกอบที่ 5	องค์ประกอบที่ 6	องค์ประกอบที่ 7	องค์ประกอบที่ 8
+0.238	-0.351	+0.236	+0.001	+0.099	-0.216	+0.355	+0.763
+0.383	-0.582	+0.305	-0.269	+0.103	-0.266	-0.180	-0.486
+0.267	+0.218	+0.591	+0.260	-0.251	+0.246	-0.551	+0.193
+0.185	+0.207	+0.422	+0.258	+0.083	+0.248	+0694	-0.356
+0.307	-0.063	-0.335	+0.780	-0.078	-0.404	-0.049	-0.102
+0.514	-0.247	-0.438	-0.550	-0.082	+0.685	+0.018	+0.057
+0.477	+0.481	-0.121	-0.421	-0.451	-0.354	+0.141	+0.002
+0.325	+0.389	-0.070	-0.074	+0.833	-0.055	-0.181	+0.061

5.5 สรุป

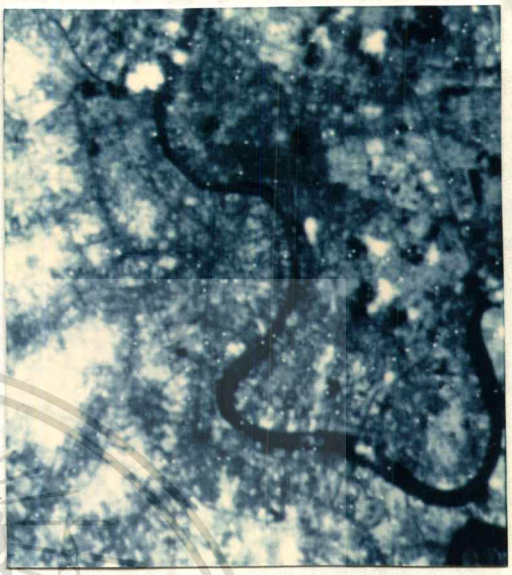
จากรูปที่ 5.13 แสดงให้เห็นการเปลี่ยนแปลงอันเนื่องมาจากสาเหตุต่างๆ ในการทำวิทยานิพนธ์นี้ไม่ได้ออกสนามไปค้นหาข้อมูลทางภูมิศาสตร์ ดังนั้นจึงไม่สามารถระบุถึงรายละเอียดบริเวณที่เกิดการเปลี่ยนแปลงเกิดจากสาเหตุอะไร แต่อย่างไรก็ตามเพื่อให้เห็นการเปลี่ยนแปลงเกิดขึ้นจริง ดูได้จากการเปลี่ยนแปลงของการกระจายของกลุ่มเมฆที่กระจายสามารถแสดงให้เห็นได้ชัดเจนมากในภาพองค์ประกอบหลักที่ 3 (รูปที่ 5.13) ซึ่งพื้นที่สีขาวเกิดจากเมฆของกลุ่มภาพแรก ขณะที่เมฆของกลุ่มภาพที่สองจะแสดงให้เห็นเป็นพื้นที่สีดำ

สำหรับภาพองค์ประกอบหลักที่ 5 (รูปที่ 5.13) แสดงให้เห็นจุดภาพของสัญญาณรบกวนต่างๆ ที่มีอยู่ในกลุ่มภาพวันที่สองจะปรากฏให้เห็นเป็นจุดภาพสีขาวและยังมีการเปลี่ยนแปลงบริเวณริมฝั่งแม่น้ำเป็นแถบสีขาวในบางช่วง ซึ่งตรงกับพื้นที่ริมฝั่งแม่น้ำที่เกิดน้ำท่วมในกลุ่มภาพวันที่สอง ขณะเดียวกันพื้นที่สีดำแสดงให้เห็นพื้นที่ริมฝั่งแม่น้ำที่เกิดน้ำท่วมในกลุ่มภาพวันแรก

จากผลการทดลองดังกล่าวพบว่าเทคนิคการวิเคราะห์องค์ประกอบหลักเป็นวิธีการใช้ในการตรวจสอบการเปลี่ยนแปลงของทรัพยากรต่างๆบนพื้นดินได้เป็นอย่างดี โดยการตรวจสอบการเปลี่ยนแปลงของทรัพยากรชนิดต่างๆจากทุกแบนด์ของภาพถ่ายดาวเทียมต่างเวลานี้จะปรากฏอยู่ในภาพองค์ประกอบหลักอันดับท้ายๆที่มีค่าสหสัมพันธ์ต่ำ วิธีการตรวจสอบดังกล่าวจะให้ความสะดวกและความถูกต้องสูงกว่าการตรวจสอบโดยการเปรียบเทียบภาพแบนด์ต่อแบนด์ของภาพถ่ายดาวเทียมต่างเวลา



องค์ประกอบหลักที่ 1



องค์ประกอบหลักที่ 2

องค์ประกอบหลักที่ 3

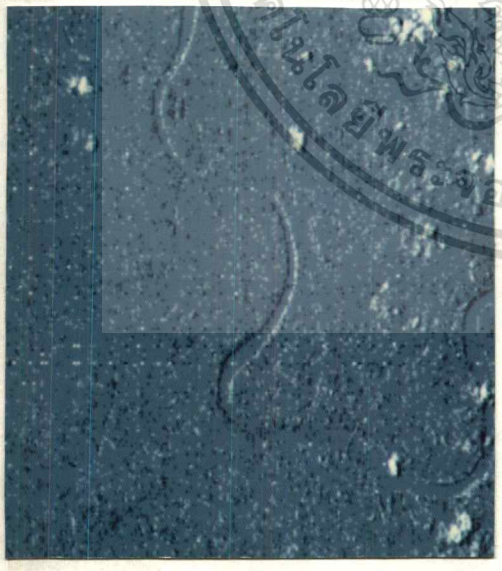
องค์ประกอบหลักที่ 4

รูปที่ 5.13 ภาพองค์ประกอบหลักต่าง ๆ จำนวน 8 องค์ประกอบหลัก

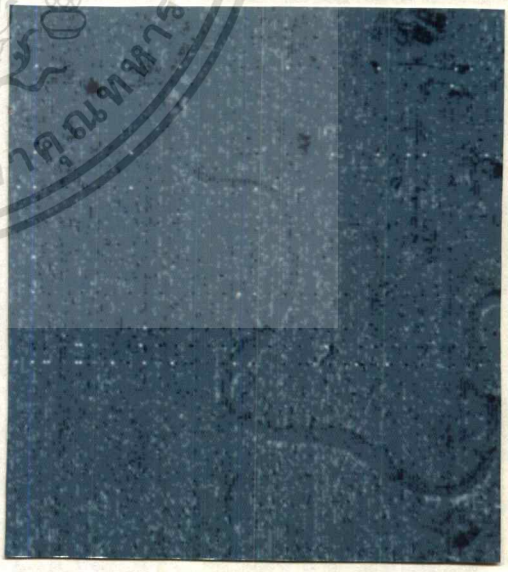
รูปที่ 5.13 ต่อ

องค์ประกอบหลักที่ 5

องค์ประกอบหลักที่ 6



องค์ประกอบหลักที่ 7



องค์ประกอบหลักที่ 8

บทที่ 6
บทสรุปและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

การเชื่อมต่อภาพสามารถกระทำได้ 3 วิธีคือ วิธีที่ 1 การเชื่อมต่อแบบดั้งเดิมโดยการเฉลี่ยค่าระดับสีเทาของภาพเฉพาะบริเวณที่ซ้อนทับกันเท่านั้น วิธีที่ 2 การเชื่อมต่อภาพด้วยการขจัดรอยตะเข็บโดยการไบอัสบริเวณรอยต่อด้วยฟังก์ชั่นเชิงเส้น วิธีที่ 3 การเชื่อมต่อภาพด้วยการขจัดรอยตะเข็บโดยการไบอัสบริเวณรอยต่อด้วยฟังก์ชั่นบัทเตอร์เวิร์ธ ซึ่งเป็นวิธีที่ใช้ในวิทยานิพนธ์ฉบับนี้ทั้งนี้เพราะสามารถรักษาความเป็นเนื้อเดียวกันของวัตถุที่อยู่ใกล้รอยต่อของภาพได้เป็นอย่างดี วิธีการเชื่อมต่อภาพปกติจะกระทำได้ครั้งละหนึ่งแบนด์เท่านั้น ถ้าหากภาพถ่ายดาวเทียมมีหลายแบนด์ก็จะต้องกระทำการเชื่อมต่อภาพเท่ากับจำนวนแบนด์ของภาพนั้น ดังนั้นในการเชื่อมต่อภาพเพื่อให้ได้เป็นภาพสีจะต้องกระทำการเชื่อมต่อภาพอย่างน้อยที่สุดสามแบนด์ โดยที่จะกำหนดให้ข้อมูลภาพแบนด์ใดแบนด์หนึ่งเป็นสีแดง สีเขียว และสีน้ำเงิน เมื่อนำมาทำการผสมสีจะได้การเชื่อมต่อภาพสีที่ปราศจากรอยตะเข็บในลักษณะของสีเทียม ทำให้ได้ภาพสีที่มีบริเวณกว้างยิ่งขึ้น ส่งผลให้การใช้ข้อมูลภาพสีเพื่อการแปลความหมายภาพกระทำได้อย่างมีประสิทธิภาพ

สำหรับการตรวจสอบการเปลี่ยนแปลงบนภาคพื้นดินโดยการใช้ภาพถ่ายดาวเทียม เนื่องจากข้อมูลภาพจากดาวเทียมสามารถครอบคลุมพื้นที่ได้บริเวณกว้าง ซึ่งมีความเหมาะสมสำหรับการตรวจสอบพื้นที่กว้าง ๆ เนื่องจากการเปลี่ยนแปลงต่าง ๆ ที่เกิดขึ้นเป็นบริเวณกว้าง ๆ ไม่สามารถจะทำการตรวจสอบได้โดยการสำรวจพื้นที่บนพื้นดิน รวมทั้งการเปลี่ยนแปลงที่เกิดขึ้นในบริเวณที่อยู่ในป่าลึกหรือในหุบเขาที่ไม่อาจเข้าไปทำการสำรวจได้ ดังนั้นการใช้ภาพถ่ายดาวเทียมตรวจสอบการเปลี่ยนแปลงจึงเป็นวิธีการที่เหมาะสมที่สุด เพื่อให้ได้มาซึ่งการเปลี่ยนแปลงที่เกิดจากสาเหตุต่าง ๆ ครอบคลุมเป็นบริเวณกว้างและกระทำตรวจสอบการเปลี่ยนแปลงทุก ๆ กรณี และกระทำพร้อม ๆ กัน จึงได้เสนอเทคนิคการวิเคราะห์องค์ประกอบหลักเพื่อทำการแปลงข้อมูลภาพสองวันที่ได้จากการสแกนจากพื้นที่เดียวกันแต่ต่างเวลากัน จะได้ข้อมูลใหม่แสดงให้เห็นการเปลี่ยนแปลงที่เกิดขึ้น วิทยานิพนธ์นี้ไม่ได้ออกสนามไปค้นหาข้อมูลทางภูมิศาสตร์ (Geographic Information) จึงทำให้ไม่สามารถจำแนกประเภทของการเปลี่ยนแปลงได้ เพียงแต่แสดงให้เห็นการเปลี่ยนแปลงที่เกิดขึ้นเท่านั้น

การวิจัยของวิทยานิพนธ์นี้ได้เขียนโปรแกรมในการทำงานเป็นภาษาซี บนเครื่องไมโครคอมพิวเตอร์ PC/AT 80486DX (33 Mhz) และในการผสมสีของข้อมูลภาพกระทำบนเครื่องไมโครคอมพิวเตอร์ NEC 9801 เพื่อให้ได้ภาพสีในขั้นตอนสุดท้าย

6.2 ปัญหาที่เกิดขึ้นและข้อเสนอแนะ

การค้นหาตำแหน่งซ้อนทับของภาพเพื่อจะนำตำแหน่งที่คำนวณได้มาทำการต่อภาพ ซึ่งถือว่าเป็นปัจจัยหลักในการต่อภาพด้วย ถ้าการคำนวณไม่สามารถกระทำได้นั่นหมายความว่าการที่จะต่อภาพดังกล่าวก็ไม่สามารถกระทำได้ ซึ่งเกิดจากสาเหตุต่าง ๆ เช่น ข้อมูลภาพมีเมฆทับบริเวณสำคัญ ๆ ของภาพ ภาพมีค่า dynamic range ค่อยข้างต่ำทำให้การแยกแยะความแตกต่างด้วยเทคนิคการทำสหสัมพันธ์ไม่สามารถกระทำได้ ดังนั้นการแก้ไขปัญหาดังกล่าวโดยการพยายามเลือกข้อมูลภาพถ่ายดาวเทียมที่ข้อมูลภาคพื้นดินมีลักษณะเด่นชัดอย่างเช่น มีแม่น้ำ ถนนตัดกัน เป็นต้น ส่วนวิธีแก้ไขปัญหาทางเทคนิคจะต้องคิดค้นเทคนิคใหม่สำหรับการค้นหาตำแหน่งซ้อนทับที่ดีกว่าเทคนิคการทำสหสัมพันธ์ ในแง่ของการใช้เวลาในการคำนวณซึ่งกรณีนี้ไม่อยู่ในขอบเขตของการวิจัยวิทยานิพนธ์นี้ ปัญหาในการประมวลผลข้อมูลที่มีพื้นที่กว้าง ๆ มากกับเครื่องไมโครคอมพิวเตอร์คือ ความเร็วในการคำนวณและหน่วยความจำหลัก เนื่องจากตัวแปลภาษาที่ใช้อยู่ปัจจุบันไม่สามารถใช้กับหน่วยจำหลักได้เกิน 1 Mb ดังนั้นการประมวลผลจึงต้องอาศัยหน่วยความจำรอง ซึ่งทำให้การประมวลผลช้า ถ้าหากมีการนำระบบไปประมวลผลกับเครื่องคอมพิวเตอร์ที่สามารถเรียกใช้หน่วยความจำหลักได้เท่ากับจำนวนที่ติดตั้งจริงจะทำการประมวลผลข้อมูลได้เร็วยิ่งขึ้น

เอกสารอ้างอิง

[1] สักรียา ชิตวงค์ ธนิตย์ ตรีสุวรรณวัฒน์ ดร. ฟูศักดิ์ ชิวสุวิทย์ , "การกำจัดรอยตะเข็บออกจากการต่อภาพสีของภาพถ่ายดาวเทียม MOS-I MESSR", การประชุม วิชาการวิศวกรรมไฟฟ้า ครั้งที่ 15 หน้า D-6-7-D-6-9, 3-4 ธันวาคม 2535.

[2] สักรียา ชิตวงค์ บัณฑิต สุมนวัฒนเดช ดร. ฟูศักดิ์ ชิวสุวิทย์ , "การตรวจสอบการเปลี่ยนแปลงภาคพื้นดินนด้วยการวิเคราะห์องค์ประกอบหลักของข้อมูลภาพถ่ายดาวเทียม MOS-I MESSR", การประชุม วิชาการวิศวกรรมไฟฟ้า ครั้งที่ 16 หน้า 599-603, 25-26 พฤศจิกายน 2536.

[3] F. Cheevasuvit, K. Dejhan and S. Chitwong, "Seam removal from colour mosaicking of MOS-1 MESSR images", Proc. of the 13th Asian Conference on Remote Sensing, Ulaanbaatar, Mongolia, pp. Q-5-1 - Q-5-6. 7-11 Oct. 1992.

[4] F. Cheevasuvit, K. Dejhan and S. Chitwong,"Monitoring Land-Cover Change by Principal Component Analysis of Multitemporal MOS-1 MESSR Data", Proc. of the South East Asian Regional Conference on Education and Research in Remote Sensing, Johor Bahru, Malaysia, Session 6, june, 1993.

[5] Y. Shiren, L. Li and G. Peng, "Two dimensional seam-point searching in digital image mosaicking ", American Society for Photogrammetry and Remote Sensing, pp. 50-53, 1989.

[6] D.L Milgram, "Computer methods for creating photomosaics", IEEE Trans. on Com., pp. 1113-1119, 1975.

[7] D.L. Milgram, "Adaptive techniques for photomosaicking", IEEE Trans. on Com., pp. 1175-1180,1977.

[8] S. Murai and M. Akiyama, "Digital Mosaic of colour aerial photographs", 14th Congress, International Society for Photogrammetry, Hamberg, pp. 426-429, 1980.

[9] F. Cheevasuvit, C. Peanvijarnpong and S. Murai, "Mosaicking of poor contrast images", Proc. of The 9th Asian Conference on Remote Sensing, Bangkok, Thailand, pp. H-1-1-1 - H-1-1-9, 23-29 Nov. 1988.

[10] R.N. Colwell, "Manual of remote Sensing", Second Edition, American Society of Photogrammetry, Falls Church, Virginia, 1985.

[11] B.R. Hunt and O. Kublek, "Karhunen-Loeve Multispectral Image Restoration, Part I; Theory", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-32, No. 3, June 1984.

[12] T. Fung, "An Assessment of TM Imagery for Land-Cover Change Detection", IEEE Trans. on Geoscience and Remote Sensing, Vol.28, pp. 681-684, July 1990.

[13] T. Fung, E. Le Frew, Application of Principal Components Analysis to Change Detection", Photogrammetry. Eng. Remote Sensing, Vol.53, No.12, pp. 1649-1658, 1987.

[14] S.K. Jensen and E.A Waltz, "Principal Components Analysis and Canonical Analysis In Remote Sensing", Proc. Am. Soc. of Photogrammetry, Falls Church, VA: Am. Soc. of Photogrammetry, pp.337-348, 1979.

[15] P.T. Eliason, T.J Donovan and P.S. Chavez, "Integration of geologic, geochemical, and geophysiscal data of the Cement oil field, Oklahoma, using spatial array processing", Geophysical, Vol.48, No.10, pp.1305-1317, Oct 1983.

[16] G.F Byrne, P.F. Crapper and K.K. Mayo, "Monitoring Land-Cover Change by Principal Component Analysis of Multitemporal LANDSAT Data", Remote Sensing Environ., Vol.10, pp.175-188, 1980.

[17] R.W. Hornbeek, Numerical Methods, Engelwood Cliffs, NJ:Prentice-Hall, 1975.

[18] R.C. Gonzalez, and R.E Woods, "Digital Image Processing" (Second edition), Reading, Addison-Weslay Publishing Inc, 1992.

[19] USER'S GUIDE FOR MOS-I MESSR DATA FORMAT, NATIONAL SPACE DEVELOPMENT AGENCY OF JAPAN, MARCH 1989.

ภาคผนวก ก.
ผลงานวิจัยที่ได้รับการตีพิมพ์

[1] สักรียา ชิตวงค์ ธนิตย์ ตรีสุวรรณวัฒน์ ดร. ฟูศักดิ์ ชิวสุวิทย์ , "การกำจัดรอยตะเข็บออกจากการต่อภาพสีของภาพถ่ายดาวเทียม MOS-I MESSR", การประชุม วิชาการวิศวกรรมไฟฟ้า ครั้งที่ 15 หน้า D-6-7-D-6-9, 3-4 ธันวาคม 2535.

[2] สักรียา ชิตวงค์ บัณฑิต สุมนวัฒนเดช ดร. ฟูศักดิ์ ชิวสุวิทย์ , "การตรวจสอบการเปลี่ยนแปลงภาคพื้นดินนด้วยการวิเคราะห์องค์ประกอบหลักของข้อมูลภาพถ่ายดาวเทียม MOS-I MESSR", การประชุม วิชาการวิศวกรรมไฟฟ้า ครั้งที่ 16 หน้า 599-603, 25-26 พฤศจิกายน 2536.

[3] F. Cheevasuvit, K. Dejhan and S. Chitwong, "Seam removal from colour mosaicking of MOS-1 MESSR images", Proc. of the 13th Asian Conference on Remote Sensing, Ulaanbaatar, Mongolia, pp. Q-5-1 - Q-5-6, 7-11 Oct. 1992.

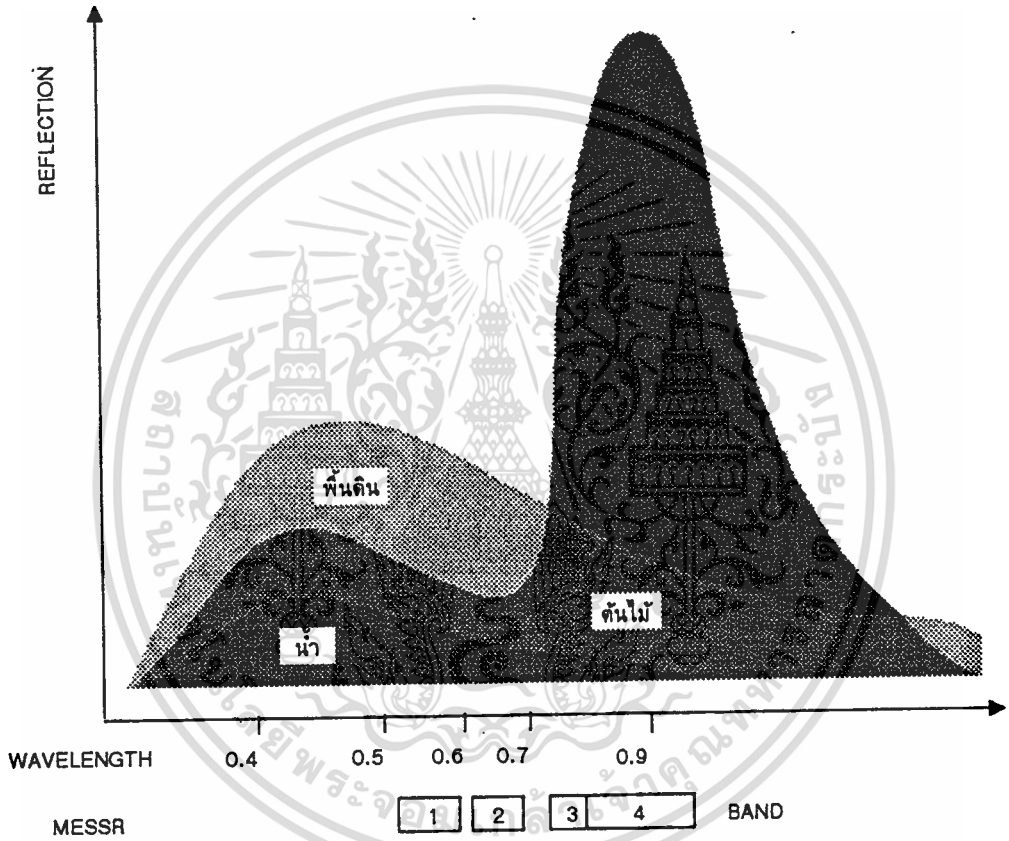
[4] F. Cheevasuvit, K. Dejhan and S. Chitwong,"Monitoring Land-Cover Change by Principal Component Analysis of Multitemporal MOS-1 MESSR Data", Proc. of The South East Asian Regional Conference on Education and Research in Remote Sensing, Johor Bahru, Malaysia, Session 6, June, 1993.

ภาคผนวก ข

รายละเอียดข้อมูลภาพของดาวเทียม MOS-I MESSR

MOS-I MESSR ย่อมาจาก Marine Observation Satellite Multi-Spectral Electronic Self-Scanning Radiometer) เป็นระบบอิเล็กทรอนิกส์สแกนถ่ายภาพหลายช่วงคลื่นสำหรับตรวจจับการสะท้อนของแสงอาทิตย์จากผิวโลก ดาวเทียม MOS ถ่ายภาพ 4 ช่วงคลื่น (Band) จาก 0.5 ถึง 1.1 ไมโครเมตร รายละเอียดภาพ 50 เมตร และแนวถ่ายภาพกว้าง 100 กิโลเมตร จุดภาพขนาด 8 บิท ให้ค่าระดับสีเทา 256 ระดับ (0-255 ระดับ) แต่ละช่วงคลื่นมีรายละเอียดและผลการกระท้อนคือ แบนด์ที่ 1 ย่าน 0.51-0.59 ไมโครเมตร แบนด์ที่ 2 ย่าน 0.61-0.69 ไมโครเมตร แบนด์ที่ 3 ย่าน 0.72-0.80 ไมโครเมตร แบนด์ที่ 4 ย่าน 0.80-1.10 ไมโครเมตร โดยที่แบนด์ที่ 1 และ 2 เป็นช่วงคลื่นที่ตามองเห็น แบนด์ที่ 3 และ 4 เป็นช่วงคลื่นอินฟราเรด สำหรับแบนด์ที่ 1 และ 2 ให้ผลการสะท้อนของพื้นดินดีที่สุด รองลงมาต้นไม้และน้ำตามลำดับ ซึ่งจะให้ผลตอบสนองดีในช่วงความถี่สูงและค่อยๆลดต่ำลงเมื่อความถี่ลดลง ในขณะที่แบนด์ที่ 3 จะให้ผลตอบสนองการกระท้อนแสงของต้นไม้ดีขึ้นเมื่อความถี่ต่ำลง และในแบนด์ที่ 4 ช่วงความถี่สูงจะให้ผลตอบสนองการสะท้อนของต้นไม้ค่อยๆเพิ่มขึ้น จนถึงประมาณกึ่งกลางของช่วงคลื่นผลตอบสนองการสะท้อนของต้นไม้ค่อยๆลดลง เช่นเดียวกับแบนด์ที่ 1 ดังกราฟรูปที่ 1

การแปลงข้อมูลภาพ สถานีรับสัญญาณทำการเก็บข้อมูลที่ยังไม่ผ่านการประมวลผลใด ๆ ไว้ในเทปความจุสูง (HDDT-High Density Digital Tape) จากนั้นข้อมูลจาก HDDT จะถูกแปลงให้อยู่ในรูปแบบของ CCT (Computer Compatible Tape) ซึ่งรูปแบบของ CCT จะขึ้นอยู่กับความต้องการของผู้ใช้ข้อมูล โดยสามารถแบ่งได้ 4 ระดับ (ระดับ 0,1,2,3) ตามเอกสารอ้างอิง [19] ในวิทยานิพนธ์นี้เลือกใช้ข้อมูล CCT ระดับ 1 ซึ่งข้อมูลได้ผ่านการประมวลผลในส่วนของ radiometric correction และ geometric correcttion จากนั้นทำการเขียนข้อมูล CCT ไปยัง HD เพื่อความสะดวกในการเรียกใช้งานสำหรับการประมวลผลข้อมูลภาพโดยใช้เครื่องคอมพิวเตอร์ต่อไป



รูปที่ 1

ภาคผนวก ค โปรแกรมการทดลอง

```
/*********************************/
/* Program name  : registration      */
/* Author : Chitwong.S               */
/*********************************/
#include<graphics.h>
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<alloc.h>
#define SIZE 256
#define LSUBSIZE  32.0
#define RSUBSIZE  64.0
void MemAlloc();
void init();
void CloseGraph();
void DrawCur(double x1,double y1);
void MovCurLeft();
void MovCurRight();
void DrawRec(double x1,double y1);
void SubImgLeft(int x1,int y1);
void SubImgRight(int x1,int y1);
float mean1();
float mean2(int n,int m);
float corrag(int n,int m);
void dispcorr();
void Dot();
void MemErr();
void FopenErr();
int graphdriver=IBM8514,graphmode=IBM8514LO;
char *driverparth= "";
unsigned char far *pict;
unsigned char **img6,**img7,**img8,**lsubimg,**rsubimg,**limg,**rimg;
float avr1,avr2,cor,tcor,wa[256],wb[256];
int lcox,lcoy,rcox,rcoy;
```

```
FILE *FP3;
main()
{
 unsigned long k,temp,temp1,Fsize,count;
 int errornumber,x,y,k1,ncox,ncoy;
 int x1,y1,x2,y2,x3,y3;
 int i,j,palette,color;
 char ch,fname1[20],fname2[20],fname3[30],rpict[256],lpict[256];
 FILE *FP1,*FP2;
 char *mes;
 mes = "function main";
 MemAlloc();
 clrscr();
 printf("\nEnter file name(r) : ");
 scanf("%s",fname1);
 FP1 = fopen(fname1,"rb");
 if(FP1 == NULL)
   FopenErr(mes);
 printf("Enter file name(r) : ");
 scanf("%s",fname2);
 FP2 = fopen(fname2,"rb");
 if(FP2 == NULL)
   FopenErr(mes);
 printf("Enter file name(w) : ");
 scanf("%s",fname3);
 FP3 = fopen(fname3,"wb");
 if(FP3 == NULL)
   FopenErr(mes);
  for(y=0;y<SIZE;y++)
    {
    fread(rpict,256,1,FP1);
    fread(lpict,256,1,FP2);
    for(x=0;x<SIZE;x++)
      {
       img6[y][x] = rpict[x];
       img7[y][x] = lpict[x];
```

```
    }
  }
init();
for(y=0;y<SIZE;y++)
  for(x=0;x<SIZE;x++)
    {
        putpixel(x+50,y+50,img6[y][x]/4+64);
        putpixel(x+310,y+50,img7[y][x]/4+64);
    }
do
{
  MovCurLeft();
  MovCurRight();
} while(0);
x1 = lcox;y1=lcoy;
x2 = rcox;y2 = rcoy;
avr1 = mean1();
printf("\nSearching Max. Correlation");
for(j=0;j<16;j++)
  for(i=0;i<16;i++)
    {
          avr2 = mean2(i,j);
          cor  = corragl(i,j);
          if(cor > tcor)
            {
             tcor = cor;
             x3 = i;
             y3 = j;
            }
    }
ncox = x2+x3;
ncoy = y2+y3;
printf("\nLeft image X : %d Y : %d Right image X : %d Y : %d",x1,y1,x2,y2);
printf("\nX : %d Y : %d Max. Corr : %f",x3,y3,tcor);
printf("\nNew X : %d Y : %d ",ncox,ncoy);
```

```
  getch();
  dispcorr(x3,y3);
  getch();
  CloseGraph();
}

/**********************/
/*  Memory alloction  */
/**********************/
void MemAlloc()
{
  int i;
  char *mes;
  mes = "function memory allocation";
  if ((img6 = (char **)malloc(sizeof(char *)*256)) == NULL)
     MemErr(mes);
  for(i=0;i<SIZE;i++)
    {
      if ((img6[i] = (char *) calloc(256,sizeof(char))) == NULL)
            MemErr(mes);
    }
  if ((img7 = (char **)malloc(sizeof(char *)*256)) == NULL)
            MemErr(mes);
  for(i=0;i<SIZE;i++)
    {
      if ((img7[i] = (char *) calloc(256,sizeof(char))) == NULL)
            MemErr(mes);
    }
  if ((lsubimg = (char **)malloc(sizeof(char *)*LSUBSIZE)) == NULL)
            MemErr(mes);
  for(i=0;i<LSUBSIZE;i++)
    {
      if ((lsubimg[i] = (char *) calloc(LSUBSIZE,sizeof(char))) == NULL)
            MemErr(mes);
    }
```

```
  if ((rsubimg = (char **)malloc(sizeof(char *)*RSUBSIZE)) == NULL)
            MemErr(mes);
  for(i=0;i<RSUBSIZE;i++)
    {
      if ((rsubimg[i] = (char *) calloc(RSUBSIZE,sizeof(char))) == NULL)
            MemErr(mes);
    }
  if ((pict = (unsigned char *) farmalloc(65536)) == NULL)
            MemErr(mes);
}

void init()
{
 int i,j;

 initgraph(&graphdriver,&graphmode,driverparth);
 for(i=64;i<128;i++){
     j = i<<2;
     setrgbpalette(i,j,j,j);
    }
}

void CloseGraph()
{
  closegraph();
  exit(1);
}

void DrawCur(double x1,double y1)
{
 int i,dot;

 for(i=0;i<10;i++)
   {
    dot = 0x0F-getpixel(x1,y1+i);
```

```
    putpixel(x1,y1+i,dot);
    dot = 0x0F-getpixel(x1,y1-i);
    putpixel(x1,y1-i,dot);
    dot = 0x0F-getpixel(x1+i,y1);
    putpixel(x1+i,y1,dot);
    dot = 0x0F-getpixel(x1-i,y1);
    putpixel(x1-i,y1,dot);
   }
return;
}

void DrawRecLeft(double x1,double y1)
{
 int i,dot;
 for(i=0;i<5;i++)
   {
    dot = 0x0F-getpixel(x1-16+i,y1-16);
    putpixel(x1-16+i,y1-16,dot);
    dot = 0x0F-getpixel(x1-16,y1-16+i);
    putpixel(x1-16,y1-16+i,dot);
    dot = 0x0F-getpixel(x1+16-i,y1-16);
    putpixel(x1+16-i,y1-16,dot);
    dot = 0x0F-getpixel(x1+16,y1-16+i);
    putpixel(x1+16,y1-16+i,dot);
    dot = 0x0F-getpixel(x1-16+i,y1+16);
    putpixel(x1-16+i,y1+16,dot);
    dot = 0x0F-getpixel(x1-16,y1+16-i);
    putpixel(x1-16,y1+16-i,dot);
    dot = 0x0F-getpixel(x1+16-i,y1+16);
    putpixel(x1+16-i,y1+16,dot);
    dot = 0x0F-getpixel(x1+16,y1+16-i);
    putpixel(x1+16,y1+16-i,dot);
   }
}
```

```
void DrawRecRight(double x1,double y1)
{
 int i,dot;
 for(i=0;i<5;i++)
   {
    dot = 0x0F-getpixel(x1-32+i,y1-32);
    putpixel(x1-32+i,y1-32,dot);
    dot = 0x0F-getpixel(x1-32,y1-32+i);
    putpixel(x1-32,y1-32+i,dot);
    dot = 0x0F-getpixel(x1+32-i,y1-32);
    putpixel(x1+32-i,y1-32,dot);
    dot = 0x0F-getpixel(x1+32,y1-32+i);
    putpixel(x1+32,y1-32+i,dot);
    dot = 0x0F-getpixel(x1-32+i,y1+32);
    putpixel(x1-32+i,y1+32,dot);
    dot = 0x0F-getpixel(x1-32,y1+32-i);
    putpixel(x1-32,y1+32-i,dot);
    dot = 0x0F-getpixel(x1+32-i,y1+32);
    putpixel(x1+32-i,y1+32,dot);
    dot = 0x0F-getpixel(x1+32,y1+32-i);
    putpixel(x1+32,y1+32-i,dot);
   }
}

void MovCurLeft()
{
 register i,j;
 int xd,yd,mask,x,y=0;
 int step=10;
 int x1 = 100;
 int y1 = 100;
 char key = 0;
 char key1,lx,ly;
```

```
while(key != 13)
  {
  DrawCur(x1,y1);
  DrawRecLeft(x1,y1);
  x = x1-16;
  y = y1-16;
  x = x - 50;
  y = y - 50;
  for(j=0;j<20;j++)
    for(i=0;i<100;i++)
      putpixel(50+i,340+j,0);
  itoa(x,lx,10);
  itoa(y,ly,10);
  outtextxy(55,350,lx);
  outtextxy(85,350,ly);
  key = getch();
  DrawCur(x1,y1);
  DrawRecLeft(x1,y1);
  switch(key)
    {
    case 0:
            key1 = getch();
            switch(key1)
              {
              case 59 :  /* F1 */
                      step = 1;
                      break;
              case 60 :  /* F2 */
                      step = 10;
                      break;
              case 61 :  /* F3 */
                      step = 20;
                      break;
              case 71 :  /* HOME */
                      y1 = 30;
```

```
                x1 = 5;
                break;
        case 72 :  /* ^ */
                y1 = y1-step;
                if(y1<66)
                  y1 = 66;
                break;
        case 75 :  /* < */
                x1 = x1-step;
                if(x1<66)
                  x1 = 66;
                break;
        case 77 :  /* > */
                x1 = x1+step;
                if(x1>=SIZE+32)
                  x1 = SIZE+32;
                break;
        case 79 :  /* END */
                y1 = SIZE+66;
                x1 = SIZE+16;
                break;
        case 80 :  /* v */
                y1 = y1+step;
                if(y1>=SIZE+32)
                  y1 = SIZE+32;
                break;
        }
  break;
  case 27 :
        SubImgLeft(x1,y1);
        break;
  case 0xd:
        break;
  }
}
```

```
 return;
}

void MovCurRight()
{
 register i,j;
 int xd,yd,mask,x,y=0;
 int step=10;
 int x1 = 100;
 int y1 = 100;
 char key = 0;
 char key1,rx,ry;
 while(key != 13)
   {
   DrawCur(x1+310,y1);
   DrawRecRight(x1+310,y1);
   x = x1-32;
   y = y1-32;
   y = y-50;
   for(j=0;j<20;j++)
     for(i=0;i<100;i++)
       putpixel(310+i,340+j,0);
   itoa(x,rx,10);
   itoa(y,ry,10);
   outtextxy(320,350,rx);
   outtextxy(350,350,ry);
   key = getch();
   DrawCur(x1+310,y1);
   DrawRecRight(x1+310,y1);
   switch(key)
     {
     case 0:
              key1 = getch();
              switch(key1)
                {
```

```
        case 59 :  /* F1 */
                   step = 1;
                   break;
        case 60 :  /* F2 */
                   step = 10;
                   break;
        case 61 :  /* F3 */
                   step = 20;
                   break;
        case 71 :  /* HOME */
                   closegraph();
                   exit(1);
                   break;
        case 72 :  /* ^ */
                   y1 = y1-step;
                   if(y1<82)
                     y1 = 82;
                   break;
        case 75 :  /* < */
                   x1 = x1-step;
                   if(x1<32)
                     x1 = 32;
                   break;
        case 77 :  /* > */
                   x1 = x1+step;
                   if(x1>=SIZE-32)
                     x1 = SIZE-32;
                   break;
        case 80 :  /* v */
                   y1 = y1+step;
                   if(y1>=SIZE+32)
                     y1 = SIZE+32;
                   break;
        }
break;
```

```
            case 27 :
                    SubImgRight(x1,y1);
                break;
            case 0xd:
                    closegraph();
                break;
      }
    }
 return;
}

void SubImgLeft(int x1,int y1)
{
 register i,j;
 int x,y;
 x = x1-16;
 y = y1-16;
 x = x - 50;
 y = y - 50;
 lcox = x;
 lcoy = y;
 for(j=0;j<LSUBSIZE;j++)
   for(i=0;i<LSUBSIZE;i++)
     {
       lsubimg[j][i] = img6[j+y][i+x];
       putpixel(i+200,j+SIZE+100,lsubimg[j][i]/4+64);
     }
}
void SubImgRight(int x1,int y1)
{
 register i,j;
 int x,y;
 x = x1-32;
 y = y1-32;
 y = y-50;
```

```
rcox = x;
rcoy = y;
for(j=0;j<RSUBSIZE;j++)
  for(i=0;i<RSUBSIZE;i++)
    {
      rsubimg[j][i] = img7[j+y][i+x];
      putpixel(i+460,j+SIZE+100,rsubimg[j][i]/4+64);
    }
}
float mean1()
{
  int i,j;
  float sum = 0.0;
  for(i=0;i<LSUBSIZE;i++)
    for(j=0;j<LSUBSIZE;j++)
      sum = lsubimg[i][j]*1.0 + sum;
  return( sum / (LSUBSIZE*LSUBSIZE) );
}

float  mean2(int n,int m)
{
  int i,j;
  float sum = 0.0;
  for(i=0;i<LSUBSIZE;i++)
    for(j=0;j<LSUBSIZE;j++)
      sum = rsubimg[i+m][j+n]*1.0 + sum;
  return( sum / (LSUBSIZE*LSUBSIZE) );
}

float corragl(int n,int m)
{
   int i,j;
   long dx=0L,dy=0L,num=0L;
   long denum1=0L,denum2=0L;
   float denum11=0.0,denum12=0.0,denum13=0.0,co = 0.0;
```

```
    for(i=0;i<LSUBSIZE;i++)
      for(j=0;j<LSUBSIZE;j++)
        {
            num    = ((lsubimg[i][j]*1.0-avr1)*(rsubimg[i+m][j+n]*1.0-avr2)) + num;
            denum1 = pow(lsubimg[i][j]*1.0    -avr1,2) + denum1;
            denum2 = pow(rsubimg[i+m][j+n]*1.0-avr2,2) + denum2;
        }
    denum11 = sqrt((float)denum1);
    denum12 = sqrt((float)denum2);
    denum13 = (denum11*denum12);
    dx = num;
    dy = denum13;
    co = dx*1.0/dy*1.0;
    return(co);
}
void dispcorr(int sx,int sy)
{
  int i,j;
  init();
  for(j=0;j<RSUBSIZE;j++)
    for(i=0;i<RSUBSIZE;i++)
      putpixel(i,j+200,rsubimg[j][i]/4+64);
  for(j=0;j<LSUBSIZE;j++)
    for(i=0;i<LSUBSIZE;i++)
      putpixel(i+sx,j+sy+200,lsubimg[j][i]/4+62);
  getch();
  closegraph();
}
void MemErr(char *m)
{
 clrscr();
 printf("\nmemory allocation error in %s",m);
 getch();
 exit(1);
}
```

```
void FopenErr(char *m)
{
 clrscr();
 printf("\nfile open error in %s",m);
 getch();
 exit(1);
}
```

```
/*****************************/
/* Program : brightness matching */
/* Author : Chitwong.S          */
/* Date : 2536                  */
/*****************************/
#include<graphics.h>
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<alloc.h>
#define SIZE 256
void MemAlloc();
void init();
void CloseGraph();
int meanl();
int meanr();
float stdl();
float stdr();
void cal_bri();
MemErr();
FopenErr();
int graphdriver=IBM8514,graphmode=IBM8514LO;
char *driverparth= "";
unsigned char far *pict;
unsigned char **img6,**img7;
main()
{
 unsigned long k,temp,temp1,Fsize,count;
 int errornumber,x,y,k1;
 int i,j,palette,color;
 char ch,fname1[20],fname2[20],fname3[20];
 FILE *FP1,*FP2,*FP3;

 char *mes;
 mes = "function main";
```

```
MemAlloc();
clrscr();
printf("\nEnter image file 1(R) : ");
scanf("%s",fname1);
printf("Enter image file 2(R) : ");
scanf("%s",fname2);
printf("Enter image file 3(W) : ");
scanf("%s",fname3);
FP1 = fopen(fname1,"rb");
FP2 = fopen(fname2,"rb");
FP3 = fopen(fname3,"wb");
if(FP1 == NULL && FP2 == NULL && FP3 == NULL)
  FopenErr(mes);
 k=0;
 Fsize=65536L;
 temp =32768L;
 do
  {
   fread(pict+k,1,temp,FP1);
   temp1 = Fsize-temp;
   Fsize = temp1;
   k += temp;
   if(Fsize < 32768L)
     temp = temp1;
  } while(!(temp1 <= 0));

 count=0L;
 for(y=0;y<SIZE;y++)
   for(x=0;x<SIZE;x++,count++)
     img6[y][x] = pict[count];

 k=0;
 Fsize=65536L;
 temp =32768L;
 do
  {
```

```
    fread(pict+k,1,temp,FP2);
    temp1 = Fsize-temp;
    Fsize = temp1;
    k += temp;
    if(Fsize < 32768L)
     temp = temp1;
   } while(!(temp1 <= 0));

  count=0L;
  for(y=0;y<SIZE;y++)
    for(x=0;x<SIZE;x++,count++)
      img7[y][x] = pict[count];
  farfree(pict);
  cal_bri(FP3);
  getch();
}

/*********************/
/* memory allocation */
/*********************/
void MemAlloc()
{
  int i;
  char *mes;
  mes = "function memory allocation";



  if ((img6 = (char **)malloc(sizeof(char *)*256)) == NULL)
     MemErr(mes);
  for(i=0;i<SIZE;i++)
    {
      if ((img6[i] = (char *) calloc(256,sizeof(char))) == NULL)
            MemErr(mes);
    }
  if ((img7 = (char **)malloc(sizeof(char *)*256)) == NULL)
```

```
            MemErr(mes);
  for(i=0;i<SIZE;i++)
    {
      if ((img7[i] = (char *) calloc(256,sizeof(char))) == NULL)
            MemErr(mes);
    }
  if ((pict = (unsigned char *) farmalloc(65536)) == NULL)
            MemErr(mes);
}

void init()
{
 int i,j;
 initgraph(&graphdriver,&graphmode,driverparth);
 for(i=64;i<128;i++){
     j = i<<2;
     setrgbpalette(i,j,j,j);
    }
}
void CloseGraph()
{
  closegraph();
  exit(1);
}

int  meanl()
{
  int i,j;
  long msuml = 0L;
  for(i=0;i<SIZE;i++)
    for(j=0;j<SIZE;j++)
      msuml = img6[i][j] + msuml;
  return( msuml / 65536L );
}

int  meanr()
```

```
{
  int i,j;
  long msumr = 0L;
  for(i=0;i<SIZE;i++)
    for(j=0;j<SIZE;j++)
      msumr = img7[i][j] + msumr;
  return( msumr / 65536L );
}

float stdl(int ml)
{
  int x,y;
  long ssuml = 0L;
  for(y=0;y<SIZE;y++)
    for(x=0;x<SIZE;x++)
      ssuml = ((img6[y][x]-ml)*(img6[y][x]-ml)) + ssuml;
  return(sqrt(ssuml / 65536L));}

float stdr(int mr)
{
  int x,y;

  long ssumr = 0L;
  for(y=0;y<SIZE;y++)
    for(x=0;x<SIZE;x++)
      ssumr = ((img7[y][x]-mr)*(img7[y][x]-mr)) + ssumr;
  return(sqrt(ssumr / 65536L));
}

void cal_bri(FILE *FP)
{
 int x,y;
 int mr,ml = 0;
 float stddl,stddr = 0.0;
 unsigned char newpixel;
 ml  = meanl();  mr = meanr();
```

```
 stddl= stdl(ml); stddr= stdr(mr);
 init();
 for(y=0;y<256;y++)
   for(x=0;x<256;x++)
     {
       putpixel(x    ,y+50,img6[y][x]/4+64);
       putpixel(x+257,y+50,img7[y][x]/4+64);
     }
 getch();
 for(y=0;y<256;y++)
   for(x=0;x<256;x++)
     {
       newpixel = ((stddr/stddl)*img6[y][x] + mr - (stddr/stddl)*ml);
       putpixel(x,y+50,newpixel/4+64);
       img7[y][x] = newpixel;
     }
   for(y=0;y<256;y++)
   fwrite(img7[y],1,256,FP);
 getch();
 closegraph();
}
void MemErr(char *m)
{
 clrscr();
 printf("\nmemory allocation error in %s",m);
 getch();
 exit(1);
}

void FopenErr(char *m)
{
 clrscr();
 printf("\nfile open error in %s",m);
 getch();
 exit(1);
}
```

```
/*****************************/
/* Program : averaging mosaicking */
/* Author :  Chitwong.S          */
/*****************************/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <math.h>
#include <io.h>
#include <stdlib.h>
#include <fcntl.h>
#include "supframe.h"
#define CLS()         printf("\033[2J")
#define COLOR(n)  printf("\033[%dm",n)
#define OFFSET 3
#define BLACK      30
#define RED        31
#define GREEN      32
#define YELLOW     33
#define BLUE       34
#define PURPLE     35
#define CYAN       36
#define WHITE      37
#define SIZE 256
void MemAlloc();
void Allocoverlap();
void seamremoval(int x1,int y1,int x2,int y2);
void genlin();
void avggray();
void saveimg();
void MemErr();
void FopenErr();
unsigned char **img6,**img7,**limg,**rimg;
float *wa1,*wa2;
```

```
long overlaprow,overlapcol;
int x3,y3;
main()
{
 int x,y,ncox,ncoy;
 int x1,y1,x2,y2,x3,y3;
 int i,j;
 char ch,fname1[20],fname2[20],fname3[30],rpict[256],lpict[256];
 char *mes;
 FILE *FP1,*FP2,*FP3;
 MemAlloc();
 clrscr();
 mes ="function main" ;
 printf("\nEnter file 1 name(r) : ");
 scanf("%s",fname1);
 FP1 = fopen(fname1,"rb");
 if(FP1 == NULL)
   FopenErr(mes);
 printf("Enter file 2 name(r) : ");
 scanf("%s",fname2);
 FP2 = fopen(fname2,"rb");
 if(FP2 == NULL)
   FopenErr(mes);
 printf("Enter file name(w) : ");
 scanf("%s",fname3);
 FP3 = fopen(fname3,"wb");
 if(FP3 == NULL)
   FopenErr(mes);
  for(y=0;y<SIZE;y++)
    {
    fread(rpict,256,1,FP1);
    fread(lpict,256,1,FP2);
    for(x=0;x<SIZE;x++)
      {
       img6[y][x] = rpict[x];
```

```
        img7[y][x] = lpict[x];
      }
    }
  fclose(FP1);
  fclose(FP2);
  x1 = 185;y1 = 124;
  ncox = 40;  ncoy = 182;
  seamremoval(x1,y1,ncox,ncoy);
  saveimg(FP3);
  fclose(FP3);
}
/*********************************************************/
/*  MemAlloc - Memory alloction for first & second image */
/*********************************************************/
void MemAlloc()
{
  int i;
  char *mes;
  mes = "function memory allocaton";
  if ((img6 = (char **)malloc(sizeof(char *)*256)) == NULL)
     MemErr(mes);
  for(i=0;i<SIZE;i++)
    {
      if ((img6[i] = (char *) calloc(256,sizeof(char))) == NULL)
            MemErr(mes);
    }
  if ((img7 = (char **)malloc(sizeof(char *)*256)) == NULL)
     MemErr(mes);
  for(i=0;i<SIZE;i++)
    {       if ((img7[i] = (char *) calloc(256,sizeof(char))) == NULL)
            {
              MemErr(mes);
            }
    }
}
```

```
void Allocoverlap(overlaprow,overlapcol)
long int overlaprow,overlapcol;
{
 int i,j;
 char *mes;
 mes = "function memory allocation for overlap";
  if ((rimg = (char **)malloc(sizeof(char *)*overlapcol)) == NULL)
     MemErr(mes);
  for(i=0;i<overlapcol;i++)
    {
      if ((rimg[i] = (char *) calloc(overlaprow,sizeof(char))) == NULL)
         MemErr(mes);
    }
  if ((limg = (char **)malloc(sizeof(char *)*overlapcol)) == NULL)
     MemErr(mes);
  for(i=0;i<overlapcol;i++)
    {
      if ((limg[i] = (char *) calloc(overlaprow,sizeof(char))) == NULL)
            MemErr(mes);
    }
wa1 = (float *) calloc(overlapcol,sizeof(float));
wa2 = (float *) calloc(overlapcol,sizeof(float));
if (wa1 == NULL && wa2 == NULL)
  MemErr(mes);
}

/*******************************/
/*   aic image horizontal function     */
/*******************************/
void seamremoval(x1,y1,x2,y2)
int x1,y1,x2,y2;
{
 int i,j,x,y,c;
 int mr=0,ml=0,mc = 0;
 float stddl=0.0,stddr=0.0;
```

```
unsigned char newpixel;
x3 = x1-x2;y3 = y1-y2;
overlaprow = SIZE-x3;
overlapcol = SIZE-abs(y3);
Allocoverlap(overlaprow,overlapcol);
for(y=0;y<SIZE;y++)
  for(x=0;x<SIZE;x++)
    {
      spset(x,y+50,img6[y][x],img6[y][x],img6[y][x]);
      spset(x+x3,y+y3+50,img7[y][x],img7[y][x],img7[y][x]);
    }
for(y=0;y<overlapcol;y++)
  for(x=0;x<overlaprow;x++)
    {
        limg[y][x] = img6[y][SIZE-overlaprow+x];
        rimg[y][x] = img7[y+abs(y3)][x];
        spset(x+400,y+50,limg[y][x],limg[y][x],limg[y][x]);
        spset(x+400,y+50+200,rimg[y][x],rimg[y][x],rimg[y][x]);
    }
printf("\nGenerate linear fn...");
genlin();
printf("\nAverage gray scale...");
avggray();
printf("\nPress any key...");
getch();
for(y=0;y<SIZE;y++)
  for(x=0;x<SIZE;x++)
    {
      spset(x,y+50,img6[y][x],img6[y][x],img6[y][x]);
      spset(x+x3,y+y3+50,img7[y][x],img7[y][x],img7[y][x]);
    }
}
```

```
void genlin()
{
 int i;
 for(i=0;i<overlapcol;i++)
   {
    wa1[i] = (1.0/overlapcol)*i;;
    wa2[i] = 1-wa1[i];
   }
}
void avggray()
{
 int y,x;
 for(y=0;y<overlapcol;y++)
   for(x=0;x<overlaprow;x++)
      {
       limg[y][x] = wa1[x]*img6[y][SIZE-overlaprow+x] + wa2[x]*img7[y+abs(y3)][x];
       rimg[y][x] = wa1[x]*img6[y][SIZE-overlaprow+x] + wa2[x]*img7[y+abs(y3)][x];
      }
 for(y=0;y<overlapcol;y++)
   for(x=0;x<overlaprow;x++)
     {
      img7[y+abs(y3)][x] = limg[y][x];
      img6[y][SIZE-overlaprow+x]= rimg[y][x];
     }
}

void saveimg(FILE *fp)
{
  register x,y;
  unsigned char red[640];
  int c;
  for(y=0;y<400;y++)
    {
     c = 0;
     for(x=0;x<640;x++,c++)
```

```
        red[c] = redget(x,y);
      fwrite(red,640,1,fp);
     }
}
void MemErr(char *m)
{
 clrscr();
 printf("\nmemory allocation error in %s",m);
 getch();
 exit(1);
}

void FopenErr(char *m)
{
 clrscr();
 printf("\nfile open error in %s",m);
 getch();
 exit(1);
}
```

```
/***********************************/
/* Program : linear function mosaicking  */
/* Author : Chitwong.S                    */
/* Date : 2536                            */
/***********************************/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <math.h>
#include <io.h>
#include <stdlib.h>
#include <fcntl.h>
#include "supframe.h"
#define CLS()          printf("\033[2J")
#define COLOR(n)  printf("\033[%dm",n)
#define SIZE       256
#define MSIZE      7
#define OFFSETH    10
#define OFFSETV    100
#define GOFF       1
#define BLACK      30
#define RED        31
#define GREEN      32
#define YELLOW     33
#define BLUE       34
#define PURPLE     35
#define CYAN       36
#define WHITE      37
void getimg(FILE *RFP);
void dispimg();
void vgenbut();

void genlinh();
void genlinv();
```

```
void hlin();
void vlin();
void vlinp();
void MemErr();
void FopenErr();
unsigned char **rimg,item[25];
float wah[OFFSETH];
float wav[OFFSETV];
main()
{
 register i,j;
 int x,y,cx1,cy1,cx2,cy2,cx3,cy3;
 int xstart,ystart,xstop,ystop,xaxis,yaxis=0;
 FILE *RFP1,*RFP2,*RFP3,*RMFP;
 char rfname1[20],rfname2[20],rfname3[20],sfname[20];
 char *mes;
 mes = "function main";
  CLS();
  rimg=(char **) malloc(sizeof(char *)*SIZE);
  if(rimg == NULL)
    MemErr(mes);
  for(y=0;y<SIZE;y++)
    {
      rimg[y]=(char *) malloc(sizeof(char)*SIZE);
      if(rimg[y] == NULL)
            MemErr(mes);
    }
  printf("Enter file image 1\n");
  COLOR(RED);
  printf("Enter Red file   :");
  COLOR(WHITE);
  scanf("%s",rfname1);
  RFP1=fopen(rfname1,"rb");
  if(RFP1 == NULL)
    FopenErr(mes);
```

```
printf("Enter file image 2\n");
COLOR(RED);
printf("Enter Red file   :");
COLOR(WHITE);
scanf("%s",rfname2);
RFP2=fopen(rfname2,"rb");
if(RFP2 == NULL)
  FopenErr(mes);
COLOR(GREEN);
printf("Enter file name for save screen : ");
COLOR(RED);
scanf("%s",sfname);
RMFP=fopen(sfname,"wb");
if(RMFP == NULL)
  FopenErr(mes);
CLS();
COLOR(WHITE);
initsf(BOTH);
cx2=145;cy2=-7;
dispimg(RFP2,cx2,cy2);
cx1=0;cy1=50;
dispimg(RFP1,cx1,cy1);
CLS();
genlinh();
genlinv();
xstart = cx2+100;
xstop  = cx1+256+100;
yaxis  = cy1;
hlin(xstart,xstop,yaxis);
ystart = cy1;
ystop  = cy2+256;
xaxis  = cx1+256+100;
vlin(ystart,ystop,xaxis);
getimg(RMFP);
}
```

```
void getimg(FILE *RFP)
{
  register x,y;
  int c;
  unsigned char red[640];
  for(y=0;y<400;y++)
    {
     c = 0;
     for(x=0;x<640;x++,c++)
          red[c]  =redget(x,y);
     fwrite(red,640,1,RFP);
    }
}
void dispimg(FILE *RFP,int cx,int cy)
{
 register i,j,x,y;
 unsigned char rpict[256];
 for(y=0;y<SIZE;y++)
   {
    fread(rpict,256,1,RFP);
    for(x=0;x<SIZE;x++)
       rimg[y][x] = rpict[x];
   }

 for(j=0;j<SIZE;j++)
   for(i=0;i<SIZE;i++)
       spset(i+cx+100,j+cy,rimg[j][i],rimg[j][i],rimg[j][i]);
}
/*************************/
/*  Generated linear function */
/*************************/
void genlinh()
{
 register i;
 for(i=0;i<OFFSETH;i++)
```

```
    wah[i] = (1/(OFFSETH*1.0))*i;
}

void genlinv()
{
 register i;
 for(i=0;i<OFFSETV;i++)
    wav[i] = (1.0/(OFFSETV*1.0))*i;
}
/*********************************/
/*          Horizontal linear function      */
/*********************************/
void hlin(xstart,xstop,yaxis)
int xstart,xstop,yaxis;
{
 register i,j;
 float meanl=0.0,meanu=0.0,xmean=0.0;
 unsigned int ru,rl,wbsize;
 float difu=0.0,difl=0.0;
 unsigned int sumu=0,suml=0;
 unsigned char pixell,pixelu;
 for(i=xstart;i<xstop;i++)
    {
     wbsize = OFFSETH;
     sumu=0,suml=0;
     for(j=1;j<wbsize;j++)
         {
           ru = redget(i,yaxis-j);
           rl = redget(i,yaxis+j);
           if(rl < 9 || ru < 9)
              {
               wbsize = j-1;
               if(wbsize < 1)
                    wbsize = 1;
               rl = 0;
```

```
                ru = 0;
               }
             suml = rl+suml;
             sumu = ru+sumu;
           }
       meanu = sumu/(wbsize*1.0);
       meanl = suml/(wbsize*1.0);
       xmean = (meanu+meanl)/2.0;
       difl  = xmean-meanl;
       difu  = xmean-meanu;
       COLOR(RED);
       for(j=1;j<wbsize;j++)
            {
              ru=redget(i,yaxis-j);
              rl=redget(i,yaxis+j);
              pixell = rl+(wah[j-1]*difl);
              pixelu = ru+(wah[j-1]*difu);
              spset(i,yaxis-j,pixelu,pixelu,pixelu);
              spset(i,yaxis+j,pixell,pixell,pixell);
              ru=redget(i,yaxis-1);
              rl=redget(i,yaxis+1);
              pixell = (rl+ru)/2;
              spset(i,yaxis,pixell,pixell,pixell);
            }
     }
}

/*********************************/
/*          Vertical linear function          */
/*********************************/
void vlin(ystart,ystop,xaxis)
int ystart,ystop,xaxis;
{
 register i,j;
 float meanl=0.0,meanr=0.0,xmean=0.0;
```

```
unsigned int rr,rl,wbsize;
float difr=0.0,difl=0.0;
unsigned int sumr,suml;
unsigned char pixell,pixelr;
  for(i=ystart;i<ystop;i++)
    {
     wbsize = OFFSETV;
     sumr=0,suml=0;
     for(j=0;j<15;j++)
          {
           rl = redget(xaxis-j,i);
           suml = rl + suml;
          }
     for(j=1;j<100;j++)
          {
           rr = redget(xaxis+j,i);
           sumr = rr + sumr;
          }
     meanr = (sumr*1.0)/100.0;
     meanl = (suml*1.0)/15.0;
     xmean = (meanr+meanl)/2.0;
     difl  = xmean-meanl;
     difr  = xmean-meanr;
     COLOR(RED);
     for(j=0;j<15;j++)
          {
            rl=redget(xaxis-j,i);
            pixell = rl+(wav[j]*difl)+0.5999;
            spset(xaxis-j,i,pixell,pixell,pixell);
          }
      for(j=0;j<100;j++)
          {
            rr=redget(xaxis+j,i);
            pixelr = rr+(wav[j]*difr)+0.5999;
            spset(xaxis+j-1,i,pixelr,pixelr,pixelr);
```

```
            }
        }
}

void FopenErr(char *m)
{
  clrscr();
  printf("\nfile open error in %s",m);
  getch();
  exit(1);
}

void MemErr(char *m)
{
  clrscr();
  printf("\nmemory allocation error in %s",m);
  getch();
  exit(1);
}
```

```
/*************************************/
/* Program : non-linear function mosaicking */
/* Author : Chitwong.S                */
/* Date : 2536                        */
/*************************************/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <math.h>
#include <io.h>
#include <stdlib.h>
#include <fcntl.h>
#include "supframe.h"
#define CLS()          printf("\033[2J")
#define COLOR(n)  printf("\033[%dm",n)
#define SIZE       256
#define MSIZE      7
#define OFFSETH    10
#define OFFSETV    100
#define GOFF       1
#define BLACK      30
#define RED        31
#define GREEN      32
#define YELLOW     33
#define BLUE       34
#define PURPLE     35
#define CYAN       36
#define WHITE      37

void getimg(FILE *RFP);
void dispimg();
void vgenbut();
void genbuth();
void genbutv();
```

```
void hbutterworth();
void vbutterworth();
void vbutterworthp();
void MemErr();
void FopenErr();
unsigned char **rimg,item[25];
float wah[OFFSETH];
float wav[OFFSETV];
main()
{
 register i,j;
 int x,y,cx1,cy1,cx2,cy2,cx3,cy3;
 int xstart,ystart,xstop,ystop,xaxis,yaxis=0;
 FILE *RFP1,*RFP2,*RFP3,*RMFP;
 char rfname1[20],rfname2[20],rfname3[20],sfname[20];
 char *mes;
 mes = "function main";
  CLS();
  rimg=(char **) malloc(sizeof(char *)*SIZE);
  if(rimg == NULL)
     MemErr(mes);
  for(y=0;y<SIZE;y++)
    {
      rimg[y]=(char *) malloc(sizeof(char)*SIZE);
      if(rimg[y] == NULL)
            MemErr(mes);
    }
  printf("Enter file image 1\n");
  COLOR(RED);
  printf("Enter Red file   :");
  COLOR(WHITE);
  scanf("%s",rfname1);
  RFP1=fopen(rfname1,"rb");
  if(RFP1 == NULL)
    FopenErr(mes);
```

```
printf("Enter file image 2\n");
COLOR(RED);
printf("Enter Red file   :");
COLOR(WHITE);
scanf("%s",rfname2);
RFP2=fopen(rfname2,"rb");
if(RFP2 == NULL)
   FopenErr(mes);
COLOR(GREEN);
printf("Enter file name for save screen : ");
COLOR(RED);
scanf("%s",sfname);
RMFP=fopen(sfname,"wb");
if(RMFP == NULL)
  FopenErr(mes);
CLS();
COLOR(WHITE);
initsf(BOTH);
cx2=145;cy2=-7;
dispimg(RFP2,cx2,cy2);
cx1=0;cy1=50;
dispimg(RFP1,cx1,cy1);
CLS();
genbuth();
genbutv();
xstart = cx2+100;
xstop  = cx1+256+100;
yaxis  = cy1;
hbutterworth(xstart,xstop,yaxis);
ystart = cy1;
ystop  = cy2+256;
xaxis  = cx1+256+100;
vbutterworth(ystart,ystop,xaxis);
getimg(RMFP);
}
```

```
void getimg(FILE *RFP)
{
  register x,y;
  int c;
  unsigned char red[640];
  for(y=0;y<400;y++)
    {
     c = 0;
     for(x=0;x<640;x++,c++)
          red[c]  =redget(x,y);
     fwrite(red,640,1,RFP);
    }
}

void dispimg(FILE *RFP,int cx,int cy)
{
 register i,j,x,y;
 unsigned char rpict[256];
 for(y=0;y<SIZE;y++)
   {
    fread(rpict,256,1,RFP);
    for(x=0;x<SIZE;x++)
       rimg[y][x] = rpict[x];
   }
 for(j=0;j<SIZE;j++)
   for(i=0;i<SIZE;i++)
       spset(i+cx+100,j+cy,rimg[j][i],rimg[j][i],rimg[j][i]);
}

/*********************************/
/*       Generated butterworth       */
/*********************************/
void genbuth()
{
 register i;
```

```
float y0,y1,p=0.0;
y0 = 3.0;
for(i=0;i<OFFSETH;i++)
  {
   y1 = i*1.0/y0;
   p = y1*y1;
   wah[i] = (1.0/(1.0+0.414*p));
  }
}

void genbutv()
{
 register i;
 float y0,y1,p=0.0,p1=0.0;
 y0 = 3.0;
 for(i=0;i<OFFSETV;i++)
   {
    y1 = i*1.0/y0;
    p = y1*y1;
    p1 = p*p;
    wav[i] = (1.0/(1.0+0.414*p1));
   }
}

/*********************************/
/*          Horizontal butterworth          */
/*********************************/
void hbutterworth(xstart,xstop,yaxis)
int xstart,xstop,yaxis;
{
 register i,j;
 float meanl=0.0,meanu=0.0,xmean=0.0;
 unsigned int ru,rl,wbsize;
 float difu=0.0,difl=0.0;
 unsigned int sumu=0,suml=0;
```

```
unsigned char pixell,pixelu;
for(i=xstart;i<xstop;i++)
    {
     wbsize = OFFSETH;
     sumu=0,suml=0;
     for(j=1;j<wbsize;j++)
          {
            ru = redget(i,yaxis-j);
            rl = redget(i,yaxis+j);
            if(rl < 9 || ru < 9)
               {
                wbsize = j-1;
                if(wbsize < 1)
                      wbsize = 1;
                rl = 0;
                ru = 0;
               }
            suml = rl+suml;
            sumu = ru+sumu;
          }
     meanu = sumu/(wbsize*1.0);
     meanl = suml/(wbsize*1.0);
     xmean = (meanu+meanl)/2.0;
     difl  = xmean-meanl;
     difu  = xmean-meanu;
     COLOR(RED);
     for(j=1;j<wbsize;j++)
          {
            ru=redget(i,yaxis-j);
            rl=redget(i,yaxis+j);
            pixell = rl+(wah[j-1]*difl);
            pixelu = ru+(wah[j-1]*difu);
            spset(i,yaxis-j,pixelu,pixelu,pixelu);
            spset(i,yaxis+j,pixell,pixell,pixell);
            ru=redget(i,yaxis-1);
```

```
            rl=redget(i,yaxis+1);
            pixell = (rl+ru)/2;
            spset(i,yaxis,pixell,pixell,pixell);
          }
    }
}
/************************************/
/*          Vertical butterworth          */
/************************************/
void vbutterworth(ystart,ystop,xaxis)
int ystart,ystop,xaxis;
{
 register i,j;
 float meanl=0.0,meanr=0.0,xmean=0.0;
 unsigned int rr,rl,wbsize;
 float difr=0.0,difl=0.0;
 unsigned int sumr,suml;
 unsigned char pixell,pixelr;
  for(i=ystart;i<ystop;i++)
   {
    wbsize = OFFSETV;
    sumr=0,suml=0;
    for(j=0;j<15;j++)
          {
           rl = redget(xaxis-j,i);
           suml = rl + suml;
          }
    for(j=1;j<100;j++)
          {
           rr = redget(xaxis+j,i);
           sumr = rr + sumr;
          }
    meanr = (sumr*1.0)/100.0;
    meanl = (suml*1.0)/15.0;
    xmean = (meanr+meanl)/2.0;
    difl = xmean-meanl;
```

```
    difr  = xmean-meanr;
    COLOR(RED);
    for(j=0;j<15;j++)
        {
          rl=redget(xaxis-j,i);
          pixell = rl+(wav[j]*difl)+0.5999;
          spset(xaxis-j,i,pixell,pixell,pixell);
        }
    for(j=0;j<100;j++)
        {
          rr=redget(xaxis+j,i);
          pixelr = rr+(wav[j]*difr)+0.5999;
          spset(xaxis+j-1,i,pixelr,pixelr,pixelr);
        }
   }
}
void MemErr(char *m)
{
  clrscr();
  printf("\nmemory allocation error in %s",m);
  getch();
  exit(1);
}

void FopenErr(char *m)
{
  clrscr();
  printf("\nfile open error in %s",m);
  getch();
  exit(1);
}□
```

```
/*********************************/
/* Program : displpay colour image   */
/* Author : Chitwong.S                */
/* Date : 2536                        */
/*********************************/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <math.h>
#include <io.h>
#include <stdlib.h>
#include <fcntl.h>
#include "supframe.h"
#define CLS()          printf("\033[2J")
#define COLOR(n)  printf("\033[%dm",n)
#define HSIZE 300
#define VSIZE 284
#define OFFSET 1
#define BLACK       30
#define RED         31
#define GREEN       32
#define YELLOW      33
#define BLUE        34
#define PURPLE      35
#define CYAN        36
#define WHITE       37
void ferr();
unsigned char *rimg,*gimg,*bimg;
int  sc;
main()
{
 register x,y;
 FILE *FP1,*FP2,*FP3,*WFP;
 char fname1[20],fname2[20],fname3[20],fname4[20];
```

```
CLS();
 rimg=(char *) malloc(sizeof(char)*HSIZE);
 gimg=(char *) malloc(sizeof(char)*HSIZE);
 bimg=(char *) malloc(sizeof(char)*HSIZE);
 if(rimg == NULL && gimg == NULL  && bimg == NULL)
   {
     printf("memory allocation failure 1\n");
     getch();
     exit(1);
   }
CLS();
COLOR(RED);
printf("Enter Red file   :");
COLOR(WHITE);
scanf("%s",fname1);
FP1=fopen(fname1,"rb");
ferr(FP1);
COLOR(GREEN);
printf("Enter Green file :");
COLOR(WHITE);
scanf("%s",fname2);
FP2=fopen(fname2,"rb");
ferr(FP2);
COLOR(BLUE);
printf("Enter Blue file  :");
COLOR(WHITE);
scanf("%s",fname3);
FP3=fopen(fname3,"rb");
ferr(FP3);
COLOR(RED);
printf("Enter file for SAVE  :");
COLOR(WHITE);
scanf("%s",fname4);
WFP=fopen(fname4,"wb");
ferr(WFP);
```

```
COLOR(RED);
printf("Enter offset scaling  : ");
COLOR(WHITE);
scanf("%d",&sc);
CLS();
for(y=0;y<VSIZE;y++)
  {
   fread(rimg,HSIZE,1,FP1);
   fread(gimg,HSIZE,1,FP2);
   fread(bimg,HSIZE,1,FP3);
    for(x=0;x<HSIZE;x++)
      {
          spset(x,y,rimg[x]*sc,gimg[x]*sc,bimg[x]*sc);
          rimg[x] = rimg[x]*sc;
      }
   fwrite(rimg,HSIZE,1,WFP);
  }
 fclose(FP1);
 fclose(FP2);
 fclose(FP3);
 fclose(WFP);
}
void ferr(FILE *FP)
{
 if(FP==NULL)
   {
          COLOR(RED);
          printf("open file failure\n");
          getch();
          COLOR(WHITE);
          exit(1);
   }
}
```

```
/*******************************************************/
/*      Program :  Principal Component Analysis        */
/*      by      :  Mr.Sakreya Chitwong                  */
/*      year    :  2536                                */
/*******************************************************/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<conio.h>
#include<math.h>
#include<alloc.h>
#include<time.h>
#define NPRINT
#define PSIZE 32768L

void  memalloc();
void  ReadData();
float Mean();
void  MulMatrix();
void  MemErr();
void  FopenErr();
void  ScaleMatrix();
int   JacobiTransform(float **co_var,float **v,int n,float err,int *iter);
void  Projection();
void  Scaling();
void  App();
void  lcs();
int   xsize,ysize,tb;
long  tsize;
char  **fname;
float *mean_val,*std_val,**co_var,**v;
char fn[5],ofn[5],drive[3];
unsigned char far *buff;
int far *ibuff,**ibuff1;
long far *lbuff;
```

```
main()
{
  int i,j,k,n,*iter,converge,max,addi,percent;
  float err,temp,peigenvalue;
  long teigenvalue;
  char *mes,cdrive[3];
  FILE **FP,**fp;
  mes = "Function main";
  clrscr();
  printf("Enter total band : ");
  scanf("%d",&tb);
  memalloc();
  FP = (FILE **) malloc(sizeof(FILE *)*tb);
  fp = (FILE **) malloc(sizeof(FILE *)*tb);
  if(FP == NULL && fp == NULL)
    MemErr(mes);
  for(i=0;i<tb;i++)
    {
      FP[i] = (FILE *) malloc(sizeof(FILE)*tb);
      fp[i] = (FILE *) malloc(sizeof(FILE)*tb);
      if(FP[i] == NULL && fp[i] == NULL)
          MemErr(mes);
    }
  if((fname = (char **)malloc(sizeof(char *)*tb)) == NULL)
     MemErr(mes);
  for(i=0;i<tb;i++)
    if((fname[i] = (char *)malloc(sizeof(char *)*25)) == NULL)
       MemErr(mes);
  for(i=0;i<tb;i++)
    {
     printf("          band %d : ",i+1);
     scanf("%s",fname[i]);
     if((fp[i] = fopen(fname[i],"rb")) == NULL)
          FopenErr(mes);
    }
```

```
printf("Size of datas    :\n");
printf("      X-size     : ");
scanf("%d",&xsize);
printf("      Y-size     : ");
scanf("%d",&ysize);
tsize = (long)ysize*xsize;
printf("Total size       : %ld",tsize);
printf("\nSave result at drive <c:/d:> -> ");
scanf("%s",cdrive);
strcpy(drive,cdrive);
for(i=0;i<tb;i++)
  {
    printf("\nmeaning band %d...",i+1);
    mean_val[i] = Mean(fp[i]);
    printf("%+.3f",mean_val[i]);
    #ifdef PRINT
          fprintf(stdprn,"\n");
          fprintf(stdprn,"mean  ");
          fprintf(stdprn,"%+.3f\t",mean_val[i]);
    #endif
    fclose(fp[i]);
  }
for(i=0;i<tb;i++)
  {
    App(drive,fn,i);
    FP[i] = fopen(fn,"wb");
    if(FP[i] == NULL)
          FopenErr(mes);
    if((fp[i] = fopen(fname[i],"rb")) == NULL)
          FopenErr(mes);
    printf("\nscaling matric band %d...",i+1);
    ScaleMatrix(fp[i],FP[i],i);
    fclose(fp[i]);
    fclose(FP[i]);
  }
```

```
printf("\ncalculating covariance matric...");
MulMatrix(co_var);
printf("\ndisplay covariance matric...");
for(j=0;j<tb;j++)
 {
 printf("\n");
 for(i=0;i<tb;i++)
  {
   printf("%+.3f\t",co_var[j][i]);
   #ifdef PRINT
       fprintf(stdprn,"%+.3f\t",co_var[j][i]);
   #endif
  }
 }
n = tb;
*iter = 0;
err  = 0.0000000001;
printf("\ncalculating eigenvectors...");
JacobiTransform(co_var,v,n,err,iter);
printf("\nBefore sort\n");
printf("\ndisplay eigenvalues...\n");
for(i=0;i<n;i++)
  printf("%+.3f\t",fabs(co_var[i][i]));
printf("\ndisplay eigenvectors ");
for(j=0;j<n;j++)
 {
 printf("\n");
 for(i=0;i<n;i++)
   printf("%+.3f\t",v[j][i]);
 }
for(j=0;j<n;j++)
 {
  temp = co_var[j][j];
  max  = j;
```

```
  for(i=j;i<n-1;i++)
    {
        if(temp < co_var[i+1][i+1])
          {
           temp = co_var[i+1][i+1];
           max  = i+1;
          }
    }
  temp = co_var[j][j];
  co_var[j][j] = co_var[max][max];
  co_var[max][max] = temp;
  for(i=0;i<n;i++)
    {
        temp = v[i][j];
        v[i][j] = v[i][max];
        v[i][max] = temp;
    }
 }
teigenvalue = 0L;
peigenvalue = 0.0;
for(i=0;i<n;i++)
   teigenvalue += (long)co_var[i][i];
printf("\nAfter sort\n");
printf("\ndisplay eigenvalues\n");
for(i=0;i<n;i++)
 {
   peigenvalue = (100.0*co_var[i][i])/teigenvalue;
   printf("%+.3f[%.3f]\t",fabs(co_var[i][i]),fabs(peigenvalue));
   #ifdef PRINT
     fprintf(stdprn,"%+.3f[%.3f]\t",fabs(co_var[i][i]),fabs(peigenvalue));
   #endif
 }
printf("\ndisplay eigenvectors ");
```

```
  for(j=0;j<n;j++)
    {
    printf("\n");
    #ifdef PRINT
      fprintf(stdprn,"\n");
    #endif
    for(i=0;i<n;i++)
      {
       printf("%+.3f\t",v[j][i]);
       #ifdef PRINT
            fprintf(stdprn,"%+.3f\t",v[j][i]);
       #endif
      }
    }
  Projection();
  Scaling();
  printf("\n\nprocessing complete...");
  getch();
}

/* ************************/
/*    Allocation memory      */
/* ********************* */
void memalloc()
{
  int i;
  long psize;
  char *mes;
  mes = "function memalloc";
  psize = PSIZE;
  if((co_var = (float **) malloc(sizeof(float *)*tb)) == NULL)
    MemErr(mes);
  for(i=0;i<tb;i++)
    if((co_var[i] = (float *) malloc(sizeof(float)*tb)) == NULL)
    MemErr(mes);
```

```
 if((v = (float **) malloc(sizeof(float *)*tb)) == NULL)
   MemErr(mes);
 for(i=0;i<tb;i++)
   if((v[i] = (float *) malloc(sizeof(float)*tb)) == NULL)
   MemErr(mes);
 if((mean_val = (float *) malloc(sizeof(float)*tb)) == NULL)
   MemErr(mes);
 if((std_val = (float *) malloc(sizeof(float)*tb)) == NULL)
   MemErr(mes);
if((buff = (unsigned char far *) farmalloc(32768L)) == NULL)
  MemErr(mes);
if((ibuff = (int far *)  farmalloc(65536L)) == NULL)
    MemErr(mes);
if((ibuff1 = (int far **) farmalloc(sizeof(int far *)*tb)) == NULL)
   MemErr(mes);
for(i=0;i<tb;i++)
  if((ibuff1[i] = (int far *) farmalloc(65536L)) == NULL)
    MemErr(mes);
if((lbuff = (long far *) farmalloc(sizeof(long)*tb)) == NULL)
    MemErr(mes);
}

/* ************************ */
/*    Calculation mean value     */
/* ************************ */
float Mean(FILE *fp)
{
 float sum,mean,count;
 unsigned int i,j;
 long size1,size2,psize;
 char *mes;
 mes = "function mean";
 psize = PSIZE;
 size1 = ((long)xsize)*((long)ysize);
 size2 = size1;
```

```
 count = 0.0;
 sum   = 0.0;
 do
  {
   fread(buff,sizeof(unsigned char),psize,fp);
   for(i=0;i<psize && size1 > 0L;i++,size1--,count++)
     sum += (float)buff[i];
   size2 -= (long)psize;
   if(size2 < (long)psize)
     psize = (int)size2;
  }while(size1 > 0L);
 mean = sum/count;
 return(mean);
}

/* ********************/
/*   Scaling matric      */
/* ********************/
void ScaleMatrix(FILE *f,FILE *F,int p)
{
 long i,j;
 long count,size1,size2;
 double stsize;
 long psize;
 char *mes;
 extern float *mean_val;
 mes = "function scaling matric";
 psize = PSIZE;
 count = 0L;
 stsize = tsize;
 stsize = sqrt(stsize);
 size1 = ((long)xsize)*((long)ysize);
 size2 = size1;
 do
  {
```

```
    fread(buff,sizeof(unsigned char),psize,f);
    for(i=0;i<psize && size1 > 0L;i++,size1--,count++)
      ibuff1[0][i] = (int)((int)buff[i] - mean_val[p]);
    fwrite(ibuff1[0],sizeof(int),psize,F);
    size2 -= (long)psize;
    if(size2 < (long)psize)
      psize = (int)size2;
   }while(size1 > 0L);
}
/* ******************************** */
/*   Calculation covariance matric      */
/*    or calculation eigenvalue         */
/* ******************************** */
void MulMatrix(float **a)
{
 unsigned int i,j,k;
 long psize;
 long size1,size2,size3,count;
 float sum;
 char *mes;
 FILE **fp1,**fp2;
 mes = "function multifier matric";
 size1 = ((long)xsize)*((long)ysize);
 psize = PSIZE;
 fp1 = (FILE **) malloc(sizeof(FILE *)*tb);
 fp2 = (FILE **) malloc(sizeof(FILE *)*tb);
 if(fp1 == NULL && fp2 == NULL)
   MemErr(mes);
 for(i=0;i<tb;i++)
   {
   fp1[i] = (FILE *) malloc(sizeof(FILE)*tb);
   fp2[i] = (FILE *) malloc(sizeof(FILE)*tb);
   if(fp1[i] == NULL && fp2[i] == NULL)
     MemErr(mes);
   }
```

```
for(k=0;k<tb;k++)
  {
   App(drive,fn,k);
   fp1[k] = fopen(fn,"rb");
   if(fp1[k] == NULL)
         FopenErr(mes);
  for(j=0;j<tb;j++)
    {
    printf("\nmultifier matric row %d col %d...",j,k);
    count = 0L;
    size2 = size1;
    size3 = size1;
    psize = 32768;
    sum = 0.0;
    App(drive,fn,j);
    fp2[j] = fopen(fn,"rb");
    if(fp2[j] == NULL)
          FopenErr(mes);
    do
         {
          fread(ibuff1[0],sizeof(int),(int)psize,fp1[k]);
          fread(ibuff1[1],sizeof(int),(int)psize,fp2[j]);
          for(i=0;i<psize && size2 > 0L;i++,size2--,count++)
            sum += (float)(ibuff1[0][i]*ibuff1[1][i]);
          size3 -= psize;
          if(size3 < psize)
             psize = size3;
         }while(size2 > 0L);
      a[j][k] = sum;    /*/(tsize-1.0);*/
      printf("%+.3f\t[%ld]",a[j][k],count);
      rewind(fp1[k]);
      fclose(fp2[j]);
    }
   fclose(fp1[k]);
  }
}
```

```
/* ********************************************/
/*   Calculation eigenvectors by Jacobi mathod        */
/* ********************************************/
int JacobiTransform(float **h,float **v,int n,float err,int *iter)
{
  int i,j,m,l,p,q,converge,it;
  float vo,U,Uf,*a,*d,*e;
  float c,s,f,alpha,beta;
  a = (float *) malloc(sizeof(float)*n);
  d = (float *) malloc(sizeof(float)*n);
  e = (float *) malloc(sizeof(float)*n);
  if(a == NULL && d == NULL && e == NULL)
    return(NULL);
  vo = 0.0;
  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      if(i!=j)
          vo += (h[i][j]*h[i][j]);
 for(i=0;i<n;i++)
   for(j=0;j<n;j++)
     if(i==j)
       v[i][j] = 1.0;
     else
       v[i][j] = 0.0;

  U  = sqrt(vo)/n;
  Uf = U*err;
  converge = 0;
  it = 0;
  do
   {
    for(l=0;l<n+1;l++)
      {
      for(m=l+1;m<n;m++)
          {
```

```
if(fabs(h[l][m]) >= U)
  {
   p = l;
   q = m;
   for(i=0;i<n;i++)
    {
     a[i] = h[p][i];
     d[i]  = h[i][p];
     e[i]  = v[i][p];
    }
   f     = h[p][p];
   alpha = (h[p][p] - h[q][q])/2.0;
   beta  = sqrt(h[p][q]*h[p][q] + alpha*alpha);
   c  = sqrt(0.5 + fabs(alpha)/(2*beta));
   s  = (alpha*(-h[p][q]))/(2.0*beta*(fabs(alpha))*c);
   for(j=0;j<n;j++)
    {
     if(j!=p && j!=q)
          {
           h[p][j] = c*h[p][j] - s*h[q][j];
           h[q][j] = s*a[j]    + c*h[q][j];
           h[j][p] = c*h[j][p] - s*h[j][q];
           h[j][q] = s*d[j]    + c*h[j][q];
          }
     v[j][p] = c*v[j][p] - s*v[j][q];
     v[j][q] = s*e[j]    + c*v[j][q];
    }
   h[p][p] = (c*c)*h[p][p] + (s*s)*h[q][q] - 2.0*c*s*h[p][q];
   h[q][q] = (s*s)*f + (c*c)*h[q][q] + 2.0*c*s*h[p][q];
   h[p][q] = 0.0;
   h[q][p] = 0.0;
  }
 }
}
```

```
    if(U <= Uf)
        converge = 1;
    else
        {
         it++;
         *iter = it;
         U = U/n;
        }
  }while(!((converge) || it > 200));
  free(a);
  free(d);
  free(e);
}

/* ******************************/
/*   Projection image [X]*[Ai]   */
/* ******************************/
void Projection()
{
  unsigned int i,j,k,addi;
  long psize;
  long l,size1,size2,size3,count;
  float sum;
  FILE **fp1,**fp2;
  char *mes;
  mes = "function projection";
  fp1 = (FILE **) malloc(sizeof(FILE *)*tb);
  fp2 = (FILE **) malloc(sizeof(FILE *)*tb);
  if(fp1 == NULL && fp2 == NULL)
    MemErr(mes);
  for(i=0L;i<tb;i++)
    {
    fp1[i] = (FILE *) malloc(sizeof(FILE)*tb);
    fp2[i] = (FILE *) malloc(sizeof(FILE)*tb);
    if(fp1[i] == NULL && fp2[i] == NULL)
```

```
    MemErr(mes);
  }
psize = 1024L;
size1 = ((long)xsize)*((long)ysize);
for(k=0;k<tb;k++)
  {
  addi = k+tb;
  App(drive,fn,addi);
  fp2[k] = fopen(fn,"wb");
  if(fp2[k] == NULL)
     FopenErr(mes);
  for(i=0;i<tb;i++)
    lbuff[i] = 0L;
  printf("\nprojection band %d...",k+1);
  count = 0L;
  size2 = size1;
  size3 = size1;
  psize = 1024L;
  do
    {
     j = 0;
     do
          {
            App(drive,fn,j);
            fp1[j] = fopen(fn,"rb");
            if(fp1[j] == NULL)
              FopenErr(mes);
            rewind(fp1[j]);
            fseek(fp1[j],lbuff[j],SEEK_SET);
            fread(ibuff1[j],sizeof(int),(int)psize,fp1[j]);
            lbuff[j] += (float)(psize*2L);
            fclose(fp1[j]);
            j++;
          }while(j < tb);
```

```
      for(i=0;i<(int)psize && size2 > 0L;i++,size2--,count++)
           {
            sum = 0.0;
            for(j=0;j<tb;j++)
              sum += (float)(ibuff1[j][i]*v[j][k]);
            ibuff[i] = (int)sum;
           }
      fwrite(ibuff,sizeof(int),(int)psize,fp2[k]);
      size3 -= (long)psize;
      if(size3 < (long)psize)
           psize = (int)size3;
     }while(size2 > 0L);
    fclose(fp2[k]);
   }
}

/* ***************************/
/*   Scaling image 256 level      */
/* ***************************/
void Scaling()
{
  unsigned int i,addi,k;
  long psize;
  long l,count,size1,size2,size3;
  float temp,max,min;
  int ch[1];
  FILE **fp1,**fp2;
  char *mes;
  mes = "function scaling";
  fp1 = (FILE **) malloc(sizeof(FILE *)*tb);
  fp2 = (FILE **) malloc(sizeof(FILE *)*tb);
  if(fp1 == NULL && fp2 == NULL)
    MemErr(mes);
  for(i=0;i<tb;i++)
    {
```

```
  fp1[i] = (FILE *) malloc(sizeof(FILE)*tb);
  fp2[i] = (FILE *) malloc(sizeof(FILE)*tb);
  if(fp1[i] == NULL && fp2[i] == NULL)
  MemErr(mes);
  }
psize = PSIZE;
size1 = ((long)xsize)*((long)ysize);
for(k=0;k<tb;k++)
  {
   printf("\nscaling gray scale band %d...",k+1);
   printf("search max. & min...");
   addi = k+tb;
   App(drive,fn,addi);
   fp2[k] = fopen(fn,"rb");
   if(fp2[k] == NULL)
     FopenErr(mes);
   size2 = size1;
   size3 = size1;
   psize = 32768;
   fread(ch,sizeof(int),1,fp2[k]);
   max = (float)ch[0];
   min = (float)ch[0];
   rewind(fp2[k]);
   count = 0L;
   do
     {
        fread(ibuff,sizeof(int),psize,fp2[k]);
        for(i=0;i<psize && size2 > 0L;i++,size2--,count++)
          {
           temp = (float)ibuff[i];
           if(temp > max)  max = temp;
           if(temp < min)  min = temp;
          }
      size3 -= (long)psize;
      if(size3 < (long)psize)
```

```
            psize = (int)size3;
        }while(size2 > 0L);
    rewind(fp2[k]);
    App(drive,fn,k);
    remove(fn);
    fp1[k] = fopen(fn,"wb");
    if(fp1[k] == NULL)
        FopenErr(mes);
    printf("scaling...");
    size2 = size1;
    size3 = size1;
    psize = 32768;
    count = 0L;
    do
      {
        fread(ibuff,sizeof(int),psize,fp2[k]);
        for(i=0;i<psize && size2 > 0L;i++,size2--,count++)
          {
            temp = (float)ibuff[i];
            buff[i] = (int)(255.0*(temp-min)/(max-min));
          }
        fwrite(buff,sizeof(char),psize,fp1[k]);
        size3 -= (long)psize;
        if(size3 < (long)psize)
          psize = (int)size3;
      }while(size2 > 0L);
    fclose(fp1[k]);
    fclose(fp2[k]);
  }
}

void App(a,b,c)
char *a,*b;
int c;
{
```

```
  int i,j;
  char p1[5],p2[5];
  strcpy(p1,a);
  i = 0;
  do
    {
     i++;
    }while(p1[i] != NULL);
  if(p1[i] == NULL)
     itoa(c+1,p2,10);
  for(j=0;p2[j] != NULL;j++)
    p1[i+j] = p2[j];
  p1[i+j] = NULL;
  strcpy(b,p1);
}

void MemErr(char *m)
{
 clrscr();
 printf("memory allocation error in %s",m);
 getch();
 exit(1);
}

void FopenErr(char *m)
{
 clrscr();
 printf("file open error in %s",m);
 getch();
 exit(1);
}
```

ประวัติผู้เขียน

นายสักรียา ชิตวงค์ เกิดวันที่ 9 สิงหาคม พ.ศ 2509 ที่อำเภอควนกาหลง จังหวัดสตูล จบการศึกษาระดับปริญญาตรีในหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต สาขาวิชาเทคโนโลยีคอมพิวเตอร์อุตสาหกรรม จากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2534 เคยทำงานในตำแหน่งผู้ช่วยนักวิจัยประจำภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปี พ.ศ 2534 ถึง 2536