การกำจัดเมฆและเงาเมฆโดยพร้อมๆกันในภาพถ่ายดาวเทียมสองภาพด้วยฮีสโตแกรมสามมิติ

Cloud and shadow removing from two satellite images simultaneously using 3-dimension histogram

ทนง ธนพันธุ์พาณิชย์
Tanong Tanaphanpanich

อาจารย์ที่ปรึกษา
รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์
Advisor
Assoc.Prof.Dr. Fusak Cheevasuvit

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2536

ISBN 974-621-109-9

CLOUD AND SHADOW REMOVING FROM TWO SATELLITE IMAGES SIMULTANEOUSLY USING 3-DIMENSION HISTOGRAM

TANONG TANAPHANPANICH

ADVISOR

ASSOC.PROF.DR. FUSAK CHEEVASUVIT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

GRADUATE SCHOOL

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

1993

ISBN 974-621-109-9

บทคัดย่อ

หัวข้อวิทยานิพนธ์	**การกำจัดเมฆและเงาเมฆโดยพร้อม ๆกันในในภาพถ่ายดาวเทียมสองภาพด้วยฮีสโตแกรมสามมิติ**
ชื่อนักศึกษา	**นาย ทนง ธนพันธุ์พาณิชย์**
อาจารย์ที่ปรึกษา	**รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์**
ระดับการศึกษา	**วิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า**
ปีการศึกษา	2536

การกำจัดกลุ่มจุดเมฆที่ปกคลุมบนภาพถ่ายดาวเทียมเป็นที่นิยมใช้กันมากสำหรับงานการแปลความหมายของภาพที่สนใจ ดังนั้นจึงได้เสนอเทคนิคในการกำจัดจุดภาพที่เป็นเมฆและเงาเมฆบนภาพถ่ายดาวเทียมโดยใช้ฮีสโตแกรมสามมิติ ฮีสโตแกรมนี้สร้างได้จากการใช้ฮีสโตแกรมของภาพถ่ายดาวเทียมจากแถบความยาวคลื่นที่มองเห็นด้วยตาจำนวนสองภาพที่มีการกระจายของเมฆและเงาเมฆต่างกันออกไป ฮีสโตแกรมของภาพทั้งสองจะถูกนำมาวางอยู่บนแกนทั้งสองของฮีสโตแกรมสามมิติ ส่วนแกนที่สามนั้นจะใช้ฮีสโตแกรมของภาพผลต่างที่เกิดจากภาพทั้งสอง จากการใช้กลุ่มเธร์สโฮลด์กับฮีสโตแกรมสามมิตินี้จะสามารถตรวจจับบริเวณเมฆและเงาเมฆจากภาพทั้งสองได้ การกำจัดเมฆและเงาเมฆของภาพทั้งสอง ทำได้โดยการแทนจุดเหล่านั้นด้วยจุดภาพในตำแหน่งที่สอดคล้องของอีกภาพหนึ่ง เทคนิคการกำจัดเมฆและเงาเมฆแบบที่เสนอนี้มีข้อดีเหนือกว่าเทคนิคอื่น ตรงที่สามารถกำจัดได้ทั้งเมฆและเงาเมฆในภาพทั้งสองได้พร้อมๆกัน

ABSTRACT

THESIS TITLE	**Cloud and shadow removing from two satellite images simultaneously using 3-dimension histogram**
STUDENT	**Tanong Tanaphanpanich**
THESIS ADVISOR	**Assoc. Prof. Dr. Fusak Cheevasuvit**
LEVEL OF STUDY	**Master of Engineering in Electrical Engineering**
ACADEMIC YEAR	1993

A method of cloud cover and cloud shadow removal by using 3-dimensional histogram is presented. Each histogram of two visible band images has been mapped onto the first two axises of the histogram, while the third is performed by the difference image histogram for detecting the cloud and shadow of both images. Therefore, cloud pixels and shadow pixels in both images can be removed and substituted by the correspondent pixels from the other images. The major improvement over the other cloud detection techniques is the cloud and shadow removal in two images can be done on two images simultaneously.

กิตติกรรมประกาศ

ขอขอบพระคุณ รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์ เป็นอย่างสูง ที่ได้ให้การประสิทธิ์ประสาทวิชา คำแนะนำแนวทางในเรื่องต่างๆจนทำให้สำเร็จลุล่วงไปได้ดี และขอขอบคุณอาโมทย์ สมบูรณ์แก้ว ที่คอยช่วยเหลือและเป็นกำลังใจ รวมทั้งเพื่อนๆพี่ๆที่ห้องวิจัยไว้เป็นอย่างสูง

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ที่มาของงานวิจัย

เนื่องจากประเทศไทยมักจะมีเมฆปกคลุมอยู่ตลอดทั้งปี ดังนั้นการนำเอาภาพถ่ายดาวเทียมมาใช้เป็นข้อมูลในการแปลความหมายหรือตรวจสอบพื้นที่ด้วยตาทำได้อย่างไม่มีประสิทธิภาพ ทั้งนี้เพราะกลุ่มเมฆและเงาเมฆที่ปกคลุมเป็นตัวอุปสรรค ดังนั้นจึงจำเป็นต้องหาวิธีการที่สะดวกและคล่องตัวในการตรวจสอบและกำจัดสิ่งเหล่านี้ไป เพื่อช่วยให้การพิจารณาหรือตรวจสอบตามบริเวณพื้นที่ที่สนใจทำได้อย่างถูกต้องและมีประสิทธิภาพ จึงทำให้เกิดแนวคิดในการตรวจสอบและกำจัดกลุ่มจุดเมฆและเงาเมฆที่ปรากฏบนภาพถ่ายดาวเทียม โดยการใช้ฮีสโตแกรมสามมิติกระทำการประมวลผลภาพในครั้งเดียวพร้อมๆกันทั้งสองภาพ

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 เพื่อต้องการตรวจสอบข้อมูลภาพถ่ายดาวเทียมที่เป็นทั้งกลุ่มเมฆและเงาเมฆ

1.2.2 เพื่อต้องการกำจัดจุดภาพที่เป็นเมฆและเงาเมฆบนภาพถ่ายดาวเทียม ตลอดทั้งสัญญาณรบกวนชนิด salt และ pepper ได้ภายในครั้งเดียวโดยพร้อมๆทั้งสองภาพ

1.3 ขอบเขตของวิทยานิพนธ์

วิทยานิพนธ์ฉบับนี้เสนอวิธีการประมวลผลข้อมูลภาพถ่ายในระยะไกลโดยกล่าวถึงเทคนิคและวิธีการกำจัดจุดภาพที่เป็นเมฆและเงาเมฆซึ่งปรากฏติดมากับข้อมูลภาพถ่ายดาวเทียมตามพื้นที่ที่ต้องการตรวจสอบ เพื่อจะช่วยให้สามารถพิจารณาข้อมูลภาพเหล่านั้นได้อย่างมีประสิทธิภาพโดยปราศจากสัญญาณรบกวนใดๆ รายละเอียดโดยย่อของวิทยานิพนธ์ฉบับนี้พอจะกล่าวได้ดังนี้

บทที่ 1 เป็นบทนำ กล่าวถึงวัตถุประสงค์ และ ขอบเขตของวิทยานิพนธ์

บทที่ 2 จะกล่าวถึงวิธีการและขั้นตอนการเตรียมข้อมูลภาพถ่ายดาวเทียมสำหรับการประมวลผลที่จำเป็นสำหรับขบวนการตรวจสอบและกำจัดจุดภาพที่เป็นเมฆหรือเงาเมฆตลอดทั้งสัญญาณรบกวนที่ติดปะปนติดมากับข้อมูลภาพที่ต้องการพิจารณา ขั้นตอนและวิธีการเตรียมข้อมูลภาพที่เกี่ยวข้องได้แก่ การทำสหสัมพันธ์ระหว่างข้อมูลภาพจำนวนสองภาพ เพื่อที่จะหาตำแหน่งพิกัดที่ถูกต้องของภาพถ่ายดาวเทียมในส่วนที่ทับซ้อนกัน ซึ่งจากขั้นตอนนี้ก็จะได้ข้อมูล

ทั้งสองที่มีตำแหน่งพิกัดพื้นที่ที่ตรงกันตามความต้องการ วิธีการปรับค่าความสว่างของภาพทั้งสองให้ที่สอดคล้องกันหรือใกล้เคียงกันทั้งสองภาพ รวมทั้งวิธีและหลักการแสดงข้อมูลในลักษณะแบบสามมิติ

บทที่ 3 จะกล่าวถึงวิธีการกำจัดเมฆแบบใช้เธร็สโฮลด์คู่ (Dual thresholds) เป็นวิธีการกำจัดเมฆโดยอาศัยข้อมูลของฮีสโตแกรมมิติเดียวเป็นหลักในการพิจารณากำหนดเธร็สโฮลด์จำนวนสองค่าสำหรับเป็นขอบเขตในการตรวจสอบและกำจัดจุดภาพที่เป็นเมฆออกจากภาพถ่ายดาวเทียมที่กำลังพิจารณา

บทที่ 4 จะกล่าวถึงเทคนิคและวิธีการกำจัดเมฆและเงาเมฆโดยใช้ฮีสโตแกรมสองมิติ ซึ่งจะอาศัยข้อมูลภาพจำนวนสองภาพสำหรับสร้างฮีสโตแกรมสองมิติ เมื่อพิจารณาข้อมูลที่ปรากฏบนฮีสโตแกรมสองมิติทำให้เราสามารถกำหนดขอบเขตของกลุ่มเธร็สโฮลด์จำนวนสองกลุ่มเพื่อการตรวจสอบและกำจัดเมฆของภาพที่กำลังสนใจโดยพร้อมๆกันได้

บทที่ 5 จะกล่าวถึงหลักและวิธีในการกำจัดเมฆและเงาเมฆพร้อมๆกันในภาพถ่ายดาวเทียมสองภาพด้วยฮีสโตแกรมสามมิติ เทคนิคในบทนี้เป็นการพัฒนาและปรับปรุงวิธีการตรวจสอบและการกำจัดเมฆหรือเงาเมฆ เพื่อให้มีความสะดวกและคล่องตัว โดยอาศัยวิธีการพล๊อทฮีสโตแกรมในลักษณะสามมิติซึ่งสามารถที่จะเปลี่ยนมุมมองของข้อมูลภาพที่ปรากฏบนฮีสโตแกรมสามมิติได้ง่ายและสะดวก ทำให้สามารถกำหนดขอบเขตของกลุ่มเธร็สโฮลด์สำหรับการตรวจสอบและกำจัดเมฆตลอดทั้งสัญญาณรบกวนประเภท Salt หรือ Pepper ได้พร้อมๆกันทั้งสองภาพ

บทที่ 6 ก็จะเป็นบทสรุปและข้อเสนอแนะในการวิจัยพัฒนาเทคนิคให้มีประสิทธิภาพยิ่งขึ้น

สำหรับในส่วนสุดท้ายจะเป็นภาคผนวก เป็นส่วนของรายละเอียดโปรแกรมการประมวลผลที่ใช้ในวิทยานิพนธ์นี้

บทที่ 2

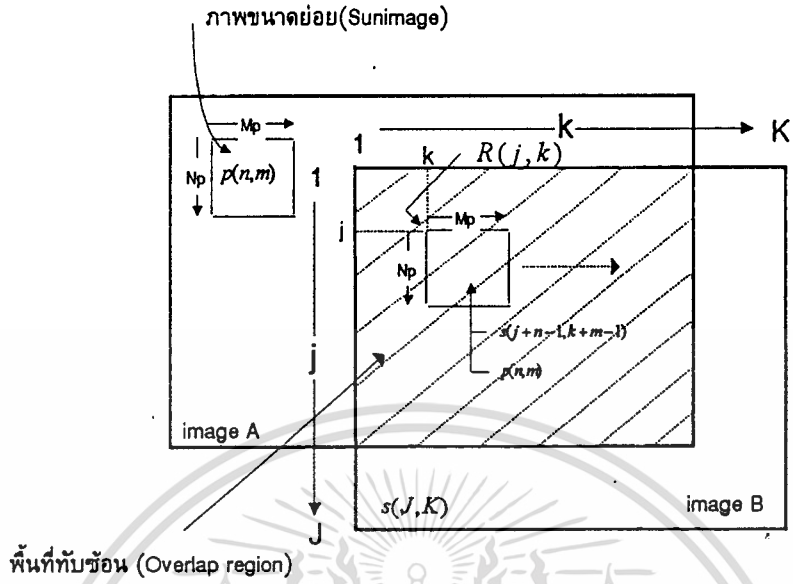
การเตรียมข้อมูลภาพสำหรับการประมวลผล

2.1 บทนำ

การบันทึกข้อมูลภาพถ่ายดาวเทียมในแต่ละวงโคจรนั้น ย่อมมีสภาพภูมิอากาศที่ผันแปรเปลี่ยนไปในแต่ละช่วงเวลา ดังนั้นทำให้ได้ข้อมูลภาพในแต่ละครั้ง มีคุณลักษณะของภาพที่ได้แตกต่างกันไป อาทิเช่น ระดับความเข้มของจุดภาพแตกต่างกันอันอาจจะเกิดจากผลกระทบจากชั้นบรรยากาศตามสภาวะต่างๆ หรืออาจจะเป็นจากตัวกล้องบันทึกภาพเอง ความผิดปกติทางรูปเรขาคณิตที่ไม่สมมาตรกัน ตลอดทั้งสัญญาณรบกวนชนิดจุดขาวและจุดดำ (Salt and Pepper noise) กระจัดกระจายรบกวนปรากฏให้เห็นในภาพถ่ายดาวเทียม เป็นต้น ดังนั้นในการที่นำเอาภาพถ่ายดาวเทียมจำนวนสองภาพที่ได้บันทึกไว้คนละช่วงเวลามาทำการกำจัดเมฆและเงาเมฆตามวิธีที่จะกล่าวในวิทยานิพนธ์นี้ จำเป็นจะต้องนำข้อมูลทั้งสองมาทำการแก้ไขปรับปรุงให้มีความสอดคล้องของจุดภาพเหมือนกันทั้งสองภาพ วิธีในการเตรียมข้อมูลภาพสำหรับการประมวลผลอันได้แก่ การทำสหสัมพันธ์เพื่อหาพิกัดทับซ้อนของภาพทั้งสอง (Correlation) การปรับค่าความสว่างของจุดภาพทั้งสองให้มีความสอดคล้องกัน (Brightness Adjustment) ตลอดทั้งวิธีการแสดงข้อมูลภาพสามมิติ ดังจะได้กล่าวต่อไป

2.2 การทำสหสัมพันธ์ของข้อมูลภาพทั้งสองที่มีจุดตำแหน่งเบียงเบนไม่ตรงกัน

การหาสหสัมพันธ์ (Correlation) เป็นวิธีการหาความสัมพันธ์ของข้อมูลภาพใดๆจำนวนสองภาพซึ่งจะสามารถทราบได้ว่าข้อมูลหรือวัตถุของทั้งสองชิ้นนั้นเป็นชนิดเดียวกันหรือเป็นพื้นที่เดียวกันหรือไม่ สำหรับวิทยานิพนธ์นี้ จะเป็นการหาตำแหน่งพิกัดของข้อมูลภาพถ่ายดาวเทียมในส่วนพื้นที่ที่ทับซ้อนกันจากข้อมูลที่ถูกบันทึกไว้คนละช่วงเวลา เพื่อให้ได้พื้นที่ที่มีจุดพิกัดตรงกัน การทำสหสัมพันธ์ของข้อมูลภาพที่จะกล่าวนี้เป็นวิธีการคำนวนแบบวัดค่าเปรียบเทียบ (Comparison measures) ระหว่างวัตถุทั้งสอง โดยการนำข้อมูลภาพมาคำนวนซ้อนทับกันระหว่างจุดภาพต่อจุดภาพตามตำแหน่งที่สอดคล้องกัน แสดงดังรูปที่ 2.1 และผลที่สุดก็จะทราบตำแหน่งพิกัดเบี่ยงเบนในส่วนที่ทับซ้อนที่ตรงกัน เพื่อจะตัดเลือกเอาเฉพาะพื้นที่ที่ตรงกันหรือทับซ้อนกันออกมาประมวลผลต่อไป

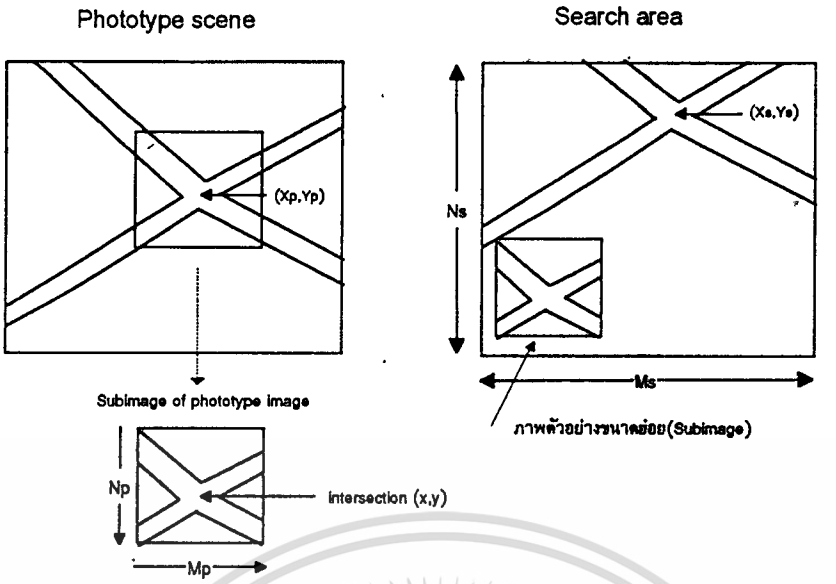


รูปที่ 2.1 หลักในการทำสหสัมพันธ์ระหว่างข้อมูลภาพจำนวนสองภาพ โดยการนำภาพทั้งสองมาหาความสัมพันธ์เพื่อหาพิกัดตำแหน่งที่ทับซ้อนกันของภาพ A กับภาพ B

เนื่องจากในการหาสหสัมพันธ์แบบจุดต่อจุดของภาพทั้งสองต้องใช้เวลาในการคำนวนมาก ดังนั้นเราจะทำการตัดเลือกเอาเฉพาะบริเวณพื้นที่ที่เด่นชัด อาทิเช่นบริเวณทางแยกต่างๆ หรือบริเวณเส้นทางไหลผ่านของแม่น้ำตัดเอามาเป็นภาพตัวอย่างขาดย่อยซึ่งเรียกว่า Subimage หรือ $p(n,m)$ มาทำการหาสหสัมพันธ์ในส่วนที่ทับซ้อนกันกับภาพที่ต้องการค้นหา (search area) หรือ $s(j,k)$ เพื่อหาตำแหน่งพิกัดพื้นที่ที่ตรงกันของภาพทั้งสอง ยังผลทำให้ลดระยะเวลาในการคำนวณได้มากกว่าการเอาภาพใหญ่ทั้งสองภาพมาหาสหสัมพันธ์ แสดงดังรูปที่ 2.2

วิธีการทำ สหสัมพันธ์ที่นำมาใช้กับวิทยานิพนธ์นี้ เป็นวิธีที่การเปรียบเทียบวัดค่าความแตกต่างระหว่างจุดภาพต่อจุดภาพของทั้งสอง ซึ่งเรียกว่า Mean normalized correlation ดังสมการที่ 2.1

$$R(j,k) = \frac{\sum_{n=1}^{Np}\sum_{m=1}^{Mp}(p(n,m)-\bar{p})(s(j+n-1,k+m-1)-\bar{s}_{j,k})}{\left(\sqrt{\sum_{n=1}^{Np}\sum_{m=1}^{Mp}(p(n,m)-\bar{p})^2}\sqrt{\sum_{n=1}^{Np}\sum_{m=1}^{Mp}(s(j+n-1,k+m-1)-\bar{s}_{j,k})^2}\right)} \quad (2.1)$$



รูปที่ 2.2 แสดงการทำสหสัมพันธ์โดยการตัดเลือกเอาบริเวณพื้นที่ที่เป็นทางแยกออกมาเป็นภาพย่อย เพื่อที่จะหาสหสัมพันธ์กับอีกภาพที่ทับซ้อนกัน

โดยที่ค่าเฉลี่ยของข้อมูลตัวอย่างภาพย่อย (subimage) ที่ใช้อ้างอิงในการค้นหา เป็นไปตามสมการ (2.2) ดังนี้

$$\bar{p} = \frac{1}{N_p M_p} \sum_{n=1}^{N_p} \sum_{m=1}^{M_p} p(n,m) \qquad (2.2)$$

และค่าเฉลี่ยของข้อมูลภาพที่ต้องการค้นหาตำแหน่งพิกัดที่ตรงกันเป็นไปตามสมการ (2.3) คือ

$$\bar{s}_{j,k} = \frac{1}{N_p M_p} \sum_{n=1}^{N_p} \sum_{m=1}^{M_p} s(j+n-1, k+m-1) \qquad (2.3)$$

เมื่อ $R(j,k)$ คือ ค่าสัมประสิทธ์ของการทำสหสัมพันธ์ (Correlation Coefficient) ที่ตำแหน่งพิกัดตามข้อมูลภาพที่ทำการค้นหา (Search area)

$p(n,m)$ คือค่าความเข้มของจุดภาพในภาพตัวอย่างขนาดย่อยที่พิกัด (m,n)

$s(n,m)$ คือค่าความเข้มของจุดภาพในภาพที่ต้องการค้นหาที่พิกัด (m,n)

ค่าสัมประสิทธิ์ของสหสัมพันธ์ $R(j,k)$ จากสมการที่ (2.1) จะมีค่าเปลี่ยนแปลงในช่วง [-1, 1] และเมื่อ $R(j,k)$ = 1 ก็จะเป็นค่าที่ดีที่สุดในการทับซ้อน (Matching) ขบวนการนี้ต้อง

คำนวณหาค่า $R(j,k)$. ในแต่ละตำแหน่ง (j,k) เพื่อเลือกค่าที่มากที่สุดซึ่งจะได้พิกัดของภาพในตำแหน่งทับซ้อนกันพอดี

2.3 การปรับค่าความสว่างของจุดภาพทั้งสองให้เท่ากัน (Brightness Adjustment)

ภาพถ่ายดาวเทียมที่ได้ถูกบันทึกไว้คนละช่วงเวลาของวงโครจรในแต่ละรอบของดาวเทียม ย่อมมีสภาพของบรรยากาศหรือภูมิอากาศที่แปรเปลี่ยนไป ทำให้ค่าความเข้มของข้อมูลภาพที่ได้มาในแต่ละครั้งมีค่าแตกต่างกัน ดังนั้นจึงจำเป็นที่จะต้องทำการปรับค่าความเข้มของข้อมูลจุดภาพทั้งสองให้มีค่าเท่ากันหรือใกล้เคียงกัน โดยการคำนวณหาค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานของจุดภาพ ซึ่งเป็นสมการปรับค่าความสว่างของจุดภาพดังนี้คือ

$$y = \frac{\sigma_i}{\sigma_f} x + m_i - \frac{\sigma_i}{\sigma_f} m_f \qquad (2.4)$$

y คือ ค่าความเข้มของจุดภาพใหม่
x คือ ค่าความเข้มของจุดภาพเก่า
m_i คือ ค่าเฉลี่ยของจุดภาพข้อมูลอ้างอิง
σ_i คือ ค่าเบี่ยงเบนมาตราฐานจุดภาพของข้อมูลอ้างอิง
m_f คือ ค่าเฉลี่ยของจุดภาพที่จะทำการปรับ
σ_f คือ ค่าเบี่ยงเบนมาตราฐานของจุดภาพที่จะทำการปรับ

เมื่อค่าเฉลี่ยของจุดภาพคำนวณได้จากสมการ

$$m = \frac{1}{N} \sum_{i=0}^{N} x_i \qquad (2.5)$$

m คือ ค่าเฉลี่ยของจุดภาพ
x_i คือ ค่าของจุดภาพ
N คือ จำนวนจุดภาพทั้งหมด

ส่วนค่าเบี่ยงเบนมาตราฐานของจุดภาพคำนวณได้ดังสมการ

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - m)^2} \quad (2.6)$$

σ คือ ค่าเบี่ยงเบนมาตราฐานของจุดภาพ

x_i คือ ค่าความเข้มของจุดภาพ

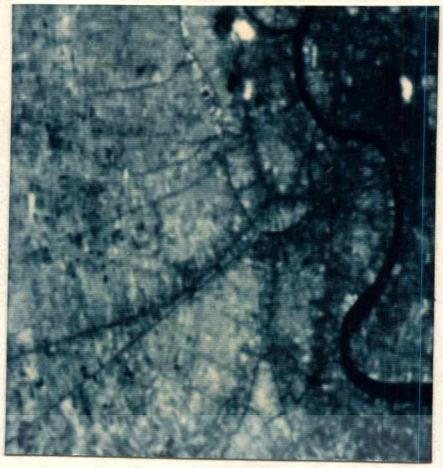
N คือ จำนวนของจุดภาพทั้งหมด

m คือ ค่าเฉลี่ยของจุดภาพ

2.4 ผลของการปรับค่าความสว่าง

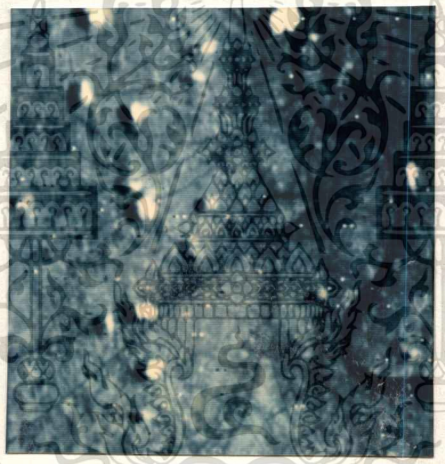
ข้อมูลภาพถ่ายดาวเทียมดังภาพที่ 2.3(a) และภาพที่ 2.3(b) เป็นภาพต้นแบบที่ผ่านการทำสหสัมพันธ์มาแล้ว จะสังเกตุเห็นได้ว่าค่าความเข้มของจุดภาพทั้งสองจะแตกต่างกัน ดังนั้นภาพหลังจากการปรับค่าความสว่างจากภาพ 2.3(a) เข้าหาภาพที่ 2.3(b) จะแสดงได้ดังรูปที่ 2.4 ซึ่งผลลัพธ์จะทำให้ภาพถ่ายดาวเทียมของทั้งสองภาพมีค่าความสว่างที่ใกล้เคียงกัน

(a)



(b)

รูปที่ 2.3 แสดงภาพดาวเทียมที่มีค่าความเข้มแตกต่างกัน



รูปที่ 2.4 แสดงภาพที่ผ่านการปรับค่าความสว่างแล้ว

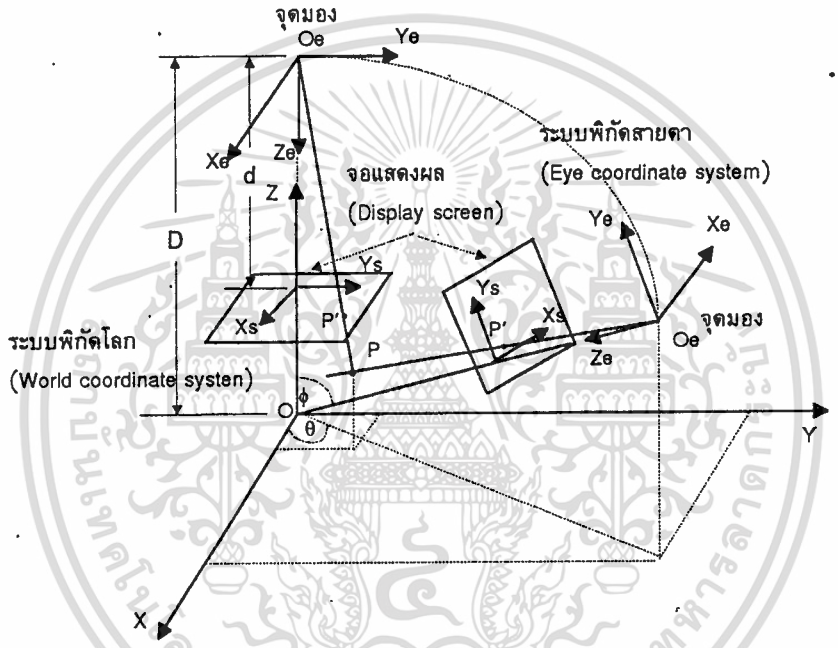
2.5 การมองในระบบ 3 มิติ (Viewing in 3D)

ในการแสดงข้อมูลของฮีสโตแกรมแบบสามมิติ จำเป็นที่จะต้องทำการแปลงจุดต่างๆให้แสดงบนจอมอนิเตอร์แบบสองมิติ ตามทฤษฎีของคอมพิวเตอร์กราฟฟิก กล่าวคือ การแปลงจุดต่างๆของวัตถุในระบบพิกัดโลก (World coordinates) ให้ไปเป็นคู่ลำดับของจอภาพบนเครื่องคอมพิวเตอร์กราฟฟิกนั้น จะต้องกำหนดจุดมองในระบบสามมิติ โดยให้จุดมอง (View point) หรือจุดที่ตามองอยู่บนจุดกำเนิดของระบบพิกัดสายตา (Eye coordinates system) ซึ่งค่าพิกัดสายตามีความสำคัญมากในการแปลงค่าพิกัดการมอง (Viewing tranformation) ให้เป็นค่าพิกัดบนระนาบสองมิติของจอภาพนั้น เราเรียกว่าการ โปรเจค (Projection) การโปรเจคในระบบสามมิติมี 2 แบบคือ การโปรเจคแบบศูนย์กลาง (Central Projection) หรือเรียกอีกอย่างว่า การโปรเจคแบบ

ขนาน (Palallel projection) และอันที่สองก็คือ การโปรเจค ณ. จุดมองใดๆ (projection through an arbitrary view point) สำหรับในวิทยานิพนธ์นี้จะใช้การโปรเจคแบบจุดมองใดๆ ซึ่งสามารถกำหนดจุดมองได้ทุกจุด

การโปรเจค ณ. จุดมองใดๆ (Projection through an arbitrary viewpoint)

พิจารณาจากรูปที่ 2.5 กำหนดให้แกนของระบบค่าพิกัดสายตาคือ (x_e, y_e, z_e) จุด Oe คือจุดมอง โดย แกน (x_s, y_s) เป็นระบบพิกัดของจอภาพ แกน x, y, z เป็นแกนในระบบพิกัดโลก และจุด P คือ จุดในระบบพิกัดโลก



รูปที่ 2.5 การโปรเจคชั่นของจุด P สู่ค่าพิกัดของจอภาพ

ในการโปรเจคแบ่งขั้นตอนหลักออกเป็น 2 ขั้นตอนคือ

ก) ทำการแปลงจุดบนพิกัดโลก (x,y,z) ไปเป็นจุดที่สอดคล้องบนพิกัดสายตา (eye coordinates system) ณ จุด (x_e, y_e, z_e)

ข) ทำการแปลงจากจุดพิกัดสายตา (x_e, y_e, z_e) ไปเป็นจุดที่สอดคล้องบนระนาบพิกัดของจอภาพ (screen coordinate) (x_s, y_s)

ในการพิจารณาการโปรเจคชั่นของวัตถุ เรากำหนดจุดมองในพิกัดแบบทรงโค้ง (spherical coordinates) ก็คือ (D , θ , ϕ)โดยมีขั้นตอนดังนี้

ขั้นที่ 1 ทำการแปลง (translate) จุดกำเนิดในระบบพิกัดโลกไปยังจุด Oe (จุดกำเนิดของคู่ลำดับสายตา) เพื่อสร้างระบบแกนใหม่ที่จุดมอง ดังรูปที่ 2.6(a) ตามสมการการแปลงคือ

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -D\sin\phi\cos\theta & -D\sin\phi\cos\theta & -D\cos\phi & 1 \end{bmatrix} \quad (2.7)$$

ขั้นที่ 2 ทำการแปลงระบบคู่ลำดับใหม่ไปเป็นระบบคู่ลำดับสายตา โดยเริ่มจากการแปลงแบบหมุนทางแกน Z' ตามเข็มนาฬิกา ดังรูปที่ 2.5(b) ด้วยมุม $(90 - \theta)$ ตามสมการที่ (2.8) จะได้ T2 เป็น

$$T_2 = \begin{bmatrix} \sin\theta & \cos\theta & 0 & 0 \\ -\cos\theta & \sin\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.8)$$

ขั้นที่ 3 ทำการแปลงโดยหมุนในแนวแกน X ในทิศทางทวนเข็มนาฬิกาเป็นมุม$(180 - \phi)$ ดังรูปที่ 2.6(c) ตามสมการ (2.9) จะได้ T3 เป็น

$$T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & -\cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.9)$$

ขั้นตอนที่ 4 เป็นขั้นสุดท้าย โดยทำการแปลงไปเป็นระบบพิกัดซ้ายมือ (left handed coordinate system) โดยทำการกลับทิศทางของแกน X ดังรูปที่ 2.6(d) หรือตามสมการการแปลงดังนี้

$$T_4 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.10)$$

จากขั้นตอนการแปลงที่ผ่านมา จะได้ผลลัพธ์ของการแปลงมุมมอง (viewing transformation) คือ

$$T = T_1.T_2.T_3.T_4 = \begin{bmatrix} -\sin\theta & -\cos\theta\cos\phi & -\cos\theta\sin\phi & 0 \\ \cos\theta & -\sin\theta\cos\phi & -\sin\theta\sin\phi & 0 \\ 0 & \sin\phi & -\cos\phi & 0 \\ 0 & 0 & D & 1 \end{bmatrix} \quad (2.11)$$

ดังนั้นถ้าหากทราบค่าของพิกัดโลก x,y,z เราจะสามารถทำการแปลงจุดเหล่านี้ไปสู่ค่าพิกัดสายตา ได้ดังนี้

$$(x_e,y_e,z_e,1) = (x,y,z,1).T$$

หรือ

$$\begin{aligned} x_e &= -x\sin\theta + y\cos\theta \\ y_e &= -x\cos\theta\cos\phi - y\sin\theta\cos\phi + z\sin\phi \\ z_e &= -x\cos\theta\sin\phi - y\sin\theta\sin\phi - z\cos\phi + D \end{aligned} \qquad (2.12)$$

จากนั้นเราก็สามารถคำนวณหาค่าของคู่ลำดับบนระนาบสองมิติของจอภาพ (screen coordinates) x_s, y_s ได้ดังสมการ

$$x_s = d\left(\frac{x_e}{z_e}\right) \qquad y_s = d\left(\frac{y_e}{z_e}\right) \qquad (2.13)$$

โดยที่ d : ระยะห่างระหว่างคู่ลำดับจอภาพกับระบบคู่ลำดับสายตา,

ความสัมพันธ์ระหว่างตัวแปรที่เกี่ยวข้องของการกำหนดจุดมอง

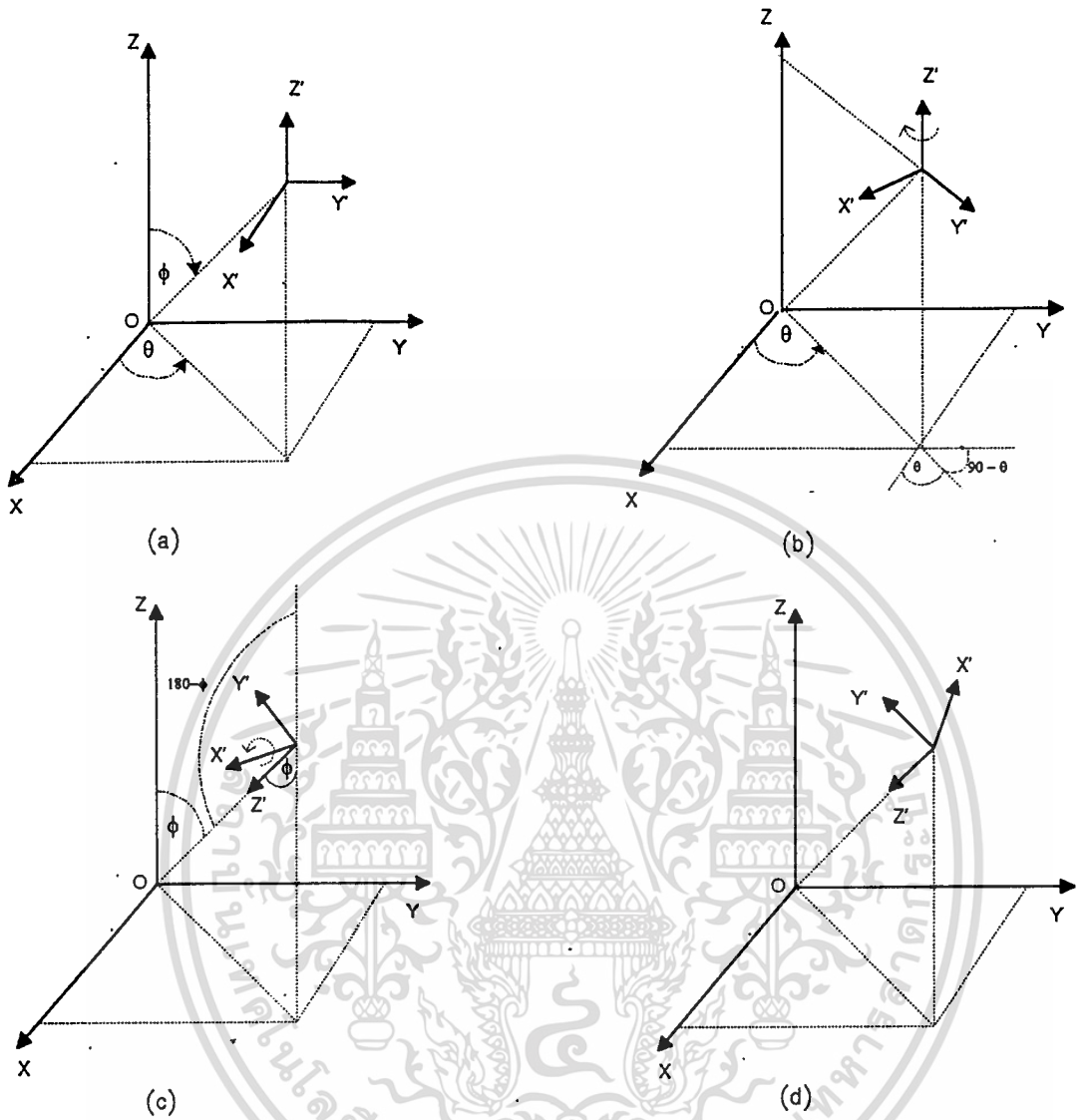
ตัวแปรต่างๆที่เกี่ยวกับการกำหนดจุดมอง (viewing parameter) (d, θ, ϕ and D) จะมีผลต่อภาพที่ปรากฏบนจอภาพดังนี้

ตัวแปร θ , ϕ เป็นค่ามุมที่ใช้ในการกำหนดจุดมองของวัตถุ ณ มุมมองต่างๆ

D คือค่าระยะจากจุดมองถึงวัตถุ โดยลากผ่านจุดกำเนิดของคู่ลำดับโลก การเปลี่ยนแปลงค่าตัวแปร D จะมีผลทำให้จุดมองเข้าใกล้หรือออกห่างจากวัตถุ

ถ้าหากให้ตัวแปร (D, θ, ϕ) คงค่าไว้ แล้วทำการเปลี่ยนแปลงค่าตัวแปร d จะทำให้มีผลต่อการเปลี่ยนแปลงขนาดของภาพที่ปรากฏบนจอมอนิเตอร์ ดังเช่น ถ้าลดค่า d ลง จะทำให้ขนาดของภาพบนจอเล็กลง

เนื่องจากทั้ง D และ d จะเป็นตัวแปรที่ควบคุมขนาดของภาพที่ปรากฏบนจอภาพ การปรับค่าทั้งสองไปด้วยกันย่อมทำให้เกิดความยุ่งยากไม่สะดวก ดังนั้นเพื่อแก้ไขปัญหาดังกล่าว เราก็จะเพิ่มตัวแปรเข้าควบคุมแทนดังหัวข้อถัดไป



รูปที่ 2.6 ลำดับขั้นตอนในการแปลงจากพิกัดโลกไปสู่พิกัดสายตา

ระบบพิกัดจอภาพเอกพันธ์ (Homogeneous screen coordinate system)

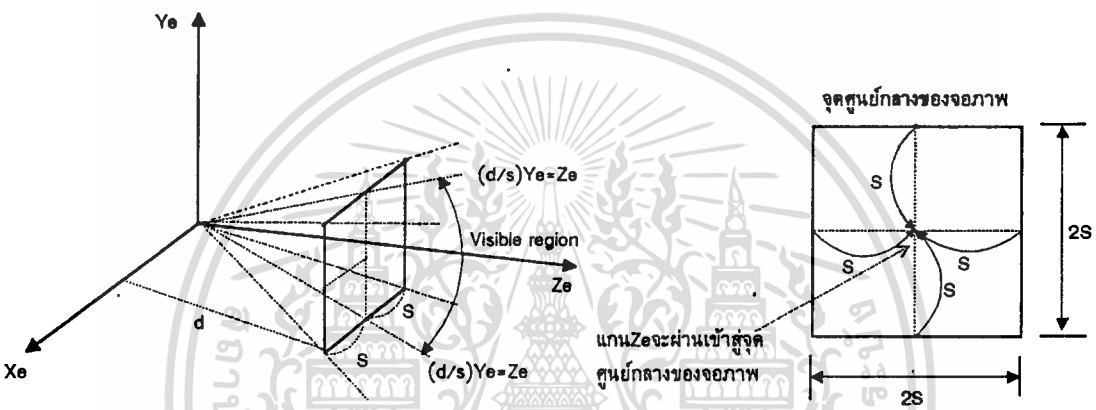
เพื่อลดความยุ่งยากในการแปรค่าของ D และ d ที่ควบคุมของของภาพ เรากำหนดให้ตัวแปร S แทนจุดศูนย์กลางขนาดของจอ (center of the screen) โดยให้แกน z_e ผ่านจุดศูนย์กลางของระนาบ S ดังรูปที่ 2.7 โดยที่ตัวแปร S จะทำหน้าที่คล้ายเลนส์ในกล้องถ่ายรูป ซึ่งมีความสัมพันธ์ในการกำหนดคู่ลำดับบนจอภาพตามสมการดังนี้

$$x_s = \left(\frac{d}{S}\right)\frac{x_e}{z_e} \qquad (2.14)$$

$$y_s = \left(\frac{d}{S}\right)\frac{y_e}{z_e} \qquad (2.15)$$

ดังนั้นเมื่อกำหนดให้จุดมองคงที่ ค่าอัตราส่วน $\frac{d}{s}$ จะเป็นตัวแปรที่ควบคุมขนาดของภาพที่ปรากฏบนจอ และที่สำคัญก็คือผลจากการหารของค่าพิกัดจอภาพโดยตัวแปร S จะทำให้ค่า x_s และ y_s จะมีค่าอยู่ระหว่าง -1 ถึง 1 คือ

$$\begin{aligned} -1 \le x_s \le 1 \\ -1 \le y_s \le 1 \end{aligned} \qquad (2.16)$$



รูปที่ 2.7

จากสมการที่ได้กล่าวมาทำให้สามารถคำนวณหาค่าพิกัดของวัตถุให้ปรากฏบนระนาบคู่ลำดับสองมิติของจอภาพที่แท้จริงได้ดังสมการ

$$\begin{aligned} x_s &= SCF\left[\left(\frac{d}{S}\right)\left(\frac{x_e}{z_e}\right)V_x + L\right] \\ y_s &= -\left(\frac{d}{S}\right)\left(\frac{y_e}{z_e}\right)V_y + M \end{aligned} \qquad (2.17)$$

โดยที่

SCF (Screen Scaling adjustment facter) คือค่าการปรับขนาดของจอมอนิเตอร์ที่ใช้แสดงผล ซึ่งเป็นไปตามอัตราส่วนคือ

$$SCF = \frac{V}{H}$$

เมื่อ

$$H = \frac{\text{ค่าความยาวทางแกนนอน (Horizontal length)}}{\text{จำนวนจุดภาพสูงสุดในแนวแกนนอน}}$$

$$V = \frac{\text{ค่าความยาวทางแกนตั้ง (Vertical length)}}{\text{จำนวนจุดภาพสูงสุดในแนวแกนตั้ง}}$$

d : ระยะห่างระหว่างคู่ลำดับจอภาพกับระบบคู่ลำดับสายตา

D : ระยะห่างระหว่างคู่ลำดับโลกกับระบบคู่ลำดับสายตา

S : ระยะห่างจากจุดกึ่งกลางของจอภาพกับขอบด้านใดด้านหนึ่งของจอภาพ

L : ตำแหน่งของจุดกึ่งกลางของจอภาพในแนวนอน

M : ตำแหน่งของจุดกึ่งกลางของจอภาพในแนวตั้ง

V_x : factor ในการคูณทางแนวนอน

V_y : factor ในการคูณทางแนวตั้ง

2.6 สรุป

ในการเตรียมข้อมูลภาพถ่ายดาวเทียมสำหรับการกำจัดเมฆและเงาเมฆที่จะกล่าวในบทต่อไปนี้ จำเป็นที่จะต้องทำให้ข้อมูลภาพทั้งสองมีความสอดคล้องกันมากที่สุด ทั้งนี้เนื่องจากวิธีการประมวลผลจะเป็นการกระทำคำนวนผลแบบจุดภาพต่อจุดภาพโดยตรง ไม่เช่นนั้นจะทำให้มีความคลาดเคลื่อนได้ วิธีการที่สำคัญได้แก่ การทำสหสัมพันธ์ของข้อมูลภาพทั้งสองเพื่อหาตำแหน่งพิกัดพื้นที่ในส่วนที่ทับซ้อนกัน ซึ่งก็จะได้ข้อมูลอยู่ในตำแหน่งที่ตรงกันทั้งสองภาพตามต้องการ วิธีการปรับค่าความสว่างของจุดภาพทั้งสองให้มีความใกล้เคียงกัน ทำให้สามารถทำการเปรียบเทียบจุดภาพได้อย่างถูกต้องเมื่อทำการกำจัดเมฆหรือเงาเมฆที่จะกล่าวต่อไป สุดท้ายของบทนี้ก็จะเกี่ยวกับหลักและวิธีที่ใช้ในการแสดงข้อมูลในรูปแบบสามมิติ โดยที่สามารถกำหนดมุมมองตามจุดต่างๆได้ ซึ่งก็จะนำไปใช้ร่วมกับการแสดงฮีสโตแกรมสามมิติในบทสุดท้ายต่อไป

บทที่ 3
การใช้เธร็สโฮลด์คู่ในการกำจัดเมฆ
(The use of dual thresholds for cloud detection)

3.1 บทนำ

การกำจัดกลุ่มจุดเมฆที่ปกคลุมบนภาพถ่ายดาวเทียมเป็นเทคนิคที่มีประโยชน์และนิยมใช้กันมาก สำหรับการแปลความหมายของภาพที่สนใจ โดยเฉพาะอย่างยิ่งในประเทศไทย มักจะมีสภาพอากาศที่มีเมฆปรากฏให้เห็นอยู่บ่อยๆ ซึ่งในการแปลความหมายของภาพที่มีจุดเมฆปกคลุมอยู่นั้น ไม่สามารถกระทำได้อย่างมีประสิทธิภาพ จึงทำให้มีผู้วิจัยหลายท่านได้ทำการศึกษาค้นคว้าเทคนิคในการตรวจจับ (Detection) และการจำแนก (Classification) บรรดากลุ่มเมฆต่างๆ ซึ่งโดยส่วนมากแล้ว แนวความคิดมักจะเกี่ยวกับข้อมูลภาพถ่ายดาวเทียมหนึ่งช่องสัญญาณและใช้เธร็สโฮลด์เพียงค่าเดียวในการตรวจจับจุดเมฆที่ปรากฏบนภาพถ่ายดาวเทียม [3,4,5] แต่ธรรมชาติของการแปรเปลี่ยนของเมฆ จะให้เกิดความยากต่อการที่จะหาวิธีการที่เที่ยงตรงและกว้างพอที่จะครอบคลุมปัญหานั้นได้ทั้งหมดต่อการตรวจจับ เพราะลักษณะเด่นของจุดภาพที่เป็นเมฆนั้นจะเปลี่ยนแปลงไปตามสภาพภูมิอากาศ ตรงบริเวณในส่วนขอบของกลุ่มเมฆ มักจะขาดความต่อเนื่องของระดับสีเทา ซึ่งจุดนี้จะยังผลทำให้เกิดปัญหายากต่อการตรวจจับโดยการกำหนดเธร็ชโฮลด์เพียงค่าเดียว ดังนั้นจึงได้ทำการพัฒนาวิธีการใหม่ โดยการกำหนดเธร็สโฮลด์จำนวนสองค่าขึ้นมา (Dual Thresholds) วิธีการนี้สามารถที่จะแก้ปัญหาที่เกิดขึ้นได้ โดยการใช้ภาพถ่ายดาวเทียมจำนวนสองภาพที่ถูกบันทึกไว้คนละช่วงเวลา และกลุ่มจุดภาพที่เป็นเมฆที่ปรากฏบนภาพถ่ายดาวเทียมจะต้องมีการกระจายอยู่ในตำแหน่งที่แตกต่างกัน หลักการก็คือ กำหนดให้เธร็สโฮลด์ค่าแรก (Ta) เป็นค่าเธร็สโฮลด์ร่วม (Common Threshold) ซึ่งกำหนดได้โดยการพิจารณาตรวจสอบโดยตรงจากฮีสโตแกรมของภาพที่ต้องการกำจัดเมฆ ในขณะที่เธร็สโฮลด์ค่าที่สอง (Tr) ซึ่งเรียกว่าเธร็สโฮลด์สัมพันธ์ (Relation Threshold) กำหนดได้โดยการพิจารณาจากฮีสโตแกรมของค่าผลต่างสัมบูรณ์ระหว่างภาพทั้งสอง จากการพิจารณาเปรียบเทียบระหว่างฮีสโตแกรมทั้งสอง ทำให้สามารถทราบการกระจายของกลุ่มเมฆที่ปรากฏอยู่ได้ และก็สามารถกำจัดออกไปจากภาพแรกได้อย่างง่ายดาย โดยการนำเอาจุดภาพที่ปราศจากเมฆของภาพที่สอง ณ. ตำแหน่งที่สอดคล้องกันมาแทนที่ลงไป

3.2 หลักวิธีการ

จากข้อมูลภาพถ่ายดาวเทียมจำนวนสองภาพคือ ภาพ A และภาพ B แสดงดังรูปที่ 3.1 เป็นภาพถ่ายที่เป็นพื้นที่บริเวณเดียวกัน โดยได้ถูกบันทึกหรือถ่ายไว้คนละช่วงเวลา ภาพทั้งสองมีการกระจายของกลุ่มเมฆในตำแหน่งที่แตกต่างกัน โดยที่ภาพ A เป็นภาพที่ต้องการนำมาวิเคราะห์พิจารณาและต้องการที่จะกำจัดเมฆ ทำการพล็อตฮีสโตแกรมของภาพ A และฮีสโตแกรมที่ได้จากค่าสัมบูรณ์ของภาพผลต่างระหว่างภาพ A และภาพ B จากนั้นก็ทำการกำหนดหรือเลือกค่าเธร็สโฮลด์ Ta และ Tr จากฮีสโตแกรมทั้งสองตามลำดับ แล้วนำค่ามาเปรียบเทียบกับข้อมูลภาพ A และ B ด้วยค่าเธร็สโฮลด์ Ta ,Tr ควบคู่พร้อมๆกันไป ถ้าหากค่าระดับสีเทาของจุดภาพ A และค่าสัมบูรณ์ของภาพผลต่าง $|A(i,j)-B(i,j)|$ ที่สอดคล้องกับภาพ B นั้นมีค่ามากกว่าเธร็สโฮลด์ทั้งสอง แล้วจุดภาพนี้ก็จะเป็นจุดภาพที่เป็นเมฆของภาพ A ไม่เช่นนั้นก็จะเป็นข้อมูลจุดภาพที่ไม่ใช่เมฆ

3.3 ขั้นตอนในการกำจัดเมฆโดยการใช้เธร็สโฮลด์คู่

ลำดับขั้นตอนในการกำจัดเมฆโดยการใช้เธร็สโฮลด์คู่ สามารถสรุปลำดับขั้นตอนแสดงได้ดังรูปที่ 3.1

วิธีการกำจัดเมฆโดยใช้เธร็สโฮลด์คู่มีลำดับดังต่อไปนี้

1: ในกรณีที่ข้อมูลภาพมีย่านไดนามิค (Dynamic range) แคบแล้ว ให้ทำการดึงค่าจุดภาพอย่างเชิงเส้น (Linear Stretch) ให้กับภาพทั้งสอง เพื่อให้สะดวกต่อการกำหนดและพิจารณาค่าของเธร็สโฮลด์ในการตรวจจับ ตามสมการที่ 3.1

$$y = \frac{(x - x_{min})(y_{max} - y_{min})}{(x_{max} - x_{min})} + y_{min} \qquad (3.1)$$

เมื่อ

x คือ ค่าระดับสีเทาเดิม

y คือ ค่าระดับสีเทาใหม่ที่ผ่านการปรับ

x_{min} คือ ค่าระดับสีเทาต่ำสุดของฮีสโตแกรมเดิม

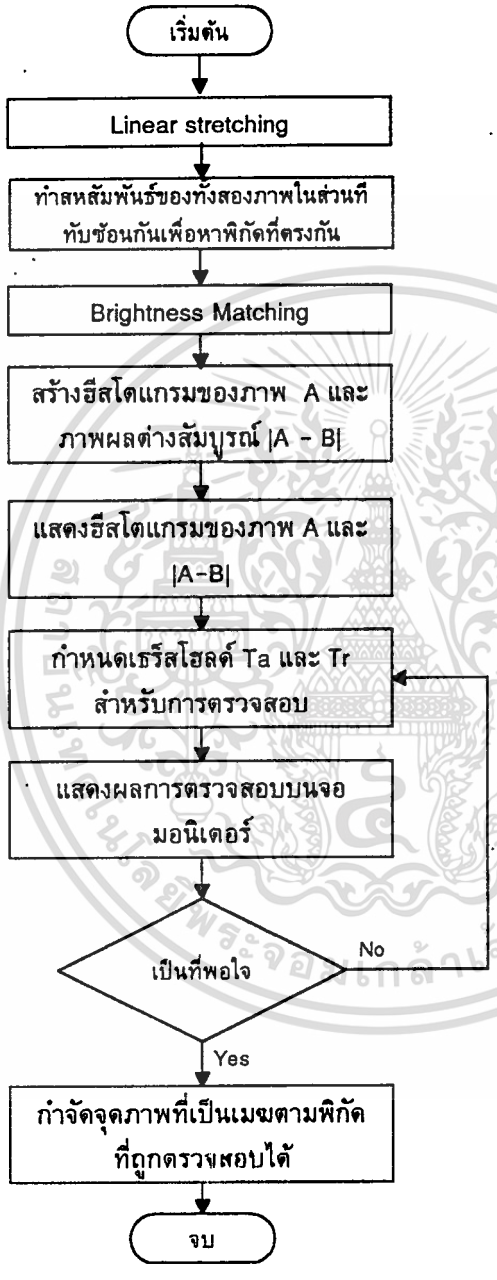
x_{max} คือ ค่าระดับสีเทาสูงสุดของฮีสโตแกรมเดิม

y_{min} คือ ค่าระดับสีเทาต่ำสุดที่ต้องการของฮีสโตแกรมใหม่

y_{max} คือ ค่าระดับสีเทาสูงสุดที่ต้องการของฮีสโตแกรมใหม่

2. การหาสหสัมพันธ์ของข้อมูลภาพที่ทับซ้อนกันเพื่อจะได้พิกัดตำแหน่งของพื้นที่ที่ตรงกันทั้งสองภาพตามสมการที่ 2.1

3. ทำการปรับค่าความสว่าง (Brightness Adjustment) ของจุดภาพทั้งสองให้มีค่าที่สอดคล้องหรือเท่ากันตามสมการที่ 2.3

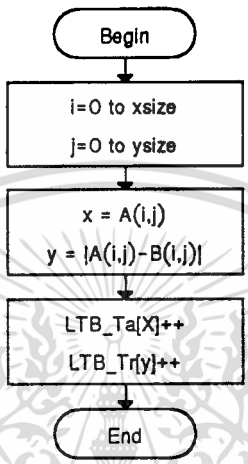


รูปที่ 3.1 แผนผังขั้นตอนในการกำจัดเมฆโดยการใช้เธร์สโฮลด์คู่ (dual thresholds)

4. ทำการสร้างฮีสโตแกรมของภาพ A และฮีสโตแกรมของภาพผลต่างสัมบูรณ์ระหว่างภาพ A และ B ตามแผนผังดังรูปที่ 3.2

5. แสดงฮีสโตแกรมของข้อมูลภาพ A และฮีสโตแกรมของภาพผลต่างสัมบูรณ์ระหว่างภาพ A และภาพ B บนจอมอนิเตอร์

6. ทำการกำหนดหรือเลือกค่าเธร็สโฮลด์ Ta และ Tr โดยการพิจารณาจากผลที่ปรากฏของฮีสโตแกรมภาพ A และค่าผลต่างสัมบูรณ์ $|A(i,j)-B(i,j)|$ ตามลำดับ เพื่อใช้ในการตรวจสอบกลุ่มเมฆ โดยสังเกตจากจอมอนิเตอร์โดยตรง

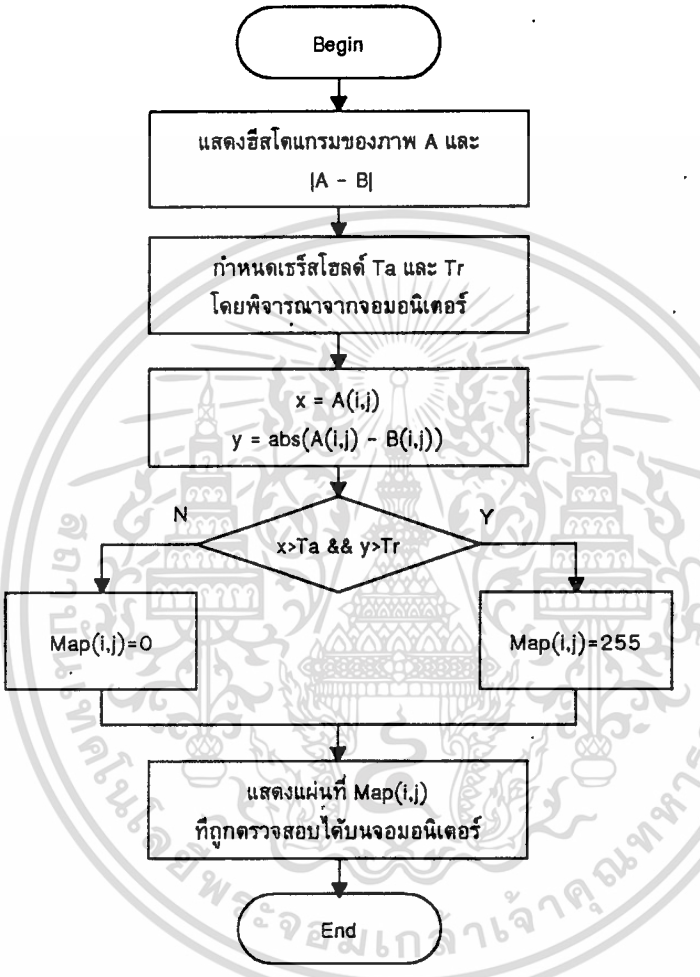


รูปที่ 3.2 แผนผังการสร้างฮีสโตแกรมของภาพแรกและฮีสโตแกรมภาพผลต่างสัมบูรณ์ของภาพทั้งสอง

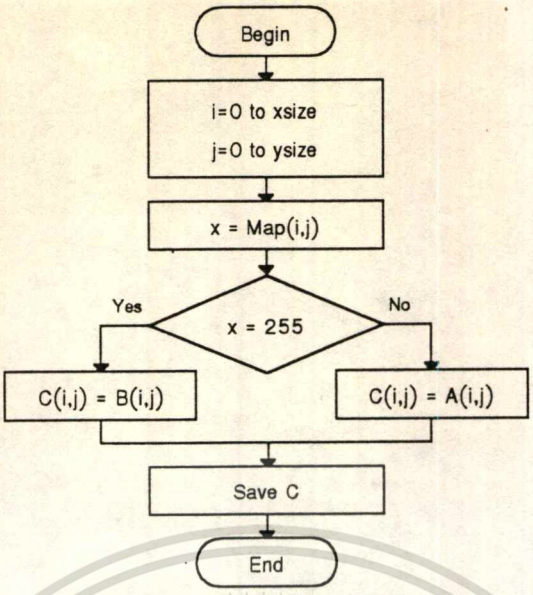
7. สร้างภาพแผนที่แสดงตำแหน่งการกระจายในภาพ A โดยการกำหนดเงื่อนไขให้ $x = A(i,j)$,$y = |A(i,j)-B(i,j)|$ ตลอดทั้งข้อมูลภาพ ถ้า $x > Ta$ และ $y > Tr$ พร้อมๆกันทั้งสองเงื่อนไข เราจะให้ $Map(i,j) = 255$ ไม่เช่นนั้นจะให้ $Map(i,j) = 0$ ค่าของ $Map(i,j)$ ก็คือโครงร่างของแผนที่ ($Map(i,j) = 255$ คือส่วนที่แทนจุดภาพเป็นเมฆ) หรือสามารถแสดงลำดับได้ดังรูปที่ 3.3

8. ตรวจสอบแผนที่ โดยนำมาแสดงผลเทียบกับภาพ A บนจอมอนิเตอร์ ถ้าหากภาพแผนที่มีตำแหน่งตรงกับการกระจายของเมฆในภาพ A ถูกต้องแล้ว ให้ผ่านไปทำกระบวนการขั้นต่อไปได้เลย ไม่เช่นนั้นก็ให้กลับไปทำการกำหนดเธร็สโฮลด์ Ta และ Tr ในข้อ 6 ใหม่อีกครั้ง โดยอาจจะทำการปรับค่าเธร็สโฮลด์ตัวใดตัวหนึ่งหรือทั้งคู่ตามความต้องการ จนกว่าจะเป็นที่พอใจ

9. ทำการกำจัดจุดภาพที่เป็นเมฆ โดยการตรวจสอบว่า ถ้า $Map(i,j) = 255$ แล้วให้ $C(i,j) = B(i,j)$ ไม่เช่นนั้นให้ $C(i,j) = A(i,j)$ซึ่งเป็นการแทนค่าจุดภาพเมฆใน A ด้วยจุดภาพของ B ตลอดทั้งข้อมูลภาพที่ถูกตรวจพบ แล้วก็จะได้ภาพ C ที่ปราศจากเมฆออกมาตามต้องการ ตามแผนผังแสดงดังรูปที่ 3.4



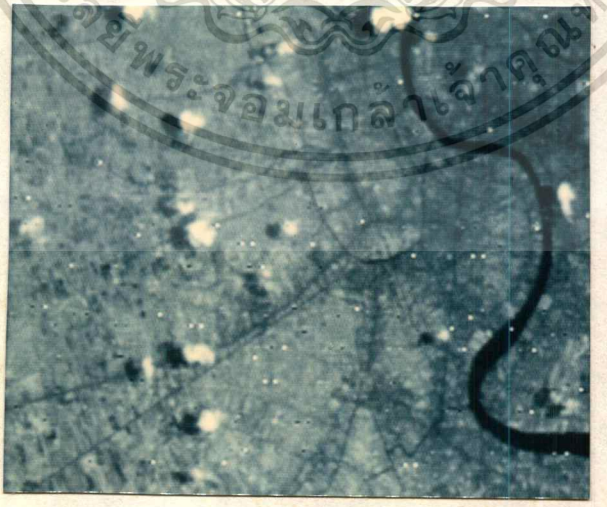
รูปที่ 3.3 แผนผังแสดงขั้นตอนในการสร้างและแสดงแผ่นที่การกระจายของจุดเมฆที่ถูกตรวจสอบได้



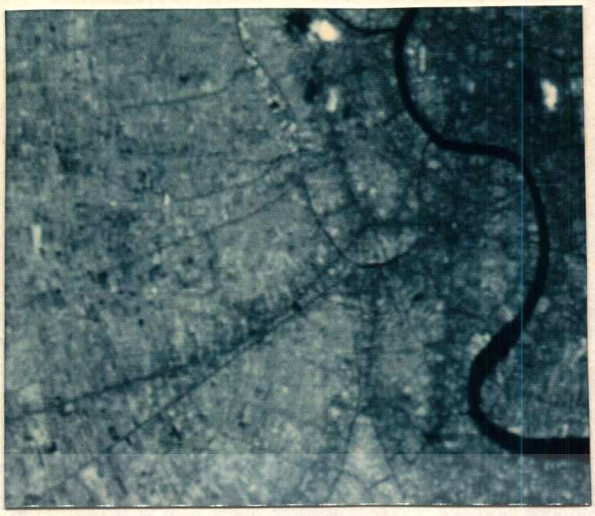
รูปที่ 3.4 แผนผังการกำจัดจุดเมฆโดยการแทนค่าจุดภาพจากตำแหน่งที่สอดคล้องกันกับอีกภาพหนึ่ง

3.4 ผลการทดลอง

จากภาพถ่ายดาวเทียมที่เป็นภาพต้นแบบ แสดงดังรูป 3.3a และ 3.3b ซึ่งจะสังเกตเห็นได้ชัดว่า บริเวณที่เป็นกลุ่มจุดสีขาวนั้น จะเป็นจุดภาพที่เป็นกลุ่มเมฆ โดยที่แต่ละภาพจะมีการกระจายของกลุ่มเมฆในตำแหน่งที่แตกต่างกัน



(a) ภาพ A

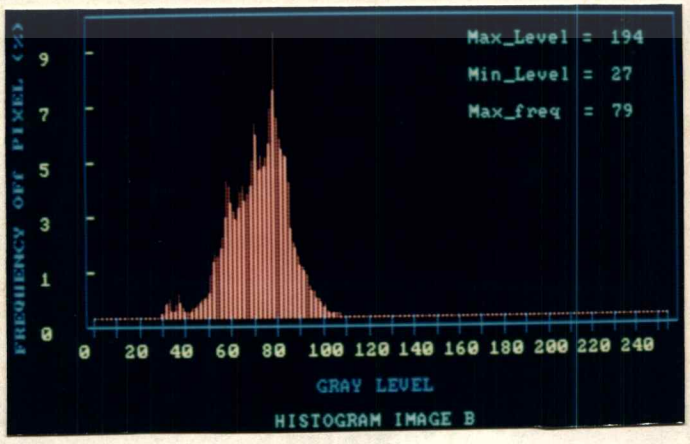


(b) ภาพ B

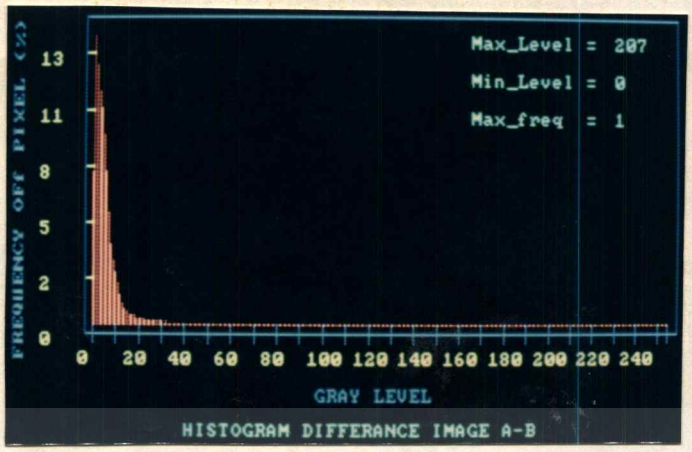
รูปที่ 3.5 แสดงภาพต้นแบบที่ใช้ในการกำจัดเมฆโดยใช้เธร็สโฮลด์คู่

จากข้อมูลภาพที่ 3.5(a) และ 3.5(b) สามารถแสดงฮีสโตแกรมของทั้งสองรูปได้ ดังรูปที่ 3.6(a),3.6(b) ตามลำดับ และฮีสโตแกรมของค่าผลต่างสัมบูรณ์ระหว่างภาพที่3.5(a)กับ3.5(b) แสดงดังรูปที่ 3.6(c)

(a) ฮีสโตแกรมของข้อมูลภาพ A



(b) ฮีสโตแกรมของข้อมูลภาพ B

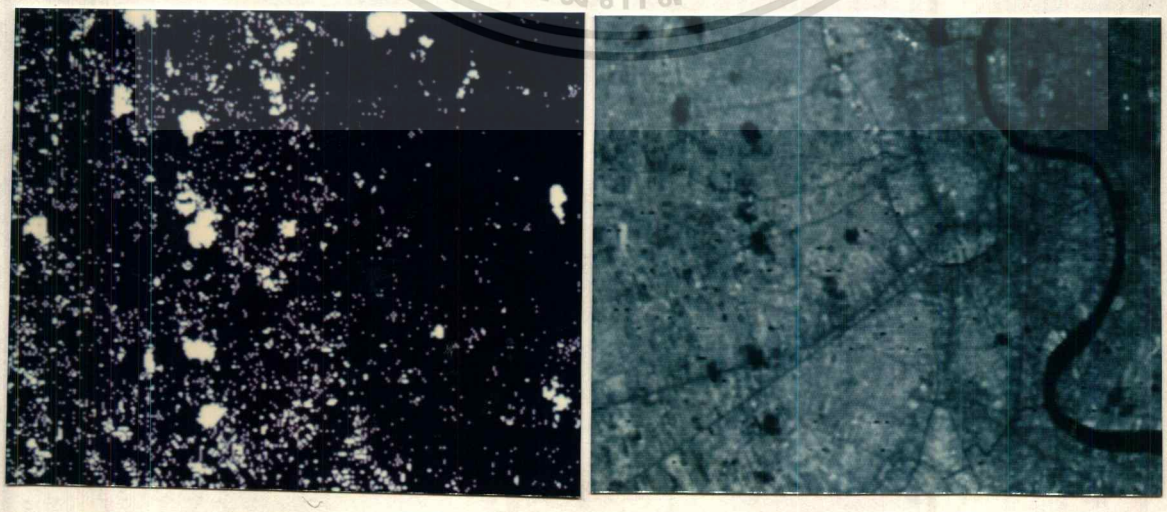


(c) ฮีสโตแกรมของข้อมูลภาพผลต่างสัมบูรณ์

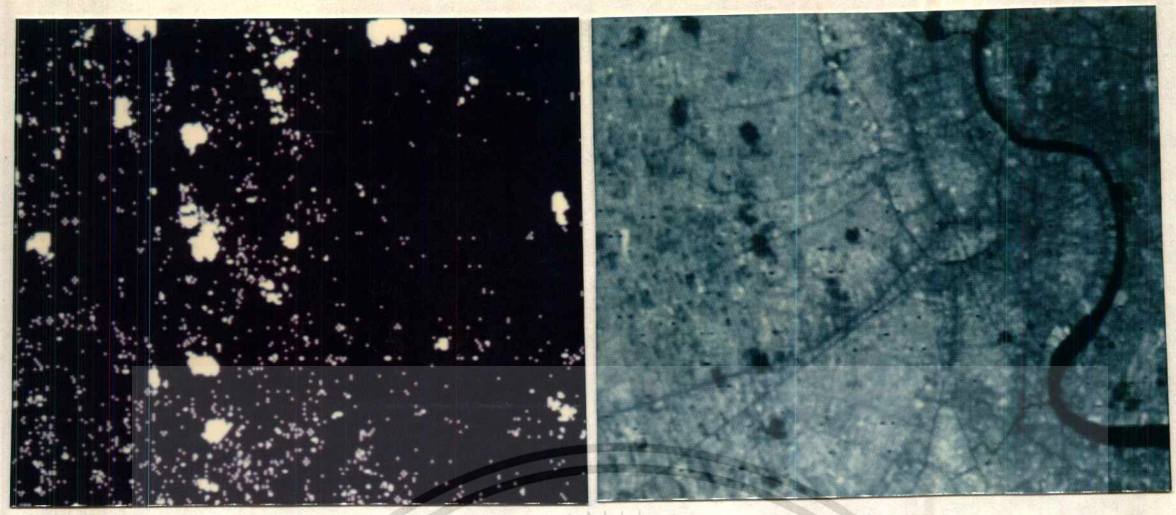
รูปที่ 3.6 แสดงฮีสโตแกรมของข้อมูลภาพที่ใช้ในการกำจัดเมฆโดยการกำหนดเธร็สโฮสด์คู่

พิจารณาฮีสโตแกรมจากภาพที่ 3.6(a) ค่าของเธร็สโฮลด์ Ta สามารถคาดคะเนได้ว่า ควรจะมีค่าประมาณตั้งแต่ 80 ขึ้นไป และจากจากรูปที่ 3.6(c) ความถี่ของฮีสโตแกรมมีค่าสูงสุดที่ระดับสีเทาประมาณ 3 ดังนั้นค่าของเธร็สโฮลด์ Tr ควรจะกำหนดให้มีค่ามากกว่าค่านี้ขึ้นไป

ในการทดลองการตรวจสอบและกำจัดเมฆโดยการใช้เธร็สโฮลด์คู่ที่ระดับแตกต่างกันหลายๆค่าสำหรับ Ta และ Tr ภาพผลลัพธ์จากการกำจัดเมฆแสดงให้เห็นได้ดังรูปที่ (3.7) โดยการกำหนดค่าเธร็สโฮลด์คู่ (Ta, Tr) เป็นค่า (80,5), (85,6) ,และ(90,10) ตามลำดับ โดยที่ในรูปที่ 3.7(1a,2a,3a) แสดงจุดภาพของแผนที่ที่ตรวจจับได้ ซึ่งพื้นที่สีขาวหมายถึงจุดภาพที่เป็นเมฆ และพื้นที่สีแดงหมายถึงจุดภาพที่ไม่ใช่เมฆ ส่วนรูปที่3.7(1b,2b,3b) แสดงผลลัพธ์ของการกำจัดเมฆ



(1a) (1b)



(2a) (2b)

(3a) (3b)

รูปที่ 3.7 แสดงภาพผลลัพธ์จากการกำจัดเมฆแบบใช้เธร็สโฮลด์คู่ที่ระดับต่างๆ

3.5 สรุป

วิธีการในบทนี้มีข้อเสียคือไม่สามารถจำแนกให้ชัดเจนได้ว่า ส่วนไหนเป็นเมฆในภาพ A และส่วนไหนเป็นเมฆในภาพ B จากฮีสโตแกรมค่าสัมบูรณ์ จึงทำให้การกำหนดเธร็สโฮลด์ที่ดีที่สุดทำได้ยาก จากการใช้ฮีสโตแกรมภาพผลต่างค่าสัมบูรณ์ จะพบว่าเงาเมฆซึ่งปกติจะมีค่าเป็นลบในภาพผลต่างถูกทำให้เป็นบวกด้วยค่าสัมบูรณ์ จึงทำให้ไม่สามารถกำจัดเงาเมฆด้วยวิธีตามเทคนิคนี้ได้

บทที่ 4

การกำจัดเมฆและเงาเมฆที่ปกคลุมโดยใช้ฮีสโตแกรมสองมิติ

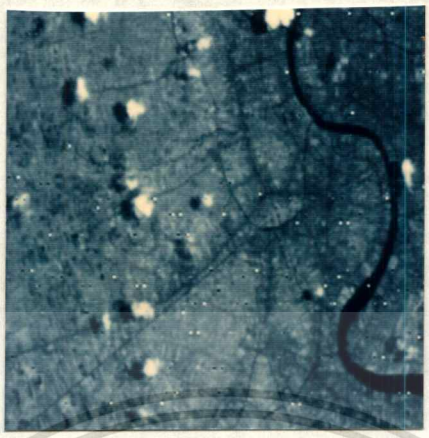
Cloud cover and cloud shadow removing base on 2-dimemsional histogram

4.1 บทนำ

จากวิธีการกำจัดเมฆโดยการใช้เธร็ซโฮลด์คู่จะเห็นได้ว่าเป็นวิธีที่สามารถกำจัดได้เฉพาะจุดภาพที่เป็นเมฆเท่านั้นซึ่งไม่สามารถตรวจสอบและกำจัดจุดภาพที่เป็นเงาได้ ตลอดทั้งทำให้ขาดความสะดวกในการใช้งาน ดังนั้นเพื่อแก้ไขข้อบกพร่องดังกล่าว จึงได้เสนอเทคนิควิธีการใหม่โดยการใช้ฮีสโตแกรมสองมิติ เป็นตัวช่วยในการพิจารณาตรวจจับและกำจัดจุดภาพที่เป็นทั้งกลุ่มเมฆและเงาเมฆ โดยการใช้ภาพถ่ายดาวเทียมจำนวนสองภาพที่ถูกบันทึกไว้ในช่วงเวลาที่แตกต่างกัน เป็นบริเวณพื้นที่เดียวกัน และจุดภาพที่เป็นเมฆหรือเงาเมฆจะต้องมีการกระจายอยู่ในตำแหน่งพื้นที่ที่แตกต่างกันเป็นสำคัญ ซึ่งวิธีการตรวจสอบและกำจัดโดยการกำหนดขอบเขตของเธร็สโฮลด์จำนวนสองกลุ่มขึ้นมาสำหรับเป็นค่าเงื่อนไขในการตรวจสอบ ดังนั้นเราสามารถแทนค่าของจุดภาพที่เป็นเมฆหรือเงาเมฆด้วยจุดภาพจากอีกภาพตามตำแหน่งที่สอดคล้องกันได้โดยพร้อมๆกันตลอดทั้งภาพ

4.2 หลักวิธีการ

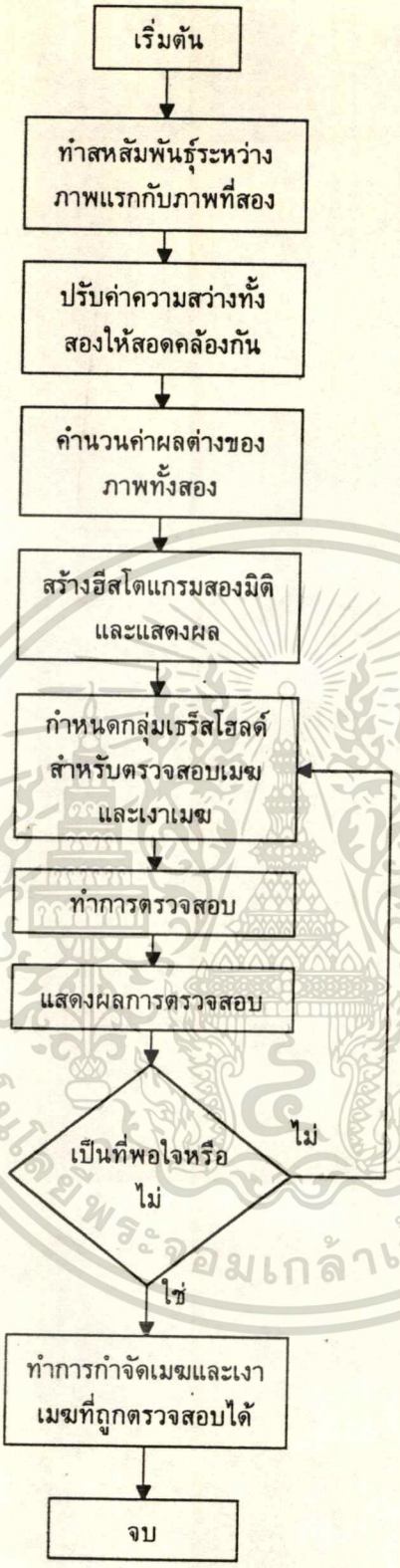
จากรูปที่ 4.1a และ 4.1b เป็นภาพถ่ายดาวเทียมในย่านความถี่ที่มองเห็นด้วยตา ซึ่งเป็นภาพถ่ายที่ถูกบันทึกหรือถ่ายไว้คนละช่วงเวลา พร้อมกันนี้ ก็มีการกระจายของกลุ่มเมฆในตำแหน่งที่แตกต่างกัน ทำการสร้างฮีสโตแกรมแบบสองมิติขึ้นมา โดยที่แกนแรกของฮีสโตแกรมสองมิติ จะแสดงให้เห็นถึงการกระจายค่าระดับความเข้มของจุดภาพที่เราต้องการจะกำจัดเมฆและเงาเมฆ ส่วนแกนที่สองของฮีสโตแกรมสองมิติ จะเป็นส่วนของค่าผลต่างระหว่างภาพถ่ายดาวเทียมทั้งสอง ซึ่งก็จะเป็นส่วนที่แสดงถึงค่าความแตกต่างของจุดภาพทั้งสอง จากนั้นก็ทำการกำหนดขอบเขตค่าเธร็สโฮลด์ขึ้นมาสองกลุ่ม โดยที่กลุ่มแรกจะใช้สำหรับการตรวจจับจุดภาพที่เป็นส่วนของกลุ่มเมฆ และในส่วนของกลุ่มที่สอง จะถูกใช้สำหรับการตรวจจับจุดภาพที่เป็นเงาเมฆ



รูปที่ 4.1a แสดงข้อมูลภาพถ่ายดาวเทียมในภาพแรกที่ต้องการกำจัดเมฆและเงาเมฆ

รูปที่ 4.1b แสดงภาพถ่ายดาวเทียมในภาพที่สอง

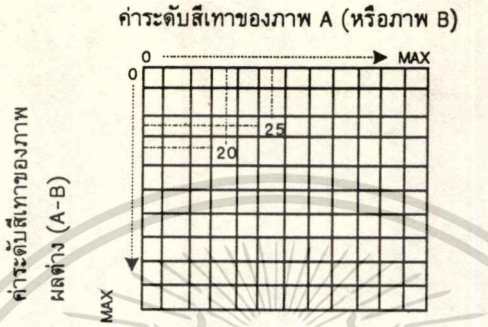
วิธีการกำจัดเมฆและเงาเมฆโดยใช้ฮีสโตแกรมสองมิติสามารถสรุปขั้นตอนวิธีการประมวลผลแสดงดังรูปที่ 4.2



รูปที่ 4.2 แสดงตารางกระบวนการกำจัดเมฆและเงาเมฆด้วยฮีสโตแกรมสองมิติ

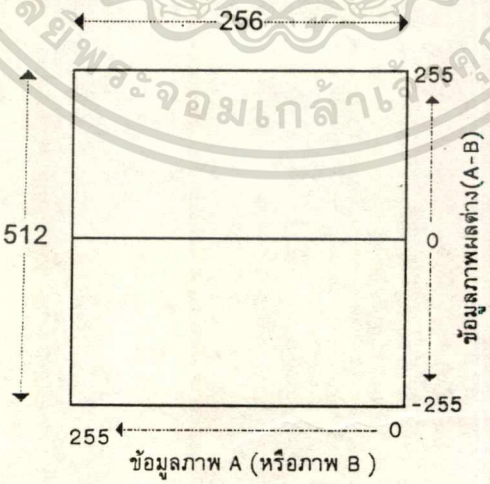
4.3 วิธีการสร้างฮีสโตแกรมสองมิติ

วิธีในการสร้างฮีสโตแกรมสองมิติ สามารถทำได้โดยการสร้าง Look-up table อะเรย์สองมิติ โดยอาศัยข้อมูลภาพถ่ายดาวเทียมจำนวนสองภาพแสดงได้ดังรูปที่ 4.3 ซึ่งแต่ละแกนของตารางสองมิติจะแทนค่าระดับสีเทาของข้อมูลภาพใดภาพหนึ่ง ข้อมูลที่อยู่ภายในตารางจะแสดงจำนวนความถี่หรือปริมาณของจุดภาพที่ระดับความเข้มค่าต่างๆตามความสัมพันธ์ของข้อมูลภาพทั้งสอง

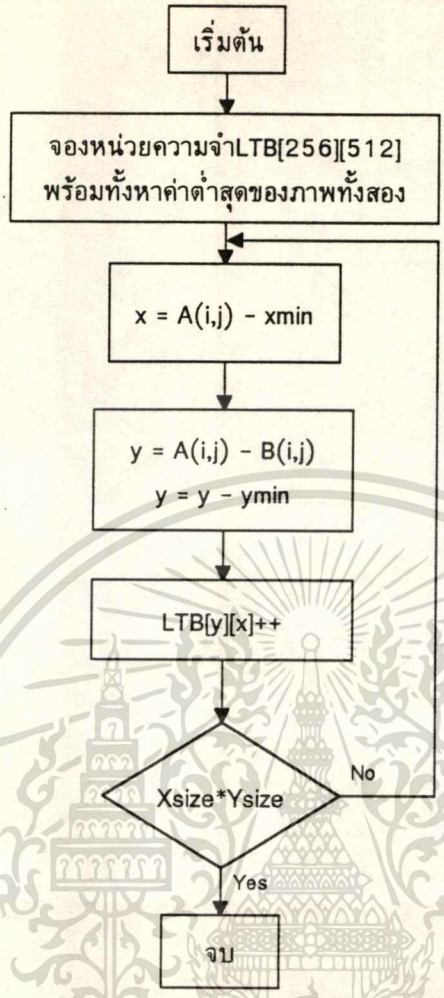


รูปที่ 4.3 ตารางอะเรย์ที่ใช้ในการสร้างฮีสโตแกรมสองมิติ

สำหรับตาราง look-up table ที่ใช้ในการกำจัดเมฆและเงาเมฆในวิทยานิพนธ์นี้ จะมีขนาดของอะเรย์สองมิติเท่ากับ [256x512] โดยที่ข้อมูลในแนวแกน x จะแทนค่าระดับสีเทาของภาพแรกที่ต้องการกำจัดเมฆและเงาเมฆซึ่งมีค่าระหว่าง 0 ถึง 255 ขณะเดียวกันในทางแกน y จะแทนข้อมูลของภาพผลต่างระหว่างภาพแรกกับภาพที่สองที่ใช้ในการเปรียบเทียบ โดยมีค่าระหว่าง -255 ถึง 255 แสดงได้ดังรูปที่ 4.4 และลำดับขั้นตอนในการสร้างฮีสโตแกรมสองมิติแสดงดังรูปที่ 4.5



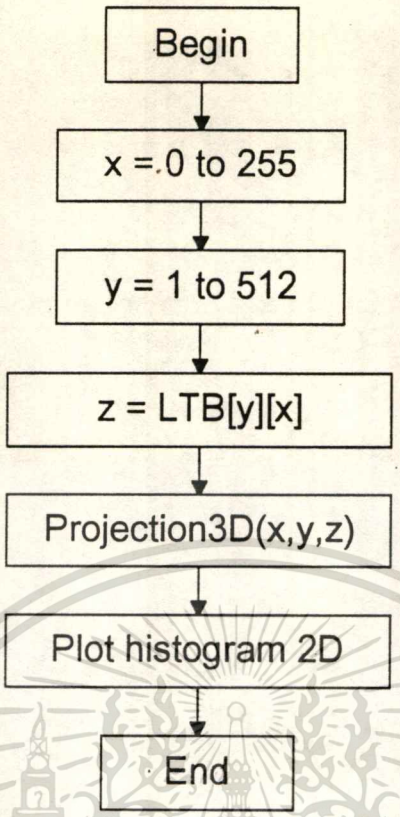
รูปที่ 4.3 ตาราง Look-Up Table ขนาด[256x512]ที่ใช้เก็บค่าฮีสโตแกรมสองมิติจากข้อมูลภาพทั้งสอง



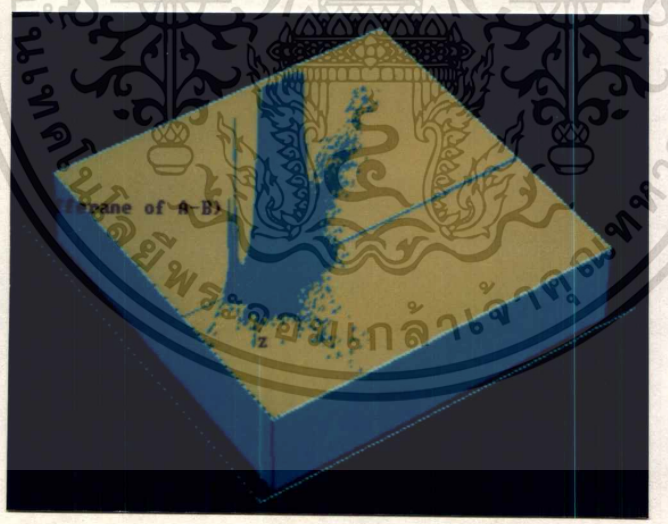
รูปที่ 4.5 ตารางแสดงขั้นตอนในการสร้างฮีสโตแกรมสองมิติจากข้อมูลภาพทั้งสอง

4.4 วิธีการแสดงข้อมูลของฮีสโตแกรมสองมิติ

วิธีการแสดงข้อมูลของฮีสโตแกรมสองมิติสามารถทำได้โดยการนำเอาข้อมูลในตาราง Look Up Table ที่มีขนาด 256x512 มาทำการแสดงให้ปรากฏบนจอมอเตอร์ โดยการโปรเจ็คชั่นตามค่าที่อ่านได้ในรูปแบบของสามมิติ ตามแผนผังดังรูปที่ 4.6 และแสดงภาพที่ได้จากการพล็อกฮีสโตแกรมสองมิติของข้อมูลภาพถ่ายดาวเทียมในรูปที่4.1(a)และภาพผลต่างระหว่างรูปที่ 4.1(a)กับรูปที่ 4.1(b) แสดงได้ดังรูปที่ 4.7



รูปที่ 4.6 แผนผังขั้นตอนในการแสดงฮีสโตแกรมสองมิติ



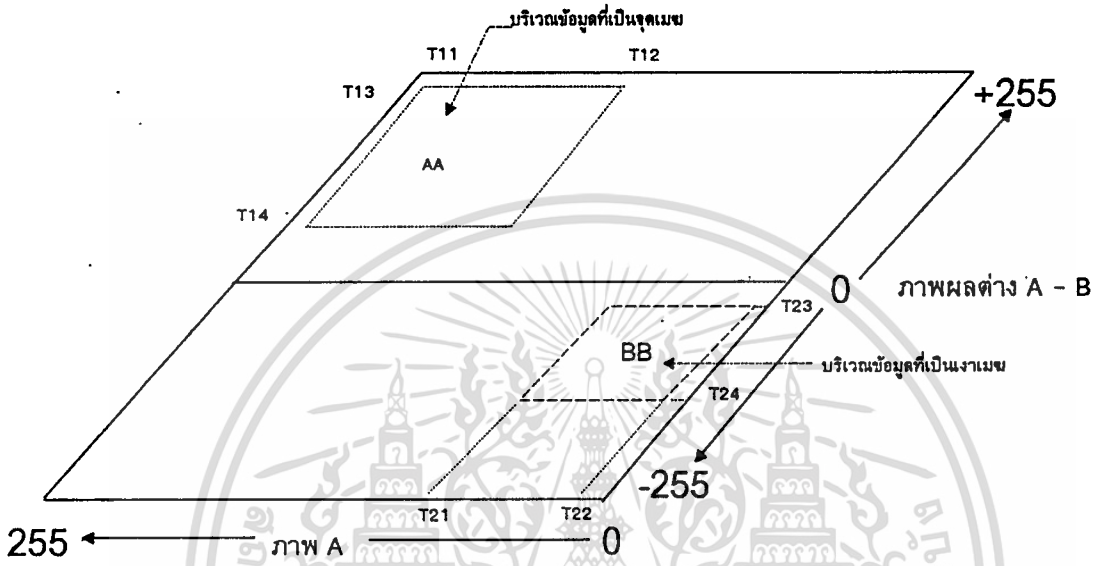
รูปที่ 4.7 ผลของการพล็อตฮีสโตแกรมสองมิติบนจอมอนิเตอร์

4.5 คุณสมบัติของเมฆและเงาเมฆที่ปรากฏบนฮีสโตแกรมสองมิติ

จากการพล็อตค่าฮีสโตแกรมสองมิติสามารถที่จะสมมติฐานข้อมูลที่ปรากฏได้ดังนี้

4.5.1. ข้อมูลจุดภาพที่เป็นเมฆ จะให้ค่าระดับความเข้มของระดับสีเทาและค่าระดับผลต่างของ (A-B) ที่สูง ดังนั้น ข้อมูลเหล่านี้จะปรากฏบริเวณตำแหน่ง AA ดังในรูปที่ 4.7

4.5.2. ข้อมูลจุดภาพที่เป็นเงาเมฆ จะให้ค่าระดับความเข้มของระดับสีเทาที่ต่ำทางแนวนอน และ ค่าผลต่างของภาพ (A-B) จะมีค่าเป็นลบในทางแกนตั้ง ซึ่งข้อมูลเหล่านี้ก็จะไปปรากฏ ณ บริเวณตำแหน่ง BB แสดงดังรูปที่ 4.7



รูปที่ 4.7 แสดงคุณสมบัติของข้อมูลเมฆและเงาเมฆที่ปรากฏบนฮีสโตแกรมสองมิติ

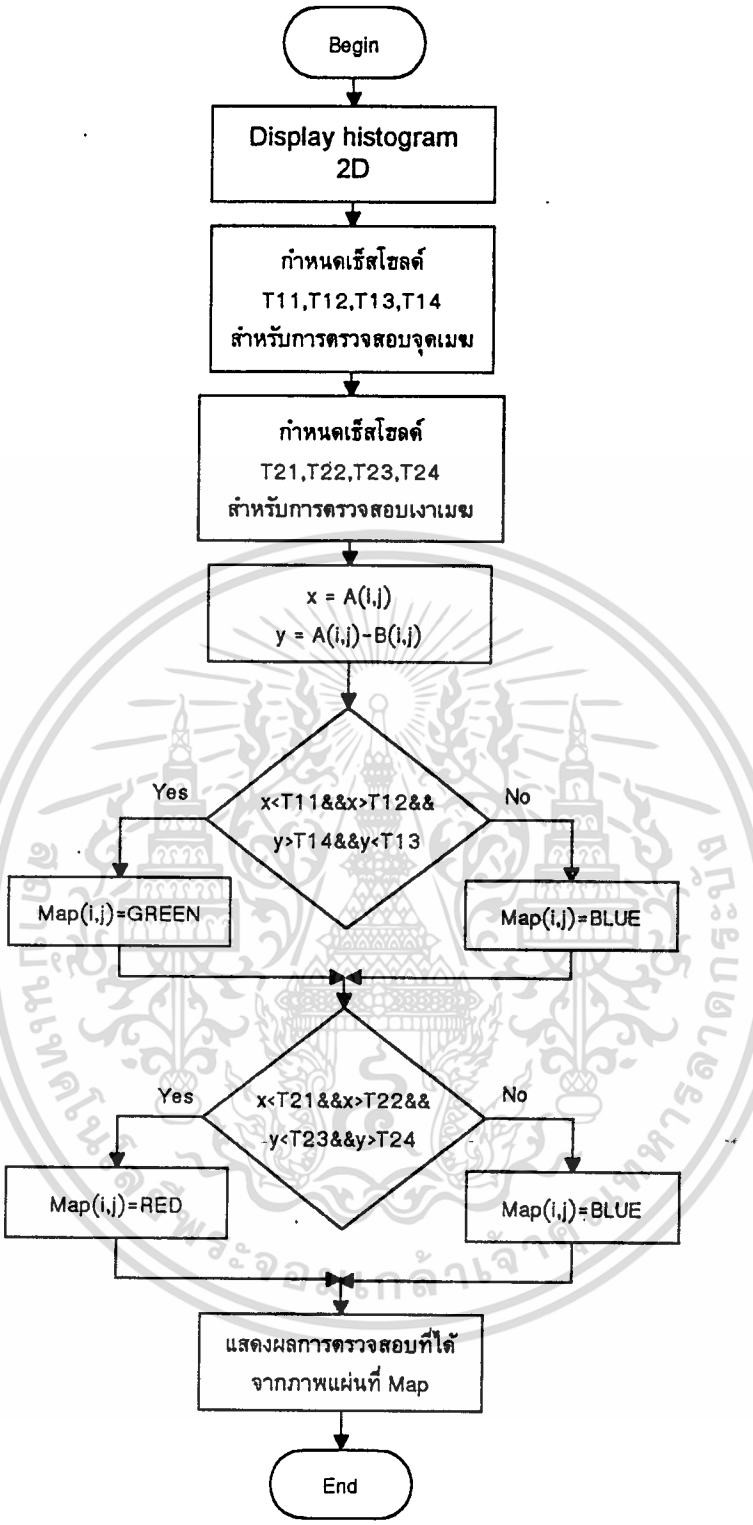
4.6 วิธีการตรวจสอบและกำจัดเมฆหรือเงาเมฆที่ปรากฏบนภาพถ่ายดาวเทียม

จากการที่เราทราบคุณสมบัติของเมฆและเงาเมฆที่ปรากฏบนฮีสโตแกรมสองมิติแล้ว เราก็สามารถกำหนดขอบเขตเงื่อนไขของเธร็สโฮสด์ขึ้นมาสองกลุ่มสำหรับการตรวจสอบกลุ่มเมฆและเงาเมฆที่ปรากฏบนภาพถ่ายดาวเทียม โดยสามารถกำหนดได้ดังนี้

4.6.1. กำหนดเธร็สโฮลด์ในกลุ่มแรกคือ T11, T12, T13, T14 สำหรับการตรวจสอบจุดภาพที่เป็นกลุ่มเมฆ ทำได้โดยการพิจารณาผลที่ปรากฏบนจอมอนิเตอร์โดยตรง

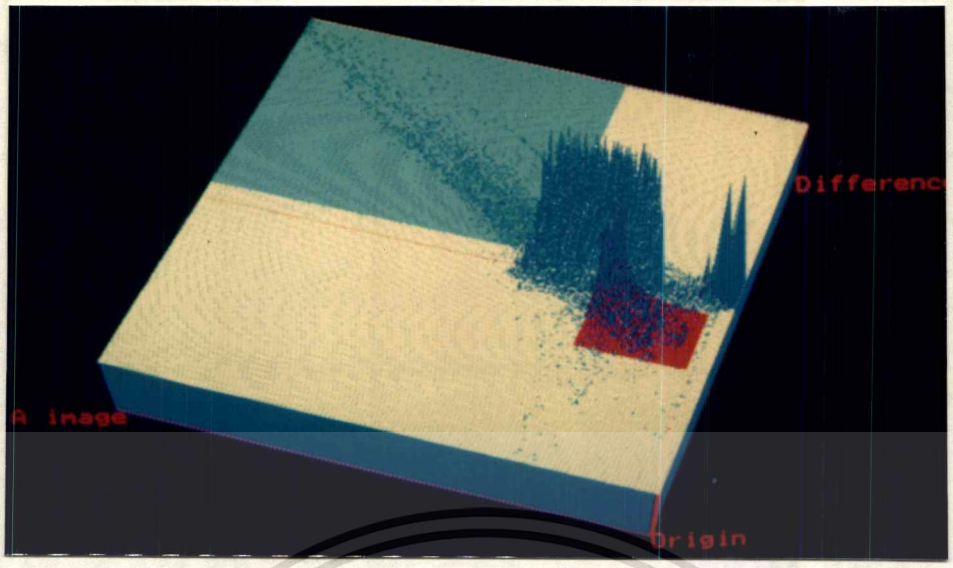
4.6.2. กำหนดเธร็สโฮลด์ในกลุ่มที่สองคือ T21, T22, T23, T24 สำหรับการตรวจสอบจุดภาพที่เป็นเงาเมฆ ทำได้โดยการพิจารณาผลที่ปรากฏบนจอมอนิเตอร์โดยตรง

ลำดับขั้นตอนในการกำหนดค่าเธร็สโฮล์จำนวนสองกลุ่มสำหรับการตรวจสอบจุดเมฆและเงาเมฆพร้อมทั้งเงื่อนไขการตรวจสอบแสดงดังรูปที่ 4.8



รูปที่ 4.8 แผนผังลำดับขั้นตอนในการตรวจสอบจุดภาพที่เป็นเมฆและเงาเมฆของภาพ A

ฮีสโตแกรมสองมิติที่แสดงในรูปที่ 4.9 เป็นรูปที่ได้ทำการกำหนดกลุ่มเธร็สโฮลด์สำหรับการตรวจสอบจุดเมฆและเงาเมฆ โดยพื้นที่สีเขียวจะเป็นขอบเขตบริเวณที่ใช้ในการตรวจสอบจุดเมฆ และพื้นที่สีแดงเป็นบริเวณที่ใช้ในการตรวจสอบเงาเมฆ



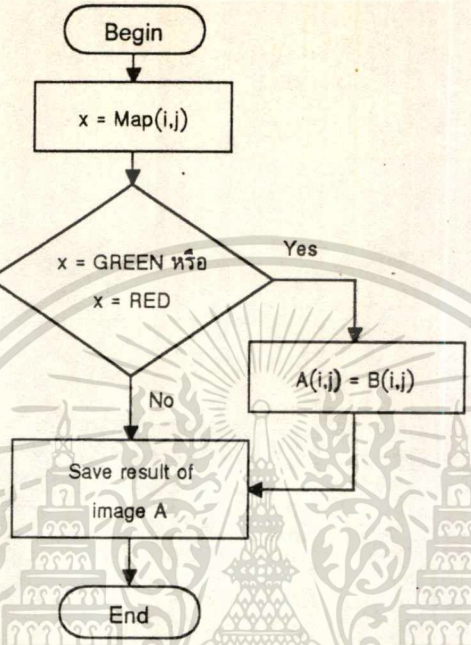
รูปที่ 4.9 แสดงฮีสโตแกรมสองมิติ ซึ่งประกอบด้วยแกนของภาพที่ต้องการกำจัดเมฆ เงาเมฆ และแกนค่าผลต่างของภาพทั้งสอง โดยที่พื้นที่สีเขียวแสดงถึงบริเวณที่มีเมฆปกคลุม ในขณะที่พื้นที่สีแดง แสดงถึงบริเวณที่มีเงาเมฆปกคลุม

เพื่อความสะดวกในการตรวจสอบผลที่ปรากฏออกมา สามารถทำได้โดยการสร้างแผนที่ที่แสดงตำแหน่งของจุดภาพที่เป็นเมฆและเงาเมฆเปรียบเทียบกับภาพถ่ายดาวเทียมที่ต้องการกำจัด เพื่อการตัดสินใจว่าถูกต้องตามที่ต้องการหรือไม่ ซึ่งแสดงดังรูปที่ 4.10

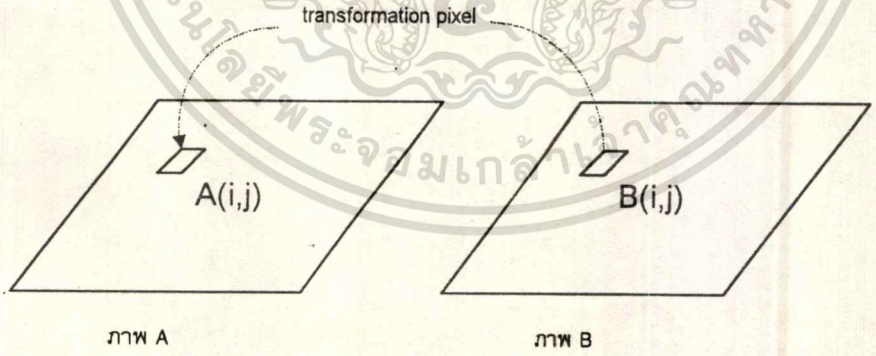
รูปที่ 4.10 แสดงตำแหน่งจุดภาพของเมฆและเงาเมฆที่ตรวจสอบได้จากภาพแรก

หลังจากที่ได้ทำการตรวจสอบด้วยสายตาจากจอมอนิเตอร์โดยการเปรียบเทียบผลที่ได้จากภาพแผนที่ที่แสดงข้อมูลที่ถูกตรวจจับได้กับภาพต้นแบบดังในรูปที่ 4.1(a) จนเป็นที่พอใจแล้ว

ขั้นตอนในการกำจัดจุดภาพที่เป็นเมฆหรือเงาเมฆ ทำได้โดยการแทนค่าจุดภาพเหล่านี้ด้วยจุดภาพของอีกภาพหนึ่ง ในตำแหน่งที่สอดคล้องกันตลอดทั้งภาพ ซึ่งแผนผังในการกำจัดเมฆและเงาเมฆแสดงดังรูปที่ 4.11 และรูปที่4.12 แสดงการแทนค่าจุดภาพจากภาพ B มายังภาพ A ณ ตำแหน่งใดๆที่สอดคล้องกัน



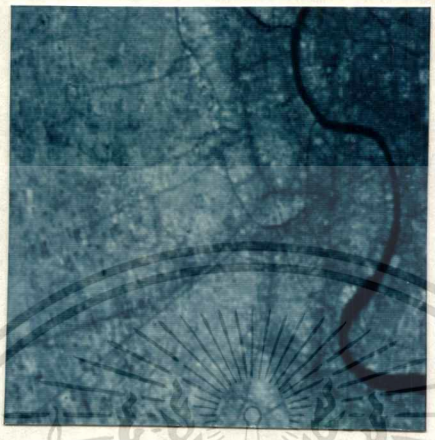
รูปที่ 4.11 แผนผังการแทนค่าจุดภาพที่ถูกตรวจจับได้ของภาพ A



รูปที่ 4.12 แสดงวิธีการแทนค่าจุดภาพที่ถูกตรวจสอบว่าเป็นเมฆหรือเงาเมฆ ด้วยจุดภาพจากอีกภาพหนึ่งในตำแหน่งที่สอดคล้องกัน

4.7 ผลการทดลอง

จากรูปที่ 4.13 แสดงถึงภาพผลลัพธ์ที่ได้จากการกำจัดเมฆและเงาเมฆโดยพร้อมๆกัน ซึ่งจะสังเกตได้ว่า เราสามารถที่จะกำจัดสิ่งเหล่านี้ได้อย่างดี ส่งผลทำให้ได้ภาพถ่ายดาวเทียมที่ปราศจากเมฆและเงาเมฆตามที่ต้องการ



รูปที่ 4.13 แสดงภาพผลลัพธ์จากการกำจัดกลุ่มเมฆและเงาเมฆด้วยฮีสโตแกรมสองมิติ

4.8 ผลสรุป

วิธีการกำจัดเมฆและเงาเมฆนี้จะเห็นได้ว่า สามารถที่จะกำหนดค่าเธร็สโฮลด์สำหรับการตรวจจับและกำจัดจุดภาพที่เป็นเมฆหรือเงาเมฆได้เป็นอย่างดี ภาพหลังจากที่ได้ทราบคุณสมบัติของจุดภาพที่ปรากฏบนฮีสโตแกรมสองมิติ โดยสามารถกำจัดทั้งเมฆและเงาเมฆได้โดยพร้อมๆกันภายในกระบวนการเดียวกัน ทำให้เพิ่มประสิทธิภาพในการกำจัดได้มากขึ้นกว่าวิธีแบบใช้เธร็สโฮลด์คู่ แต่อย่างไรก็ตาม กระบวนการกำจัดเมฆและเงาเมฆจากวิธีการในบทนี้ ต้องแยกพิจารณาและประมวลผลทีละภาพ จึงทำให้ด้อยประสิทธิภาพในการทำงาน ดังนั้นในบทต่อไปจะเป็นการกำจัดเมฆและเงาเมฆโดยพิจารณาและประมวลผลในภาพทั้งสองพร้อมๆกัน

บทที่ 5
การกำจัดเมฆและเงาเมฆพร้อมๆกันในภาพถ่ายดาวเทียมสองภาพ ด้วยฮีสโตแกรมสามมิติ

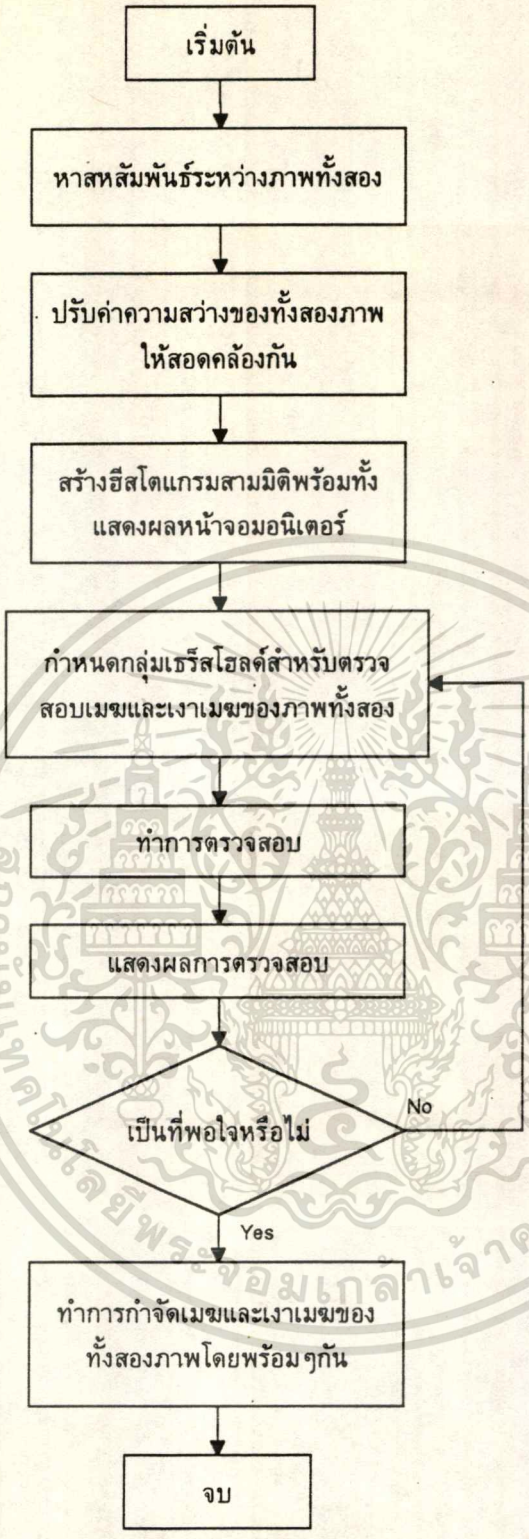
Cloud and shadow removing from two satellite images simultaneously using 3-dimensional histogram

5.1 บทนำ

จากบทที่แล้วได้กล่าวถึงวิธีการกำจัดเมฆและเงาเมฆด้วยฮีสโตแกรมสองมิติ ซึ่งจะสังเกตได้ว่า วิธีการดังกล่าวสามารถกำจัดเพียงครั้งละภาพเท่านั้น ดังนั้นในบทนี้จึงได้ทำการปรับปรุงวิธีการดังกล่าวขึ้น โดยเสนอเทคนิคในการกำจัดจุดภาพที่เป็นเมฆและเงาเมฆบนภาพถ่ายดาวเทียมด้วยการใช้ฮีสโตแกรมสามมิติ ฮีสโตแกรมนี้สร้างได้จากการใช้ฮีสโตแกรมของภาพถ่ายดาวเทียมในแถบความยาวคลื่นที่มองเห็นด้วยตาจำนวนสองภาพ ที่มีการกระจายของเมฆและเงาเมฆต่างกันออกไป ฮีสโตแกรมของภาพทั้งสอง จะถูกนำมาวางอยู่บนแกนทั้งสองของฮีสโตแกรมสามมิติ ส่วนแกนที่สามนั้น จะเป็นข้อมูลฮีสโตแกรมของภาพผลต่างที่เกิดจากภาพทั้งสอง ข้อมูลที่ปรากฏบนฮีสโตแกรมสามมิตินี้ ก็จะมีคุณสมบัติของจุดภาพที่เป็นเมฆหรือเงาเมฆของทั้งสองภาพในตำแหน่งที่แตกต่างกัน ทำให้สามารถกำหนดขอบเขตของกลุ่มเธร็สโฮลด์เป็นจุดอ้างอิงสำหรับการกำจัดเมฆและเงาเมฆ และจากการใช้กลุ่มเธร็สโฮลด์กับฮีสโตแกรมสามมิตินี้ จะสามารถตรวจจับบริเวณเมฆและเงาเมฆจากภาพทั้งสองได้ การกำจัดเมฆและเงาเมฆของภาพทั้งสอง ทำได้โดยการแทนจุดภาพเหล่านั้น ด้วยจุดภาพในตำแหน่งที่สอดคล้องของอีกภาพหนึ่ง เทคนิคการกำจัดเมฆและเงาเมฆแบบที่เสนอนี้ มีข้อดีเหนือกว่าเทคนิคอื่น ตรงที่สามารถกำจัดได้ทั้งเมฆและเงาเมฆ ตลอดทั้งสัญญาณรบกวนประเภท Salt และ Pepper ในภาพทั้งสองได้พร้อมๆกัน

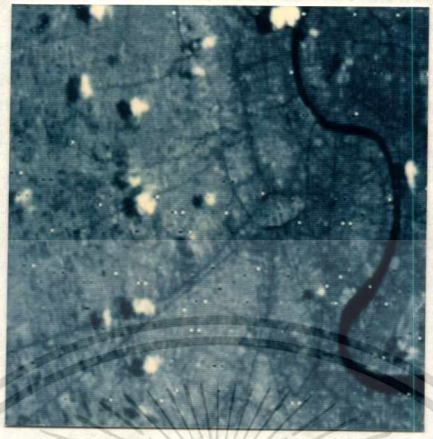
5.2 หลักวิธีในการกำจัดเมฆและเงาเมฆด้วยฮีสโตแกรมสามมิติ

วิธีการและขั้นตอนในการกำจัดเมฆและเงาเมฆพร้อมๆกันในภาพถ่ายดาวเทียมสองภาพ ด้วยฮีสโตแกรมสามมิติสามารถแสดงได้ดังรูปที่ 5.1

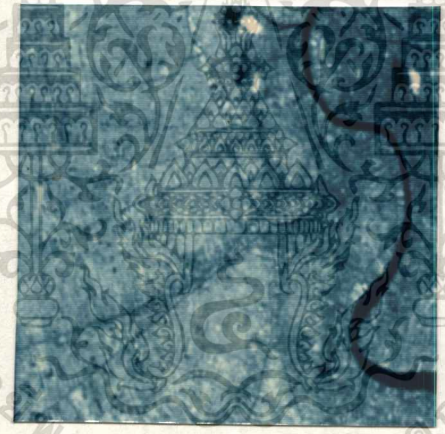


รูปที่ 5.1 แผนผังขั้นตอนในการกำจัดเมฆและเงาเมฆโดยพร้อมๆกันในภาพถ่ายดาวเทียม จำนวนสองภาพด้วยฮีสโตแกรมสามมิติ

ข้อมูลภาพถ่ายดาวเทียมที่ใช้ในกำจัดเมฆและเงาเมฆแสดงดังรูปที่ 5.2 โดยเป็นภาพในพื้นที่เดียวกันซึ่งผ่านการทำสหสัมพันธ์และการปรับค่าความสว่างของภาพให้สอดคล้องกันตามสมการที่ได้กล่าวมาแล้วในบทที่ 2



(a) ภาพดาวเทียมภาพแรก



(b) ภาพดาวเทียมภาพที่สอง

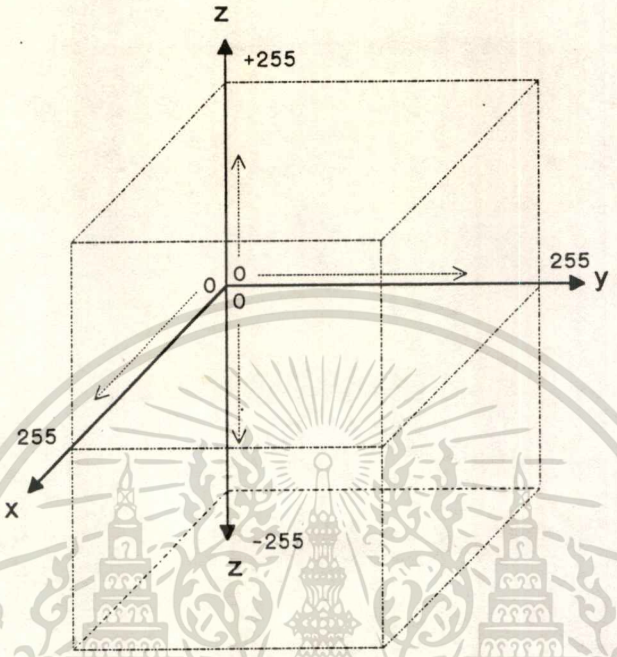
รูปที่ 5.2 ภาพถ่ายดาวเทียมทั้งสองที่ใช้ในการกำจัดเมฆและเงาเมฆ

5.3 วิธีการสร้างฮีสโตแกรมสามมิติ

5.3.1 การสร้างสี่เหลี่ยมลูกบาศก์เพื่อกำหนดขอบเขตของฮีสโตแกรมสามมิติ

ในการแสดงฮีสโตแกรมสามมิติจากข้อมูลภาพถ่ายดาวเทียมทั้งสองนั้น ในขั้นแรกจะต้องทำการสร้างสี่เหลี่ยมลูกบาศก์ในลักษณะรูปสามมิติ เพื่อใช้แสดงขอบเขตของฮีสโตแกรมสามมิติภายใต้ลูกบาศก์ที่ทำการสร้างไว้ เพื่อสะดวกต่อการกำหนดกลุ่มของเธร็สโฮลด์สำหรับการตรวจสอบและกำจัดเมฆหรือเงาเมฆของภาพทั้งสอง ตลอดทั้งสามารถกำหนดจุดมุมมองจากด้าน

ต่างๆได้คล่องตัวและพิจารณากระจายของจุดภาพที่ปรากฏบนฮีสโตแกรมสามมิติได้ง่าย โดยกำหนดให้ทางด้านแกน x แทนค่าระดับสีเทาตั้งแต่ 0 ถึง 255 ทางด้านแกน y จะแทนค่าระดับสีเทาตั้งแต่ 0 ถึง 255 เช่นกัน ขณะเดียวกันส่วนของแกน z จะมีค่าระหว่าง -255 ถึง +255 แสดงได้ดังรูปที่ 5.3



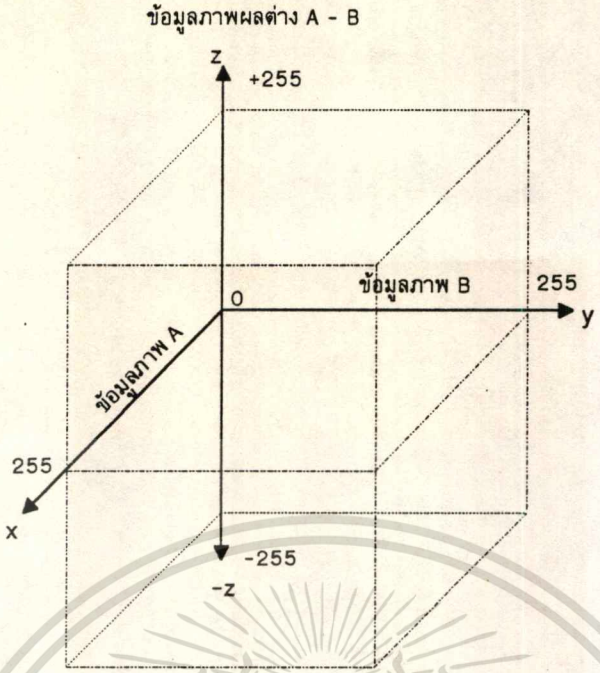
รูปที่ 5.3 แสดงการกำหนดขนาดของสี่เหลี่ยมลูกบาศก์สำหรับการพล็อตฮีสโตแกรมสามมิติ

5.3.2 การพล็อตค่าของฮีสโตแกรมสามมิติจากข้อมูลภาพสองภาพ

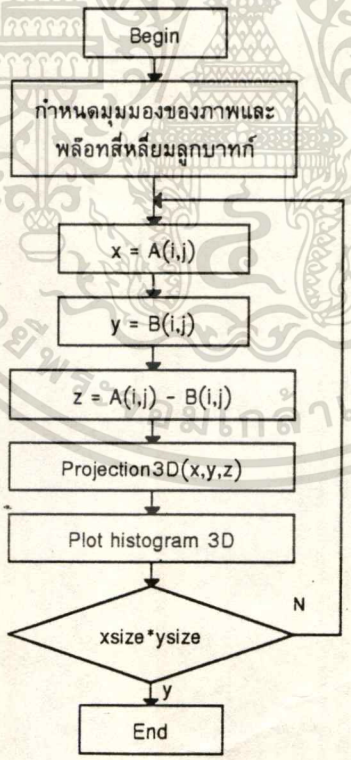
จากสี่เหลี่ยมลูกบาศก์ที่ได้ทำการสร้างขึ้นจะกำหนดให้แต่ละแกนของลูกบาศก์สามมิติแทนค่าฮีสโตแกรมจากข้อมูลภาพถ่ายดาวเทียมทั้งสองได้ดังนี้

1. ฮีสโตแกรมของภาพแรกจะถูกพล็อทค่าลงบนแกน x
2. ฮีสโตแกรมของภาพที่สองแรกจะถูกพล็อตค่าลงบนแกน y
3. ฮีสโตแกรมของภาพผลต่างระหว่างภาพแรกกับภาพที่สองจะถูกพล็อทค่าลงบนแกน z

จากข้อกำหนดดังกล่าวสามารถแสดงได้ดังรูปที่ 5.4

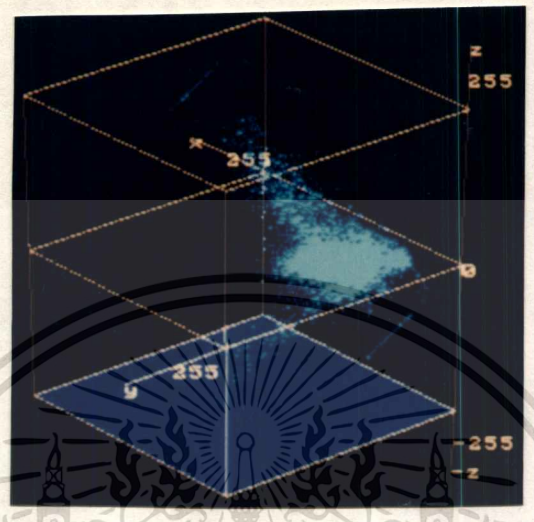


รูปที่ 5.4 การกำหนดให้แต่ละแกนของลูกบาศก์สามมิติแทนข้อมูลฮีสโตแกรมในภาพทั้งสอง



รูปที่ 5.5 แผนผังแสดงขั้นตอนการพล็อตฮีสโตแกรมสามมิติจากข้อมูลภาพทั้งสอง

ลำดับขั้นตอนในการพล๊อตฮีสโตแกรมสามมิติจากข้อมูลภาพถ่ายดาวเทียมทั้งสองแสดงได้ดังแผนผังในรูปที่ 5.5 ดังนั้นในรูปที่ 5.6 แสดงการกระจายของข้อมูลในฮีสโตแกรมสามมิติจากข้อมูลของภาพถ่ายดาวเทียมในรูปที่ 5.2(a) กับ รูปที่ 5.2(b) โดยที่ทางด้านแกน x จะแทนข้อมูลฮีสโตแกรมในรูปที่ 5.2(a) และทางแกน y แทนข้อมูลฮีสโตแกรมในรูปที่ 5.2(b) ส่วนทางแกน z แทนค่าผลต่างระหว่างรูปทั้งสองตามลำดับ



รูปที่ 5.6 แสดงการกระจายของข้อมูลปรากฏบนฮีสโตแกรมสามมิติของภาพทั้งสอง

5.4 คุณสมบัติของเมฆและเงาเมฆที่ปรากฏบนฮีสโตแกรมสามมิติ

ผลจากการพล๊อตฮีสโตแกรมสามมิติของข้อมูลภาพถ่ายดาวเทียมจำนวนสองภาพที่ต้องการตรวจสอบหรือกำจัดเมฆและเงาเมฆ ซึ่งคุณสมบัติของจุดภาพที่เป็นเมฆและเงาเมฆที่ปรากฏบนฮีสโตแกรมสามมิติมีข้อสมมติฐานดังนี้

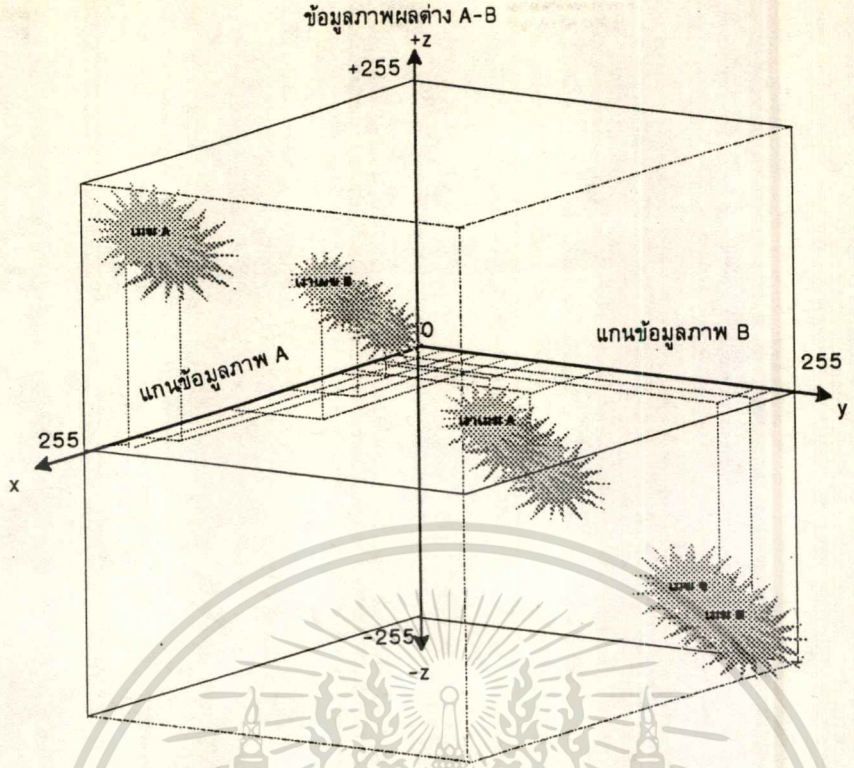
5.4.1. จุดภาพที่เป็นข้อมูลเมฆในภาพแรกจะมีค่าสูงในแกน x และมีค่าผลต่างสูงในทางบวกบนแกน z

5.4.2. จุดภาพที่เป็นข้อมูลเงาเมฆในภาพแรกจะมีค่าต่ำในแกน x และมีค่าผลต่างต่ำในทางลบบนแกน z

5.4.3. จุดภาพที่เป็นข้อมูลเมฆในภาพที่สองจะมีค่าสูงในแกน y และมีค่าผลต่างสูงในทางลบบนแกน z

5.4.4. จุดภาพที่เป็นข้อมูลเงาเมฆในภาพที่สองจะมีค่าต่ำในแกน y และมีค่าผลต่างต่ำในทางบวกบนแกน z

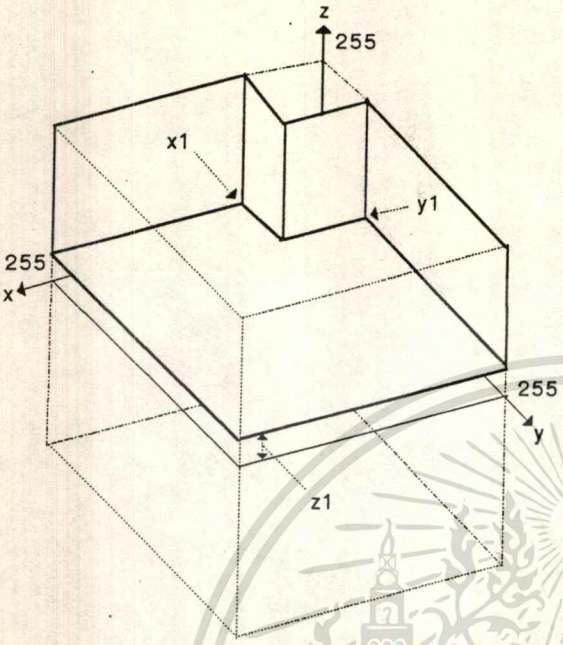
จากข้อสมมติฐานทั้ง 4 สามารถแสดงตำแหน่งการกระจายของฮีสโตแกรมจากข้อมูลภาพทั้งสองที่ปรากฏภายใต้ลูกบาศก์สี่หลี่ยมได้ดังรูปที่ 5.7



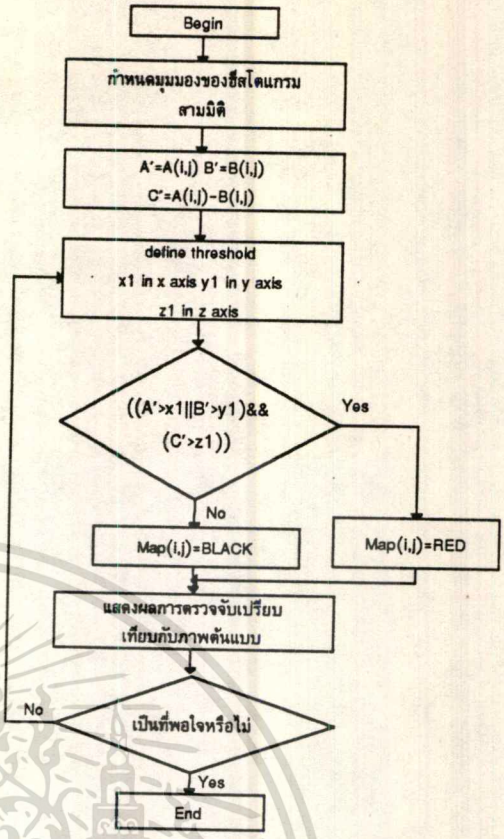
รูปที่ 5.7 คุณสมบัติของข้อมูลเมฆและเงาเมฆของภาพทั้งสองที่ปรากฏบนฮีสโตแกรมสามมิติ

5.5 การตรวจจับจุดภาพที่เป็นเมฆและเงาเมฆ (Cloud and shadow detection)

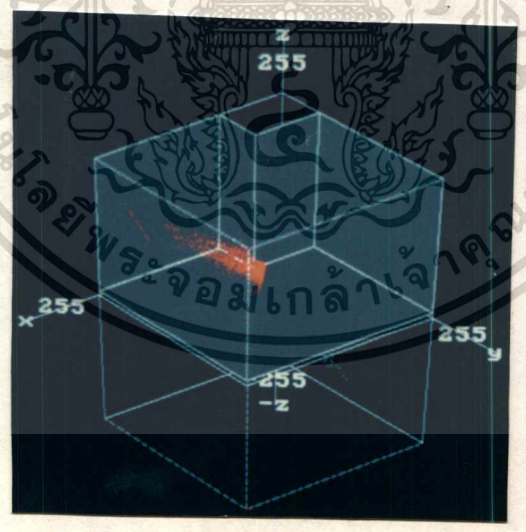
จากข้อสมมติฐานทั้งสี่ข้อดังกล่าว ทำให้สามารถที่จะกำหนดขอบเขตของค่าเธร็สโฮลด์ไว้ในแต่ละแกนของฮีสโตแกรมสามมิติ เพื่อที่จะทำการตรวจจับจุดภาพที่เป็นเมฆและเงาเมฆของแต่ละภาพได้พร้อมๆกัน เพื่อความสะดวกในการพิจารณาข้อมูลที่ปรากฏบนฮีสโตแกรมสามมิติ เราสามารถที่จะกำหนดหรือหมุนลูกบาศก์สี่เหลี่ยมให้อยู่ในมุมมองตามจุดต่างๆได้ ทำให้การพิจารณากำหนดเธร็สโฮลด์สำหรับการตรวจจับได้อย่างมีประสิทธิภาพและคล่องตัว จาก รูปที่ 5.8(a), 5.9(a), 5.10(a) และ 5.11(a) แสดงการกำหนดเธร็สโฮลด์เป็น plane สำหรับการตรวจจับข้อมูลที่เป็นเมฆและเงาเมฆของทั้งสองภาพที่มุมมองต่างๆกัน พร้อมทั้งแสดงแผนผังลำดับการตรวจจับ และจากการใช้เธร็สโฮลด์ในรูป plane มาตัดแบ่งหรือกำหนดขอบเขตพื้นที่จากฮีสโตแกรมสามมิตินี้ จะพบว่าการกระจายของกลุ่มจุดที่เป็นเมฆและเงาเมฆในภาพแรกแสดงดังรูปที่ 5.8(c) และ 5.9(c) และการกระจายของกลุ่มจุดที่เป็นเมฆและเงาเมฆในภาพที่สอง แสดงได้ดัง รูปที่ 5.10(c) และ 5.11(c)



(a) เธร็สโฮลด์ plane ตรวจจับเมฆ

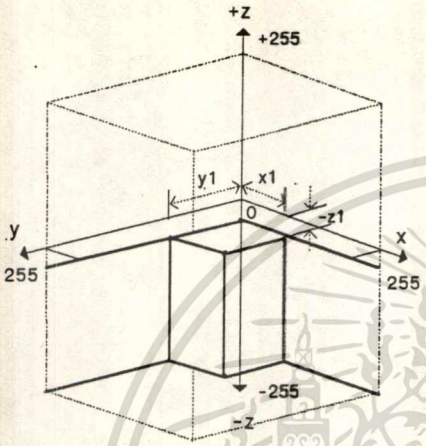


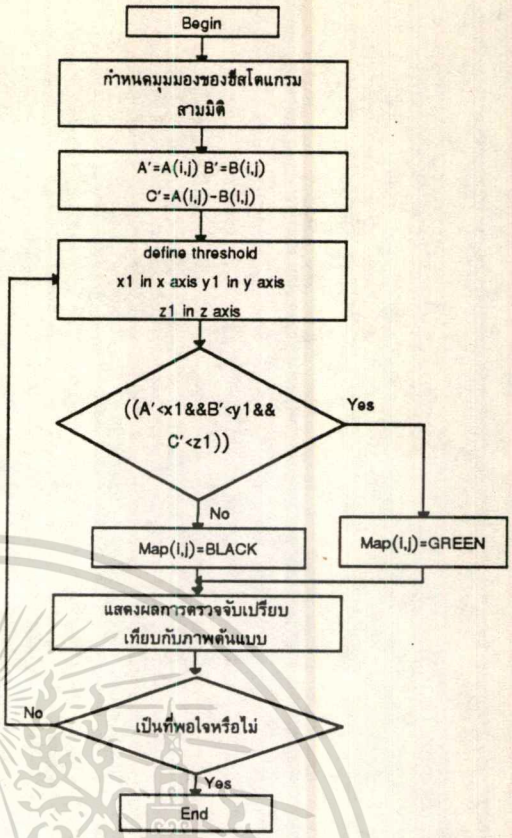
(b) แผนผังลำดับการตรวจจับ



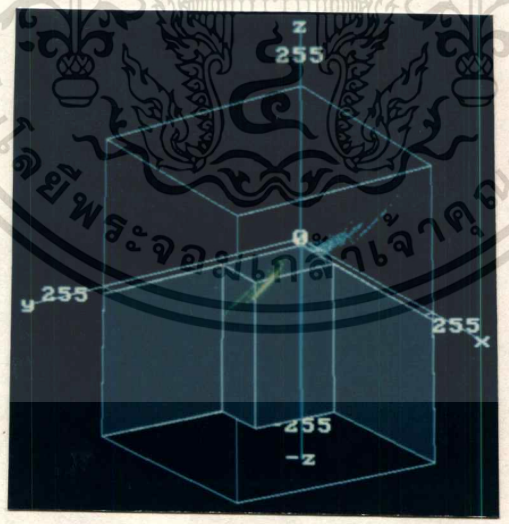
(c) การกระจายของจุดภาพที่เป็นเมฆของภาพแรก

รูปที่ 5.8 การใช้เธร็สโฮลด์ในการแบ่งกลุ่มเมฆและผลที่ปรากฏจากฮีสโตแกรมสามมิติจากภาพแรก



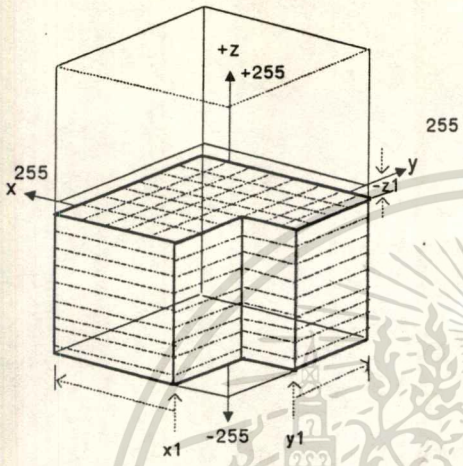


(a) เธร็สโฮลด์ plane ตรวจจับเงาเมฆ (b) แผนผังลำดับการตรวจจับ

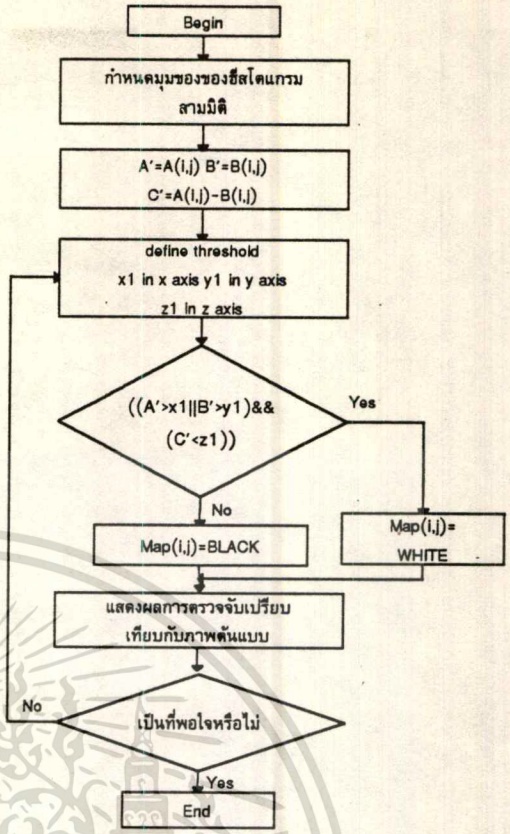


(b) แสดงการกระจายจุดภาพที่เป็นเงาเมฆในภาพแรก

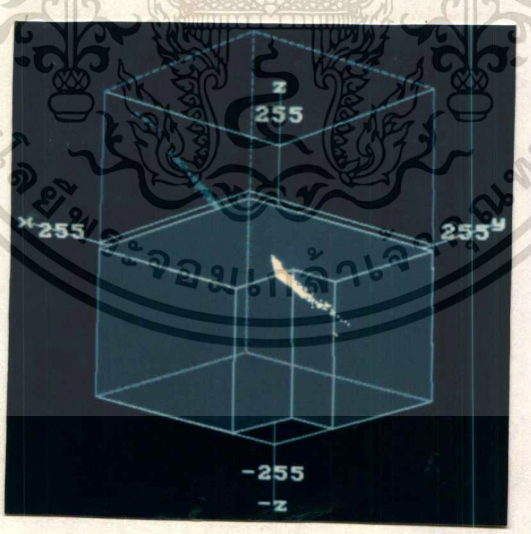
รูปที่ 5.9 การใช้เธร็สโฮลด์ในการแบ่งกลุ่มของเงาเมฆและผลที่ปรากฏจากฮีสโตแกรมสามมิติจากภาพแรก



a) เธร็สโฮลด์ plane ตรวจจับเมฆ

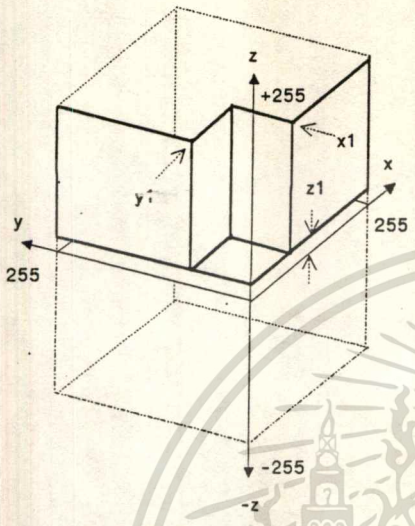


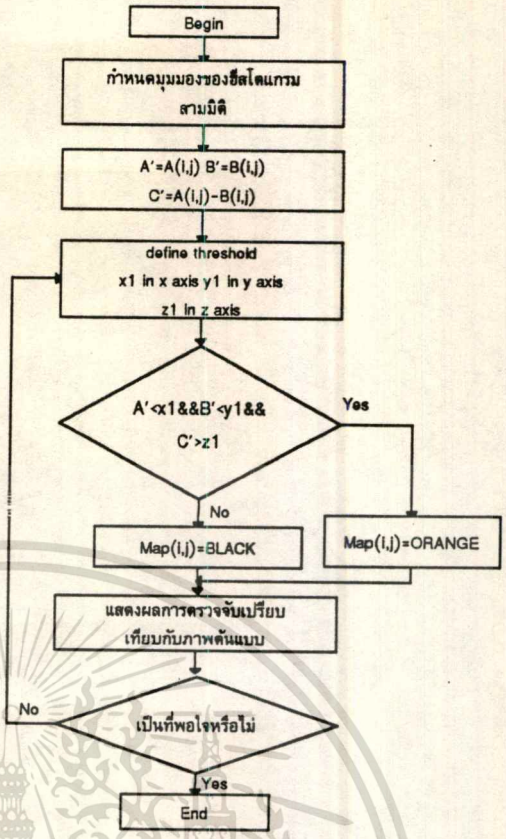
(b) แผนผังลำดับการตรวจจับ



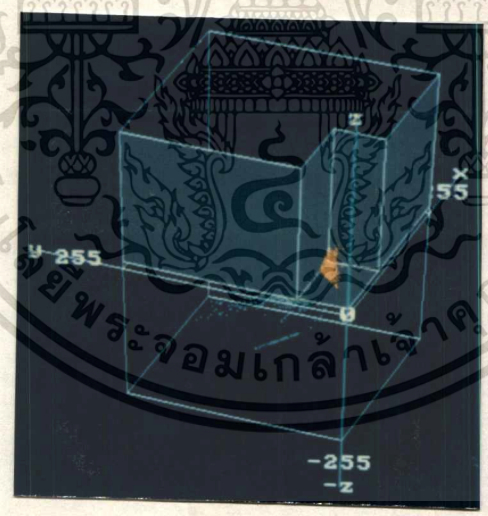
(c) การกระจายของกลุ่มจุดภาพที่เป็นเมฆของภาพที่สอง

รูปที่ 5.10 การใช้เธร็สโฮลด์ในการแบ่งกลุ่มของเมฆและผลที่ปรากฏจากฮีสโตแกรมสามมิติจากภาพที่สอง





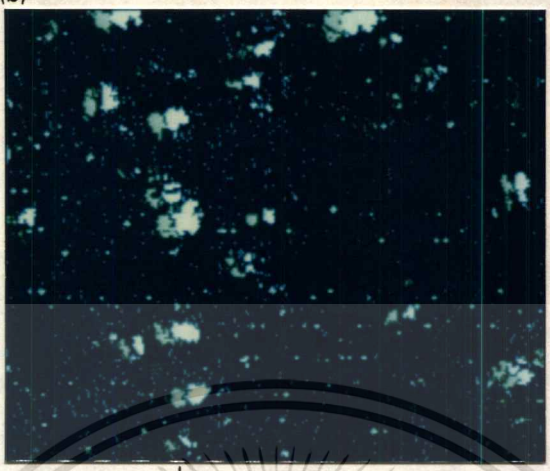
a) เธร็สโฮลด์ plane ตรวจจับเงาเมฆ (b) แผนผังลำดับการตรวจจับ



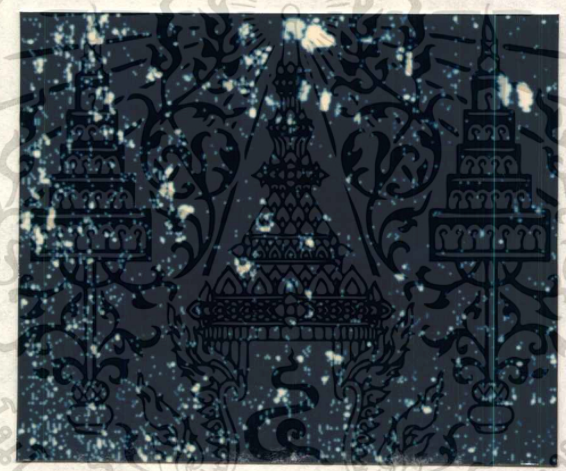
(c) การกระจายของกลุ่มจุดภาพที่เป็นเงาเมฆของภาพที่สอง

รูปที่ 5.11 การใช้เธร็สโฮลด์ในการแบ่งกลุ่มของเงาเมฆและผลที่ปรากฏจากฮีสโตแกรมสามมิติจากภาพที่สอง

จุดภาพที่เป็นเมฆและเงาเมฆจากภาพถ่ายดาวเทียมจำนวนสองภาพที่ผ่านการตรวจจับจากการใช้ฮีสโตแกรมสามมิติ แล้วทำการสร้างภาพเพื่อแสดงตำแหน่งของจุดภาพที่ตรวจสอบได้ ดังรูปที่ 5.9(a) และ 5.9(b)



(a) กลุ่มจุดภาพที่เป็นเมฆและเงาเมฆในภาพแรก



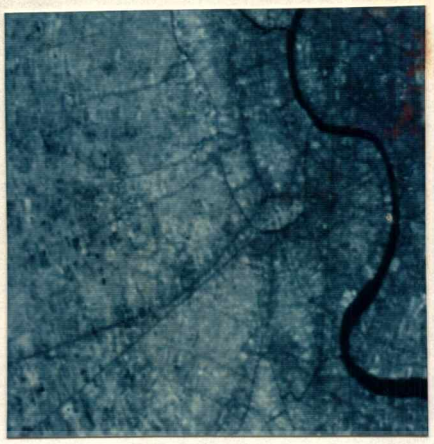
(b) กลุ่มจุดภาพที่เป็นเมฆและเงาเมฆในภาพที่สอง

รูปที่ 5.12 กลุ่มจุดภาพที่เป็นเมฆและเงาเมฆที่ถูกตรวจสอบได้จากฮีสโตแกรมสามมิติ

5.6 การกำจัดกลุ่มเมฆและเงาเมฆ
(Cloud and shadow removing)

ตำแหน่งของกลุ่มจุดที่เป็นมฆและเงาเมฆบนแผนที่ สามารถที่จะถูกกำจัดออกไปได้โดยการแทนค่าของจุดภาพ ณ ตำแหน่งที่สอดคล้องกันกับอีกภาพหนึ่ง ดังนั้นภาพถ่ายดาวเทียมที่มีเมฆและเงาเมฆปรากฏอยู่ สามารถจะถูกกำจัดออกไปได้โดนพร้อมๆกันทั้งสองภาพ

ภาพผลลัพธ์จากการกำจัดเมฆและเงาเมฆของทั้งสองภาพแสดงได้ดังรูปที่ 5.10



(a) ภาพผลลัพธ์ของรูปแรก

(b) ภาพผลลัพธ์ของรูปที่สอง

รูปที่5.13 แสดงภาพผลลัพธ์จากการกำจัดเมฆและเงาเมฆทั้งสองภาพจากการใช้ฮีสโตแกรมสามมิติ

5.7 สรุป

เทคนิคในการใช้ฮีสโตแกรมสามมิติมาช่วยในการตรวจสอบและพิจารณากำหนดเธร็สโฮลด์พบว่ามีข้อดีเหนือกว่าวิธีที่ได้กล่าวมาแล้ว ก็คือ สามารถที่จะกำจัดได้ทั้งเมฆและเงาเมฆ ของทั้งสองภาพได้พร้อมๆกัน ทำให้การใช้งานสะดวกและรวดเร็ว ฮีสโตแกรมสามมิติที่นำเสนอนี้ไม่เพียงแต่จะกำจัดเมฆและเงาเมฆได้เท่านั้น แต่ยังสามารถกำจัดสัญญาณรบกวนแบบ Salt และ Pepper ได้ดีอีกด้วย ดังจะเห็นได้จากการตรวจพบสัญญาณรบกวนดังกล่าวในรูปที่ 5.12

บทที่ 6
บทสรุป

6.1 สรุปผลการวิจัย

การประมวลผลภาพถ่ายดาวเทียมโดยวิธีการกำจัดเมฆและเงาเมฆ (Cloud and shadow removing) มีประโยชน์ต่อการพิจารณาและวิเคราะห์ข้อมูลในพื้นที่ที่สนใจได้เป็นอย่างดี ไม่เพียงแต่จุดภาพที่เป็นเมฆหรือเงาเมฆจะถูกกำจัดออกไปได้แต่ยังสามารถที่จะกำจัดสัญญาณรบกวนประเภท Salt และ Pepper ได้อีกด้วย ส่งผลทำให้ข้อมูลภาพมีความชัดเจนยิ่งขึ้น ซึ่งเหมาะสำหรับการนำพื้นที่เหล่านั้นไปทำแผนที่ทางภูมิศาสตร์ได้สะดวกและถูกต้อง จากวิธีการกำจัดเมฆและเงาเมฆที่ได้กล่าวผ่านมาแล้วนั้น เห็นได้ว่าจะอาศัยข้อมูลของฮีสโตแกรมเป็นเครื่องมือที่สำคัญในการพิจารณากำหนดจุดตัดของเธร็สโฮลด์ จากวิธีการกำจัดเมฆโดยการใช้เธร็สโฮลด์คู่ (Dual threshold for detection cloud) จะมีขีดจำกัดต่อการพิจารณากำหนดค่าเธร็สโฮลด์ ถ้าหากข้อมูลฮีสโตแกรมที่ปรากฏมีย่านไดนามิกด์ที่แคบ ดังนั้นจึงจำเป็นที่จะต้องทำการปรับค่าของจุดภาพให้มีย่านไดนามิกด์ที่กว้างพอ โดยการใช้วิธีดึงค่าจุดภาพอย่างเชิงเส้น (Linear stretching) นอกจากนี้สามารถกำจัดเฉพาะจุดภาพที่เป็นข้อมูลเมฆเท่านั้น ซึ่งต่อมาได้พัฒนาวิธีการให้มีประสิทธิภาพยิ่งขึ้น โดยการใช้ฮีสโตแกรมสองมิติเป็นเครื่องมือช่วยในการพิจารณาและกำหนดกลุ่มเธร็สโฮลด์สำหรับการตรวจจับ วิธีการนี้ สามารถตรวจสอบและกำจัดได้ทั้งจุดภาพที่เป็นเมฆและเงาเมฆ ซึ่งมีประสิทธิภาพและดีกว่าวิธีแรก อย่างไรก็ตามการกำจัดเมฆที่ปกคลุมด้วยฮีสโตแกรมสองมิติสามารถตรวจจับและกำจัดได้เพียงครั้งละภาพเท่านั้น ดังนั้นจึงได้พัฒนาเทคนิคใหม่ให้มีความสะดวกและคล่องตัวยิ่งขึ้น โดยการใช้ฮีสโตแกรมสามมิติทำการกำจัดเมฆและเงาเมฆบนภาพถ่ายดาวเทียมไปพร้อมๆกันทั้งสองภาพ จากวิธีการที่เสนอนี้ สามารถพิจารณารายละเอียดของเมฆและเงาเมฆทั้งหมดที่ปรากฏบนฮีสโตแกรมสามมิติได้ซึ่งสามารถที่จะเปลี่ยนมุมมองของข้อมูลที่ปรากฏในแต่ละด้านของฮีสโตแกรมสามมิติได้สะดวกและคล่องตัว โดยการกำหนดเธร็สโฮลด์เป็น Plane สำหรับการตรวจจับและกำจัดได้อย่างมีประสิทธิภาพ ซึ่งสามารถกำจัดความไม่ต่อเนื่องของระดับสีเทาในบริเวณขอบของเมฆหรือเงาเมฆ รวมทั้งสัญญาณรบกวนประเภท Salt และ Pepper ได้อย่างสมบูรณ์ ส่งผลทำให้ได้ข้อมูลภาพที่มีความเด่นชัดเหมาะให้การนำไปวิเคราะห์และตรวจสอบภูมิประเทศตามพื้นที่ที่สนใจได้เป็นอย่างดี เช่นในการทำแผนที่ทางภูมิศาสตร์

6.2 ปัญหาและข้อเสนอแนะ

วิธีการประมวลผลข้อมูลภาพถ่ายในการกำจัดเมฆและเงาเมฆที่วิจัยพัฒนาขึ้นนี้เป็นขบวนการที่ต้องอาศัยข้อมูลภาพถ่ายดาวเทียมจำนวนสองภาพ ในบริเวณพื้นที่เดียวกัน และต้องมีการกระจายของกลุ่มข้อมูลเมฆหรือเงาเมฆตลอดทั้งสัญญาณรบกวนประเภท Salt และ Pepper ในตำแหน่งพิกัดที่แตกต่างกัน สิ่งสำคัญอย่างหนึ่งก็คือ ข้อมูลทั้งสองภาพจะต้องมีความสมมาตร (Match) กันทางเรขาคณิตดีที่สุด ตลอดทั้งค่าความเข้มของจุดภาพทั้งสองจะต้องมีค่าใกล้เคียงกันมากที่สุดหรือเท่ากัน จึงจะสามารถตรวจจับได้อย่างมีประสิทธิภาพ ทั้งนี้เนื่องจากวิธีในการกำจัดเมฆและเงาเมฆที่ปกคลุมจะอาศัยการเปรียบเทียบค่าความเข้มของจุดภาพต่อจุดภาพในตำแหน่งที่สอดคล้องกันตลอดทั้งภาพ จึงทำให้มีขีดจำกัดต่อขนาดของภาพที่จะนำมาประมวลผล ถ้าหากภาพทั้งสองมีขนาดใหญ่ ทำให้มีปัญหาในด้านความเร็วของการประมวลผลและหน่วยความจำของระบบที่วิ่งภาพใต้การทำงานของดอส(DOS) ดังนั้นวิธีการแก้ปัญหาของหน่วยความจำ (memory) ทำได้โดยการพัฒนาโปรแกรมประยุกต์ให้ทำงานบนวินโดส์ (Windows) อย่างไรก็ดี ในอนาคตอุปกรณ์คอมพิวเตอร์มีการพัฒนาให้การประมวณผลข้อมูลได้เร็วยิ่งขึ้น รวมทั้งหน่วยความจำมีแนวโน้มที่ราคาต่อหน่วยลดลง จะส่งผลทำให้การประมวลผลมีความรวดเร็วและสะดวกยิ่งขึ้น

เอกสารอ้างอิง

[1] L. Zhirog and M.J. McDonnel, ***"Technical note : Using dual thresholds for cloud and mosaicking"***. Int J. Remote Sensing, Vol. 7, No. 10, pp.1349-1358, 1986.

[2] F. Cheevasuvit, K. Dejhan, T. Tanapanpanich and D. Lisawadiratanakul, ***"Cloud cover and cloud shadow removing based on 2-dimemsional histogram "***, Proceeding of the 13th Asean Conferance in Remote Sensing , Ulabatar, Mongolia, pp. H-1-4-1 - H-1-4-6, Oct.7-11, 1992.

[3] ***"Manual of Remote Sensing "***, Vol. 1, The American Society of Photogrammetry, Falls Church, va., 1975.

[4] J.A. Parikh and A. Rosenfeld, ***"Automatic segmemtation and classification of infared meteorological satellite data***, IEEE Trans. Syst. May Cybernet., Vol. 8, pp 736,1978.

[5] R.G. Grane and M.R. Anderson, ***"Satellite discrimination of snow/cloud surface"***, Int J. Remote Sensing, Vol.5, pp312, 1984.

[6] Z.K. Liu and B.R. Hunt, ***" Note : A New Approach to Removing Cloud Cover from Satellite Imagery"***,Computer Vision, Graphics, and Image Processing Vol. 25, pp 252-256, 1984.

[7] P. Zamperoni, ***"Contour tracing of grey-scale images based on 2-dimensional histogram"***, Pattern Recognition, Vol. 15, No. 3, pp.161-165, 1982.

[8] A.M. Seddon and G.E. Hunt, ***" Segmentation of clouds using cluster analysis"*** Int. J. Remote Sensing, Vol. 6 , pp 717, 1985.

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

[10] ทนง ธนพันธุ์พาณิชย์ อาโมทย์ สมบูรณ์แก้ว และ ฟูศักดิ์ ชีวสุวิทย์, ***"การกำจัดเมฆและเงาเมฆที่ปกคลุมโดยการใช้ฮีสโตแกรม 2 มิติ"*** การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 15 หน้า 6-5 ถึง 6-6, 3-4 ธันวาคม 2535.

[11] ทนง ธนพันธุ์พาณิชย์ และ ฟูศักดิ์ ชีวสุวิทย์, ***"การกำจัดเมฆและเงาเมฆพร้อม ๆ กันในภาพถ่ายดาวเทียมสองภาพด้วยใช้ฮีสโตแกรม 3 มิติ"*** การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 16 หน้า 612-614, 25-26 พฤศจิกายน 2536.

[12] C.S. Park ***"Interactive Microcomputer Graphics"*** Addison-Wesley Publishing Company pp.133-169, 1985.

ประวัติผู้เขียน

นายทนง ธนพันธุ์พาณิชย์ เกิดวันที่ 29 มกราคม พ.ศ. 2510 ที่อำเภอสะเดา จังหวัดสงขลา จบการศึกษาระดับปริญญาตรีในหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต เกียรตินิยมอันดับสอง สาขาเทคโนโลยีโทรคมนาคม คณะวิศวกรรมศาสตร สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2533 ปัจจุบันทำงานอยู่ที่ บริษัท. ยิบอินซอย จำกัด เป็น SENIOR ENGINEER รับผิดชอบทางด้านการติดตั้งและดูแลระบบ Networks ที่อยู่ภายใต้ระบบ UNIX ทั้ง System 5 Release 4 (SVR4) ,และ SUN Solaris2.3 เป็นเวลา 3 ปี

ภาคผนวก ก.
ผลงานวิจัยที่ได้รับการตีพิมพ์

[1] F. Cheevasuvit, K. Dejhan, T. Tanapanpanich and D. Lisawadiratanakul, ***"Cloudcover and cloud shadow removing based on 2-dimemsional histogram "***, Proceeding of the 13th Asean Conferance in Remote Sensing , pp.H-1-4-1 - H-1-4-6, Ulannbatar, Mongolia, Oct.7-11, 1992.

[2] ทนง ธนพันธุ์พาณิชย์ อาโมทย์ สมบูรณ์แก้ว และ ฟูศักดิ์ ชีวสุวิทย์ ***"การกำจัดเมฆและเงาเมฆที่ปกคลุมโดยการใช้ฮีสโตแกรม 2 มิติ"*** การประชุมวิชาการทางวิศวกรรม ไฟฟ้า ครั้งที่ 15 หน้า 6-5 ถึง 6-6, 3-4 ธันวาคม 2535

[3] ทนง ธนพันธุ์พาณิชย์ และ ฟูศักดิ์ ชีวสุวิทย์ ***"การกำจัดเมฆและเงาเมฆพร้อม ๆ กันในภาพถ่ายดาวเทียมสองภาพด้วยฮีสโตแกรมสามมิติ"*** ประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 16 หน้า 612-614, 25-26 พฤศจิกายน 2537

ภาคผนวก ข.

โปรแกรมการประมวลผล

```
/*-------------------------------------------------------------------*
 * PROGRAM : detect cloud and shadow in 3 dimension histogram program    *
 * IMAGE   : BMP  SEEK                                                   *
 * DATE    : 2537                                                        *
 * DISPLAY : SVGA                                                        *
 *-------------------------------------------------------------------*/
#include  <stdio.h>
#include  <stdlib.h>
#include  <graphics.h>
#include  <alloc.h>
#include  <math.h>
#include  <string.h>
#include  <bios.h>
#include  <mem.h>
#include  <dos.h>
#include  <conio.h>
#include  "n_menu.h"
#include  "bmp_3d.h"
#define   X· x=0;x<xpic;x++
#define   Y  y=0;y<ypic;y++
#define   REMOVE_CUD_A  ((A[x]>=xcudA || B[x]>=ycudA) && (A[x]-B[x]>=zcudA))
#define   REMOVE_SHA_A  ((A[x]<=xshaA && B[x]<=yshaA) && (A[x]-B[x]<=zshaA))
#define   REMOVE_CUD_B  ((A[x]>=xcudB || B[x]>=ycudB) && (A[x]-B[x]<=zcudB))
#define   REMOVE_SHA_B  ((A[x]<=xshaB && B[x]<=yshaB) && (A[x]-B[x]>=zshaB))
#define   PI 0.017460317
int       xcudA,ycudA,zcudA,xshaA,yshaA,zshaA,  xcudB,ycudB,zcudB,xshaB,yshaB,zshaB;
int       XPIX,YPIX,MaxX,MaxY;
int       H1,H2,H3,H4,H5,H6,H7,H8,H9,H10,H11,H12;
int       HZPOSX,HZPOSY,HZNEGX,HZNEGY;
int       XB=50,YB=50,ZB=50, xth,yth,zth;
int       CUDA,SHAA,CUDB,SHAB,type;
int       xcudA,ycudA,zcudA,xshaA,yshaA,zshaA,  xcudB,ycudB,zcudB,xshaB,yshaB,zshaB;
static    char name0[40],name1[40];
#define   FILLCOLOR       151
#define   BGCOLOR         BLACK
#define   COLOR_AXE       65 /*CYAN*/
#define   COLOR_AXE1      65 /*CYAN*/
#define   BOXCOLOR        65 /*CYAN*/
#define   MAP_BG          79
#define   COLOR_SUM       68 /*YELLOW*/
#define   COLOR_GENERAL   127 /*BLUE */
#define   COLOR_CUD_A     100 /*RED*/
#define   COLOR_SHA_A     72  /*GREEN */
#define   COLOR_CUD_B     63  /*WHITE */
#define   COLOR_SHA_B     103 /*BROWN */
#define   DETECT_A        1
#define   DETECT_B        2
#define   REMOVE_A        3
#define   REMOVE_B        4
int       color,maxx,maxy;xpic,ypic,XPIX,YPIX,sizeX,sizeY;
float     z_ax[8],zee ;SCF,sin1,sin2,cos1,cos2,w,xtheta,zphi;
int       Vd,nv,k1,k2,Vx,Vy,L,M;
float     xe,ye,ze,zs;
char      *thr[] = {"DETECT CL_A","DETECT SHA_A","DETECT CL_B","DETECT SHA_B","DETECT_ALL"," EXIT "};
```

```
void sound_()
{
  sound(500);delay(90);nosound();
}
/*-------------------------------------*/
/*  parameter define for 3_D view port    */
/*-------------------------------------*/
void init_viewpara_(float xtheta,float zphi)
{
   float    x_angle=0,z_angle=0;
   x_angle = xtheta*PI;   z_angle = zphi*PI;
   sin1 = sin(x_angle), sin2 = sin(z_angle);
   cos1 = cos(x_angle), cos2 = cos(z_angle);
}
/*-------------------------------*/
/*   Function for Projection in 3_D*/
/*-------------------------------*/
void projection_(float x,float y,float z,int *xs,int *ys)
{
  float     x_ax,y_ax,z_ax;
  float     xee,yee,xss,yss,zss;
  xee = -x*sin1 + y*cos1;  /*-- convert to eye_cordinate --- */
  yee = -x*cos1*cos2 - y*sin1*cos2 + z*sin2;
  zee = -x*sin2*cos1 - y*sin2*sin1 - z*cos2 + D;
  xss = (Vd/s)*(xee/zee);  /*-- convert to screen coordinate -- */
  yss = (Vd/s)*(yee/zee);
  *xs = SCF*(xss*Vx + L);
  *ys = -yss*Vy +M;
}
/*-------------------------------------*/
/*   draw cude line side 3D  ( 17/1/93 )     */
/*-------------------------------------*/
void cudeside2(offsetx,offsety,color,f_color)
int       offsetx,offsety,color,f_color;
{
  int       i,j,k,xn,yn,point,xpoint,ypoint,vecter[5][2],Sc[8][2];
  float     weight[4],index[4],minindex;
  float     x,y,z;
  float     cude1[8][3]={{0,0, 0}, {0,0, 0.4}, {0,0.4, 0},{0,0.4,0.4},
                         {0.4,0, 0}, {0.4,0, 0.4}, {0.4,0.4, 0},{0.4,0.4,0.4},
                        };                    /* for draw line cude 8 side */
  int       sidecude1[4][5]={{0,2,3,1,0}, {0,1,5,4,0}, /* side plan */
                             {4,6,7,5,4}, {3,2,6,7,3}};

  for(i=0;i<8;i++){
    x = cude1[i][0], y = cude1[i][1], z = cude1[i][2];
          projection_(x,y,z,&xn,&yn);
    Sc[i][0] = xn-offsetx, Sc[i][1] = yn-offsety;   /*-- cereen coordinate --*/
    z_ax[i]=zee;
    }
  /* sort */
 for(i=0;i<4;i++){
   weight[i]=0;
   for(j=0;j<5;j++){
```

```
      x= sidecude1[i][j];
      weight[i] = weight[i] + z_ax[x];
      }
    }
  for(i=0;i<4;i++) index[i]=i;
  for(i=0;i<4;i++){
     minindex = i;
     for(j=i+1;j<4;j++) if(weight[index[minindex]] > weight[index[j]]) minindex=j;
     x=index[i];
     index[i] = index[minindex];
     index[minindex]=x;
     }
   for(i=0;i<4;i++){
     for(j=0;j<5;j++){
       point = sidecude1[index[i]][j];      /*— get side of direction[5][5]—*/
       vecter[j][0] = Sc[point][0];   vecter[j][1] = Sc[point][1];
       }
     setcolor(color);
     for(k=0;k<5;k++){
       xpoint=vecter[k][0]; ypoint=vecter[k][1];
       if(k==0)          moveto( xpoint , ypoint );
       else  lineto(xpoint ,ypoint );
       }
     }
}
/*————————————————————*
 *  drawing histogram 3 dimemsion *
 *————————————————————*/
void drawhis(char name0[], char name1[], float h)
{
int       bytes;
BMPHEAD bmp;
double    a,c,b;
int       xi,yi,colorA,colorB;
FILE      *inf0,*inf1;
static    unsigned char A[800], B[800], C[800];
int       x, y, z, Maxa,Maxb,Maxc,t, org_px, org_py, buf_px, buf_py;
 h=h/255.0;
 sizeBMP(name0);
 inf0 = fopen(name0,"rb");
 inf1 = fopen(name1,"rb");
 if((inf0&&inf1) == NULL)
   {
    printf(" Open file and Alloc error..\n");
    exit(1);
   }
 bytes=XPIX;
 if(bytes & 0x0003) {
          bytes |= 0x0003;
          ++bytes;
 }
 Maxa=Maxb=Maxc=255.0;
 fseek(inf0,1078L,SEEK_SET);
 fseek(inf1,1078L,SEEK_SET);
```

```
for(y=0;y<YPIX;y++)
  {
   memset((unsigned char *)&A,0,bytes);
   memset((unsigned char *)&B,0,bytes);
   fread(A,1,bytes,inf0);
   fread(B,1,bytes,inf1);
   for(x=0;x<XPIX;x++)
        {
         a=(A[x]/(float)Maxa-0.5);
         b=(B[x]/(float)Maxb-0.5);
         c=(float)((int)A[x]-(int)B[x]);
         c=c/(float)Maxc;
         projection_(a,b,c,&xi,&yi);
         switch(type)
         {
         case 5:
        /*all put*/
                  putpixel((int)xi,(int)yi, COLOR_SUM /*GENERAL*/);
         break;
         case 4:
         /*all detect*/
             if(REMOVE_CUD_A)        putpixel((int)xi,(int)yi, COLOR_CUD_A);
             else if(REMOVE_SHA_A)   putpixel((int)xi,(int)yi, COLOR_SHA_A);
             else if(REMOVE_CUD_B)   putpixel((int)xi,(int)yi, COLOR_CUD_B);
             else if(REMOVE_SHA_B)   putpixel((int)xi,(int)yi, COLOR_SHA_B);
             else                    putpixel((int)xi,(int)yi, COLOR_GENERAL);
         break;
         case 0:
         /*detect cud A*/
           if(c>h)
             {
             if(REMOVE_CUD_A)        putpixel((int)xi,(int)yi,  COLOR_CUD_A);
             }

           if(c<h&&getpixel(xi,yi)!=FILLCOLOR&&getpixel(xi,yi)!=BOXCOLOR&&getpixel(xi,yi)!=COLOR_CUD_A)
             {
                  putpixel((int)xi,(int)yi, COLOR_GENERAL);
             }
        break;
         case 1:
         /*detect shadow B*/
           if(c>h)
             {
             if(REMOVE_SHA_B)
                  putpixel((int)xi,(int)yi, COLOR_SHA_B);
             }
           if(c<h&&getpixel(xi,yi)!=FILLCOLOR&&getpixel(xi,yi)!=BOXCOLOR&&getpixel(xi,yi)!=COLOR_SHA_B)
             {
                  putpixel((int)xi,(int)yi, COLOR_GENERAL);
             }
         break;
         case 2:
         /*detect cud B*/
           if(c<h)
```

```
                {
                if(REMOVE_CUD_B)   putpixel((int)xi,(int)yi, COLOR_CUD_B);
                }
              if(c>h&&getpixel(xi,yi)!=FILLCOLOR&&getpixel(xi,yi)!=BOXCOLOR&&getpixel(xi,yi)!=COLOR_CUD_B)
                {
                      putpixel((int)xi,(int)yi, COLOR_GENERAL);
                }
            break;
            case 3:
            /*detect shadow A*/
              if(c<h)
                {
                if(REMOVE_SHA_A)      putpixel((int)xi,(int)yi, COLOR_SHA_A);
                }
             if(c>h&&getpixel(xi,yi)!=FILLCOLOR&&getpixel(xi,yi)!=BOXCOLOR&&getpixel(xi,yi)!=COLOR_SHA_A)
                {
                      putpixel((int)xi,(int)yi, COLOR_GENERAL);
                }
            break;
            }
        }
     }
 fclose(inf0);
 fclose(inf1);
}
void detect_cloud(char name0[], char name1[],int xx,int yy,int type)
{
int         bytes;
BMPHEAD bmp;
int         a,b,c,color;
int         xi,yi,colorA,colorB;
FILE        *inf0,*inf1;
static      unsigned char A[800], B[800], C[800];
int         x, y, z, cat, org_px, org_py, buf_px, buf_py;
 sizeBMP(name0);
 inf0 = fopen(name0,"rb");
 inf1 = fopen(name1,"rb");
 if((inf0&&inf1) == NULL)
   {
    printf(" Open file and Alloc error..\n");
    exit(1);
   }
 bytes=XPIX;
 if(bytes & 0x0003) {
            bytes |= 0x0003;
            ++bytes;
   }
 fseek(inf0,1078L,SEEK_SET);
 fseek(inf1,1078L,SEEK_SET);
 for(y=0;y<YPIX;y++)
    {
     memset((unsigned char *)&A,0,bytes);
     memset((unsigned char *)&B,0,bytes);
     fread(A,1,bytes,inf0);
```

```
fread(B,1,bytes,inf1);
    for(x=0;x<XPIX;x++)
     {
     switch(type){
     case 1 :
       if(REMOVE_CUD_A)                color = COLOR_CUD_A;
              else if(REMOVE_SHA_A)   color =  COLOR_SHA_A;
              else color =  BLACK;
               break;
     case 2 :
       if(REMOVE_CUD_B)   color =  COLOR_CUD_B;
              else if(REMOVE_SHA_B)   color =  COLOR_SHA_B;
               else color =  BLACK;
              break ;
     case 3 :
       if(REMOVE_CUD_A)  color = (B[x]>>2);
       else if(REMOVE_SHA_A) color = (B[x]>>2);
       else color = (A[x]>>2);
      break;
     case 4 :
       if(REMOVE_CUD_B) color = (A[x]>>2);
       else if (REMOVE_SHA_B) color = (A[x]>>2);
       else color = (B[x]>>2);
       break;
     case 5 :  /* detect && remove cloud A */
    if(REMOVE_CUD_A)  color = (B[x]>>2);
       else color = (A[x]>>2);
       break;
     case 6 :  /* detect && remove shadow A */
       if(REMOVE_SHA_A) color = (B[x]>>2);
       else color = (A[x]>>2);
       break;
     case 7 :  /* detcet && remove cloud B */
       if(REMOVE_CUD_B)  color = (A[x]>>2);
       else color = (B[x]>>2);
       break;
     case 8 :  /* detect && remove shadow B */
              if(REMOVE_SHA_B) color = (A[x]>>2);
        else color = (B[x]>>2);
        break;
     case 9 :  /* Create MAP cloud A */
        if(REMOVE_CUD_A)       color = COLOR_CUD_A;
        else color = MAP_BG;
        break;
     case 10 : /* Create MAP shadow A */
        if(REMOVE_SHA_A)   color =  COLOR_SHA_A;
        else color =  MAP_BG;
        break;
     case 11 : /* Create MAP cloud B */
       if(REMOVE_CUD_B)   color =  COLOR_CUD_B;
       else color = MAP_BG;
       break;
     case 12 :  /* Create MAP shadow B */
       if(REMOVE_SHA_B)   color =  COLOR_SHA_B;
```

```
                else color =  MAP_BG;
                break ;
              }
              putpixel(MaxX/2-XPIX/2+x+xx,   MaxY/2-YPIX/2+y+yy,   color);
            }
    }
   fclose(inf0);
   fclose(inf1);
}
void putline(x0,y0,x1,y1,color)
int x0,y0,x1,y1,color;
{
register int x,y;
 /*case1*/
 if(x0==x1&&y0==y1){
          putpixel(x0,y0,color);
 }
 else
 /*case2*/
 if(abs(x1-x0)>=abs(y1-y0)){
          if(x1<x0) { x=x1; y=y1; x1=x0; y1=y0; x0=x; y0=y;}
          for(x=x0;x<=x1;x++){
                y=(y0+((x-x0)*(long)(y1-y0))/(float)(x1-x0))+0.5;
                putpixel(x,y,color);
          }
 }
 /*case3*/
 else {
          if(y1<y0) { x=x1; y=y1; x1=x0; y1=y0; x0=x; y0=y; }
          for(y=y0;y<=y1;y++){
                x=(x0+((y-y0)*(long)(x1-x0))/(float)(y1-y0))+0.5;
                putpixel(x,y,color);
          }
  }
}
void drawcube_()  /* modify use for projection */
{
int x1,x2,x3,x4,y1,y2,y3,y4;
int s1,s2,s3,s4,t1,t2,t3,t4,bx1,bx2,by1,by2;
 setfillstyle(1,11/*3*/);
 setcolor(4);
 setlinestyle(0,1,1);
 /*ZERO   x    y    z*/
 projection_(-0.5, -0.5, 0.0, &s1, &t1);
 projection_(-0.5, 0.5, 0.0, &s2, &t2);
 projection_( 0.5, 0.5, 0.0, &s3, &t3);
 projection_( 0.5, -0.5, 0.0, &s4, &t4);
 putline(s1,t1,s2,t2,COLOR_AXE);
 putline(s3,t3,s2,t2,COLOR_AXE1);
 putline(s3,t3,s4,t4,COLOR_AXE1);
 putline(s1,t1,s4,t4,COLOR_AXE);
 projection_(-0.5, 0.9, 0.0, &bx1, &by1); /*add y axis*/
 putline(s2,t2,bx1,by1,COLOR_AXE);
 setcolor(COLOR_AXE);
```

```
outtextxy((s2+bx1)/2,(t2+by1)/2,"255");
outtextxy(bx1,by1,"y");
projection_( 0.9,-0.5, 0.0, &bx1, &by1); /*add x axis*/
putline(s4,t4,bx1,by1,COLOR_AXE);
setcolor(COLOR_AXE);
outtextxy(bx1,by1,"x");
outtextxy((bx1+s4)/2,(by1+t4)/2,"255");
/*LABEL ZERO*/
setcolor(COLOR_AXE);
outtextxy(s1,t1,"0");
/*LOW  x   y    z*/
projection_(-0.5, -0.5, -1.0, &x1, &y1);
projection_(-0.5,  0.5, -1.0, &x2, &y2);
projection_( 0.5,  0.5, -1.0, &x3, &y3);
projection_( 0.5, -0.5, -1.0, &x4, &y4);
putline(x1,y1,x2,y2,COLOR_AXE1);
putline(x3,y3,x2,y2,COLOR_AXE1);
putline(x3,y3,x4,y4,COLOR_AXE1);
putline(x1,y1,x4,y4,COLOR_AXE1);
/*TOP  x   y   z*/
projection_(-0.5, -0.5, 1.0, &s1, &t1);
projection_(-0.5,  0.5, 1.0, &s2, &t2);
projection_( 0.5,  0.5, 1.0, &s3, &t3);
projection_( 0.5, -0.5, 1.0, &s4, &t4);
putline(s1,t1,s2,t2,COLOR_AXE1);
putline(s3,t3,s2,t2,COLOR_AXE1);
putline(s3,t3,s4,t4,COLOR_AXE1);
putline(s1,t1,s4,t4,COLOR_AXE1);
/*CONNECT LOW & TOP*/
projection_(-0.5, -0.5, 1.4, &bx1, &by1);
projection_(-0.5, -0.5, -1.4, &bx2, &by2);
HZPOSX=bx1;
HZPOSY=by1;
HZNEGX=bx2;
HZNEGY=by2;
putline(bx1,by1,bx2,by2,COLOR_AXE);
setcolor(COLOR_AXE);
outtextxy(bx1,by1,"z");
outtextxy((bx1+s1)/2,(by1+t1)/2,"255");
outtextxy(bx2,by2,"-z");
outtextxy((bx2+x1)/2,(by2+y1)/2,"-255");
putline(x2,y2,s2,t2,COLOR_AXE1);
putline(x3,y3,s3,t3,COLOR_AXE1);
putline(x4,y4,s4,t4,COLOR_AXE1);
/*special for hidden*/
/*putplane(0.0);*/
H1=s2; H2=t2;
H3=s3; H4=t3;
H5=s4; H6=t4;
H7=x3; H8=y3;
H9=x4; H10=y4;
H11=x2;H12=y2;
setlinestyle(2,1,1);
line(H3,H4,H1,H2);
```

```
 line(H3,H4,H5,H6);
 line(H7,H8,H3,H4);
 line(H7,H8,H11,H12);
 line(H7,H8,H9,H10);
 setlinestyle(0,1,1);
}
void plane(x1,x2,x3,x4, y1,y2,y3,y4, style)
int x1,x2,x3,x4, y1,y2,y3,y4,style;
{
 static buf[10];
 setfillstyle(style,FILLCOLOR);
 setcolor(BOXCOLOR);
 setlinestyle(0,1,1);
 buf[0]=x1; buf[1]=y1;
 buf[2]=x2; buf[3]=y2;
 buf[4]=x3; buf[5]=y3;
 buf[6]=x4; buf[7]=y4;
 buf[8]=x1; buf[9]=y1;
 fillpoly(5, buf);
}
void putplane(z)
float z;
{
int x1,x2,x3,x4,y1,y2,y3,y4;
 setfillstyle(1,MAGENTA);
 setcolor(MAGENTA);
 setlinestyle(0,1,1);
 z=z/255.0;
 projection_(-0.5, -0.5, z, &x1, &y1);
 projection_(-0.5,  0.5, z, &x2, &y2);
 projection_( 0.5,  0.5, z, &x3, &y3);
 projection_( 0.5, -0.5, z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
}
void putplane_(z)
float z;
{
int         x1,x2,x3,x4,y1,y2,y3,y4;
 setfillstyle(1,MAGENTA);
 setcolor(MAGENTA);
 setlinestyle(0,1,1);
 z=z/255.0;
 projection_(-0.5 ,-0.5 ,z ,&x1, &y1);
 projection_(-0.5 , 0.5 ,z ,&x2, &y2);
 projection_( 0.5 , 0.5 ,z ,&x3, &y3);
 projection_( 0.5 ,-0.5 ,z ,&x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
}
void putAcudarea(x,y,z)
float       x,y,z;
{
int         x1,x2,x3,x4,y1,y2,y3,y4;
 setfillstyle(1,11/*3*/);
 setcolor(4);
```

```
 setlinestyle(0,1,1);
 x=x/255.0 - 0.5;
 y=y/255.0 - 0.5;
 z=z/255.0;
 projection_( 0.5, -0.5, 1.0, &x1, &y1);
 projection_(   x, -0.5, 1.0, &x2, &y2);
 projection_(   x, -0.5,   z, &x3, &y3);
 projection_( 0.5, -0.5,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(   x,    y, 1.0, &x1, &y1);
 projection_(   x, -0.5, 1.0, &x2, &y2);
 projection_(   x, -0.5,   z, &x3, &y3);
 projection_(   x,    y,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(-0.5,    y, 1.0, &x1, &y1);
 projection_(   x,    y, 1.0, &x2, &y2);
 projection_(   x,    y,   z, &x3, &y3);
 projection_(-0.5,    y,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(-0.5,  0.5, 1.0, &x1, &y1);
 projection_(-0.5,    y, 1.0, &x2, &y2);
 projection_(-0.5,    y,   z, &x3, &y3);
 projection_(-0.5,  0.5,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
}
void putAshaarea(x,y,z)
float       x,y,z;
{
int x1,x2,x3,x4,y1,y2,y3,y4;
 setfillstyle(1,11/*3*/);
 setcolor(4);
 setlinestyle(0,1,1);
 x=x/255.0 - 0.5;
 y=y/255.0 - 0.5;
 z=z/255.0;.
 projection_( 0.5, -0.5,-1.0, &x1, &y1);
 projection_(   x, -0.5,-1.0, &x2, &y2);
 projection_(   x, -0.5,   z, &x3, &y3);
 projection_( 0.5, -0.5,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(   x,    y,-1.0, &x1, &y1);
 projection_(   x, -0.5,-1.0, &x2, &y2);
 projection_(   x, -0.5,   z, &x3, &y3);
 projection_(   x,    y,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(-0.5,    y,-1.0, &x1, &y1);
 projection_(   x,    y,-1.0, &x2, &y2);
 projection_(   x,    y,   z, &x3, &y3);
 projection_(-0.5,    y,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(-0.5,  0.5,-1.0, &x1, &y1);
 projection_(-0.5,    y,-1.0, &x2, &y2);
 projection_(-0.5,    y,   z, &x3, &y3);
 projection_(-0.5,  0.5,   z, &x4, &y4);
```

```
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
}
void putBcudarea(x,y,z)
float       x,y,z;
{
int x1,x2,x3,x4,y1,y2,y3,y4;
 setfillstyle(1,11/*3*/);
 setcolor(4);
 setlinestyle(0,1,1);
 x=x/255.0 - 0.5;
 y=y/255.0 - 0.5;
 z=z/255.0;
 projection_( 0.5, -0.5,-1.0, &x1, &y1);
 projection_(   x, -0.5,-1.0, &x2, &y2);
 projection_(   x, -0.5,   z, &x3, &y3);
 projection_( 0.5, -0.5,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(   x,    y,-1.0, &x1, &y1);
 projection_(   x, -0.5,-1.0, &x2, &y2);
 projection_(   x, -0.5,   z, &x3, &y3);
 projection_(   x,    y,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(-0.5,    y,-1.0, &x1, &y1);
 projection_(   x,    y,-1.0, &x2, &y2);
 projection_(   x,    y,   z, &x3, &y3);
 projection_(-0.5,    y,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(-0.5,  0.5,-1.0, &x1, &y1);
 projection_(-0.5,    y,-1.0, &x2, &y2);
 projection_(-0.5,    y,   z, &x3, &y3);
 projection_(-0.5,  0.5,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
}
void putBshaarea(x,y,z)
float       x,y,z;
{
int         x1,x2,x3,x4,y1,y2,y3,y4;
 setfillstyle(1,11/*3*/);
 setcolor(BOXCOLOR);
 setlinestyle(0,1,1);
 x=x/255.0 - 0.5;
 y=y/255.0 - 0.5;
 z=z/255.0;
 projection_( 0.5, -0.5, 1.0, &x1, &y1);
 projection_(   x, -0.5, 1.0, &x2, &y2);
 projection_(   x, -0.5,   z, &x3, &y3);
 projection_( 0.5, -0.5,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(   x,    y, 1.0, &x1, &y1);
 projection_(   x, -0.5, 1.0, &x2, &y2);
 projection_(   x, -0.5,   z, &x3, &y3);
 projection_(   x,    y,   z, &x4, &y4);
 plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 projection_(-0.5,    y, 1.0, &x1, &y1);
```

```
  projection_(  x,    y, 1.0, &x2, &y2);
  projection_(  x,    y,   z, &x3, &y3);
  projection_(-0.5,   y,   z, &x4, &y4);
  plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
  projection_(-0.5,  0.5, 1.0, &x1, &y1);
  projection_(-0.5,   y, 1.0, &x2, &y2);
  projection_(-0.5,   y,   z, &x3, &y3);
  projection_(-0.5,  0.5,  z, &x4, &y4);
  plane(x1,x2,x3,x4,y1,y2,y3,y4, 1);
 }
  check_key(x1,y1,sizeX,sizeY,gy,gx,page)
 int        *x1,*y1,*page,sizeX,sizeY,gy,gx;
 {
   int      key,yy1,xx1,oldX1,oldY1,oldpage,tmp_p;
   xx1 = *x1;  yy1 = *y1;
   oldpage = tmp_p = *page;
   oldX1=xx1;oldY1=yy1;
   while((key=getkey()) != ENTER){
     if(key==DOWN){
           yy1 += sizeY+gy;   tmp_p++;
           if(tmp_p == 3) { xx1+=sizeX; yy1 = maxy/4*2+90;}
           if(tmp_p>5) { tmp_p=0, /*yy1=*y1,*/ yy1 = maxy/4*2+90;
                         xx1 = sizeX*4+10;/* xx1=*x1;*/}
           }
     if(key==UP){
           yy1 -= sizeY+gy;    tmp_p--;
           if(tmp_p<0) { xx1 += sizeX; yy1 = maxy/4*2+90+((sizeY+gy)*2) ;
                         tmp_p=5;}
           if(tmp_p>5) { xx1 = sizeX*4+10 ; yy1 = maxy/4*2+90; tmp_p=0; }
           if(tmp_p == 2) { xx1 = *x1; yy1 = yy1 = maxy/4*2+90+(sizeY+gy)*2;}
           }
   keyfunc(oldX1,oldY1,oldX1+sizeX-gx,
           oldY1+sizeY,50,10,60,thr[oldpage],75);
   keyfunc(xx1,yy1,xx1+sizeX-gx,yy1+sizeY,10,50,40,thr[tmp_p],70);
   oldpage=tmp_p;
   oldX1=xx1;oldY1=yy1;
   }
   sound(2000); delay(30);  nosound();
   *x1 = xx1; *y1 = yy1;
   *page = tmp_p;
 }
 void disp(char name0[], char name1[],int xx,int yy)
 {
 int        bytes;
 BMPHEAD bmp;
 int        xi,yi,color;
 FILE       *inf0,*inf1;
 static     unsigned char A[800], B[800], C[800];
 int        x,y,z,cat,org_px,org_py,buf_px,buf_py;
  sizeBMP(name0);
  inf0 = fopen(name0,"rb");
  inf1 = fopen(name1,"rb");
    {
     printf(" Open file and Alloc error..\n");
```

```
    exit(1);
   }
  bytes=XPIX;
  if(bytes & 0x0003) {
          bytes |= 0x0003;
          ++bytes;
  }
  fseek(inf0,1078L,SEEK_SET);
  fseek(inf1,1078L,SEEK_SET);
  for(y=0;y<YPIX;y++)
    {
     memset((unsigned char *)&A,0,bytes);
     fread(A,1,bytes,inf0);
     for(x=0;x<XPIX;x++)
     {
          color=(A[x]>>2);
          putpixel(MaxX/2-XPIX/2+x+xx,  MaxY/2-YPIX/2+y+yy,  color);
     }
    }
  getch();

  for(y=0;y<YPIX;y++)
    {
     memset((unsigned char *)&B,0,bytes);
     fread(B,1,bytes,inf1);
     for(x=0;x<XPIX;x++)
     {
          color=(B[x]>>2);
          putpixel(MaxX/2-XPIX/2+x+xx,  MaxY/2-YPIX/2+y+yy,  color);
     }
    }
  fclose(inf0);
  fclose(inf1);
  sound(2000);delay(1000);nosound();
}
void main_menu1()
{
   int     x,x1,x2,y1,y2,gx,gy,sizeX,sizeY,index,quit;
   sizeX = maxx/6;   sizeY = 50;   gx=20;gy=20;
   x1=maxx/4*3-100,   y1=maxy/4*2+80;
   box_(sizeX*4,0,maxx,maxy/4*2-102,40,15,80);
   keyfunc(sizeX*4,maxy/4*2-100,maxx,y1-2,60,20,90,"",70);
   box_(0,0,sizeX*4,maxy,60,20,100);
   keyfunc(sizeX*4,y1,maxx,maxy,60,20,80,"",70);
   x1=sizeX*4+10;   y1=y1+10;
   for(x=0;x<6;x++){
          keyfunc(x1,y1,x1+sizeX-gx,y1+sizeY,60,20,50,thr[x],70);
     y1+=sizeY+gy;
     if(x==2){
          x1+=sizeX; y1=maxy/4*2+90;
          }
    }
  x1=sizeX*4+10;  y1=maxy/4*2+80+10;  index=0;
  keyfunc(x1,y1,x1+sizeX-gx,y1+sizeY,20,60,50,thr[index],70);
```

```
    do {
        check_key(&x1,&y1,sizeX,sizeY,gy,gx,&index);
        switch(index) {
            case 0 : thr_cloud_A();         break;
            case 1 : thr_shadow_A(); break;
            case 2 : thr_cloud_B();         break;
            case 3 : thr_shadow_B(); break;
            case 4 : plothis_sum();  break;
            default: closegraph();exit(1);
        };
            keyfunc(x1,y1,x1+sizeX-gx,y1+sizeY,20,60,50,thr[index],70);

    } while(1);
}
 plothis_sum()
{
   int      x1,y1,sizeX,sizeY;
   float    zf;
   sizeX = maxx/6;
   y1=maxy/4*2+80;

   M = Vy = getmaxy()/2.0-100;
   L = Vx = getmaxx()/2.0-150;
   init_viewpara_(140.0,50.0);
   keyfunc(maxx/6*4+10,maxy/4*2-80,maxx-10,y1-10,60,20,10/*90*/,"",70);
   cudeside2(-450,-80,60,70);
   clsgr(15,15,sizeX*4-15,maxy-15,BLACK);  type=5;
   putplane_(-255.0);  drawcube_();
   drawhis(name0,name1,zf); sound_(); getch();
   clsgr(15,15,sizeX*4-15,maxy-15,BLACK);  type=4;
   putplane_(-255.0);  drawcube_();
   drawhis(name0,name1,zf); sound_();getch();
   disp(name0,name1,-maxx/6,maxy/4);sound_();  getch();
   detect_cloud(name0,name1,-maxx/6,maxy/4,DETECT_A);
   outtextxy(maxx/6,maxy-40,"MAP CLOUD AND SHADOW OF IMAGE A");
   sound_();getch();
   clsgr(maxx/6,maxy-60,maxx/2,maxy-30,BLACK);
   detect_cloud(name0,name1,-maxx/6,maxy/4,DETECT_B);
   outtextxy(maxx/6,maxy-40,"MAP CLOUD AND SHADOW OF IMAGE B");
   sound_();getch();
   clsgr(maxx/6,maxy-60,maxx/2,maxy-30,BLACK);
   detect_cloud(name0,name1,-maxx/6,maxy/4,REMOVE_A);
   outtextxy(maxx/4,maxy-40,"RESULT OF IMAGE A");
   sound_();getch();
   clsgr(maxx/6,maxy-60,maxx/2,maxy-30,BLACK);
   detect_cloud(name0,name1,-maxx/6,maxy/4,REMOVE_B);
   outtextxy(maxx/4,maxy-40,"RESULT OF IMAGE B");
   sound_();getch();
}
int check_(int x,int y,int i,int color)
{
   int key;
   char str[10];
   itoa(i,str,10);outtextxy(x,y,str);
```

```
    while((key=getkey()) != ENTER){
      if(key==DOWN){ i--;   if(i<=0) i=0;   }
      if(key==UP){ i++;  if(i>=255) i=255;  }
      clsgr(x,y,x+30,y+10,color);
      itoa(i,str,10);outtextxy(x,y,str);
      }
            sound_(); return(i);
}
 thr_cloud_A()
{
  int y1;
  float xf,yf,zf;
  sizeX = maxx/6;
  sizeY = maxy/4;
  y1=maxy/4*2+80;
  init_viewpara_(50.0,50.0);     type=0;
  keyfunc(maxx/6*4+10,maxy/4*2-80,maxx-10,y1-10,60,20,10/*90*/,"",70);
  cudeside2(-380,-50,60,70);
  do{
   do {
      keyfunc(sizeX*4+10,10,maxx-10,maxy/4*2-112,40,15,112,"",70);
      setcolor(72);outtextxy(sizeX*4+30,20," SELECT THRESHOLD xcudA ");
      xcudA = check_(sizeX*4+90,40,xcudA,122);
      outtextxy(sizeX*4+30,60," SELECT THRESHOLD ycudA ");
      ycudA = check_(sizeX*4+90,80,ycudA,122);
      outtextxy(sizeX*4+30,100," SELECT THRESHOLD zcudA ");
      zcudA = check_(sizeX*4+90,120,zcudA,122);
      outtextxy(sizeX*4+60,140,"Are you OK.:");
     }
   while(toupper(getche())!='Y');sound_();
            xth=xf=xcudA; yth=yf=ycudA; zth=zf=zcudA;
            clsgr(15,15,sizeX*4-15,maxy-15,BLACK);

            drawcube_();      putplane(zf);
            putAcudarea(xf,yf,zf);
            drawhis(name0,name1,zf);
            setcolor(112/*97CYAN*/);
            line(H3,H4,H1,H2);          line(H3,H4,H5,H6);
            line(H7,H8,H3,H4);          line(H7,H8,H11,H12);
            line(H7,H8,H9,H10);         setcolor(71/*RED*/);
            setlinestyle(2,1,1);
            line(H3,H4,H1,H2);          line(H3,H4,H5,H6);
            line(H7,H8,H3,H4);          line(H7,H8,H11,H12);
            line(H7,H8,H9,H10);          setlinestyle(0,1,1);sound_();getch();
            detect_cloud(name0,name1,-maxx/4,-maxy/4,9); /* map cluod */
            detect_cloud(name0,name1,-maxx/4,maxy/4,5);  /* detect cloud */
            outtextxy(sizeX*4+60,160,"Are you Accept.:");
  }
  while(toupper(getche())!='Y');sound_();
}
 thr_shadow_A()
{
  float xf,yf,zf;
  int y1;
```

```
  y1=maxy/4*2+80;
  sizeX = maxx/6;
  sizeY = maxy/4;
  init_viewpara_(214.0,120.0);
  keyfunc(maxx/6*4+10,maxy/4*2-80,maxx-10,y1-10,60,20,10/*90*/,"",70);
  cudeside2(-380,-50,60,70);
  do{
   do {
       keyfunc(sizeX*4+10,10,maxx-10,maxy/4*2-112,40,15,112,"",70);
       setcolor(72);outtextxy(sizeX*4+30,20," SELECT THRESHOLD xshaA ");
       xshaA = check_(sizeX*4+90,40,xshaA,122);
       outtextxy(sizeX*4+30,60," SELECT THRESHOLD yshaA ");
       yshaA = check_(sizeX*4+90,80,yshaA,122);
       outtextxy(sizeX*4+30,100," SELECT THRESHOLD zshaA ");
       zshaA = check_(sizeX*4+90,120,zshaA,122);
       outtextxy(sizeX*4+60,140,"Are you OK.:");
      }
   while(toupper(getche())!='Y');sound_();
             xth=xf=xshaA; yth=yf=yshaA; zth=zf=zshaA;
             clsgr(15,15,sizeX*4-15,maxy-15,BLACK);

             type=3;    /* detect shadow A*/
          drawcube_();          putplane(zf);
             putAshaarea(xf,yf,zf);
             drawhis(name0,name1,zf);
             setcolor(97 /*CYAN*/);
             putline(HZPOSX,HZPOSY,HZNEGX,HZNEGY,72 /*CYAN*/);
             sound_();getch();
             detect_cloud(name0,name1,-maxx/4,-maxy/4,10); /* map cluod */
             detect_cloud(name0,name1,-maxx/4,maxy/4,6);  /* detect cloud */
             outtextxy(sizeX*4+60,160,"Are you Accept.:");
  }
  while(toupper(getche())!='Y');sound_();
}
 thr_cloud_B()
{
  float     xf,yf,zf;
  int       y1;
  y1=maxy/4*2+80;
  sizeX = maxx/6;
  sizeY = maxy/4;
  init_viewpara_(50.0,115.0);    type=2;
  keyfunc(maxx/6*4+10,maxy/4*2-80,maxx-10,y1-10,60,20,10/*90*/,"",70);
  cudeside2(-380,-80,60,70);
  do{
   do {
       keyfunc(sizeX*4+10,10,maxx-10,maxy/4*2-112,40,15,112,"",70);
       setcolor(72);outtextxy(sizeX*4+30,20," SELECT THRESHOLD xcudB ");
       xcudB = check_(sizeX*4+90,40,xcudB,122);
       outtextxy(sizeX*4+30,60," SELECT THRESHOLD ycudB ");
       ycudB = check_(sizeX*4+90,80,ycudB,122);
       outtextxy(sizeX*4+30,100," SELECT THRESHOLD zcudB ");
       zcudB = check_(sizeX*4+90,120,zcudB,122);
       outtextxy(sizeX*4+60,140,"Are you OK.:");
```

```
    }
  while(toupper(getche())!='Y');sound_();
        xth=xf=xcudB; yth=yf=ycudB; zth=zf=zcudB;
        clsgr(15,15,sizeX*4-15,maxy-15,BLACK);
              drawcube_();          putplane(zf);
        putBcudarea(xf,yf,zf);
        drawhis(name0,name1,zf);
        setcolor(112/*97CYAN*/);
        line(H3,H4,H1,H2);          line(H3,H4,H5,H6);
        line(H7,H8,H3,H4);          line(H7,H8,H11,H12);
        line(H7,H8,H9,H10);         setcolor(71/*RED*/);
        setlinestyle(2,1,1);
        line(H3,H4,H1,H2);          line(H3,H4,H5,H6);
        line(H7,H8,H3,H4);          line(H7,H8,H11,H12);
        line(H7,H8,H9,H10);          setlinestyle(0,1,1);sound_();getch();
        detect_cloud(name0,name1,-maxx/4,-maxy/4,11); /* map cluod */
        detect_cloud(name0,name1,-maxx/4,maxy/4,7);  /* detect cloud */
        outtextxy(sizeX*4+60,160,"Are you Accept.:");
  }
  while(toupper(getche())!='Y');sound_();
}
 thr_shadow_B()
{
  float    xf,yf,zf;
  int      y1;
  y1=maxy/4*2+80;
  sizeX = maxx/6;
  sizeY = maxy/4;
  init_viewpara_(200.0,50.0);
  keyfunc(maxx/6*4+10,maxy/4*2-80,maxx-10,y1-10,60,20,10/*90*/,"",70);
  cudeside2(-380,-80,60,70);
  do{
   do {
      keyfunc(sizeX*4+10,10,maxx-10,maxy/4*2-112,40,15,112,"",70);
      setcolor(72);outtextxy(sizeX*4+30,20," SELECT THRESHOLD xshaB ");
      xshaB = check_(sizeX*4+90,40,xshaB,122);
      outtextxy(sizeX*4+30,60," SELECT THRESHOLD yshaB ");
      yshaB = check_(sizeX*4+90,80,yshaB,122);
      outtextxy(sizeX*4+30,100," SELECT THRESHOLD zshaB ");
      zshaB = check_(sizeX*4+90,120,zshaB,122);
      outtextxy(sizeX*4+60,140,"Are you OK.:");
    }
  while(toupper(getche())!='Y');sound_();
        xth=xf=xshaA; yth=yf=yshaA; zth=zf=zshaA;
        clsgr(15,15,sizeX*4-15,maxy-15,BLACK);
        type=1;    /* detect shadow A*/
      drawcube_();          putplane(zf);
        putBshaarea(xf,yf,zf);
        drawhis(name0,name1,zf);
        setcolor(97 /*CYAN*/);
        putline(HZPOSX,HZPOSY,HZNEGX,HZNEGY,72 /*CYAN*/);
        sound_();getch();
        detect_cloud(name0,name1,-maxx/4,-maxy/4,12); /* map cluod */
        detect_cloud(name0,name1,-maxx/4,maxy/4,8);  /* detect cloud */
```

```
            outtextxy(sizeX*4+60,160,"Are you Accept.:");
  }
  while(toupper(getche())!='Y');sound_();
}
main()
{
  int  x1,y1,sizeX,sizeY,index,old_color,key;
  int x2,y2,x3,y3,x4,y4;
  float zf;
  clrscr();
  printf("\n... HISTOGRAM 3-D CLOUD AND SHADOW DETECTION ...");
  printf("\n... VERSION 1.0 ...\n... FILE IMAGE BMP FORMAT ...");
  printf("\n... TYPE ...D=286x190.... E=128x128... O=OTHER...");
  key=toupper(getch());
  if(key == 'D') {
   printf("\n... IMAGE SIZE 286x190 ...\n... LOADING...IMAGE...WAIT...");
   strcpy(name0,"a.bmp");  strcpy(name1,"b.bmp"); /* 286x190 */
   }
  else if(key == 'E') {
   printf("\n... IMAGE SIZE 128x128 ...\n... LOADING...IMAGE...WAIT...");
   strcpy(name0,"a128.bmp"); strcpy(name1,"b128.bmp");
  }
  else  {
   printf("\nfile A :: "); scanf("%s",&name0);
   printf("file B :: ");   scanf("%s",&name1);
   ReadSize(name0,&xpic,&ypic);
  }
  xcudA=86;    xshaA=60;
  ycudA=60;    yshaA=98;
  zcudA=11;    zshaA=-11;
  xcudB=60;  xshaB=94;
  ycudB=98;  yshaB=56;
  zcudB=-10; zshaB=13;
  opengraph();
  MaxX=maxx=getmaxx();MaxY= maxy = getmaxy();
/*-----------------------*
 *  view port parameter *
 *-----------------------*/
  SCF = (200.0/MaxY)/(270.0/MaxX);
  L = Vx = getmaxx()/2.0-100;
  M = Vy = getmaxy()/2.0-50;
  Vd = 80; s = 10;
  D=18.0;
  init_viewpara_(140.0,50.0);
/*-----------------------*/
  main_menu1();
  closegraph();
}□
```

```
/*-------------------------------------------------------*
 * PROGRAM : 2-Dimemsin histogram cloude and shadow remove*
 * IMAGE   : BMP..seek                                    *
 * DATE    : 5/2537                                       *
 * BY      : Tanong T.                                    *
 *-------------------------------------------------------*/
#include<stdio.h>
#include<stdlib.h>
#include<dos.h>
#include<math.h>
#include<graphics.h>
#include<alloc.h>
#include<bios.h>
#include<conio.h>
#include<string.h>
#include<key.h>
#define  X  x=0;x<XPIX;x++
#define  Y  y=0;y<YPIX;y++
#define  PI 0.017460317
unsigned char huge *A,huge *B,huge *Map,huge *LTB;
int      XPIX=64,YPIX=64,XP,YP,MaxX,MaxY;
int      xcud1,xcud2,ycud1,ycud2,xsha1,xsha2,ysha1,ysha2;
#include "bmp_3d.h"
#define  REMOVE_CLOUD_A ((A[x]>=xcud1 && A[x]<=xcud2)&&(A[x]-B[x]>=ycud1 && A[x]-B[x]<=ycud2))
#define  REMOVE_SHADOW_A ((A[x]>=xsha1 && A[x]<=xsha2)&&(A[x]-B[x]<=ysha1 && A[x]-B[x]>=ysha2))
#define  MAP_BG          79
#define  BLACK_G          0
#define  COLOR_SUM       YELLOW
#define  COLOR_GENERAL   BULE
#define  COLOR_CUD_A     RED
#define  COLOR_SHA_A     GREEN
#define  COLOR_AXE       CYAN
#define  COLOR_AXE1      CYAN
#define  COLOR_HIS       127
#define  COLOR_PLANE     64
#define  SHOW_HIS2D      4
#define  REMOVE_ALL      1
#define  DETECT_ALL      2
#define  DETECT_CUD      3
#define  DETECT_SHA      4
char     Gray50[] = {0xAA,0x55,0xAA,0x55,0xAA,0x55,0xAA,0x55};
char      *thr[] = {"DETECT CL_A","DETECT SHA_A","DETECT_ALL","SHOW_ORIG","SHOW_RESULT"," EXIT "};
long     z,min,max;
int      ymin,ymax,xmin,xmax;
int      x1,x2,x3,x4,y1,y2,y3,y4;
static   char name0[20],name1[20];
int       color,maxx,maxy;
int      xpic,ypic,XPIX,YPIX,sizeX,sizeY;
float    SCF,sin1,sin2,cos1,cos2,w,xtheta,zphi,zee,z_ax[8];depth;
int      Vd,Vx,Vy,L,M;
float    xe,ye,ze,zs;
int huge  detect256(void)
{
  return 4;
  /* 2 =  640 * 480      3 =  800 * 600      4 = 1024 * 768   */
}
void opengraph(void)
{
  int gd,gm,i;
```

```
  installuserdriver("svga256",detect256);
  gd = DETECT;
  initgraph(&gd,&gm,"");
  for(i=0;i<64;i++)
    setrgbpalette(i,i,i,i);
}
void sound_()
{
  sound(500);delay(90);nosound();
}
/*-------------------------------------*/
/*  parameter define for 3_D view port     */
/*-------------------------------------*/
void init_viewpara_(float xtheta,float zphi)
{
   float x_angle=0,z_angle=0;
   x_angle = xtheta*PI;   z_angle = zphi*PI;
   sin1 = sin(x_angle), sin2 = sin(z_angle);
   cos1 = cos(x_angle), cos2 = cos(z_angle);
}
void  clsgr(int x1,int y1,int x2,int y2,int color)
{
  setfillpattern(Gray50,color);
  bar(x1,y1,x2,y2);
}
/*-------------------------------------*/
/*   Function for Projection in 3_D      */
/*-------------------------------------*/
void projection_(float x,float y,float z,int *xs,int *ys,float *ze)
{
  float  x_ax,y_ax,z_ax;
  float  xee,yee,xss,yss,zss;
  xee = -x*sin1 + y*cos1;  /*-- convert to eye_cordinate --- */
  yee = -x*cos1*cos2 - y*sin1*cos2 + z*sin2;
  zee = -x*sin2*cos1 - y*sin2*sin1 - z*cos2 + D;
  xss = (Vd/s)*(xee/zee);  /*-- convert to screen coordinate -- */
  yss = (Vd/s)*(yee/zee);
  *xs =  SCF*(xss*Vx + L);
  *ys =  -yss*Vy +M;
  *ze =  zee;
}
void disp(char name0[], char name1[],int xx,int yy)
{
int *     bytes,xi,yi,color;
BMPHEAD bmp;
FILE      *inf0,*inf1;
static    unsigned char A[800], B[800];
int       x,y,z,cat,org_px,org_py,buf_px,buf_py;
 sizeBMP(name0);
 inf0 = fopen(name0,"rb");
 inf1 = fopen(name1,"rb");
 if((inf0&&inf1) == NULL)
   {
    printf(" Open file and Alloc error..\n");
    exit(1);
   }
 bytes=XPIX;
 if(bytes & 0x0003) {
          bytes |= 0x0003;
```

```
        ++bytes;
 }
fseek(inf0,1078L,SEEK_SET);
fseek(inf1,1078L,SEEK_SET);
for(y=0;y<YPIX;y++)
  {
   memset((unsigned char *)&A,0,bytes);
   fread(A,1,bytes,inf0);
   for(x=0;x<XPIX;x++)
   {
        color=(A[x]>>2);
        putpixel(MaxX/2-XPIX/2+x+xx,   MaxY/8+y-yy,   color);
   }
  }
for(y=0;y<YPIX;y++)
  {
   memset((unsigned char *)&B,0,bytes);
   fread(B,1,bytes,inf1);
   for(x=0;x<XPIX;x++)
   {
        color=(B[x]>>2);
        putpixel(MaxX/2-XPIX/2+x+xx,   MaxY/2+y+yy,   color);
   }
  }
fclose(inf0);
fclose(inf1);
sound(2000);delay(1000);nosound();
}
void putline(x0,y0,x1,y1,color)
int x0,y0,x1,y1,color;
{
register int x,y;
/*case1*/
if(x0==x1&&y0==y1){
        putpixel(x0,y0,color);
}
else
/*case2*/
if(abs(x1-x0)>=abs(y1-y0)){
        if(x1<x0) { x=x1; y=y1; x1=x0; y1=y0; x0=x; y0=y;}
        for(x=x0;x<=x1;x++){
                y=(y0+((x-x0)*(long)(y1-y0))/(float)(x1-x0))+0.5;
                putpixel(x,y,color);
        }
}
/*case3*/
else {
        if(y1<y0) { x=x1; y=y1; x1=x0; y1=y0; x0=x; y0=y; }
        for(y=y0;y<=y1;y++){
                x=(x0+((y-y0)*(long)(x1-x0))/(float)(y1-y0))+0.5;
                putpixel(x,y,color);
        }
 }
}
/*-------------------------------------*
 *   Create histogram 2-dimension      *
 *-------------------------------------*/
void Make_2D(name0,name1,ymin_,ymax_,xmin_,xmax_)
char name0[],name1[];
```

```
int *ymin_,*ymax_,*xmin_,*xmax_;
{
int          bytes,xi,yi,color;
BMPHEAD   bmp;
FILE         *inf0,*inf1;
static       unsigned char A[800], B[800];
int          x,y,cat,org_px,org_py,buf_px,buf_py;
register  int xh,yh;
register  long dog,duck,z;
int          ymin,ymax,xmin,xmax;
char         strbuf[5];
 xmin=ymin=255; xmax=ymax=0;
 sizeBMP(name0);
 inf0 = fopen(name0,"rb");
 inf1 = fopen(name1,"rb");
 if((inf0&&inf1) == NULL)
   {
   printf(" Open file and Alloc error..\n");
    exit(1);
   }
 bytes=XPIX;
 if(bytes & 0x0003) {
          bytes |= 0x0003;
          ++bytes;
}
fseek(inf0,1078L,SEEK_SET);
fseek(inf1,1078L,SEEK_SET);
for(y=0;y<YPIX;y++)
  {
    memset((unsigned char *)&A,0,bytes);
    memset((unsigned char *)&B,0,bytes);
    fread(A,1,bytes,inf0);
    fread(B,1,bytes,inf1);
    for(x=0;x<XPIX;x++)
          {
           xh = A[x];
           yh = A[x]-B[x];
           if(yh<ymin) ymin=yh; else if(yh>ymax) ymax=yh;
           if(xh<xmin) xmin=xh; else if(xh>xmax) xmax=xh;
          }
 }
cat=xmin; itoa(cat,strbuf,10); outtextxy(20,150,strbuf);
cat=ymin; itoa(cat,strbuf,10); outtextxy(20,210,strbuf);
cat=ymax; itoa(cat,strbuf,10); outtextxy(20,240,strbuf);
*ymin_=ymin; *ymax_=ymax; *xmin_=xmin; *xmax_=xmax;
XP=abs(xmax-xmin);
YP=abs(ymax-ymin);
z=XP*(long)YP;                /*protect error*/
if((LTB=farcalloc(z,sizeof(char)))==NULL)          /* allocate LTB 2-dimemsion */
  { closegraph(); printf("\nVar LTB::Alloc Error"); exit(1);}
fseek(inf0,1078L,SEEK_SET);
fseek(inf1,1078L,SEEK_SET);
for(y=0;y<YPIX;y++)
  {
    memset((unsigned char *)&A,0,bytes);
    memset((unsigned char *)&B,0,bytes);
    fread(A,1,bytes,inf0);
    fread(B,1,bytes,inf1);
          for(x=0;x<XPIX;x++)
```

```
            {
            xh=A[x]-xmin;
            yh=A[x]-B[x]-ymin;
            duck=XP*(long)yh+xh;
            if(LTB[duck]<90) LTB[duck]++;
            }
          }
  fclose(inf0);
  fclose(inf1);
  sound(500);delay(1000);nosound();
}
 max_min_LTB(max_,min_)
 long *max_,*min_ ;
 {
  register int x,y,cat;
  register long dog,max,min;
  x=y=0;
  max=min=LTB[0];
  for(y=0;y<YP;y++)
     for(x=1;x<XP;x++)
          {
           dog=XP*(long)y+x;
           cat=LTB[dog];
           if(max<cat) max=cat;
           if(min>cat) min=cat;
          }
 *max_=max; *min_=min;
}
void Detect_pixel(name0, name1, xx, yy,type)
int xx,yy,type;
char name0[],name1[];
{
int         bytes;
BMPHEAD bmp;
int         xi,yi,color,color_map;
FILE        *inf0,*inf1;
static      unsigned char A[800], B[800];
int         x,y,z,cat,org_px,org_py,buf_px,buf_py;
 sizeBMP(name0);
 inf0 = fopen(name0,"rb");
 inf1 = fopen(name1,"rb");
 if((inf0&&inf1) == NULL)
   {
    printf(" Open file and Alloc error..\n");
    exit(1);
   }
 bytes=XPIX;
 if(bytes & 0x0003) {
          bytes |= 0x0003;
          ++bytes;
 }
 fseek(inf0,1078L,SEEK_SET);
 fseek(inf1,1078L,SEEK_SET);
 for(y=0;y<YPIX;y++)
    {
     memset((unsigned char *)&A,0,bytes);
     fread(A,1,bytes,inf0);
     memset((unsigned char *)&B,0,bytes);
     fread(B,1,bytes,inf1);
```

```
    for(x=0;x<XPIX;x++)
    {
    switch(type){
    case 1 :
              if(REMOVE_CLOUD_A)  color = (B[x]>>2);
              else if(REMOVE_SHADOW_A) color = (B[x]>>2);
              else color = (A[x]>>2);
              break;
    case 2 :
              if(REMOVE_CLOUD_A) { color = (B[x]>>2);color_map = COLOR_CUD_A;}
              else if(REMOVE_SHADOW_A) {color = (B[x]>>2);color_map = COLOR_SHA_A;}
              else {color = (A[x]>>2);color_map = COLOR_GENERAL;}
           putpixel(MaxX/2-XPIX/2+x+xx,   MaxY/2+y+yy,   color_map);
              break;
    case 3 :
              if(REMOVE_CLOUD_A) { color_map = COLOR_CUD_A ;
                                   color = (B[x]>>2); }
              else { color_map =  COLOR_GENERAL; color = (A[x]>>2);}
           putpixel(MaxX/2-XPIX/2+x+xx,   MaxY/2+y+yy,   color_map);
              break;
    case 4 :
              if(REMOVE_SHADOW_A){ color_map = COLOR_SHA_A ;
                                   color = (B[x]>>2); }
              else {color_map = COLOR_GENERAL;
                    color = (A[x]>>2); }
              putpixel(MaxX/2-XPIX/2+x+xx,   MaxY/2+y+yy,   color_map);
              break;
          }
          putpixel(MaxX/2-XPIX/2+x+xx,   maxy/8+y-yy,   color);
      }
    }
 fclose(inf0);
 fclose(inf1);
 sound(2000);delay(1000);nosound();
}
/*-------------------------------*
 * Display histogram 2-dimention
 *-------------------------------*/
void Disp_2d(int max,int ymin)
{
  int       x,y,x1,y1,x2,y2,color,zero_,flag=1,line=0;
  long      ling;
  float     xx,yy,zz,depth,depth1,depth2;
  zero_ = abs(ymin);                          /* line zero */
  xx=5/(float)XP-0.5;  yy=5/(float)YP-0.5;
  zz=LTB[100]/(float)max/2-0.5;
  projection_(xx,yy,zz,&x1,&y1,&depth1);
  xx=5/(float)XP-0.5;  yy=6/(float)YP-0.5;
  zz=LTB[100]/(float)max/2-0.5;
  projection_(xx,yy,zz,&x1,&y1,&depth2);
  if(depth1<depth2)flag=0;              /* check depth of view point */
  if(flag){
   for(y=0;y<YP;y++)
     for(x=0;x<XP;x++){
           ling = XP*(long)y+x;
           xx = x/(float)XP-0.5;
           yy = y/(float)YP-0.5;
           zz = LTB[ling]/(float)max/2-0.5;
           if( (LTB[ling]) > 0){color=COLOR_HIS;line=1;}
```

```
            else if(line != 1) color=COLOR_PLANE;
            else {color = COLOR_HIS; line=0;}
            setcolor(color);
            projection_(xx,yy,zz,&x1,&y1,&depth);
            if(y != zero_){
                if(x==0) moveto(x1,y1);
                lineto(x1,y1);
              }
             else{
                setcolor(127);
                lineto(x1,y1);
             }
        }
     }
    else{
     for(y=YP-1;y>0;y--)
        for(x=0;x<XP;x++){
            ling = XP*(long)y+x;
            xx = x/(float)XP-0.5;
            yy = y/(float)YP-0.5;
            zz = LTB[ling]/(float)max/2-0.5;
            if( (LTB[ling]) > 0){color=COLOR_HIS;line=1;}
            else if(line != 1) color=COLOR_PLANE;
            else {color = COLOR_HIS; line=0;}
            setcolor(color);
            projection_(xx,yy,zz,&x1,&y1,&depth);
            if(y != zero_){
                if(x == 0) moveto(x1,y1);
                lineto(x1,y1);
             }
             else{
                setcolor(127);
                lineto(x1,y1);
             }
        }
    }
}
void Disp_2d_color(int max,int xmin,int ymin)
{
  int       x,y,x1,y1,x2,y2,color,zero_flag=1,line=0;
  long      ling;
  float     xx,yy,zz,depth,depth1,depth2;
  zero_ = abs(ymin);                          /* line zero */
  xx=5/(float)XP-0.5;  yy=5/(float)YP-0.5;
  zz=LTB[100]/(float)max/2-0.5;
  projection_(xx,yy,zz,&x1,&y1,&depth1);
  xx=5/(float)XP-0.5;  yy=6/(float)YP-0.5;
  zz=LTB[100]/(float)max/2-0.5;
  projection_(xx,yy,zz,&x1,&y1,&depth2);
  if(depth1<depth2)flag=0;              /* check depth of view point */
  if(flag){
   for(y=0;y<YP;y++)
      for(x=0;x<XP;x++){
            ling = XP*(long)y+x;
            xx = x/(float)XP-0.5;
            yy = y/(float)YP-0.5;
            zz = LTB[ling]/(float)max/2-0.5;
            if( (LTB[ling]) > 0){color=COLOR_HIS;line=1;}
            else if(line != 1){
```

```
            if(x+xmin >= xcud1 && x+xmin <= xcud2 &&
                  y+ymin >= ycud1 && y+ymin <= ycud2)
                        color = COLOR_CUD_A;
            else if(x+xmin >= xsha1 && x+xmin <= xsha2 &&
                     y+ymin <= ysha1 && y+ymin >= ysha2)
                           color = COLOR_SHA_A;
             else color=COLOR_PLANE;
            }
          else {color = COLOR_HIS; line=0;}
          setcolor(color);
          projection_(xx,yy,zz,&x1,&y1,&depth);
          if(y != zero_){
              if(x==0) moveto(x1,y1);
              lineto(x1,y1);
            }
           else{
             setcolor(127);
             lineto(x1,y1);
           }
      }
    }
  else{
   for(y=YP-1;y>0;y--)
     for(x=0;x<XP;x++){
          ling = XP*(long)y+x;
          xx = x/(float)XP-0.5;
          yy = y/(float)YP-0.5;
          zz = LTB[ling]/(float)max/2-0.5;
          if( (LTB[ling]) > 0){color=COLOR_HIS;line=1;}
          else if(line != 1){
            if(x+xmin >= xcud1 && x+xmin <= xcud2 &&
                  y+ymin >= ycud1 && y+ymin <= ycud2)
                        color = COLOR_CUD_A;
            else if(x+xmin >= xsha1 && x+xmin <= xsha2 &&
                     y+ymin <= ysha1 && y+ymin >= ysha2)
                           color = COLOR_SHA_A;
            else color=COLOR_PLANE;
           }
          else {color = COLOR_HIS; line=0;}
          setcolor(color);
          projection_(xx,yy,zz,&x1,&y1,&depth);
          if(y != zero_){
              if(x == 0) moveto(x1,y1);
              lineto(x1,y1);
           }
           else{
             setcolor(127);
             lineto(x1,y1);
           }
      }
    }
}
void draw_axis()
{
 int    x1,x2,x3,x4,y1,y2,y3,y4;
 float  depth;
 sizeX = maxx/6;
 clsgr(15,15,sizeX*4-15,maxy-15,BLACK);
 L = Vx = getmaxx()/2.0-150;
```

```
M = Vy = getmaxy()/2.0-50;
Vd = 80; s = 10;
D=8.0;
setcolor(72);
projection_( -0.5,-0.5,-0.5, &x1, &y1 ,&depth);
projection_( -0.5, 0.5,-0.5, &x2, &y2 ,&depth);
projection_(  0.5, 0.5,-0.5, &x3, &y3 ,&depth);
projection_(  0.5,-0.5,-0.5, &x4, &y4 ,&depth);
putline(x1,y1,x2,y2,COLOR_AXE);
putline(x3,y3,x2,y2,COLOR_AXE1);
putline(x3,y3,x4,y4,COLOR_AXE1);
putline(x1,y1,x4,y4,COLOR_AXE);
projection_( 0.5, -0.55,-0.5, &x1, &y1 ,&depth);
projection_(  -0.55,-0.55,-0.5, &x2, &y2 ,&depth);
projection_(  -0.55, 0.5, -0.5, &x3, &y3 ,&depth);
projection_(  -0.55,-0.55, 0.0, &x4, &y4 ,&depth);
putline(x1,y1,x2,y2,COLOR_AXE);
putline(x2,y2,x3,y3,COLOR_AXE1);
putline(x2,y2,x4,y4,COLOR_AXE1);
projection_(  0.55, -0.55,-0.5, &x1, &y1 ,&depth);
projection_(  -0.55, 0.58,-0.5, &x2, &y2 ,&depth);
projection_(  -0.55,-0.55,0.0, &x3, &y3 ,&depth);
outtextxy(x1,y1," x "); outtextxy(x2,y2," y "); outtextxy(x3,y3," z ");
}
/*-------------------------------------*/
/*   draw cude line side 3D  ( 17/1/93 )     */
/*-------------------------------------*/
void cudeside2(offsetx,offsety,color,f_color)
int offsetx,offsety,color,f_color;
{
  int i,j,k,xn,yn,point,xpoint,ypoint,vecter[5][2],Sc[8][2];
  float weight[4],index[4],minindex,depth,x,y,z;
  float cude1[8][3]={{0,0, 0}, {0,0, 0.4}, {0,0.4, 0},{0,0.4,0.4},
                     {0.4,0, 0}, {0.4,0, 0.4}, {0.4,0.4, 0},{0.4,0.4,0.4},
                    };                    /* for draw line cude 8 side */
  int sidecude1[4][5]={{0,2,3,1,0}, {0,1,5,4,0}, /* side plan */
                       {4,6,7,5,4}, {3,2,6,7,3}};
  /*-------------------------*
   *  view port parameter    *
   *-------------------------*/
  L = Vx = maxx-maxx/6;
  M = Vy = getmaxy()/2.0-50;
  Vd = 80; s = 25;
  D=12.0;
  for(i=0;i<8;i++){
    x = cude1[i][0], y = cude1[i][1], z = cude1[i][2];
          projection_(x,y,z,&xn,&yn,&depth);
    Sc[i][0] = xn+offsetx, Sc[i][1] = yn+offsety;   /*-- cereen coordinate --*/
    z_ax[i]=zee;
    }
  /* sort */
 for(i=0;i<4;i++){
   weight[i]=0;
   for(j=0;j<5;j++){
     x= sidecude1[i][j];
     weight[i] = weight[i] + z_ax[x];
     }
   }
 for(i=0;i<4;i++) index[i]=i;
```

```
 for(i=0;i<4;i++){
    minindex = i;
    for(j=i+1;j<4;j++) if(weight[index[minindex]] > weight[index[j]]) minindex=j;
    x=index[i];
    index[i] = index[minindex];
    index[minindex]=x;
    }
  for(i=0;i<4;i++){
    for(j=0;j<5;j++){
      point = sidecude1[index[i]][j];      /*-- get side of direction[5][5]---*/
      vecter[j][0] = Sc[point][0];   vecter[j][1] = Sc[point][1];
      }
    setcolor(color);
   for(k=0;k<5;k++){
      xpoint=vecter[k][0]; ypoint=vecter[k][1];
      if(k==0)          moveto( xpoint , ypoint );
      else  lineto(xpoint ,ypoint );
      }
   }
}
void  keyfunc(x1,y1,x2,y2,t_color,b_color,b_groud,str,back)
 int x1,y1,x2,y2,t_color,b_color,b_groud,back;
 char *str;
 {
          int x,l,h;
          settextjustify(0,2);  setfillpattern(Gray50,b_groud);
          if(x1 < 0) x1=0;      if(x2 > maxx) x2=maxx;
       if(y1 < 0) y1=0;         if(y2 > maxy) y2=maxy;
          bar(x1,y1,x2,y2);
          setcolor(t_color);    setlinestyle(0,1,1);
          line(x1,y1,x2,y1);    line(x1+1,y1+1,x2-1,y1+1);
          line(x1,y1,x1,y2);    line(x1+1,y1+1,x1+1,y2-1);
          setcolor(b_color);
          line(x2-1,y1+1,x2-1,y2-1);      line(x2,y1,x2,y2);
          line(x1+2,y2-1,x2-1,y2-1);      line(x1,y2,x2,y2);
          x = (x2-x1)/2;          x = x1+x;
          l = textwidth(str); h = textheight(str);
          setcolor(back);         outtextxy(x-(l/2),1+(y1+y2)/2-h/2,str);
}
void  box_(x1,y1,x2,y2,t_color,b_color,b_groud)
 int x1,y1,x2,y2,t_color,b_color,b_groud;
 {
          int w=7;
          settextjustify(0,2);  setfillpattern(Gray50,b_groud);
          if(x1 < 0) x1=0;      if(x2 > maxx) x2=maxx;
       if(y1 < 0) y1=0;         if(y2 > maxy) y2=maxy;
          bar(x1,y1,x2,y2);
          setcolor(t_color);    setlinestyle(0,1,1);
          line(x1,y1,x2,y1);    line(x1+1,y1+1,x2-1,y1+1);
          line(x1,y1,x1,y2);    line(x1+1,y1+1,x1+1,y2-1);
          line(x2-w,y1+w,x2-w,y2-w);   line(x2-w-1,y1+w,x2-w-1,y2-w);
          line(x1+w,y2-w,x2-w,y2-w);   line(x1+w,y2-w-1,x2-w,y2-w-1);
          setcolor(b_color);
          line(x2-1,y1+1,x2-1,y2-1);      line(x2,y1,x2,y2);
          line(x1+2,y2-1,x2-1,y2-1);      line(x1,y2,x2,y2);
          line(x1+w,y1+w,x2-w,y1+w); line(x1+w,y1+w+1,x2-w,y1+w+1);
          line(x1+w,y1+w,x1+w,y2-w);      line(x1+w+1,y1+w,x1+w+1,y2-w);
}
check_key(x1,y1,sizeX,sizeY,gy,gx,page)
```

```
int *x1,*y1,*page,sizeX,sizeY,gy,gx;
{
  int key,yy1,xx1,oldX1,oldY1,oldpage,tmp_p;
  xx1 = *x1;  yy1 = *y1;
  oldpage = tmp_p = *page;
  oldX1=xx1;oldY1=yy1;
  while((key=getkey()) != ENTER){
    if(key==DOWN){
          yy1 += sizeY+gy;  tmp_p++;
          if(tmp_p == 3) { xx1+=sizeX; yy1 = maxy/4*2+90;}
          if(tmp_p>5) { tmp_p=0, /*yy1=*y1,*/ yy1 = maxy/4*2+90;
                        xx1 = sizeX*4+10;/* xx1=*x1;*/}
          }
    if(key==UP){
          yy1 -= sizeY+gy;   tmp_p--;
          if(tmp_p<0) { xx1 += sizeX; yy1 = maxy/4*2+90+((sizeY+gy)*2) ;
                    tmp_p=5;}
          if(tmp_p>5) { xx1 = sizeX*4+10 ; yy1 = maxy/4*2+90; tmp_p=0; }
          if(tmp_p == 2) { xx1 = *x1; yy1 = yy1 = maxy/4*2+90+(sizeY+gy)*2;}
          }
  keyfunc(oldX1,oldY1,oldX1+sizeX-gx,
          oldY1+sizeY,50,10,60,thr[oldpage],75);
  keyfunc(xx1,yy1,xx1+sizeX-gx,yy1+sizeY,10,50,40,thr[tmp_p],70);
  oldpage=tmp_p;
  oldX1=xx1;oldY1=yy1;
  }
  sound(2000); delay(30);  nosound();
  *x1 = xx1; *y1 = yy1;
  *page = tmp_p;
}
int check_(int x,int y,int i,int color)
{
   int key;
   char str[10];
   itoa(i,str,10);outtextxy(x,y,str);
   while((key=getkey()) != ENTER){
    if(key==DOWN){ i--;   if(i<=-150) i=-150;   }
    if(key==UP){ i++;  if(i>=255) i=255;  }
    clsgr(x,y,x+30,y+10,color);
    itoa(i,str,10);outtextxy(x,y,str);
    }
          sound_(); return(i);
}
void show_original()
{
  int y1;
  sizeX = maxx/6;
  sizeY = maxy/4;
  y1=maxy/4*2+80;
  clsgr(15,15,sizeX*4-15,maxy-15,BLACK);
  keyfunc(maxx/6*4+10,maxy/4*2-80,maxx-10,y1-10,60,20,10/*90*/,"",70);
  disp(name0,name1,-200,0);
  sound_();
}
void Show_result()
{
  int y1;
  sizeX = maxx/6;
  sizeY = maxy/4;
```

```
  y1=maxy/4*2+80;
  keyfunc(maxx/6*4+10,maxy/4*2-80,maxx-10,y1-10,60,20,10/*90*/,"",70);
  clsgr(15,15,sizeX*4-15,maxy-15,BLACK);
  Detect_pixel(name0, name1,-200, -sizeY,REMOVE_ALL);
  sound_();
}
void Remove_all(int max,int xmin,int ymin)
{
  int y1;
  char str[10];
  float xf,yf,zf;
  sizeX = maxx/6;
  sizeY = maxy/4;
  y1=maxy/4*2+80;
  keyfunc(maxx/6*4+10,maxy/4*2-80,maxx-10,y1-10,60,20,10/*90*/,"",70);
     keyfunc(sizeX*4+10,10,maxx-10,maxy/4*2-112,40,15,112,"",70);
     outtextxy(sizeX*4+10,30,"   THRESHOLD THAT SELECTED"); setcolor(72);
     outtextxy(sizeX*4+20,60," xcud1 = ");
     itoa(xcud1,str,10);outtextxy(sizeX*4+90,60,str);
     outtextxy(sizeX*4+130,60," xcud2 = ");
     itoa(xcud2,str,10);outtextxy(sizeX*4+200,60,str);
     outtextxy(sizeX*4+20,80," ycud1 = ");
     itoa(ycud1,str,10);outtextxy(sizeX*4+90,80,str);
     outtextxy(sizeX*4+130,80," ycud2 = ");
     itoa(ycud2,str,10);outtextxy(sizeX*4+200,80,str);
     outtextxy(sizeX*4+20,100," xsha1 = ");
     itoa(xsha1,str,10);outtextxy(sizeX*4+90,100,str);
     outtextxy(sizeX*4+130,100," xsha2 = ");
     itoa(xsha2,str,10);outtextxy(sizeX*4+200,100,str);
     outtextxy(sizeX*4+20,120," ysha1 = ");
     itoa(ysha1,str,10);outtextxy(sizeX*4+90,120,str);
     outtextxy(sizeX*4+130,120," ysha2 = ");
     itoa(ysha2,str,10);outtextxy(sizeX*4+200,120,str);
     clsgr(15,15,sizeX*4-15,maxy-15,BLACK);
     Detect_pixel(name0, name1,-200, 0,DETECT_ALL);
     cudeside2(50,50,60,70);
     Disp_2d_color(max,xmin,ymin);sound_();
}
void thr_cloud_A(int max,int xmin,int ymin)
{
  int y1;
  float xf,yf,zf;
  sizeX = maxx/6;
  sizeY = maxy/4;
  y1=maxy/4*2+80;
  init_viewpara_(140.0,50.0);
  draw_axis();
  Disp_2d_color(max,xmin,ymin);
  sound_();
  keyfunc(maxx/6*4+10,maxy/4*2-80,maxx-10,y1-10,60,20,10/*90*/,"",70);
  cudeside2(50,50,60,70);
  do{
   do {
     keyfunc(sizeX*4+10,10,maxx-10,maxy/4*2-112,40,15,112,"",70);
     setcolor(72);outtextxy(sizeX*4+30,30," SELECT THRESHOLD CLOUD A");
     outtextxy(sizeX*4+50,60," xcud1 = ");
     xcud1 = check_(sizeX*4+130,60,xcud1,122);
     outtextxy(sizeX*4+50,80," xcud2 = ");
     xcud2 = check_(sizeX*4+130,80,xcud2,122);
```

```
        outtextxy(sizeX*4+50,100," ycud1 = ");
        ycud1 = check_(sizeX*4+130,100,ycud1,122);
        outtextxy(sizeX*4+50,120," ycud2 = ");
        ycud2 = check_(sizeX*4+130,120,ycud2,122);
        outtextxy(sizeX*4+60,140,"Are you OK.:");
       }
    while(toupper(getche())!='Y');sound_();
            clsgr(15,15,sizeX*4-15,maxy-15,BLACK);
            draw_axis();
            Disp_2d_color(max,xmin,ymin);sound_();getch();
            Detect_pixel(name0, name1,-200, 0,DETECT_CUD);
            outtextxy(sizeX*4+60,160,"Are you Accept.:");
  }
  while(toupper(getche())!='Y');sound_();
}
void thr_shadow_A(int max,int xmin, int ymin)
{
  int y1;
  float xf,yf,zf;
  init_viewpara_(320.0,45.0);
  draw_axis();
  Disp_2d_color(max,xmin,ymin);
  sound_();
  sizeX = maxx/6;
  sizeY = maxy/4;
  y1=maxy/4*2+80;
  keyfunc(maxx/6*4+10,maxy/4*2-80,maxx-10,y1-10,60,20,10/*90*/,"",70);
  cudeside2(-50,50,73,70);
  do{
   do {
      keyfunc(sizeX*4+10,10,maxx-10,maxy/4*2-112,40,15,112,"",70);
      setcolor(72);outtextxy(sizeX*4+30,30," SELECT THRESHOLD SHADOW A");
      outtextxy(sizeX*4+50,60," xsha1 = ");
      xsha1 = check_(sizeX*4+130,60,xsha1,122);
      outtextxy(sizeX*4+50,80," xsha2 = ");
      xsha2 = check_(sizeX*4+130,80,xsha2,122);
      outtextxy(sizeX*4+50,100," ysha1 = ");
      ysha1 = check_(sizeX*4+130,100,ysha1,122);
      outtextxy(sizeX*4+50,120," ysha2 = ");
      ysha2 = check_(sizeX*4+130,120,ysha2,122);
      outtextxy(sizeX*4+60,140,"Are you OK.:");
     }
   while(toupper(getche())!='Y');sound_();
            clsgr(15,15,sizeX*4-15,maxy-15,BLACK);
        draw_axis();
            Disp_2d_color(max,xmin,ymin);sound_(); getch();
            Detect_pixel(name0, name1,-200, 0,DETECT_SHA);
            outtextxy(sizeX*4+60,160,"Are you Accept.:");
  }
  while(toupper(getche())!='Y');sound_();
}
void main_menu1()
{
  int x,x1,x2,y1,y2,gx,gy,sizeX,sizeY,index,quit;
  sizeX = maxx/6;   sizeY = 50;   gx=20;gy=20;
  x1=maxx/4*3-100,   y1=maxy/4*2+80;
  Make_2D(name0,name1,&ymin,&ymax,&xmin,&xmax);
  max_min_LTB(&max,&min);
  box_(sizeX*4,0,maxx,maxy/4*2-102,40,15,80);
```

```
    keyfunc(sizeX*4,maxy/4*2-100,maxx,y1-2,60,20,90,"",70);
    box_(0,0,sizeX*4,maxy,60,20,100);
    keyfunc(sizeX*4,y1,maxx,maxy,60,20,80,"",70);
    x1=sizeX*4+10;   y1=y1+10;
    for(x=0;x<6;x++){
        keyfunc(x1,y1,x1+sizeX-gx,y1+sizeY,60,20,50,thr[x],70);
        y1+=sizeY+gy;
        if(x==2){
              x1+=sizeX; y1=maxy/4*2+90;
              }
      }
   x1=sizeX*4+10;  y1=maxy/4*2+80+10;  index=0;
   keyfunc(x1,y1,x1+sizeX-gx,y1+sizeY,20,60,50,thr[index],70);
   do {
      check_key(&x1,&y1,sizeX,sizeY,gy,gx,&index);
      switch(index) {
           case 0 : thr_cloud_A(max,xmin,ymin);   break;
           case 1 : thr_shadow_A(max,xmin,ymin); break;
           case 2 : Remove_all(max,xmin,ymin);   break;
           case 3 : show_original(); break;
           case 4 : Show_result();  break;
           default: closegraph();exit(1);
      };
           keyfunc(x1,y1,x1+sizeX-gx,y1+sizeY,20,60,50,thr[index],70);
    } while(1);
}
/*-------------------------------------------------------*
 * Program mian 2-d histogram remove cloud and shadow  *
 * ------------------------------------------------------*/
void main(void )
{
 int key;
 clrscr();
 printf("\n... 2-DIMEMSION FOR CLOUD DETECTION ...");
 printf("\n... VERSION 1.0 ...\n... FILE IMAGE BMP FORMAT ...");
 printf("\n... TYPE ...D=286x190.... E=128x128... O=OTHER...");
 key=toupper(getch());
 if(key == 'D') {
  printf("\n... IMAGE SIZE 286x190 ...\n... LOADING...IMAGE...WAIT...");
  strcpy(name0,"a286.bmp");  strcpy(name1,"b286.bmp"); /* 286x190 */
  }
 else if(key == 'E') {
  printf("\n... IMAGE SIZE 128x128 ...\n... LOADING...IMAGE...WAIT...");
  strcpy(name0,"a128.bmp"); strcpy(name1,"b128.bmp");
 }
 else  {
  printf("\nfile A :: "); scanf("%s",&name0);
  printf("file B :: ");   scanf("%s",&name1);
  ReadSize(name0,&xpic,&ypic);
 }
  /*-----------------------*/
  /* init threshold parameter */
  /*-----------------------*/
  xcud1=93  , xcud2= 255;
  ycud1=10  , ycud2= 210;
  xsha1=10  , xsha2= 67;
  ysha1=-5  , ysha2=-81;
  opengraph();
  MaxX=maxx=getmaxx();MaxY= maxy = getmaxy();
```

```
/*-------------------------*
 *  view port parameter    *
 *-------------------------*/
  SCF = (200.0/MaxY)/(270.0/MaxX);
  L = Vx = getmaxx()/2.0-150;
  M = Vy = getmaxy()/2.0-50;
  Vd = 80; s = 10;
  D=8.0;
 init_viewpara_(140.0,45.0);
 main_menu1();
 closegraph();
}
```

```
/*-------------------------------------------------*
 * PPROGRSAM   : dual threshold for cloud detector  *
 * Date        : 4/2537                             *
 * DATA        : image BMP format                   *
 *-------------------------------------------------*/
#include  <stdio.h>
#include  <graphics.h>
#include  <math.h>
#include  <alloc.h>
#include  "n_menu.h"
#include  "bmp_3d.h"
unsigned char huge *A;
unsigned char huge *B;
unsigned char huge *C;
#define   X x=0;x<xpic;x++
#define   Y y=0;y<ypic;y++
int       maxx,maxy,size,xpic,ypic,Ta,Tr,color_old=70,back=10;;
int       xmin,ymin,xmax,ymax;
int       g_min,g_max,f_max,fd_max;
int       LTB_a[256],LTB_b[256],LTB_diff[256];
long      z;

void  display_hist(float size,int xx,int yy,int *LTB)
{
  int      x1,y1,y11,x11,x2,y2,i,maxhist,scale,winhight=100,max_y,vert,tab,tab1,pixel1_,init=0;
  char str[5];
  float pixel,pixel1;
  maxhist = LTB[0];
  x1=(int)(xx/size)-5; y1=(int)(yy/size)+5; x2=(int)(xx+256)/size+5 ;
  for(i=0;i<256;i++)if(LTB[i]>maxhist)maxhist=LTB[i];
  scale = maxhist/winhight;  max_y = (yy-maxhist/scale)/size;
  y2 = max_y-10;
  for(i=0;i<256;i++){ setcolor(75);
    x11=(xx+i)/size; y11=yy/size;
    line(x11,y11,x11,(yy-LTB[i]/scale)/size);
    if(fmod(i,20.0)==0.0||i==255){ itoa(i,str,10);setcolor(70);
       outtextxy(x11-10,y11+15,str);} setcolor(80);
    if(fmod(i,10.0)==0.0||i==255)  line(x11,y11,x11,y11+10);
  }
 z = (long)xpic*ypic;
 pixel1 = (float)(maxhist)/z*100;          /* find percent of total pixel */
 tab = (int)pixel1/5;
 vert = y11 - max_y;
 tab1 = vert/5;
 for(i=0;i<=vert;i++){
    if(fmod(i,tab1)==0.0){
    line(x1,y1-i,x1+5,y1-i);  itoa(init,str,10);init+=tab;
    setcolor(70);  outtextxy(x1-25,y1-i,str); }
    }
  setcolor(55);
  line(x1,y1,x2,y1);line(x1,y2,x1,y1);line(x1,y2,x2,y2);line(x2,y2,x2,y1);
  settextstyle(3,VERT_DIR,1);outtextxy(x1-30,y2,"FREQUENCY OFf PIXEL (%)");
  settextstyle(3,HORIZ_DIR,1);outtextxy((x2-x1)*3/5,y1+30,"GRAY LEVEL ");
}
void  LTB_hist(unsigned char huge *image,int *LTB)
{
  int      x,y;
  long     j;
  for(x = 0; x<256; x++)  LTB[x] = 0; /* clear value in LTB */
```

```
  for(Y)
    for(X){
      j = xpic*(long)y+x;
      LTB[image[j]]++;                /* count gray value in to LTB */
      }
  y =  LTB[0];
  for(x = 0; x<256; x++)
   if(LTB[x] > y ) { f_max = x; y = LTB[x];}
}
void  LTB_diff_hist(ImageA,ImageB)
unsigned char huge *ImageA, huge *ImageB;
{
  int      x,y,diff;
  long     j;
  for(x = 0; x<256; x++)  LTB_diff[x] = 0;  /* clear value in LTB */
  for(Y)
    for(X){
      j = xpic*(long)y+x;
      diff = abs(ImageA[j]-ImageB[j]);
      LTB_diff[diff]++;
      }
   y =  LTB_diff[0];
   for(x = 0; x<256; x++)
    if(LTB_diff[x] > y ) { fd_max = x; y = LTB_diff[x];}
}
max_min(Image)
unsigned char huge *Image;
{
  int      x,y,j;
  long     l;
  g_min=200;g_max=0;
  for(X)
    for(Y){
      l = xpic*(long)y+x; j = Image[l];
      if(g_min>j) g_min=j;   if(g_max<j) g_max=j;
      }
}
max_min_diff(ImageA,ImageB)
unsigned char huge *ImageA, huge *ImageB;
{
  int      x,y,diff;
  long     j;
  g_min=200;g_max=0;
  for(Y)
    for(X){
      j = xpic*(long)y+x;
      diff = abs(ImageA[j]-ImageB[j]);
      if(g_min>diff) g_min=diff;   if(g_max<diff) g_max=diff;
      }
}
int check_(int x,int y,int i,int color)
{
   int      key;
   char     str[10];
   itoa(i,str,10);outtextxy(x,y,str);
   while((key=getkey()) != ENTER){
     if(key==DOWN){ i--;   if(i<=0) i=0;   }
     if(key==UP){ i++;  if(i>=255) i=255;  }
     clsgr(x,y,x+30,y+10,color);
```

```
        itoa(i,str,10);outtextxy(x,y,str);
        }
       sound_();return(i);
  }
  sound_()
  {
    sound(1000);delay(90);nosound();
  }
  /*-------------------
   detection cloud pixel
   -------------------*/
  detect_1dimension(int x1,int y1)
  {
    int       x,y,a,diff,key;
    long      k;
     for(Y)
       for(X){
             k = xpic*(long)y+x;
             a = A[k];
             diff = abs(A[k]-B[k]);
             if(a>Ta && diff>Tr){
              putpixel(x1+x,y1/2+y,60); putpixel(x1+x,y1*4+y,(B[k]>>2));
              }
             else{
              putpixel(x1+x,y1/2+y,110); putpixel(x1+x,y1*4+y,(A[k]>>2));
              }
       }
  }
  /*-------------------
     Remove cloud pixel
   -------------------*/
  remove_1d()
  {
    int x,y,a,diff;
    long k;
    k=(long)xpic*(long)ypic;
    if((C=farcalloc(k,sizeof(char)))==NULL)
     { closegraph();printf("\n Can'nt allocate memmort of image A");exit(1);}
     for(Y)
       for(X){
             k = xpic*(long)y+x;
             a = A[k];
             diff = abs(A[k]-B[k]);
             if(a>Ta && diff>Tr)
               C[k] = B[k];
             else
               C[k] = A[k];
     }
  }
  /*-----------------------------------
   main program dual threshold detection
   -----------------------------------0--*/
  main()
  {
    int       x1,y1,sizeX,sizeY,index,old_color,key;
    static char name1[20],name2[20],str[10];
    clrscr();
    printf("\n... THE USE OF DUAL THRESHOLD FOR CLOUD DETECTION ...");
    printf("\n... VERSION 1.0 ...\n... FILE IMAGE BMP FORMAT ...");
```

```
printf("\n... TYPE ...D=286x190.... E=286x202...O=OTHER...");
key=toupper(getch());
if(key == 'D')
{
 printf("\n... IMAGE SIZE 286x190 ...\n... LOADING...IMAGE...WAIT...");
 xpic=286; ypic=190 ;
 strcpy(name1,"a190.bmp");  strcpy(name2,"b190.bmp");
}
else if(key == 'E')
{
 printf("\n... IMAGE SIZE 286x202 ...\n... LOADING...IMAGE...WAIT...");
 xpic=286; ypic=202 ;
 strcpy(name1,"a286.bmp"); strcpy(name2,"b286.bmp");
}
else
{
 printf("\nfile A :: "); scanf("%s",&name1);
 printf("file B :: ");   scanf("%s",&name2);
 ReadSize(name1,&xpic,&ypic);
}
open_memnew();              /* allocate memory */
readBMP(A,name1);                   /* read data image A */
readBMP(B,name2);                   /* read data image B */
opengraph();
maxx=getmaxx();maxy=getmaxy();
sizeX = maxx/6;  x1= 30;
sizeY = maxy/4; y1=maxy/4*2+80;
box_(sizeX*4,0,maxx,maxy/4*2-102,40,15,80);
keyfunc(sizeX*4,maxy/4*2-100,maxx,y1-2,60,20,90,"",70);
box_(0,0,sizeX*4,maxy,60,20,100);
keyfunc(sizeX*4,y1,maxx,maxy,60,20,80,"",70);
do{
   clsgr(15,15,sizeX*4-15,maxy-15,back);
   LTB_hist(A,LTB_a);
   display_hist(0.6,40,140,LTB_a); max_min(A);setcolor(73);
   outtextxy(sizeX*2+x1,80,"Max_Level ="); itoa(g_max,str,10); outtextxy(sizeX*3+20,80,str);
   outtextxy(sizeX*2+x1,100,"Min_Level ="); itoa(g_min,str,10); outtextxy(sizeX*3+20,100,str);
   outtextxy(sizeX*2+x1,120,"Max_freq  ="); itoa(f_max,str,10); outtextxy(sizeX*3+20,120,str);
   LTB_diff_hist(A,B);
   display_hist(0.6,40,300,LTB_diff);max_min_diff(A,B);setcolor(73);
   outtextxy(sizeX*2+x1,340,"Max_Level ="); itoa(g_max,str,10); outtextxy(sizeX*3+20,340,str);
   outtextxy(sizeX*2+x1,360,"Min_Level ="); itoa(g_min,str,10); outtextxy(sizeX*3+20,360,str);
   outtextxy(sizeX*2+x1,380,"Max_freq  ="); itoa(fd_max,str,10); outtextxy(sizeX*3+20,380,str);
   do{
     box_(sizeX*4+10,10,maxx-10,maxy/4*2-112,40,15,112);
     setcolor(72);outtextxy(sizeX*4+30,40," SELECT THRESHOLD Ta ");
     Ta = check_(sizeX*4+90,60,f_max,122);
     outtextxy(sizeX*4+30,80," SELECT THRESHOLD Tr ");
     Tr = check_(sizeX*4+90,100,fd_max,122);
     outtextxy(sizeX*4+60,120,"Are you OK.:");
     }
 while(toupper(getche())!='Y');sound_();
          clsgr(15,15,sizeX*4-15,maxy-15,back);
          detect_1dimension(sizeX,sizeY/2);
          outtextxy(sizeX*4+60,140,"Are you Accept.:");
 }
while(toupper(getche())!='Y');sound_();
          outtextxy(sizeX*4+60,160,"Are you Save.:");
          if(toupper(getche())=='Y'){ remove_1d();
```

```
                    saveBMP("re_1d.bmp",C); farfree(C);}
            sound_();
  closegraph();
}
```

```
/*-------------------------------------------
 * PROGRAM : read and write bitmat BMP format
 *         : BMP_3d.h
 *-------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <string.h>
#include <graphics.h>
#include <conio.h>
int     xpic,ypic,XPIX,YPIX;
typedef struct {
        char id[2];
        long filesize;
        int  reserved[2];
        long headersize;
        long infoSize;
        long width;
        long depth;
        int  biPlanes;
        int  bits;
        long biCompression;
        long biSizeImage;
        long biXPelsPerMeter;
        long biYPelsPerMeter;
        long biClrUsed;
        long biClrImportant;
   } BMPHEAD;
BMPHEAD bmp;
/*-------------------------------*/
unsigned char huge *A;
unsigned char huge *B;
int map_l,map_h,Th_shB_clA,Th_shA_clB;          /* threshold for detect */
float D,xtheta,zphi,s;
int  COLOR,pre_mode;
/*-------------------------------*/
void ReadSize(Name,x_size,y_size)
char Name[];
int *x_size,*y_size;
{
  FILE *infile;
  if((infile = fopen(Name,"rb")) == NULL)
    { printf(" Open file BMP format error ... \n");exit(1); }
  memset((char *)&bmp,0,(long)sizeof(BMPHEAD));
  if(fread((char *)&bmp,1,(long)sizeof(BMPHEAD),infile) != sizeof(BMPHEAD))
   { printf(" Error Reading BMP Header ... \n");exit(1); }
  *x_size = (int)bmp.width;
  *y_size = (int)bmp.depth;
  fclose(infile);
}
void readBMP(image,name)
unsigned char huge *image;
char name[];
{
long                    z;
int                     x,y,bytes;
FILE                    *infile;
static unsigned char    imagebuff[1024];
BMPHEAD                 bmp;
```

```
  infile = fopen(name,"rb");
  if(infile == NULL) {
          printf(" Open file error..\n");
          exit(1);
  }
 /* //read HEADER */
  memset((char *)&bmp,0,(long)sizeof(BMPHEAD));
  if(fread((char *)&bmp,1,(long)sizeof(BMPHEAD),infile) != sizeof(BMPHEAD)) {
          printf("fread HEADER error");
          exit(1);
  }
 /* //for THE odd XPIX */
  xpic=bytes=bmp.width;
  ypic=bmp.depth;
  if(bytes & 0x0003) {
          bytes |= 0x0003;
          ++bytes;
  }
 /* //write ARRAY*/
  fseek(infile,bmp.headersize,SEEK_SET);
  memset((unsigned char *)&imagebuff,0,bytes);
  for(y=0;y<bmp.depth;++y){
          fread(imagebuff,1,bytes,infile);
          for(x=0;x<xpic;x++){
              z=xpic*(long)y+x;
              image[z]=imagebuff[x];
          }
  }
  fclose(infile);
}
/*-------------------------------------*/
void saveBMP(name,image)
char      name[];
unsigned char huge *image;
{
long                    z;
int                     x,y,bytes;
FILE                    *outfile;
static unsigned char    imagebuff[1024];
BMPHEAD                 bmp;
 /*for THE odd XPIX*/
 bytes=xpic;
 if(bytes & 0x0003) {
          bytes |= 0x0003;
          ++bytes;
 }
 memset((char *)&bmp,0,(long)sizeof(BMPHEAD));
 memcpy(bmp.id,"BM",2);
 bmp.headersize=1078L;
 bmp.width=(long)xpic;
 bmp.depth=(long)ypic;
 bmp.filesize=bmp.headersize+(long)bytes*(long)bmp.depth;
 bmp.infoSize=0x28L;
 bmp.bits=8;
 bmp.biPlanes=1;
 bmp.biCompression=0L;
 bmp.biSizeImage=bmp.filesize-bmp.headersize;
 bmp.biXPelsPerMeter=0L;
 bmp.biYPelsPerMeter=0L;
```

```
    bmp.biClrUsed=0L;
    bmp.biClrImportant=0L;
    outfile = fopen(name,"wb");
    if(outfile == NULL) {
            printf(" Open file error..\n");
            exit(1);
    }
    /*write HEADER*/
    if(fwrite((char *)&bmp,1,(long)sizeof(BMPHEAD),outfile) != sizeof(BMPHEAD)) {
            printf("fwrite HEADER error");
            exit(1);
    }
    /*write PALETTE*/
    for(x=0;x<256;++x){
            fputc(x,outfile);/*B*/
            fputc(x,outfile);/*G*/
            fputc(x,outfile);/*R*/
            fputc(0,outfile);
    }
    /*write IMAGE*/
    memset((unsigned char *)&imagebuff,0,bytes);
    for(y=0;y<bmp.depth;++y){
            for(x=0;x<xpic;++x) {
                z=xpic*(long)y+x;
                imagebuff[x]=image[z];
            }
    fwrite(imagebuff,1,bytes,outfile);
    }
    fclose(outfile);
}
/*-------------------------------------*/
void sizeBMP(name)
char name[];
{
FILE                    *infile;
BMPHEAD                 bmp;
 infile = fopen(name,"rb");
 if(infile == NULL) {
            printf(" Open file error..\n");
            exit(1);
 }
 /*read HEADER*/
 memset((char *)&bmp,0,(long)sizeof(BMPHEAD));
 if(fread((char *)&bmp,1,(long)sizeof(BMPHEAD),infile) != sizeof(BMPHEAD)) {
            printf("fread HEADER error");
            exit(1);
 }
 XPIX=xpic=bmp.width;
 YPIX=ypic=bmp.depth;
 fclose(infile);
}
```