การคัดแยกเนื้อเยื่อสมองจากภาพถ่ายทางการแพทย์

HUMAN BRAIN SEGMENTATION FROM MRI

โดย

นายไตรรัตน์ ตั้งพิทยาเวทย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

การคัดแยกเนื้อเยื่อสมองจากภาพถ่ายทางการแพทย์

HUMAN BRAIN SEGMENTATION FROM MRI

โดย

นายไตรรัตน์ ตั้งพิทยาเวทย์

อาจารย์ที่ปรึกษา

รศ.ดร. มนัส สังวรศิลป์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์ ปีการศึกษา 2542

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การคัดแยกเนื้อเยื่อสมองจากภาพถ่ายทางการแพทย์

HUMAN BRAIN SEGMENTATION FROM MRI

ผู้จัดทำ

นายไตรรัตน์ ตั้งพิทยาเวทย์

..... อาจารย์ที่ปรึกษา

(รศ.ดร. มนัส สังวรศิลป์)

ปริญญานิพนธ์เรื่อง การคัดแยกเนื้อเยื่อสมองจากภาพถ่ายทางการแพทย์
HUMAN BRAIN SEGMENTATION FROM MRI

จัดทำโดย นายไตรรัตน์ ตั้งพิทยาเวทย์ รหัส 39014184
อาจารย์ที่ปรึกษา รศ.ดร. มนัส สังวรศิลป์

ปริญญานิพนธ์ฉบับนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....อาจารย์ที่ปรึกษา
(รศ.ดร. มนัส สังวรศิลป์)

การคัดแยกเนื้อเยื่อสมองจากภาพถ่ายทางการแพทย์

นายไตรรัตน์ ตั้งพิทยาเวทย์
รศ.ดร. มนัส สังวรศิลป์ (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2542

บทคัดย่อ

ภาพถ่ายที่ได้จากเครื่อง MRI (Magnetic Resonance Images) มีประโยชน์อย่างมากต่อการวินิจฉัยโรคของแพทย์ แต่เนื่องจากภาพที่ได้มีลักษณะเป็นภาพตัดขวาง 2 มิติ จึงยากต่อการตีความจากภาพ ซึ่งหากเราสามารถแยกเนื้อเยื่อสมองออกเป็นกลุ่มต่างๆได้ จะทำให้แพทย์สามารถนำไปวินิจฉัยโรคได้ง่ายขึ้น

ดังนั้นในปริญญานิพนธ์นี้จะนำเสนอวิธีในการคัดแยกเนื้อเยื่อสมองแต่ละส่วนออกมา โดยได้นำเสนอ 2 วิธี คือ 1) Multithreshold ซึ่งใช้วิธีในการกำหนดช่วงของระดับเทาของเนื้อเยื่อสมองแต่ละส่วน โดยการหาค่าจุดแบ่งของแต่ละช่วงของระดับเทานั้น 2) Fuzzy C-Mean Algorithm ใช้ค่าความเป็นสมาชิกของระดับเทาแต่ละระดับที่มีต่อจุดศูนย์กลางกลุ่มของข้อมูล เป็นตัวแบ่งกลุ่มของระดับเทา ซึ่งผลที่จากแต่ละวิธี คือ สามารถแยกเนื้อเยื่อสมองออกเป็นส่วนต่างๆ ซึ่งได้แก่ White Matter , Gray Matter , Cerebrospinal-fluid , skull ซึ่งแต่ละวิธีใช้เวลาในการประมวลผลแตกต่างกันไป นอกจากนี้ยังมีการพัฒนาให้ใช้เวลาในการประมวลผลที่น้อยลง

HUMAN BRAIN SEGMENTATION FROM MRI

MR. Trairat Tungpittayavat

Assoct.Prof.Dr.Manus Sungworasil (Advisor)

1999

Abstract

Images from MRI (Magnetic Resonance Images) scanner are very useful in medical diagnoses, but these are presented in 2D format, so it is difficult to diagnose. Thus if we can segment MRI Images to each tissue, doctor can comfortably diagnose these tissues .

This thesis presents 2 methods for human brain segmentation. 1) Multithreshold , determining gray level ranges of each tissue by finding many threshold points . 2) Fuzzy C-Mean Algorithm, using membership of each gray level that depends on cluster center for segmenting classes of gray level. In addition, later algorithm is developed to take a small running time. Results of this thesis show that human brain images can be segmented to several distinct tissues such as gray matter, white matter, cerebrospinal-fluid (CSF) and skull.

สารบัญ

หน้า

สารบัญรูป

หน้า

สารบัญตาราง

หน้า

บทที่ 1

บทนำ

ภาพถ่ายทางการแพทย์ที่ได้จากเครื่อง Magnetic Resonance Imaging (MRI) Scanner เป็นประโยชน์อย่างมากในการตรวจวินิจฉัยโรคของแพทย์ ซึ่งภาพที่ได้จาก MRI Scanner นั้นเป็นชุดต่อเนื่องของภาพตัดขวางสองมิติ ซึ่งต้องการแพทย์ผู้เชี่ยวชาญในการตีความจากภาพ ดังนั้นหากเราสามารถทำการแบ่งกลุ่มของเนื้อเยื่อต่างๆ ออกมาได้ จะทำให้แพทย์ตีความเพื่อวินิจฉัยโรคได้ง่ายขึ้น

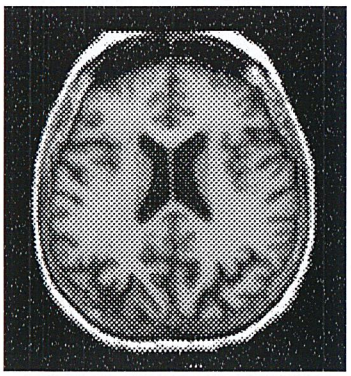
โดยในการทำการคัดแยกเนื้อเยื่อสมองจากภาพ MRI นั้นเราแบ่งแผ่น slice ที่สนใจศึกษาออกเป็น 2 แบบคือ แบบปรกติ และแบบไม่ปรกติ ซึ่ง slice แบบปรกตินั้นประกอบด้วยเนื้อเยื่อสมองทั่วๆไป เช่น กะโหลก (ส่วนของกะโหลกในภาพ MRI นั้นไม่สามารถมองเห็นได้ ซึ่งจะอธิบายในบทต่อไป) , ไขมันใต้ผิวหนัง , white matter , gray matter , cerebrospinal-fluid (CSF) ส่วน slice ที่ไม่ปรกติ ประกอบด้วย ทั้งส่วนของเนื้อเยื่อในสมองที่ปรกติ และไม่ปรกติ เช่น เนื้องอก เป็นต้น ซึ่งในการทำการแบ่งกลุ่มของเนื้อเยื่อสมองนั้นค่อนข้างจะมีปัญหาโดยมี สาเหตุ 2 ประการคือ

1. ส่วนของเนื้อเยื่อที่ผิดปรกติ อาจใกล้เคียงกับเนื้อเยื่อที่ปรกติบริเวณรอบๆ
2. การที่มีเนื้อเยื่อหลายๆชนิดอยู่ในบริเวณเดียวกัน ปัญหานี้แก้ไขได้โดยการใช้อัลกอริธึมในการจำแนกแบบกลุ่ม (clustering algorithm)

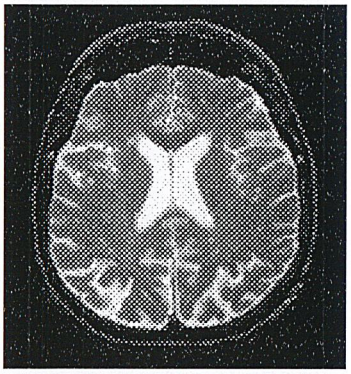
ภาพ MRI ถ่ายในแนวขวางศรีษะโดยในแต่ละชั้นประกอบด้วย ภาพ 3 ภาพ ที่ถ่ายที่ระดับความเข้มสนามแม่เหล็ก 3 ระดับ คือ T1 weight (T1) , T2 weight (T2) , Proton Density (PD) ซึ่งโดยทั่วไปแล้ว เราสามารถแบ่งส่วนของสมองออกเป็น 5 – 6 คลาส คือ

1. คลาสของอากาศ	แทนด้วย	A
2. คลาสของกะโหลก	แทนด้วย	B
3. คลาสของ white matter	แทนด้วย	W
4. คลาสของ gray matter	แทนด้วย	G
5. คลาสของ CSF	แทนด้วย	C
6. คลาสของเนื้องอก	แทนด้วย	T

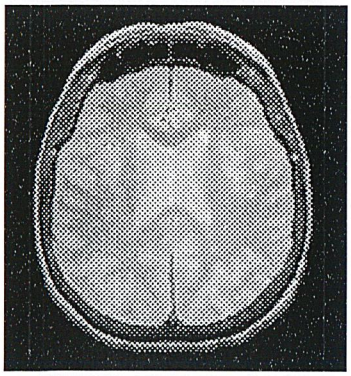
ภาพต้นแบบที่นำมาใช้ในการประมวลผล แสดงได้ดังรูปที่ 1.1



T1

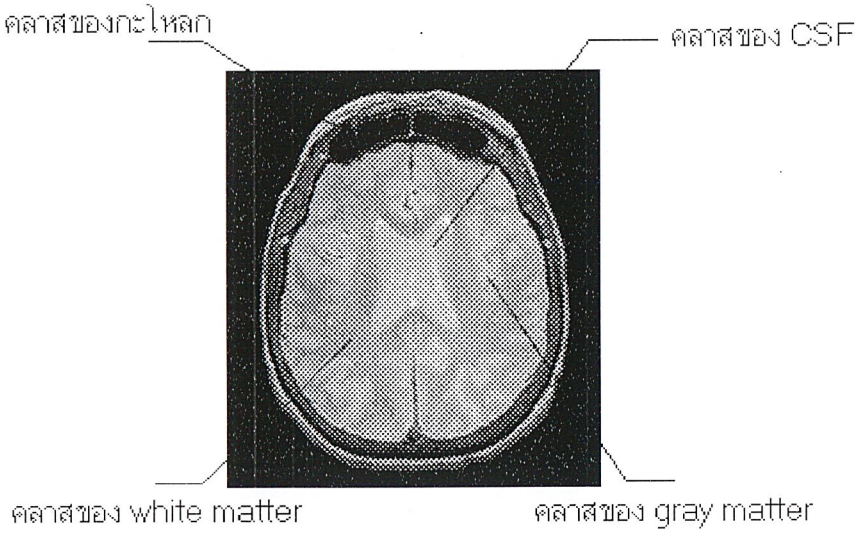


T2



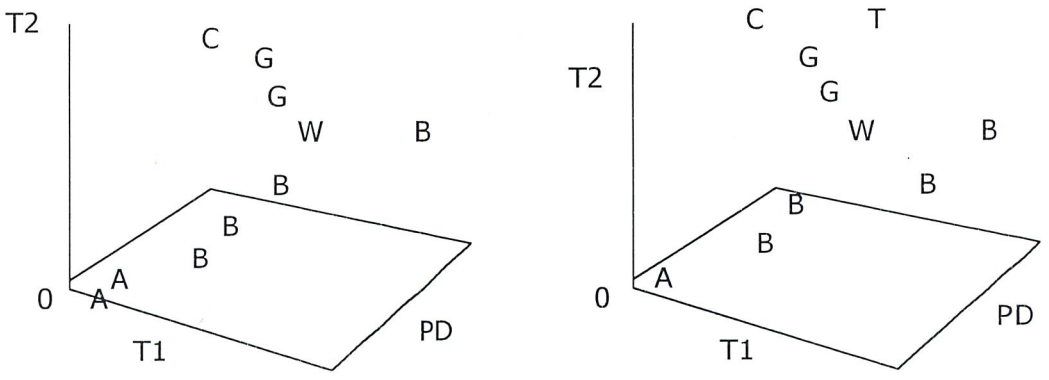
PD

รูปที่ 1.1 แสดงภาพ T1,T2,PD ชั้นที่ 33

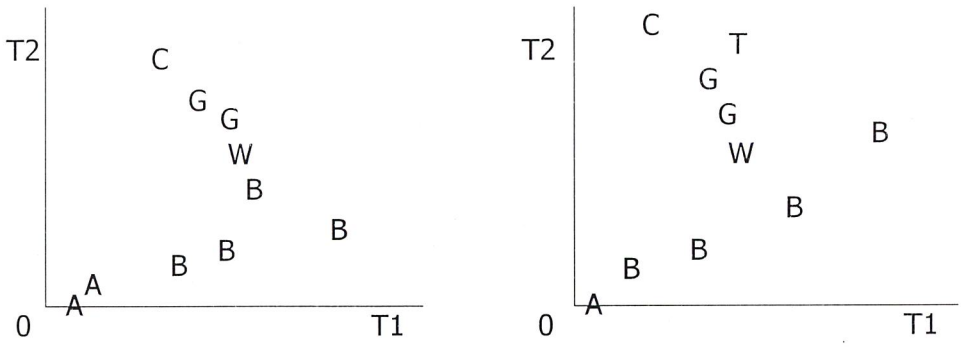


รูปที่ 1.2 แสดงตำแหน่งของคลาสต่างๆ

จากรูปเป็นการชี้ตำแหน่งของคลาสต่างๆ คลาสของอากาศอยู่ในบริเวณทั่วไป คลาสของกะโหลกคือ บริเวณนอกสุด คลาสของ white matter มีลักษณะเป็นแขนงยื่นออกไป 4 ข้าง คลาสของ CSF มีลักษณะเป็น "X" อยู่ในบริเวณด้านในของ white matter และคลาสของ gray matter อยู่ในบริเวณด้านนอกระหว่าง white matter และ กะโหลก ซึ่งโดยทั่วไปแล้ว ถ้าเป็น slice แบบไม่ปรกติ เช่น มีเนื้องอก พบว่าโดยทั่วไปแล้วเนื้องอกนั้นจะอยู่ในบริเวณ white matter และ CSF

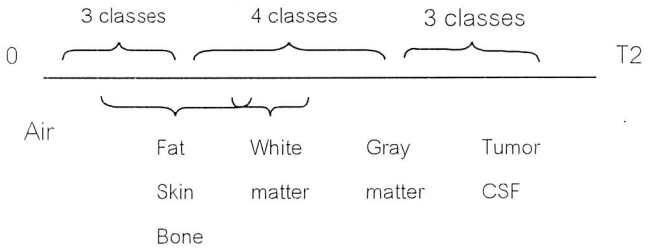


รูปที่ 1.3 แสดงคลาสต่างๆ ใน 3 มิติ (T1,T2,PD)



รูปที่ 1.4 แสดงคลาสต่างๆในระนาบ T1 และ T2

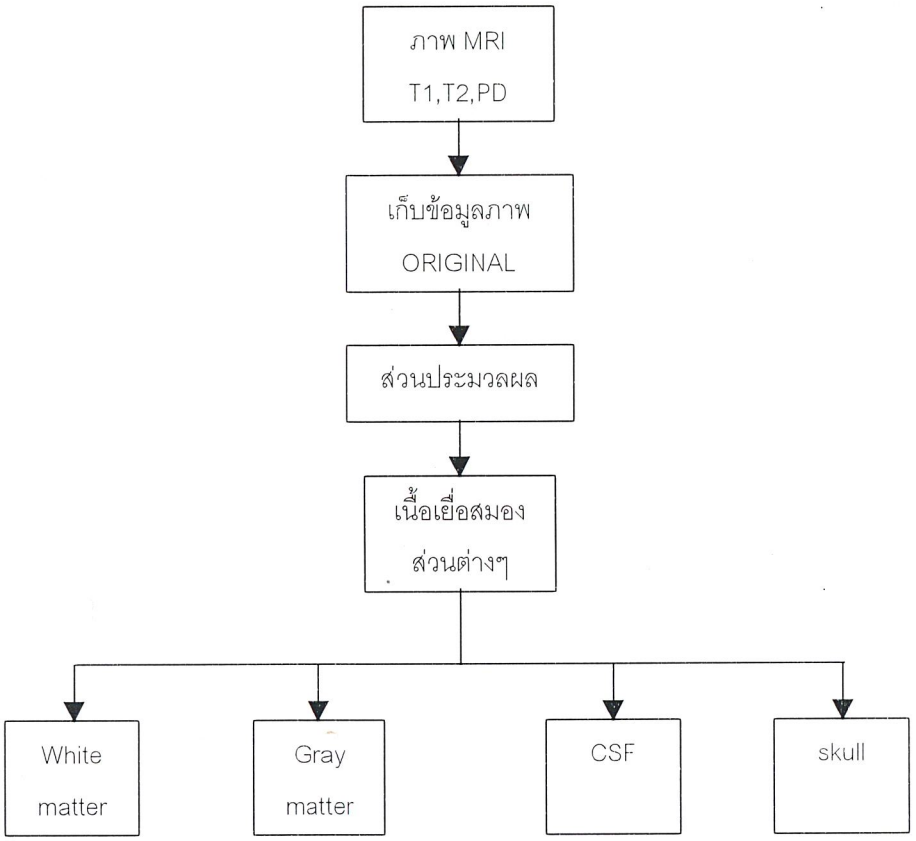
เราสามารถ projection คลาสต่างๆ ลงบนแกน T2 ได้ซึ่งเห็นว่า แต่ละคลาสไม่ได้มีขอบเขตที่แน่นอนทำให้การ segmentation ทำได้ยากจึงนำ clustering algorithm มาใช้



รูปที่ 1.5 แสดงคลาสต่างๆ ที่ projection ลงบนแกน T2

เนื่องจากพบว่า เนื้อเยื่อสมองส่วนต่างๆ มีขอบเขตในการอยู่ในแต่ละคลาสไม่แน่ นอน ซึ่งนี่ก็คือปัญหาที่ต้องนำมาพิจารณาเป็นอย่างยิ่งในการทำการ Segmentation ให้ได้มาซึ่งเนื้อเยื่อสมองส่วนต่างๆ โดยปริญญานิพนธ์นี้นำเสนอวิธีในการทำ Segmentation เนื้อเยื่อสมองส่วนต่างๆ ทั้งสิ้น 2 วิธี อันได้แก่ Multithreshold และ Fuzzy C-Mean Algorithm โดยจะอธิบายถึงรายละเอียดในบทต่อไป โดยทั้ง 2 วิธีมีส่วนประกอบในการทำงาน ดังรูปที่ 1.5

<u>ส่วนประกอบโดยรวมของการทำ **Segmentation**</u>



รูปที่ 1.6 แสดงส่วนประกอบต่างๆของการทำ Segmentation

ส่วนประกอบต่างๆ สามารถอธิบายได้ดังนี้

1. **ส่วนของภาพต้นแบบ** : เป็นไฟล์ภาพ MRI ในการทำการ Segmentation แต่ละครั้ง (ครั้งละ 1 ชั้น) จะใช้ภาพ MRI จำนวน 3 slices อันได้แก่ T1 , T2 และ PD
2. **ส่วนเก็บข้อมูลภาพต้นแบบ** : ภาพ MRI ต้นแบบแต่ละภาพ จะประกอบด้วยข้อมูลต่างๆ ซึ่งในการทำ Segmentation เราจะทำการเก็บข้อมูลภาพนั้นๆ ในรูปของอะเรย์ (array) 1 มิติ เพื่อความสะดวกในการนำไปกระทำกระบวนการทาง Image Processing
3. **ส่วนประมวลผล** : การทำ Segmentation ในปริญญานิพนธ์ฉบับนี้นำเสนอ รูปแบบในการ Segmentation ทั้งสิ้น 2 วิธีอันได้แก่ Multithreshold และ Fuzzy C-Mean Algorithm
4. **ส่วนเก็บข้อมูลเนื้อเยื่อสมองส่วนต่างๆ** : หลังจากผ่านกระบวนการประมวลผลแล้ว เราจะได้ข้อมูลของเนื้อเยื่อสมองส่วนต่างๆ ซึ่งเราต้องทำการเก็บข้อมูลเหล่านั้นเอาไว้เพื่อนำไปแสดงผลหรือกระทำการอื่นๆ ต่อไป

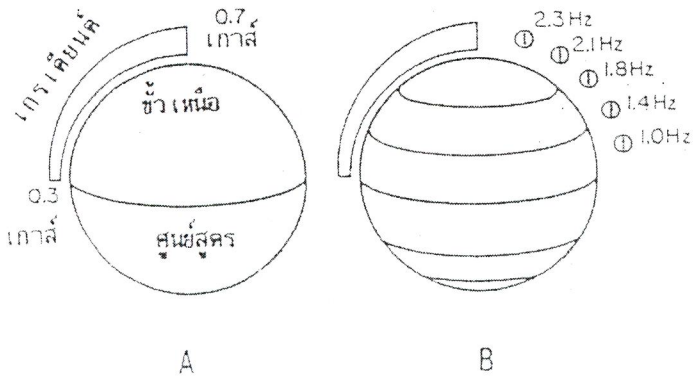
บทที่ 2
หลักการพื้นฐานของ
MRI

2.1 แมกเนติกเรโซแนนซ์ (Magnetic Resonance)

แมกเนติกเรโซแนนซ์ (Magnetic Resonance) เป็นอันตรกริยาระหว่างแม่เหล็กกับสนามแม่เหล็ก ยกตัวอย่าง เข็มทิศที่มีแม่เหล็กกับสนามแม่เหล็กโลก ซึ่งอันตรกริยาระหว่างแม่เหล็กกับสนามแม่เหล็กโลก คือ การปรับแนวของเข็มทิศให้ชี้ในแนว เหนือ-ใต้ เสมอ

ถ้าเราทำการเขี่ยให้ปลายเข็มทิศแกว่งออกจากแนวเหนือ-ใต้ เข็มทิศจะแกว่งไป-มา จนกว่าแรงที่เข็มทิศได้รับจะหมดไปเพราะความฝืดและการปะทะกับอากาศ โดยความถี่ของการแกว่งของเข็มทิศขึ้นกับ เข็มทิศ(ขนาด , น้ำหนัก , และกำลังแม่เหล็ก) และขึ้นกับกำลังของสนามแม่เหล็กโลก เราเรียกความถี่ของการแกว่งนี้ว่า **ความถี่ธรรมชาติ (Natural Frequency)** ซึ่งเป็นสัดส่วนโดยตรงกับกำลังของสนามแม่เหล็ก

กำลังของสนามแม่เหล็กโลกไม่สม่ำเสมอเท่ากันทุกๆแห่ง ที่ขั้วโลกทั้งสองจะมีกำลังมากที่สุด และค่อยๆ ลดลงตามระยะห่างจากขั้วโลก จนน้อยที่สุดบริเวณแนวเส้นศูนย์สูตร เราเรียกสนามแม่เหล็กที่ค่อยๆ ลดหรือค่อยๆ เพิ่มอย่างสม่ำเสมอว่า **สนามแม่เหล็กเกรเดียนต์ (magnetic field gradient)** แสดงได้ดังรูปที่ 2.1



รูปที่ 2.1 (A) สนามแม่เหล็กโลกมีกำลังไม่สม่ำเสมอกัน ที่บริเวณขั้วโลกมีกำลังสนามแม่เหล็กมากกว่าที่เส้นศูนย์สูตร (B) ที่ขั้วโลกเข็มทิศจะกวัดแกว่งเร็วกว่า ความถี่การกวัดแกว่งของเข็มทิศใช้คำนวณหาระดับของเส้นรุ้งอย่างหยาบได้

หลักพื้นฐานสำคัญของแมกเนติกเรโซแนนซ์ จากตัวอย่างเข็มทิศ

1) เมื่อวางเข็มทิศในสนามแม่เหล็ก มันจะปรับแนวขนานกับสนามแม่เหล็ก
2) เมื่อถูกกระตุ้น เข็มทิศจะแกว่งด้วยความถี่ธรรมชาติที่เป็นสัดส่วนกับกำลังของสนามแม่เหล็ก
3) ในสนามแม่เหล็กเกรเดียนต์ ที่รู้การเปลี่ยนแปลงกำลังสนามแม่เหล็ก สามารถคำนวณตำแหน่งของเข็มทิศได้จากความถี่ของการแกว่ง

ขบวนการกระตุ้น : ในการแกว่งชิงช้า หลังจากออกแรงผลักครั้งแรก ชิงช้าจะแกว่งด้วยความถี่ธรรมชาติ ถ้าต้องการให้ชิงช้าแกว่งสูงขึ้น ต้องออกแรงผลักให้ตรงกับจังหวะการแกว่งของชิงช้า เพราะชิงช้าจะรับแรงผลักได้เต็มที่มากที่สุด ถ้าออกแรงผลักไม่ตรงกับการแกว่งของชิงช้า จะเสียแรงโดยเปล่าประโยชน์

ในการกระตุ้นเข็มทิศก็เช่นกัน หากเราทำการเคาะด้วยจังหวะเท่ากับความถี่ธรรมชาติของเข็มทิศจะทำให้เข็มทิศแกว่งแรงขึ้นมากที่สุด **ความถี่ของการกระตุ้นซึ่งทำให้เข็มทิศรับแรงกระตุ้นไปได้มากที่สุด เรียกว่า ความถี่เรโซแนนซ์ (resonance frequency) ซึ่งเท่ากับความถี่ธรรมชาติ**

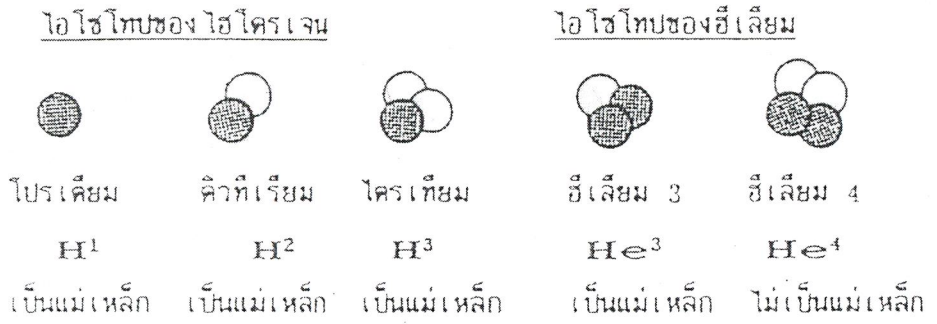
2.2 นิวเคลียร์แมกเนติกเรโซแนนซ์ (Nuclear Magnetic Resonance : NMR)

สนามแม่เหล็กเกิดขึ้นจากการเคลื่อนที่ของอนุภาคที่มีประจุ(บวกหรือลบ)และเนื่องจากไฮโดรเจน เป็นธาตุที่เป็นส่วนประกอบของสิ่งมีชีวิตมากที่สุด (สองในสามส่วน) ซึ่งนิวเคลียสของไฮโดรเจนประกอบด้วยโปรตอนซึ่งมีประจุไฟฟ้าบวกเพียงอนุภาคเดียว โดยคุณสมบัติสองประการที่ทำให้นิวเคลียสของไฮโดรเจนมีสภาพเป็นแม่เหล็กคือ

1) อนุภาคโปรตอนไม่ได้บรรจุอยู่เต็มปริมาตรของนิวเคลียส นิวเคลียสจึงมีที่ว่างทำให้โปรตอนสามารถเคลื่อนที่ได้

2) สปิน (spin) อนุภาคโปรตอนวิ่งวนเป็นวงกลมอยู่รอบแกนของนิวเคลียส ทำให้เกิดเป็นสนามแม่เหล็กขึ้น ในขณะเดียวกันแกนของนิวเคลียสของไฮโดรเจนไม่ได้อยู่นิ่ง แต่จะมีการหมุนควง (precession) อยู่ตลอดเวลา ความถี่ของการหมุนควงของแกนนิวเคลียสของไฮโดรเจน เรียกว่า **ความถี่ลาร์มอร์** (Larmor frequency) ซึ่งก็คือความถี่ธรรมชาติและความถี่เรโซแนนซ์นั่นเอง

การที่นิวเคลียสของอะตอมสามารถแสดงคุณสมบัติทางแม่เหล็กได้หรือไม่นั้นสามารถแสดงได้ดังรูปที่ 2.2



รูปที่ 2.2 นิวเคลียสประกอบด้วยนิวตรอนและโปรตอน ถ้านิวเคลียสใดมีโปรตอนหรือนิวตรอนที่ไม่มีคู่ หรือมีทั้งโปรตอนและนิวตรอนที่ไม่มีคู่ นิวเคลียสนั้นก็จะเป็นนิวเคลียสแม่เหล็ก

ภาพแสดงนิวเคลียส 5 นิวเคลียส สีดำคือโปรตอน ส่วนสีขาวคือนิวตรอน โปรตอนหรือนิวตรอนที่เป็นคู่จะมีสปินกลับทางกันและหักล้างกันหมด (ตัวอย่าง เช่น ฮีเลียม 4) นิวเคลียสที่เป็นแม่เหล็กต้องมีโปรตอนที่ไม่มีคู่ (ไอโซโทปของไฮโดรเจนทั้งสามชนิด) หรือนิวตรอนที่ไม่มีคู่ (เช่น ฮีเลียม 3) หรือมีทั้งโปรตอนและนิวตรอนที่ไม่มีคู่ (ดิวทีเรียม)

ขบวนการกระตุ้น : เรากระตุ้นนิวเคลียสของไฮโดรเจนโดยใช้ **คลื่นวิทยุ** เดินทางด้วยความเร็วเท่ากับแสง แต่มีคลื่นความถี่ต่างๆ กัน การกระตุ้นนิวเคลียสให้ได้ผลเต็มที่ ต้องปรับความถี่ของคลื่นวิทยุให้ตรงกับความถี่เรโซแนนซ์ของนิวเคลียสไฮโดรเจนที่กำลังของสนามแม่เหล็กขณะนั้น

การที่นิวเคลียสแม่เหล็กดูดกลืนพลังงานจากคลื่นวิทยุก็คือปรากฏการณ์นิวเคลียสแมกเนติกเรโซแนนซ์นั่นเอง

การตรวจหาสัญญาณ

หลังจากถูกกระตุ้นด้วยคลื่นวิทยุที่เหมาะสม นิวเคลียสของไฮโดรเจนจะดูดกลืนพลังงานจากคลื่นวิทยุไว้ เมื่อหยุดกระตุ้น นิวเคลียสของไฮโดรเจนก็จะปล่อยพลังงานกลับคืนออกมาเป็นคลื่นวิทยุ ซึ่งสามารถตรวจจับได้ด้วยเสาอากาศรับคลื่นสั้นและเครื่องวิทยุ ถึงแม้ว่าสัญญาณจากนิวเคลียสแต่ละนิวเคลียสจะอ่อนมากจนไม่สามารถตรวจจับได้ แต่สัญญาณจากนิวเคลียสหลายๆ ล้านนิวเคลียสรวมกันนั้นแรงพอที่สามารถตรวจจับได้ โดยเราจะใช้ความสัมพันธ์ระหว่างกำลังของสนามแม่เหล็กและความถี่เรโซแนนซ์ในการหาตำแหน่งของนิวเคลียสของไฮโดรเจน

การหาตำแหน่งของนิวเคลียสของไฮโดรเจนภายในร่างการคล้ายกับการหาตำแหน่งของเข็มทิศในสนามแม่เหล็กโลก ภายใต้สนามแม่เหล็กที่มีการค่อยๆลดหรือเพิ่มกำลังอย่างสม่ำเสมอ (gradient magnetic field) ตำแหน่งของนิวเคลียสของไฮโดรเจนสามารถคำนวณได้จากความถี่เรโซ

แนนซ์ สนามแม่เหล็กโลกมีการลดหรือเพิ่มระดับกำลังโดยธรรมชาติ สนามแม่เหล็กเกรเดียนต์ลักษณะเดียวกับสนามแม่เหล็กโลกซึ่งจำเป็นสำหรับการสร้างภาพด้วยเครื่อง MRI

2.3 การสร้างภาพ

ถ้าเอาศีรษะไปแช่แข็ง แล้วใช้เลื่อยตัดศีรษะตามขวางออกเป็นแผ่น แต่ละแผ่นหนา 1ซม. แล้วนำไปตัดแบ่งออกเป็นแท่งเล็กๆ ปริมาตร $1\times1\times1$ mm^3 เรียกว่า **โวกเซล (voxel)** แต่ละชิ้นรู้ตำแหน่งแน่นอน แล้วนำเนื้อเยื่อชิ้นเล็กๆ เหล่านั้นมาวิเคราะห์หาปริมาณไฮโดรเจน

ในหัวข้อนี้จะอธิบายว่าเครื่อง MRI หาตำแหน่งของเนื้อเยื่อแต่ละโวกเซลได้อย่างไร และนำมารวมกันสร้างภาพตัดขวางได้อย่างไร

คุณสมบัติของโปรตอน (นิวเคลียสของไฮโดรเจน) ในขบวนการสร้างภาพของเครื่อง MRI คือ

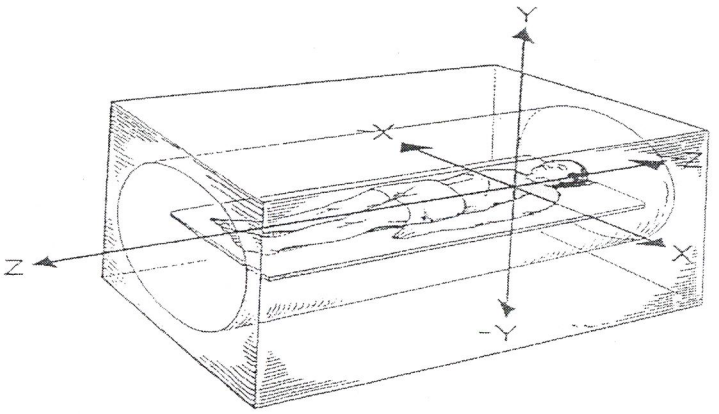
1) โปรตอนปรับแนวตามสนามแม่เหล็กกำลังสูงของเครื่อง MRI
2) โปรตอนจะเรโซแนนซ์กับความถี่ที่เหมาะสม (ความถี่ลาร์มอร์) โดยขึ้นกับกำลังของสนามแม่เหล็ก
3) ในสนามแม่เหล็กเกรเดียนต์ความถี่เรโซแนนซ์ของโปรตอนทำให้บอกได้ว่าโปรตอนอยู่ที่ตำแหน่งใดในสนามแม่เหล็กเกรเดียนต์

การสร้างภาพเนื้อเยื่อตามระนาบตัดขวาง

ขั้นตอนแรกสุดในการสร้างภาพตามระนาบตัดขวาง คือการแยกเนื้อเยื่อบริเวณที่จะตรวจออกจากเนื้อเยื่อบริเวณใกล้เคียง วิธีที่ใช้กันมากที่สุดคือ เลือกกระตุ้นเนื้อเยื่อเฉพาะบริเวณที่ต้องการโดยอาศัยสนามแม่เหล็กเกรเดียนต์

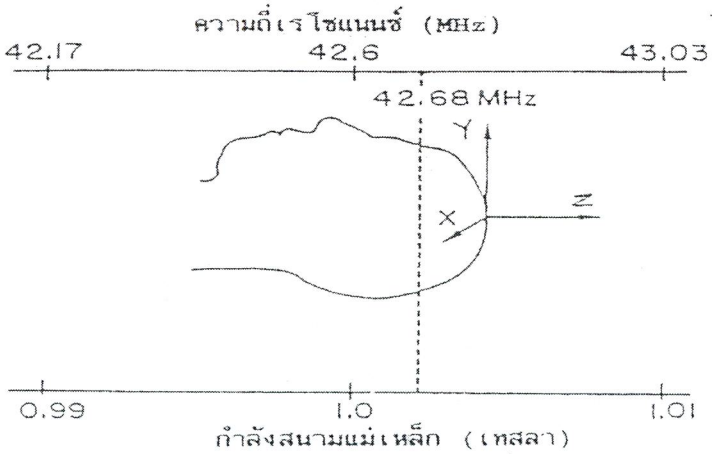
การแยกเนื้อเยื่อตามระนาบตัดขวาง

การแยกเนื้อเยื่อบริเวณที่จะตรวจออกจากเนื้อเยื่อบริเวณใกล้เคียง เครื่องจะสร้างสนามแม่เหล็กเกรเดียนต์ที่มีกำลังแม่เหล็กลดหลั่นตามแนวยาวของร่างกายหรือจากศีรษะไปปลายเท้า แนวนี้กำหนดเป็นแกน Z (ดังรูปที่ 2.3) สนามแม่เหล็กกำลังสูงอยู่ทางศีรษะและกำลังต่ำกว่าอยู่ทางปลายเท้า สนามแม่เหล็กเกรเดียนต์เป็นสนามแม่เหล็กกำลังต่ำที่ทำให้มีการเปลี่ยนแปลงกำลังของสนามแม่เหล็กหลักเพียงประมาณ 1 เกาส์ต่อหนึ่งซม. ถ้าสนามแม่เหล็กมีกำลัง 1 เทสลา ก็จะมีการเปลี่ยนแปลงกำลังของสนามแม่เหล็กเพียง 0.01% ต่อซม. ความแตกต่างเพียงเล็กน้อยนี้สามารถทำให้เลือกกระตุ้นเนื้อเยื่อที่มีความหนาเพียงไม่กี่มล.โดยไม่รบกวนเนื้อเยื่อส่วนอื่น สนามแม่เหล็กเกรเดียนต์จะถูกสร้างขึ้นเฉพาะบริเวณที่ทำการตรวจเท่านั้น



รูปที่ 2.3 แสดงความสัมพันธ์ระหว่างทิศทางของสนามแม่เหล็กหลักกับการกำหนดระบบพิกัด(coordinate system) ในเครื่อง MRI แม่เหล็กไฟฟ้าที่ทิศทางของสนามแม่เหล็กหลักอยู่ตามความยาวของร่างกายซึ่งกำหนดให้เป็นแกน Z

โปรตอนภายในร่างกายจะมีความถี่เรโซแนนท์ในสนามแม่เหล็กเกรเดียนต์ ไฮโดรเจนที่อยู่ ในตำแหน่งสูงขึ้นไปทางศีรษะ จะอยู่ภายใต้สนามแม่เหล็กที่มีกำลังสูงขึ้นเรื่อยๆ จึงเกิดเรโซแนนซ์ที่ ความถี่สูงขึ้นเรื่อยๆ ดังรูปที่ 2.4 จะมีเนื้อเยื่ออยู่แถบเดียวเท่านั้นที่อยู่ในสนามแม่เหล็กที่มีกำลัง 1 เทสลาพอดี ซึ่งหมายถึง จะมีโปรตอนบริเวณนี้เท่านั้นที่ดูดกลืนพลังงานจากคลื่นวิทยุหรือถูกกระตุ้น ถ้าต้องการกระตุ้นเนื้อเยื่อบริเวณอื่นก็จะสามารถทำได้โดยการปรับคลื่นวิทยุกระตุ้นให้มี ความถี่สูงขึ้นหรือต่ำลง หรืออาจทำได้โดยการเลื่อนสนามแม่เหล็กเกรเดียนต์



รูปที่ 2.4 แสดงส่วนศีรษะซึ่งอยู่ในสนามแม่เหล็กเกรเดียนต์ตามยาวซึ่งมีกำลังต่ำทางด้านซ้ายและกำลังจะสูงขึ้นเรื่อยๆ ไปทางขวา ความถี่เรโซแนนท์ของโปรตอนจะแตกต่างกันโดยขึ้นกับตำแหน่งในสนามแม่เหล็กเกรเดียนต์

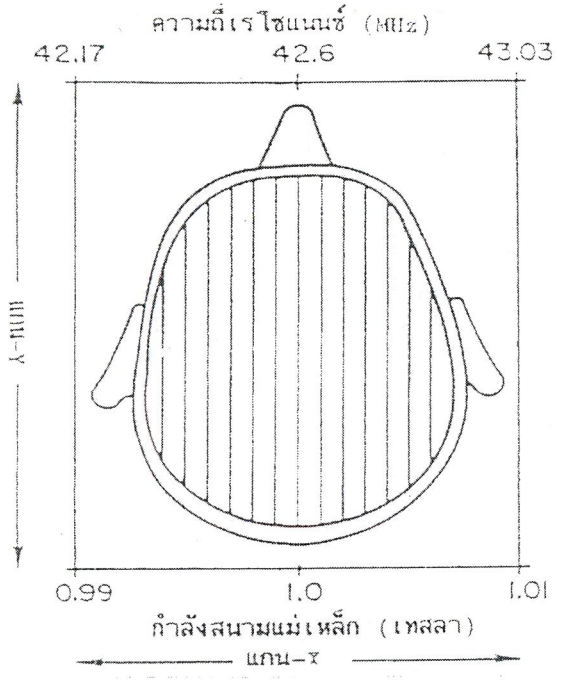
การหาตำแหน่งสัญญาณภายในเนื้อเยื่อหนึ่งแผ่น

หลังจากแยกเนื้อเยื่อบริเวณที่สนใจออกมาได้แล้ว ซึ่งหลังจากการกระตุ้น โปรตอนจะปล่อยคลื่นวิทยุออกมาทันที ซึ่งสัญญาณที่ได้นี้สามารถนำไปสร้างภาพได้แล้ว แต่ความเข้มของสัญญาณจะบอกเพียงค่าเฉลี่ยของโปรตอนจากทุกบริเวณของเนื้อเยื่อแผ่นนั้นเท่านั้น

เพื่อจะบอกให้ได้ว่าสัญญาณที่ได้มาจากส่วนใดของเนื้อเยื่อที่ถูกกระตุ้น จะต้องใช้สนามแม่เหล็กเกรเดียนต์ในทิศทางใหม่ ซึ่งจะให้หลังจากที่คลื่นวิทยุกระตุ้นเริ่มแรกถูกปิด (เนื้อเยื่อจะปล่อยสัญญาณออกมา) ซึ่งสนามแม่เหล็กเกรเดียนต์สนามที่สองจะถูกสร้างขึ้นตามระนาบ XY ทำให้สนามแม่เหล็กด้านหนึ่งของแผ่นเนื้อเยื่อมีกำลังมากกว่าอีกด้านหนึ่ง (ดังรูปที่ 2.5) ด้วยอิทธิพลของสนามแม่เหล็กเกรเดียนต์ใหม่ จะทำให้นิวเคลียสซึ่งเดิมเคยหมุนควงด้วยความถี่เดียวกันจะปรับตัวตามสนามแม่เหล็กเกรเดียนต์ใหม่และเรโซแนนท์ด้วยความถี่ที่สูงขึ้นหรือต่ำลง ขึ้นกับตำแหน่งในสนามแม่เหล็กเกรเดียนต์ใหม่ นิวเคลียสทางด้านหนึ่งของแผ่นเนื้อเยื่อซึ่งมีกำลังสนามแม่เหล็กสูงกว่าจะเรโซแนนท์ด้วยความถี่สูงกว่าอีกด้านหนึ่ง

สนามแม่เหล็กเกรเดียนต์ตามขวางจะถูกเปิดทันทีหลังจากที่ปิดสนามแม่เหล็กเกรเดียนต์ตามยาวในขณะที่โปรตอนยังหมุนควงอยู่จากการกระตุ้นครั้งแรก โปรตอนจะปรับความถี่เรโซแนนท์ตามสนามแม่เหล็กเกรเดียนต์ใหม่โดยไม่ต้องกระตุ้นซ้ำ ด้วยวิธีนี้เนื้อเยื่อก็จะปล่อยสัญญาณคลื่นวิทยุขนาดความถี่ต่างๆ กันออกมาโดยขึ้นกับตำแหน่งในสนามแม่เหล็กเกรเดียนต์ใหม่ เปรียบเหมือนการฝานเนื้อเยื่อแผ่นเดิมออกเป็นแถบเล็กๆ เนื้อเยื่อแต่ละแถบจะปล่อยคลื่นวิทยุความถี่หนึ่งออกมา กำลังของสัญญาณในแต่ละความถี่จะบอกเราว่า เนื้อเยื่อหนึ่งแถบมีปริมาณไฮโดรเจนมากหรือน้อย เหมือนคีย์ของเปียโน ถ้าทราบความถี่ของเสียเปียโนก็จะสามารถหาได้ว่าเสียงนั้นได้มาจากการกดคีย์ตำแหน่งใด

โปรตอนจะเปลี่ยนความถี่ตามกำลังของสนามแม่เหล็กใหม่ไม่ว่าความถี่เดิมหลังการกระตุ้นจะเป็นเท่าใด เปรียบเหมือนการผ่อนสายกีต้าร์ภายหลังการดีดสายซึ่งจะทำให้เสียงกีต้าร์เปลี่ยนไป นั่นคือ สัญญาณจากเนื้อเยื่อหนึ่งแผ่นจึงประกอบด้วยหลายความถี่ ถ้าวัดกำลังของคลื่นวิทยุในแต่ละความถี่ก็จะได้ปริมาณค่าของไฮโดรเจนในเนื้อเยื่อแต่ละแถบ และในเครื่อง MRI จะมีเครื่องวิเคราะห์ความถี่สำหรับวิเคราะห์สัญญาณวิทยุจากเนื้อเยื่อว่ามีความถี่ใดบ้าง แต่ละความถี่มีกำลังของสัญญาณมากหรือน้อย(ดูจากแอมปลิจูด) ขนาดของความถี่จะบอกว่าสัญญาณมาจากเนื้อเยื่อแถบใด ส่วนแอมปลิจูดของแต่ละความถี่จะบอกปริมาณไฮโดรเจนในเนื้อเยื่อแต่ละแถบ

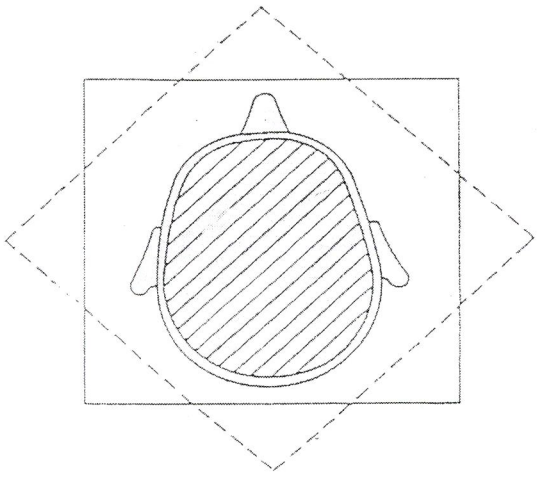


รูปที่ 2.5 เนื้อเยื่อ slice สัมผัสกับสนามแม่เหล็กเกรเดียนต์ตามขวาง (แกน X) ตัวเลขด้านล่างแสดงกำลังสนามแม่เหล็ก ตัวเลขด้านบนแสดงความถี่ของไฮโดรเจน เนื้อเยื่อแต่ละแถบในภาพจะมีความถี่เรโซแนนท์เท่ากัน

การหมุนสนามแม่เหล็กเกรเดียนต์ในระนาบขวาง

หลังจากรับสัญญาณและเก็บข้อมูลแล้ว ขบวนการทั้งหมดจะถูกทำซ้ำ เริ่มจากการสร้างสนามแม่เหล็กเกรเดียนต์ตามแกน Z กระตุ้นเนื้อเยื่อบริเวณเดิมด้วยคลื่นวิทยุ ปิดสนามแม่เหล็กเกรเดียนต์ตามแกน Z แล้วเปิดสนามแม่เหล็กเกรเดียนต์ตามระนาบตัดขวาง (ระนาบXY) ทันที แต่ครั้งนี้ทิศทางต่างจากครั้งแรก เท่ากับทำการวัดความหนาแน่นของไฮโดรเจนในเนื้อเยื่อแผ่นเดิมในทิศทางที่ต่างจากครั้งแรก(ดังรูปที่ 2.6)

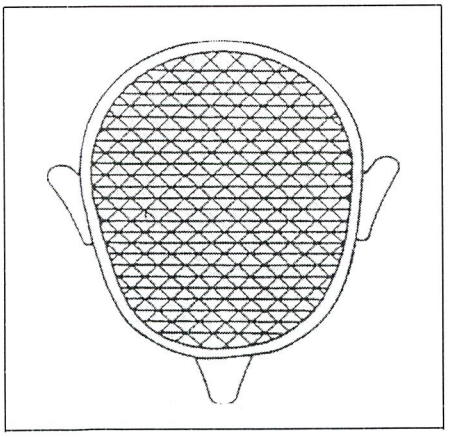
ในการสร้างภาพตามระนาบตัดขวางหนึ่งภาพ ขบวนการข้างต้นจะถูกทำซ้ำแล้วซ้ำอีกโดยเปลี่ยนทิศทางของสนามแม่เหล็กเกรเดียนต์ตามระนาบตัดขวางไปเรื่อยๆ เมื่อจบขบวนการรวบรวมข้อมูล เนื้อเยื่อตามระนาบตัดขวางหนึ่งแผ่นจะถูกตัดออกเป็นแถบเล็กๆ ในทิศทางต่างกันมากมาย



รูปที่ 2.6 เนื้อเยื่อหนึ่ง slice จะถูกตรวจหลายครั้ง เพื่อรวบรวมข้อมูลให้เพียงพอในการสร้างภาพโดยการเปลี่ยนทิศทางของสนามแม่เหล็กเกรเดียนต์ตามระนาบ XY การตรวจแต่ละครั้งจะได้ข้อมูลในแถบเนื้อเยื่อเล็กชุดใหม่ ในภาพจะแสดงเกรเดียนต์เพียงสองทิศทาง(เส้นทึบกับเส้นประ)

การสร้างภาพด้วยคอมพิวเตอร์

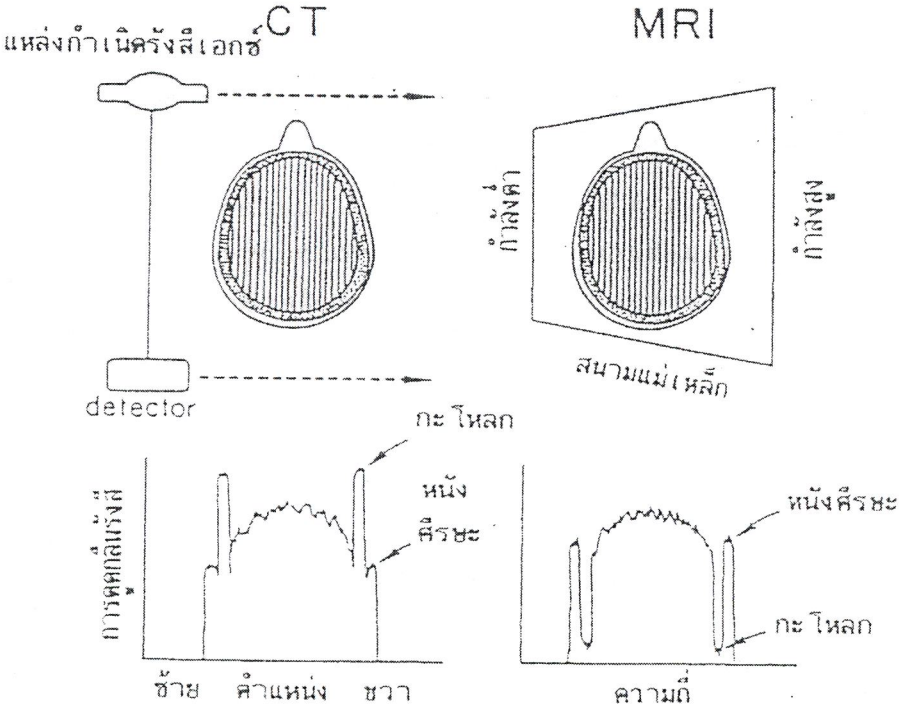
ถึงจุดนี้ เนื้อเยื่อตามระนาบตัดขวางหนึ่งแผ่นถูกตัดออกเป็นแถบหลายร้อยชุด ต่อไปเป็นการแบ่งเนื้อเยื่อแต่ละแถบออกเป็นโวกเซลแล้ววัดปริมาณของไฮโดรเจนซึ่งจะถูกเก็บไว้ในลักษณะเป็นแถบความสว่างต่างๆกัน ความสว่างขึ้นกับปริมาณไฮโดรเจนในเนื้อเยื่อแถบนั้นในการตรวจเนื้อเยื่อระนาบขวางหนึ่งแผ่นจะมีแถบแสงลักษณะดังกล่าวหลายพันแถบเก็บไว้ ดังรูป 2.7



รูปที่ 2.7 ระหว่างการตรวจ สนามแม่เหล็กเกรเดียนต์ตามระนาบ XY จะถูกเปลี่ยนมุมไปครั้งละไม่กี่องศา ภาพจะสร้างจากเส้นซึ่งตัดผ่านกันหลายพันเส้น

ถ้าเอาแถบแสงทั้งหมดมาซ้อนกันน่าจะได้ภาพขึ้นมา แต่การเอาแถบแสงมาซ้อนกันเฉยๆ (simple back projection) ภาพที่ได้จะพร่ามัวจนไม่มีประโยชน์ในการวินิจฉัย ต้องมีการแปลงข้อมูลด้วยวิธีการทางคณิตศาสตร์ชั้นสูงเรียกว่า Fourier Transform จึงจะได้ภาพที่ชัดเจน วิธีการซ้อนแถบแสงภายหลังการทำ Fourier Transform เรียกว่า Filtered back projection ซึ่งเป็นวิธี การเดียวกับการสร้างภาพในเครื่อง CT

เครื่อง CT วัดความหนาแน่นของเนื้อเยื่อในแถบแคบๆ ระหว่างหลอดรังสี detector หลังจากหลอดรังสีกับ detector ทำการวัดตลอดความกว้างของศีรษะก็จะมีการเปลี่ยนมุมของหลอดรังสีกับ detector แล้วทำการวัดเนื้อเยื่อในแถบแคบๆ ขนานกันชุดใหม่ในทิศทางทำมุมกับชุดแรกเล็กน้อย ข้อมูลที่ได้จากการวัดจะถูกเก็บไว้ในหน่วยความจำในลักษณะเป็นแถบแสงซึ่งมีความสว่างแตกต่างกันขึ้นกับปริมาณรังสีที่หายไปในเนื้อเยื่อแถบนั้น



รูปที่ 2.8 CT และ MRI ใช้วิธี การสร้างภาพแบบเดียวกัน MRI วัดปริมาณไฮโดรเจนในเนื้อเยื่อแต่ละแถบ ส่วน CT วัดการดูดกลืนรังสี ความแตกต่างระหว่าง MRI กับ CT แสดงในภาพโปรไฟล์ข้างล่าง ในภาพ CT กระดูกซึ่งดูดกลืนรังสีมากจะเห็นเป็นสีขาว ในภาพ MRI กระดูกมีสัญญาณต่ำจะเห็นเป็นสีดำ

บทที่ 3

Image Processing

หลักการพื้นฐานทางอิมเมจโปรเซสซิ่ง

อิมเมจโปรเซสซิ่ง **(image processing)** เป็นกระบวนการที่ใช้ในการจัดการข้อมูลที่เป็นรูปภาพต่างๆ ให้อยู่ในลักษณะของสัญญาณไฟฟ้าเพื่อได้นำข้อมูลที่เป็นสัญญาณไฟฟ้านั้นไปใช้ประโยชน์ต่อไป เช่นการตกแต่งภาพ , การส่งภาพไปตามสายสัญญาณจากที่หนึ่งไปยังอีกที่ หนึ่ง , การเก็บข้อมูลรูปภาพไว้ในหน่วยความจำเพื่อใช้เป็นแฟ้มข้อมูลต่างๆนอกเหนือไปจากนี้ยังสามารถนำไปใช้งานด้านการรักษาความปลอดภัย, การตรวจลายนิ้วมือ เป็นต้น

3.1 จุดภาพ หรือ พิกเซล (pixel)

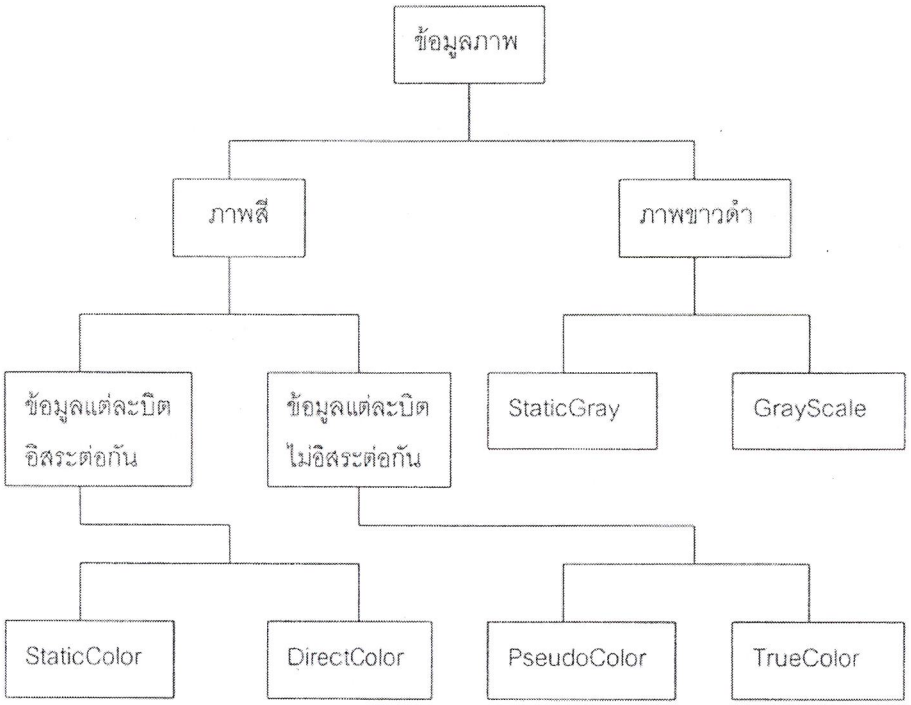
ภาพที่ถูกแปลงเป็นสัญญาณไฟฟ้าได้รับการแบ่งรายละเอียดของภาพเป็นตารางเล็กๆ ซึ่งเรียกว่า พิกเซล (pixel) และเมื่อมีการจัดเรียงพิกเซลในรูปแบบของเมตทริกซ์ นั่นคือจัดเรียงพิกเซลเป็น แถว(row) และหลัก (column) ทำให้เกิดเป็นภาพได้

3.2 เกรย์สเกล (gray scale)

เกรย์สเกล หมายถึง ความแตกต่างของระดับความเข้มของแสง โดยเกรย์สเกลหนึ่งๆ อาจแบ่งเป็น 13,20 หรือ 100 ระดับ โดยระดับที่ว่านี้ก็คือ ระดับเทา (gray level) ซึ่งในภาพหนึ่งๆ ถ้าต้องการแบ่งระดับความเข้มแสงหรือระดับเทาให้มีหลายๆ ระดับนั้น เราสามารถทำได้โดยการเพิ่มจำนวนบิตในการเก็บข้อมูลภาพ เช่น หากต้องการภาพที่มีระดับเทา 16 ระดับ ต้องทำการเก็บข้อมูลภาพเป็นเลขฐานสอง จำนวน 4 บิต หรือหากต้องการให้ภาพมีระดับเทา 256 ระดับ ต้องเก็บข้อมูลภาพเป็นเลขฐานสองจำนวน 8 บิต เป็นต้น

3.3 ลักษณะการเก็บข้อมูลภาพ

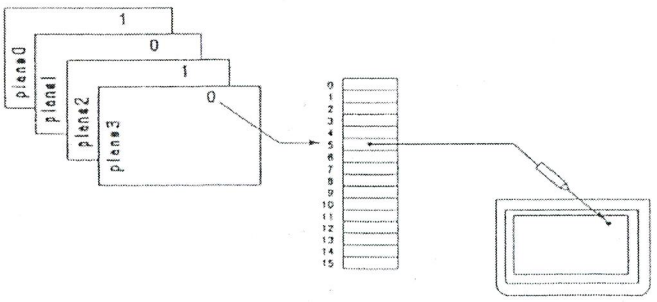
เนื่องจากการประมวลผลทางอิมเมจโปรเซสซิ่งเป็นการนำข้อมูลภาพมาทำการประมวลผลในรูปแบบต่างๆ ดังนั้นจึงมีความจำเป็นอย่างยิ่งที่เราต้องรู้ถึงการกเก็บข้อมูลภาพ ซึ่งข้อมูลภาพโดยทั่วไปมีการจัดเก็บในหลายๆ รูปแบบดังนี้



รูปที่ 3.1 แสดงการจัดเก็บข้อมูลภาพในรูปแบบต่างๆ

นั่นคือเราสามารถแบ่งรูปแบบในการจัดเก็บข้อมูลภาพได้ดังนี้

3.3.1) ภาพขาวดำ เป็นภาพที่มีการเก็บข้อมูลภาพเป็นแบบ grayscale นั่นคือ ในแต่ละ pixel มีข้อมูลอยู่ 1 byte ซึ่งมีขนาดกี่บิตก็ขึ้นอยู่กับจำนวนของระดับเทาที่ต้องการ



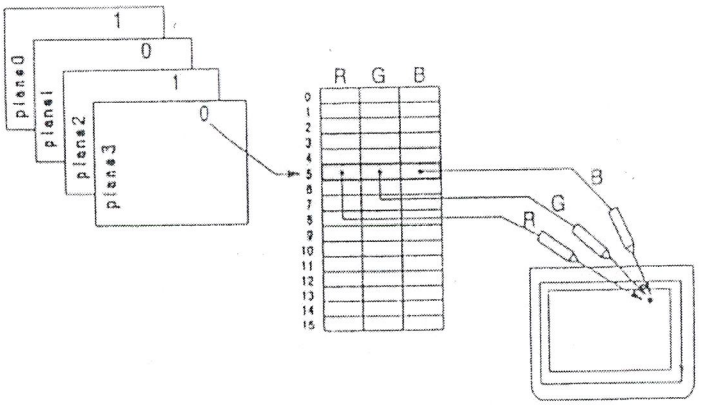
รูปที่ 3.2 การแสดงภาพในรูปแบบของ GrayScale/StaticGray

จากรูปที่ 3.2 สามารถอธิบายถึงการแสดงภาพขาวดำได้ดังนี้ คือ นำข้อมูลภาพที่เก็บไว้แล้วในหน่วยความจำมาเป็นตัวชี้ถึงตำแหน่งของระดับสีที่มีข้อมูลอยู่แล้ว จากรูปได้ว่า ข้อมูลภาพที่มีค่าเป็นเลขฐานสองจำนวน 4 บิต (1010 มีระดับสีเท่ากับ16 ระดับ) คือมีค่าเท่ากับ 5 และเรานำค่านี้ไปเป็นตำแหน่งของระดับสีที่ได้มีการกำหนดไว้แล้ว โดยรูปแบบของการจัดเก็บระดับสีที่ได้กำหนดไว้นี้มีอยู่ 2 รูปแบบคือ

- GrayScale คือ เราสามารถทำการเปลี่ยนแปลงค่าระดับสีของแต่ละตำแหน่งได้
- StaticGray คือ เราไม่สามารถทำการเปลี่ยนแปลงค่าระดับสีของแต่ละตำแหน่งได้

3.1.2) ภาพสี เนื่องจากข้อมูลของภาพสีถูกจัดเก็บอยู่ในรูป RGB บิต คือในแต่ละ pixel มีข้อมูลอยู่ 1 byte ซึ่งแต่ละ byte ประกอบด้วย RGB บิต และการแสดงภาพของภาพสีมีวิธีดังนี้ คือ

1) ตำแหน่งของ RGB บิตที่ถูกชี้โดยข้อมูลภาพ อยู่ตำแหน่งเดียวกัน สามารถแสดงการแสดงภาพในลักษณะนี้ได้ดังรูป

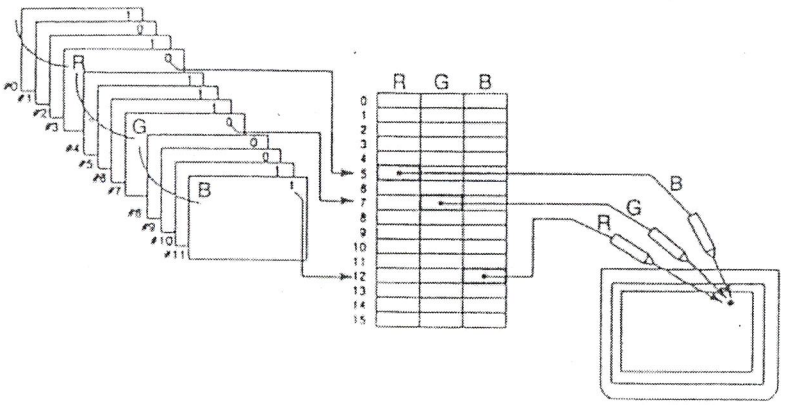


รูปที่ 3.3 การแสดงภาพในรูปแบบของ PseudoColor/StaticColor

จากรูปที่ 3.3 เห็นว่า ณ ตำแหน่งที่ข้อมูลภาพชี้อยู่เป็นตำแหน่งที่ GRB บิต อยู่ในตำแหน่งเดียวกัน และเช่นเดียวกัน รูปแบบของการจัดเก็บระดับสีที่ได้กำหนดไวนี้มีอยู่ 2 รูปแบบ คือ

- PseudoColor คือ เราสามารถทำการเปลี่ยนแปลงค่าระดับสีของแต่ละตำแหน่งได้
- StaticColor คือ เราไม่สามารถทำการเปลี่ยนแปลงค่าระดับสีของแต่ละตำแหน่งได้

2) ตำแหน่งของ RGB บิตที่ถูกชี้โดยข้อมูลภาพอยู่คนละตำแหน่งกัน สามารถแสดงการแสดงภาพในลักษณะนี้ได้ดังรูป



รูปที่ 3.4 การแสดงภาพในรูปแบบของ DirectColor/TrueColor

จากรูปที่ 3.4 เห็นว่าในแต่ละ pixel มีข้อมูลอยู่12 บิตนั่นคือ ในการชี้ตำแหน่งของค่า RGB บิต เป็นการแยกข้อมูลออกเป็นกลุ่มๆละ 4 บิตเพื่อแยกชี้ตำแหน่งให้แก่ค่า R , G และ B บิตนั่นก็คือตำแหน่งของแต่ละบิตเป็นอิสระต่อกัน ดังนั้นเห็นว่าการเก็บข้อมูลในรูปแบบนี้สามารถให้ระดับของสีที่มากกว่าแบบที่ผ่านๆมา รูปแบบของการจัดเก็บระดับสีที่ได้กำหนดไว้นี้มีอยู่ 2 รูปแบบคือ

- DirectColor คือ เราสามารถทำการเปลี่ยนแปลงค่าระดับสีของแต่ละตำแหน่งได้
- TrueColor คือ เราไม่สามารถทำการเปลี่ยนแปลงค่าระดับสีของแต่ละตำแหน่งได้

ภาพที่นำมาใช้ในปริญญานิพนธ์นี้ เป็นภาพในลักษณะ True Color คือเป็นแบบ 24 บิต ซึ่งแต่ละบิตเป็นระดับสีของค่า RGB (Red Green Blue) ซึ่งการแสดงผลทำโดยการส่งค่าสี RGB ผ่านการแปลงจาก ดิจิตอลเป็นอนาลอก (D/A) ทีละ 8 บิต ซึ่งความแตกต่างของภาพสีกับภาพที่เป็นระดับเทา คือ ภาพที่เป็นระดับเทาค่าของข้อมูลใน RGB บิตเท่ากัน ส่วนภาพสีค่าของข้อมูลใน RGB บิตไม่เท่ากัน

3.4 ความหมายและนิยามของภาพในระบบดิจิตอล

ภาพ (image) ในความหมายทางคณิตศาสตร์หมายถึง ฟังก์ชัน 2 มิติ $f(x,y)$ โดย x และ y เป็นแกนพิกัดในระนาบ 2 มิติ ค่าฟังก์ชัน $f(x,y)$ เป็นสัดส่วนกับความสว่างหรือความเข้มของภาพที่ตำแหน่ง x,y ซึ่งเราเรียกว่า ระดับเทา ซึ่งปกติเราให้จุดกำเนิดของแกนพิกัด (coordinate)

อยู่ทางซ้ายของภาพ เนื่องจากแสงเป็นพลังงานรูปหนึ่ง ดังนั้น $f(x,y)$ ต้องไม่เป็นศูนย์ และมีค่าจำกัดนั่นคือ

$$0 < f(x,y) < \alpha \tag{3.1}$$

โดยธรรมชาติของแสง ซึ่งต้องมีแหล่งกำเนิดแสงและส่วนที่สะท้อนของแสง ดังนั้นเราสามารถแยกฟังก์ชัน $f(x,y)$ ออกเป็นสองส่วนคือ อิลูมิเนชันคอมโพเนนท์ (illumination component) และรีเฟล็กแทนท์คอมโพเนนท์ (reflectant component) ได้ว่า

$$f(x,y) = I(x,y) * r(x,y) \tag{3.2}$$

เมื่อ

$$0 < I(x,y) < \alpha \tag{3.3}$$

และ

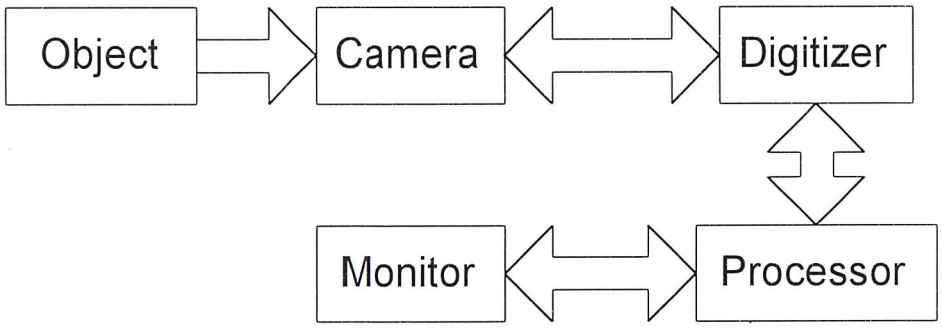
$$0 < r(x,y) < 1 \tag{3.4}$$

จากสมการ 3.4 แสดงให้เห็นว่า ฟังก์ชันการสะท้อนถูกจำกัดขอบเขตระหว่าง 0 (หมายถึงการดูดซึมสมบูรณ์) และ 1 (หมายถึง การสะท้อนโดยสมบูรณ์) ธรรมชาติของ $I(x,y)$ ซึ่งขึ้นอยู่กับแหล่งกำเนิดแสง ในขณะที่ $r(x,y)$ ขึ้นอยู่กับวัตถุที่สะท้อนแสงมาเข้าตา

ระบบการประมวลผลทางดิจิตอล

ระบบการประมวลผลทางดิจิตอล ประกอบไปด้วยส่วนประกอบหลักๆ 3 ส่วนคือ

1. ส่วนแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล ซึ่งเรียกว่า ดิจิไตเซอร์ (Digitizer) เป็นตัวรับข้อมูลภาพแล้วแปลงจากสัญญาณอนาลอกไปเป็นดิจิตอล เช่น กล้องดิจิตอล เป็นต้น
2. ส่วนประมวลผล (Processor) เป็นส่วนที่ใช้ในการประมวลผลและทำการวิเคราะห์ข้อมูลภาพ
3. ส่วนแสดงผล (Display) ทำหน้าที่เปลี่ยนข้อมูลตัวเลข (ระดับเทา) ที่เก็บเป็น อะเรย์ (array) ให้อยู่ในรูปที่เหมาะสม และสื่อความหมายกับมนุษย์ได้ คือเป็นภาพปกติทั่วๆ ไป

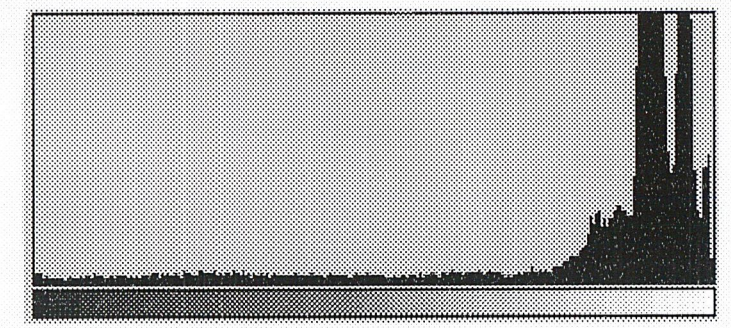


รูปที่ 3.5 แสดงระบบการประมวลผลทางดิจิตอล

3.5 ฮิสโตแกรม (Histogram)

ฮิสโตแกรม เป็นกราฟที่แสดงความสัมพันธ์ระหว่างระดับเทากับจำนวนของจุดภาพในภาพที่ระดับเทาใดๆ โดยรูปร่างและลักษณะของฮิสโตแกรมบ่งบอกได้ถึงคุณสมบัติต่างๆ ของภาพนั้น อย่างเช่นหากภาพนั้นมีฮิสโตแกรมที่กลุ่มของข้อมูลส่วนใหญ่อยู่ทางด้านซ้ายของฮิสโตแกรม บ่งบอกได้ว่าภาพนั้นเป็นภาพที่ค่อนข้างมืด หรือหากกลุ่มของข้อมูลส่วนใหญ่อยู่ทางขวามือของฮิสโตแกรม บ่งบอกได้ว่าภาพนั้นเป็นภาพที่ค่อนข้างสว่าง ดังแสดงได้ดังรูป

รูปที่ 3.6 แสดงภาพที่มีความสว่างของภาพสูง



รูปที่ 3.7 Histogramของภาพในรูปที่ 3.6

จากรูปที่ 3.6 พบว่าฮิสโตแกรมที่ได้ ดังรูปที่ 3.7 มีลักษณะของกลุ่มข้อมูลส่วนใหญ่อยู่ทางขวาของฮิสโตแกรม ซึ่งนั่นก็หมายถึง ภาพที่นำมาเป็นต้นแบบ (รูปที่ 3.6) เป็นภาพที่มีความสว่างสูง นอกจากนี้เรายังนำฮิตโตแกรมมาช่วยในการประมวลผลทางด้าน อิมเมจโปรเซสซิ่งได้อย่างกว้างขวาง เช่น การนำมาฮิสโตแกรมมาใช้ในการกำหนดค่าเทรชโฮล (threshold) เป็นต้น

บทที่ 4

Human Brain Segmentation โดยใช้วิธี Multithreshold

4.1 Single Threshold

พื้นฐานในการแบ่งกลุ่มของระดับเทา

เราเริ่มจากทฤษฎีพื้นฐานที่ใช้ในการแบ่งกลุ่มของระดับเทาในภาพ โดยเริ่มจากการแบ่งกลุ่มของระดับเทาของภาพออกเป็น 2 กลุ่ม นั่นคือ เป็นการแปลงจากภาพที่เป็นระดับเทา (gray level image) ให้เป็น ภาพ 2 ระดับ (bilevel :โมโนโครม หรือ ขาวกับดำ) ซึ่งภาพ 2 ระดับ นี้สามารถบอกเราได้ถึง จำนวน , ตำแหน่ง และรูปร่างของวัตถุต่างๆ ที่อยู่ในภาพ แล้วจากนั้นเราจะนำข้อมูลเหล่านี้ไปใช้ในการแบ่งกลุ่มของระดับเทาในภาพออกเป็นจำนวนกลุ่มมากขึ้นได้ โดยมีสมมุติฐานเบื้องต้นว่า จุดภาพ(pixel) ที่มีระดับเทาใกล้เคียงกัน น่าจะแสดงถึงขอบเขตของวัตถุที่เป็นชนิดเดียวกัน

ทฤษฎีพื้นฐานที่ใช้ในการแปลงจากภาพที่เป็นระดับเทา (gray level) ให้เป็นภาพ 2 ระดับ (bilevel) ทำได้โดยการกำหนดระดับเทาขึ้นมาระดับหนึ่ง และกำหนดให้ทุกจุดภาพในภาพที่ มีค่าระดับเทาต่ำกว่าค่าของระดับเทาที่ตั้งไว้ ให้จุดภาพนั้นเป็นสีดำ (ระดับเทาเป็น 0) และ กำหนดให้ทุกจุดภาพในภาพที่มีค่าระดับเทามากกว่าหรือเท่ากับค่าของระดับเทาที่ตั้งไว้ ให้จุดภาพนั้นเป็นสีขาว (ระดับเทาเป็น 255) ทำให้เราสามารถแปลงจากภาพที่เป็น gray level ให้ เป็น bilevel ได้ และเราเรียกค่าของระดับเทาที่กำหนดขึ้นนี้ว่า ค่า *Threshold T* นั่นคือ

ถ้ากำหนดให้กลุ่มแรกเป็นกลุ่มของจุดภาพที่มีสีดำ เราสามารถกำหนดกลุ่มนี้ได้จาก

$$I(i,j) < T \tag{4.1}$$

และเราสามารถกำหนดกลุ่มของจุดภาพที่อยู่นอกเหนือกลุ่มแรกเป็นกลุ่มของจุดภาพที่ เป็นสีขาวโดย

$$I(i,j) \geq T \tag{4.2}$$

การแปลงภาพที่เป็น gray level ให้เป็น bilevel ในลักษณะนี้เราเรียกว่าการกำหนดค่า single threshold คือเป็นการกำหนดค่าระดับเทาให้เป็นค่า threshold เพียงหนึ่งค่า ซึ่งทำให้ เรา

สามารถแยกภาพในส่วนของวัตถุออกจากส่วนของพื้นหลังได้ และหลังจากนั้นเราสามารถนำทฤษฎีพื้นฐานมาใช้ในการประยุกต์เพื่อให้สามารถแบ่งกลุ่มของระดับเทาออกเป็นหลายๆ กลุ่มต่อไป

ดังนั้น เราจึงต้องเริ่มจากการหาค่าของ threshold ที่เหมาะสมออกมาให้ได้ก่อน ซึ่งค่า threshold ที่เหมาะสมจะต้องเป็นค่าที่ทำให้เราสามารถแยกส่วนที่เป็นวัตถุ และส่วนที่เป็นพื้นหลังของภาพออกจากกันได้อย่างสมบูรณ์ที่สุด วิธีในการกำหนดค่า threshold มีหลายวิธีดังนี้

4.1.1) การกำหนดค่า threshold จากค่าเฉลี่ย (mean) ของระดับเทาทั้งหมดในภาพ

เป็นการนำค่าเฉลี่ยของระดับเทาของจุดภาพทั้งหมดมากำหนดเป็นค่า threshold ซึ่งหมายถึงการกำหนดให้จำนวนจุดภาพในภาพ ครึ่งหนึ่งเป็นสีดำและอีกครึ่งหนึ่งเป็นสีขาว

เราสามารถกำหนดค่า threshold ได้จาก

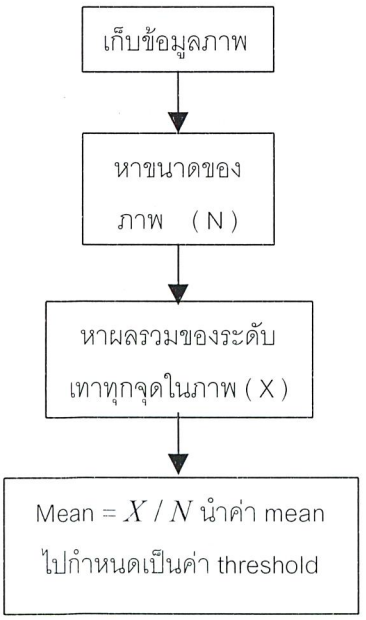
$$T = \frac{\sum_{i=1}^{n} g[i]}{n} \tag{4.3}$$

กำหนดให้ T คือ ค่า threshold

$g[i]$ คือ ค่าของระดับเทาของจุดภาพที่ i

n คือ จำนวนจุดภาพทั้งหมด

แสดงเป็นขั้นตอนในรูปของ flowchart ได้ดังนี้



รูปที่ 4.1 แสดง flowchart ของวิธีในหัวข้อ4.1.1

พบว่าการกำหนดค่า threshold จากค่าเฉลี่ยของระดับเทาแบบนี้ไม่เหมาะสมนัก เนื่องจากมีภาพจำนวนน้อยที่จะมีจำนวนจุดภาพที่เป็นสีดำอยู่ครึ่งหนึ่ง หรือก็คือมีจำนวนจุดที่เป็นส่วนของวัตถุเท่ากับจำนวนจุดที่เป็นส่วนของพื้นหลัง

4.1.2) การกำหนดค่า threshold โดยการกำหนดเปอร์เซ็นต์ของจุดภาพที่เป็นสี ดำ

ในการกำหนดค่า threshold โดยการกำหนดให้จำนวนจุดภาพที่เป็นสีขาว เท่ากับจำนวนจุดภาพที่เป็นสีดำ (วิธีในหัวข้อ 4.1.1) ไม่เหมาะสมนักเพราะมีภาพจำนวนน้อยที่จะมีจำนวนจุดภาพสีดำเป็นครึ่งหนึ่ง แต่ส่วนใหญ่แล้วจำนวนจุดภาพที่เป็นสีขาวกับสีดำเป็นอัตราส่วนกัน ดังนั้นเราสามารถหาค่า threshold ได้โดยการกำหนดเปอร์เซ็นต์ของจำนวนจุดภาพที่เป็นสีดำตามความเหมาะสม ขั้นตอนในการกำหนดค่า threshold โดยวิธีนี้ทำได้ดังนี้

ขั้นที่ 1 หาจำนวนจุดภาพที่เป็นสีดำ ตามเปอร์เซ็นต์ที่ต้องการ

$$M = N * (per) \tag{4.4}$$

เมื่อ M คือ จำนวนจุดภาพที่ต้องการให้เป็นสีดำ

N คือ จำนวนจุดภาพทั้งหมด

per คือ เปอร์เซ็นต์ของจุดภาพสีดำในภาพ

ขั้นที่ 2 หาค่าของจำนวนจุดภาพสะสมตั้งแต่ระดับเทาที่ 0 จนกระทั่งถึงระดับเทาที่ทำให้ผลรวมของจำนวนจุดภาพสะสม มากกว่าหรือเท่ากับ จำนวนจุดภาพที่ต้องการให้เป็นสีดำ

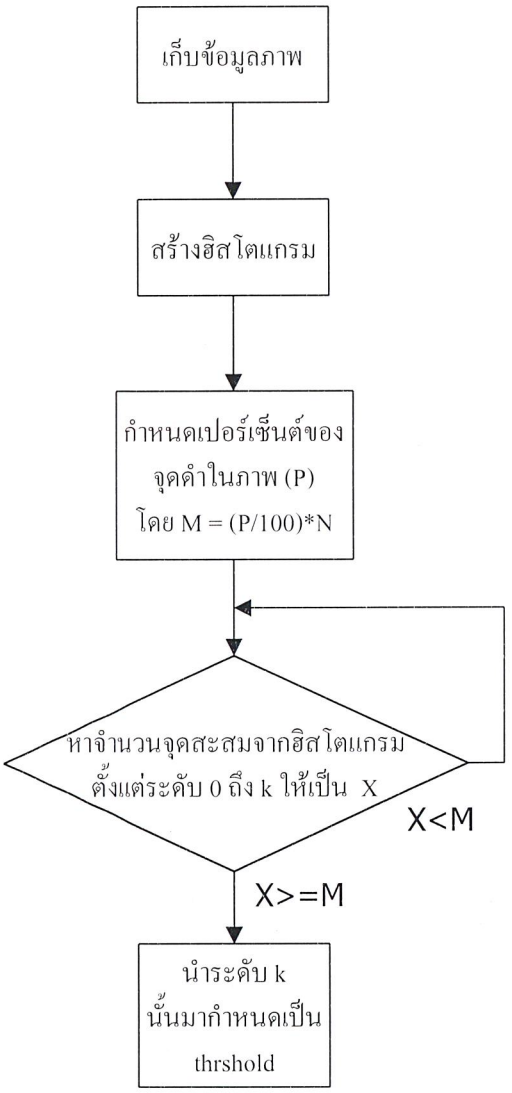
$$\text{if} \quad \sum_{i=0}^{255} h[i] \geq M \tag{4.5}$$

$$\text{then} \quad T = i$$

เมื่อ $h[i]$ คือจำนวนจุดภาพที่มีระดับเทาเป็น i

นั่นคือ ที่ระดับเทาที่ทำให้เงื่อนไขเป็นจริง เราจะนำค่าที่ระดับเทานั้น มาเป็นค่า threshold

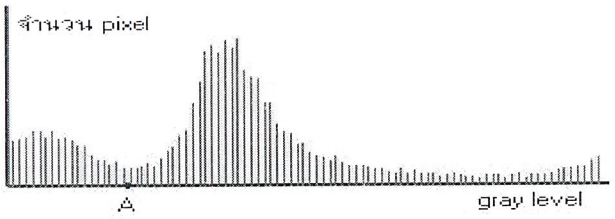
แสดงขั้นตอนในรูปของ flowchart ได้ดังนี้



รูปที่ 4.2 แสดง flowchart ของวิธีในหัวข้อ 4.1.2

4.1.3) การกำหนดค่า Threshold โดยใช้จุดต่ำสุดของจุดสูงสุด 2 จุดใน ฮิสโตแกรม (Twopeaks Method)

ฮิสโตแกรม (Histogram) คือ กราฟที่แสดงความสัมพันธ์ระหว่าง จำนวนของจุดภาพที่ระดับเทาต่างๆ นั่นคือ จะบอกถึงว่า ที่ระดับเทาต่างๆ มีจำนวนจุดภาพกี่จุดในภาพที่มีระดับเท่านั้น โดยการกำหนดค่าของ threshold โดยวิธีนี้ใช้ได้ดีกับภาพที่มีลักษณะของฮิสโตแกรมเป็นดังรูป



รูปที่ 4.3 แสดงฮิสโตแกรมที่มีลักษณะเป็น twopeaks

นั่นคือ ภาพที่ให้ฮิสโตแกรมเป็นลักษณะที่มีกลุ่มของข้อมูลที่สูงสุดอยู่ 2 กลุ่ม เราสามารถกำหนดให้ระดับเทาที่ต่ำสุดในช่วงของจุดยอดทั้งสองนั้นเป็นค่า Threshold ซึ่งขั้นตอนในการหาค่า Threshold โดยวิธีนี้อธิบายได้ดังนี้

ขั้นที่ 1 หาฮิสโตแกรม

ขั้นที่ 2 หาระดับเทาที่มีจำนวนจุดอยู่มากที่สุด

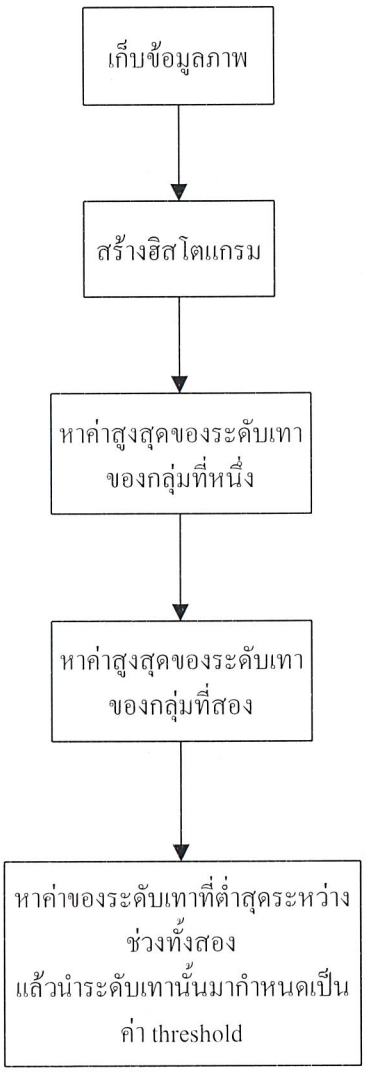
ขั้นที่ 3 หาระดับเทาที่มีจำนวนจุดสูงสุดเป็นอันดับที่สองจาก

$$\max\{((k-j)^2 h[k])(0 \le k \le 255) \qquad (4.6)$$

ขั้นที่ 4 หาระดับเทาที่ต่ำสุดในช่วงระหว่างจุดสูงสุดทั้งสอง แล้วนำระดับเทานั้นมาเป็น Threshold

พบว่าการกำหนดค่า Threshold โดยวิธีนี้ไม่สามารถใช้ได้กับภาพที่มีลักษณะของ histogram ที่มีจำนวน peaks มากๆ ได้ แต่จะใช้เป็นการกำหนด Threshold อย่างง่ายได้

แสดงเป็นขั้นตอนในรูปของ flowchart ได้ดังนี้



รูปที่ 4.4 แสดง flowchart ของวิธีในหัวข้อ 4.1.3

4.1.4) การกำหนดค่า Threshold โดยวิธี Iterative Selection

จากวิธีในการกำหนดค่า Threshold จากค่าเฉลี่ยของระดับเทาในภาพ ตามหัวข้อ 4.1.1 พบว่าค่า Threshold ที่ได้ไม่เหมาะสมกับการนำไปใช้กับภาพส่วนใหญ่ ดังนั้นเราสามารถทำการปรับปรุงการกำหนดค่า โดยจากเดิมที่เรานำค่าเฉลี่ยของระดับเทาของภาพมาเป็นค่า Threshold เลย แต่ในวิธีนี้ มีขั้นตอนการทำดังนี้

ขั้นที่ 1 หาค่าเฉลี่ยของระดับเทาของภาพ จากสมการที่ 4.3 ให้เป็นระดับ k

ขั้นที่ 2 จากนั้นทำการหาค่าเฉลี่ยของระดับเทาในช่วงที่ต่ำกว่าระดับ k จาก

$$T_b = \frac{\sum_{i=0}^{k} i \cdot h[i]}{\sum_{i=0}^{k} h[i]} \tag{4.7}$$

เมื่อ T_b คือ ค่าเฉลี่ยของระดับเทาในช่วงที่ต่ำกว่าระดับ k

i คือ ระดับเทาที่ i

$h[i]$ คือ จำนวนจุดภาพที่มีระดับเทาเป็น i

ขั้นที่ 3 หาค่าเฉลี่ยของระดับเทาในช่วงที่สูงกว่าระดับ k (T_o)จาก

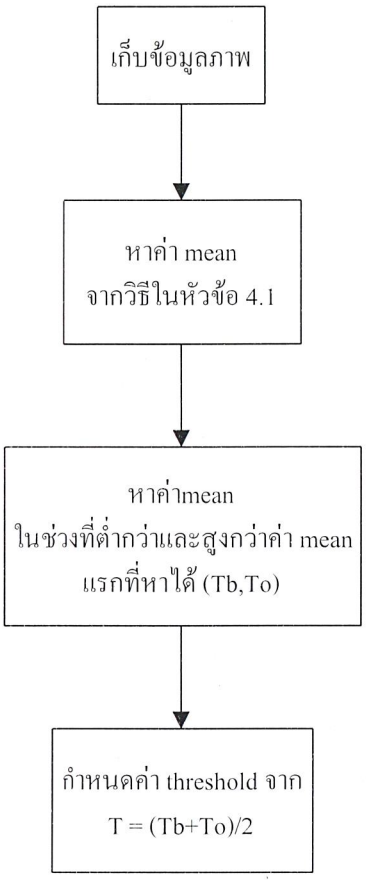
$$T_o = \frac{\sum_{i=k+1}^{255} i \cdot h[i]}{\sum_{i=k+1}^{255} h[i]} \tag{4.8}$$

ขั้นที่ 4 จากนั้นนำค่า T_b และ T_o มากำหนดค่า threshold จาก

$$T = (T_b + T_o)/2 \tag{4.9}$$

นั่นก็คือ เราไม่นำค่าเฉลี่ยที่หาได้จากทั้งภาพมาเป็นค่า threshold เลยแต่เราจะนำค่าเฉลี่ยของระดับเทาในช่วงที่ต่ำกว่าและสูงกว่ามาทำการหาค่าเฉลี่ยอีกครั้ง แล้วจึงนำค่าเฉลี่ยนั้นมากำหนดเป็นค่า threshold

แสดงเป็นขั้นตอนในรูปของ flowchart ได้ดังนี้



รูปที่ 4.5 แสดง flowchart ของวิธีในหัวข้อ 4.1.4

4.1.5) การกำหนดค่า threshold โดย Method of Gray-Level Histogram

จากการกำหนดค่า threshold ด้วยวิธีการต่างๆ ที่ได้กล่าวมาแล้วข้างต้น มีพื้นฐานความคิดอยู่ว่า ค่าเฉลี่ยของระดับเทาในส่วนที่เป็นวัตถุกับส่วนที่เป็นพื้นหลังแตกต่างกัน แต่ในวิธีนี้เป็นการนำเอาค่าความน่าจะเป็นที่จุดภาพจุดต่างๆ ในภาพ จะอยู่ในระดับเทาต่างๆ มาเป็นตัวกำหนด โดยมีวิธีดังนี้

ขั้นที่ 1 หาความน่าจะเป็นที่แต่ละจุดภาพในภาพจะมีโอกาสอยู่ในระดับเทาใดๆ

$$p_i = \frac{n_i}{N} \qquad (4.10)$$

เมื่อ p_i คือ ความน่าจะเป็นที่จุดภาพแต่ละจุดจะมีโอกาสมีระดับเทาเป็น i

$h[i]$ คือ จำนวนจุดภาพที่มีระดับเทาเป็น i

N คือ จำนวนจุดภาพทั้งหมดในภาพ

ขั้นที่ 2 กำหนดระดับเทา k (เริ่มตั้งแต่ 1) แล้วทำการหาผลรวมของความน่าจะเป็น p_i ในช่วงระดับเทาที่น้อยกว่า และมากกว่า k ระดับ

$$w_0 = \sum_{i=0}^{k} p_i = w(k) \tag{4.11}$$

$$w_1 = \sum_{i=k+1}^{255} p_i = 1 - w(k) \tag{4.12}$$

เมื่อ w_0 คือ ค่าความน่าจะเป็นสะสมในช่วงของระดับเทาที่ต่ำกว่าระดับ k

w_1 คือ ค่าความน่าจะเป็นสะสมในช่วงของระดับเทาที่สูงกว่าระดับ k

ขั้นที่ 3 หาค่าเฉลี่ยของความน่าจะเป็นในช่วงทั้งสอง

$$u_0 = \frac{\sum_{i=0}^{k} i \cdot p_i}{w_0} \tag{4.13}$$

$$u_1 = \frac{\sum_{i=k+1}^{255} i \cdot p_i}{w_1} \tag{4.14}$$

เมื่อ u_0 คือ ค่าเฉลี่ยในช่วงของระดับเทาที่ต่ำกว่าระดับ k

u_1 คือ ค่าเฉลี่ยในช่วงของระดับเทาที่สูงกว่าระดับ k

ขั้นที่ 4 หาค่าของความแปรปรวน (variance)รวมของข้อมูลในช่วงทั้งสอง

$$\sigma^2 = w_0(u_0 - u_T)^2 + w_1(u_1 - u_T)^2 \tag{4.15}$$

โดย

$$u_T = \sum_{i=0}^{255} i \cdot p_i \tag{4.16}$$

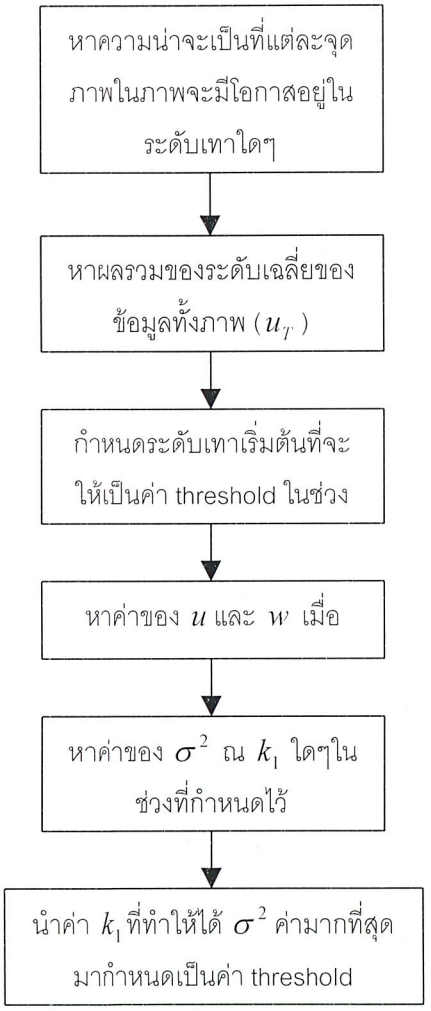
และ $w_0 u_0 + w_1 u_1 = u_T$ และ $w_0 + w_1 = 1$

ขั้นที่ 5 จากนั้นกลับไปทำขั้นที่ 2 ใหม่โดย เปลี่ยนค่า $k = k + 1$ จนถึง $k = 255$ แล้วทำการเปรียบเทียบหาว่าที่ค่าระดับ k ใดที่ทำให้ได้ค่าของ σ^2 สูงสุด แล้วนำค่า k_* มาเป็นค่า threshold

$$\sigma^2(k_*) = \max_{1 \le k \le 255} \sigma^2(k) \qquad (4.17)$$

นั่นคือ วิธีนี้เป็นการนำค่าความน่าจะเป็นที่แต่ละจุดภาพในภาพจะมีโอกาสมีระดับเทาเป็นระดับต่างๆ มาใช้ในการกำหนดค่า threshold โดยนำมาทำการหาค่าความแปรปรวน (variance) ซึ่งระดับ k ใดๆ เป็นค่า threshold ที่เหมาะสมได้นั้น ต้องทำให้การกระจายของข้อมูลภาพในช่วงที่ต่ำกว่าและสูงกว่ามันมีค่าสูงสุด

แสดงขั้นตอนในรูปของ flowchart ได้ดังนี้



รูปที่ 4.6 แสดง flowchart ของวิธีในหัวข้อ 4.1.5

4.2 Human Brain Segmentation By Multithreshold

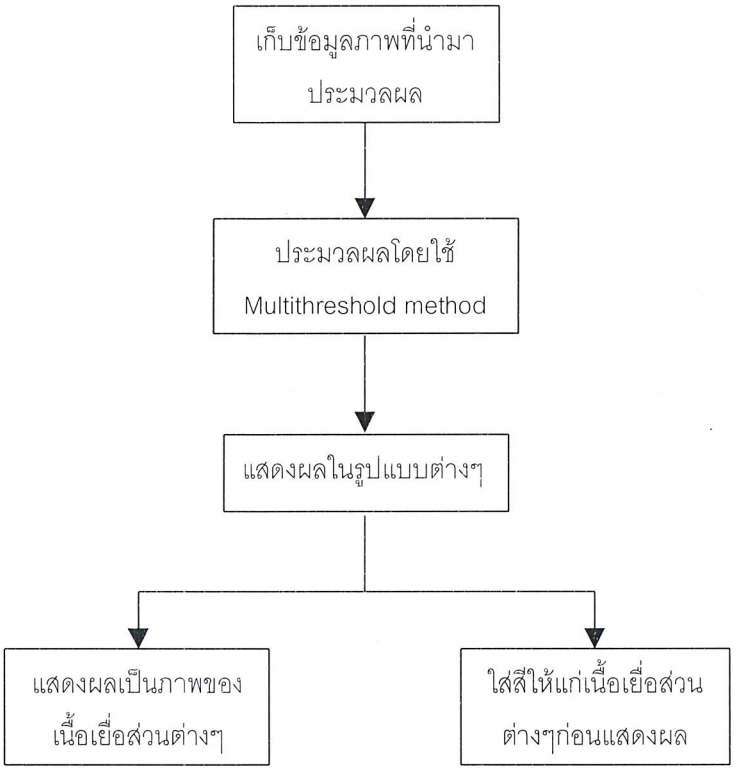
จากหัวข้อ 4.1 ทำให้เราสามารถแบ่งกลุ่มของภาพออกได้เป็น 2 กลุ่มคือ กลุ่มของภาพที่เป็นส่วนของวัตถุและกลุ่มของภาพที่เป็นส่วนของพื้นหลังซึ่งในปริญญานิพนธ์นี้เป็นการแบ่งกลุ่มของเนื้อเยื่อสมองออกเป็นเนื้อเยื่อกลุ่มต่างๆซึ่งหมายถึงเราจะแบ่งกลุ่มของข้อมูลออกเป็นหลายๆ กลุ่ม ดังนั้นจึงต้องมีการนำทฤษฎีพื้นฐานที่ได้กล่าวมาแล้วในหัวข้อ 4.1 มาประยุกต์ให้สามารถแบ่งกลุ่มของข้อมูลออกเป็นหลายๆ กลุ่มได้ ซึ่งวิธีที่จะนำมาประยุกต์ใช้ในการทำ Multithreshold ก็คือวิธีในหัวข้อ 4.1.5

โดยเรากำหนดกลุ่มของข้อมูลที่ต้องการแบ่งออกเป็น 5 กลุ่มข้อมูลคือ

- White matter
- Gray matter
- CSF
- ไขมันใต้ผิวหนัง(ซึ่งในส่วนนี้ใช้ในการดูรูปร่างของกะโหลก เนื่องจาก จากบทที่ 2 พบว่าในภาพ MRI ไม่สามารถมองเห็นส่วนของกะโหลกนี้แต่ ที่ทำการกำหนดคลาสนี้ขึ้นเนื่องจากคิดว่าน่าจะสามารถใช้ในการดูลักษณะของกะโหลกได้บ้างไม่มากก็น้อย ซึ่งในทางปฏิบัติหากต้องการดู ลักษณะของกะโหลกแล้วเราจะใช้การวินิจฉัย จากภาพ CT ซึ่งสามารถเห็นส่วนของกะโหลกอย่างชัดเจน ดังนั้นในปริญญานิพนธ์ฉบับนี้เมื่อมีการกล่าวถึง เนื้อเยื่อในส่วนกะโหลก จะหมายถึงส่วนของไขมันใต้ผิวหนัง)
- ส่วนของพื้นหลังของภาพ เนื่องจากเป็นส่วนหนึ่งของข้อมูลภาพ จึงต้องนำมาจัดให้เป็นกลุ่มหนึ่ง

หมายเหตุ ภาพ MRI ที่นำมาใช้ในปริญญานิพนธ์นี้เป็นภาพ MRI ในชั้นที่ 33 ซึ่งเป็นชั้นที่อยู่ ในบริเวณกึ่งกลางศรีษะ และในชั้นนี้สามารถเห็นเนื้อเยื่อต่างๆ ได้ชัดเจน

ขั้นตอนในการทำ Segmentation



รูปที่ 4.7 แสดงขั้นตอนการทำ Automatic Segmentation

4.2.1) ส่วนเก็บข้อมูลภาพ

เนื่องจากในปริญญานิพนธ์เป็นการเขียนโปรแกรมที่ทำงานบนลีนุกซ์ซึ่งบนลีนุกซ์มีไลบารี (library) ที่ช่วยในการทำงานให้ง่ายขึ้น โดยในปริญญานิพนธ์นี้ได้ใช้ ไลบารี 2 ตัวนั่นคือ Xlib (X library) และ Imlib (Image library)

ใน Imlib มี structure ที่ใช้ในการเก็บข้อมูลภาพนั่นคือ ImlibImage ซึ่งมีโครงสร้างภายใน structure ดังนี้

```
typedef struct_ImlibImage
        {
          int                   rgb_width,rgb_height;
```

```
    unsigned char  *rgb_data;
    unsigned char  *alpha_data;
    char                    *filename;
} ImlibImage;
```

นั่นคือ เราทำการโหลดข้อมูลภาพ mri slice ต่างๆ มาเก็บไว้ใน structure ImlibImage ด้วยคำสั่ง"**Imlib_load_image**" ซึ่งข้อมูลที่ถูกเก็บไว้ได้แก่ ชื่อภาพ(filename) , ขนาดภาพ (rgb_width,rgb_height) และข้อมูลภาพ(rgb_data) จากนั้นเราก็สามารถนำข้อมูลจากภาพไปประมวลผลต่อไป

4.2.2) ส่วนประมวลผล

เรานำทฤษฎี Method of Gray-level Histogram ในหัวข้อที่ 4.1.5 มาทำการประยุกต์ เพื่อให้สามารถใช้ในการแบ่งกลุ่มของข้อมูลภาพได้หลายๆ กลุ่ม นั่นคือต้องมีการกำหนดค่าที่ จะเป็น threshold มากกว่า 1 ค่า (single threshold) โดยขึ้นอยู่กับจำนวนกลุ่มของข้อมูลที่ ต้องการแบ่ง โดยในปริญญานิพนธ์นี้ทำการแบ่งกลุ่มของเนื้อเยื่อสมองออกเป็น 5 กลุ่ม ดังที่ได้กล่าวมาแล้วในตอนต้น ดังนั้นเราจึงต้องทำการกำหนดระดับเทาที่จะให้เป็นค่า threshold จำนวน 4 ค่า โดยขั้นตอนในการทำ segmentation ด้วยวิธี Multithreshold มีดังนี้

ขั้นที่ 1 หาความน่าจะเป็นที่แต่ละจุดภาพในภาพจะมีโอกาสอยู่ในระดับเทาระดับใดๆ

$$p_i = \frac{n_i}{N} \tag{4.18}$$

เมื่อ p_i คือ ความน่าจะเป็นที่จุดภาพแต่ละจุดจะมีโอกาสมีระดับเทาเป็น i

$h[i]$ คือ จำนวนจุดภาพที่มีระดับเทาเป็น i

N คือ จำนวนจุดภาพทั้งหมดในภาพ

ขั้นที่ 2 หาผลรวมของระดับเฉลี่ยของข้อมูลทั้งภาพ (u_T)

$$u_T = \sum_{i=0}^{255} i \cdot p_i \tag{4.19}$$

ขั้นที่ 3 กำหนดระดับเทาเริ่มต้น (k_1, k_2, k_3, k_4) ที่จะให้เป็นระดับ threshold ซึ่งแต่ละระดับต้องต่างกันอย่างน้อย 1 ระดับ (เพื่อให้เกิดช่วงห่างของแต่ละระดับ) โดยกำหนดเป็นช่วงดังนี้

$$\begin{aligned} 0 \le k_1 &< 250 \\ k_1 + 2 \le k_2 &< 252 \\ k_2 + 2 \le k_3 &< 254 \\ k_3 + 2 \le k_4 &< 256 \end{aligned} \tag{4.20}$$

ขั้นที่ 4 หาผลรวมของความน่าจะเป็นแต่ละช่วง (w_1, w_2, w_3, w_4)

$$\begin{aligned} w_1 &= \sum_{i=0}^{k1} p_i \\ w_2 &= \sum_{i=k_1+2}^{k_2} p_i \\ w_3 &= \sum_{i=k_2+2}^{k_3} p_i \\ w_4 &= \sum_{i=k_3+2}^{k_4} p_i \end{aligned} \tag{4.21}$$

ขั้นที่ 5 หาค่าเฉลี่ยของความน่าจะเป็นในแต่ละช่วง (u_1, u_2, u_3, u_4)

$$\begin{aligned} u_1 &= \frac{\sum_{i=0}^{k_1} i \cdot p_i}{w_1} \\ u_2 &= \frac{\sum_{i=k_1+2}^{k_2} i \cdot p_i}{w_2} \\ u_3 &= \frac{\sum_{i=k_2+2}^{k_3} i \cdot p_i}{w_3} \\ u_4 &= \frac{\sum_{i=k_3+2}^{k_4} i \cdot p_i}{w_4} \end{aligned} \tag{4.22}$$

ขั้นที่ 6 หาค่าของความแปรปรวน (variance)รวมของข้อมูลทุกช่วง

$$\sigma^2 = w_1(u_1 - u_T)^2 + w_2(u_2 - u_T)^2 + w_3(u_3 - u_T)^2 + w_4(u_4 - u_T)^2 \tag{4.23}$$

ขั้นที่ 7 จากนั้นกลับไปทำขั้นที่ 4 ใหม่โดยเพิ่มค่าของ $k_n = k_n + 2$ ($n = 1,2,3,4$) จนครบช่วงที่ได้ตั้งไว้ในขั้นที่ 3 แล้วทำการเปรียบเทียบหาว่าที่ค่าระดับ k_1 , k_2 , k_3 , k_4 ใดที่ทำให้ได้ค่าของ σ^2 สูงสุด แล้วนำค่า k_* นั้นมากำหนดเป็นค่า threshold

$$\sigma^2\left(k_1^*, k_2^*, k_3^*, k_4^*\right) = \max_{1 \le k_1 < k_2 < k_3 < k_4 < 256} \sigma^2\left(k_1, k_2, k_3, k_4\right) \qquad (4.24)$$

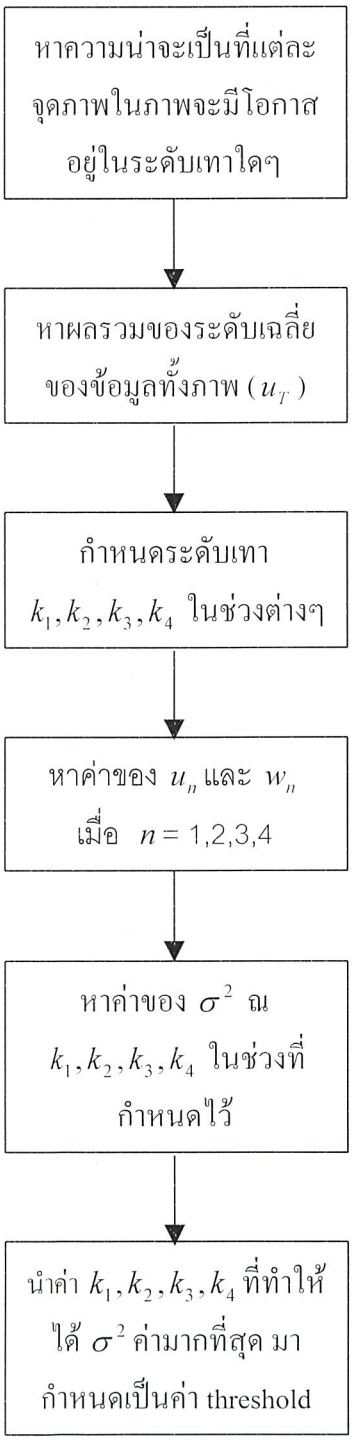
4.2.3) ส่วนแสดงผล

หลังจากที่เรานำข้อมูลภาพเริ่มต้น ไปผ่านกระบวนการประมวลผลแล้ว ทำให้เราได้กลุ่มของเนื้อเยื่อสมองกลุ่มต่างๆ ซึ่งจากนั้นเราจะนำข้อมูลในแต่ละกลุ่มนี้ไปแสดงผลในรูปแบบต่างๆ ดังนี้

- แสดงเป็นภาพของข้อมูลในแต่ละกลุ่ม ซึ่งก็คือแสดงเป็นส่วนของเนื้อเยื่อต่างๆ อันได้แก่ ส่วนของกะโหลก , ส่วนของ white matter , ส่วนของ gray matter และส่วนของ CSF
- แสดงเป็นภาพที่มีการใส่สีให้แก่กลุ่มของข้อมูลที่ต่างกัน แล้วนำส่วนของเนื้อเยื่อต่างๆ ที่ผ่านการใส่สีแล้ว มาประกอบกลับให้เป็นรูปๆเดียว

โดยการใส่สีให้แก่กลุ่มของข้อมูลทำได้โดย เนื่องจากการเก็บข้อมูลภาพในลักษณะ true color หรือในแต่ละจุดภาพมีข้อมูลอยู่ 24 บิตคือ RGB บิต และในภาพที่เป็นระดับเทา ข้อมูลใน RGB บิตจะมีระดับเดียวกัน ดังนั้นหากต้องการทำให้ข้อมูลภาพเป็นสีต่าง ก็สามารถทำได้โดยการกำหนดให้ค่าของข้อมูลใน RGB บิตมีค่าไม่เท่ากัน เช่น หากต้องการสีแดง ก็ให้ทำการกำหนดให้ค่าของระดับใน R บิตมีค่า 255 ส่วนระดับของข้อมูลใน G และ B บิตให้เป็น 0 ก็จะทำให้ที่จุดภาพนั้นมีค่าเป็นสีแดง เป็นต้น

ขั้นตอนในการ segmentation โดยใช้ Multithreshold Method



รูปที่ 4.8 แสดงการใช้ Multithreshold ในการทำ segmentation

บทที่ 5

Human Brain Segmentation
โดยใช้
Fuzzy C-Mean Algorithm

5.1 ทฤษฎีพื้นฐานของฟัซซี่ลอจิก

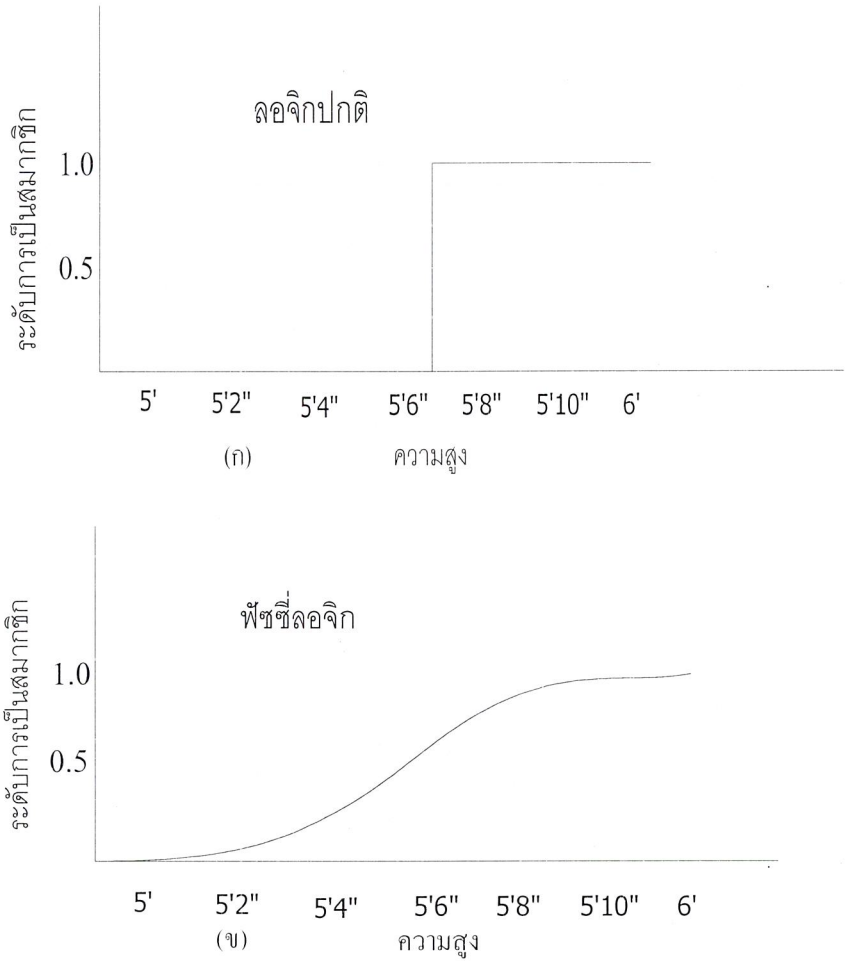
ฟัซซี่ลอจิกมีพื้นฐานอยู่บนทฤษฎีของฟัซซี่เซต โดยฟัซซี่เซตเป็นทฤษฎีในทางคณิตศาสตร์แขนงหนึ่ง ที่ใช้วิเคราะห์ข้อมูลที่มีความคลุมเครือ มีลักษณะการตัดสินใจคล้ายการตัดสินใจของมนุษย์ และได้ถูกนำไปประยุกต์ใช้กับงานหลายแขนง เช่น ควบคุมระบบ (System Control) , การประมวลผลภาพ (Image Processing) , หรือในระบบผู้เชี่ยวชาญแขนงต่างๆ จากคุณสมบัติของฟัซซี่เซตซึ่งมีลักษณะของการพิจารณาปัญหาแตกต่างจากเซตปกติ (Crisp Set)

ลักษณะสำคัญของทฤษฎีฟัซซี่เซต คือ การยอมรับสมาชิกที่มีลักษณะตามเซตเพียงบางส่วนเข้ามาเป็นสมาชิก โดยมีการจัดค่าระดับของการเป็นสมาชิก (Membership Function : μ) ซึ่งมีค่าอยู่ตั้งแต่ 0 ถึง 1

ทฤษฎีเซตปกติ (Crisp Set) จะเป็นการเน้นชัดเจนเลยว่าเป็นสมาชิกของเซตหรือไม่เท่านั้น ไม่มีการเป็นสมาชิกของเซตเพียงบางส่วน ยกตัวอย่างเช่น แมวเป็นสมาชิกของสัตว์เลี้ยงลูกด้วยนม แต่กบไม่ใช่ ดังนั้นเซตเหล่านี้จะถูกเรียกว่า คลิซเซต (Crisp Set)

ทฤษฎีฟัซซี่เซตเป็นการรวมสมาชิกของหลายคลิซเซตที่มีอยู่จริงและพอจะมีลักษณะเข้ารวมกลุ่มได้ ตัวอย่างเช่น คลิซเซตของสัตว์เลี้ยงลูกด้วยนม จะเกิดปัญหาในการยอมรับ platypus (สัตว์น้ำชนิดหนึ่งของออสเตรเลียที่มีลักษณะคล้ายตุ่น แต่ออกลูกเป็นไข่) เข้าเป็นสมาชิก แต่ถ้าหากเป็นฟัซซี่เซตของสัตว์เลี้ยงลูกด้วยนมแล้ว สามารถที่ยอมรับเข้าเป็นสมาชิกได้เนื่องจากความเป็นสัตว์เลี้ยงลูกด้วยนม

ในลอจิกทั่วไปจะกำหนดเพียงว่าเป็น “0” หรือ “1” เป็นสมาชิกของเซตหรือไม่ หรือเพียงว่าถูกกับผิด แต่ฟัซซี่เซตจะยอมรับสมาชิกที่มีลักษณะที่ถูกเพียงบางส่วนและผิดเพียงบางส่วน ซึ่งรูปที่ 5.1 แสดงความแตกต่างของลอจิกปกติกับฟัซซี่ลอจิก



รูปที่ 5.1 เปรียบเทียบลักษณะของกราฟระหว่างลอจิกธรรมดากับฟัซซี่ลอจิกในกรณีเกี่ยวกับความสูงของผู้หญิง ถ้าเป็นลอจิกปกติ (รูปที่ 5.1 ก) ผู้หญิงที่มีความสูงตั้งแต่ 5ฟุต 7 นิ้วเท่านั้นจึงจะถือว่าเป็นผู้หญิงสูง แต่ถ้าหากในฟัซซี่เซต (รูปที่ 5.1 ข) ผู้หญิงที่มีความสูงตั้งแต่ 5ฟุต 4 นิ้ว ก็ จะถือว่าเป็นผู้หญิงสูง แต่จะมีระดับการยอมรับมากน้อยแตกต่างกัน

รูปที่ 5.1 เป็นการแสดงขอบเขตของความสูงของผู้หญิง ในรูปที่ 5.1 (ก) ถ้าผู้หญิงที่สูงตั้งแต่ 5 ฟุต 7 นิ้ว จะถือว่าเป็นผู้หญิงสูง แต่ถ้าต่ำกว่านี้ไม่ถือว่าผู้หญิงคนนั้นสูง หากพิจารณาในฟัซซี่เซต จะมีช่วงความจริงเกิดขึ้น สำหรับในรูปที่ 5.1 (ข) ถ้าผู้หญิงสูง 5 ฟุตพอดี บอกได้อย่างชัดเจนว่าผู้หญิงคนนั้นไม่สูง ถ้าเธอสูง 6 ฟุตพอดี ถึงจะบอกได้ว่าเธอเป็นคนสูง แต่ถ้าเธอสูง 5 ฟุต 8 นิ้วถือว่า

ถูกเพียง 75% ถ้าถือตามหลักการณ์นี้ ส่วนจากถูกไปผิดนั้นเปลี่ยนแปลงทีละน้อย ซึ่งสามารถเป็นไปได้ทั้งส่วนถูกและไม่ถูก

การประยุกต์ใช้งานทฤษฎีฟัซซี่เซตจะต้องแสดงค่าระดับ (degree) ซึ่งเป็นค่าที่เป็นไปได้ที่จะเป็นสมาชิกของเซตเราใช้ μ แทนค่าระดับความเป็นสมาชิก ดังนี้

$$\mu_A(X) \rightarrow [0,1]$$

หมายความว่า ระดับความเป็นสมาชิกของ X ในฟัซซี่เซต A อยู่ในช่วงตั้งแต่ 0 ถึง 1 เมื่อประยุกต์เข้ากับฟัซซี่ลอจิกค่า μ ถูกเรียกว่า ค่าความจริงที่แสดงค่าระดับขอบเขตของเซตคือ

$$0 \leq X \leq 1$$

โดยค่า 0 หมายถึง ไม่เป็นสมาชิกเลย และ 1 หมายถึงการเป็นสมาชิกอย่างสมบูรณ์

ข้อได้เปรียบของทฤษฎีฟัซซี่ลอจิก

ทฤษฎีฟัซซี่ลอจิกมีข้อได้เปรียบเหนือระบบลอจิกแบบทั่วไปอยู่ 2 ประการคือ

1) เซตของฟัซซี่ลอจิก สามารถให้คำจำกัดความของคุณภาพทางภาษาได้ดีกว่า เช่น คำว่า ค่อนข้าง หรือ เกือบจะ นำไปใช้ประกอบกับคำแสดงคุณลักษณะทั่วๆไป เช่น สูง , เตี้ย , ดี , เลว , ร้อน , เย็น และอื่นๆ และยังสามารถให้ค่าความสำคัญของแต่ละสมาชิกในเซตได้ ซึ่งจะมีค่าตั้งแต่ 0 ถึง 1

2) เอาท์พุทที่ได้จากระบบฟัซซี่ลอจิก มีการเปลี่ยนแปลงแบบค่อยๆ เป็นๆ ไปอย่างต่อเนื่อง ถึงแม้ว่าอินพุทของระบบจะเปลี่ยนแปลงอย่างทันทีทันใดก็ตาม

นิยามต่างๆ ของฟัซซี่เซต

นิยามที่ 1 ฟัซซี่เซต

ถ้า X เป็นสมาชิกของฟัซซี่เซต A เราเรียก μ_A ว่าเป็นค่าหรืออัตราความเป็นสมาชิก

นิยามที่ 2 การยูเนียนของฟัซซี่เซต

ถ้า A และ B ประกอบด้วยค่าความเป็นสมาชิก $\mu_A(x)$ และ $\mu_B(x)$ ตามลำดับ และ $C = A \cup B$

เราสามารถนิยามได้ว่า

$$\mu_C(x) = MAX(\mu_A(x), \mu_B(x)), x \in X$$

นิยามที่ 3 การ Intersection ของฟัซซี่เซต

ถ้า A และ B มีค่าความเป็นสมาชิก $\mu_A(x)$ และ $\mu_B(y)$ ตามลำดับ และ $C = A \cap B$

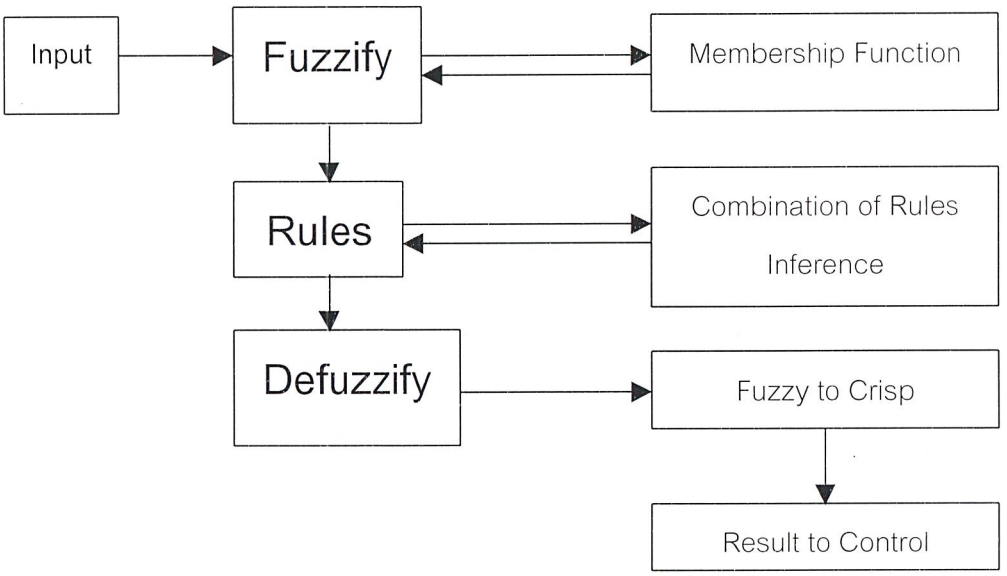
เราสามารถนิยามได้ว่า

$$\mu_C(x) = MIN(\mu_A(x), \mu_B(y)), x \in X$$

ถ้า A และ B มีค่าความเป็นสมาชิก $\mu_A(x)$ และ $\mu_B(y)$ ตามลำดับ และ S = A+B เราสามารถนิยามได้ว่า

$$\mu_C(x) = SUPMIN(\mu_A(x), \mu_B(y))$$

ขั้นตอนการออกแบบระบบฟัซซี่ลอจิก



รูปที่ 5.2 แสดงขั้นตอนในการออกแบบระบบฟัซซี่ลอจิก

ขั้นตอนต่างๆ อธิบายได้ดังนี้

1. **Input/Output** : เป็นขั้นตอนที่เราทำการศึกษาว่า Input และ Output ของระบบคืออะไร และ Input แต่ละตัว มีผลต่อ Output อย่างไร

2. **Fuzzify** : เป็นขั้นตอนที่ทำหน้าที่แปลงข้อมูลที่รับเข้ามา (Input) จากระบบที่จะควบคุม ไปเป็นอินพุทของระบบควบคุมแบบฟัซซี่ ซึ่งอยู่ในรูปของค่าความเป็นสมาชิก (membership) ในฟัซซี่ เซต โดยมีค่าได้ตั้งแต่ 0 ถึง 1

3. **Rules** : เป็นขั้นตอนของการตั้งกฎของฟัซซี่ การตีความหรือการวิเคราะห์ซึ่งใช้ตัวดำเนินการ 'and' โดยใช้ Minimum แทน หรือ 'or' โดยใช้ Maximum ของตัวแปรเงื่อนไข (condition) ในการคำนวณผลลัพธ์รวมจากกฎฟัซซี่ ซึ่งโดยปกติจะใช้เทคนิคของกฎ Minimum เพื่อที่จะทำให้ความแข็งแกร่งของกฎที่ได้เลือกอยู่นั้นขึ้นกับค่าของตัวแปรส่วนเงื่อนไข ที่มีค่าความเป็นสมาชิกน้อยที่สุดโดยลักษณะของ IF/THEN Rules มีดังนี้

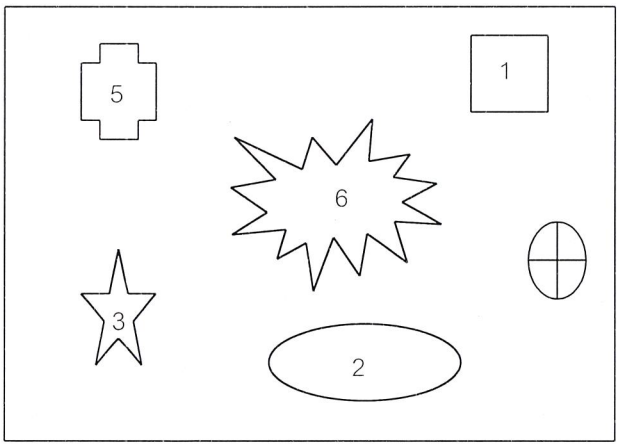
If เงื่อนไข Then ผลลัพธ์

หรือ If เงื่อนไขที่ 1 และ เงื่อนไขที่ 2 Then ผลลัพธ์

4. **Defuzzify** : เป็นขั้นตอนการแปลงข้อมูล ซึ่งเป็นผลลัพธ์ที่จากการประมวลผลกฎฟัซซี่ในขั้นตอนที่ 3 โดยการแปลงจากค่าความเป็นสมาชิกของฟัซซี่เซตไปเป็นค่าของคลิซเซต เพื่อนำไปควบคุมระบบอีกทีหนึ่ง

ตัวอย่างการวิเคราะห์แบบฟัซซี่

โดยปกติแล้วในการแบ่งกลุ่มของข้อมูล เราต้องทำการกำหนดก่อนว่า เราจะทำการแบ่งข้อมูลออกเป็นกี่กลุ่ม และแต่ละกลุ่มนั้นเราจะนำคำจำกัดความใดมาเป็นตัวแบ่งกลุ่ม เช่นรูปที่ 5.1 หากเราจะทำการแบ่งกลุ่มของรูป โดยจะแบ่งตามรูปร่างของรูป เราจะสามารถแบ่งข้อมูล ออกได้เป็น 3 กลุ่ม คือ รูปที่ 1และ 5 น่าจะอยู่กลุ่มเดียวกัน เพราะมีรูปร่างเป็นเหลี่ยมเหมือนกัน รูปที่ 2 และ 4 น่าจะอยู่กลุ่มเดียวกันเพราะมีรูปร่างเป็นวงกลมเหมือนกัน และ รูปที่ 3 และ 6 จะอยู่ในกลุ่มเดียวกัน เพราะมีลักษณะเป็นแฉกเหมือนกัน เป็นต้น



รูปที่ 5.3 แสดงรูปแบบของข้อมูล

จะเห็นว่าในการพิจารณาข้างต้นจะอยู่ในลักษณะของคลิซเซต ซึ่งก็คือพิจารณาแค่ว่ารูปแต่ละรูปอยู่ในเซตที่เรากำหนดไว้หรือไม่แต่ในทางปฏิบัติแล้วจะเห็นว่าหากนำการตัดสินใจของมนุษย์เข้ามาร่วมด้วยในการพิจารณา จะพบว่าการแบ่งกลุ่มของรูปต่างๆ จะเกิดความครุมเครือขึ้น ซึ่งทำให้เกิดปัญหาในการแบ่งกลุ่มเช่นขอบเขตระหว่างแต่ละกลุ่มไม่แน่นอนแทนที่จะเป็นขอบเขตที่แน่นอนคือ แต่ละภาพอาจจะอยู่ในกลุ่มมากกว่า 1 กลุ่ม อย่างเช่น ในรูป 5.1 จะเห็นว่าเราสามารถแบ่งให้รูปที่ 5 อยู่ในกลุ่มเดียวกันกับรูปที่ 3 และ 6 ด้วยก็ได้เนื่องจากรูปที่ 5 ก็มีลักษณะเป็นแฉกเช่นกัน ดังนั้นในกรณีที่รูปมีความใกล้เคียงกันมาก ทำให้การแบ่งกลุ่มข้อมูลทำได้ลำบาก

จากปัญหาดังกล่าว เราได้นำเอา fuzzy model มาใช้แก้ได้ คือในระบบ non fuzzy model ข้อมูลแต่ละจุดจะถูกจัดให้อยู่ในกลุ่มโดยมีค่าความเป็นสมาชิกภาพ (membership หมายถึง ค่าความน่าจะเป็นที่ข้อมูลจะอยู่ในกลุ่มนั้น) เป็น 0 หรือ 1 เท่านั้น (ใช่หรือไม่ใช่) แต่ใน fuzzy model ค่าความเป็นสมาชิกภาพของข้อมูลจะมีค่าอยู่ระหว่าง 0 ถึง 1 [0,1] ซึ่งจะทำให้การคำนวณทำได้ ยืดหยุ่นกว่า

การนำ Fuzzy C-Mean มาใช้ในการคัดแยกเนื้อเยื่อสมอง

จากกรรมวิธีในการคัดแยกเนื้อเยื่อสมองที่ได้นำเสนอไปแล้วก่อนหน้านี้ (Multithreshold) ซึ่งเป็นการกำหนดจุดแบ่ง (threshold) ของช่วงของระดับเทา(gray level) ออกเป็นช่วงต่างๆ ตามระดับเทาของเนื้อเยื่อนั้นๆ ซึ่งจะเห็นว่าวิธีนี้จะมีลักษณะการคิดแบบคลิซเซต นั่นก็คือ เมื่อเราได้ช่วงของระดับเทาของเนื้อเยื่อส่วนต่างๆ แล้ว เราก็จะตัดจุดภาพที่ระดับเทาไม่อยู่ ในช่วงที่ต้องการออก ซึ่งจะเห็นว่าจุดภาพที่มีระดับเทาใกล้เคียงกันจุด Threshold แต่ถูกตัดทิ้ง อาจจะเป็นเนื้อเยื่อสมองในส่วนเดียวกันกับเนื้อเยื่อที่มีระดับเทาอยู่ในช่วงที่กำหนดก็เป็นได้ ดังนั้นวิธี ใหม่ที่จะนำเสนอต่อไปนี้จะเป็นวิธีที่ทำให้ลักษณะของจุดแบ่งมีความยืดหยุ่นกว่าจุดแบ่งในลักษณะคลิซเซต ซึ่งก็คือการนำทฤษฎีของ Fuzzy logic มาช่วย ด้วยเหตุที่ว่าลักษณะการคิดในแบบ Fuzzy logic (ดังที่ได้กล่าวมาแล้วข้างต้น) จะมีการกำหนดค่าความเป็นสมาชิกของเซตอยู่ในช่วง 0 ถึง 1 ซึ่งจะทำให้มีความยืดหยุ่นกว่า

จากภาพถ่ายทางการแพทย์ MRI จะเห็นว่าเนื้อเยื่อสมองแต่ละชนิดจะมีค่าของระดับเทา (gray level) ที่ใกล้เคียงกัน ซึ่งนั่นก็หมายถึงเป็นการยากที่เราจะทำการแยกเนื้อเยื่อสมองแต่ละส่วนออกมาให้ได้สมบูรณ์ที่สุด และจากสมมุติฐานที่ว่า “เนื้อเยื่อสมองที่เป็นเนื้อเยื่อส่วนเดียวกันน่าจะมีค่าของระดับเทาที่ใกล้เคียงกัน” ทำให้เราจะไม่สามารถกำหนดค่าที่จะเป็นจุดตัด(Threshold) ในแบบคลิซเซตได้ (คือเป็นการกำหนดว่าหากจุดภาพใดมีค่าระดับเทาไม่อยู่ในช่วง Threshold ที่กำหนด ก็จะไม่ถือว่าจุดภาพนั้นอยู่ในเซตที่กำหนดทันที) ซึ่งหากเราใช้วิธีในการวิเคราะห์โดยใช้ Fuzzy C-Mean เข้ามาช่วยโดยเราจะทำการกำหนดค่าความเป็นสมาชิกของแต่ ละจุดภาพที่มีต่อเนื้อเยื่อแต่ละส่วน และหากจุดภาพใดๆ มีค่าความเป็นสมาชิกที่มีต่อเซตของเนื้อเยื่อกลุ่มใดมากที่สุดก็จะจัดให้จุดภาพนั้นอยู่ในเซตของเนื้อเยื่อนั้น ซึ่งจะเห็นว่าด้วยกรรมวิธีคิดที่คล้ายกับการตัดสินใจของมนุษย์นี้ น่าจะให้ผลลัพธ์ในการคัดแยกเนื้อเยื่อสมองได้ดีกว่าในการกำหนดค่า Threshold ตามกรรมวิธีที่ได้นำเสนอไปแล้วก่อนหน้านี้

5.2 Hard and Fuzzy C-Mean Partitions

สมมุติว่าเรามีข้อมูลดิบ $X = \{x_1, x_2, ..., x_n\}$ ซึ่ง n คือจำนวนข้อมูลทั้งหมดที่นำมาแบ่ง และ x_k เป็น x ใดๆ $x_i \in R^p$ เป็นข้อมูลที่มีมิติเท่ากับ p ให้ $P(x)$ คือ power set ของ X หรือก็คือ เซตของทุกสับเซตของ X ให้ Hard C-Partition ของ X ก็คือ $\{A_i \in P(x); 1 \le i \le c$ หรือ A_i ก็คือ เซตของข้อมูลที่อยู่ใน cluster เดียวกัน และ c คือ จำนวนของ cluster ที่ต้องการจะแบ่ง เพราะฉะนั้น $\{A_1, A_2, ..., A_c\}$ คือกลุ่มของข้อมูลจำนวน c cluster โดยใน Hard C-Partition สามารถเขียนฟังก์ชันสมาชิกภาพ (membership function) ของ x_k ใน A_i ได้ดังสมการที่ 5.1 โดย u_{ik} คือค่าสมาชิกภาพ (membership value) ซึ่งใช้บอกค่าความน่าจะเป็นที่ x_k จะเป็นสมาชิกของเซต A_i

$$u_{ik} = \begin{cases} 1, x_k \in A_i \\ 0, x_k \notin A_i \end{cases} \tag{5.1}$$

ในกรณีของ Hard C-Partition x_k จะต้องเป็นสมาชิกของ cluster เพียง 1 cluster เท่านั้นถ้าให้ $x_k \in X, A_i \in P(x), i$=1,2,.....,c และ k = 1,2.....,n เพราะฉะนั้น สามารถเขียนเป็นสมการได้ดังสมการที่ 5.2 และ 5.3

$$u_{ik} \in \{0,1\}, 1 \le i \le c, 1 \le k \le n \tag{5.2}$$

$$\sum_{i=1}^{c} u_{ik} = 1, \forall k \in \{1,2,...., n\} \tag{5.3}$$

เนื่องจาก cluster แต่ละ cluster จะต้องมีสมาชิกที่เป็นข้อมูลดิบอย่างน้อย 1 ตัว และมีจำนวนไม่เกินจำนวนข้อมูลดิบทั้งหมด สามารถเขียนเป็นสมการได้ดังสมการที่ 5.4

$$0 < \sum_{k=1}^{n} u_{ik} < n, \forall i \in \{1,2,....., c\} \tag{5.4}$$

นิยามสำหรับ matrix ของ Hard C-Partition เขียนได้ดังนี้

นิยาม 5.1 กำหนดให้ $X = \{x_1, ..., x_n\}$ เป็นเซตของข้อมูลดิบ และ V_{cn} คือเซตของ real matrix $U = [u_{ik}]$ ขนาด $c \times n$ แสดงได้ดังสมการที่ 5.5 และ c เป็นเลขจำนวนเต็มซึ่งมีค่าอยู่ในช่วง $2 \le c \le n$ หรือก็คือจำนวน cluster ที่ต้องการจะแบ่งจะต้องมากกว่า 1 และต้องไม่เกินจำนวนข้อมูลดิบทั้งหมด เพราะฉะนั้นเซตของ Hard C- Partition สำหรับ X แสดงได้ดังสมการที่ 5.6

$$V_{cn} = \{U_1, U_2, ..., U_m / u_{ik} \in \{0,1\} and (2.3) true\} \tag{5.5}$$

$$M_c = \{U \in V_{cn}\} \tag{5.6}$$

ตัวอย่างที่ 1 ถ้าเราให้ X เป็น set ของส่วนสูงของคน 3 คน

X = { x1 = 178 cm. , x2 = 189 cm. , x3 = 165 cm.}

ถ้าเราต้องการแบ่งเป็น 2 cluster (c = 2) จากสมการ (5.2) และ (5.3) เราอาจได้ Hard C-Partition ของ X 3 แบบดังนี้

$$U_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad U_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad U_3 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

หากเราต้องการที่จะแบ่งกลุ่มโดยใช้ความสูง 180 cm. เป็นเกณฑ์แล้ว U_3 น่าจะเป็น partition ที่ เหมาะสมที่สุด แต่ในความเป็นจริงจะเห็นว่า คนที่สูง179 ก็สูงใกล้เคียงกับเกณฑ์ที่ตั้งไว้มาก เพราะฉะนั้นทางแก้ปัญหาก็คือ ให้ u_{ik} มีค่าอยู่ในช่วง [0,1] โดยนำ Fuzzy C-Partition มาใช้

นิยาม 5.2 Fuzzy C-Mean Partition ให้ X และ c เป็นตามนิยาม 5.1

$$cn = \{U_1, U_2, \ldots\ldots, U_m / u_{ik} \in [0,1] and (4.3) true\} \qquad (5.7)$$

$$M_{fc} = \{U \in V_{cn}\} \qquad (5.8)$$

เพราะฉะนั้นจะทำให้เราได้ค่าสมาชิกภาพที่แต่ละคนมีต่อแต่ละกลุ่มใหม่ได้เป็น

$$U = \begin{bmatrix} 0.9 & 0.2 & 0.9 \\ 0.2 & 0.8 & 0.1 \end{bmatrix}$$

5.2.1) Object Function Clustering and Hard C-Means Algorithm

ในการเลือกว่าจะใช้ partition จาก M_c หรือ M_{fc} มีวิธีในการเลือก 3 วิธีคือ

- วิธี hierarchical
- วิธี graph-theoretic
- วิธี objective function

แต่ที่นิยมคือ วิธี objective function ซึ่งสมการที่นิยมใช้เรียกว่า *over all within-group sum of square errors*

$$J_w(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} \left\| x_k - v_i \right\|^2 \qquad (5.9)$$

ซึ่ง $U = [u_{ik}] \in M_c$ หรือ $M_{fc}, V = (v_1, \ldots\ldots, v_c)$ซึ่ง v_i เป็นจุดศูนย์กลางของ cluster A_i ดังนี้

$$V_i = \frac{\sum_{k=1}^{n} U_{ik} X_k}{\sum_{k=1}^{n} U_{ik}} \qquad (5.10)$$

หรือก็คือ v_i คือค่าเฉลี่ย (ของhard c-partitionX) หรือค่าน้ำหนักเฉลี่ย (ของfuzzy c-partition) ทุกจุดข้อมูลใน A_i จากนี้เราสมมุติว่า $x_k, v_i \in R^p$ ซึ่ง U คือ hard c-partition เพราะฉะนั้น $J_w(U,V)$ จากสมการ (5.7)เขียนใหม่ได้ดังนี้

$$J_w(U,V) = \sum_{i=1}^{c}\left(\sum_{x_i \in A_i}\|x_k - v_i\|^2\right) \qquad (5.11)$$

จากสมการนี้จึงได้ชื่อว่า *over all within-group sum of square errors* ซึ่ง $u_{ik}\|x_k - v_i\|^2$ ก็คือ square errors ที่เกิดขึ้นระหว่าง x_i กับ v_i โดยค่า $J_w(U,V)$ ควรจะน้อยถ้าข้อมูลนั้นอยู่ใน hard cluster A_i ที่มีศูนย์กลางที่ v_i สามารถเขียนได้ดังสมการที่ 5.12

$$if\left(M_c = U_m or M_{fc} = U_m\right) then\left(J_w(U_m, V_m) = MIN\left(\forall J_w(U,V)\right)\right) \qquad (5.12)$$

เพราะฉะนั้น ทั้ง hard c-means algorithm และ fuzzy c-means algorithm ก็คือการวนซ้ำเพื่อหาค่าของ $J_w(U,V)$ ที่ดีที่สุดหรือก็คือที่น้อยที่สุดนั่นเอง

Hard C-Means Algorithm :

<u>ขั้นที่ 1</u> สมมุติว่า n จุดข้อมูล $X = \{x_1, x_2, ..., x_n\}$ ซึ่ง $x_i \in R^p$ โดย $2 \le c \le n$ และเริ่มต้นให้ $U^{(0)} \in M_c$

<u>ขั้นที่ 2</u> ให้ $l = 0,1,2,.....$ เป็นตัวบอกจำนวนรอบ จะได้ c-means vectors

$$v_i^{(l)} = \frac{\sum_{k=1}^{n} u_{ik}^{(l)} x_k}{\sum_{k=1}^{n} u_{ik}^{(l)}} \qquad (5.13)$$

ซึ่ง $\left[u_{ik}^{(l)}\right] = U^{(l)}$ และ i = 1,2,.....,c

<u>ขั้นที่ 3</u> ทำการเปลี่ยนแปลงจาก $U^{(l)} \to U^{(l+1)}$ โดยใช้

$$u_{ik}^{(l+1)} = \begin{cases} 1 & \left\|x_k - v_i^{(l)}\right\| = \min_{1 \le j \le c}\left(\left\|x_k - v_i^{(l)}\right\|\right) \\ 0 & otherwise \end{cases} \qquad (5.14)$$

<u>ขั้นที่ 4</u> เปรียบเทียบ $U^{(l)}$ กับ $U^{(l+1)}$:

$if \left\|U^{(l+1)} - U^{(l)}\right\| < \varepsilon$ ค่าความผิดพลาดที่ยอมรับได้

then $l = l + 1$ และกลับไปทำขั้นที่ 2

จากขั้นตอน hard c-mean algorithm สามารถอธิบายได้ดังนี้คือ กำหนดจำนวน cluster(ขั้นที่1) , หาจุดศูนย์กลาง (ขั้นที่2) , คำนวณ cluster membership ใหม่จาก minimum square errors ระหว่างจุดข้อมูลกับจุดศูนย์กลาง(ขั้นที่3) , หยุดเมื่อค่าความผิดพลาดยอมรับได้ (ขั้นที่4)

5.2.2) The Fuzzy C-Means Algorithm

ใน fuzzy c-means algorithm $U = [u_{ik}] \in M_{fc}$ และ $V = (v_1, ..., v_c)$ ซึ่ง $v_i \in R^p$ จะมี objective function เป็น

$$J_w(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}\|x_k - v_i\|^2 \qquad (5.15)$$

โดย m คือ ค่า weight constant เป็นค่าจำนวนจริงที่อยู่ในช่วง $m \in (1,\infty)$

ทฤษฎี 5.3 ให้ $X = \{x_1, x_2, ..., x_n\}$, $x_i \in R^p$ เป็นเซตของข้อมูล ให้ $c \in \{2,3,....,n-1\}$ และ $m \in (1,\infty)$ และสมมุติว่า $\|x_k - v_i\| \neq 0$ สำหรับ $1 \le k \le n, 1 \le i \le c$ ดังนั้น $U = [u_{ik}]$ และ $V = (v_1, ..., v_c)$ เป็น local minimum ของ $J_w(U,V)$ เมื่อ

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left(\frac{\|x_k - v_i\|}{\|x_k - v_j\|}\right)^{\frac{2}{m-1}}}, 1 \le k \le n, 1 \le i \le c \qquad (5.16)$$

$$v_i = \frac{\sum_{k=1}^{n}(u_{ik})^m x_k}{\sum_{k=1}^{n}(u_{ik})^m}, 1 \le i \le c \qquad (5.17)$$

หมายเหตุ ขั้นตอนการพิสูจน์สมการ(5.16) และ (5.17) ทำได้โดยวิธี Lagrange multiplier ซึ่งไม่ นำมาแสดงในที่นี้

Fuzzy C-Means Algorithm :

ขั้นที่ 1 สมมุติว่า n จุดข้อมูล X = {x1,....,xn ซึ่ง $xi \in R^p$ โดย $c \in \{2,3,....,n-1\}$ และกำหนดค่าเริ่มต้นให้กับ m และค่าเริ่มต้นของ $U^{(0)} \in M_{fc}$

ขั้นที่ 2 ให้ $l = 0,1,2,....$ จะได้ c-means vectors

$$v_i^{(l)} = \frac{\sum_{k=1}^{n} \left(u_{ik}^{(l)}\right)^m x_k}{\sum_{k=1}^{n} \left(u_{ik}^{(l)}\right)^m}, 1 \le i \le c \qquad (5.18)$$

ขั้นที่ 3 เปลี่ยนแปลง $U^{(l)} = \left[u_{ik}^{(l)}\right] \rightarrow U^{(l+1)} = \left[u_{ik}^{(l+1)}\right]$ โดยใช้

$$u_{ik}^{(l+1)} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\left\| x_k - v_i^{(l)} \right\|}{\left\| x_k - v_j^{(l)} \right\|} \right)^{\frac{2}{m-1}}}, 1 \le k \le n, 1 \le i \le c \qquad (5.19)$$

ขั้นที่ 4 เปรียบเทียบ $U^{(l)}$ กับ $U^{(l+1)}$:

if $\left\| U^{(l+1)} - U^{(l)} \right\| < \varepsilon$ ค่าความผิดพลาดที่ยอมรับได้

then $l = l + 1$ และกลับไปทำขั้นที่ 2

จากวิธี fuzzy c-means algorithm จะพบว่ามีพารามิเตอร์อีก 1 ตัวที่มากกว่า hard c-means algorithm คือ m(weight constant) มีค่าอยู่ในช่วง $m \in (1,\alpha)$ ซึ่งค่า m จะควบคุมความคลุมเครือ (fuzziness) ของการประมวลผลทั้งหมด ถ้า $m \rightarrow \alpha$ จะทำให้ fuzzy c-means algorithm มีความใกล้เคียงกับ hard c-means algorithm มาก (คือเกือบทำให้ค่าสมาชิกภาพมีเพียง 1 หรือ 0 เท่านั้น) ถ้า $m \rightarrow \alpha$ จะทำให้ค่า $J_m \rightarrow 0$ เนื่องจากค่าสมาชิกภาพอยู่ในช่วง $[0,1]$

เมื่อยกกำลังสูงจะมีค่าเข้าใกล้ศูนย์ หรือก็คือ เมื่อค่า m มีค่าสูงขึ้น การแบ่งค่าสมาชิกภาพของข้อมูลนั้นไปให้กับแต่ละ cluster จะมีความละเอียดมากขึ้น สรุปได้ว่า ค่า m จะควบคุมขอบเขตของค่าสมาชิกภาพของข้อมูลนั้นๆ ในแต่ละ cluster ซึ่งถ้ากำหนดให้พารามิเตอร์ทุกตัวในอัลกอริธึมคงที่ เมื่อเพิ่มค่า m จะทำให้ J_m ลดลงและทำให้ใช้เวลาในการลู่เข้านานขึ้นด้วย ในปัจจุบันยังไม่มีทฤษฎีใดที่ใช้ในการเลือกค่า m อย่างไรก็ตาม ระบบจะลู่เข้าอย่างแน่นอนเมื่อ $m \in (1,\alpha)$ ส่วนการลู่เข้าของระบบแสดงได้ดังสมการที่ 5.20 ส่วนการพิสูจน์จะไม่นำมาแสดงในที่นี้

$$J_m\left(U^{(l+1)}, V^{(l+1)}\right) \le J_m\left(U^{(l)}, V^{(l)}\right) \qquad (5.20)$$

ตัวอย่างที่ 2 หากต้องการทำการแบ่งกลุ่มของข้อมูลที่มีข้อมูลตั้งแต่ 0-19 ออกเป็น 4 กลุ่มโดยใช้ Fuzzy C-Mean ทำการกำหนดค่าจุดศูนย์กลางกลุ่มเริ่มต้น ดังนี้

จุดศูนย์กลางกลุ่มที่ 1 = 4

จุดศูนย์กลางกลุ่มที่ 2 = 9

จุดศูนย์กลางกลุ่มที่ 3 = 14

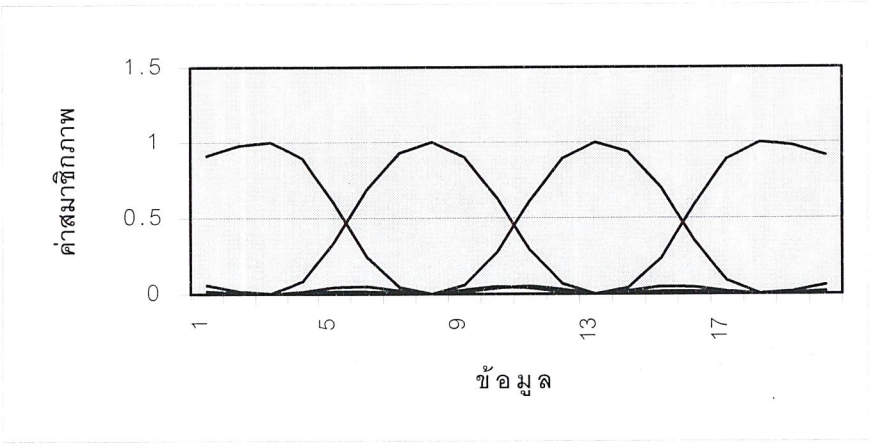
จุดศูนย์กลางกลุ่มที่ 4 = 19

ใช้สมการที่ 5.18 และ 5.19 ในการคำนวณ สุดท้ายจะได้ค่าความเป็นสมาชิกของข้อมูลแต่ละตัวดังนี้

ข้อมูล	ค่าสมาชิกที่ต่อจุดศูนย์กลางกลุ่มต่างๆ			
	กลุ่มที่ 1	กลุ่มที่ 2	กลุ่มที่ 3	กลุ่มที่ 4
0	0.9099	0.0605	0.01975	0.0098
1	0.9756	0.0171	0.00487	0.0023
2	0.9975	0.0019	0.00045	0.000197
3	0.8923	0.0853	0.0158	0.0065
4	0.597	0.3417	0.0444	0.0167
5	0.2465	0.6862	0.0502	0.017
6	0.0448	0.9276	0.0212	0.0063
7	0.000198	0.9995	0.0002	0.00005
8	0.0268	0.8999	0.06	0.012
9	0.052	0.63	0.2779	0.0399
10	0.0409	0.2924	0.6138	0.053
11	0.0132	0.0675	0.8905	0.0288
12	0.00014	0.0006	0.9987	0.00054
13	0.0057	0.0195	0.9344	0.04
14	0.0165	0.04925	0.6983	0.236
15	0.0167	0.04479	0.3523	0.586
16	0.0067	0.0163	0.0894	0.8875
17	0.00023	0.00053	0.0022	0.9969

18	0.0022	0.0047	0.0167	0.9764
19	0.0096	0.0195	0.0602	0.9107

ตารางที่ 5.1 แสดงค่าสมาชิกของข้อมูลแต่ละตัวที่มีต่อค่าศูนย์กลางกลุ่มแต่ละกลุ่ม



รูปที่ 5.4 ความสัมพันธ์ระหว่างค่าสมาชิกของข้อมูลแต่ละตัวที่มีต่อค่าศูนย์กลางกลุ่มแต่ละกลุ่ม

เราสามารถแบ่งข้อมูลออกเป็น 4 กลุ่มโดยดูค่าสมาชิกว่าค่าสมาชิกที่ข้อมูลแต่ละตัวมีต่อกลุ่มใดมากที่ก็จัดให้ข้อมูลตัวนั้นอยู่ในกลุ่มนั้น ซึ่งสามารถจัดได้ดังนี้

กลุ่มที่ 1 ได้แก่ 0 - 4

กลุ่มที่ 2 ได้แก่ 5 - 9

กลุ่มที่ 3 ได้แก่ 10 - 14

กลุ่มที่ 4 ได้แก่ 15 - 19

5.3 Segmentation By Fuzzy C-means Algorithm

จากทฤษฎีข้างต้น เราจะนำ fuzzy c-means algorithm มาใช้ในการแบ่งกลุ่มของข้อมูลจากภาพ MRI โดยเราจะกำหนดกลุ่มของข้อมูลที่ต้องการแบ่งออกเป็น 5 กลุ่มข้อมูลคือ

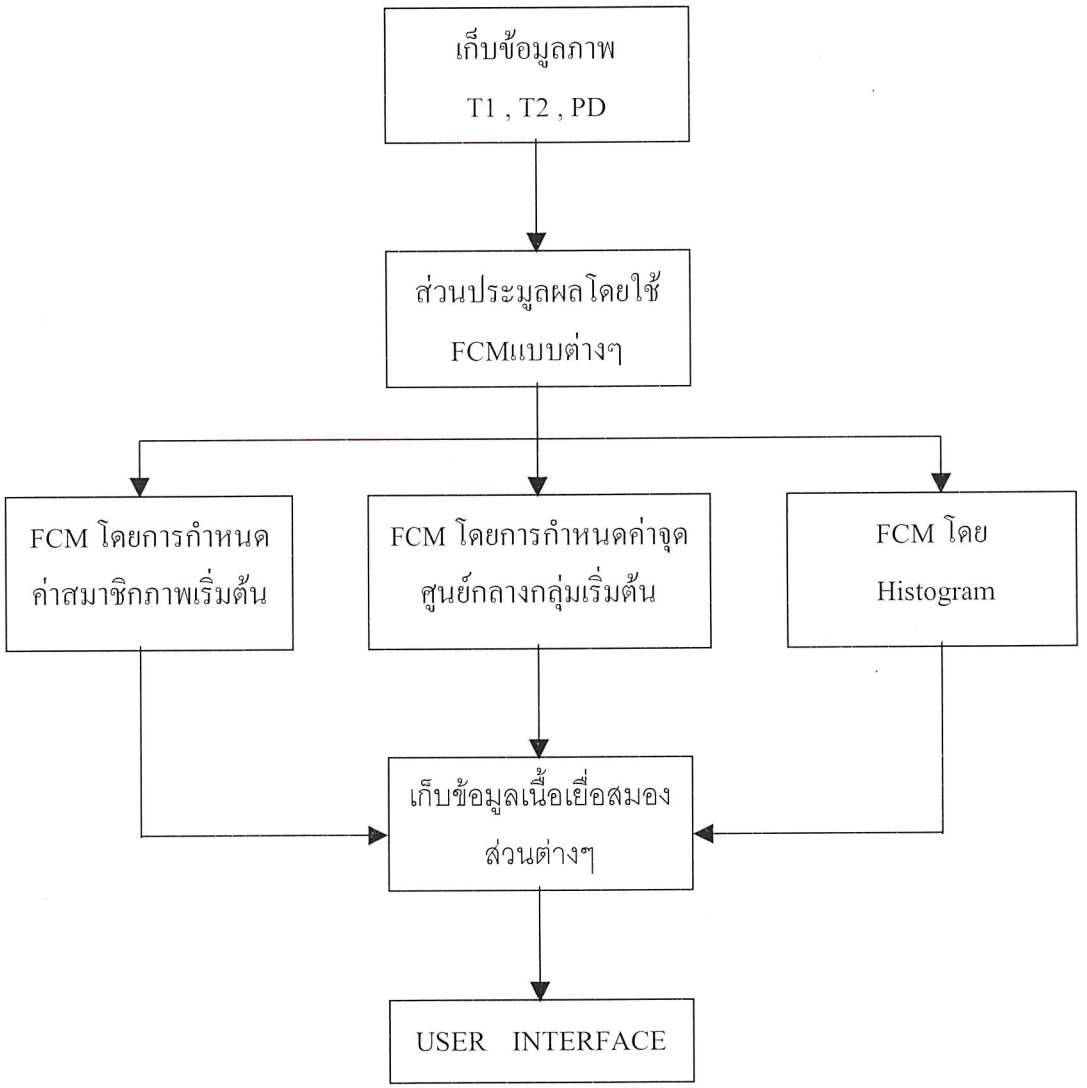
- White matter
- Gray matter
- CSF
- ไขมันใต้ผิวหนัง (ซึ่งในส่วนนี้จะใช้ในการดูรูปร่างของกะโหลก เนื่องจาก จากบทที่ 2 จะพบว่าในภาพ MRI จะไม่สามารถมองเห็นส่วนของกะโหลกนี้แต่ ที่ทำการกำหนดคลาสนี้ขึ้นเนื่องจากคิดว่าน่าจะสามารถใช้ในการดูลักษณะของกะโหลกได้บ้างไม่มากก็น้อย ซึ่งในทางปฏิบัติหากต้องการดูลักษณะของกะโหลกแล้วเราจะใช้การวินิจฉัย จากภาพ CT ซึ่งจะเห็นส่วนของกะโหลกอย่างชัดเจน ดังนั้นในปริญญานิพนธ์ฉบับนี้เมื่อมีการกล่าวถึง เนื้อเยื่อในส่วนกะโหลก จะหมายถึง ส่วนของไขมันใต้ผิวหนัง)
- ส่วนของพื้นหลังของภาพและกะโหลก เนื่องจากพื้นหลังของภาพเป็นส่วนหนึ่งของข้อมูลภาพ จึงต้องนำมาจัดให้เป็นกลุ่มหนึ่ง

หมายเหตุ ภาพ MRI ที่นำมาใช้ในปริญญานิพนธ์นี้จะเป็นภาพ MRI ในชั้นที่ 33 ซึ่งเป็นชั้นที่อยู่ ในบริเวณกึ่งกลางศรีษะ และในชั้นนี้จะเห็นเนื้อเยื่อต่างๆ ได้ชัดเจน

ตัวแปรที่จะใช้ประกอบคำอธิบายมีดังนี้

- c คือ จำนวนกลุ่มของข้อมูลที่ต้องการจะแบ่ง โดยในที่นี้ให้ $c = 5$
- n คือ จำนวนของข้อมูลทั้งหมด ซึ่งภาพ MRI ที่ใช้จะมีขนาด 256×256 นั่นก็คือ $n = 65536$
- U คือ เมตริกของค่าสมาชิกภาพ ซึ่งขนาดของ U ขึ้นอยู่กับวิธีในการประมวลผล
- V คือ เมตริกของค่าจุดศูนย์กลางข้อมูล ซึ่งขนาดของ V ขึ้นอยู่กับวิธีในการประมวลผล
- ε คือ ค่าความผิดพลาดที่ยอมรับได้ ขึ้นกับวิธีในการประมวลผลเช่นกัน

ขั้นตอนในการทำ Automatic segmentation



รูปที่ 5.5 แสดงส่วนประกอบในการทำงานทั้งหมด

ในส่วนของการเก็บข้อมูลภาพของ Fuzzy C-Mean Algorithm นี้จะใช้การเก็บข้อมูลในลักษณะแบบเดียวกับวิธี Multithreshold จึงไม่ขอนำมากล่าวอีกในหัวข้อนี้

ส่วนประมวลผล

ในระบบทางฟัซซี่ สภาวะเริ่มต้นและการกำหนดฟังก์ชั่นสมาชิกภาพ (membership) มีผลต่อการลู่เข้า(convergence) ของระบบมากซึ่งผู้ใช้งานจะต้องกำหนดให้สอดคล้องกับระบบและลักษณะการลู่เข้าที่ผู้ใช้งานต้องการ เนื่องจากสิ่งเหล่านี้จะมีผลทั้งความเร็วในการลู่เข้าและลักษณะการลู่เข้า ยกตัวอย่างเช่น ถ้าระบบมีความเป็นเชิงเส้นก็จะต้องพยายามที่จะกำหนดฟังก์ชั่นสมาชิกภาพให้มีความเป็นเชิงเส้นด้วยเพื่อให้สอดคล้องรับระบบ ถ้าผู้ใช้งานกำหนดค่าไม่สอดคล้องกับระบบ ก็จะทำให้ระบบมีลักษณะการลู่เข้าที่ผิดพลาดและใช้เวลานานหรืออาจจะไม่ลู่เข้าเลยก็ได้

จากหัวข้อ 5.2 จะพบว่ามีพารามิเตอร์ 2 ตัวที่ผู้ใช้งานต้องเป็นผู้กำหนด คือ ค่าสมาชิกภาพเริ่มต้น (initial membership value) และค่า m (weight constant) ซึ่งทั้ง 2 ค่านี้จะมีผลต่อการแบ่งกลุ่มข้อมูลมากโดยค่าสมาชิกภาพเริ่มต้นต้องกำหนดให้สอดคล้องกับสมการที่ 5.6 ซึ่งจะมีผลต่อการแบ่งกลุ่มมาก และค่า m ควรจะอยู่ในช่วง $[1.1,5]$

5.3.1) <u>การประมวลผลโดย กำหนดค่าสมาชิกภาพเริ่มต้น</u>

ในปริญญานิพนธ์นี้ได้ทำการกำหนดค่าสมาชิกภาพเริ่มต้น และค่าของ m ดังนี้

$$U^{(0)} = \begin{bmatrix} 1 & 1 & 0 & 0 & \ldots\ldots & 0 & 0 & 1 & 1 & \ldots\ldots \\ 0 & 0 & 1 & 1 & \ldots\ldots & 0 & 0 & 0 & 0 & \ldots\ldots \\ 0 & 0 & 0 & 0 & \ldots\ldots & 0 & 0 & 0 & 0 & \ldots\ldots \\ 0 & 0 & 0 & 0 & \ldots\ldots & 0 & 0 & 0 & 0 & \ldots\ldots \\ 0 & 0 & 0 & 0 & \ldots\ldots & 1 & 1 & 0 & 0 & \ldots\ldots \end{bmatrix} \qquad (5.21)$$

$$m = 2$$

$$\varepsilon = 0.005$$

โดยค่าของ ε คือค่าของความผิดพลาดที่ยอมรับได้ ซึ่งจะเป็นการตรวจสอบการลู่เข้าโดยการตรวจสอบจะเปรียบเทียบค่า $U^{(l)}$ กับ $U^{(l+1)}$ ว่าน้อยกว่า ε หรือยัง ถ้าน้อยกว่าแล้วก็ให้จบการทำงานในส่วนประมวลผล

<u>ขั้นตอนการทำงาน</u>

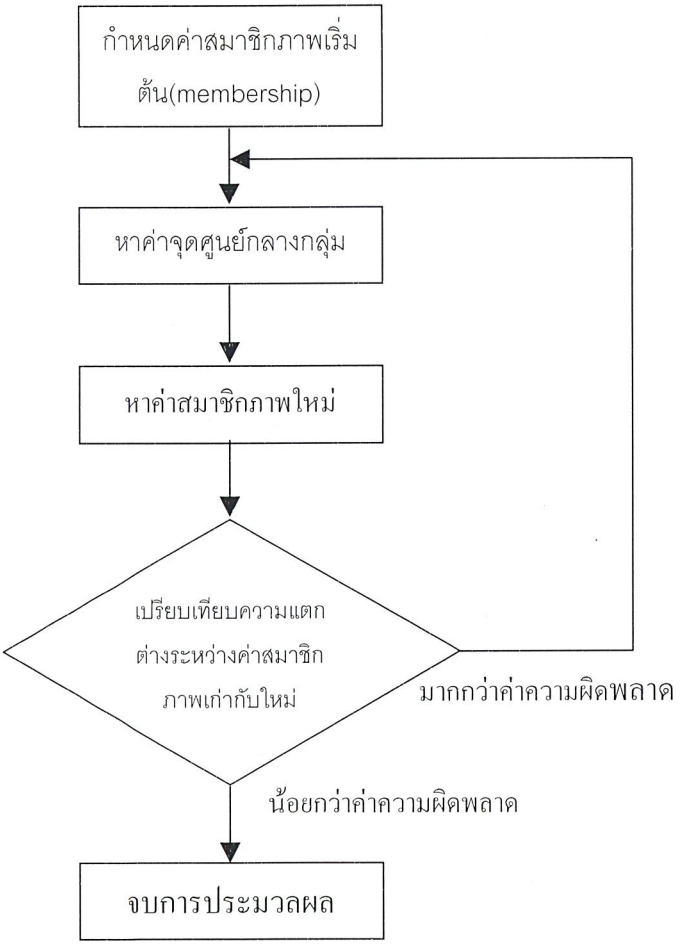
<u>ขั้นตอนที่ 1</u> ทำการกำหนดค่าสมาชิกภาพเริ่มต้น โดยใช้ $U^{(0)}$ ($U^{(l)}$) ตามสมการที่ 5.21 ค่า $m = 2$,ค่า $\varepsilon = 0.005$ จำนวนกลุ่มที่แบ่ง (c) = 5

<u>ขั้นตอนที่ 2</u> นำค่าสมาชิกเริ่มต้น ($U^{(0)} = U^{(l)}$) มาทำการหาค่าจุดศูนย์กลางกลุ่ม โดยใช้สมการที่ 5.18

ขั้นตอนที่ 3 นำค่าจุดศูนย์กลางกลุ่มที่ได้ไปหาค่าสมาชิกภาพใหม่ $(U^{(l+1)})$ โดยใช้สมการที่ 5.19

ขั้นตอนที่ 4 เปรียบเทียบค่า $U^{(l)}$ กับ $U^{(l+1)}$ ว่ามากกว่าค่าผิดพลาดที่ยอมรับได้หรือไม่ ถ้าไม่ให้กลับไปทำขั้นตอนที่ 2 โดยให้ $l = l+1$ จนกว่าค่าความแตกต่างจะน้อยกว่าค่าผิดพลาดที่ยอมรับได้

ขั้นตอนที่ 5 ถ้าค่าความแตกต่างระหว่าง $U^{(l)}$ กับ $U^{(l+1)}$ น้อยกว่าค่าความผิดพลาดที่ยอมรับได้แล้วให้จบขั้นตอนการประมวลผล จากนั้นจึงนำไปแสดงผล



รูปที่ 5.6 แสดงในการประมวลผลโดยกำหนดค่าสมาชิกเริ่มต้น

5.3.2) การประมวลผลโดยกำหนดค่าจุดศูนย์กลางกลุ่มเริ่มต้น

การประมวลผลแบบนี้จะคล้ายกับการประมวลผลโดยกำหนดค่าสมาชิกภาพเริ่มต้น แต่เราจะกำหนดค่าของจุดศูนย์กลางกลุ่มขึ้นมาก่อน นั่นคือ จากเดิมที่เรากำหนดค่าสมาชิกภาพขึ้นมาก่อน จะพบว่าการลู่เข้าของการประมวลผลค่อนข้างช้า เราจะทดลองทำการประมวลผลแบบใหม่คือ กำหนดค่าของจุดศูนย์กลางกลุ่มเริ่มต้น ซึ่งมีขั้นตอนดังนี้

ขั้นตอนที่ 1 ทำการกำหนดค่าจุดศูนย์กลางกลุ่มเริ่มต้น ดังนี้

$$V^{(0)} = \begin{bmatrix} 0 \\ 51 \\ 102 \\ 153 \\ 204 \end{bmatrix} \qquad (5.22)$$

$$m = 2$$

โดยในการกำหนดค่าจุดศูนย์กลางกลุ่ม กำหนดได้โดยการแบ่งช่วง gray level 256 ระดับ ออกเป็น 5 ช่วงเท่าๆกัน

ขั้นตอนที่ 2 นำค่าจุดศูนย์กลางกลุ่มมาหาค่าสมาชิกภาพจาก

$$U_{ik} = \frac{1}{\sum_{j=1}^{c}\left(\frac{\|X_k - V_i\|}{\|X_k - V_j\|}\right)^{\frac{2}{m-1}}}, 1 \le i \le c, 1 \le k \le n \qquad (5.23)$$

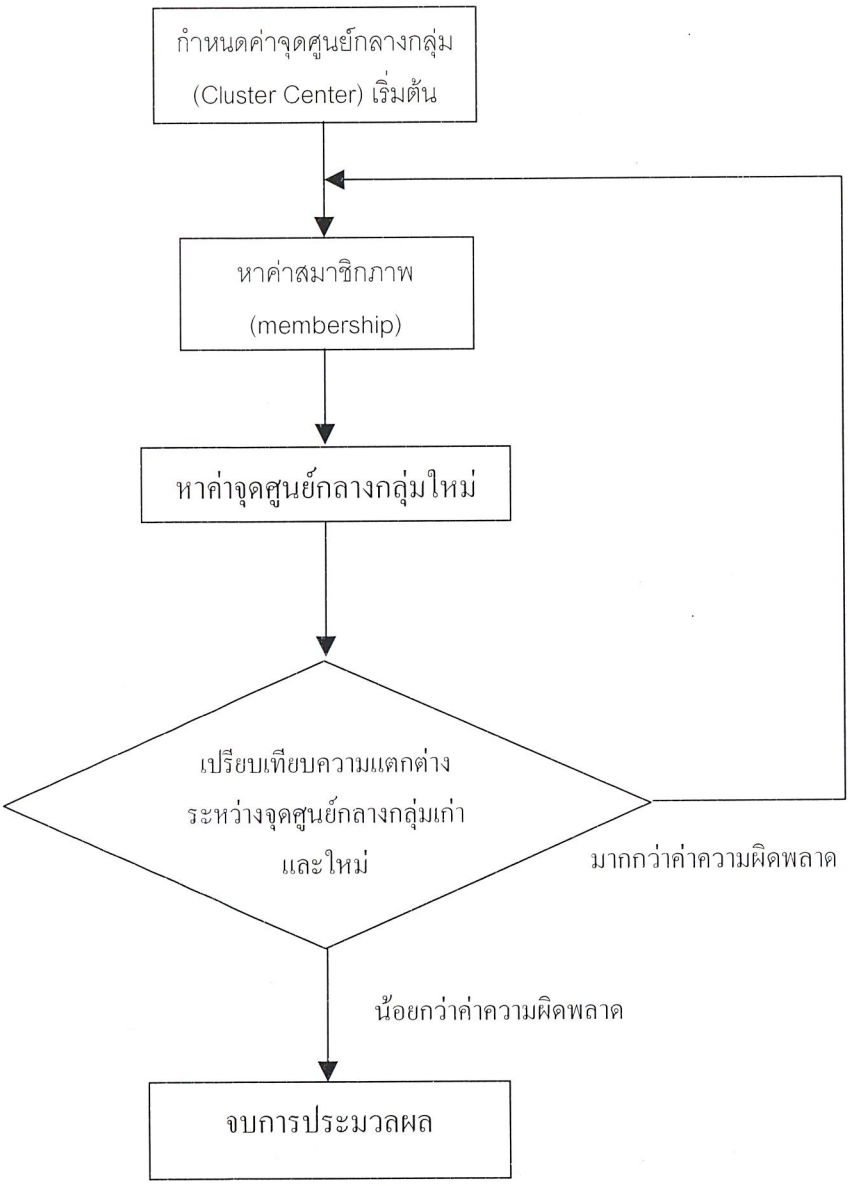
ขั้นตอนที่ 3 นำค่าสมาชิกภาพมาหาค่าจุดศูนย์กลางกลุ่มใหม่ ($V^{(l+1)}$) จาก

$$V^{(l+1)} = \frac{\sum_{k=1}^{n}(U_{ik})^m X_k}{\sum_{k=1}^{n}(U_{ik})^m} \qquad (5.24)$$

ขั้นตอนที่ 4 เปรียบเทียบ ผลต่างระหว่าง $V^{(l+1)}$ กับ $V^{(l)}$ กับค่าความผิดพลาดที่ยอมรับได้

$$\sum \left\| V^{(l+1)} - V^{(l)} \right\| < \varepsilon$$

โดยกำหนดให้ ε มีค่าเท่ากับ 0.01 หากค่าผลต่างที่ได้มากกว่าค่า ε ให้กลับไปทำขั้นตอนที่ 2 ใหม่ จนกว่าผลต่างที่ได้มีค่าน้อยกว่าค่า ε จึงเป็นอันสิ้นสุดการประมวลผล



รูปที่ 5.7 แสดงขั้นตอนการประมวลผลโดยกำหนดค่าจุดศูนย์กลางเริ่มต้น

5.3.3) การประมวลผลโดยใช้ Histogram

จาก 2 วิธีข้างต้น จะพบว่าในการหาค่าความเป็นสมาชิกภาพแต่ละครั้งนั้น เราจะต้องทำการหาค่าความเป็นสมาชิก ให้กับทุกๆ จุดภาพ จำนวน 66536 จุด(256×256) ซึ่งในกรณีที่จุดภาพมีค่าระดับเทาเท่ากันเราก็ยังต้องนำมาคิด จึงพบว่าต้องใช้เวลาในการประมวลผลส่วนนี้ค่อนข้างมาก ดังนั้นในวิธีการประมวลผลโดยใช้ Histogram นี้ เราจะทำการจัดให้จุดภาพที่มีระดับเทาเท่ากันถูกนำมาคิดแค่ครั้งเดียวเท่านั้น โดยทำการหา Histogram ของภาพๆนั้น นั่นคือจะทำให้เราหาค่าความเป็นสมาชิกภาพในแต่ละครั้งแค่ 256 จุด (ระดับเทา) ซึ่งหลังจากการทำ Histogram แล้ว จากนั้นจึงค่อยนำ FCM มาคิดตามปกติ ซึ่งมีขั้นตอนดังนี้

ขั้นที่ 1 สร้าง Histogram ซึ่ง Histogram ที่จะนำมาใช้ เป็นการแสดงความสัมพันธ์ระหว่าง ระดับ gray ต่างๆ กับ ปริมาณของจำนวน pixel ในภาพที่มีระดับ gray อยู่ในระดับนั้นๆ นั่นคือ เราจะทำการหาว่าในภาพๆ หนึ่งมีจำนวน pixel ที่อยู่ในระดับ gray เดียวกัน เป็นจำนวนเท่าใด ซึ่งจะเป็นตัวบอกว่าในแต่ละระดับเทามีจำนวนจุดภาพกี่จุดของภาพที่นำมาคิด ที่มีระดับเทานั้นโดย

กำหนด h(k) คือ histogram ที่เก็บค่าจำนวนจุดภาพในแต่ละระดับเทา

k คือ ระดับเทาแต่ละระดับ 0-255

ขั้นที่ 2 กำหนดค่าสมาชิกภาพเริ่มต้น

ให้ c คือ จำนวนกลุ่มของข้อมูลที่ต้องการแบ่ง (c = 5)

n คือ จำนวนข้อมูลทั้งหมด ในที่นี้จะหมายถึง ระดับเทาทั้งหมด (n =256)

นั่นคือ เราจะทำการกำหนดค่าสมาชิกภาพของแต่ละระดับเทา เป็นเมทริกขนาด c×n ดังนี้

$$U^{(0)} = \begin{bmatrix} 1 & 1 & 0 & 0 & ..... & 0 & 0 & 1 & 1 & ..... \\ 0 & 0 & 1 & 1 & ..... & 0 & 0 & 0 & 0 & ..... \\ 0 & 0 & 0 & 0 & ..... & 0 & 0 & 0 & 0 & ..... \\ 0 & 0 & 0 & 0 & ..... & 0 & 0 & 0 & 0 & ..... \\ 0 & 0 & 0 & 0 & ..... & 1 & 1 & 0 & 0 & ..... \end{bmatrix}_{c\times n}$$

ขั้นที่3 หาค่าศูนย์กลางกลุ่มแต่ละกลุ่มของระดับเทา

กำหนด V_i คือ ค่าศูนย์กลางกลุ่มของกลุ่มที่ i

i คือ กลุ่มใดๆ , i = 1.....5

โดย

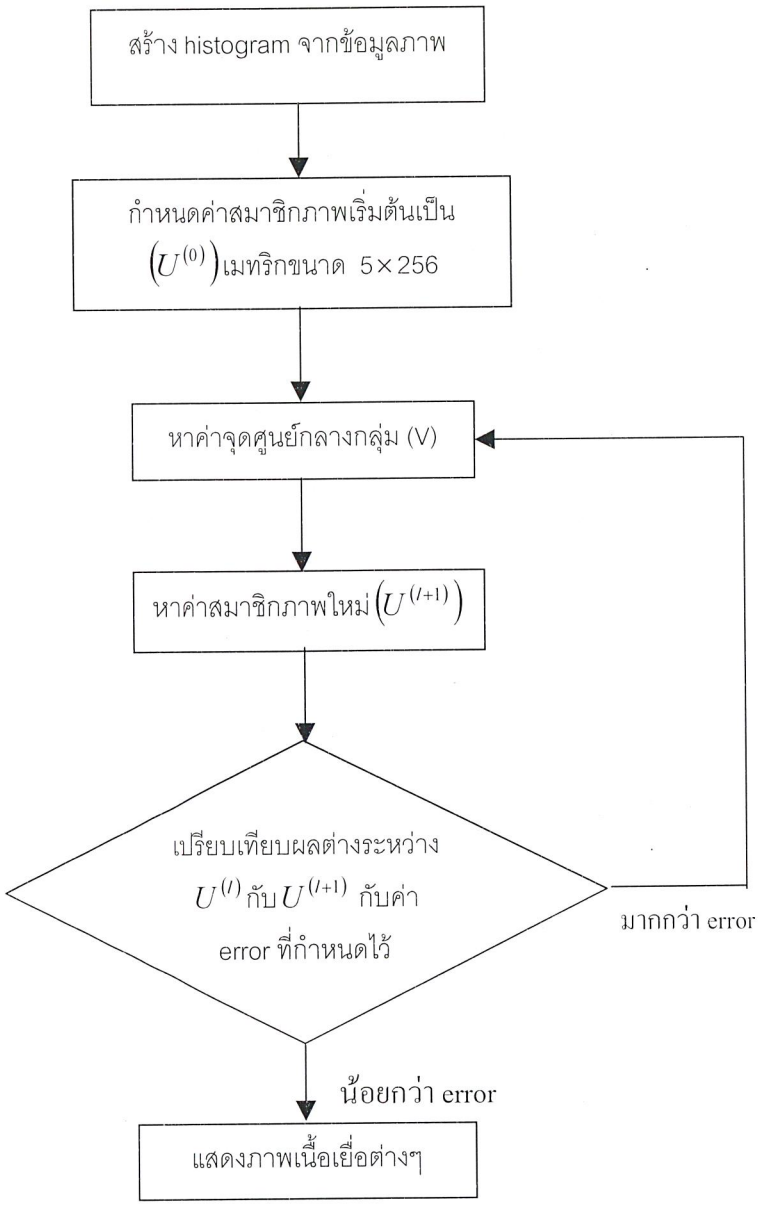
$$V_i = \frac{\sum_{0}^{255} (U_{ik})^m h(k)k}{\sum_{0}^{255} (U_{ik})^m h(k)} \tag{5.25}$$

ขั้นที่ 4 นำค่าศูนย์กลางกลุ่มที่ได้มาหาค่าสมาชิกภาพใหม่

$$u_{ik}^{(l+1)} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\left\| h(k) - v_i^{(l)} \right\|}{\left\| h(k) - v_j^{(l)} \right\|} \right)^{\frac{2}{m-1}}}, 1 \le k \le n, 1 \le i \le c \tag{5.26}$$

ขั้นที่ 5 เปรียบเทียบค่า $U^{(l)}$ กับ $U^{(l+1)}$ ว่ามากกว่าค่าผิดพลาดที่ยอมรับได้หรือไม่ (ในวิธีนี้ได้กำหนดค่า ε ไว้เท่ากับ 0.000001) ถ้าไม่ ให้กลับไปขั้นตอนที่ 3 โดยให้ $l = l + 1$ จนกว่าค่าความแตกต่างจะน้อยกว่าค่าผิดพลาดที่ยอมรับได้

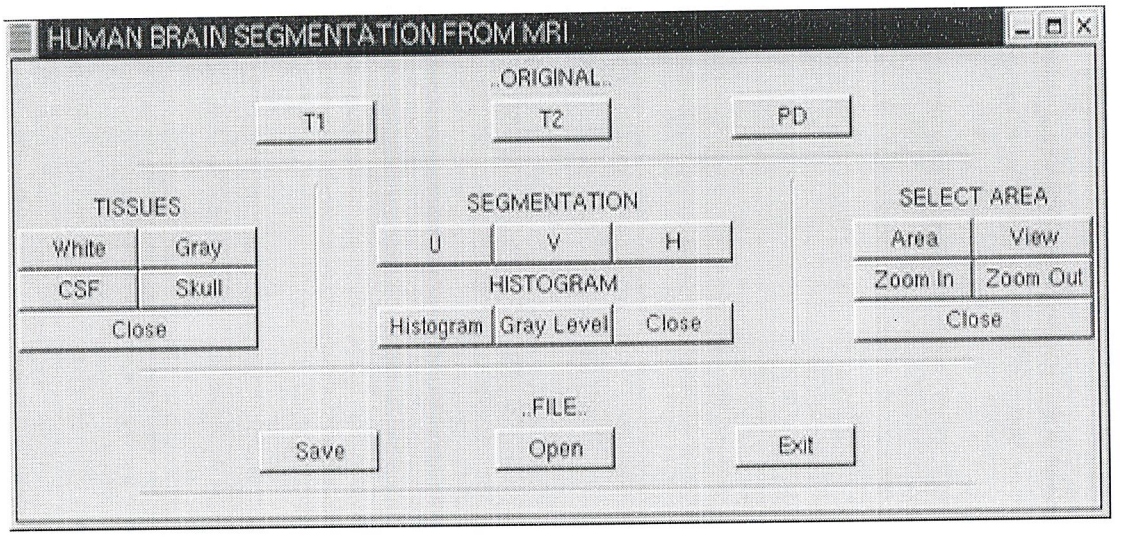
ขั้นที่ 6 ถ้าค่าความแตกต่างระหว่าง $U^{(l)}$ กับ $U^{(l+1)}$ น้อยกว่าค่าความผิดพลาดที่ยอมรับได้แล้วให้ก็จะจบขั้นตอนการประมวลผล



รูปที่ 5.8 แสดงการใช้ FCM โดยการใช้ histogram เข้ามาช่วย

5.4 USER INTERFACE

ในปริญญานิพนธ์ฉบับนี้ ได้อำนวยความสะดวกในการทำการ Segmentation ด้วยการเพิ่มส่วนในการทำงานติดต่อกับผู้ใช้ (User Interface)



รูปที่ 5.9 แสดงส่วนของ User Interface

โดยส่วนประกอบต่างๆ ของ User Interface ประกอบด้วยดังนี้

1) ส่วนของรูปต้นแบบ (ORIGINAL) : ใช้สำหรับดูภาพ MRI ที่นำมาทำการ Segmentation ซึ่งจะประกอบไปด้วยภาพ MRI 1 ชั้นโดยใน 1 ชั้นนี้จะประกอบด้วย Slice ทั้งสิ้น 3 Slices อันได้แก่ T1 , T2 และ PD โดยหากต้องการดูภาพ slice ใดก็กดที่ปุ่มนั้น

2) ส่วนของเนื้อเยื่อสมองส่วนต่างๆ (TISSUES) : ใช้สำหรับดูภาพของเนื้อเยื่อสมองส่วนต่างๆ ซึ่งได้จากการทำ Segmentation แล้ว ซึ่งเนื้อเยื่อต่างๆ ได้แก่ White (White Matter) , Gray (Gray matter) , CSF (CerebroSpinal-Fluid) , Skull (เป็นส่วนของไขมันที่อยู่ใต้ผิวหนังซึ่งเราได้อธิบายไปแล้วว่า ในปริญญานิพนธ์ฉบับนี้ จะนำส่วนของไขมันใต้ผิวหนังมาใช้ในการพิจารณาแทนคลาสของกะโหลกซึ่งจะไม่สามารถมองเห็นได้ในภาพ MRI) โดยหากต้องการดูภาพของเนื้อเยื่อส่วนใดก็กดที่ปุ่มนั้น นอกจากนี้ยังมีปุ่ม close ที่ใช้ในการปิดภาพของ tisues

3) ส่วนในการประมวลผล(Segmentation) : เป็นส่วนในการทำการ Segmentation โดยจะมีวิธีให้เลือกใช้ 3 วิธีได้แก่

- ปุ่ม "U" : เป็นการ Segmentation โดยใช้ Fuzzy C-Mean algorithm ด้วยการกำหนดค่าสมาชิกภาพ (membership)เริ่มต้น

- ปุ่ม "V" : เป็นการ Segmentation โดยใช้ Fuzzy C-Mean algorithm ด้วยการกำหนดค่าจุดศูนย์กลางกลุ่มข้อมูล(Cluster Center)เริ่มต้น

- ปุ่ม "H" : เป็นการ Segmentation โดยใช้ Fuzzy C-Mean algorithm ด้วยการใช้ Histogram เข้ามาช่วย

4) ส่วนในการดูฮิสโตแกรม (Histogram) , ระดับเทา (Gray Level) และตำแหน่งที่ mouse ชี้อยู่ในภาพ : ใช้ในการดูกราฟที่แสดงความสัมพันธ์ระหว่างระดับเทาระดับต่างๆ กับจำนวนของจุดภาพที่มีระดับเทาอยู่ในระดับเทานั้นๆ (ปุ่ม Histogram)และปุ่ม close จะใช้สำหรับปิดหน้าต่างของ Histogram นอกจากนี้ยังใช้ในการดูระดับของแต่ละจุดของภาพตามตำแหน่งของ mouse ที่ผู้ใช้ทำการเลื่อนและยังสามารถบอกตำแหน่งของ mouse ที่เคลื่อนที่อยู่บนภาพได้ โดยอ้างอิงกับหน้าต่างนั้นๆ (ปุ่ม Gray Level)

5) ส่วนที่ใช้ในการเลือกบางส่วนของภาพ (SELECT AREA) : ใช้ในการตัดบางส่วนที่สนใจของภาพออกมาเพื่อทำให้ง่ายต่อการพิจารณา นอกจากนี้เรายังสามารถขยายและย่อขนาดของส่วนของภาพที่ได้ทำการเลือกมาได้อีกด้วย ซึ่งปุ่มต่างๆจะทำหน้าที่ดังนี้

ปุ่ม Select จะใช้ในการเลือกพื้นที่ที่ต้องการในลักษณะของกรอบสี่เหลี่ยมเมื่อได้พื้นที่ที่ต้องการแล้วให้คลิกขวาเพื่อจบการเลือกพื้นที่ หลังจากนั้น ปุ่ม View ใช้ในการดูพื้นที่ที่ได้เลือกไว้ ปุ่ม Zoom In , Zoom Out ใช้ในการขยายและย่อพื้นที่ที่เลือกไว้ตามลำดับปุ่ม close ใช้สำหรับปิดหน้าต่างของพื้นที่ที่เลือกไว้

6) File : เป็นส่วนที่ใช้เกี่ยวกับการจัดการของภาพ ซึ่งได้แก่ ปุ่ม Save จะใช้ในการ Save ภาพที่ต้องการ , ปุ่ม Open ใช้ในการเปิดภาพที่ต้องการ และปุ่ม Exit ใช้สำหรับการออกจากโปรแกรม

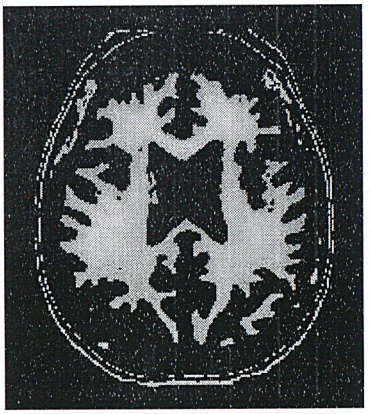
บทที่ 6

ผลการทดลอง

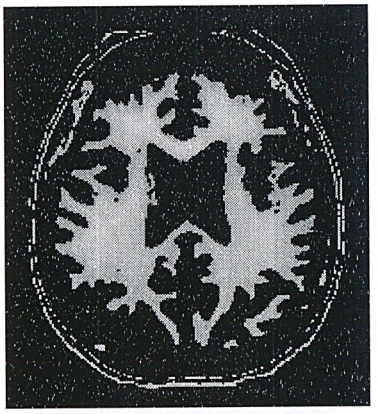
ภาพที่นำมาใช้เป็นต้นแบบในการทำ Human Brain Segmentation เป็นภาพ T1 , T2 ,PD จาก slice ชั้นที่ 33

6.1 เนื้อเยื่อสมองต่างๆที่ได้จากแต่ละวิธี

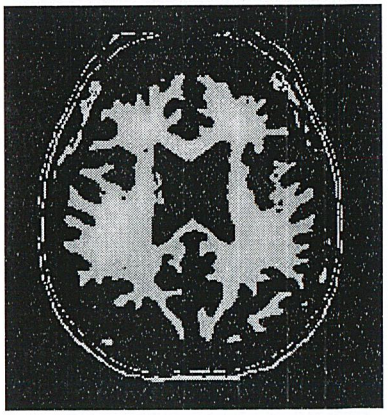
6.1.1) เนื้อเยื่อ White Matter



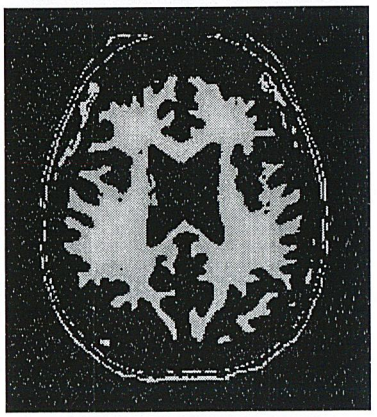
Multithreshold



FCM-Initial U



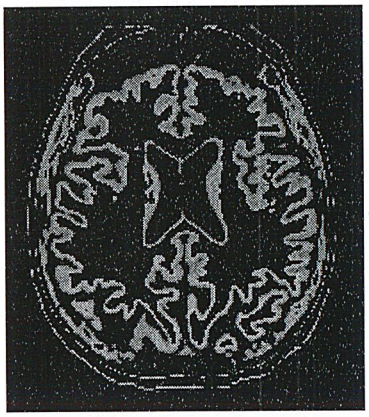
FCM-Initial V



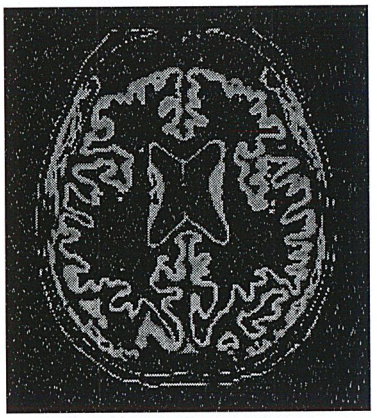
FCM- Histogram

รูปที่ 6.1 เปรียบเทียบเนื้อเยื่อ White Matter ที่ได้จากวิธีต่างๆ

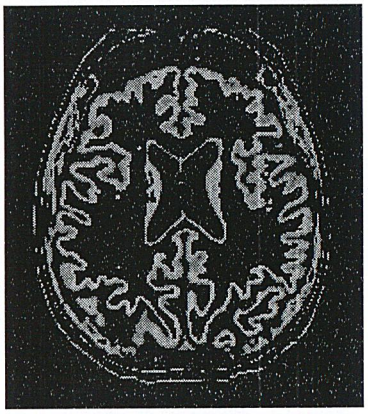
6.1.2) เนื้อเยื่อ Gray Matter



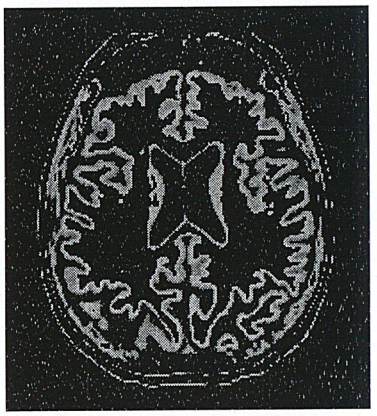
Multithreshold



FCM-Initial U



FCM-Initial V



FCM- Histogram

รูปที่ 6.2 เปรียบเทียบเนื้อเยื่อ Gray Matter ที่ได้จากวิธีต่างๆ

6.1.3) CSF (Cerebrospinal-fluid)

Multithreshold

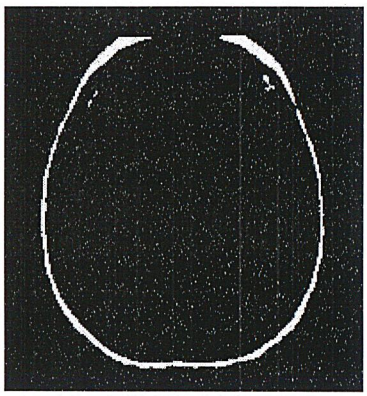
FCM-Initial U

FCM-Initial V

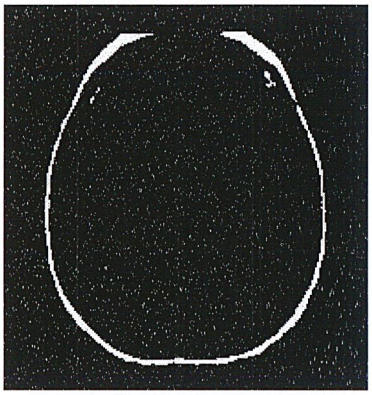
FCM- Histogram

รูปที่ 6.3 เปรียบเทียบ CSFที่ได้จากวิธีต่างๆ

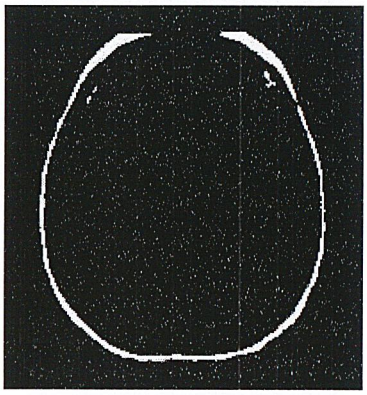
6.1.4) Skull (Fat)



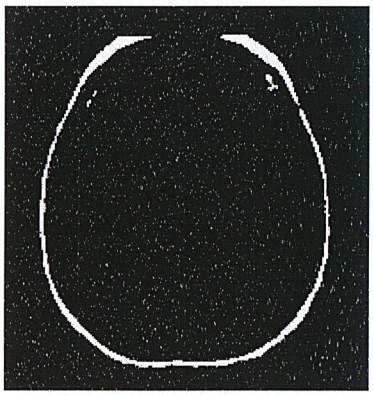
Multithreshold



FCM-Initial U



FCM-Initial V



FCM- Histogram

รูปที่ 6.4 เปรียบเทียบ skull (fat) ที่ได้จากวิธีต่างๆ

6.2 ช่วงระดับเทา(Gray Level) ของเนื้อเยื่อแต่ละชนิด

	White Matter	Gray Matter	CSF	Skull(Fat)
M	142-207	89-142	190-255	207-255
U	145-207	92-142	193-255	210-255
V	145-207	92-142	196-255	210-255
H	145-198	92-142	204-255	210-255

ตารางที่ 6.1 แสดงช่วงของระดับเทา (Gray Level) ของเนื้อเยื่อส่วนต่างๆ ในแต่ละวิธี

6.3 เวลาที่ใช้ในการประมวลผลของแต่ละวิธี

	เวลาที่ใช้
M	25 นาที
U	7 .56 นาที
V	1.37 นาที
H	15.62 วินาที

ตารางที่ 6.2 แสดงเวลาในการประมวลผลของแต่ละวิธี

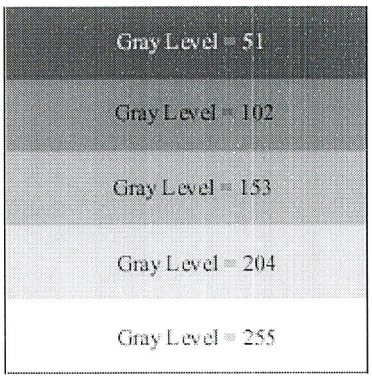
หมายเหตุ : คอมพิวเตอร์ที่ใช้ CPU Pentium -166 MHz

6.4 ค่าจุดศูนย์กลางกลุ่มข้อมูลแต่ละกลุ่มของ Fuzzy C-Mean แต่ละแบบ

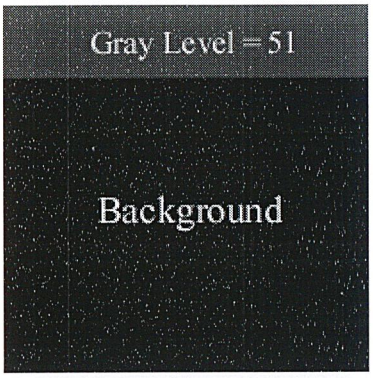
	U	V	H
ค่าศูนย์กลางของกลุ่มที่ 1	0.553	0.568	0.568
ค่าศูนย์กลางของกลุ่มที่ 2	61.559	62.203	62.179
ค่าศูนย์กลางของกลุ่มที่ 3	118.384	118.975	118.95
ค่าศูนย์กลางของกลุ่มที่ 4	169.918	170.089	170.087
ค่าศูนย์กลางของกลุ่มที่ 5	248.113	248.171	248.167

ตารางที่ 6.3 แสดงค่าศูนย์กลางกลุ่ม(Cluster Center) ของแต่ละกลุ่ม ด้วยวิธีต่างๆ

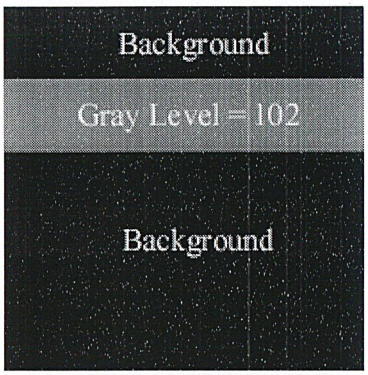
6.5 การใช้ FCM ในการแบ่งกลุ่มของภาพที่มีระดับเทาเพียง 5 ระดับ



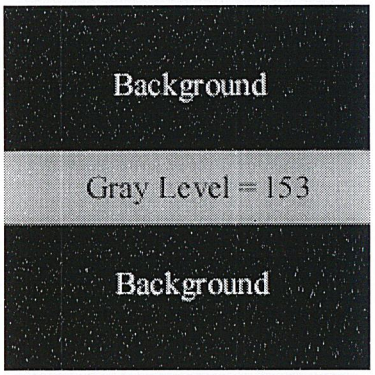
(ก)



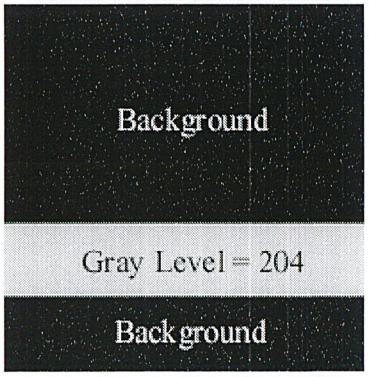
(ข)



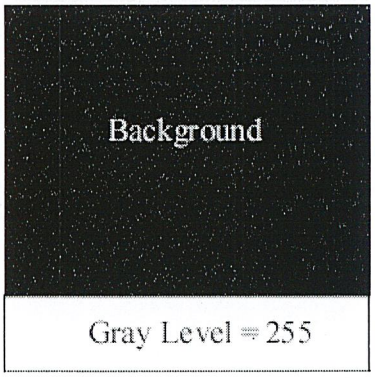
(ค)



(ง)



(จ)



(ฉ)

รูปที่ 6.5 (ก) รูปต้นแบบที่มีระดับเทาเพียง 5 ระดับดังแสดงในรูป (ข)-(ฉ) แสดงกลุ่มของข้อมูลกลุ่มต่างๆ ที่ได้จากการ Segmentation

บทที่ 7

สรุปและวิเคราะห์ผลการทดลอง

จากผลการทดลองในบทที่ 6 พบว่าเนื้อเยื่อสมองแต่ละส่วนที่ได้จากวิธีการคัดแยกต่างๆกัน เมื่อมองด้วยตาเปล่า จะให้ผลออกมาใกล้เคียงกันมาก แต่จริงๆแล้วช่วงของระดับเทาของเนื้อเยื่อสมองแต่ละชนิดไม่เท่ากันเลยทีเดียว ซึ่งเห็นได้จากตารางที่ 6.1 ซึ่งแสดงช่วงของระดับเทา(Gray Level) ของเนื้อเยื่อสมองแต่ละส่วนที่ได้จากวิธีในการประมวลผลแบบต่างๆ แต่ความแตกต่างของช่วงของระดับเทาดังกล่าวของแต่ละวิธี ยังกล่าวได้ว่าแตกต่างกันน้อย จะมีเฉพาะบางวิธีที่ให้ช่วงระดับเทาของเนื้อเยื่อที่แตกต่างจากวิธีอื่นอย่างชัดเจน เช่น ส่วนของ White Matter และ CSF จาก Fuzzy C-Mean ที่ใช้ Histogram ที่ให้ช่วงของระดับเทาที่ต่างจากวิธีอื่น นอกจากนี้การใช้ Fuzzy C-Mean Algorithm แบบต่างๆยังให้ผลที่ใกล้เคียงกัน ทั้งนี้เนื่องจากค่าของจุดศูนย์กลางกลุ่มที่ได้ (ดังตารางที่ 6.3) ของทั้ง 3 แบบได้ค่าที่ใกล้เคียงกันนั่นเอง ดังนั้นเราจึงยังไม่เราจึงยังไม่สามารถสรุปได้ว่าวิธีใดที่เหมาะกับการนำไปใช้มากกว่ากัน จึงจำเป็นต้องอาศัยปัจจัยอื่นๆเข้ามาช่วยในการพิจารณา

เวลาเป็นอีกปัจจัยหนึ่งซึ่งจำเป็นต้องสนใจอย่างมาก เนื่องจากเวลาจะทำให้เรารู้ว่าวิธี การใดที่เหมาะสำหรับการนำไปใช้งานจริงในกรณีที่แต่ละวิธีให้ผลลัพธ์ออกมาเหมือนๆกัน โดยจากตารางที่ 7.2 แสดงให้เห็นว่าเวลาที่ใช้ในการประมวลผลแต่ละวิธีแตกต่างกันมาก คือ วิธี Multithreshold จะใช้เวลาในการประมวลผลนานที่สุด รองลงมาก็เป็น Fuzzy C-Mean แบบกำหนดค่าสมาชิกภาพเริ่มต้น , แบบกำหนดค่าจุดศูนย์กลางกลุ่มเริ่มต้น และแบบ Histogram จะใช้เวลาในการประมวลผลสั้นที่สุด ต่อไปเป็นการวิเคราะห์ถึงสาเหตุที่ทำให้แต่ละวิธีใช้เวลาในการประมวลผลแตกต่างกันมาก

Multithreshold วิธีนี้ทำการคัดแยกโดยการกำหนดจุดแบ่งหลายๆจุด ตามแต่จำนวนกลุ่มที่เราแบ่ง โดยอัลกอริธึมเป็นการหาค่าจุดแบ่ง(Threshold)ที่ทำให้ได้ค่าความแปรปรวน(σ^2) รวมมากที่สุด ซึ่งหมายถึงเราต้องทำการหาค่า σ^2 ให้ครบทุกช่วงของระดับเทา(ดังที่ได้เสนอช่วงในการหาค่าแล้วในบทที่ 4) จากนั้นจึงมาดูว่าจุดแบ่งใดที่ทำให้ได้ค่าความแปรปรวนรวมมากที่ สุด ซึ่งแน่นอนว่าต้องใช้เวลาในการประมวลผลนานมาก นอกจากนี้ความเร็วในการประมวลผลด้วยวิธี Mulitthreshold ยังขึ้นอยู่กับ 1) ขนาดของรูปที่นำมาทำการคัดแยก 2) จำนวนกลุ่มข้อมูลที่ ต้องการคัดแยก ดังนั้นเราจึงพบว่าวิธีการประมวลผลแบบ Multithreshold อาจไม่เหมาะสำหรับการนำมาใช้

ในการคัดแยกออกเป็นหลายๆกลุ่ม แต่อาจจะเหมาะสำหรับการคัดแยกที่ต้องการแยกกลุ่มของข้อมูลออกเป็นจำนวนกลุ่มซึ่งไม่มากนัก

ส่วนวิธีการที่ใช้เวลาในการประมวลผลรองลงมา คือ Fuzzy C-Mean แบบกำหนดค่าสมาชิกภาพเริ่มต้น ซึ่งวิธีนี้เราก็เป็นอีกวิธีหนึ่งที่ต้องใช้เวลาในการประมวลผลค่อนข้างนานเนื่องจากในแต่ละรอบของการประมวลผลเราต้องทำการหาค่าสมาชิกให้กับทุกจุดภาพ (pixel) ในภาพ คือ 66,536 จุดภาพ ดังนั้นจึงทำให้ต้องใช้เวลาในการประมวลผลค่อนข้างนาน ต่อมาเป็นวิธี Fuzzy C-Mean แบบกำหนดค่าศูนย์กลางกลุ่มเริ่มต้น ซึ่งวิธีนี้จะคล้ายกับแบบกำหนดค่าสมาชิกเริ่มต้น แต่เราสามารถทำการกำหนดค่าจุดศูนย์กลางกลุ่มให้ใกล้เคียงกับค่าที่ถูกต้องได้ ซึ่งทำให้การวนทำซ้ำของ Fuzzy C-Mean น้อยลงซึ่งจะดีกว่าแบบกำหนดค่าสมาชิกภาพเริ่มต้นที่ต้องวนทำซ้ำจนกว่าจะได้ค่าศูนย์กลางกลุ่มและค่าสมาชิกภาพที่เหมาะสม ซึ่งจะทำให้วิธีการนี้ใช้เวลาในการประมวลผลน้อยกว่า 2 แบบแรก สุดท้ายเป็นวิธีที่ได้รับการพัฒนาให้ใช้เวลาในการประมวลผลน้อยที่สุด นั่นคือวิธี Fuzzy C-Mean แบบใช้ Histogram เนื่องจาก Fuzzy C-Mean Algorithm ทั้งแบบที่ทำการกำหนดค่าสมาชิกเริ่มต้นและ แบบกำหนดค่าจุดศูนย์กลางกลุ่มเริ่มต้นเราจะต้องทำการหาค่าความเป็นสมาชิกให้กับทุกๆ จุดภาพ จำนวน 66536 จุด (256×256) ในแต่ละรอบของการประมวลผล(จนกว่าจะน้อยกว่าค่า error ที่กำหนดไว้) ซึ่งในกรณีที่ จุดภาพมีค่าระดับเทาเท่ากันเราก็ยังต้องนำมาพิจารณาทุกๆ จุดซึ่งแน่นอนว่า จะต้องใช้เวลาในการประมวลผลค่อนข้างมาก ซึ่งจะต่างจาก Fuzzy C-Mean Algorithm แบบใช้ Histogram ที่ เราจะทำการจัดให้จุดภาพที่มี ระดับเทาเท่ากันถูกนำมาคิดแค่ครั้งเดียวเท่านั้น โดยทำการหา Histogram ของภาพๆนั้น นั่นคือจะทำให้เราหาค่าความเป็นสมาชิกภาพในแต่ละครั้งแค่ 256 จุด (ระดับเทา) เท่านั้น ซึ่งทำให้ใช้เวลาน้อยลงมาก

จากนั้นเรามาพิจารณาถึงปัจจัยที่มีผลต่อการคัดแยกกลุ่มของเนื้อเยื่อสมอง โดยใช้ Fuzzy C-Mean Algorithm ทั้งในด้านความเร็วในการประมวลผล และ ด้านของกลุ่มของเนื้อเยื่อสมองที่ได้ พบว่ามีปัจจัยต่าง ๆ ดังนี้

- **ค่าจุดศูนย์กลางกลุ่มของข้อมูล (Cluster Center)** : จากตารางที่ 6.3 พบว่าค่าศูนย์กลางกลุ่มข้อมูลที่ได้จากการประมวลผลทั้ง 3 แบบ จะมี ค่าที่ใกล้เคียงกันมาก นี่ก็เป็นสาเหตุหนึ่งที่ทำให้ช่วงของระดับเทาของเนื้อเยื่อแต่ละส่วนที่ได้จากการคัดแยกในแต่ละแบบได้ ค่าที่ใกล้เคียงกัน เนื่องจากค่าศูนย์กลางกลุ่มข้อมูลจะเป็นตัวกำหนดค่าความเป็นสมาชิกที่แต่ละระดับเทามีต่อค่าศูนย์กลางกลุ่มข้อมูลนั้น

- ค่าความผิดพลาดที่กำหนดไว้(error) : ค่าความผิดพลาดที่ได้กำหนดไว้หากกำหนดไว้ไม่เหมาะสม อย่างเช่น มากเกินไป ก็อาจทำให้การแบ่งกลุ่มของข้อมูลอาจยังไม่ถูกต้องเนื่องจาก Fuzzy

C-Mean Algorithm ก็คือการวนทำซ้ำเพื่อหาค่าความเป็นสมาชิกและค่าจุดศูนย์กลางกลุ่มที่เหมาะสม นอกจากนี้ ค่าความผิดพลาดนี้ยังมีผลต่อเวลาที่ใช้ในการประมวลผลอีกด้วย อย่างเช่น หากเรากำหนดค่าความผิดพลาดไว้น้อยมากๆ ก็จะต้องทำการวนซ้ำเพื่อหาค่าค่าความเป็นสมาชิกและค่าจุดศูนย์กลางกลุ่มที่เหมาะสมนานยิ่งขึ้น

จากการเปรียบเทียบวิธีในการประมวลผลแบบต่างๆ เราสรุปได้ว่า วิธีการคัดแยกโดยใช้ Multithreshold ให้ผลสำเร็จตามวัตถุประสงค์ของปริญญานิพนธ์นี้ นั่นคือ สามารถคัดแยกเนื้อเยื่อสมองออกเป็นเนื้อเยื่อส่วนต่างๆ ได้ แต่จะไม่เหมาะสำหรับการนำไปใช้งานใน ทางปฏิบัติ เนื่องจากต้องใช้เวลาในการประมวลผลนานเกินไป อันเนื่องมาจาก กลุ่มของข้อมูลที่ ถูกกำหนดให้คัดแยกมีจำนวนมาก ส่วนการนำ Fuzzy C-Mean Algorithm มาใช้ในการคัดแยกเนื้อเยื่อสมองจะเห็นว่าเหมาะสำหรับการนำไปใช้งานจริงมากกว่า เนื่องจากใช้เวลาในการประมวลผลที่น้อยมากเมื่อเทียบกับวิธี Multithreshold นอกจากนี้เรายังทำการพัฒนา ให้ สามารถใช้เวลาในการประมวลผลที่สั้นลงได้ ซึ่งในตอนแรกของการเริ่มต้นทำปริญญานิพนธ์ได้ ใช้ Fuzzy C-Mean Algorithm ในแบบกำหนดค่าความเป็นสมาชิกเริ่มต้น ซึ่งใช้เวลาในการประมวลผลที่สั้นกว่า Multithreshold และจากนั้นได้มีการนำวิธีการกำหนดค่าจุดศูนย์กลางกลุ่มเริ่มต้นมาใช้ซึ่งทำให้ใช้เวลาในการประมวลผลที่สั้นลงไปอีก จนกระทั่งมาถึงวิธีสุดท้าย นั่นคือการนำ Histogram มาใช้ในการวิเคราะห์ ซึ่งใช้เวลาในการประมวลผลที่สั้นที่สุด

เอกสารอ้างอิง

1. Byron S.Gottfried แปลและเรียบเรียงโดย ศรัณย์ อินทโกสุม , การเขียนโปรแกรมด้วยภาษาซี , MC Graw-Hill ; Inc , 1990
2. Nabajyoti Barkakati , X Window System™ Programing(2nd edition) , SAMS Publishing , 1994
3. J.R.Parker , Algorithms for image processing and computer vision , Wwiley Computer Publishing , 1996
4. Nobuyuki Otsu , " A Threshold Selection Method from Gray-Level Histogram" , IEEE Trans. on Systems , MAN , and Cybernetics , vol.SMC-9 , NO.1 , January 1979
5. Chunlin Li , Dmitry B.Goldgof , and Lawrance O.Hall ,"Knowladge-Based Classification and Tissue Labeling of MR Images of Human Brain" , IEEE Trans.on Medical Imaging , Vol. 12 , No.4 , December , 1993
6. Ye Xiu Qing , Huang Zhen Hua , Xiao Qiang ,"Histogram Based Fuzzy C-Mean Algorithm For Image Segmentation" , Zhejiang University ,Hangzhou , P.R. China
7. James C.Bezdek ,"Clustering In Banach Spaces" ,Division of Computer Science , University of West Florida ,USA

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยความอนุเคราะห์จากหลายๆท่าน โดยเฉพาะอย่างยิ่ง รศ.ดร.มนัส สังวรศิลป์ อาจารย์ที่ปรึกษาที่ได้กรุณาให้คำปรึกษาและคำแนะนำต่างๆ ที่เป็นประโยชน์อย่างยิ่งต่อปริญญานิพนธ์นี้ ขอขอบคุณ คุณประเสริฐ สำหรับแรงบันดาลใจเริ่มต้น , คุณนนท์ สำหรับวิชาความรู้ต่างๆที่ถ่ายทอดให้น้องคนนี้ , คุณวรเทพ สำหรับแนวคิดในการแก้ปัญหาต่างๆ , เพื่อนๆ พี่ๆ น้องๆ ทุกท่านในห้อง B204 ตลอดจน เพื่อนๆ ภาควิชาอิเล็กทรอนิกส์ทุกท่านที่ร่วมทุกข์ร่วมสุขกันมาโดยตลอด สุดท้ายนี้ ขอบพระคุณ คุณพ่อคุณแม่ สำหรับความรัก ,ความเข้าใจและความเชื่อใจที่มีให้กับลูกคนนี้เสมอมา

(ไตรรัตน์ ตั้งพิทยาเวทย์).....

(นายไตรรัตน์ ตั้งพิทยาเวทย์)

ภาคผนวก ก

โปรแกรม segmentation โดยใช้ multithreshold

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
#include<Imlib.h>
#include<X11/keysym.h>
  Display *d;
  Window w0,w1,w2,w3,w4,w5;
  unsigned long black,white;
  ImlibData *id;
  ImlibImage *im,*ip,*ip1,*ip2,*ip3,*ip4,*ip5;
  Pixmap p,m,p1 ;
  int w,h,i,j,l,size,q;
  unsigned char *value,*b1,*b2,*b3,*b4,*b5;
  XEvent e;
  unsigned int code ;
  int x,n;
  long a,b,c,r,s,f,g,o;
  float u0,u1,u,u2,ut,u3,u4,histo[256];
  float V0,V1,Vb,Vs;
  long t1,t2,t3,t4;
  float Pu0,Pu1,Pu2,Pu3,Pu4;
void mean(long k1,long k2,long k3,long k4)
{
  u=u0=u1=u2=u3=u4=0;
  Pu0=Pu1=Pu2=Pu3=Pu4=0;
  a=b=c=f=o=0;
  /*sigma*/
  for( a = 0; a<=k1 ; a++)
     Pu0 += histo[a];  for( b = k1+1 ; b<=k2 ; b++)
     Pu1 += histo[b];
 for( c = k2+1 ; c<=k3 ; c++)
     Pu2 += histo[c];for( f = k3+1 ;f<=k4 ; f++)
```

```
        Pu3 += histo[f];
    for( o = k4+1 ;o<256 ; o++)
        Pu4 += histo[o];
    /*  mean*/
    u=0;
    for( a = 0; a<=k1 ; a++)
            u += a*histo[a];
    u0 = u/Pu0;
     u=0;
    for( b = k1+1 ; b<=k2 ; b++)
            u += b*histo[b];
    u1 = u/Pu1;
    u=0;
    for( c = k2+1 ; c<=k3 ; c++)
        u += c*histo[c];
    u2 = u/Pu2;
    u=0;
    for( f = k3+1 ; f<=k4 ; f++)
        u += f*histo[f];
    u3 = u/Pu3;
    u=0;
    for( o = k4+1 ; o<256 ; o++)
         u += o*histo[o];
   u4 = u/Pu4;
}
void max()
{
   float ut,uk;
    Vb = 0;
    Vb = (Pu0*(u0-ut)*(u0-ut)) + (Pu1*(u1-ut)*(u1-ut))
        + (Pu2*(u2-ut)*(u2-ut))+ (Pu3*(u3-ut)*(u3-ut))
        + (Pu4*(u4-ut)*(u4-ut));
}
int main(int argc , char **argv)
{
```

```
 if(argc<=1)
 {
   printf("Usage :\n %s image_file\n",argv[0]);
   exit(1);
 }
 d = XOpenDisplay (NULL);
 black = BlackPixel(d,0);
 white = WhitePixel(d,0);
 id = Imlib_init(d);
 im = Imlib_load_image(id,argv[1]);
  w = im->rgb_width ;
  h = im->rgb_height;
 size = w*h;
 value =(unsigned char *)malloc(size*3*sizeof (unsigned char));
 b1 =(unsigned char *)malloc(size*3*sizeof (unsigned char));
 b2 =(unsigned char *)malloc(size*3*sizeof (unsigned char));
 b3 =(unsigned char *)malloc(size*3*sizeof (unsigned char));
 b4 =(unsigned char *)malloc(size*3*sizeof (unsigned char));
 b5 =(unsigned char *)malloc(size*3*sizeof (unsigned char));
 value = im->rgb_data;
 im = Imlib_create_image_from_data(id,value,NULL,w,h);
 Imlib_render(id,im,w,h);
 p = Imlib_move_image(id,im);
 m = Imlib_move_mask(id,im);
 XSetWindowBackgroundPixmap(d,w0,p);
for (i=0;i<256;i++)
    histo[i]  =0;
 for (i=0;i < size*3; i+= 3)
    histo[value[i]] += 1;
 for (i=0;i<256;i++)
  histo[i] /= (float)size;
ut =0;
for(j=1;j<256;j++)
  ut += j*histo[j];
Vs = t1 = t2 =r=t3=0;
```

```
for(i=1;i<250;i++)
{
 for(j=i+2;j<252;j++)
   {
        for(l=j+2;l<254;l++)
     {
          for(q=j+2;q<256;q++)
          {
            mean(i,j,l,q);
            max();
            if ( Vb > Vs)
            {
             Vs = Vb;
             t1 = i;
             t2 = j;
             t3 = l;
             t4 = q;
       printf("T1 = %ld   T2 = %ld  T3 = %ld  T4 =%ld  \n",t1,t2,t3,t4);
} } } } }
printf("*****************complete**************\n");
/*******Display*******/
for (i = 0;i<size*3;i+=3)
        {
          if(value[i]<=t1)
          {
            b1[i]   = value[i];
            b1[i+1] = value[i];
            b1[i+2] = value[i];
          }
          if(value[i]>t1 && value[i]<=t2)
          {
            b2[i]   = value[i];
            b2[i+1] = value[i];
            b2[i+2] = value[i];
          }
```

```
            if(value[i]>t2 && value[i]<=t3)
            {
              b3[i]   = value[i];
              b3[i+1] = value[i];
              b3[i+2] = value[i];
            }
            if(value[i]>t3 && value[i]<=t4)
            {
              b4[i]   = value[i];
              b4[i+1] = value[i];
              b4[i+2] = value[i];
            }
            if(value[i]>t4 && value[i]<256)
            {
              b5[i]   = value[i];
              b5[i+1] = value[i];
              b5[i+2] = value[i];
            }
    }
w0 = XCreateSimpleWindow (d, RootWindow (d,0), 100, 50,w ,h,
                                    2, black, white);
w1 = XCreateSimpleWindow (d, RootWindow(d,0), 0, 0, w ,h,
                                    2, black, white);
w2 = XCreateSimpleWindow (d, RootWindow (d,0), 100, 50,w ,h,
                                    2, black, white);
w3 = XCreateSimpleWindow (d, RootWindow(d,0), 0, 0, w ,h,
                                    2, black, white);
w4 = XCreateSimpleWindow (d, RootWindow(d,0), 0, 0, w ,h,
                                    2, black, white);
w5 = XCreateSimpleWindow (d, RootWindow(d,0), 0, 0, w ,h,
                                    2, black, white);
XStoreName(d,w0,"original");
XStoreName(d,w1,"class1");
XStoreName(d,w2,"class2");
XStoreName(d,w3,"class3");
```

```
XStoreName(d,w4,"class4");
XStoreName(d,w5,"class5");
XSelectInput(d,w0,StructureNotifyMask|KeyPressMask);
 if(m) XShapeCombineMask (d,w0,ShapeBounding,0,0,m,ShapeSet);
 XMapWindow(d,w0);
ip1 = Imlib_create_image_from_data(id,b1,NULL,w,h);
Imlib_render(id,ip1,w,h);
Imlib_changed_image(id,ip1);
Imlib_apply_image(id,ip1,w1);
ip2 = Imlib_create_image_from_data(id,b2,NULL,w,h);
Imlib_apply_image(id,ip2,w2);
ip3 = Imlib_create_image_from_data(id,b3,NULL,w,h);
Imlib_apply_image(id,ip3,w3);
ip4 = Imlib_create_image_from_data(id,b4,NULL,w,h);
Imlib_apply_image(id,ip4,w4);
ip5 = Imlib_create_image_from_data(id,b5,NULL,w,h);
Imlib_apply_image(id,ip5,w5);
XMapWindow(d,w0);
XMapWindow(d,w1);
XMapWindow(d,w2);
XMapWindow(d,w3);
XMapWindow(d,w4);
XMapWindow(d,w5);
while (1)
        {
                XNextEvent(d,&e);
                if (e.type==ConfigureNotify)
                  {
                  w=e.xconfigure.width;
            h=e.xconfigure.height;
                   Imlib_render(id,im,w,h);
                   Imlib_free_pixmap(id,p);
                   p=Imlib_move_image(id,im);
                   m=Imlib_move_mask(id,im);
                   XSetWindowBackgroundPixmap(d,w0,p);
```

```
                    if (m) XShapeCombineMask(d,w0,ShapeBounding,
                                        0,0,m,ShapeSet);
                    XClearWindow(d,w0);
                    XSync(d,False);
                }
            if(e.xkey.keycode == 0x0018)
                exit(0);
            }
}
```

โปรแกรมใส่สีให้กับกลุ่มของข้อมูลก่อนแสดงผล

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
#include<Imlib.h>
#include<X11/keysym.h>
 Display *d;
Window w0,w1,w2,w3,w4,w5;
unsigned long black,white;
ImlibData *id;
ImlibImage *im,*ip,*ip1,*ip2,*ip3,*ip4,*ip5;
Pixmap p,m,p1 ;
int w,h,i,j,l,size,q;
unsigned char *value,*b1,*b2,*b3,*b4,*b5;
XEvent e;
usigned int code ;
 int x,n;
 long a,b,c,r,s,f,g,o;
 float u0,u1,u,u2,ut,u3,u4,histo[256];
float V0,V1,Vb,Vs;
long t1,t2,t3,t4;
float Pu0,Pu1,Pu2,Pu3,Pu4;
void mean(long k1,long k2,long k3,long k4)
{
   u=u0=u1=u2=u3=u4=0;
  Pu0=Pu1=Pu2=Pu3=Pu4=0;
  a=b=c=f=o=0;
  /*sigma*/
for( a = 0; a<=k1 ; a++)
    Pu0 += histo[a];
for( b = k1+1 ; b<=k2 ; b++)
   Pu1 += histo[b];
for( c = k2+1 ; c<=k3 ; c++)
   Pu2 += histo[c];
```

```
 for( f = k3+1 ;f<=k4 ; f++)
    Pu3 += histo[f];
 for( o = k4+1 ;o<256 ; o++)
    Pu4 += histo[o];
 /*  mean*/
    u=0;
    for( a = 0; a<=k1 ; a++)
        u += a*histo[a];
    u0 = u/Pu0;
   u=0;
 for( b = k1+1 ; b<=k2 ; b++)
    u  += b*histo[b];
u1 = u/Pu1;
  u=0;
for( c = k2+1 ; c<=k3 ; c++)
  u += c*histo[c];
u2 = u/Pu2;
  u=0;
 for( f = k3+1 ; f<=k4 ; f++)
    u += f*histo[f];
 u3 = u/Pu3;
 u=0;
 for( o = k4+1 ; o<256 ; o++)
    u += o*histo[o];
u4 = u/Pu4;
}
void max()
{
  float ut,uk;
   Vb = 0;
   Vb = (Pu0*(u0-ut)*(u0-ut)) + (Pu1*(u1-ut)*(u1-ut))
        + (Pu2*(u2-ut)*(u2-ut))+ (Pu3*(u3-ut)*(u3-ut))
       + (Pu4*(u4-ut)*(u4-ut));
}
int main(int argc , char **argv)
```

```
{
  if(argc<=1)
  {
    printf("Usage :\n %s image_file\n",argv[0]);
    exit(1);
  }
  d = XOpenDisplay (NULL);
 black = BlackPixel(d,0);
 white = WhitePixel(d,0);
id = Imlib_init(d);
im = Imlib_load_image(id,argv[1]);
w = im->rgb_width ;
h = im->rgb_height;
size = w*h;
value =(unsigned char *)malloc(size*3*sizeof (unsigned char));
b1 =(unsigned char *)malloc(size*3*sizeof (unsigned char));
value = im->rgb_data;
im = Imlib_create_image_from_data(id,value,NULL,w,h);
Imlib_render(id,im,w,h);
p = Imlib_move_image(id,im);
m = Imlib_move_mask(id,im);
XSetWindowBackgroundPixmap(d,w0,p);
 for (i=0;i<256;i++)
     histo[i]  =0;
for (i=0;i < size*3; i+= 3)
    histo[value[i]] += 1;
for (i=0;i<256;i++)
  histo[i] /= (float)size;
  ut =0;
for(j=1;j<256;j++)
   ut += j*histo[j];
Vs = t1 = t2 =r=t3=0;
for(i=1;i<250;i++)
{
 for(j=i+2;j<252;j++)
```

```
    {
for(l=j+2;l<254;l++)
{
 for(q=j+2;q<256;q++)
 {
  mean(i,j,l,q);
  max();
   if ( Vb > Vs)
  {
   Vs = Vb;
   t1 = i;
   t2 = j;
   t3 = l;
  t4 = q;
 printf("T1 = %ld  T2 = %ld  T3 = %ld  T4 =%ld  \n",t1,t2,t3,t4);
  } } }  }  }
 printf("*****************complete**************\n");
for(i=0;i<size*3;i+=3)
 {
   if (value[i]<t1)
 {
   b1[i]   = value[i];
   b1[i+1] = value[i];
   b1[i+2] = value[i];
 }
if(value[i]>t1 && value[i]<=t2)
  {
    b1[i]   = 100;
    b1[i+1] = 0;
    b1[i+2] = 200;
  }
  if(value[i]>t2 && value[i]<=t3)
  {
    b1[i]   = 255;
    b1[i+1] = 0;
```

```
      b1[i+2] = 0;
    }
 if(value[i]>t3 && value[i]<=t4)
    {
      b1[i]   = 0;
       b1[i+1] = 255;
      b1[i+2] = 0;
    }
    if(value[i]>t4 && value[i]<256)
    {
     b1[i]   = 0;
    b1[i+1] = 0;
    b1[i+2] = 255;
    }
w0 = XCreateSimpleWindow (d, RootWindow (d,0), 100, 50,w ,h,
                                        2, black, white);
 w1 = XCreateSimpleWindow (d, RootWindow(d,0), 0, 0, w ,h,
                                        2, black, white);
XStoreName(d,w0,"original");
XStoreName(d,w1,"clustered");
XSelectInput(d,w0,StructureNotifyMask|KeyPressMask);
 if(m) XShapeCombineMask (d,w0,ShapeBounding,0,0,m,ShapeSet);
XMapWindow(d,w0);
ip1 = Imlib_create_image_from_data(id,b1,NULL,w,h);
Imlib_render(id,ip1,w,h);
 Imlib_changed_image(id,ip1);
Imlib_apply_image(id,ip1,w1);
 XMapWindow(d,w0);
 XMapWindow(d,w1);
```

ภาคผนวก ข

Human Brain Segmentation

By

Fuzzy C-Mean Algorithm

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<gtk/gtk.h>
#include<X11/Xlib.h>
#include<X11/Xutil.h>
#include<Imlib.h>
#include<X11/extensions/shape.h>
#include<X11/keysym.h>

Display *d;
ImlibData *id;
ImlibImage *im1,*im2,*im3,*im5,*ip0,*ip1,*ip2,
        *ip3,*ip4,*ip,*io1,*io2,*kok;
Window win,win0,win1,win2,win3,win4,win5,win6;
Pixmap p1,p2,p3,po,mo,pt,mt,pa,ma;
int w,h,w1,h1;
XEvent ev;
unsigned int code;
GC gc,gc1,theGC;
XSetWindowAttributes att;
XGCValues xgcv;

GtkWidget *window,*window1,*window2;
GtkWidget *button;
GtkWidget *table,*table1,*table2;
GtkTooltips *tooltip;
GtkWidget *separator;
GtkWidget *label;
GtkWidget *text1,*text2,*text3;
GtkAdjustment *adj;
GtkWidget *filew;
GtkWidget *vbox;
gfloat new_val;
unsigned long black,white,histo[256],
gc_mask,histo1[256],histo2[256];
unsigned char
*b1,*nbuff,*rbuff,*pbuff,*b2,*b3,*bo,*bs,
    *bd,*dbuff ,*buff0,*buff1,*buff2,*buff3,*buff4
    ,*obuff,*sbuff,*sa,*hbuff0,*hbuff1,*hbuff2,
    *hbuff3,*hbuff4,*a1,*a2;
int clus,ox1,ox2,ox3,sx1,sx2,sx3,oy1,oy2,oy3,sy1,
    sy2,sy3,dx,dy,ox4,oy4; su,ou,AppDone,hi
    long  m,n,p,count,gray,size,xx1,yy1,size1,xx2,
    yy2,count1,count2,count3,
num,i,j,z,p,r,g,q,s,k,x,y;
float total[5],VH[5],VU[5],V1[5],V2[5],CV[5];
float
erU[5][65540],erH[5][256],erV[5],to0,to1,a,b,c,e,max
    ,miss0,miss1, UH[5][260],UU[5][65540],UV[5]
[65536]
    ,PH[5][256],PU[5][65540], v1,v2;
Atom at1,at2;

GtkWidget *pbar ;
int ptimer;
int pstat;
gint pvalue;
gchar *yu;
gchar *vb;
char **ar;

/*******EXIT PROGRAM********/
void delete_event(GtkWidget *widget , gpointer data)
{
  gtk_main_quit();
```

```
}
/*******OPEN  FILES*********/
void destroy (GtkWidget *widget, gpointer data)
{
  gtk_widget_destroy(filew);
}
file_ok_sel (GtkWidget *ww, GtkFileSelection *fs)
{
   black = BlackPixel(d,0);
   white = WhitePixel(d,0);
   vb = gtk_file_selection_get_filename
            (GTK_FILE_SELECTION (fs));
   im5 = Imlib_load_image(id,vb);
   w1 = im5->rgb_width;
   h1 = im5->rgb_height;
   win3 = XCreateSimpleWindow(d,RootWindow(d,0)
            ,0,0,w1,h1,  2,black,white);
   XStoreName(d,win3,vb);
   Imlib_apply_image(id,im5,win3);
   XMapWindow(d,win3);
   XFlush(d);
   gtk_widget_destroy(filew);
}
open_save(GtkWidget *widget , gpointer data)
{
filew = gtk_file_selection_new ("File selection");
gtk_signal_connect (GTK_OBJECT (filew), "destroy",
     (GtkSignalFunc) destroy, &filew);
gtk_signal_connect (GTK_OBJECT
(GTK_FILE_SELECTION
     (filew)->ok_button),"clicked",
     (GtkSignalFunc) file_ok_sel, filew );
gtk_signal_connect_object (GTK_OBJECT
(GTK_FILE_SELECTION
    (filew)->ncel_button), "clicked",
    (GtkSignalFunc) gtk_widget_destroy,
GTK_OBJECT (filew));
gtk_file_selection_set_filename
(GTK_FILE_SELECTION(filew," ");
 gtk_widget_show(filew);
}
/***********SAVE IMAGE**************/
void destroy1(GtkWidget *widget, gpointer data)
{
  gtk_widget_destroy(window1);
}
ok1(GtkWidget *widget , gpointer data)
{
  while(1)
  {
  XNextEvent(d,&ev);
  switch(ev.type)
  {
  case ButtonPress :
     if(ev.xbutton.button ==1)
      {
       if (ev.xany.window == win)
        {
          yu = gtk_entry_get_text(GTK_ENTRY(text1));
          Imlib_save_image(id,ip1,yu,NULL);
          goto pam;
        }
      if (ev.xany.window == win1)
       {
         yu = gtk_entry_get_text(GTK_ENTRY(text1));
         Imlib_save_image(id,ip0,yu,NULL);
          goto pam;
        }
    if (ev.xany.window == win2)
      {
      yu = gtk_entry_get_text(GTK_ENTRY(text1));
      Imlib_save_image(id,ip,yu,NULL);
      goto pam;
       } } }
      }
  pam:       gtk_widget_destroy(window1);
}
   save_image(GtkWidget *widget , gpointer data)
```

```
{
  window1 =
gtk_window_new(GTK_WINDOW_DIALOG);
  gtk_window_set_title(GTK_WINDOW
(window1),"SAVE IMAGE");

gtk_signal_connect(GTK_OBJECT(window1),"delete_
event",
            GTK_SIGNAL_FUNC(delete_event),NULL);
  gtk_container_border_width(GTK_CONTAINER
(window1),4);
  table1 = gtk_table_new(3,2,TRUE);
  gtk_container_add(GTK_CONTAINER
(window1),table1);
  label = gtk_label_new("..IMAGE NAME..");
  gtk_table_attach_defaults(GTK_TABLE
(table1),label,0,2,0,1);
  gtk_widget_show(label);
  text1 = gtk_entry_new_with_max_length(100);
  gtk_entry_set_text(GTK_ENTRY(text1),"ENTER FILE
NAME");
  gtk_table_attach_defaults(GTK_TABLE
(table1),text1,0,2,1,2);
  gtk_widget_show(text1);
  button = gtk_button_new_with_label("SAVE");
  gtk_table_attach_defaults(GTK_TABLE
(table1),button,0,1,2,3);
  gtk_signal_connect(GTK_OBJECT(button),"clicked",
            GTK_SIGNAL_FUNC(ok1),NULL);
  gtk_widget_show(button);
  button = gtk_button_new_with_label("CANCEL");
  gtk_table_attach_defaults(GTK_TABLE
(table1),button,1,2,2,3);
  gtk_signal_connect(GTK_OBJECT(button),"clicked",
            GTK_SIGNAL_FUNC(destroy1),NULL);
  gtk_widget_show(button);
  gtk_widget_show(table1);
  gtk_widget_show(window1);
}
/**********OPEN ORGINAL***************/
open_T1(GtkWidget *widget , gpointer data)
{
  pstat =TRUE;
  ou++;
  bo = b1;
  O_buff();
  XStoreName(d,win,"T1");
  Imlib_apply_image(id,im1,win);
  ip1 = Imlib_create_image_from_data
(id,bo,NULL,w,h);
  XMapWindow(d,win);
  XSync(d,False);
}
open_T2(GtkWidget *widget , gpointer data)
{
  ou++;
  bo = b2;
  O_buff();
  XStoreName(d,win,"T2");
  Imlib_apply_image(id,im2,win);
  ip1 = Imlib_create_image_from_data
(id,bo,NULL,w,h);
  XMapWindow(d,win);
  XSync(d,False);
}
open_PD(GtkWidget *widget , gpointer data)
{
  ou++;
  bo = b3;
  O_buff();
  XStoreName(d,win,"PD");
  Imlib_apply_image(id,im3,win);
  ip1 = Imlib_create_image_from_data
(id,bo,NULL,w,h);
  XMapWindow(d,win);
  XSync(d,False);
}
/**********OPEN TISSUE*****************/
```

```
white_matter(GtkWidget *widget , gpointer data)
{
   su++;
   XStoreName(d,win1,"WHITE MATTER");
   pbuff = buff1;
   change_rgb();
   ip0 = Imlib_create_image_from_data
(id,nbuff,NULL,w,h);
   Imlib_apply_image(id,ip0,win1);
   ip2 = Imlib_create_image_from_data
(id,nbuff,NULL,w,h);
   XMapWindow(d,win1);
   XSync(d,False);
   n = 0;
   for (i=0;i<256;i++)
   histo[i] = 0;
   for (i=0;i<size;i++)
   histo[pbuff[i]] +=1;
}
gray_matter (GtkWidget *widget , gpointer data)
{
su++;
XStoreName(d,win1,"GRAY MATTER");
pbuff = buff3;
change_rgb();
ip0 =
Imlib_create_image_from_data(id,nbuff,NULL,w,h);
Imlib_apply_image(id,ip0,win1);
ip2 =
Imlib_create_image_from_data(id,nbuff,NULL,w,h);
XMapWindow(d,win1);
XSync(d,False);
 n = 0;
for (i=0;i<256;i++)
histo[i] = 0;
 for (i=0;i<size;i++)
 histo[pbuff[i]] +=1;
{
Skull (GtkWidget *widget , gpointer data)
{
su++;
XStoreName(d,win1,"SKULL");
pbuff = buff2;
change_rgb();
ip0 =
Imlib_create_image_from_data(id,nbuff,NULL,w,h);
Imlib_apply_image(id,ip0,win1);
ip2 =
Imlib_create_image_from_data(id,nbuff,NULL,w,h);
XMapWindow(d,win1);
XSync(d,False);
n = 0;
for (i=0;i<256;i++)
histo[i] = 0;
for (i=0;i<size;i++)
 histo[pbuff[i]] +=1;
}
CSF(GtkWidget *widget , gpointer data)
{
 su++;
 XStoreName(d,win1,"CSF");
 pbuff = buff0;
 change_rgb();
 ip0 = Imlib_create_image_from_data
(id,nbuff,NULL,w,h);
 Imlib_apply_image(id,ip0,win1);
 ip2 = Imlib_create_image_from_data
(id,nbuff,NULL,w,h);
 XMapWindow(d,win1);
 XSync(d,False);
 n = 0;
 for (i=0;i<256;i++)
   histo[i] = 0;
for (i=0;i<size;i++)
   histo[pbuff[i]] +=1;
}
close_tissue(GtkWidget *widget , gpointer data)
{
```

```
 XUnmapWindow(d,win1);
 XSync(d,FALSE);
}
/****************SELECT AREA***************/
area (GtkWidget *widget , gpointer data)
{
    if (count != 0)
    {
     XUnmapWindow(d,win2);
     XSync(d,FALSE);
     }
    count1 = count1+1;
     region();
}
region()
{
 while(1)
{
 XNextEvent(d,&ev);
 switch(ev.type)
{
 case ButtonPress :
  if (ev.xbutton.button ==3)
  {
    if(ev.xany.window == win)
   {
     i=0;
     for(m=0;m<size;m++)
       for(n=0;n<=2;n++)
      {
        a1[i] = obuff[m];
       ++i;
      }
io1 =
Imlib_create_image_from_data(id,a1,NULL,w,h);
goto pop;
  }
if(ev.xany.window == win1)
 {
  i=0;
 for(m=0;m<size;m++)
  for(n=0;n<=2;n++)
  {
   a2[i] = pbuff[m];
   ++i;
  }
io2 =
Imlib_create_image_from_data(id,a2,NULL,w,h);
 goto pop;
}
}
}
if (ev.xbutton.button ==1)
{
ox1 = ev.xbutton.x;
oy1 = ev.xbutton.y;
if(ev.xany.window == win)
{
  if ( su == 0 && ou !=0)
   {
 Imlib_apply_image(id,ip1,win);
 XMapWindow(d,win);
 XFlush(d);
  }
 if (ou ==0 && su !=0)
 {
 Imlib_apply_image(id,ip2,win1);
 XMapWindow(d,win1);
 XFlush(d);
 }
 if(ou != 0 && su !=0)
 {
 Imlib_apply_image(id,ip1,win);
 XMapWindow(d,win);
Imlib_apply_image(id,ip2,win1);
XMapWindow(d,win1);
XFlush(d);
 }
```

```
}
if(ev.xany.window == win1)
{
if ( su == 0 && ou !=0)
 {
  Imlib_apply_image(id,ip1,win);
  XMapWindow(d,win);
  XFlush(d);
 }
 if (ou ==0 && su !=0)
 {
Imlib_apply_image(id,ip2,win1);
XMapWindow(d,win1);
 XFlush(d);
}
if(ou != 0 && su !=0)
{
Imlib_apply_image(id,ip1,win);
XMapWindow(d,win);
Imlib_apply_image(id,ip2,win1);
 Imlib_apply_image(id,ip2,win1);
 XMapWindow(d,win1);
 XFlush(d);
  }
 }
break;
}
case ButtonRelease:
     if (ev.xbutton.button ==1)
      {
        ox2 = ev.xbutton.x;
        oy2 = ev.xbutton.y;
  if (ox1<ox3 && oy1>oy3)/*down-left to up-right*/
        {
         oy4 = oy1;
         oy1 = oy2;
         oy2 = oy4;
         }
 if (ox1>ox3 && oy1<oy3) /*up-right to down-left*/

        {
         ox4 = ox1;
        ox1 = ox2;
        ox2 = ox4;
         }
if (ox1>ox3 && oy1>oy3)/*down-right to up-left*/

       {
        ox4 = ox1;
        ox1 = ox2;
       ox2 = ox4;
       oy4 = oy1;
       o y1 = oy2;
       oy2 = oy4;
       }
             break;
          }
  case MotionNotify:
     if(ev.xany.window == win)
     {
     if (ou !=0 && su !=0)
    {
   Imlib_apply_image(id,ip2,win1);
   XMapWindow(d,win1);
    }
  dbuff = obuff;
  Imlib_apply_image(id,ip1,win);
  XMapWindow(d,win);
  XFlush(d);
  ox3 = ev.xbutton.x;/* ox1,oy1,ox2,oy2 fix it*/
  oy3 = ev.xbutton.y;/*ox3,oy3 depend on drag*/
  dx =  ox3-ox1 ;
  dy =  oy3-oy1 ;
 if(dx<0)
   dx = dx*(-1);
 if(dy<0)
    dy = dy*(-1);
```

```
if (ox1<ox3 && oy1>oy3)/*down-left to up-right*/
   {
    XDrawRectangle(d,win,gc,ox1,oy3,dx,dy);
   }
if (ox1>ox3 && oy1<oy3) /*up-right to down-left*/
 {
  XDrawRectangle(d,win,gc,ox3,oy1,dx,dy);
 }
if (ox1<ox3 && oy1<oy3)/*up-left to down-right*/
 {
  XDrawRectangle(d,win,gc,ox1,oy1,dx,dy);
  }
if (ox1>ox3 && oy1>oy3)/*down-right to up-left*/
 {
   XDrawRectangle(d,win,gc,ox3,oy3,dx,dy);
 }
 }
  if(ev.xany.window == win1)
  {
   if(ou!=0 && su!=0)
  {
 Imlib_apply_image(id,ip1,win);
 XMapWindow(d,win);
 }
 dbuff = pbuff;
 Imlib_apply_image(id,ip2,win1);
 XMapWindow(d,win1);
 XFlush(d);
 ox3 = ev.xbutton.x;
 oy3 = ev.xbutton.y;
 dx =  ox3-ox1 ;
 dy =  oy3-oy1 ;
if(dx<0)
   dx = dx*(-1);
 if(dy<0)
   dy = dy*(-1);
if (ox1<ox3 && oy1>oy3)/*down-left to up-right*/
  {
  XDrawRectangle(d,win1,gc,ox1,oy3,dx,dy);
  }
if (ox1>ox3 && oy1<oy3) /*up-right to down-left*/
 {
  XDrawRectangle(d,win1,gc,ox3,oy1,dx,dy);
 }
  if (ox1<ox3 && oy1<oy3)/*up-left to down-right*/
   {
    XDrawRectangle(d,win1,gc,ox1,oy1,dx,dy);
   }
 if (ox1>ox3 && oy1>oy3)/*down-right to up-left*/
   {
   XDrawRectangle(d,win1,gc,ox3,oy3,dx,dy);
    }
 }
 break;
}
pop:
  xx1 =  (ox2-ox1)+1 ;
  yy1 =  (oy2-oy1)+1 ;
}
opens (GtkWidget *widget , gpointer data)
{
count2 = count2+1;
if (count1 !=0)
{
  if ( su == 0 && ou !=0)
 {
   Imlib_apply_image(id,io1,win);
   XMapWindow(d,win);
   XFlush(d);
 }
 if (ou ==0 && su !=0)
 {
   Imlib_apply_image(id,io2,win1);
   XMapWindow(d,win1);
   XFlush(d);
 }
   if(ou != 0 && su !=0)
 {
```

```
  Imlib_apply_image(id,io1,win);
  XMapWindow(d,win);
  Imlib_apply_image(id,io2,win1);
  XMapWindow(d,win1);
  XFlush(d);
 }
count = count +1;
black = BlackPixel(d,0);
white = WhitePixel(d,0);
win2  = XCreateSimpleWindow(d,RootWindow(d,0),0,
    0,xx1,yy1, 2,black,white);
XSelectInput(d,win2,StructureNotifyMask
|ButtonPressMask
  |ButtonReleaseMask | ButtonMotionMask
                              |KeyPressMask);
bd   = (unsigned char*)malloc(xx1*yy1*3*
   sizeof(unsigned char));
k=0;
i =0;j=0;
 for(j=oy1;j<=oy2;j++)
 {
  for(i=ox1;i<=ox2;i++)
   {
    bd[k]     = dbuff[i+j*w];
    bd[k+1]   = dbuff[i+j*w];
    bd[k+2]   = dbuff[i+j*w];
    k = k+3 ;
   }
}
XStoreName(d,win2,"SELECTED AREA");
ip =
Imlib_create_image_from_data(id,bd,NULL,xx1,yy1);
imlib_apply_image(id,ip,win2);
XMapWindow(d,win2);
XFlush(d);
free(bd);
xx2 = xx1;
yy2 = yy1;
}
gain(GtkWidget *widget , gpointer data)
{
 if(count != 0)
 {
  XUnmapWindow(d,win2);
  XSync(d,FALSE);
  black = BlackPixel(d,0);
  white = WhitePixel(d,0);
  xx2 +=5;
  yy2 +=5;
  XResizeWindow(d,win2,xx2,yy2);
  Imlib_render(id,ip,xx2,yy2);
  p1 = Imlib_move_image(id,ip);
  p2 = Imlib_move_mask(id,ip);
  XSetWindowBackgroundPixmap(d,win2,p1);
  if(p2)XShapeCombineMask(d,win2,ShapeBounding
          ,0,0,p2,ShapeSet);
  XMapWindow(d,win2);
  XFlush(d);
 }
}
lose(GtkWidget *widget , gpointer data)
{
  if(count != 0)
 {
  XUnmapWindow(d,win2);
  XSync(d,FALSE);
  black = BlackPixel(d,0);
  white = WhitePixel(d,0);
  xx2 -=5;
  yy2 -=5;
  XResizeWindow(d,win2,xx2,yy2);
  Imlib_render(id,ip,xx2,yy2);
  p1 = Imlib_move_image(id,ip);
  p2 = Imlib_move_mask(id,ip);
  XSetWindowBackgroundPixmap(d,win2,p1);
  if(p2)XShapeCombineMask(d,win2,ShapeBounding
          ,0,0,p2,ShapeSet);
  XMapWindow(d,win2);
```

```
  XFlush(d);
 }
}
close_viewer(GtkWidget *widget , gpointer data)
{
 XUnmapWindow(d,win2);
 XSync(d,FALSE);
}
/*********Fuzzy C-Mean BY Histogram**************/
segment_H(GtkWidget *widget , gpointer data)
{
  for (i=0;i<size;i++)
 histo[rbuff[i]] +=1;
 goto topH;
  topH :
        total_UikH();
        find_VH();
        find_newUH();
        find_erH();
 for ( m=0;m<clus; m++)
 {
  for (n=0 ; n<256; n++)
  {
    if (erH[m][n]>0.000001)
    {
     for (m=0 ; m<clus ; m++)
       for (n=0 ; n<256 ; n++)
       {
       UH[m][n] = PH[m][n];}
       goto topH ;
      }
  }
 }
 q=i=j=k=s=0;
    for(n=0;n<256;n++)
    {
     max = 0;
    for(m=0;m<5;m++)
    {
 make_UH();
 for (i=0;i<size;i++)
 {
  buff0[i] = 0;
  buff1[i] = 0;
  buff2[i] = 0;
  buff3[i] = 0;
 }
 bs = b1;
 R_buff();

    if (max < PH[m][n])
    { max = PH[m][n];
      gray = m;
    }
  }
 if( gray == 1)
  {
  hbuff1[i] = n ;
   i++;
  }
 if( gray == 2)
  {
   hbuff2[j] = n ;
   j++;
  }
if( gray == 3)
  {
  hbuff3[k] = n;
  k++;
   }
}
for (x=0;x<size;x++)
 {
  for (p=0;p<i;p++)
  {
   if( rbuff[x]==hbuff3[p])
     buff1[x] = rbuff[x];
  }
```

```
for (r=0;r<j;r++)
 {
  if( rbuff[x]==hbuff2[r])
     buff2[x] = rbuff[x];          }
 for (g=0;g<k;g++)
 {
 if( rbuff[x]==hbuff1[g])
    buff3[x] = rbuff[x];
 }
}
 bs=b2;
 count = 0;
 R_buff();
 for (i=0;i<size;i++)
 histo[rbuff[i]] +=1;
 goto lowH;
 lowH :
       total_UikH();
       find_VH();
       find_newUH();
       find_erH();
 for ( m=0;m<clus; m++)
 {
  for (n=0 ; n<256; n++)
  {
  if (erH[m][n]>0.000001)
  {
    for (m=0 ; m<clus ; m++)
     for (n=0 ; n<256 ; n++)
     { UH[m][n] = PH[m][n];}
      goto lowH ;
     }
  }
 }
 q=0;
 for(n=0;n<256;n++)
 {
  max = 0;
 for(m=0;m<5;m++)
  {
  if (max < PH[m][n])
   { max = PH[m][n];
     gray = m;
  }
 }
 if( gray == 2)
 {
  hbuff0[q] = n ;
  q++;
 }
}
for (x=0;x<size;x++)
 {
  for (y=0;y<q;y++)
  {
  if( rbuff[x]==hbuff0[y])
    buff0[x] = rbuff[x];
  }
 }
}
make_UH()
{
  for(m=0;m<5;m++)
  {
  for(n=0;n<260;n+=10)
  {
  if( m==0)
  {
  UH[m][n]    =1;
  UH[m][n+1] =1;
  UH[m][n+2] =0;
  UH[m][n+3] =0;
  UH[m][n+4] =0;
  UH[m][n+5] =0;
  UH[m][n+6] =0;
  UH[m][n+7] =0;
  UH[m][n+8] =0;
  UH[m][n+9] =0;
```

```
    }
  if (m==1)
   {
  UH[m][n]   = 0;
  UH[m][n+1] = 0;
  UH[m][n+2] = 1;
  UH[m][n+3] = 1;
  UH[m][n+4] = 0;
  UH[m][n+5] = 0;
  UH[m][n+6] = 0;
  UH[m][n+7] = 0;
  UH[m][n+8] = 0;
  UH[m][n+9] = 0;
    }
 if (m==2)
  {
  UH[m][n]   = 0;
  UH[m][n+1] = 0;
  UH[m][n+2] = 0;
  UH[m][n+3] = 0;
  UH[m][n+4] = 1;
  UH[m][n+5] = 1;
  UH[m][n+6] = 0;
  UH[m][n+7] = 0;
  UH[m][n+8] = 0;
  UH[m][n+9] = 0;
    }
if (m==3)
  {
 UH[m][n]   = 0;
 UH[m][n+1] = 0;
 UH[m][n+2] = 0;
 UH[m][n+3] = 0;
 UH[m][n+4] = 0;
 UH[m][n+5] = 0;
 UH[m][n+6] = 1;
 UH[m][n+7] = 1;
 UH[m][n+8] = 0;
 UH[m][n+9] = 0;
  }
if (m==4)
  {
 UH[m][n]   = 0;
 UH[m][n+1] = 0;
 UH[m][n+2] = 0;
 UH[m][n+3] = 0;
 UH[m][n+4] = 0;
 UH[m][n+5] = 0;
 UH[m][n+6] = 0;
 UH[m][n+7] = 0;
 UH[m][n+8] = 1;
 UH[m][n+9] = 1;
 }
 }
   }
}
R_buff()
{
 j=0;
 for (i=0;i<size*3;i+=3)
 {
 rbuff[j] = bs[i];
++j;
 }
}
O_buff()
{
  j=0;
  for (i=0;i<size*3;i+=3)
 {
  rbuff[j] = bo[i];
 ++j;
 }
}
change_rgb()
{
  i=0;
  for(m=0;m<size;m++)
```

```
    for(n=0;n<=2;n++)
   {
   nbuff[i] = pbuff[m];
   ++i;
    }
}
total_UikH()
{
  for (m=0;m<clus;m++)
 {
   to1 = to0 = 0;
   for(num = 0 ; num<256 ;num++)
   {
      to0 =(histo[num]*pow(UH[m][num],2)) + to1;
      to1 = to0;
    }
     totai[m] = to1;
  }
}
find_VH()
{
   for (m=0;m<clus;m++)
   {
     v1 =v2 =0;
    for (n=0;n<256;n++)
    {
     v1 = (histo[n]*n*pow(UH[m][n],2)) + v2 ;
     v2 = v1;
    }
    VH[m] = v2/total[m] ;
    }
}
find_newUH()
{
   for (m=0 ; m<clus ; m++)
    {
     for (n=0 ; n<256; n++)
     {
     a=0;
a = n - VH[m];
 if(a<(float)0)
 { a = a*(-1);}
 if(a!=0 && a>0)
 {
 c=e=0;
 for (i=0 ; i<clus ; i++)
 {
 b=0;
 b = (n - VH[i]) ;
if(b<0)
 b = b*(-1);
 c = pow((a/b),2) + e;
 e = c;
 }
 PH[m][n] = 1/e;
}
 if(a==0)
  PH[m][n] = 1;
 }
}
}
find_erH()
{
  for (m=0 ; m<clus ; m++)
   for (n=0 ; n<256 ; n++)
   {
   erH[m][n] = ( PH[m][n] - UH[m][n] ) ;
   if (erH[m][n]<0)
    erH[m][n] = erH[m][n]*(-1);
   }
}
/********Fuzzy C-Mean BY U**********/
segment_U(GtkWidget *widget , gpointer data)
{
  make_UU();
  count3 = count3 +1;
  bs = b1;
  R_buff();
```

```
count =0;
goto topU;
topU :  total_UikU();
        find_VU();
        find_newUU();
        find_erU();
for ( m=0;m<clus; m++)
{
  for (n=0 ; n<size; n++)
  {
  if (erU[m][n]>0.005)
  {
   for (m=0 ; m<clus ; m++)
    for (n=0 ; n<size ; n++)
   { UU[m][n] = PU[m][n];}
    goto topU ;
    }
  }
}
   gray = 0;
    for(n=0;n<size;n++)
   {
    max = 0;
    for(m=0;m<5;m++)
   {
    if (max < PU[m][n])
     {  max = PU[m][n];
        gray = m;
     }
   }
 if( gray == 1)
{
 buff1[n] = rbuff[n] ;
}
 if( gray == 2)
  {
  buff3[n] = rbuff[n] ;
  }
 if( gray == 4)
    {
  buff2[n] = rbuff[n]
    }
}
 bs=b2;
count = 0;
R_buff();
goto lowU;
lowU :
        total_UikU();
        find_VU();
        find_newUU();
        find_erU();
 for ( m=0;m<clus; m++)
 {
  for (n=0 ; n<size; n++)
  {
   if (erU[m][n]>0.005)
   {
    for (m=0 ; m<clus ; m++)
    for (n=0 ; n<size ; n++)
      { UU[m][n] = PU[m][n];}
       goto lowU ;
      }
  }
 }
  gray = 0;
  for(n=0;n<size;n++)
  {
   max = 0;
   for(m=0;m<5;m++)
   {
    if (max < PU[m][n])
    {  max = PU[m][n];
     gray = m;
    }
  }
  if( gray == 3)
 {
```

```
  buff0[n] = rbuff[n] ;
}
}
}
make_UU()
{
 for(m=0;m<clus;m++)
  {
   for(n=0;n<65540;n+=10)
   {
   if( m==0)
   {
    UU[m][n]  =1;
    UU[m][n+1] =1;
    UU[m][n+2] =0;
    UU[m][n+3] =0;
    UU[m][n+4] =0;
    UU[m][n+5] =0;
    UU[m][n+6] =0;
    UU[m][n+7] =0;
    UU[m][n+8] =0;
    UU[m][n+9] =0;
     }
  if (m==1)
    {
    UU[m][n]  = 0;
    UU[m][n+1] = 0;
    UU[m][n+2] = 1;
    UU[m][n+3] = 1;
    UU[m][n+4] = 0;
    UU[m][n+5] = 0;
    UU[m][n+6] = 0;
    UU[m][n+7] = 0;
    UU[m][n+8] = 0;
    UU[m][n+9] = 0;
    }
  if (m==2)
    {
     UU[m][n]  = 0;
     UU[m][n+1] = 0;
     UU[m][n+2] = 0;
     UU[m][n+3] = 0;
     UU[m][n+4] = 1;
     UU[m][n+5] = 1;
     UU[m][n+6] = 0;
     UU[m][n+7] = 0;
     UU[m][n+8] = 0;
     UU[m][n+9] = 0;
      }
    if (m==3)
      {
     UU[m][n]  = 0;
     UU[m][n+1] = 0;
     UU[m][n+2] = 0;
     UU[m][n+3] = 0;
     UU[m][n+4] = 0;
     UU[m][n+5] = 0;
     UU[m][n+6] = 1;
     UU[m][n+7] = 1;
     UU[m][n+8] = 0;
     UU[m][n+9] = 0;
     }
    if (m==4)
     {
      UU[m][n]  = 0;
      UU[m][n+1] = 0;
      UU[m][n+2] = 0;
      UU[m][n+3] = 0;
      UU[m][n+4] = 0;
      UU[m][n+5] = 0;
      UU[m][n+6] = 0;
      UU[m][n+7] = 0;
      UU[m][n+8] = 1;
      UU[m][n+9] = 1;
    }
    }
    }
}
```

```
total_UikU()
{
  for (m=0;m<clus;m++)
 {
   to1 = to0 = 0;
   for(num = 0 ; num<size ;num++)
   {
    to0 =pow(UU[m][num],2) + to1;
    to1 = to0;
   }
    total[m] = to1;
 }
}
find_VU()
{
  for (m=0;m<clus;m++)
   {
     v1 =v2 =0;
   for (n=0;n<size;n++)
   {
    v1 = (rbuff[n]*pow(UU[m][n] 2)) + v2 ;
    v2 = v1;
   }
     VU[m] = v2/total[m] ;
   }
}
find_newUU()
{
  for (m=0 ; m<clus ; m++)
   {
    for (n=0 ; n<size; n++)
    {
      a=0;
     a = rbuff[n] - VU[m];
     if(a<(float)0)
     { a = a*(-1);}
     if(a!=0 && a>0)
     {
       c=e=0;
    for (i=0 ; i<clus ; i++)
    {
     b=0;
     b = (rbuff[n] - VU[i]) ;
     if(b<0)
      b = b*(-1);
      c = pow((a/b),2) + e;
      e = c;
     }
     PU[m][n] = 1/e;
  }
 if(a==0)
   PU[m][n] = 1;
}
}
}
find_erU()
{
   for (m=0 ; m<clus ; m++)
    for (n=0 ; n<size ; n++)
    {
    erU[m][n] = ( PU[m][n] - UU[m][n] ) ;
    if (erU[m][n]<0)
    erU[m][n] = erU[m][n]*(-1);
    }
}
/**********Fuzzy C-Mean BY V*************/
segment_V(GtkWidget *widget , gpointer data)
{
  V1[0] = 0.5;
  V1[1] = 62.2;
  V1[2] = 118.97;
  V1[3] = 170.5;
  V1[4] = 248.1;
  count = 0;
  bs = b1;
  R_buff();
  topV :  find_UV();
          total_UikV();
```

```
        find_newV();
        find_erV();
  miss0 = miss1=0;
  for ( m=0;m<clus; m++)
   {
    miss1 = erV[m] + miss0;
    miss0 = miss1;
   }
  if (miss0>0.01)
 {
   for (m=0 ; m<clus ; m++)
   { V1[m] = V2[m];}
    goto topV ;
   }
   for(n=0;n<size;n++)
   {
    max = 0;
   for(m=0;m<clus;m++)
   {
    if( max < UV[m][n] )
     {
    max = UV[m][n];
   gray = m;
     }
 }
if( gray == 3)
   buff1[n] = rbuff[n];
if( gray == 2)
   buff3[n] = rbuff[n];
if( gray == 4)
   buff2[n] = rbuff[n];}
  V1[0] = 0.40;
  V1[1] = 40.89;
  V1[2] = 95.73;
  V1[3] = 150.9;
  V1[4] = 235.15;
  bs = b2;
  R_buff();
  count = 0 ;
lowV :  find_UV();
        total_UikV();
        find_newV();
        find_erV();
  miss0 = miss1=0;
  for ( m=0;m<clus; m++)
   {
   miss1 = erV[m] + miss0;
   miss0 = miss1;
   }
  if (miss0>0.01)
 {
  for (m=0 ; m<clus ; m++)
    { V1[m] = V2[m];}
    goto lowV ;
    }
   for(n=0;n<size;n++)
   {
    max = 0;
    for(m=0;m<clus;m++)
   {
   if( max < UV[m][n] )
    {
    max = UV[m][n];
    gray = m;
    }
 }
 if( gray == 4)
   buff0[n] = rbuff[n];
}
}
find_UV()
{
  for (m=0 ; m<clus ; m++)
   {
    for (n=0 ; n<size; n++)
    {
      a=0;
     a = rbuff[n] - V1[m];
```

```
    if(a<0)
    {  a = a*(-1);}
     if(a!=0 && a>0)
    {
    c=e=0;
   for (i=0 ; i<cius ; i++)
   {
     b=0;
     b = (rbuff[n] - V1[i]) ;
      if(b<0)
        b = b*(-1);
    if(b==0)
        b = 0.1;
    c = pow((a/b),2) + e;
    e = c;
}
  UV[m][n] = 1/e;
}
   if(a==0)
      UV[m][n] = 1;
 }
 }
}
total_UikV()
{
  for (m=0;m<clus;m++)
  {
  to0 = to1=0;
  for (num = 0;num<size;num++)
  {
   to0 = pow(UV[m][num],2) + to1;
    for (i=0;i < 256; i++)
    histo2[i] =0;
    while(1)
    {
      XNextEvent (d,&ev);
       switch(ev.type)
      {
   case ButtonPress:
  to1 = to0;
 }
total[m] = to1;
  }
}
find_newV()
{
   for (m=0;m<clus;m++)
   {
    v1 =v2 = 0;
    for (n=0;n<size;n++)
    {
     v1 = (rbuff[n]*pow(UV[m][n],2)) + v2 ;
     v2 = v1;
    }
      V2[m] = v2/total[m] ;
    }
 }
find_erV()
{
   for (m=0 ; m<clus ; m++)
    {
    erV[m] = ( V2[m] - V1[m] ) ;
    if (erV[m]<0)
    erV[m] = erV[m]*(-1);
   }
}
/*****************Gray Level*********************/
gray_level(GtkWidget *widget , gpointer data)
{

   if (ev.xbutton.button == 3 )
      goto more;
  case MotionNotify:
    if (ev.xany.window ==win4)
    {
      if (ev.xkey.x >= 20 && ev.xkey.x <= 275
      && ev.xkey.y >= 10 && ev.xkey.y <= 133 )
      {
```

```
    printf("\nGRAY LVEL = %d  ,  INTENSITY = %d\n"
            ,ev.xkey.x-20 , histo1[ev.xkey.x-20] );
     }
    }
   if (ev.xany.window ==win)
    {
     printf("\nPOINTER = (%d,%d)  ,  GRAY LVEL =
%d \n"
            ,ev.xkey.x,ev.xkey.y, obuff[ev.xkey.x+
(ev.xkey.y*w)] );
    }
      if (ev.xany.window == win1)
    {
 printf("\nPOINTER = (%d,%d)  ,  GRAY LVEL = %d
\n"
       ,ev.xkey.x,ev.xkey.y, pbuff[ev.xkey.x+
(ev.xkey.y*w)] );
          }
     }
    }more:
}
/************Histogram*******************/
close_histogram(GtkWidget *widget , gpointer data)
{
  XUnmapWindow(d,win4);
  XSync(d,FALSE);
}
histogram(GtkWidget *widget , gpointer data)
{
   for (i=0;i < 256; i++)
    histo1[i] =0;
    while(1)
   {
   XNextEvent(d,&ev);
   switch(ev.type)
   {
    case ButtonPress :
     if(ev.xbutton.button ==1)
     {
     if (ev.xany.window == win)
     {
      for (i=0;i < size; i++)
        histo1[obuff[i]] += 1;
      goto his;
      }
      if (ev.xany.window == win1)
      {
       for (i=0;i < size; i++)
         histo1[pbuff[i]] += 1;
       goto his;
       }
    }
   }
  }
his:
 if(hi != 0)
  {
   XUnmapWindow(d,win4);
   XSync(d,FALSE);
  }
  hi = hi +1;
  x = 20;
  y = 133;
  black = BlackPixel(d,0);
  white = WhitePixel(d,0);
  win4 = XCreateSimpleWindow (d,
DefaultRootWindow
      (d),200 , 100, 294 ,145,  2, black, white);
XSelectInput(d,win4,StructureNotifyMask|KeyPressMa
    sk|ExposureMask PointerMotionMask
    |Button1MotionMask|ButtonPressMask);
XStoreName(d,win4,"HISTOGRAM");
  att.backing_store = Always;
XChangeWindowAttributes(d,win4,CWBackingStore,&
    att);
XMapWindow(d,win4);
xgcv.line_width = 2;
```

```
   gc_mask = GCForeground
|GCBackground|GCLineWidth;
   theGC = XCreateGC (d,win4,gc_mask,&xgcv);
   XSetForeground(d,theGC,MyColor(d,"black"));
   XDrawLine(d,win4,theGC,20,10,20,135);
   XDrawLine(d,win4,theGC,280,135,20,135);
   XSetForeground(d,theGC,MyColor(d,"red"));
  for(i=0 ; i < 256 ; i++)
    {
    for (j=0 ; j < histo1[i]/10  ; j++)
      {
       XDrawPoint(d,win4,theGC,x,y);
       y = y - 1;
      }
      x += 1;
      y =133;
    }
    XFlush(d);
}
int  main(int argc,char **argv)
{
  if (argc<=1)
    {
      printf("<filename> <T1_image file><T2_image
file>
            <PD_image file>\n");
     exit(1);
    }
  clus =5;
  count =0;
  count1 =0;
  count2 =0;
  count3 =0;
  hi =0;
  d  = XOpenDisplay(NULL);
  id = Imlib_init(d);
  im1 = Imlib_load_image(id,argv[1]);
  im2 = Imlib_load_image(id,argv[2]);
  im3 = Imlib_load_image(id,argv[3]);
  black = BlackPixel(d,0);
  white = WhitePixel(d,0);
  w = im1->rgb_width;
  h = im1->rgb_height;
  size = w*h;
 win = XCreateWindow(d,DefaultRootWindow(d)
   ,0,0,w,h,0,id->x.depth,  InputOutput,id->
x.visual,0,&att);
 win1 = XCreateSimpleWindow(d,RootWindow
(d,0),0,0,w,h,
  2,black,white);
  su = 0;
  ou = 0;
a1= (unsigned char*)malloc(size*3*sizeof(unsigned
char));
a2 = (unsigned char*)malloc(size*3*sizeof(unsigned
char));
b1= (unsigned char*)malloc(size*3*sizeof(unsigned
char));
b2= (unsigned char*)malloc(size*3*sizeof(unsigned
char));
b3= (unsigned char*)malloc(size*3*sizeof(unsigned
char));
bo= (unsigned char*)malloc(size*3*sizeof(unsigned
char));
bs= (unsigned char*)malloc(size*3*sizeof(unsigned
char));
nbuff= (unsigned char*)malloc(size*3*sizeof
(unsigned char));
sa   = (unsigned char*)malloc(size*sizeof(unsigned
char));
dbuff = (unsigned char*)malloc(size*sizeof(unsigned
char));
obuff = (unsigned char*)malloc(size*sizeof(unsigned
char));
pbuff = (unsigned char*)malloc(size*sizeof(unsigned
char));
rbuff = (unsigned char*)malloc(size*sizeof(unsigned
char));
```

```
buff0 = (unsigned char*)malloc(size*sizeof(unsigned
char));
buff1 = (unsigned char*)malloc(size*sizeof(unsigned
char));
buff2 = (unsigned char*)malloc(size*sizeof(unsigned
char));
buff3 = (unsigned char*)malloc(size*sizeof(unsigned
char));
buff4 = (unsigned char*)malloc(size*sizeof(unsigned
char));
hbuff0 = (unsigned char*)malloc(256*sizeof(unsigned
char));
hbuff1 = (unsigned char*)malloc(256*sizeof(unsigned
char));
hbuff2 = (unsigned char*)malloc(256*sizeof(unsigned
char));
hbuff3 = (unsigned char*)malloc(256*sizeof(unsigned
char));
hbuff4 = (unsigned char*)malloc(256*sizeof(unsigned
char));
b1 = im1->rgb_data;
b2 = im2->rgb_data;
b3 = im3->rgb_data;
XSelectInput(d,win,StructureNotifyMask
|ButtonPressMask
   |ButtonReleaseMask | ButtonMotionMask |
KeyPressMask );
XSelectInput(d,win1,StructureNotifyMask
|ButtonPressMask
   |ButtonReleaseMask | ButtonMotionMask |
KeyPressMask);
 gc = XCreateGC(d,win,0,0);
XSetLineAttributes(d,gc,1,LineOnOffDash,CapRound,
JoinRound);
XSetForeground(d,gc,white);
gc1 = XCreateGC(d,win1,0,0);
XSetLineAttributes(d,gc1,1,LineOnOffDash,CapRoun
d,JoinRound);
XSetForeground(d,gc1,white);
gtk_init(&argc,&argv);
window =
gtk_window_new(GTK_WINDOW_TOPLEVEL);
gtk_window_set_title(GTK_WINDOW(window),
          "HUMAN BRAIN SEGMENTATION FROM
MRI");
gtk_signal_connect(GTK_OBJECT(window),"delete_e
                  vent"
   ,GTK_SIGNAL_FUNC(delete_event),NULL);
gtk_container_border_width(GTK_CONTAINER
                  (window),2);
 table = gtk_table_new(11,9,1);
 gtk_container_add(GTK_CONTAINER
                  (window),table);
/***********ORIGINAL***************/
label = gtk_label_new("..ORIGINAL..");
gtk_table_attach_defaults(GTK_TABLE(table),label,3,
6,0,1);
gtk_widget_show(label);
button = gtk_button_new_with_label("T1");
gtk_table_attach_defaults(GTK_TABLE
(table),button,2,3,1,2);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
    GTK_SIGNAL_FUNC(open_T1),NULL);
gtk_widget_show(button);
button = gtk_button_new_with_label("T2");
gtk_table_attach_defaults(GTK_TABLE
(table),button,4,5,1,2);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
   GTK_SIGNAL_FUNC(open_T2),NULL);
gtk_widget_show(button);
button = gtk_button_new_with_label("PD");
gtk_table_attach_defaults(GTK_TABLE
(table),button,6,7,1,2);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
   GTK_SIGNAL_FUNC(open_PD),NULL);
 gtk_widget_show(button);
/***********************************/
separator=gtk_hseparator_new();
```

```
gtk_table_attach_defaults(GTK_TABLE
(table),separator,1,8,2,3);
gtk_widget_show(separator);
/***********TISSUES***********/
label = gtk_label_new("TISSUES");
gtk_table_attach_defaults(GTK_TABLE(table),label,0,
2,3,4);
gtk_widget_show(label);
button = gtk_button_new_with_label("White");
gtk_table_attach_defaults(GTK_TABLE
(table),button,0,1,4,5);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
    GTK_SIGNAL_FUNC(white_matter),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),
   GTK_WIDGET(button),"White Matter",NULL);
button = gtk_button_new_with_label("Gray");
gtk_table_attach_defaults(GTK_TABLE
(table),button,1,2,4,5);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
  GTK_SIGNAL_FUNC(gray_matter),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),
   GTK_WIDGET(button),"Gray Matter",NULL);
button = gtk_button_new_with_label("CSF");
gtk_table_attach_defaults(GTK_TABLE
(table),button,0,1,5,6)
gtk_signal_connect(GTK_OBJECT(button),"clicked",
   GTK_SIGNAL_FUNC(CSF),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),
  GTK_WIDGET(button),"cerebrospinal-fluid",NULL);
 button = gtk_button_new_with_label("Skull");
 gtk_table_attach_defaults(GTK_TABLE
(table),button,1,2,5,6);
 gtk_signal_connect(GTK_OBJECT(button),"clicked",
      GTK_SIGNAL_FUNC(Skull),NULL);
 gtk_widget_show(button);
 tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),
   GTK_WIDGET(button),"fat under skin",NULL);
button = gtk_button_new_with_label("Close");
gtk_table_attach_defaults(GTK_TABLE
     (table),button,0,2,6,7);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
     GTK_SIGNAL_FUNC(close_tissue),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),GTK_WI
       DGET(button),"close tissue viewer",NULL);
/*************************************/
separator=gtk_vseparator_new();
gtk_table_attach_defaults(GTK_TABLE
(table),separator,2,3,3,7);
gtk_widget_show(separator);
/***********Segmentation***********/
label = gtk_label_new("SEGMENTATION");
gtk_table_attach_defaults(GTK_TABLE(table),label,3,
     6,3,4);
gtk_widget_show(label);
button = gtk_button_new_with_label("U");
gtk_table_attach_defaults(GTK_TABLE
(table),button,3,4,4,5);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
     GTK_SIGNAL_FUNC(segment_U),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS
    (tooltip),GTK_WIDGET(button),"determine
     U",NULL);
button = gtk_button_new_with_label("V");
gtk_table_attach_defaults(GTK_TABLE
     (table),button,4,5,4,5);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
     GTK_SIGNAL_FUNC(segment_V),NULL);
```

```
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),GTK_WI
    DGET(button),"determine V",NULL);
button = gtk_button_new_with_label("H");
gtk_table_attach_defaults(GTK_TABLE
    (table),button,5,6,4,5);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
    GTK_SIGNAL_FUNC(segment_H),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),GTK_WI
    DGET(button),"use histogram",NULL);
/************histogram***********/
label = gtk_label_new("HISTOGRAM");
gtk_table_attach_defaults(GTK_TABLE(table),label,3,
    6,5,6);
gtk_widget_show(label);
button = gtk_button_new_with_label("Histogram");
gtk_table_attach_defaults(GTK_TABLE
    (table),button,3,4,6,7);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
    GTK_SIGNAL_FUNC(histogram),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),GTK_WI
    DGET(button),"histogram",NULL);
button = gtk_button_new_with_label("Gray Level");
gtk_table_attach_defaults(GTK_TABLE
    (table),button,4,5,6,7);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
    GTK_SIGNAL_FUNC(gray_level),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
  gtk_tooltips_set_tip(GTK_TOOLTIPS
      (tooltip),GTK_WIDGET(button),"gray level &
      intensity & pointer",NULL);
  button = gtk_button_new_with_label
      ("Close");  gtk_table_attach_defaults
      (GTK_TABLE(table),button,5,6,6,7);
  gtk_signal_connect(GTK_OBJECT(button),"clicked",
      GTK_SIGNAL_FUNC(close_histogram),NULL);
  gtk_widget_show(button);
  tooltip = gtk_tooltips_new();
  gtk_tooltips_set_tip(GTK_TOOLTIPS
      (tooltip),GTK_WIDGET(button),"close
      histogram",NULL);
/*********************************/
  separator=gtk_vseparator_new();
  gtk_table_attach_defaults(GTK_TABLE
(table),separator,6,7,3,7);
  gtk_widget_show(separator);
/**********select area**********/
  label = gtk_label_new("SELECT AREA");
  gtk_table_attach_defaults(GTK_TABLE
(table),label,7,9,3,4);
  gtk_widget_show(label);
  button = gtk_button_new_with_label("Area");
  gtk_table_attach_defaults(GTK_TABLE
(table),button,7,8,4,5);
  gtk_signal_connect(GTK_OBJECT(button),"clicked",
GTK_SIGNAL_FUNC(area),NULL);
  gtk_widget_show(button);
  tooltip = gtk_tooltips_new();
  gtk_tooltips_set_tip(GTK_TOOLTIPS
      (tooltip),GTK_WIDGET(button),"right click for
      end select",NULL);
  button = gtk_button_new_with_label("View");
  gtk_table_attach_defaults(GTK_TABLE
      (table),button,8,9,4,5);
  gtk_signal_connect(GTK_OBJECT(button),"clicked",
      GTK_SIGNAL_FUNC(opens),NULL);
  gtk_widget_show(button);
  tooltip = gtk_tooltips_new();
```

```
   gtk_tooltips_set_tip(GTK_TOOLTIPS
(tooltip),GTK_WIDGET(button),"view selected
area",NULL);
   button = gtk_button_new_with_label("Zoom In");
   gtk_table_attach_defaults(GTK_TABLE
(table),button,7,8,5,6);

gtk_signal_connect(GTK_OBJECT(button),"clicked",
GTK_SIGNAL_FUNC(gain),NULL);
   gtk_widget_show(button);
   tooltip = gtk_tooltips_new();
   gtk_tooltips_set_tip(GTK_TOOLTIPS
(tooltip),GTK_WIDGET(button),          "zoom
in",NULL);
  button = gtk_button_new_with_label("Zoom Out");
  gtk_table_attach_defaults(GTK_TABLE
(table),button,8,9,5,6);
  gtk_signal_connect(GTK_OBJECT(button),"clicked",
GTK_SIGNAL_FUNC(lose),NULL);
  gtk_widget_show(button);
  tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),GTK_WI
    DGET(button),"zoom out",NULL);
button = gtk_button_new_with_label("Close");
gtk_table_attach_defaults(GTK_TABLE
    (table),button,7,9,6,7);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
GTK_SIGNAL_FUNC(close_viewer),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),GTK_WI
    DGET(button),"close area viewer",NULL);
separator=gtk_hseparator_new();
gtk_table_attach_defaults(GTK_TABLE
    (table),separator,1,8,7,8);
gtk_widget_show(separator);
label = gtk_label_new("..FILE..");
gtk_table_attach_defaults(GTK_TABLE(table),label,4,
    5,8,9);
gtk_widget_show(label);
button = gtk_button_new_with_label("Save");
gtk_table_attach_defaults(GTK_TABLE
    (table),button,2,3,9,10);
 gtk_signal_connect(GTK_OBJECT(button),"clicked",
    GTK_SIGNAL_FUNC(save_image),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),GTK_WI
    DGET(button),"save image",NULL);
button = gtk_button_new_with_label("Open");
gtk_table_attach_defaults(GTK_TABLE
    (table),button,4,5,9,10);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
    GTK_SIGNAL_FUNC(open_save),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),GTK_WI
    DGET(button),"open image",NULL);
 button = gtk_button_new_with_label("Exit");
gtk_table_attach_defaults(GTK_TABLE
    (table),button,6,7,9,10);
gtk_signal_connect(GTK_OBJECT(button),"clicked",
    GTK_SIGNAL_FUNC(delete_event),NULL);
gtk_widget_show(button);
tooltip = gtk_tooltips_new();
gtk_tooltips_set_tip(GTK_TOOLTIPS(tooltip),GTK_WI
    DGET(button),"exit program",NULL);
separator=gtk_hseparator_new();gtk_table_attach_de
    faults(GTK_TABLE(table),separator,1,8,10,11);
gtk_widget_show(separator);
gtk_widget_show(table);
gtk_widget_show(window);
gtk_main();
}
/*********************************************************/
```

END.