รายงานการวิจัย

การพัฒนาโปรแกรมฐานข้อมูลและรูปภาพทางการแพทย์

(MEDICAL IMAGE AND DATABASE SOFTWARE)

กวิน สนธิเพิ่มพูน

มนัส นามะสนธิ

สำนักวิจัยและบริการคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ได้รับทุนอุดหนุนการวิจัยจากสำนักงานคณะกรรมการวิจัยแห่งชาติ

พ.ศ. 2537

บทคัดย่อ

ในระบบภาพรังสีเอ็กซ์ของคนไข้เดิมนั้นในการค้นหาหรือการจัดเก็บข้อมูล จะต้องทำการจัดเก็บฟิลม์เอ็กซ์เรย์ของคนไข้พร้อมประวัติ ซึ่งต้องใช้พื้นที่มากในการจัดเก็บและใช้เวลาค้นหานานซึ่งบางครั้งอาจจะหาไม่พบ ดังนั้นจึงเป็นเหตุผลทำการพัฒนาโปรแกรมข้อมูลภาพทางการแพทย์ขึ้น เนื่องจากปัจจุบันมีโปรแกรมการจัดฐานข้อมูลหลายตัวสามารถเก็บข้อมูลภาพร่วมกับข้อมูลอักขระอื่นได้ในการจัดเก็บข้อมูล จะพัฒนาโปรแกรมที่สามารถนำข้อมูลภาพมาทำการประมวลผลภาพทางดิจิตอล เพื่อเป็นเครื่องมือช่วยในการวินิจฉัยของแพทย์โดยพยายามหาวิธีที่เหมาะสมหรับการปรับปรุงภาพเพื่อให้ผลที่ดีที่สุด ภาพที่นำมาจัดเก็บได้มาจากกล้องถ่ายภาพทีวีถ่ายจากฟิลม์เอ็กซ์เรย์ หรือได้จากอุปกรณ์รังสีเอ็กซ์โดยตรง จึงทำให้แพทย์สะดวกขึ้นในการทำงาน สำหรับวิเคราะห์อาการคนไข้

ABSTRACT

This thesis deal with the development of software for restore and diagnostic tool of medical image database. These tool is using digital image processing algorithm to enhance quality image or use as utitlity tool for diagnostic. That medical image recieved from X-ray film or X-ray camera in X-ray TV fluoroscopy directly by connectd to hardware equipment of computer. So user can retrieve image data with other information in database.

สารบัญ

บทที่ 1 บทนำ

1.1 คำนำ 1-1

1.2 วัตถุประสงค์ในโครงงานวิจัยและพัฒนา 1-2

บทที่ 2 ระบบฐานข้อมูลเดิม

2.1 ระบบงานเดิม 2-1

2.2 การจัดเก็บตารางในฐานข้อมูล 2-2

2.3 ลักษณะการทำงานของระบบงาน 2-4

2.4 รูปแบบการจัดเก็บข้อมูลภาพ 2-7

บทที่ 3 พื้นฐานการประมวลผลภาพดิตอล

3.1 วิธีการสปาเชี่ยนโดเมน 3-1

3.2 การทำคอนโวลูชั่นในสปาเชี่ยนโดเมน 3-2

3.3 การเปิดตาราง 3-6

บทที่ 4 วิธีการปรับปรุงภาพเพื่อช่วยในการวินิจฉัย

4.1 การขยายภาพ 4-1

4.2 การปรับความสว่างของภาพ 4-2

4.3 การปรับปรุงคอนทราส 4-4

4.4 การปรับปรุงฮิสโตแกรมของภาพ 4-8

4.5 การทำภาพให้ราบเรียบ 4-16

4.6 การทำภาพให้คมชัด 4-20

4.7 การหาขอบภาพ 4-23

4.8 การหาค่าเทรสโฮวด์ 4-26

บทที่ 5 การปรับปรุงภาพโดยแยกส่วนอวัยวะของภาพเอ็กซ์เรย์ทรวงอก

5.1 การหาค่าเทรสโฮวด์โดยอัตโนมัติ 5-1

5.2 การปรับปรุงภาพด้วยวิธีฮิสโตแกรมอิควอไรเซชั่นแบบแยกส่วน 5-14

5.3 การทำอันชาร์ปมาส์กแบบปรับได้แปรตามค่าระดับเทาภาพ 5-14

บทที่ 6 บทสรุปและข้อเสนอแนะ 6-1

บทที่ 1
บทนำ

1. บทนำ

ในปัจจุบันการเก็บรักษาภาพถ่ายเอ็กซ์เรย์มีจำนวนมากตามโรงพยาบาลต่างๆ รวมทั้งระบบข้อมูล ของรายละเอียดส่วนบุคคลของผู้ที่มาทำการเอ็กซ์เรย์มีจำนวนมาก และยุ่งยากต่อการค้นหาข้อมูลของผู้ป่วย ในแต่ละราย เพื่อให้แพทย์ผู้ทำการตรวจมาวินิจฉัย ที่จะต้องเก็บรักษาฟิลม์เอ็กซ์เรย์ และประวัติต่างๆ ของคนไข้ ซึ่งไม่สะดวกและล่าช้าในการทำงานของ แพทย์ที่ทำการอ่านและวินิจฉัยฟิลม์เอ็กซ์เรย์นั้น ตลอดจนถึงขั้นตอนและเวลาในการที่จะฉายรังสีเอ็กซ์เรย์ในแต่ละครั้งจะต้องผ่านขั้นตอนในการล้างภาพฟิลม์เอ็กซ์เรย์ดังกล่าวดังนั้นจึงมีการคิดที่จะนำคอมพิวเตอร์มาประยุกต์ใช้รวมถึงนำทฤษฎีทางด้านการประมวลผลข้อมูลภาพ (Digital Image Processing)มาใช้ เพื่อจะลดขั้นตอนและเวลาในการทำงานให้รวดเร็วยิ่งขึ้น สำหรับอุปกรณ์ที่นำมาใช้ในการรับข้อมูลภาพนั้นจะใช้กล้องถ่ายภาพทีวีถ่ายจากฟิลม์เอ็กซ์เรย์หรือต่อโดยตรงจากกล้องถ่ายภาพทีวีที่มีอยู่ในระบบอุปกรณ์รังสีเอ็กซ์อยู่แล้ว สัญญาณภาพที่ได้จากระบบอุปกรณ์รังสีเอ็กซ์นั้น เป็นสัญญาณในระบบ PAL ในการแปลงข้อมูลจากสัญญาณชนิดนี้เป็นสัญญาณทางดิจิตอลที่ใช้ในคอมพิวเตอร์จะต้องใช้การ์ดดิจิไตซ์ที่สามารถแปลงข้อมูลชนิดนี้ได้ เพื่อที่จะแปลงข้อมูลเป็นไบนารีและจัดรูปแบบ (format) ให้ข้อมูลภาพ เป็นชนิดบิตแมต โดยจะเก็บข้อมูลชนิดนี้ในฐานข้อมูลร่วมกับข้อมูลอื่นสามารถเรียกดูข้อมูลเกี่ยวกับประวัติคนไข้ได้พร้อมๆ กับภาพเอ็กซ์เรย์นี้ สำหรับในส่วนที่จะนำมาแสดงผลภาพทางจอคอมพิวเตอร์นั้นจะนำทฤษฎีทางด้านการประมวลผลข้อมูลภาพมาใช้ในการปรับปรุงภาพ (Image Enhancement) เพื่อช่วยเหลือในการดูข้อมูลภาพที่ภาพต้นแบบมีลักษณะไม่ชัดเจนนัก ตัวอย่างสำหรับอัลกอริทึมต่างๆ ที่จะนำมาใช้นั้นเช่น การปรับปรุงภาพโดยวิธีฮิสโตแกรมอิควอไรเซชั่น (Histogram Equalization) วิธีการนี้ทำเพื่อที่จะกระจายระดับความเข้มของภาพให้มีลักษณะเท่ากัน เพื่อที่จะให้ภาพไม่ออกไปในลักษณะโทนใดโทนหนึ่งมากเกินไปจะทำให้สามารถมองเห็นรายละเอียดของภาพในส่วนที่ไม่ชัดเจนนักได้ดีขึ้น , การทำภาพให้คมชัด (Image Sharpening) ใช้ในกรณีที่ข้อมูลภาพที่ได้มีลักษณะมัว (blur) จะใช้วิธีการนี้เพื่อให้ภาพคมชัดนั่นเอง , การทำภาพให้ราบเรียบ (Image Smoothing) วิธีการนี้ใช้ในกรณีที่ภาพที่ได้มีสัญญาณรบกวน วิธีการนี้สามารถที่จะลดสัญญาณรบกวนลงไปได้ , การหาขอบภาพ (Edge Detection) วิธีการนี้ใช้เพื่อที่จะดูขอบภาพของรูปนั้น สำหรับภาพทางการแพทย์นั้นมีอยู่หลายชนิดด้วยกัน เช่น ภาพอุลตร้าซาวด์,ภาพเอ็กซ์เรย์สมอง , ภาพเอ็กซ์เรย์ทรวงอก เป็นต้น กรณีของภาพเอ็กเรย์ทรวงอกเป็นภาพที่มีรายละเอียดของภาพมาก จึงได้ทำการเสนอวิธีการปรับปรุงภาพทรวงอกโดยแยกส่วนอวัยวะของส่วนที่เป็นปอดกับอวัยวะฉากหลังโดยการหาเทรสโฮวด์โดยอัตโนมัติ เพื่อที่แยกการปรับปรุงภาพใน

ทำฮิสโตแกรมอิควอไรเซชั่นด้วยการแยกฮิสโตแกรมในแต่ละส่วน และการปรับภาพให้คมชัดด้วยวิธีอันชาร์ปมาส์กแบบปรับได้ตามลักษณะของภาพ

โปรแกรมนี้ใช้ไมโครซอฟท์แอกแซส(Microsoft Access) ในการจัดการฐานข้อมูล ส่วนโปรแกรมที่ใช้ในการดึงข้อมูลภาพมาประมวลผลทางดิจิตอลใช้บอร์ดแลนด์ซีเวอร์ชั่น 3.1 (Borland C Version 3.1) บนไมโครซอฟท์วินโดว์ สำหรับเนื้อหาในวิทยานิพนธ์นี้จะเน้นในส่วนการนำเอาอัลกอริทึมต่าง ๆ ในการประมวลผลภาพเป็นส่วนใหญ่

1.2 วัตถุประสงค์ของโครงการวิจัย

เพื่อที่จะพัฒนาโปรแกรมที่เก็บข้อมูลภาพรังสีเอ็กซ์เรย์ในฐานข้อมูล และสามารถที่จะนำเอาอัลกอริทึมการประมวลผลภาพมาใช้เพื่อช่วยในการวินิจฉัยภาพเพื่อเพิ่มประสิทธิภาพของระบบงานให้ดีกว่าเดิม อุปกรณ์ที่ใช้ในโครงงานได้แก่ การ์ดแสดงผลที่สามารถแปลงสัญญาณภาพเป็นข้อมูลดิจิตอลยี่ห้อ TARGA รุ่น TARGA+ 16/32P,กล้องถ่ายวีดีโอขาวดำชนิดซีซีดียี่ห้อ SONY รุ่น SSC_M370E สำหรับขอบเขตของวิทยานิพนธ์ฉบับนี้ มีดังต่อไปนี้

บทที่ 1 บทนำ

บทที่ 2 เป็นการกล่าวถึงระบบฐานข้อมูลเดิมว่ามีลักษณะการทำงาน การจัดเก็บตารางในฐานข้อมูล รูปแบบการจัดเก็บข้อมูลภาพ การติดต่อเชื่อมโยงกับโปรแกรม การประมวลผลข้อมูลภาพเป็นอย่างไร

บทที่ 3 จะกล่าวความรู้พื้นฐานที่ควรทราบในการประมวลผลภาพดิจิตอล เช่นการทำงานบนสปาเซี่ยนโดเมนหรือโดเมนความถี่ การคำนวณทางคณิตศาสตร์โดยการทำคอนโวลูชั่นบนสปาเซี่ยนโดเมนที่เป็นพื้นฐานของการคำนวณที่ใช้ในอัลกอริทึมต่างๆ การเพิ่มความเร็วในการคำนวณผลด้วยวิธีการเปิดตาราง

บทที่ 4 กล่าวถึงอัลกอริทึมและวิธีการต่างๆที่ทางการประมวลผลภาพที่นำมาใช้ในการปรับปรุงภาพหรือเป็นเครื่องช่วยในการปรับปรุงภาพ เช่น การขยายภาพ, กา ร ปรับความสว่างของภาพ, การปรับปรุงคอนทราส , การปรับปรุงฮิสโตแกรมของภาพ , การทำภาพให้เรียบ , การทำภาพให้คมชัด , การหาขอบภาพ , การหาค่าเทรสโฮวด์

บทที่ 5 เป็นการปรับปรุงภาพโดยทำการแยกภาพออกเป็นสองส่วนเป็นภาพวัตถุกับฉากหลังแล้วทำการปรับปรุงภาพแต่ละส่วนด้วยวิธีฮิสโตแกรมอิควอไรเซชั่น จะสามารถกระจายฮิสโตแกรมได้กว้างขึ้นเนื่องจากใช้ฮิสโตแกรมเฉพาะส่วนนั้น รวมไปถึงการทำอันชาร์ปมาส์กิ้งแบบปรับได้

บทที่ 6 บทสรุปของผลของวิธีการต่างๆ และข้อเสนอแนะจากการทำวิทยานิพนธ์นี้

บทที่ 2
ระบบฐานข้อมูลเดิม

2.1 ระบบงานเดิม

สำหรับภาพทางการแพทย์เป็นข้อมูลสำคัญที่แพทย์ใช้ประกอบวินิจฉัย ตัวอย่างภาพทางการแพทย์ที่ใช้บ่อยได้แก่ภาพจากเครื่องเอ็กซ์เรย์ธรรมดา เครื่องอุลตร้าซาวด์ หรือ เครื่องเอ็กซ์เรย์คอมพิวเตอร์ รังสีแพทย์เป็นผู้แปรผลภาพเหล่านี้เพื่อให้แพทย์ที่ส่งผู้ป่วยนำ ข้อมูลไปใช้โดยการถ่ายภาพ ลงบนแผ่นฟิลม์และพิมพ์รายงานผล การเก็บภาพบนฟิล์มมีข้อ เสียคือ ราคาแพง สิ้นเปลืองเนื้อที่ในการเก็บ และกรณีฟิลม์หายไม่สามารถเรียกกลับมา ได้ต้องนำผู้ป่วยมาทำการตรวจใหม่ นอกจากนี้การส่งข้อมูลให้แพทย์ผู้รักษาล่าช้า

ฐานข้อมูลเดิมนั้นเป็นฐานข้อมูลของโรงพยาบาลรามาธิบดี ที่พัฒนาขึ้นบน ไมโครซอฟท์วินโดว์ โดยใช้ซอฟ์ทแวร์ในการจัดการฐานข้อมูลไมโครซอฟ์ทแอกเซส (Microsoft Access) โปรแกรมดังกล่าวเป็นผลงานของภาควิชารังสีเอ็กซ์เรย์โรงพยาบาลรามาธิบดีร่วมมือกับสถาบันเทคโนโลยีแห่งเอเซีย สำหรับความสามารถของโปรแกรมดังกล่าวมีดังนี้คือ บันทึกข้อมูลภาพเอ็กซ์เรย์จากอุปกรณ์รังสีเอ็กซ์เรย์หรือถ่ายจากฟิลม์เอ็กซ์เรย์ของผู้ป่วย การค้นหาข้อมูลภาพการตรวจเก่า, การลบข้อมูลภาพออกฐานข้อมูล การพิมพ์รายงานผลการตรวจออกทางเครื่องพิมพ์เลเซอร์ การเก็บข้อมูลของผู้ป่วยที่ใช้นั้นจะเก็บประวัติของผู้ป่วย เช่น หมายเลขประจำตัวผู้ป่วย วันเดือนปีเกิดผู้ป่วย เพศ ในการที่ผู้ป่วยจะมาถ่ายเอ็กซ์เรย์นั้นจะต้องผ่านการ ตรวจจากแผนกอื่น ๆ ของโรงพยาบาลก่อนที่จะนำผู้ป่วยส่งมาถ่ายภาพรังสีเอ็กซ์ ที่แผนกรังสีวิทยา เช่นผู้ป่วยอาจจะถูกส่งมาจากแผนกอายุรกรรม, ศัลยกรรม, ทันตกรรม,หรือ สูตินารีเป็นต้น ซึ่งจะต้องระบุอวัยวะที่ทำการตรวจว่าเป็นส่วนใด ข้อมูลที่จะต้องทราบในการตรวจแต่ละครั้งคือวันที่ทำการตรวจ, ลำดับที่ในการตรวจในหนึ่งวัน, จำนวนภาพที่จะต้องทำการตรวจ หนึ่งคนอาจจะต้องมีการตรวจหรือถ่ายฟิลม์มากกว่าหนึ่งครั้ง, ห้องที่ใช้ในการตรวจ, การบันทึกจุดสำคัญในการตรวจของแพทย์และจะต้องบันทึกรายงานของ แพทย์ การจัดเก็บข้อมูลต่าง ๆ ในฐานข้อมูลนั้นได้ออกแบบไว้เป็น 3 ตาราง ตารางที่ 1 จะเป็นตารางที่ใช้เก็บข้อมูลประวัติของผู้ป่วย เพื่อใช้เป็นข้อมูล อ้างอิงของผู้ป่วยจะจัดเก็บ หมายเลขประจำตัวผู้ป่วย, ชื่อนามสกุลของผู้ป่วย, วันเดือน ปีเกิด, เพศ ตารางที่ 2 เป็นการจัดเก็บข้อมูลรายชื่อแผนกของโรงพยาบาลทั้งหมด ตารางที่ 3 จะใช้ตารางหลักในการทำงานของระบบงานนี้จะจัดเก็บประวัติการตรวจ เอ็กซ์เรย์ของผู้ป่วย ซึ่งภาพเอ็กซ์เรย์จะเก็บอยู่ในตารางข้อมูล สำหรับรายละเอียดของ การเก็บข้อมูลในแต่ละตารางจะกล่าวในหัวข้อต่อไป

2.2 การจัดเก็บตารางในฐานข้อมูล

ตารางที่ใช้ในระบบงานนั้นมีอยู่ 3 ตาราง ได้แก่ ตารางเก็บประวัติส่วนตัวของผู้ป่วย, ตารางที่เก็บรหัสและชื่อแผนกต่างๆ,ตารางเก็บประวัติการตรวจเอ็กเรย์ผู้ป่วยมีรายละเอียดดังนี้

ตารางที่1 ตารางเก็บประวัติส่วนตัวของผู้ป่วย (Table Patiant)

เลขที่	ชื่อ	ชนิดข้อมูล	ขนาด	คำอธิบาย
1*	HNNO	Character	8	หมายเลขประจำตัวผู้ป่วย
2	Name	Character	45	ชื่อสกุลผู้ป่วย
3	BrithDate	Date	8	วันเดือนปีเกิดผู้ป่วย
4	Sex	Numeric	1	เพศ

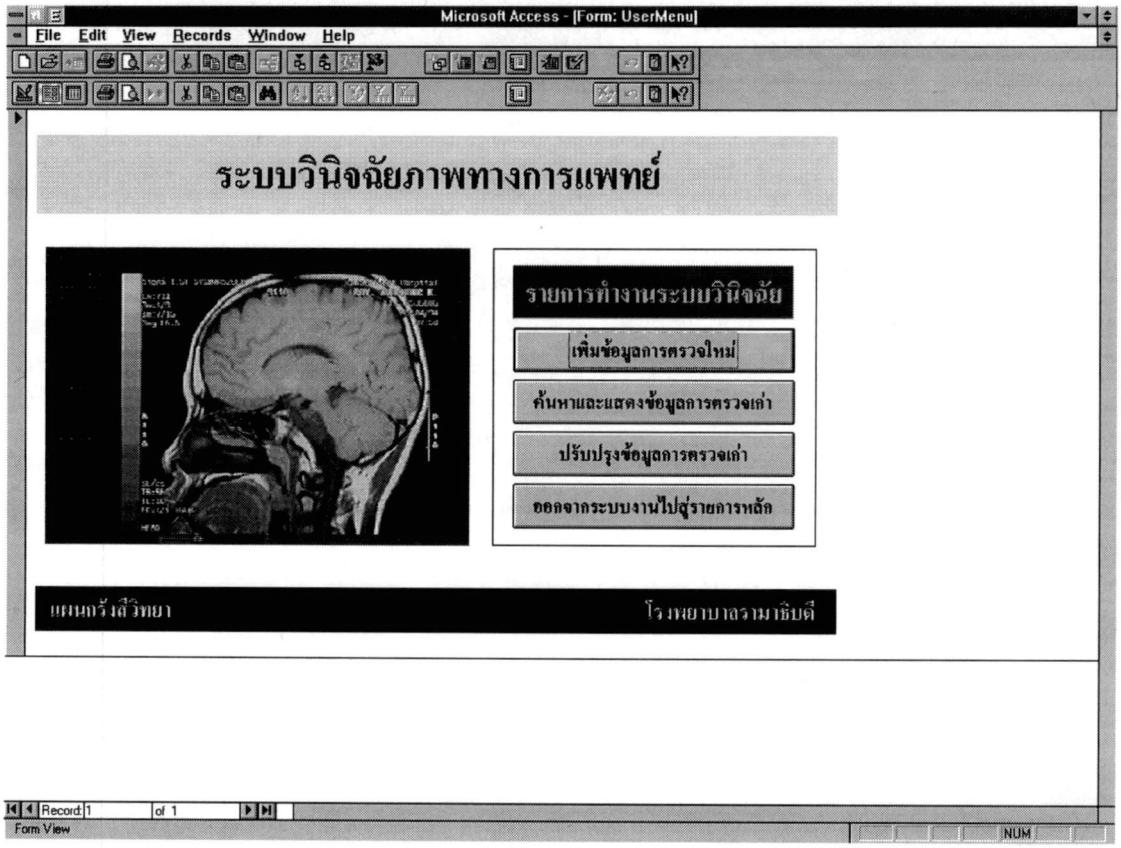
ตารางที่2 ตารางเก็บรหัสและชื่อแผนกต่างๆ (Table Location)

เลขที่	ชื่อ	ชนิดข้อมูล	ขนาด	คำอธิบาย
1*	Location	Character	3	แผนกที่ทำการส่งผู้ป่วย
2	SectionName	Character	20	ชื่อแผนก

ตารางที่3 ตารางการเก็บประวัติการตรวจเอ็กซ์เรย์ของผู้ป่วย (Table X-Rays)

เลขที่	ชื่อ	ชนิดข้อมูล	ขนาด	คำอธิบาย
1*	HNNO	Character	8	หมายเลขประจำตัวผู้ป่วย
2*	ExamDate	Date	8	วันที่ทำการตรวจ
3*	ExamNo	Numeric	2	ลำดับที่ในการตรวจในหนึ่งวัน
4*	RunNo	Numeric	2	ลำดับที่ของรูปภาพในการตรวจครั้งหนึ่งๆ
5	ExamTime	Numeric	2	เวลาในการตรวจ
6	Photo	Picture	-	รูปภาพจากการตรวจ
7	Room	Character	5	ห้องที่ใช้ในการตรวจ
8	Note	Character	20	การบันทึกจุดสำคัญ
9	Report	Character	40	รายงานแพทย์
10	Location	Character	3	แผนกที่ทำการส่งผู้ป่วยมาตรวจ
11	Organ	Character	20	อวัยวะที่ทำการตรวจ

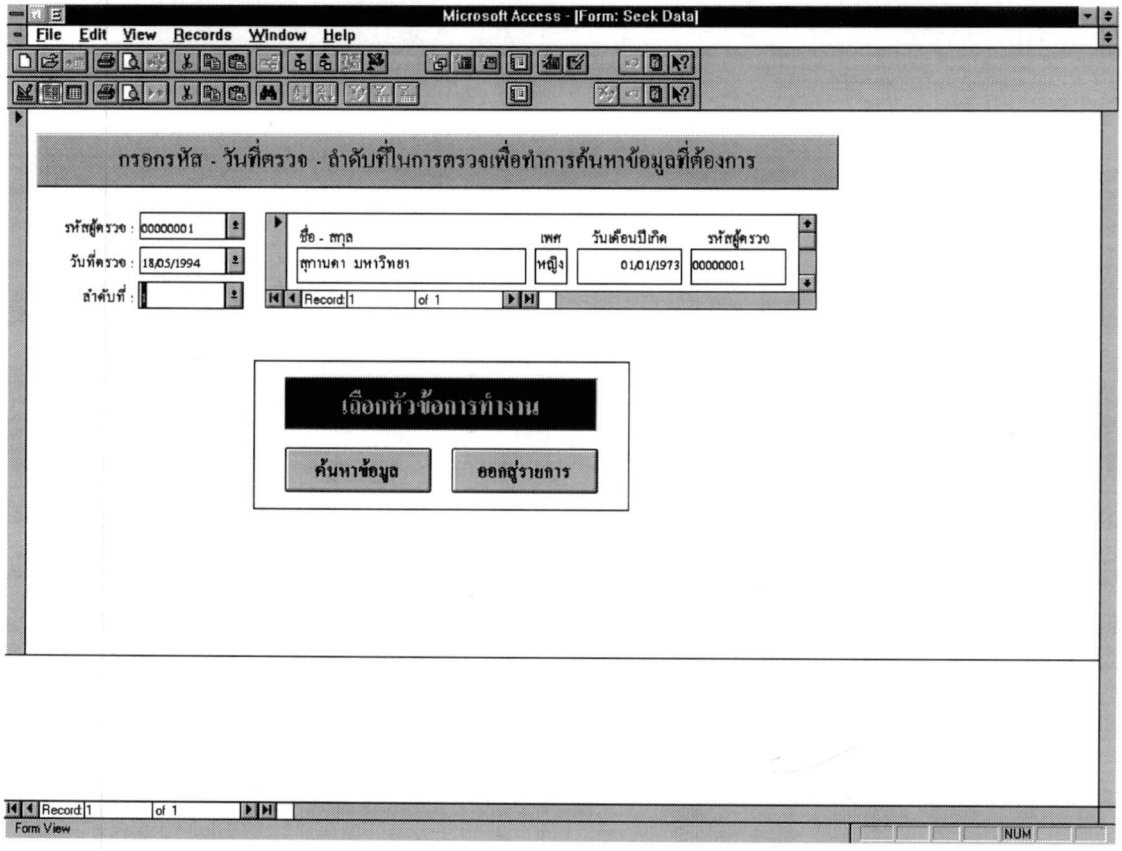
หมายเหตุ * หมายถึง คอลัมภ์ที่ใช้เป็นดัชนีในการค้นหาข้อมูล



รูปที่ 2.1 แสดงรายการทำงานระบบวินิจฉัยทางการแพทย์

2.3 ลักษณะการทำงานของระบบงาน

จากรูปที่ 2.1 จะแสดงรายการต่างๆที่ใช้ในระบบงานได้แก่ การเพิ่มเติมการตรวจใหม่ , การค้นหาข้อมูลการตรวจเก่า , การปรับปรุงข้อมูล การเก็บข้อมูลจะเก็บไว้ในตารางที่ได้กล่าวไว้แล้วในส่วนการค้นหาข้อมูลนั้นจะใช้คอลัมภ์รหัสผู้ตรวจ ,วันที่ตรวจ ,ลำดับที่ในการตรวจ เป็นดัชนีในการค้นหาข้อมูล เนื่องจากในการตรวจผู้ป่วยแต่ละครั้ง จะทำการเอ็กซ์เรย์และเก็บภาพมากกว่าหนึ่งภาพดังนั้นในการแสดงผลข้อมูลภาพจะแสดงภาพเอ็กซ์เรย์ทั้งหมดของผู้ป่วย โดยที่หนึ่งจอภาพสามารถแสดงภาพได้ 12 ภาพ สามารถจะดูรายงานแพทย์ของภาพได้ทีละหนึ่งภาพ สำหรับหน้าจอการแสดงรายงานแพทย์ผู้ทำการตรวจ แสดงในรูปที่ 2.3 (a) จากหน้าจอนี้ถ้าต้องการจะดูเฉพาะส่วนที่เป็นภาพอย่างเดียวให้เลือกปุ่มแรกสุดของรายการ Tool เมื่อแสดงภาพดังต้องการแล้ว ถ้าต้องการที่จะทำการปรับปรุงภาพหรือประมวลผลข้อมูลภาพทางดิจิตอลให้เลือกปุ่ม Process เพื่อที่เป็นเครื่องมือช่วยเหลือในการวินิจฉัยภาพของแพทย์ ในส่วนการทำงานเพิ่มเติมข้อมูลใหม่และปรับปรุงข้อมูลการตรวจเก่านั้นจะมีลักษณะการทำงานคล้ายๆกัน สำหรับฟังก์ชั่นต่างๆที่ใช้ในการประมวลผลข้อมูลภาพจะกล่าวถึงในบทต่อไป

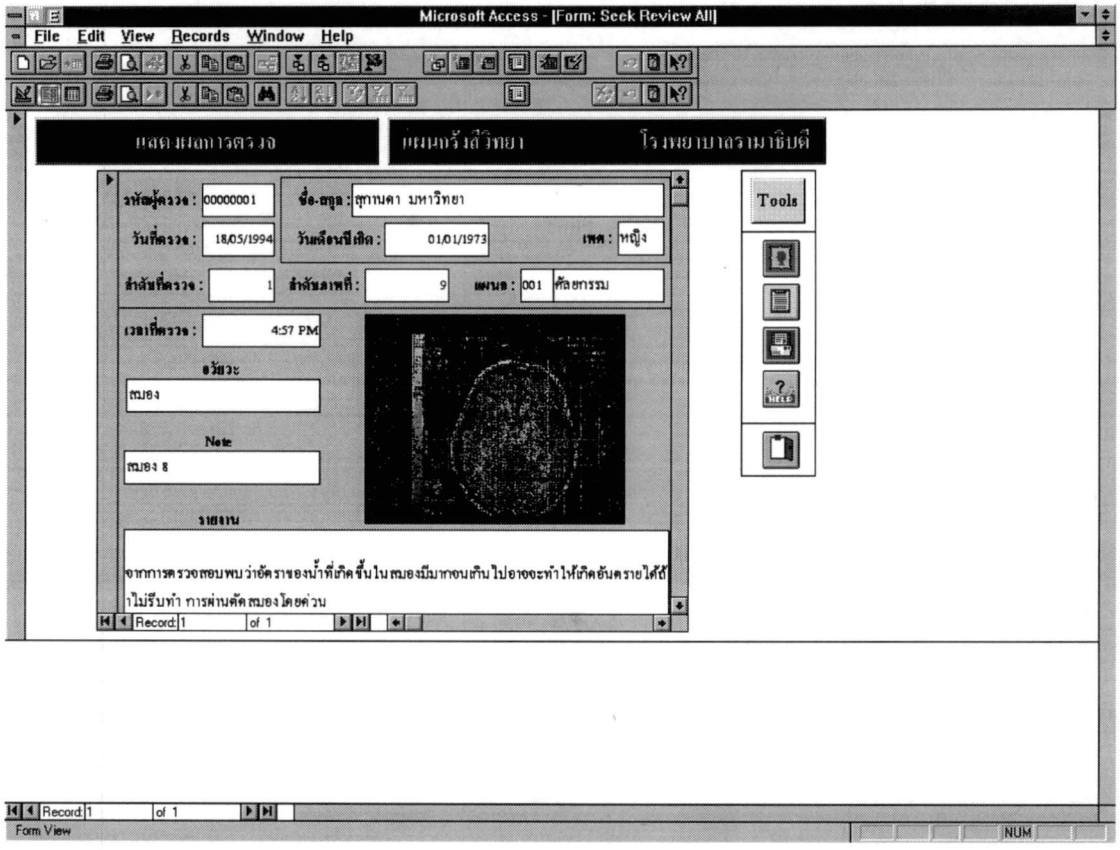


(a)

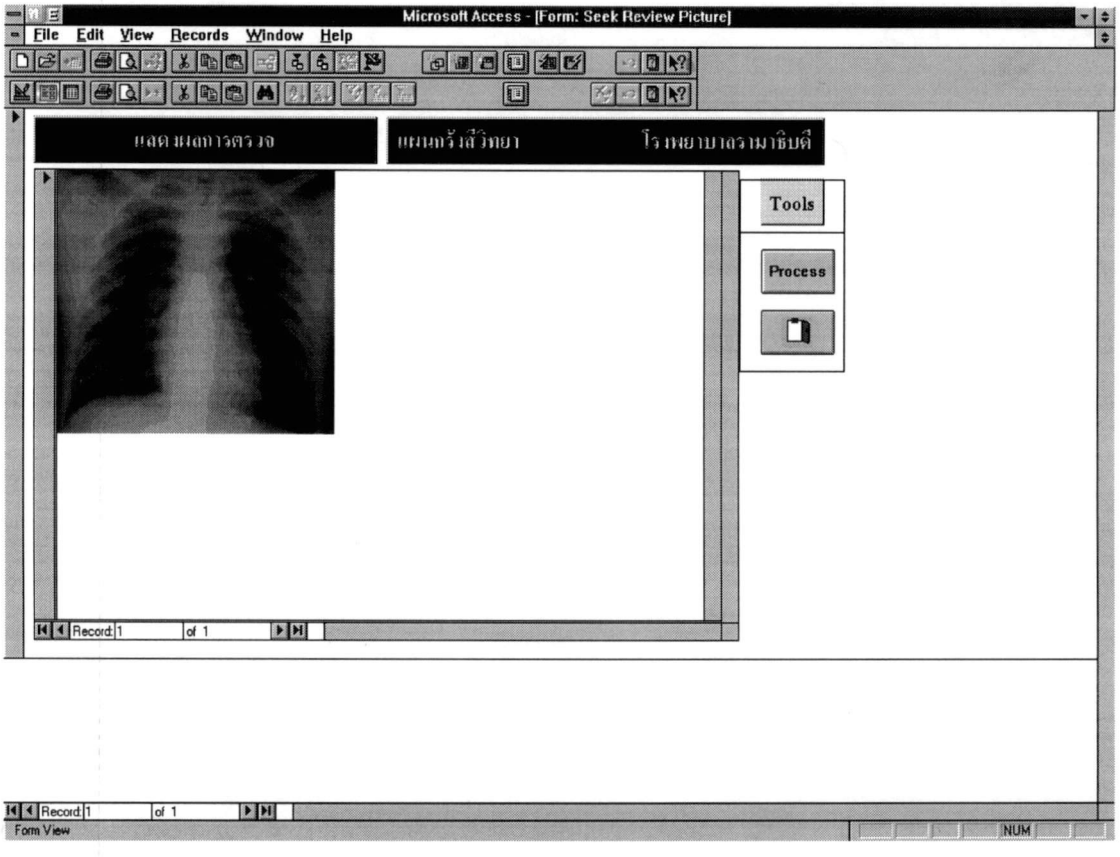
(b)

รูปที่ 2.2 (a) แสดงการค้นหาข้อมูลจากรหัสผู้ตรวจ , วันที่ตรวจ และลำดับที่ในการตรวจ

(b) แสดงข้อมูลภาพเอ็กซ์เรย์ของผู้ป่วย



(a)



(b)

รูปที่ 2.3 (a) แสดงรายงานต่างๆ ของภาพที่ทำการตรวจ (b) แสดงหน้าจอที่จะเข้าสู่การประมวลผลข้อมูลภาพ

2.3.2 รูปแบบข้อมูลภาพถ่ายโดยทั่วไป

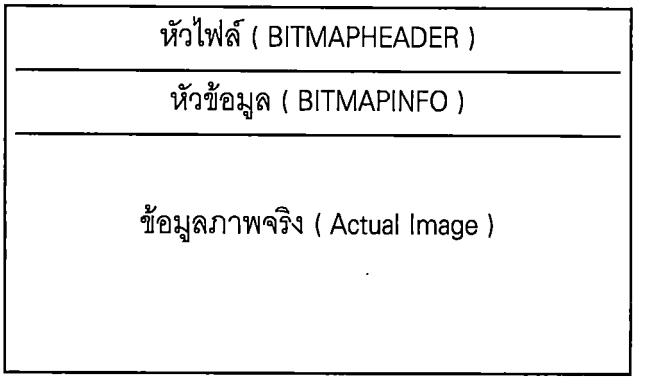
รูปแบบที่มีใช้อยู่โดยทั่วไปมีรูปแบบดังนี้คือ GIF IFF/LBM WPG BMP PIC TGA PCX และ TIFF สำหรับรูปแบบใหญ่ ๆ ที่นิยมใช้กันมากคือ PCX GIF BMP และ TIFF แต่รูปแบบหนึ่งที่ได้รับความนิยมมากที่สุด คือ BMP หรือ บิตแมพ สาเหตุก็เพราะว่าข้อมูล แบบบิตแมพนั้นไม่มีการบีบข้อมูล (COMPRESSION) จึงมีความรวดเร็วและสะดวกในการ อ่านข้อมูลอย่างมาก และเป็นรูปแบบข้อมูลกลาง ที่ใช้ในการแลกเปลี่ยนระหว่างกันของ โปรแกรมประยุกต์ต่าง ๆ ใช้ในการส่งข้อมูลไปให้อุปกรณ์ต่าง ๆ เช่น อุปกรณ์การแสดง ผลเป็นต้น แต่ในรูปแบบบิตแมพนี้ต้องเปลืองเนื้อที่ฮาร์ดดิส ในการจัดเก็บข้อมูลมากกว่า รูปแบบอื่นที่ใช้ในการเข้ารหัสบีบข้อมูล ซึ่งโปรแกรมแอกแซสก็ใช้รูปแบบไฟล์ชนิดนี้

2.3.2.1 รูปแบบข้อมูลแบบบิตแมพ (BMP)

รูปแบบสำหรับใช้กับโปรแกรมระบายสีในวินโดว์ version 3.0 เป็นต้นมา โดยรูปแบบข้อมูลแบบนี้สามารถเป็นข้อมูลสีได้ตั้งแต่ 1 ถึง 24 บิท มีส่วนหัว(header) ในการบอกรายละเอียดต่าง ๆ ของภาพ โดยกำหนดในลักษณะโครงสร้าง (structure) ในภาษาระดับสูง ในที่นี้ใช้ภาษาซี ข้อมูลภาพในรูปแบบบิทแมพประกอบด้วยสาม ส่วนคือ

- หัวไฟล์ (BITMAPFILEHEADER)
-หัวข้อมูล (BITMAPINFO) ซึ่งรวมทั้งข้อมูลต่าง ๆ และพาเลตของสี (color palette)
- ข้อมูลภาพจริง

ในรูปที่ 2.5 แสดงส่วนประกอบกันของโครงสร้างที่ใหญ่ที่สุดของภาพถ่ายในรูปแบบบิทแมพ ซึ่งข้อมูลพาเลตสี และข้อมูลภาพจะเปลี่ยนโครงสร้าง ไปตามรูปแบบของจำนวนสีของภาพ และวิธีการถอดรหัส ที่ใช้ในการบีบข้อมูลภาพ ดังจะได้กล่าวต่อไป



รูปที่ 2.5 แสดงโครงสร้างที่ใหญ่ที่สุดของภาพถ่ายในรูปแบบบิทแมพ

2.3.2.1.1 หัวไฟล์ (BITMAPFILEHEADER)

หัวของไฟล์ (BITMAPFILEHEADER) จะประกอบด้วยข้อมูลเกี่ยวกับชนิดของ รูปแบบข้อมูลภาพ ขนาดของภาพและโครงร่าง ดังนี้

BITMAPFILEHEADER ประกอบด้วย

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

โดย bfType เป็นชนิดของรูปแบบข้อมูลภาพในที่นี้สำหรับบิทแมพคือ "BM"
bfSize เป็นขนาดของไฟล์ทั้งหมด
bfReserved1,2 ไม่ได้ใช้งาน ปกติกำหนดให้มีค่าเท่ากับศูนย์
bfOffBits เป็นขนาดของหัวไฟล์ทั้งหมดใช้เพื่อข้ามไปจุดเริ่มต้นของข้อมูลภาพจริง

2.3.2.1.2 หัวข้อมูล (BITMAPINFO)

หัวข้อมูลจะเป็นตัวกำหนดขนาดต่าง ๆ (dimensions) และข้อมูลสี (color information) ดังนี้

BITMAPINFO ประกอบด้วย

- BITMAPINFOHEADER bmiHeader;
- RGBQUAD bmiColors[1];

โดยที่ bmiHeader เป็นข้อมูลเกี่ยวกับขนาดต่าง ๆ จำนวนสี และรูปแบบของสี
bmiColors เป็นอาร์เรย์ข้อมูลสีอาร์จีบีที่เป็นตัวกำหนดสีในการแสดงผลของภาพที่เป็นรายละเอียดต่าง ๆ จะอยู่ใน BITMAPINFO HEADER ดังนี้

STRUCTURE BITMAPIINFOHEADER ประกอบด้วย

-DWORD biSize;

-DWORD biWidth;

-DWORD biHeight;

-WORD biPlanes;

-WORD biBitCount;

-DWORD biCompression;

-DWORD biSizeimage;

-DWORD biXPelsPerMeter;

-DWORD biYPelsPerMeter;

-DWORD biCirUsed;

-DWORD biCirImportant;

โดยที่ biSize เป็นขนาดของส่วนหัวข้อมูล (BITMAPINFOHEADER) มีหน่วยเป็นไบต์

biWIdth เป็นความกว้างของภาพ (มีหน่วยเป็น จุดภาพส้น)

biHeight เป็นความสูงของภาพ (มีหน่วยเป็น เส้น)

biPlanes เป็นจำนวนหน้าของสี (color plane) สำหรับอุปกรณ์ เป้าหมาย (targetdevice) ปกติกำหนดให้เป็น 1 เสมอ

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

biCompression ชนิดของการบีบอัด (compression) ดูรายละเอียดได้ในตารางที่ 2.2

biSizeImage เป็นขนาดของภาพ (มีหน่วยเป็นไบต์)

biXPelsPerMeter ความละเอียดสำหรับอุปกรณ์เป้าหมายในแนวนอนต่อหนึ่งเมตร

biYPelsPerMeter ความละเอียดสำหรับอุปกรณ์เป้าหมายในแนวตั้งต่อหนึ่งเมตร

biCirUsed จำนวนดัชนีสี (color index) ในตารางสี (color table) มีค่าได้ 3 กรณีโดยดูรายละเอียดได้ในตารางที่ 2.3

biCirImportant จำนวนความสำคัญของดัชนีสี (color index) ในการแสดงผล ถ้าเป็นศูนย์ ทุกสีจะมีความสำคัญเท่ากันหมด

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

ตารางที่ 2.1 แสดง จำนวนบิทต่อจุดภาพ ของ biBitCount

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

ตารางที่ 2.2 แสดงรูปแบบการบีบอัด (compression formats) ของ biCompression

ค่า	อธิบาย	
0	ใช้จำนวนของสีสูงสุดเท่ากับ 2 biBitCount	
ไม่เป็นศูนย์	ถ้า biBitsCount<24 biCirUsed จะระบุ จำนวนสีที่ใช้ในการแสดงผล เช่นเมื่อ biBitsCount เท่ากับ 8 จะมีค่า biCirUsed เท่ากับ 256 สี เป็นต้น	ถ้า biBitsCount=24 biCirUsed จะระบุ ขนาดของ ตารางอ้างอิง (reference table) เพื่อให้วินโดว์ได้รับรู้และ ใช้ในการแสดงผลพาเลต สีให้เหมาะสมที่สุด

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

สำหรับส่วน RGBQUAD ที่เก็บรายละเอียดของ palette สีประกอบด้วยส่วนประกอบดังนี้

STRUCTURE RGBQUAD ประกอบด้วย

BYTE rgbBlue;

BYTE rgbGreen;

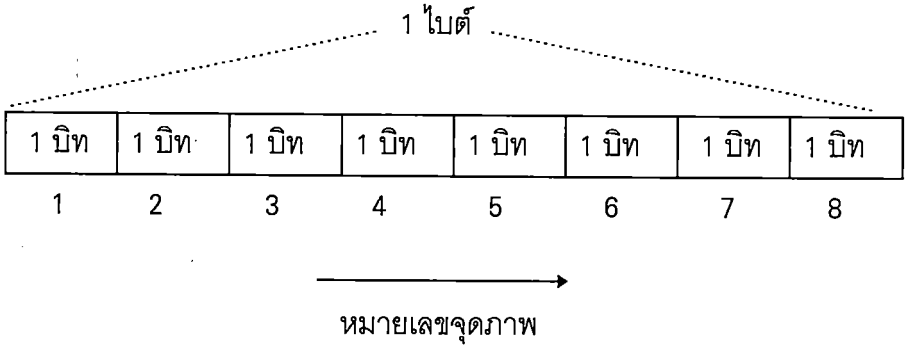
BYTE rgbRed;

BYTE rgbReserved; ไม่ใช้งาน (ปกติตั้งให้เป็นศูนย์)

2.3.2.1.3 ข้อมูลภาพจริง (actual image information)

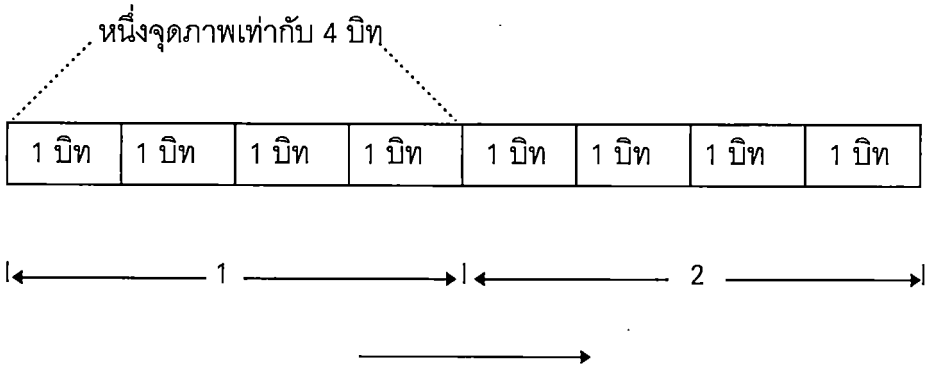
ส่วนของข้อมูลบิตแมพจริงจะมีการเก็บที่แตกต่างกันตามจำนวนบิทของข้อมูลภาพ (biBitcount) และลักษณะการบีบอัด biCompression แต่โดยทั่วไปแล้วข้อมูลในรูปแบบบิตแมพปกติที่นิยมในปัจจุบันในการส่งข้ามระหว่างโปรแกรมประยุกต์ต่าง ๆ นั้น จะไม่ใช้การบีบอัดข้อมูล ดังนั้น จึงแบ่งรูปแบบข้อมูลในลักษณะที่นิยมใช้กันในปัจจุบันได้เพียงสามกรณีดังนี้

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



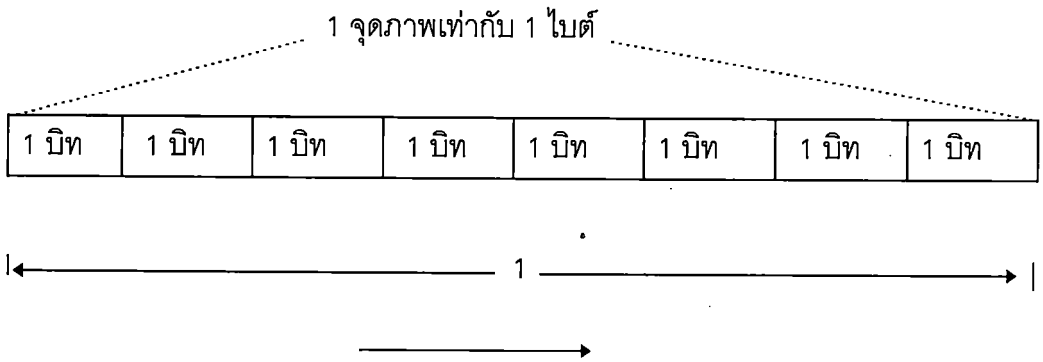
รูปที่ 2.6 แสดงข้อมูลภาพบิตแมพ 2 สี (monochrome Bitmap)

กรณีที่ 2 ถ้าเป็น 4 บิทต่อจุดภาพ หมายความว่ามีดัชนีสี bimiColors ที่เป็น แอร์เรย์ จำนวน 16 แอร์เรย์ คือมี 16 สี โดยข้อมูลแต่ละ 4 บิท จะแทนหนึ่งจุดภาพ ดังรูปที่ 2.7



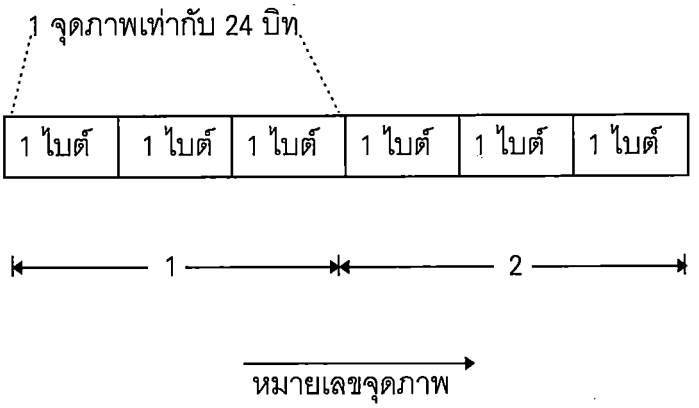
รูปที่ 2.7 แสดงข้อมูลภาพบิตแมพ 16 สี

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



รูปที่ 2.8 แสดงข้อมูลภาพบิตแมพ 256 สี

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



รูปที่ 2.9 แสดงข้อมูลภาพบิตแมพ 16.7 ล้านสี

ภาพที่ใช้เก็บในฐานข้อมูลเป็นภาพขนาด 8 บิตต่อจุดภาพ เป็นภาพระดับเทา(Gray Scale) ขาวดำ 256 ระดับ การผสมสีในตารางเทียบสี ค่าสีแดง(R),ค่าสีเขียว(G),ค่าสีน้ำเงิน(B) จะมีค่าเท่ากันในทุกระดับตารางเทียบสี

บทที่ 3
พื้นฐานการประมวลผลภาพดิจิตอล

วิธีการปรับปรุงภาพดิจิตอล เราสามารถแบ่งได้ 2 แบบคือ สปาเชี่ยนโดเมน (Spatial Domain) และโดเมนความถี่ (Frequency Domain) ในที่นี้จะกล่าวถึงเฉพาะวิธีการสปาเชี่ยนโดเมน

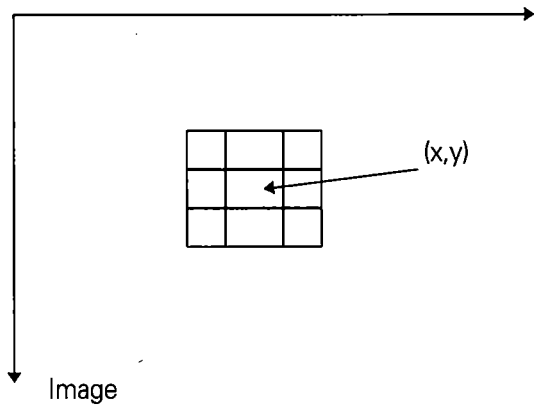
3.1 วิธีการสปาเชี่ยนโดเมน (Spatial Domain)

วิธีการนี้เราจะปรับปรุงภาพหรือกระทำ โดยตรงกับจุดในภาพ (Pixel) ฟังก์ชั่นในการปรับปรุงภาพในวิธีนี้สามารถเขียนเป็นสมการได้ดังนี้

$$g(x,y) = T [f(x,y)]$$

ซึ่ง f(x,y) ก็คือภาพที่จะนำมาใช้ในการประมวลผล g(x,y) คือภาพที่ประมวลผลแล้วและ *T* ก็คือตัวปฏิบัติการ(operation)บน f ที่กำหนดในบริเวณ (x,y)

ในการกำหนดบริเวณรอบ ๆ (neighborhood) ของจุด (x,y) เราสามารถกำหนดได้โดยใช้บริเวณสี่เหลี่ยมจตุรัสได้ดังรูปข้างล่างนี้ การกำหนดบริเวณของจุด (x,y) ยังมีการกำหนดรูปร่างอย่างอื่นด้วยเช่น วงกลม เป็นต้น แต่ไม่เป็นที่นิยม



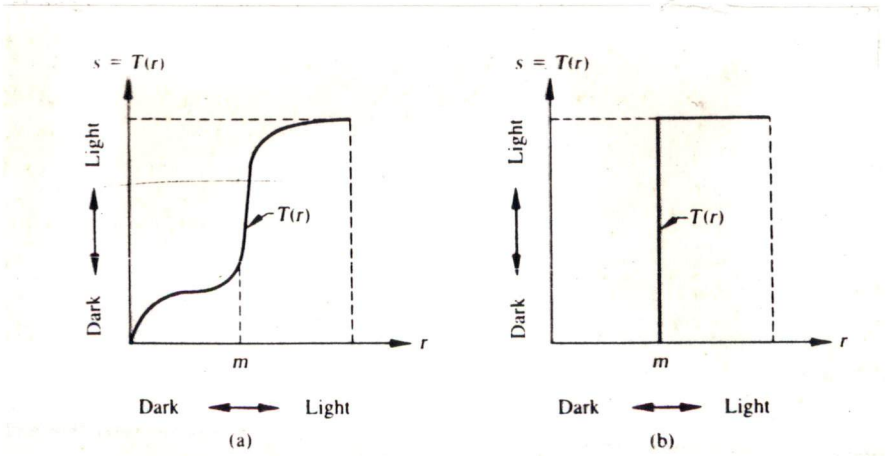
รูปที่ 3.1 แสดงบริเวณ 3x3 ล้อมรอบที่จุด (x,y) ในภาพ

แบบที่ง่ายที่สุดของตัวปฏิบัติการ *T* มีบริเวณ 1X1 ในกรณีที่ g จะขึ้นอยู่กับค่าของ f ที่ (x,y) เท่านั้นและ *T* จะกลายเป็นฟังก์ชั่นแปลง (Tranformation Function) หรือ

ฟังก์ชั่นการแมป (Mapping Function) ซึ่งอยู่ในรูป

$$s = T(r)$$

เมื่อ r และ s คือตัวแปรที่ใช้แทนค่าระดับความเข้มของ f(x,y) และ g(x,y) ที่จุด (x,y) ถ้า T (r) มีรูปแบบในภาพที่ 3.2(a) ผลของการปรับปรุงภาพจะทำให้ภาพที่มีระดับความเข้มต่ำกว่า m ถูกบีบให้อยู่ในช่วงแคบ ๆ ของ s และจะมีผลตรงกันข้ามกันในช่วงที่มีระดับความเข้มสูงกว่า m เทคนิคนี้เราเรียกว่าการปรับค่าคอนทราส (Contrast Stretching) และถ้า T(r) มีรูปแบบในภาพที่ 3.2 (b) ผลของการทำก็จะทำให้ภาพที่ใช้ฟังก์ชั่นนี้มีระดับความเข้ม 2 ระดับ



รูปที่ 3.2 แสดงฟังก์ชั่นการแปลงของระดับความเข้มสำหรับการปรับปรุงค่าคอนทราส (Contrast Enhancement)

3.2 <u>การทำคอนโวลูชันในสปาเชียนโดเมน</u>

การทำคอนโวลูชันเป็นการทำงานในรูปแบบที่เรียกว่าเป็นการทำงานที่ต้องใช้จุดข้างเคียง(Neighbourhood Operation)มาพิจารณาในการคำนวณการกระทำกับจุดข้างเคียงนั้นผลลัพธ์ได้ค่าออกมา ณ ตำแหน่งเดียวกันกับอินพุต และใช้จุดข้างเคียงที่อยู่รอบ ๆ ซึ่งการทำคอนโวลูชันเป็นการคำนวณผลรวมของผลคูณ (sum of product) รูปที่ 3.3 แสดงหน้ากากของคอนโวลูชัน (convolution mask) และจุดข้างเคียงจากภาพ สมาชิกแต่ละตัวในหน้ากากเรียกว่าน้ำหนัก (weight) ค่าน้ำหนักในหน้ากากเป็นตัวคำนวณผลของการทำคอนโวลูชัน ซึ่งจะกำหนดลักษณะการกรองตามการประยุกต์ใช้ในงานต่าง ๆ

M(-1,-1) M(0,-1) M(1,-1)

M(-1, 0) M(0, 0) M(1, 0)

M(-1, 1) M(0, 1) M(1, 1)

(a) หน้ากากของการทำคอนโวลูชั่นขนาด 3x3

F(i-1,*j-1*) F(i,*j-1*) *F(i+1,j-1)*

F(*i-1*,*j*) F(*i*,*j*) F(*i+1*,*j*)

F(*i-1*,*j+1*) F(*i*,*j+1*) F(*i+1*,*j+1*)

(b) จุดข้างเคียง (neighbourhood) ต่าง ๆ ในบริเวณเมตริกซ์
ขนาด 3x3 ของภาพต้นแบบ

รูปที่ 3.3 แสดงคู่ของการทำคอนโวลูชั่นในสปาเชียลโดเมน

ตัวชี้ในหน้ากากมีจุดเริ่มต้นที่จุดศูนย์กลาง โดยเริ่มที่ตำแหน่งมุมบนด้านซ้าย ของภาพ สามารถเขียนสมการคอนโวลูชันได้ดังนี้

$$C(i,j) = \sum_{k=i-1}^{i+1}\sum_{l=j-1}^{j+1} F(k,l)M(i-k,k-l) \qquad (3.1)$$

แต่ละจุดภาพจะเป็นการรวนเอามาจากจุดรอบข้างและคูณด้วยค่าที่ตรงตำแหน่งเดียวกันกับหน้ากาก นำผลคูณของแต่ละตัวนั้นมารวมกันก็จะได้ ค่าจุดภาพผลลัพธ์ ซึ่งการคำนวณที่ทำนั้นสามารถกระจายแสดงออกมาได้ดังนี้

$$\begin{aligned} C(i,j) = {} & F(i-1,j-1)M(1,1)+F(i,j-1)M(0,1)+F(i+1,j-1)M(-1,1)+ \\ & F(i-1,j\ )M(1,0)+F(i,j\ )M(0,0)+F(i+1,j\ )M(-1,0)+ \\ & F(i-1,i+1)M(1,-1)+F(i,j+1)M(0,-1)+F(i+1,j+1)M(-1,-1) \end{aligned} \qquad (3.2)$$

จากสมการข้างบนแสดงการจัดเรียงเทอมต่าง ๆ โดยลำดับในหน้ากากมีลำดับ จากบนซ้ายไปยังด้านล่างขวา สำหรับค่าตำแหน่งที่ตรงกันในหน้ากากจะมีลำดับตรงกันข้าม คือเริ่มจาก

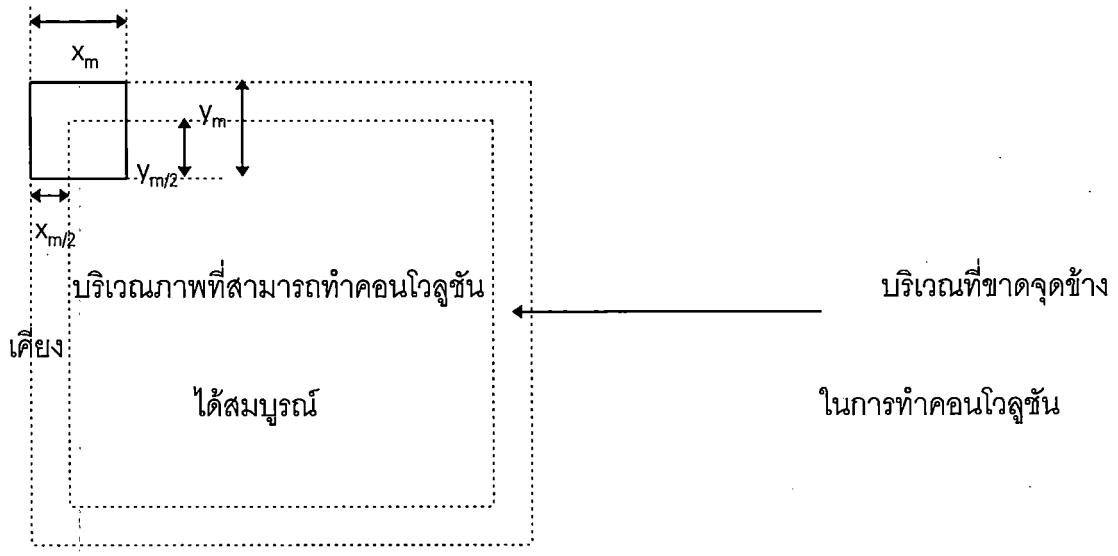
ด้านล่างขวาขึ้นไปยังด้านบนซ้าย สำหรับสิ่งที่น่าสนใจของการตรงกันข้ามของ ลำดับนี้ก็คือว่าถ้าหมุนหน้ากาก 180 องศา ลำดับของเทอมภาพกับเทอมหน้ากาก จะมีลำดับในทิศทางเดียวกันโดยได้การ กระจายดังนี้

$$C(i,j) = F(i-1,j-1)M(-1,-1)+F(i,j-1)M(0,-1)+F(i+1,j-1)M(1,-1)+$$
$$F(i-1,j\ )M(-1,\ 0)+F(i,j\ )M(0,\ 0)+F(i+1,j\ )M(1,\ 0)+ \qquad (3.3)$$
$$F(i-1,i+1)M(-1,-1)+F(i,j+1)M(0,\ 1)+F(i+1,j+1)M(1,\ 1)$$

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

- การขาดจุดข้างเคียงในบริเวณของการทำคอนโวลูชั่นในบริเวณกรอบภาพ

ในขบวนการคำนวณแบบ Neighbourhood operation นี้เป็นไปไม่ได้ที่จะทำคอนโวลูชันในบริเวณขอบรอบนอกของภาพ (บริเวณกรอบภาพ) เพราะในการคำนวณ ขาดจุดภาพที่อยู่ภายนอกอาณาบริเวณของภาพ ดังแสดงในรูปที่ 3.4



รูปที่ 3.4 การขาดจุดข้างเคียงในบริเวณของการทำคอนโวลูชันในบริเวณกรอบภาพ

ดังนั้นโดยทั่วไปแล้ววิธีการที่จะแก้ปัญหาเหล่านี้ได้ ก็มักใช้วิธีการกำหนดแถบจุดภาพในบริเวณกรอบของภาพทั้ง 4 ด้านให้เป็นค่าเดิม หรือเป็นศูนย์แล้วแต่กรณี ความกว้างของแถบด้านบนและด้านล่างของภาพมีค่าเท่ากับ $y_m/2$ และด้านข้างซ้ายและขวาเท่ากับ $X_m/2$ เมื่อ x_m คือความกว้างของหน้ากาก และ y_m คือความสูงของหน้ากากตัวอย่างเช่น หน้ากากขนาด 3x3 จะมีแถบที่มีความกว้าง 1 จุดภาพซึ่งไม่สามารถคำนวณได้รอบ ๆ ภาพ

ในการใช้งานจะต้องรู้ว่าจะนำภาพที่ได้ไปทำอะไรต่อไป เพราะอาจจะเป็นไปได้ที่ภาพผลลัพธ์อาจจะมีขนาดเล็กกว่าภาพต้นแบบ ตามจำนวนแถว และคอลัมน์ที่ไม่สามารถทำการคำนวณได้ ถ้าเงื่อนไขไม่เป็นเช่นนั้นและต้องการรักษาขนาดของภาพผลลัพธ์ ให้เท่ากับภาพต้นแบบก็ใช้การใส่ค่าเดิมหรือศูนย์แล้วแต่กรณี ลงไปในกรอบที่ไม่สามารถคำนวณได้ ซึ่งวิธีนี้เป็นที่นิยมใช้กันอย่างมากเพราะภาพผลลัพธ์สามารถนำไปเปรียบเทียบกับภาพต้นแบบได้ โดยตรงและในอนาคตก็สามารถนำไปคำนวณกันได้ต่อไปในขณะที่ยังรักษาแนวของจุดภาพให้ตรงกันอยู่อย่างนั้น

บางระบบต้องการการคำนวณที่สมบูรณ์แบบอย่างแท้จริงโดยการใช้ข้อมูลในบริเวณ อื่น ๆ มาใช้แทนบริเวณแถวและคอลัมน์ของจุดภาพที่อยู่เลยออกไปจากภาพต้นแบบ วิธีการหนึ่งนั้นทำได้โดยการใช้โรว์ของทางด้านล่างของภาพมาใช้เป็นข้อมูลที่ต้องการ ของการคำนวณทางด้านบนและในทำนองเดียวกัน โดยการใช้แถวของทางด้านบนของภาพมาใช้เป็นข้อมูลที่ต้องการของการคำนวณทางด้านล่าง ในลักษณะที่คล้ายกันของคอลัมน์ทางด้านขวาของภาพก็นำมาใช้เป็นข้อมูลที่ต้องการของการคำนวณทางด้านซ้ายเสร็จสมบูรณ์ และคอลัมน์ทางด้านซ้ายของภาพก็นำมาใช้เป็นข้อมูลที่ต้องการของการคำนวณทางด้านขวา เช่นกัน ถึงแม้ว่าวิธีการเช่นนี้ไม่เป็นการเหมาะสมนักก็ตาม แต่เพื่อให้ผลลัพธ์มีขนาดภาพเท่ากับภาพต้นแบบ และในการประยุกต์ใช้งานบางอย่าง ต้องหลีกเลี่ยงการเกิดการเพี้ยนอย่างรุนแรงของวิธีการคำนวณจากการแทนกรอบที่คำนวณไม่ได้ด้วยศูนย์เช่นในกรณีทำเกรเดียนท์เป็นต้น

บางระบบสามารถใช้วิธีแทนบริเวณกรอบที่คำนวณไม่ได้นั้น ด้วยกรอบของภาพต้นแบบในตำแหน่งที่ตรงกันได้อย่างทันที ถ้าหากเป็นการประยุกต์ที่ผลลัพธ์ค่อนข้างคล้ายกับภาพต้นแบบอย่างมากเช่นการทำให้เรียบ (smoothing) เป็นต้น ถ้าไม่มีสัญญาณรบกวนอยู่ในภาพต้นแบบในส่วนของกรอบก็จะไม่สามารถสังเกตุข้อแตกต่างได้เลย

- เวลาที่ใช้

การทำคอนโวลูชันค่อนข้างจะใช้การคำนวณมากครั้ง เช่นถ้าเป็นหน้ากาก 3x3 ในการที่จะได้ผลลัพธ์ 1 จุดภาพจะต้องใช้การคูณ 9 ครั้ง การบวก 9 ครั้ง และการหารอีก 1 ครั้ง ถ้ามีการทำ

นอร์มอลไลซ์พิจารณาการทำคอนโวลูชันของหน้ากาก ที่มีโรว์เท่ากับคอลัมน์แล้วจำนวนการบวกจะเท่ากับจำนวนการคูณ ตารางที่ 3.1 แสดงจำนวนครั้งที่ต้องใช้ในการคำนวณ โดยทดลองใช้หน้ากากขนาดต่าง ๆ โดยใช้ภาพต้นแบบขนาด 512x512 โดยสมมุติให้สามารถทำคอนโวลูชันที่บริเวณกรอบของภาพได้ ตัวอย่างเมื่อใช้หน้ากากขนาด 3x3 ต้องใช้จำนวนครั้งในการคำนวณเท่ากับ 2,359,296x2+262,144 = 4,980,736 ครั้งเป็นต้น

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

ตารางที่ 3.1 ภาระในการคำนวณโดยการทดลองใช้ตารางหน้ากากจตุรัสขนาดต่าง ๆ กับภาพต้นแบบขนาด 512x512 จุดภาพ

เมื่อขนาดของหน้ากากใหญ่ขึ้นจะเห็นว่าจำนวนครั้งในการใช้คำนวณ จะเพิ่มขึ้นมากอย่างมาก ซึ่งในตารางนั้นจะเป็นการคิดจำนวนครั้งในการคำนวณเท่านั้น มันไม่ได้นับการทำงานที่แฝงอยู่อื่น ๆ ที่จำเป็นต้องใช้เลย ได้แก่การแยกจุดภาพที่ถูกต้องจากภาพ และหน้ากากเขียนเก็บข้อมูลของภาพผลลัพธ์ รักษาดัชนีตัวชี้และตัวนับและควบคุมการทำงานทั้งหมด ดังนั้นจำนวนครั้งในการทำงานทั้งหมดในโครงสร้างการประมวลผล จึงใช้เวลามากกว่าที่แสดงในตารางข้างต้นมาก ดังนั้นเมื่อต้องมีการทำคอนโวลูชัน จึงมักนิยมใช้ขนาดของหน้ากากที่เล็กที่สุดเท่าที่จะทำได้

3.3 การเปิดตาราง (look-up table)

ในขบวนการเพิ่มประสิทธิภาพของการประมวลผลภาพนั้น มีการใช้การเปิดตารางกันอย่างกว้างขวางในการประยุกต์ใช้งานต่าง ๆ ที่ต้องการความเร็วสูง เช่นการปรับปรุงคอนทราสของ

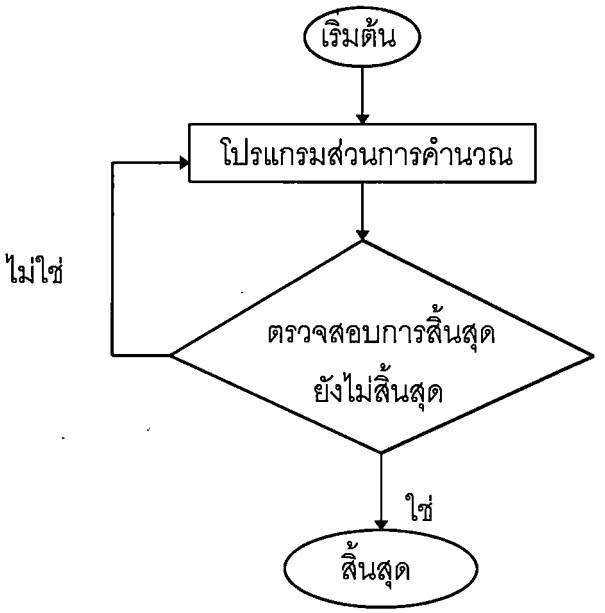
ภาพที่เป็นการแปลง (transformation) ของค่าระดับสีเทาแต่ละจุดภาพไปยังค่าระดับสีเทาใหม่ สำหรับการปรับปรุงภาพอย่างง่ายแล้ว ฟังก์ชันการแปลงจะเป็นฟังก์ชันเดียวกันสำหรับ ทุก ๆ จุดในภาพดังนั้นอาจเขียนได้ว่า

$$G'_L = T(G_L) \qquad (3.5)$$

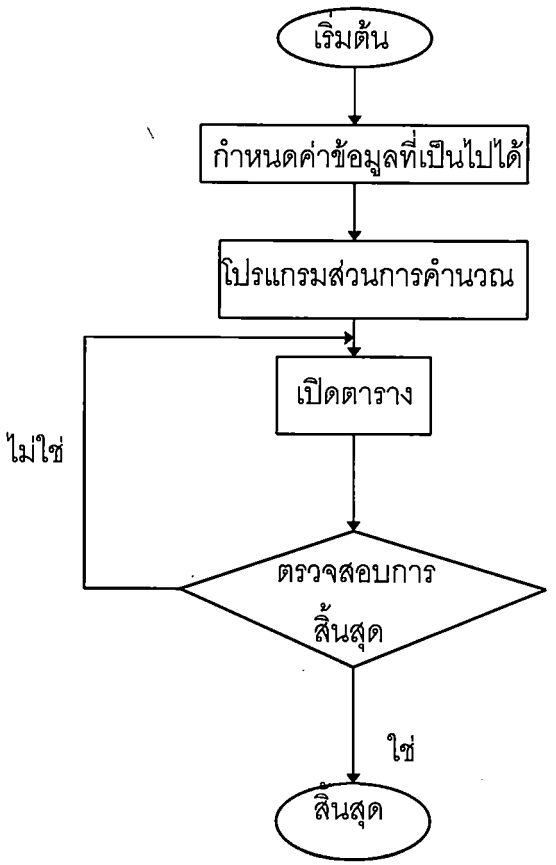
โดยที่ G_L และ G'_L เป็นค่าระดับสีเทาอินพุทและเอาต์พุทตามลำดับ และมี T คือฟังก์ชันการแปลง

ขั้นตอนในการทำการประมวลผลโดยตรงแสดงดังรูปที่ 3.5(a) ส่วนในการใช้วิธีการเปิดตารางแสดงดังรูปที่ 3.5(b) ซึ่งเป็นการเขียนโดยภาษาระดับสูงต่าง ๆ ฟังก์ชันการแปลง $T(G_L)$อาจจะกำหนดให้เป็นแอร์เรย์ (array) ที่มีจำนวนเต็ม (integer) G_L เป็นตัวชี้อยู่ ถ้ามีการใช้ฟังก์ชัน T เดียวกันในการแปลงทุก ๆ จุดภายในภาพจึงสามารถใช้การคำนวณสำหรับทุก ๆ ค่าที่เป็นไปได้ของตัวชี้ G_L ก่อนการประมวลผล และทำการเก็บค่าระหว่างการคำนวณค่าต่าง ๆ ไว้ในแอร์เรย์ ภาพต้นแบบที่มีขนาด 8 บิทต่อหนึ่งจุดภาพจะต้องการตารางแอร์เรย์จำนวน 256 แถว ในการประมวลผลถัดมาก็เป็นการนำค่าระดับสีเทาของจุดภาพต่าง ๆ มาเป็นตัวชี้ตาราง T ค่าที่สอดคล้องของ T จะเป็นค่าระดับสีเทาเอาต์พุท จะไม่มีการคำนวณระหว่างการเปิดตารางอีก ใช้เพียงตัวชี้ตำแหน่งที่อยู่ในตำแหน่งตารางที่ต้องการเท่านั้น

การเปิดตารางสามารถใช้สำหรับจุดประสงค์อื่น ๆ ได้อีกมากมาย ประหยัดจำนวนครั้งในการคำนวณซึ่งหมายถึงเวลาที่ใช้ก็ลดลงนั้นเองเห็นได้ชัดเจนในการคำนวณเลขทศนิยมที่ใช้เวลาในการคำนวณสูง



(a) วิธีการคำนวณโดยตรง



(b)วิธีการเพิ่มความเร็วโดยการเปิดตาราง

รูปที่ 3.5 แสดงลำดับขั้นตอนวิธีการคำนวณโดยตรงเทียบกับการใช้วิธีการเปิดตาราง

บทที่ 4
วิธีการปรับปรุงภาพเพื่อช่วยในการวินิจฉัยภาพ

ในบทนี้จะกล่าวถึงอัลกอริทึมในการปรับปรุงภาพ ต่างๆ ที่นำมาใช้ในระบบงานโดยจะกล่าวถึงเป็นหัวข้อดังนี้

4.1 การขยายภาพ (ZOOMING)

บ่อยครั้งที่จำเป็นต้องการขยายภาพ เพื่อดูรายละเอียดของภาพที่ใหญ่ขึ้นจะได้มองเห็นรายละเอียดเล็กๆได้ชัดเจนขึ้น การขยายภาพนั้นมีอยู่ 2 วิธีด้วยกันคือ

4.1.1 การทำซ้ำ (Replication)

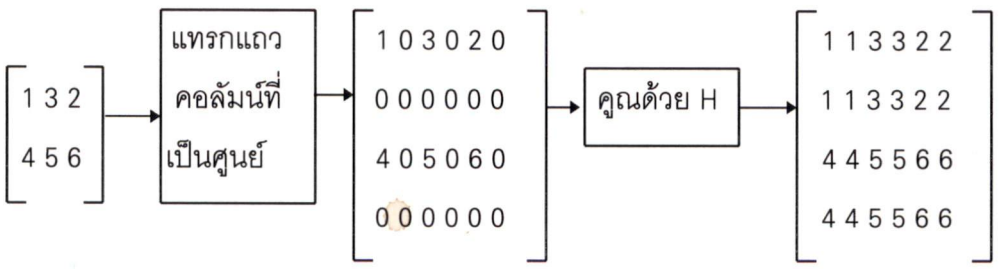
การทำซ้ำคือ เมื่อต้องการขยายภาพให้มีขนาดใหญ่ขึ้นก็จะแทรกแถวและคอลัมน์ที่เป็นศูนย์ไปทั้งแนวนอนและแนวตั้งของภาพ เช่น ถ้าต้องการขยายภาพขนาด MxN ถ้าแทรกแถวและคอลัมภ์ที่เป็นศูนย์เข้าไปจะได้เมตริกซ์ขนาด 2Mx2N และทำการคูณ (convolve) ด้วยเมตริก H ที่มีขนาด

$$H = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

ดังนั้นจะได้

$$g(m,n) = f(k,l), k = \text{Int}[m/2], \; L = \text{Int}[n/2] \qquad m,n=0,1,2,$$

โดยที่ g(m,n)เป็นฟังก์ชันที่ทำการขยาย และ f(k,l)เป็นฟังก์ชันเดิม จะแสดงตัวอย่างของการขยายภาพด้วยวิธีการทำซ้ำ ดังนี้



การขยายภาพด้วยวิธีนี้คือ การขยายขนาดของจุด ซึ่งภาพที่ได้จะมีลักษณะไม่เรียบ เป็นสี่เหลี่ยมของจุดติดกันไปทั้งภาพเพื่อที่จะแก้ปัญหานี้จะต้องใช้วิธีการแทรกแบบเชิงเส้นตรง

4.1.2 <u>การแทรกแบบเชิงเส้นตรง</u> (Linear Interpolation)

การทำการแทรกแบบเชิงเส้นตรงจะพิจารณาจุดตามแนวเส้นตรงที่จุดที่กำลังทำนั้น แทรกอยู่ระหว่างจุดในแถวหรือแนวตั้งจุดนั้นจะทำการอินเทอร์โพเรชั่นลักษณะตามแนวเส้นตรง ตัวอย่างมีเมตริกซ์ขนาด 2x2 การทำการแทรกเชิงเส้นตรงตามแนวนอน จะได้ดังนี้

$$g_1(m,2n) = f(m,n) \qquad , 0 \le m \le M-1,\ 0 \le n \le N-1 \qquad (4.1.2\text{-}1)$$

$$g_1(m,2n+1) = 1/2[f(m,n)+f(m,n+1)] \qquad , 0 \le m \le M-1,\ 0 \le n \le N-1$$

การทำแทรกเชิงเส้นตรงตามแนวตั้งจากผลลัพธ์ครั้งแรก

$$g(2m,n) = g_1(m,n) \qquad (4.1.2\text{-}2)$$

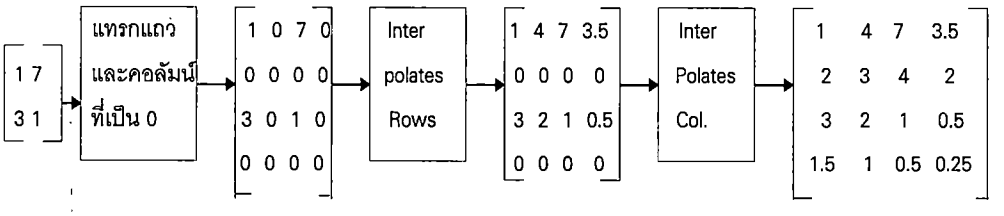
$$g(2m+1,n) = [g_1(m,n)+v_1(m+1,n)] \quad , 0 \le m \le M-1,\ 0 \le N \le 2N-1$$

สมมุติว่าภาพที่เริ่มต้นมีแถวและคอลัมน์รอบนอกเป็นศูนย์ขนาด [0,M-1] x [0,N-1] วิธีการที่กล่าวมาข้างต้นนั้นสามารถทำได้โดยคอนโวโลชั่นภาพขนาด 2Mx2N ที่ทำการแทรกแถวและคอลัมน์ที่เป็นศูนย์จากภาพเดิมด้วยเมตริกซ์

$$H = \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & \boxed{1} & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}$$

จุดศูนย์กลางของเมตริกซ์ H คือจุดที่กำลังพิจารณาในภาพ

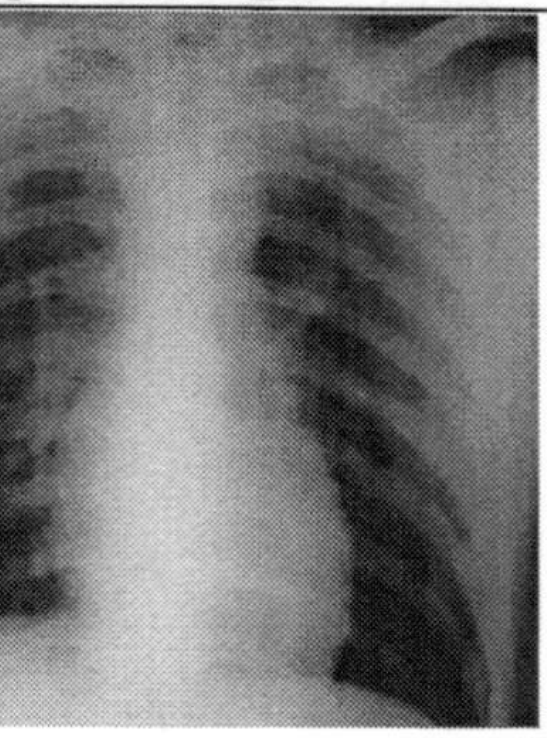
ตัวอย่างการทำแทรกเชิงเส้นตรง (Linear Interpolation) ด้วยเมตริกซ์ ขนาด 2x2



4.2 <u>การปรับค่าความสว่างของภาพ</u> (Brightness Adjust)

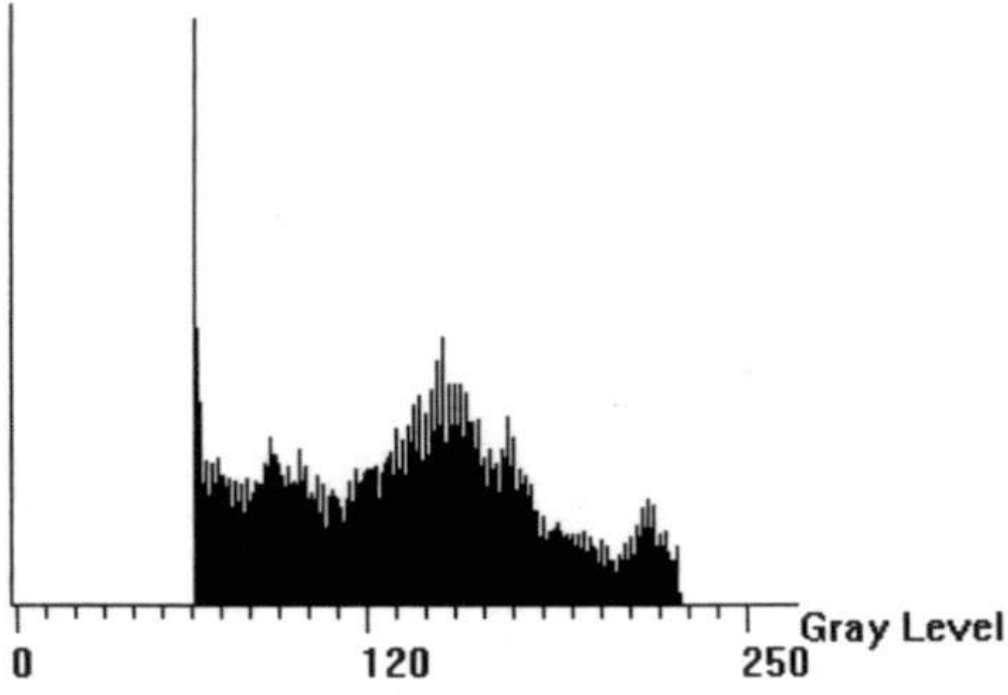
Image Processing

ge Chest Tool



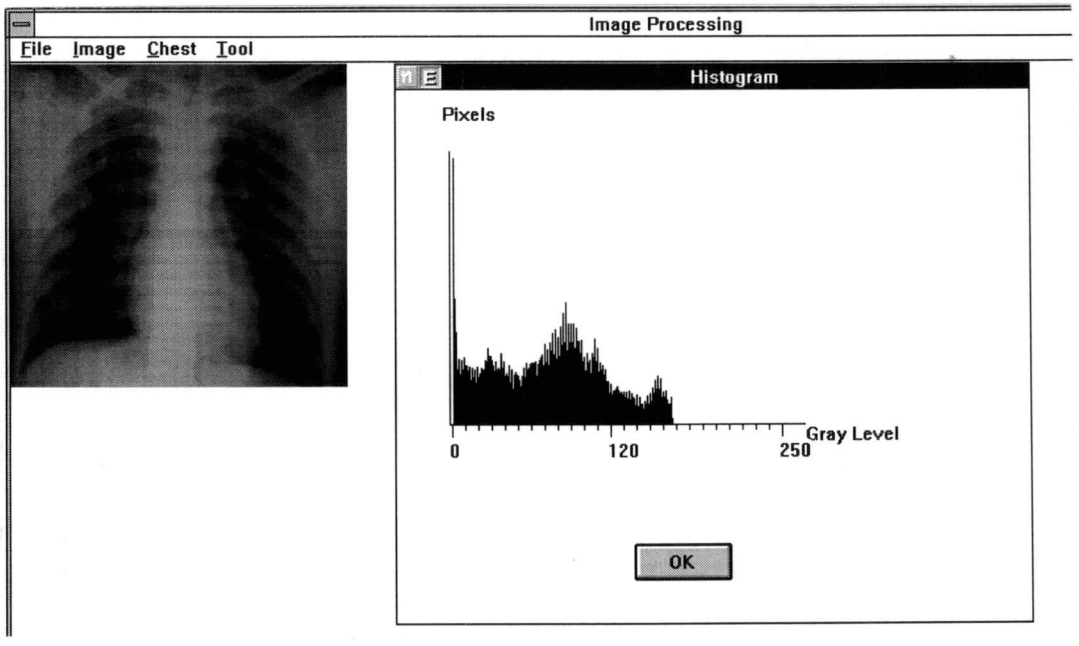
Histogram

Pixels



OK

การปรับปรุงความสว่างของภาพสามารถที่จะกระทำได้จากซอฟท์แวร์ โดยการบวกหรือลบด้วยค่าคงที่เข้าไปที่ภาพทั้งภาพเพื่อที่จะเพิ่มหรือลดความสว่าง ตามลำดับ (คือการขยับฮิสโตแกรมของภาพทั้งภาพไปทางบวกหรือลบ) ภาพที่ 4.1 แสดงการปรับความสว่างก่อนและหลัง



(a)

(b)

รูปที่ 4.1 (a) ก่อนการปรับความสว่าง (b)หลังการปรับความสว่าง

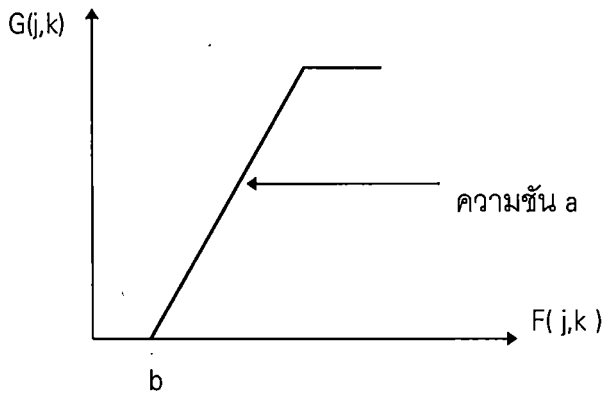
4.3 การปรับปรุงคอนทราสของภาพ

ในบางครั้งภาพที่ทำการพิจารณานั้นมีคอนทราสของภาพต่ำคือภาพมีระดับเทาของจุดภาพใกล้เคียงกันตลอดทั้งภาพ ถ้ามองจากฮิสโตแกรมของภาพ จะพบว่าการกระจายของระดับเทาของจุดภาพจะรวมกันเป็นช่วงแคบๆทำให้ไม่สามารถมองเห็นรายละเอียดเล็กๆของภาพในการปรับปรุงคอนทราสของภาพ โดยการเปลี่ยนลักษณะการกระจายฮิสโตแกรมของภาพใหม่ ให้ฮิสโตแกรมของภาพกว้างขึ้น และมีความต่างของค่าระดับสีเทาสูงขึ้นซึ่งจะทำให้สามารถมองเห็นรายละเอียดที่อยู่ในภาพเดิม ที่ไม่สามารถมองเห็นได้ ชัดเจนยิ่งขึ้นการปรับปรุงคอนทราสนั้นสามารถใช้ฟังก์ชันแบบเชิงเส้น และฟังก์ชันไม่เชิงเส้น อาจเป็นฟังก์ชันเส้นตรง, ฟังก์ชันกำลังสองเป็นต้น เพื่อที่จะมาทำการกระจายฮิสโตรแกรมเสียใหม่

- การปรับปรุงคอนทราสโดยใช้ฟังก์ชันเส้นตรง

จากสมการที่ 4.3.1 ในเทอมสมการเส้นตรง y=aF(j,k)+b นั้นจะมี aเป็นพารามิเตอร์เกณฑ์ (gain) ที่มีผลต่อความชัน (slope)ในการ mapping และมี b เป็นพารามิเตอร์ไบอัส (bias) เป็นค่าคงที่ที่บวกเข้าไปทุก ๆ จุดภาพ เมื่อนำมาใช้ในการปรับปรุงคอนทราสของภาพต้องคำนึงถึงค่าที่ต่ำกว่าจุดตัดแกนเดิมที่ต้องกำหนดให้เป็นศูนย์ และค่าสูงที่เลยออกนอกย่านที่ต้องกำหนดให้เป็น 255 ดังแสดงในรูปที่ 4.3.1

$$G(j,k) = \begin{cases} 0 & \text{ถ้า } 0<F(j,k)<\text{lower} \\ aF(j,k)+b & \text{ถ้า } \text{lower}<F(j,k)<\text{upper} \\ 255 & \text{ถ้า } \text{upper}<F(j,k)<255 \end{cases} \qquad (4.3.1)$$



รูปที่ 4.3.1 แสดงฟังก์ชั่นในการดึงอย่างเชิงเส้นอย่างง่าย

จะเห็นว่าการกำหนดตัวแปรเกณฑ์ และไบอัสเพื่อให้ได้ฟังก์ชันเส้นตรงในการแปลงที่ต้องการนั้นกำหนดในลักษณะที่สื่อความหมายให้เข้าใจได้ยาก ดังนั้นในรูปที่กำหนดได้ง่ายเป็นรูปแบบที่นิยมใช้มากที่สุด ดังนี้

$$G(j,k) = \begin{cases} y_{min} & \text{ถ้า} \quad 0 \le F(j,k) \le x_{min} \\ F'(j,k) & \text{ถ้า} \quad x_{min} < F(j,k) < x_{max} \\ y_{max} & \text{ถ้า} \quad x_{max} \le F(j,k) \le 255 \end{cases} \qquad (4.3.2)$$

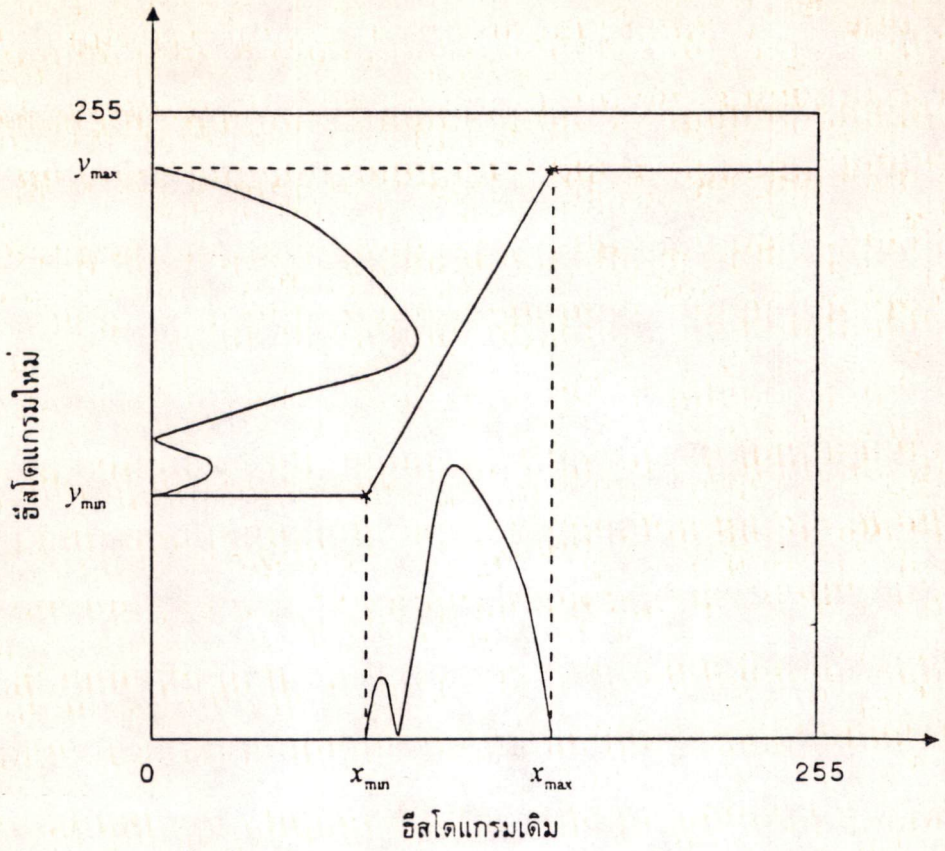
เมื่อมีการเพิ่มการกำหนดสมการเส้นตรงในรูปแบบที่ง่ายขึ้น ดังนี้

$$F'(j,k) = \frac{[\,(F(j,k) - x_{min})(y_{max} - y_{min})\,]}{(x_{max} - x_{min})} + y_{min} \qquad (4.3.3)$$

ในขณะที่วิธีการปกติในสมการที่(4.3.1)ไม่สามารถทำได้นั้นสมการที่(4.3.2) และ (4.3.3) มีข้อได้เปรียบอีกอย่างหนึ่งคือสามารถแปลงการกระจายไปสู่ทุก ๆ ตำแหน่งบนฮีสโตแกรมใหม่ได้ทั้งหมดดังแสดงในรูปที่ 4.3.2 โดยสามารถกำหนดย่านบนแกนใหม่ได้ตามต้องการ

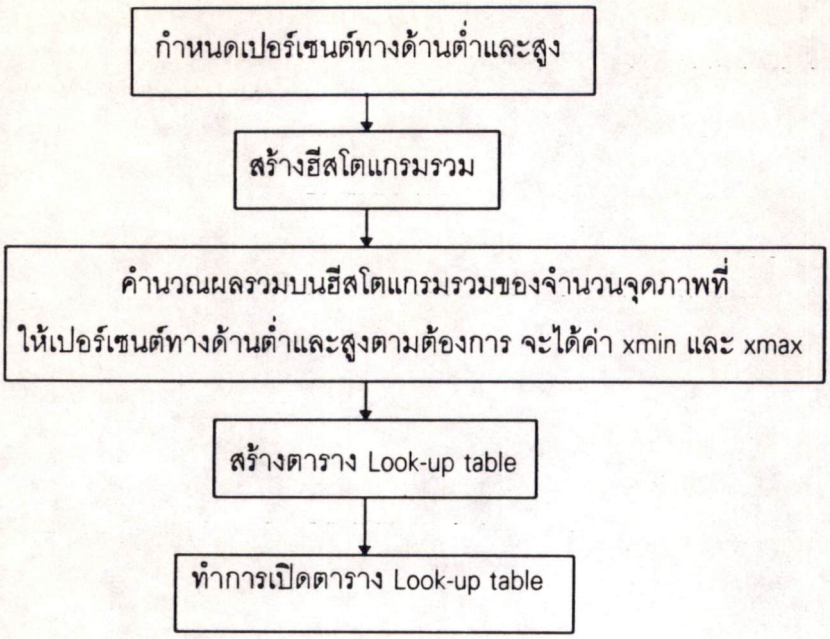
ในการประมวลผลสามารถนำวิธีการเปิดตารางดังที่ได้กล่าวมาแล้วในหัวข้อก่อนเพื่อช่วยเพิ่มความเร็วในการคำนวณได้ และนอกจากการให้ค่าสูงสุดและต่ำสุดโดยตรงแล้วยังสามารถใช้การคำนวณหาค่าสูงสุดและต่ำสุดของค่าระดับสีเทาแล้วทำการดึงอย่างอัตโนมัติ(automatic linear contrast stretching) ได้เช่นกันจึงมีประโยชน์อย่างมากในการนำไปใช้ดึงค่าที่เลยออกนอกย่านให้กลับเข้ามาอยู่ในย่าน 0 - 255

นอกจากนี้ในทางปฏิบัติจะพบภาพที่มีค่าระดับสีเทาเด่นออกมาจากกลุ่มค่าระดับสีเทาส่วนใหญ่ของภาพ ซึ่งค่าระดับสีเทานี้เมื่อเทียบกันแล้วจะเป็นเพียงจำนวนน้อย ๆเท่านั้น ดังนั้นในการดึงอย่างอัตโนมัติจึงไม่อาจให้ค่าคอนทราสที่ดีออกมาได้ตัวอย่างเช่นเมื่อมีสัญญาณรบกวนที่มีค่าระดับสีเทาสูงอยู่ 1 จุดภาพที่เป็นค่าระดับสีเทาสูงสุดในภาพเมื่อทำการดึงอย่างอัตโนมัติค่าระดับสีเทาของสัญญาณรบกวนนี้จะเป็นค่าx_{max} ทำให้เกิดเกณฑ์ของการดึงกลุ่มข้อมูลหลักผิดไปโดยได้คอนทราสที่น้อยกว่าที่ควรจะเป็น รูปที่ 4.3.4 แสดงฮีสโตแกรมของภาพผลลัพธ์ที่ได้จากการดึงอย่างอัตโนมัติด้วยค่าสูงสุดและต่ำสุดที่หาได้

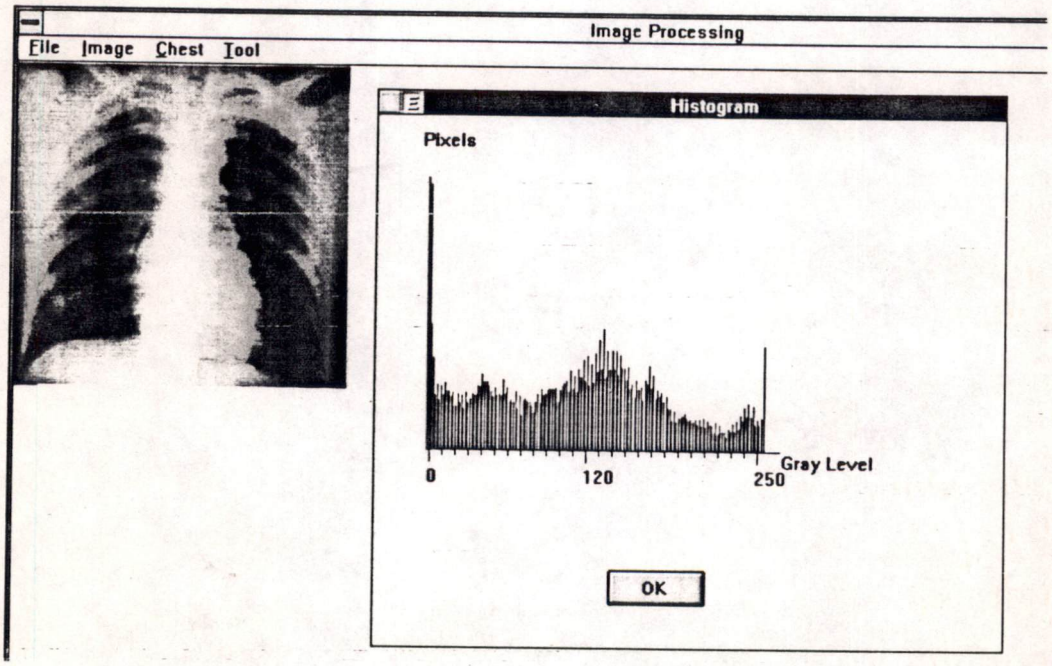


รูปที่ 4.3.2 แสดงการแปลงฮีสโตแกรมเดิมไปยังฮิสโตแกรมใหม่

ในกรณีที่มีค่าแตกกลุ่มเป็นค่าเล็กๆจะทำให้ไม่สามารถให้ค่าคอนทราสที่ดีออกมาได้ดังนั้นในทางปฏิบัติวิธีการแก้ปัญหาทั้งหมดนี้ให้หมดไป ทำได้โดยการนำเปอร์เซนต์ของจำนวนจุดภาพมาช่วยแก้ข้อบกพร่องของจุดภาพจำนวนน้อย ๆ เหล่านี้ทั้งทางด้านต่ำและทางด้านสูงของฮีสโตแกรม ภาพผลลัพธ์ที่ได้จะมีคอนทราสของกลุ่มข้อมูลส่วนใหญ่สูงตามที่ต้องการสำหรับรูปที่ 4.3.2 แสดงฮีสโตแกรมที่ทำการดึงอย่างอัตโนมัติ ด้วยค่าสูงสุดและต่ำสุด ที่นำค่าเปอร์เซนต์ของจำนวนจุดภาพทั้งทางด้านต่ำและทางด้านสูงมาใช้โดยข้อมูลทางด้านต่ำและทางด้านสูงใช้จำนวนจุดเท่ากับ 1 เปอร์เซนต์ของจำนวนจุดภาพทั้งหมด



รูปที่ 4.3.3 โฟลว์ชาร์ตลำดับขั้นตอนการดึงเชิงเส้นอย่างอัตโนมัติด้วยค่าสูงสุดและต่ำสุดที่หาได้จากการกำหนดเปอร์เซนต์ทางด้านต่ำและสูง



รูปที่ 4.3.4 แสดงภาพและฮิสโตแกรมที่ทำการดึงอย่างอัตโนมัติ

4.4 <u>การปรับปรุงฮิสโตแกรม</u> (Histogram Modification Technique)

ฮิสโตแกรมของภาพสามารถที่จะแสดงถึงลักษณะโดยรวมของภาพได้วิธีการนี้จะเป็นการปรับปรุงภาพโดยทำการเปลี่ยนแปลงหรือแก้ไขฮิสโตแกรมภาพให้เป็นไปตามฮิสโตแกรมที่กำหนดให้ เมื่อเราให้ตัวแปร r แทนระดับความเข้มของจุดในภาพที่จะต้องการปรับปรุง เพื่อให้ง่ายขึ้นเราจะแสดงค่า r ให้อยู่ในช่วง

$$0 \leq r \leq 1 \qquad (4.4.1)$$

โดยที่ r=0 แทนสีดำ และ r=1 แทนสีขาวในระดับเทา

สำหรับ r ใด ๆ ที่มีค่าอยู่ในช่วง [0,1] และเราจะสนใจรูปแบบการแปลงอยู่ในรูป

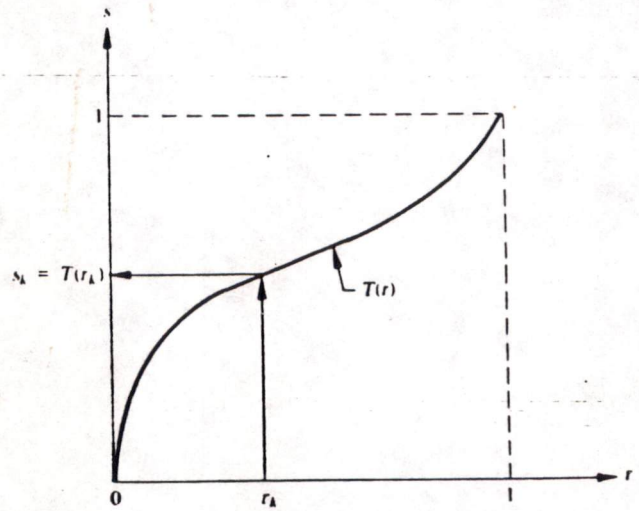
$$S = T(r) \qquad (4.4.2)$$

ที่ค่าของระดับเทาในทุก ๆ S เกิดจากค่า r ในภาพเดิมซึ่งตั้งสมมุติฐานว่าฟังก์ชันในการแปลงเป็นดังสมการที่ (4.4.2) และจะต้องอยู่ในเงื่อนไข

(1) T(r) เป็นฟังก์ชันตัวแปลงเดียว (single-value)และเป็นฟังก์ชันเพิ่มจากจุดเริ่มต้นไปจุดปลายสุดทางเดียว (monotonically increasing)ในช่วง $0 \leq r \leq 1$

(2) $0 \leq T(r) \leq 1$ สำหรับ $0 \leq r \leq 1$

เงื่อนไขในข้อที่ 1 เพื่อที่จะป้องกันและรักษาลำดับของระดับเทาจากขาวไปดำในขณะที่เงื่อนไขที่ 2 เพื่อเป็นข้อยืนยันการแปลงค่า ค่าของระดับเทาของจุดจะอยู่ในช่วงที่กำหนด สำหรับตัวอย่างฟังก์ชันแปลงที่เป็นไปตามเงื่อนไขทั้ง 2 แสดงในรูปที่ 4.4.1

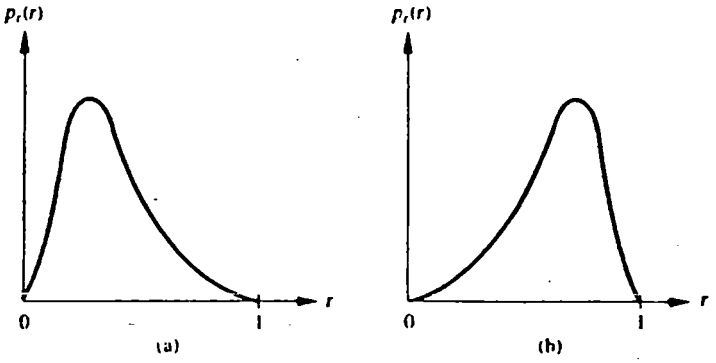


รูปที่ 4.4.1 รูปแบบฟังก์ชันแปลงของค่าระดับเทา

ในการทำฟังก์ชันแปลงอินเวอร์ส (inverse transform) จาก s กลับไป r แสดงดังข้างล่างนี้

$$r = T^{-1}(s) \qquad 0 \leq s \leq 1 \qquad (4.4.3)$$

โดยที่ $T^{-1}(s)$ ค่าของตัวแปร s จะต้องอยู่ภายใต้เงื่อนไขของสองข้อที่กล่าวข้างบนระดับเทาในภาพจะอยู่กระจายเป็นแบบสุ่ม (random) ในช่วง [0,1] สมมุติให้ตัวแปรนี้เป็นตัวแปรแบบต่อเนื่อง รูปภาพเดิมและฟังก์ชันแปลงของระดับเทาจะขึ้นอยู่กับฟังก์ชันความหนาแน่นของความของความน่าจะเป็น (probability density function) $P_r(r)$ และ $P_s(s)$ ตามลำดับ ซึ่งค่าฟังก์ชันความหนาแน่น (density function) ของระดับสามารถจะบอกลักษณะทั่วไปของภาพได้ ตัวอย่างดังในภาพ4.4.2(a) จากฟังก์ชันความหนาแน่นในภาพ ค่าของระดับเทาส่วนใหญ่จะอยู่ในช่วงเทาดำของระดับเทา แสดงว่าภาพนี้มีลักษณะค่อนข้างดำ หรือในภาพ 4.4.2(b) ลักษณะของภาพจะออกมาในโทนค่อนข้างสว่างหรือ สีขาว เพราะระดับเทาส่วนใหญ่อยู่ในโทนสีขาว



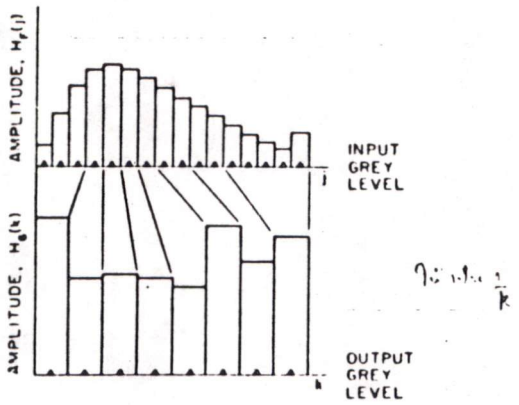
รูปที่ 4.4.2 แสดงฟังก์ชันความหนาแน่นของระดับเทา (a)ภาพที่มืด (b)ภาพที่สว่าง

วิธีการปรับปรุงนี้จะเป็นการปรับปรุงค่าของฟังก์ชันของความน่าจะเป็นของระดับเทาโดยใช้ฟังก์ชันแปลง เป็นตัวกำหนดค่า

4.4.1 <u>การปรับปรุงภาพโดยวิธีฮิสโตแกรมอิควอไรเซชั่น</u> (Histogram Equalization)

จากรูปที่ 4.4.3 แสดงตัวอย่างของฮิสโตแกรมอิควอไรเซชั่น จากรูป $H_F(j)$ สำหรับ j=1,2,...,J แสดงสัดส่วนระหว่างขนาดของฮิสโตแกรมที่ระดับที่ j กับจำนวนจุดทั้งหมดของภาพวิธีการฮิสโตแกรมอิควอไรเซชั่น จะสร้างภาพผลลัพธ์ที่มีกระจายของระดับเทา G ให้มีขนาดเท่ากันทุก ๆ ระดับ $H_G(k)=1/k$ สำหรับ k=1,2,...,K จากตัวอย่างในรูป 4.4.3 จำนวนระดับเทาของฮิสโตแกรม

ผลลัพธ์จะกำหนดเพียงครึ่งหนึ่งของจำนวนระดับของฮิสโตแกรมเริ่มต้น สำหรับวิธีการแปลงมีดังนี้ เริ่มจากระดับเทาต่ำสุดของฮิสโตแกรมต้นแบบ จะทำการรวมฮิสโตแกรมแต่ละระดับเทาให้มีค่าเข้าใกล้ค่าเฉลี่ย แล้วสร้างระดับเทาแรกในภาพใหม่เป็นระดับเทาเริ่มต้น วิธีการนี้จะทำต่อไปที่ค่าระดับเทาที่สูงขึ้นจนกระทั่งจบ ถ้าจำนวนของระดับเทาของภาพเริ่มต้นมีจำนวนมาก ค่าระดับเทาของภาพที่ปรับปรุง มักจะมีค่าระดับเทาน้อยกว่าภาพต้นแบบ ในการทำฮิสโตแกรมอิควอไรเซชั่นสามารถกำหนดจำนวนระดับเทาของภาพต้นแบบให้เท่ากับภาพที่ปรับปรุงซึ่งในการปรับปรุงภาพจะจัดการกระจายของฮิสโตแกรมในภาพใหม่



รูปที่ 4.4.3 ตัวอย่างการใช้วิธีฮิสโตแกรมอิควอไรเซชั่น ด้วยระดับเทาที่ไม่เท่ากัน โ ดยประมาณ

การปรับปรุงฮิสโตแกรมสามารถพิจารณาในรูปฟังก์ชั่นแปลง $g_k = T\{ f_j \}$ ขนาดของตัวแปร f_j เป็นดังนี้ $f_1 \leq f_j \leq f_J$ ที่แปลงไปยังตัวแปรผลลัพธ์ $g_1 \leq g_k \leq g_K$ โดยที่การแจกแจงความน่าจะเป็นของต้นแบบ(input probability)$P_R\{ f_j=a_j \}$ เป็นตัวกำหนดการแจกแจงความน่าจะเป็นผลลัพธ์$P_R\{g_k=b_k\}$ ที่ a_j และ b_k เป็นค่าของความน่าจะเป็นที่ระดับ j และ k ดังนั้นผลรวมการกระจายของความน่าจะเป็น(probility distribution)ของภาพต้นแบบและผลลัพธ์เป็นหนึ่ง

$$\sum_{j=1}^{J} P_R\{ f_j = a_j \} = 1 \qquad (4.4.1a)$$

$$\sum_{k=1}^{K} P_R\{ g_k = b_k \} = 1 \qquad (4.4.1b)$$

นอกจากนั้นค่าของการแจกแจงสะสม (cumulative distribution) จะต้องมีค่าเท่ากัน ที่ $b_k=T(a_j)$

$$\sum_{n=1}^{k} P_R\{ g_n = b_n\} = \sum_{m=1}^{j} P_R\{ f_m = a_m\} \qquad (4.4.2)$$

ผลรวมของการแจกแจงความน่าจะเป็นสะสมทางด้านขวามือของสมการ (4.4.2) (ภาพต้นแบบ) สามารถแทนด้วยค่าฮิสโตแกรมสะสม (Cummulative Histogram)

$$\sum_{n=1}^{k} P_R\{ g_n = b_n\} = \sum_{m=1}^{j} H_F(m) \qquad (4.4.3)$$

จากสมการที่ 4.4.3 จะต้องเปลี่ยนให้อยู่ในรูปของ g_k ที่หาได้จาก f_j ผลลัพธ์ของการทำคือสร้างตารางสำหรับแต่ละระดับเทาผลลัพธ์จากระดับเทาต้นแบบ ฟังก์ชั่นการแปลงฮิสโตแกรมหาได้จาก การประมาณค่าการกระจายความน่าจะเป็นแบบไม่ต่อเนื่องจากการกระจายความน่าจะเป็นแบบต่อเนื่อง (continuous probabilty density)ผลลัพธ์ของการประมาณคือ

$$\int_{g\_min}^{g} p_g(g)\, dg = \int_{f\_min}^{f} p_f(f)\, df \qquad (4.4.4)$$

ที่ $p_f(f)$ และ $p_g(g)$ เป็นความหนาแน่นความน่าจะเป็น f และ g ตามลำดับตัวแปรอินทิกริท (integral) ทางด้านขวามือของสมการ (4.4.4) คือ ฟังก์ชั่นการกระจายความน่าจะเป็นสะสม ของภาพต้นแบบของตัวแปร f ดังนั้น

$$\int_{g\_min}^{g} p_g(g)\, dg = P_f(f) \qquad (4.4.5)$$

โดยที่ $P_f(f) = \int_{f\_min}^{f} p_f(f)\, df$

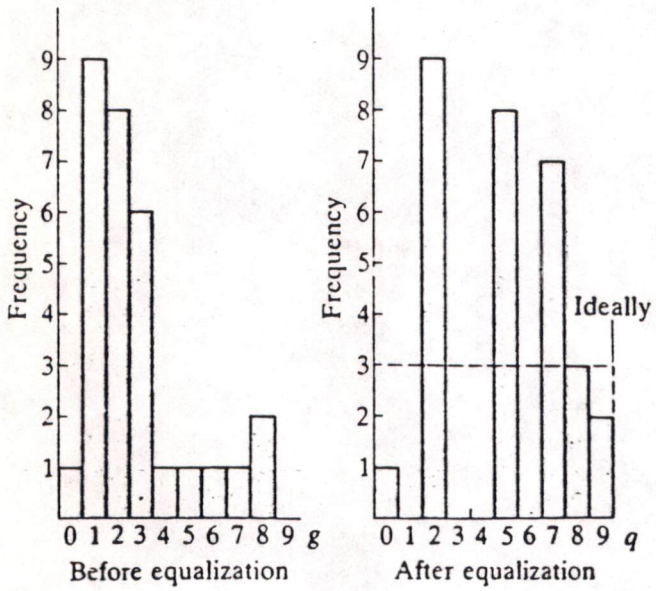
เนื่องจากต้องการผลลัพธ์ของฮิสโตรแกรมที่มีการแจกแจงเท่ากัน ดังนั้นจะได้

$$P_g(g) = 1 / (g_{max} - g_{min}) \qquad (4.4.6)$$

สำหรับ $g_{min}<g<g_{max}$ฟังก์ชั่นแปลงของฮิสโตแกรมอิควอไรเซชั่นจะได้ดังสมการข้างล่าง

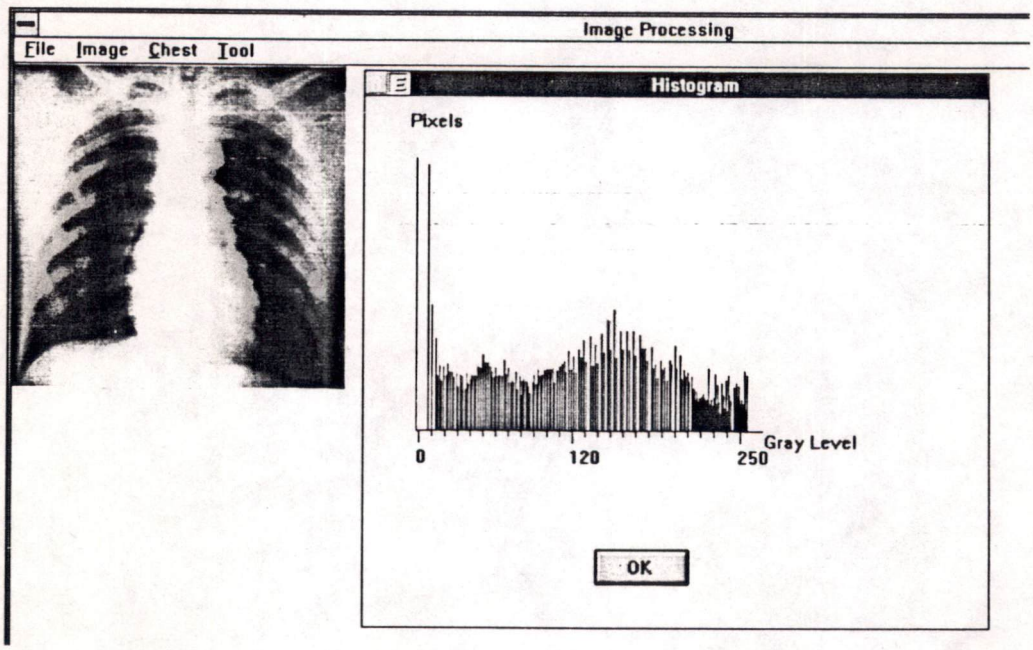
$$g = [g_{max} - g_{min}] P_f(f) + g_{min} \qquad (4.4.7)$$

ในตารางที่ 4.4.1 แสดงรูปแบบของฮิสโตแกรมผลลัพธ์ที่ต้องการและฟังก์ชั่นแปลงที่นอกเหนือจากการแจกแจงแบบยูนิฟอร์ม(Uniform Distribution)



g	Frequency	Cumulative frequency F(g)
0	1	1	0
1	9	10	2
2	8	18	5
3	6	24	7
4	1	25	7
5	1	26	8
6	1	27	8
7	1	28	8
8	2	30	9
9	0	30	9

รูปที่ 4.4.4 จะแสดงตัวอย่างของการแปลงฮิสโตแกรม



รูปที่ 4.4.5 แสดงการแปลงโดยฮิสโตแกรมอิควอไรเซชั่น

วิธีการฮิสโตแกรมอิควอไรเซชั่น สามารถที่จะได้ผลดีกับรายละเอียดของภาพใน ส่วนของภาพที่มีระดับเทาค่อนข้างดำในกรณีที่คุณภาพของภาพเริ่มต้นดีอยู่แล้วบ่อยครั้งที่วิธี นี้ทำให้คุณภาพด้อยลง

TABLE 10.2-1. Histogram modification transfer functions

Output Probability Density Model		Transfer Function*
Uniform	$p_g(g) = \frac{1}{g_{max} - g_{min}} \quad g_{min} \le g \le g_{max}$	$g = [g_{max} - g_{min}]P_f(f) + g_{min}$
Exponential	$p_g(g) = \alpha \exp\{-\alpha(g - g_{min})\} \quad g \ge g_{min}$	$g = g_{min} - \frac{1}{\alpha}\ln[1 - P_f(f)]$
Rayleigh	$p_g(g) = \frac{g - g_{min}}{\alpha^2}\exp\left\{-\frac{(g - g_{min})^2}{2\alpha^2}\right\} \quad g \ge g_{min}$	$g = g_{min} + \left[2\alpha^2 \ln\left(\frac{1}{1 - P_f(f)}\right)\right]^{1/2}$
Hyperbolic (Cube Root)	$p_g(g) = \frac{1}{3}\frac{g^{-2/3}}{g_{max}^{1/3} - g_{min}^{1/3}}$	$g = \{[g_{max}^{1/3} - g_{min}^{1/3}][P_f(f)] - g_{min}^{1/3}\}^3$
Hyperbolic (Logarithmic)	$p_g(g) = \frac{1}{g[\ln g_{max} - \ln g_{min}]}$	$g = g_{min}\left[\frac{g_{max}}{g_{min}}\right]^{P_f(f)}$

*The cumulative probability distribution $P_f(f)$ of the input image is approximated by its cumulative histogram

$$P_f(f) \approx \sum_{m=0}^{f} H_f(m)$$

ตารางที่ 4.4.1 แสดงฟังก์ชั่นการแปลงฮิสโตแกรมต่างๆ

4.4.2 <u>การปรับปรุงโดยวิธีฮิสโตแกรมอิควอไรเซชั่นโดยกำหนดบริเวณ</u>

(Regionally Histogram Equalization)

วิธีการนี้เป็นการประยุกต์วิธีการทำฮิสโตแกรมให้เท่ากันโดยแทนที่จะทำการคำนวณ โดยนำทั้งภาพมาสร้างฟังก์ชั่นแปลงแล้วทำการปรับปรุง เราเพียงแต่เป็นผู้กำหนดบริเวณที่น่าสนใจในการปรับปรุงภาพ และสร้างฟังก์ชั่นแปลง และทำการปรับปรุงที่ส่วนนั้น ข้อดีคือเมื่อต้องการจะ

ปรับปรุงดูส่วนของภาพที่ให้รายละเอียดไม่ชัดสามารถที่จะทำการปรับปรุงค่าระดับเทาเฉพาะส่วนนั้นจึงไม่ทำให้ส่วนอื่นของภาพที่มีคุณภาพดีอยู่แล้วกระทบกระเทือน

4.4.3 การปรับปรุงฮิสโตแกรมแบบปรับได้ (Adaptive Histogram Modification)

การปรับปรุงฮิสโตแกรมทั่วไปจะกระทำกับภาพทั้งหมดในการสร้างฟังก์ชั่นแปลง วิธีการนี้จะใช้การสร้างหน้าต่างและใช้ฮิสโตแกรมของแต่ละหน้าต่าง ในการสร้างฟังก์ชั่นแปลงในการปรับปรุงภาพ Pizer etal(9) เสนอวิธีการปรับปรุงภาพโดยฮิสโตแกรมอิควอไรเซชั่นแบบปรับได้ โดยแบ่งภาพออกเป็นส่วน ๆ เป็นตารางและใช้จุดตัดของตาราง (grid point) เป็นจุดศูนย์กลางในการสร้างหน้าต่างขึ้นมาแล้วหาค่าฮิสโตแกรมของแต่ละหน้าต่างการสร้างฟังก์ชั่นแปลงของจุดจะพิจารณาจากฮิสโตแกรมของหน้าต่าง ๆ สี่อันรอบ ๆ จุดนั้นๆแสดงดังรูป 4.4.6

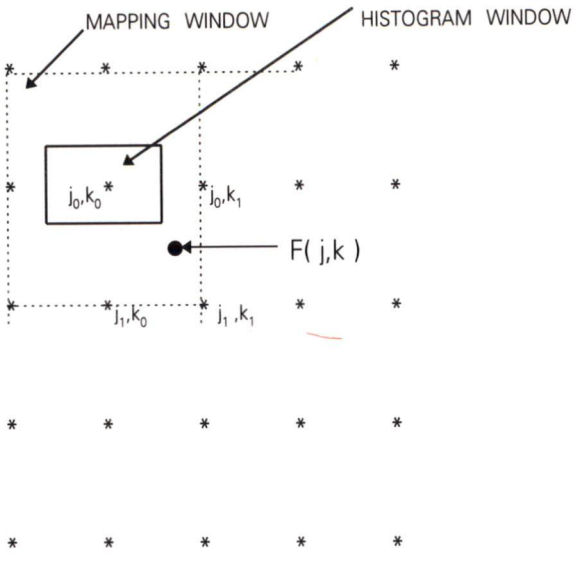
ขนาดของหน้าต่างที่ใช้สร้างฮิสโตแกรมสามารถที่ใหญ่กว่าหรือเล็กกว่าขนาดจุดตัด(grid space)ที่ใช้แบ่งภาพออกเป็นส่วนให้ M_{00},M_{01},M_{10},M_{11} เป็นฮิสโตแกรมในแต่ละหน้าต่าง ๆ ที่สร้างขึ้นรอบ ๆ จุด F(j,k) ในการสร้างฮิสโตแกรมฟังก์ชั่นแปลงของจุด F(i,k) จะใช้การทำอินเทอร์โพเรชั่นเชิงเส้น (bilinear Interpolation)กับหน้าต่างทั้งสี่ที่อยู่รอบ ๆ สามารถเขียนเป็นสมการดังนี้

$$M = a[bM_{00} + (1 - b)M_{10}] + (1 - a)[bM_{01} + (1 - b))M_{11}\} \qquad (4.4.3\text{-}1)$$

ที่

$$a = (k - k_0)/ (k_1 - k_0)$$

$$b = (j - j_0)/ (j_1 - j_0)$$



รูปที่ 4.4.6 แสดงการทำแปลงฮิสโตแกรมแบบปรับได้;
* = จุดตัดของตารางที่แบ่งภาพ (grid point) , ● = จุดที่จะคำนวณ

4.5 การทำภาพให้ราบเรียบ (Image Smoothing)

การทำภาพให้ราบเรียบนั้นเป็นการช่วยขจัดหรือลดสัญญาณรบกวน (noise)ที่สามารถเกิดขึ้นได้ในภาพดิจิตอล ซึ่งอาจเกิดมาจากระบบถ่ายภาพไม่ดีพอหรืออาจเกิดจาก การสื่อสารข้อมูล การทำภาพให้ราบเรียบนั้นสามารถกระทำได้ทั้งสปาเซี่ยนโดเมนและโดเมนความถี่ ในที่นี้จะกล่าวเฉพาะส่วนที่เป็นสปาเซียนโดเมนโดยใช้การทำคอนวูโลชั่น

4.5.1 การเฉลี่ยรอบจุด (Neigborhood Averaging (Mean))

ให้ภาพขนาด NXN ของภาพ f(x,y) จะทำให้กลมกลืน g(x,y) จากการเฉลี่ยค่าระดับความเข้มที่ทุกๆจุด (x,y) จากบริเวณโดยรอบของจุด (x,y) ซึ่งเราจะเขียนได้ดังนี้

$$g(x,y) = 1/M \sum_{(n,m)\in s} f(n,m) \qquad (4.5.1)$$

โดย x,y = 0,1,..,N-1 s เป็นเซ็ทของคู่ลำดับที่อยู่ในบริเวณของจุด (x,y) ซึ่งรวม(x,y) ด้วยและ M คือจำนวนจุดที่อยู่ในบริเวณนั้น ซึ่งถ้าเป็นบริเวณ 3X3 เราก็สามารถเขียนอยู่ในรูปฟิลเตอร์เวท (filter weights) เป็นดังนี้

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

square window

หรือในบางกรณีกำหนดให้ M อยู่ในรูปร่างเลขบวกเราก็จะได้มาส์กดังนี้

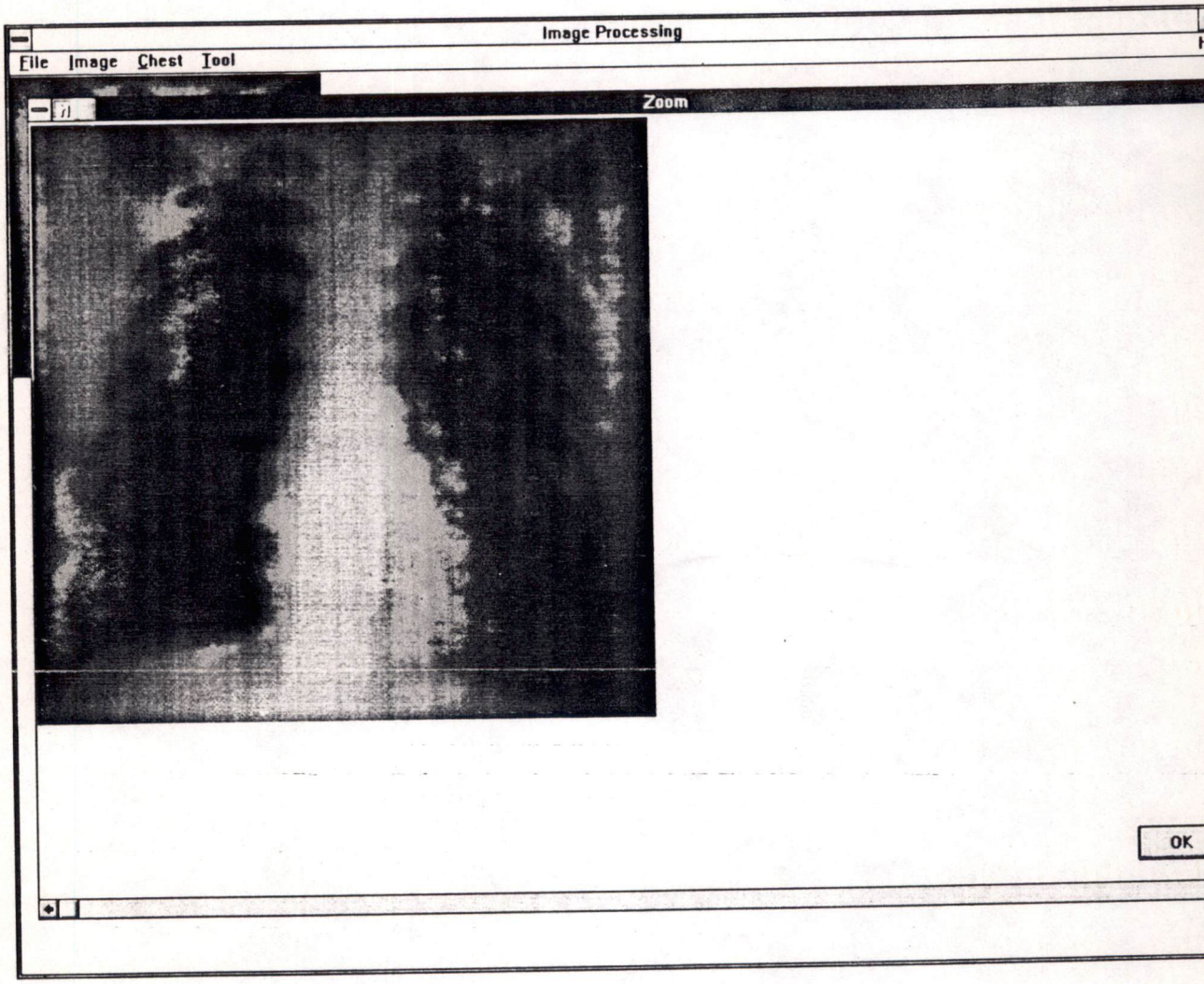
	1/5	
1/5	1/5	1/5
	1/5	

plus shaped window

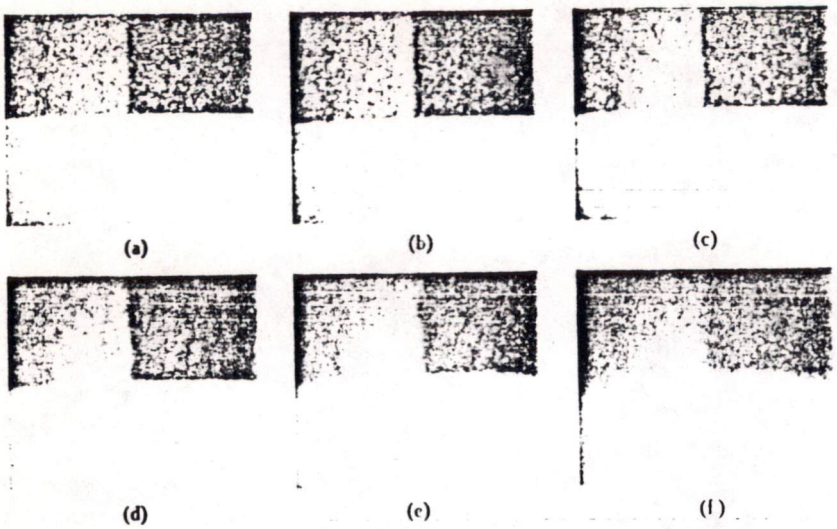
จะเห็นได้ว่าวิธีนี้ จะมีผลข้างเคียงคือทำให้ภาพมัว วิธีการแก้คือ การกำหนดค่า T (threshold)ซึ่งจะทำโดยใช้สมการดังต่อไปนี้

$$g(x,y) = \begin{cases} 1/M \sum f(n,m) & \text{เมื่อ } |f(x,y) - 1/M \sum f(m,n)| < T \\ & (n,m) \in s \\ f(x,y) & \text{ค่าอื่นๆ} \end{cases} \qquad (4.5.2)$$

โดยค่า T ที่ไม่เป็นลบ



รูปที่ 4.5.2 แสดงภาพที่ทำให้เรียบด้วยวิธีหาค่าเฉลี่ย



รูปที่ 4.5.3 แสดงผลของการใช้วิธีการเฉลี่ยรอบจุด (a) แสดงภาพที่มีสัญญาณรบกวนและจะถูกปรับปรุง (b) ถึง (f) แสดงภาพที่ปรับปรุงแล้วโดยการใช้ขนาดของบริเวณ nXn ด้วย n = 3,5,9,15 และ 31

4.5.2 <u>ค่าเฉลี่ยถ่วงน้ำหนัก</u>(Weighted Mean)

ค่าเฉลี่ยถ่วงน้ำหนัก ใช้บ่อยในกรณีที่ให้ค่าน้ำหนักของแต่ละจุด จากจุดศูนย์กลางไม่เท่ากัน สำหรับขนาด 3X3 ค่าที่ใช้อาจเป็นดังนี้

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

square window

การมาส์กแบบนี้มีชื่อว่า Gussian low pass filter ซึ่งมันจะทำหน้าที่เป็นตัวกรองความถี่ต่ำ สำหรับการมาส์กในรูปเลขบวก (plus shaped windows) ก็จะแสดงดังข้างล่างนี้

	1/6	
1/6	1/3	1/6
	1/6	

plus shaped windows

วิธีการนี้จะได้ผลดีกว่าวิธีแรก คือภาพจะไม่มัวมากและเวลาที่ใช้ก็ใกล้เคียงกับวิธีแรก

4.5.3 <u>มีเดียนและโหมดฟิลเตอร์</u> (Median and Mode Filter)

วิธีการในหัวข้อที่แล้วจะเห็นได้ว่าเกิดผลข้างเคียงคือ ภาพที่ใช้วิธีนั้นจะมีลักษณะมัวถึงแม้ว่าจะมีการใช้ค่าเทรสโซลด์ (threshold) เพราะในการคำนวณหาค่าที่เหมาะสมค่อนข้างยุ่งยาก วิธีการนี้จะแทนค่าจุดด้วยค่ามีเดียน (Median) ของจุดรอบ ๆ วิธีนี้มีประสิทธิภาพเมื่อภาพมีสัญญาณรบกวนที่มีระดับความเข้มมากๆ เป็นจุดๆ (Spikelike)

ข้อเสียของวิธีนี้คือ มีความล่าช้า อีกวิธีหนึ่งที่คล้าย ๆ กัน คือ การแทนค่าด้วยค่าโหมด (Mode)

รูปที่ 4.5.4 (a) แสดงภาพจริง (b) ภาพที่ถูกสัญญาณรบกวน (c) ผลของการใช้ 5 X5 เฉลี่ยรอบจุด (d) ผลของการใช้ 5X5 มีเดียนฟิลเตอร์

4.6 <u>การปรับปรุงภาพให้คมชัด</u> (Image Sharpening)

เป็นวิธีการที่ต้องการเพิ่มความคมชัดของขอบภาพ เพื่อที่จะสามารถเห็นรายละเอียดของภาพได้ดีขึ้น วิธีที่นิยมใช้กันมากคือ วิธีการทำอันชาร์ปมาส์กิ้ง จะพยายามเน้นในส่วนที่เป็นขอบภาพโดยใช้พารามิเตอร์ในการเน้นขอบภาพ สามารถแบ่งเป็น 2 วิธีคือ การทำอันชาร์ปมาส์กแบบปรับไม่ได้ และการทำอันชาร์ปมาส์กิ้งแบบปรับได้

4.6.1 <u>การทำอันชาร์ปมาส์กิ้งแบบปรับไม่ได้</u>(Non-Adaptive Unsharpmasking)

การทำอันชาร์ปมาส์กิ้งคำนวณได้จากการกรองภาพความถี่ต่ำ (low-pass filtering) จากภาพต้นแบบ แล้วนำค่านี้ไปลบจากภาพต้นฉบับเดิม ภาพที่ได้จะเป็นภาพที่มีความถี่สูง (high frequency image) ซึ่งภาพความถี่สูงนี้ก็จะเป็นบริเวณของขอบภาพหรือรายละเอียดของภาพหลังจากได้ภาพความถี่สูงแล้ว จะทำการเพิ่มหรือเน้นภาพนี้ด้วยการคูณพารามิเตอร์เข้าไปแล้วนำไปบวกกับภาพต้นแบบ จะแสดงสัมการข้างล่างนี้

$$g(x,y) = f(x,y) + \beta[f(x,y) - \overline{f}(x,y)] \qquad (4.6.1)$$

โดยที่ $g(x,y)$ = ค่าระดับเทาของภาพที่ตำแหน่ง (x,y)

$f(x,y)$ = ค่าระดับเทาของภาพหลังการทำอันชาร์ปมาส์กิ้ง

$\overline{f}(x,y)$ = ค่าระดับเทาของภาพความถี่ต่ำ ที่ตำแหน่ง x,y เป็นสัมประสิทธิ์ของภาพความถี่สูง (high frequency emphasis parameter) ที่ > 0

β = พารามิเตอร์เน้น (Emphasis Parameter)

สำหรับภาพความถี่ต่ำ $\overline{f}(x,y)$ หาได้จากการทำให้เรียบ (smoothing)คือการหาค่าเฉลี่ยของจุดภาพต่าง ๆ เทมเพล็กโดยการนำมาทำคอนโวลูชั่นกับข้อมูลภาพเดิมจะเขียนเทมเพล็กเป็นเมตริกซ์ขนาด 3x3 ได้ดังนี้

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

หรือจะใช้เทมเพล็กที่กล่าวมาแล้วในหัวข้อการทำภาพให้เรียบ ถ้าให้ค่า β มีค่าเท่ากับ 1 จากการมาส์กของอินชาร์ปมาส์กิ้งจะหาได้ดังนี้ สมมุติให้จุด x_4 เป็นจุดที่กำลังประมวลผลแทนจุด $f(x,y)$

$$\bar{f}(x,y) = 1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} X_0 & X_1 & X_2 \\ X_3 & X_4 & X_5 \\ X_6 & X_7 & X_8 \end{bmatrix}$$

$$= \frac{X_0+X_1+X_2+X_3+X_4+X_5+X_6+X_7+X_8}{9}$$

การหาความถี่สูงในแต่ละตำแหน่งจุดภาพ X_4 ได้จาก

$$f(x,y) - \bar{f}(x,y) = X_4 - \left[\frac{X_0+X_1+X_2+X_3+X_4+X_5+X_6+X_7+X_8}{9} \right]$$

$$= \frac{9X_4-X_0-X_1-X_2-X_3-X_4-X_5-X_6-X_7-X_8}{9}$$

เมื่อทำการเน้นขอบจะได้ว่า

$$g(x,y)=f(x,y)+[f(x,y)- \bar{f}(x,y)] = X_4+ \left[\frac{9X_4-X_0-X_1-X_2-X_3-X_4-X_5-X_6-X_7-X_8}{9} \right]$$

$$= \frac{9X_4+9X_4-X_0-X_1-X_2-X_3-X_4-X_5-X_6-X_7-X_8}{9}$$

$$= \frac{18X_4-X_0-X_1-X_2-X_3-X_4-X_5-X_6-X_7-X_8}{9}$$

$$= \frac{-X_0-X_1-X_2-X_3-17X_4-X_5-X_6-X_7-X_8}{9}$$

นั่นคือเมื่อนำมาเขียนในรูปเมตริกซ์ใหม่จะได้ว่า

$$g(x,y) = 1/9 \begin{bmatrix} -1 & -1 & -1 \\ -1 & 17 & -1 \\ -1 & -1 & -1 \end{bmatrix} \otimes \begin{bmatrix} X_0 & X_1 & X_2 \\ X_3 & X_4 & X_5 \\ X_6 & X_7 & X_8 \end{bmatrix}$$

ดังนั้น $\bar{f}(x,y)$คือค่าเฉลี่ยเฉพาะบริเวณ (local mean) และ $f(x,y) - \bar{f}(x,y)$ เป็นค่าความแปรปรวนจากค่าเฉลี่ยเฉพาะบริเวณ (deviation from local mean) หรือเรียกว่าเป็นค่าคอนทราสเฉพาะส่วน (local contrast)

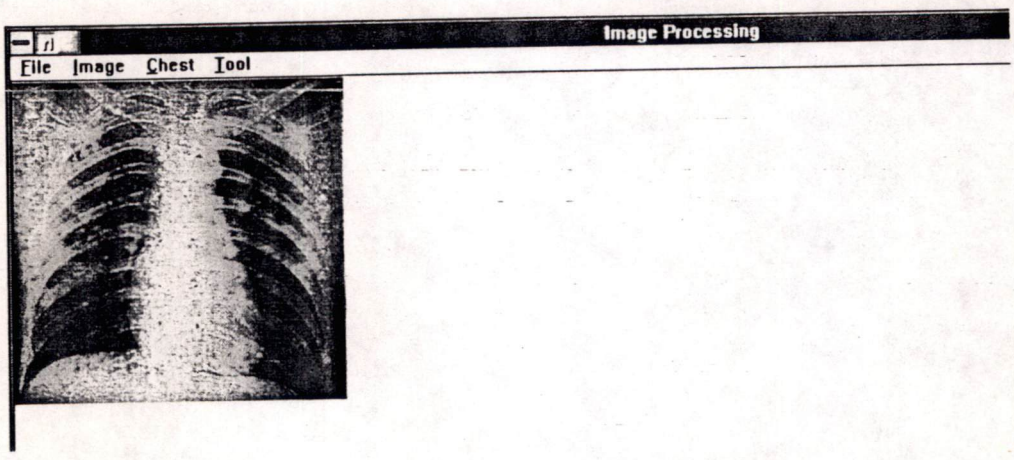
4.6.2 <u>การทำอันซาร์ปมาส์กิ้งแบบปรับได้</u> (Adaptive Unsharp Masking)

การทำอันซาร์ปมาส์กิ้งแบบปรับได้นั้น ตัวแปรสัมประสิทธิ์ในการเน้นขอบภาพ(emphasis parameter) จะเปลี่ยนแปลงหรือแปรตามค่าของจุดที่ f(x,y) หรือค่าเฉลี่ย เฉพาะบริเวณ (local mean) $\bar{f}(x,y)$ หรือแปรตามค่าคอนทราสเฉพาะส่วน (Local Contrast) $f(x,y) - \bar{f}(x,y)$ สามารถเขียนความสัมพันธ์ได้คือ

$\beta = \beta(f(x,y), \bar{f}(x,y), f(x,y) - \bar{f}(x,y)$ จากสมการที่ (4.6.1) สามารถเขียนได้เป็น

$$g(x,y) = f(x,y) + \beta\{ f(x,y), \bar{f}(x,y), f(x,y) - \bar{f}(x,y) \} .[f(x,y) - \bar{f}(x,y)] \qquad (4.6.2\text{-}1)$$

สำหรับวิธีนี้จะกล่าวถึงต่อไปในบทที่ 5



รูปที่ 4.6.1 แสดงภาพที่ปรับความคมชัด (Unsharp Masking)

4.7การหาขอบภาพ (Edge Detection)

เพื่อช่วยในการวิเคราะห์และวินิจฉัยภาพ โปรแกรมสามารถที่จะทำการหาขอบภาพได้สำหรับวิธีการที่ใช้ในการหาขอบภาพนั้นยังใช้หลักการเดิม คือการมาส์กเป็นเมตริกซ์ ห รือการทำคอนวูโลชั่ น จะนิยามขอบภาพว่าเป็นบริเวณระหว่างพื้นที่ 2 ส่วนซึ่งมีระดับความเข้มต่างกันมาก ขอบภาพนั้นจะหาได้จากการหาอนุพันธ์ (Differential) ในการหาอนุพันธ์อันดับหนึ่งจะได้ความชันหรือ อัตราการเปลี่ยนแปลงของระดับความเข้ม เมื่อหาอนุพันธ์อันดับที่สอง จะได้ค่าหรือบริเวณที่มีการเปลี่ยนแปลงมากๆหรือสูงสุด ดังนั้นบริเวณที่ว่านี้คือบริเวณที่เป็นขอบภาพ วิธีการหาขอบภาพมีอยู่หลายวิธีด้วยกัน วิธีการเหล่านี้ใช้การหา ค่าเกรเดียน (Gradient Operation)

$$G[\, f(x,y)\,] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix} \qquad (4.7.1)$$

จากสมการข้างบนค่าเกรเดียนของภาพที่ f(x,y) ที่ตำแหน่ง (x,y) กำหนดเป็นเวกเตอร์ 2 มิติ เป็นที่ทราบกันดีอยู่แล้วในเรื่องการวิเคราะห์เวกเตอร์ (Vector Analysis) ว่าเวกเตอร์ G แสดงทิศทางของอัตราการเปลี่ยนแปลงของ f ที่ตำแหน่ง(x,y) สำหรับการหาขอบภาพเราสนใจขนาดของเวกเตอร์ เราจะได้ค่าเกรเดียนเป็น

$$G[\, f(x,y)\,] = [\, G_x^2 + G_y^2\,]^{1/2} \qquad (4.7.2)$$

ขนาดที่ได้นี้เป็นอัตราการเปลี่ยนแปลงของ f(x,y) ต่อหน่วยระยะทางของ G เราสามารถประมาณโดยใช้ค่าสมบูรณ์

$$G[\, f(x,y)\,] = |\, G_x\,| + |\, G_y\,| \qquad (4.7.3)$$

ถ้าเรามาส์กเป็นเมตริกซ์ขนาด 3x3 ตามรูปที่แสดง จากสมการ (4.7.1) เราจะคำนวณหาค่า $\partial f / \partial x$ และ $\partial f / \partial y$ ที่ทุกๆจุดภาพ จะหาโดยใช้วิธีการใช้เมตริกซ์ขนาด 2x2

f(x,y)	f(x,y+1)
f(x+1,y)	f(x+1,y+1)

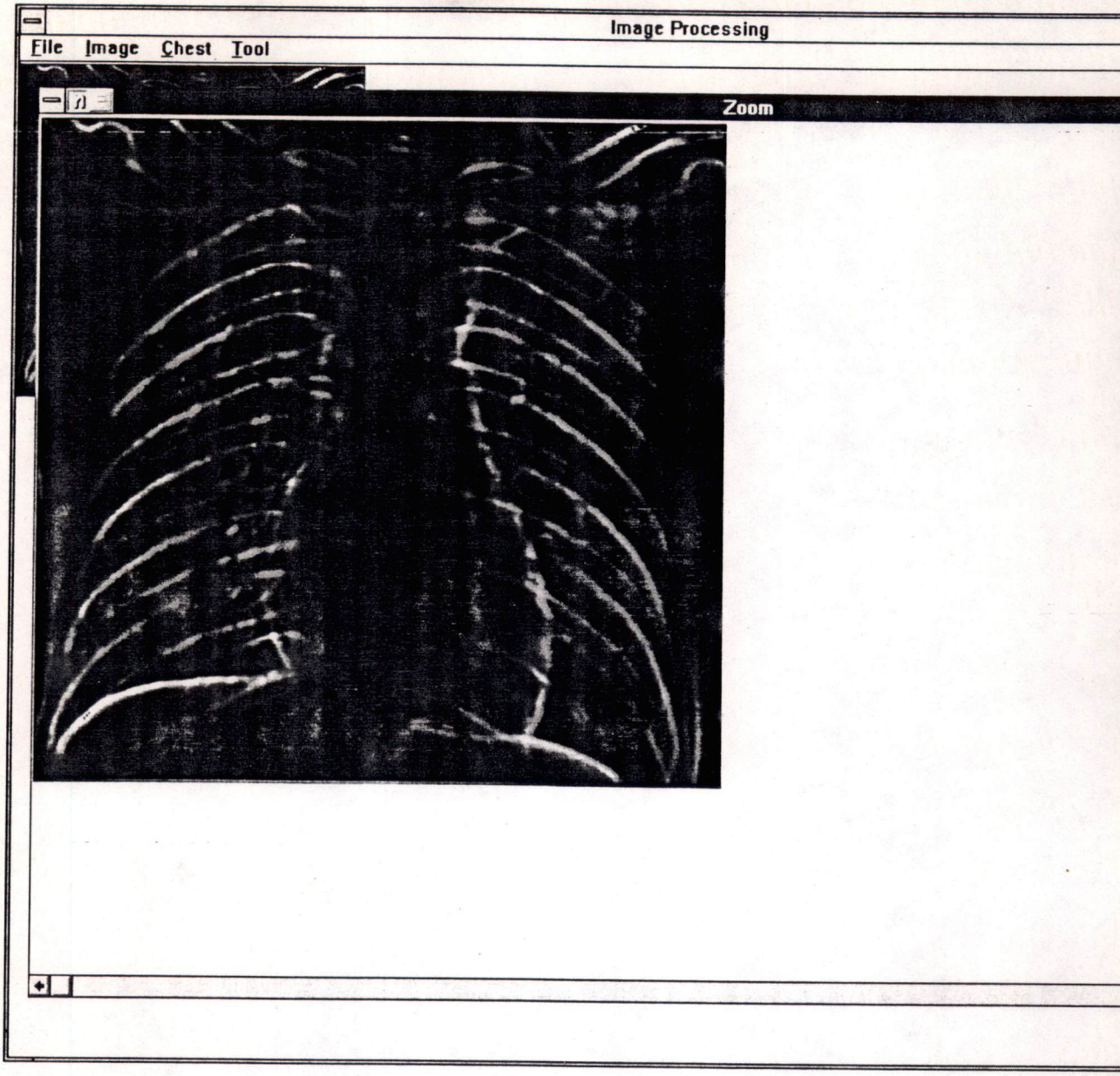
รูปที่ 4.7.1 เมตริกซ์ขนาด 2X2 ที่ใช้คำนวณ

วิธีการมาส์กแบบนี้เรียกว่า วิธีโรเบิรต์เกรเดียน (Robert Gradient) เขียนเป็นสมการได้ดังข้างล่างนี้

$$G[f(x,y)] = | f(x,y) - f(x+1,y+1) | + | f(x+1,y) - f(x,y-1) |$$

จะเห็นได้ว่าการประมาณข้างต้น เป็นค่าของเกรเดียนที่เป็นสัดส่วนกับค่าความแตกต่างระดับเทาของจุดภาพบริเวณใกล้เคียงกัน ค่าเกรเดียนจะมีค่ามากบริเวณที่เป็นขอบภาพหรือบริเวณที่มีการเปลี่ยนแปลงระดับความเข้มมากและค่าเกรเดียนจะมีค่าน้อยบริเวณที่มีการเปลี่ยนแปลงระดับเทาน้อย ตารางข้างล่างนี้แสดงการมาส์กเป็นรูปแบบวิธีการต่างๆ โดยค่าที่อยู่ในกรอบสี่เหลี่ยมคือค่าที่กำลังพิจารณา

	H_1	H_2
Robert	$\begin{bmatrix} \boxed{0} & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & \boxed{0} & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & \boxed{0} & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Isotropic	$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & \boxed{0} & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & \boxed{0} & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$



รูปที่ 4.7.2 แสดงภาพที่ทำการหาขอบภาพด้วยวิธีโซเบล (Sobel Operation)

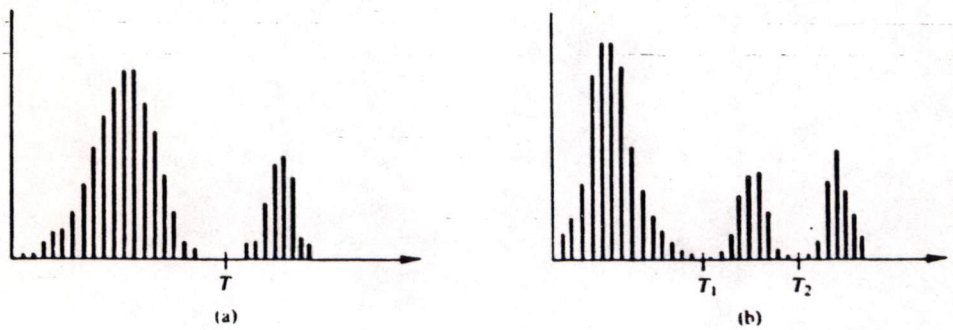
4.8 <u>การหาค่าเทรสโฮวด์</u> (Thresholding)

เทรสโฮวด์เป็นวิธีการหนึ่งที่สำคัญในการแยกส่วนภาพ (Segmentation) เทคนิคการหาค่าเทรสโฮวด์จะกล่าวต่อไปดังนี้ สมมุติค่าระดับเทาของฮิสโตแกรมดังรูปที่ 4.8.1(a)ของภาพ f(x,y) ที่ประกอบด้วยวัตถุที่สว่างและฉากหลังที่มืด ที่ถูกแบ่งออกเป็นสองส่วนของระดับเทา วิธีการที่จะแยกสองส่วนนี้ออกจากกันโดยใช้ค่าเทรสโฮวด์ T สำหรับทุก ๆ จุด(x,y) ที่ f(x,y)>T จะเรียกจุดเหล่านั้นว่าเป็นวัตถุ นอกจากนั้นเป็นส่วนของฉากหลังแต่ค่าของฮิสโตแกรมที่มีรูปแบบเป็นดังรูป 4.8.1(b) จะแบ่งลักษณะของภาพได้เป็น 3 ส่วนตัวอย่าง สองส่วนที่เป็นวัตถุสว่าง บนฉากหลังที่มืด ดังนั้นสามารถที่จะใช้วิธีการเดียวกันในการจำแนกจุด (x,y) เป็นดังนี้ ถ้า $T_1<f(x,y)\leq T_2$ ให้เป็นวัตถุหนึ่งและถ้า $f(x,y)>T_2$เป็นอีกวัตถุหนึ่ง และส่วนที่เหลือคือ $f(x,y)\leq T_1$ เป็นส่วนของฉากหลัง ลักษณะแบบนี้เราจะเรียกว่าเทรสโฮวด์หลายระดับ (Multilevel Thresholding) การหาค่าเทรสโฮวด์แบบนี้จะทำแยกกว่าเทรสโฮวด์ตัวเดียวที่จะมาแยกวัตถุที่สนใจในที่จะกล่าวถึงเฉพาะค่าเทรสโฮวด์ตัวเดียวสำหรับหลักการในการหาจะเขียนเป็นสมการฟังก์ชั่น T ที่มีรูปแบบดังข้างล่างนี้

$$T=T[x,y,P(x,y),f(x,y)] \qquad (4.8.1)$$

ที่ f(x,y) เป็นค่าระดับเทาที่จุด (x,y) และ P(x,y) แสดงคุณสมบัติเฉพาะส่วนของจุดนั้นเช่น ค่าเฉลี่ยของระดับเทารอบ ๆ จุด (x,y) เราสามารถสร้างภาพเทรสโฮวด์ g(x,y) โดยนิยาม

$$g(x,y) = \begin{cases} 1 & \text{ถ้า } f(x,y) > T \\ 0 & \text{ถ้า } f(x,y) \leq T \end{cases} \qquad (4.8.2)$$



รูปที่ 4.8.1 แสดงฮิสโตแกรมระดับเทา (a) เทรสโฮวด์เดียว (b) หลายเทรสโฮวด์

จากสมการข้างบนนี้ พบว่าค่าจุดที่ถูกแปลงเป็น 1 เป็นส่วนของวัตถุขณะที่ค่าระดับเทาของจุดที่ถูกแปลงเป็น 0 คือส่วนของภาพที่เป็นฉากหลัง เมื่อค่า T ขึ้นอยู่กับค่า f(x,y) เพียงอย่างเดียวจะเรียกว่าโกลบอลเทรสโฮวด์ (global Threshold) ตัวอย่างเช่นในภาพ 4.1.8(a) ถ้า ขึ้นอยู่กับ f(x,y) และ P(x,y) แล้วเรียกว่าค่าเทรสโฮวด์นั้นว่าเป็นโลคอลเทรสโฮวด์ (Local Threshold) ถ้าค่าเทรสโฮวด์ขึ้นอยู่กับตำแหน่งสปาเซี่ยน(coordinates spatial) ของ x และ y เรียกว่าไดนามิกเทรสโฮวด์ (dynamic threshold)

4.8.1 <u>วิธีการโกลบอลเทรสโฮวดิ้ง</u> (Global Thresholding Technique)

วิธีการง่าย ๆ ทีมักจะถูกใช้ในการแบ่งแยกส่วนของภาพคือ แบ่งค่าระดับเทาออกเป็นส่วนของระดับเทาและใช้ค่าเทรสโฮวด์เป็นตัวกำหนดขอบเขตของจุดภาพ ที่แสดงส่วนของภาพนั้น ๆ เทคนิคการทำวิธีการนี้คือ สมมุติให้ค่าระดับเทาของภาพ f (x,y) มีค่าฮิสโ ตแกรมแสดงดังรูป 4.8.2 (a) เราจะแยกฮิสโตแกรมที่มีจำนวนจุดมากใน f(x,y) ที่มืดออกจากส่วนที่เหลือที่มีกระจายของจำนวนจุดค่อนข้างจะเท่ากันออกจากกันฮิสโตแกรมดังรูปนี้เป็นฮิสโตแกรมที่มีฉากหลังค่อนข้างจะมีค่าระดับเทาไม่ต่างกันนักเราจะแบ่งภาพนี้ออกเป็น2 ส่วน โดยใช้ค่าเทรสโฮวด์ T แสดงดังรูป 4.8.2(b) ส่วนของ B_1 จะเป็นฉากหลังของภาพ ขณะที่ B_2 จะเป็นส่วนของวัตถุภาพจะถูกสแกนทั้งแนวตั้งและแนวนอน ดูการเปลี่ยนแปลงจะระดับเทาของส่วนช่วงระดับเทาหนึ่งไปยังอีกช่วงระดับเทาหนึ่งเมื่อแยกระดับความ เข้มออกเป็น 2 ส่วน คือ B_1และ B_2 เราจะได้จาก

<u>Pass 1</u> ในแต่ละแถวในภาพ f(x,y)(x=0,1,...,N-1) สร้างแถวที่มีลักษณะเดียวกัน g(x,y) โดยให้มีความสัมพันธ์กันสำหรับ y=1,2...,N-1

$$g_1(x,y) = \begin{cases} L_E & \text{ถ้าระดับเทาของ } f(x,y) \text{ และ } f(x,y-1) \text{ มีค่าระดับเทาอยู่คนละช่วงความถี่ของระดับเทา} \\ L_B & \text{ที่อื่น ๆ} \end{cases}$$

ที่ L_E และ L_B คือค่าระดับเทาของขอบภาพ และค่าระดับเทาของฉากหลังตามลำดับ

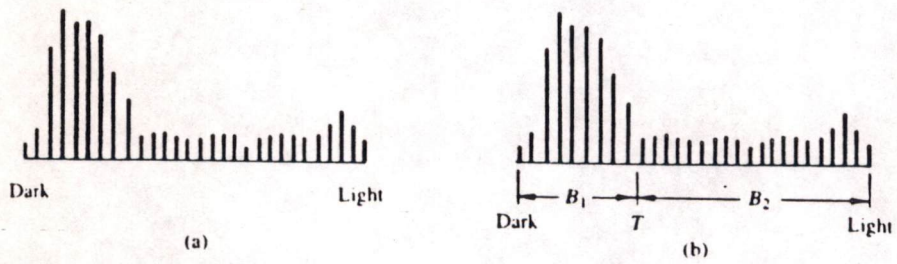
<u>Pass 2</u> แต่ละหลัก (column) ใน f(x,y) (i.e.,y=0,1,...,N-1)สร้างคอลัมน์ที่มีลักษณะเดียวกันในภาพ $g_2(x,y)$ ให้มีความสัมพันธ์ดังนี้ x=1,2,...,N-1

$$g_2(x,y) = \begin{cases} L_E & \text{ถ้าระดับเทาของ } f(x-1,y) \text{อยู่กันคนละช่วงความถี่ของระดับเทา} \\ L_B & \text{ที่อื่น ๆ} \end{cases}$$

เราจะแยกจุดบนภาพที่เป็นขอบวัตถุออกจากฉากหลังโดยใช้สมการต่อไปนี้ สำหรับ

x,y=0,1,...,N-1

$$g(x,y) = \begin{cases} L_E & \text{ถ้า } g_1(x,y) \text{หรือ} g_2(x,y) \text{ค่าหนึ่งค่าใดมีค่าเท่ากับ } L_E \\ L_B & \text{ที่อื่น ๆ} \end{cases}$$



รูปที่ 4.8.2 ฮิสโตแกรมเทรสโฮวด์

<u>การผิดพลาดของการหาค่าเทรสโฮวด์</u> (Thresholding error)

ในการแยกภาพออกเป็นสองส่วน ความผิดพลาดเกิดขึ้นจากการจำแนกของจุดที่เป็นส่วนของฉากหลัง และวัตถุโดยใช้ค่าเทรสโฮวด์ ซึ่งจะแยกค่าระดับเทาไปในช่วงระดับเทาหนึ่ง สามารถเกิดความผิดพลาดได้สองชนิดคือ

ชนิด 1 (Type1) ไม่นำค่าของจุดนั้นมาอยู่ในเขตหรือช่วงระดับเทา ทั้งที่ควรจะรวมอยู่ด้วย

ชนิด 2 (Type2) การนำจุดที่ไม่ควรอยู่ในกลุ่มระดับเทา แต่นำมารวมอยู่ด้วย

บทที่ 5
การปรับปรุงภาพโดยแยกส่วนของภาพรังสีเอ็กซ์เรย์ทรวงอก

ในบทนี้จะกล่าวถึงการปรับปรุงภาพรังสีเอ็กซ์เรย์ทรวงอก โดยแยกภาพออกเป็นสองส่วน เป็นปอดกับอวัยวะที่เป็นฉากหลังแล้วทำการปรับปรุงในแต่ละส่วนด้วยการหาค่าเทรสโฮวด์อย่างอัตโนมัติ เนื่องจากภาพรังเอ็กซ์เรย์ทรวงอกเป็นภาพที่มีรายละเอียดของภาพค่อนข้างสูง จะทำการปรับปรุงภาพในแต่ส่วนด้วยการปรับปรุงค่าระดับเทาของภาพ(Gray Scale Modification) ด้วยฮิสโตแกรมอิควอไรเซชั่นและทำอันชาร์ปมาส์กกิ้งแบบปรับได้ (Adaptive UnSharp Masking) เพื่อที่สามารถเห็นรายละเอียดของภาพในส่วนที่มืดได้ดีขึ้น สำหรับขั้นตอนการทำงานต่างๆ จะกล่าวเป็นข้อๆไป

5.1 การหาค่าเทรสโฮวด์โดยอัตโนมัติ

การหาค่าเทรสโฮวด์โดยอัตโนมัติมีอยู่ด้วยกันหลายวิธีด้วยกัน M.IBRAHIM SEZAN [5] ได้เสนอวิธีการหาค่าเทรสโฮวด์โดยการตรวจสอบค่าพีกของฮิสโตแกรม (Peak Detection Algorithm) เนื่องจากวิธีนี้ต้องการตัวแปรที่กำหนดขึ้นก่อนที่ได้มาจากการทดลองแทนค่าดูอยู่หลายตัวแปรจึงทำให้ไม่สะดวกในการใช้งาน ดังนั้นจึงทดลองใช้วิธีการหาค่าเทรสโฮวด์ของภาพระดับเทาโดยใช้ค่าเอนโทรปีของฮิสโตแกรมแทน สำหรับรายละเอียดของแต่ละวิธีจะกล่าวเป็นหัวข้อดังนี้

5.1.1 การหาค่าเทรสโฮวด์ด้วยการตรวจสอบค่าพีกของฮิสโตแกรม

(Peak Detection Algorithm)

วิธีการเลือกค่าเทรสโฮวด์มี 3 ขั้นตอนคือ อันดับที่ 1 ตรวจหาส่วนพีก(peak) ของฮิสโตแกรมภาพโดยใช้อัลกอริทึมตรวจสอบพีก (peak detection algorithm) อันดับที่ 2 ค่าของส่วนพีกของฮิสโตแกรมแต่ละอันจะถูกแบ่งโดยใช้ส่วนของขอบที่ติดกัน อันดับที่ 3 ค่าของส่วนพีก (peak) ที่ติดกันจะถูกรวมกันเป็นพีกเดียว(single peak)เพื่อที่จะแบ่งค่าพีกให้เหลือสองส่วนโดยพิจารณาว่ามีการชิดติดกันเพียงพอหรือไม่ อันดับสุดท้าย กำหนดกฎขึ้นมา 4 ข้อในการหาค่าเทรสโฮวด์แบ่งส่วนของปอดกับฉากหลัง

1) วิธีการตรวจหาส่วนพีก (The peak detection algorithm)

วิธีการหาค่าพีกเริ่มต้นจะต้องสร้างสัญญาณการตรวจสอบพีก (peak detection signal) จากฮิสโตแกรมของภาพแล้วเพื่อจะหาค่าพีกของฮิสโตแกรม จากค่าจุดตัดผ่านแกนศูนย์ (zero-crossing) ของสัญญาณการตรวจสอบ และค่าสูงสุดระหว่างค่าจุดตัดผ่านแกนศูนย์ในส่วนนั้น

ค่าสัญญาณการตรวจสอบพีก (peak detection signal) r_N หาได้จากการทำคอนโวโลชั่นของฮิสโตแกรม h กับเคอร์นัลตรวจสอบพีก (peak detection kernal)

$$r_N = P_N \otimes h \qquad (5.1.1)$$

ที่ P_N หาได้จากการทำคอนโวโลชั่นของดิฟฟีเรนเซอร์ δ(differencer)กับค่าสมูทเตอร์ (smoother) Λ_N ที่กำหนดดังนี้

$$\delta(n) = \begin{cases} -1 & , n = -1 \\ 1 & , n = 0 \end{cases} \qquad (5.1.2)$$

และ

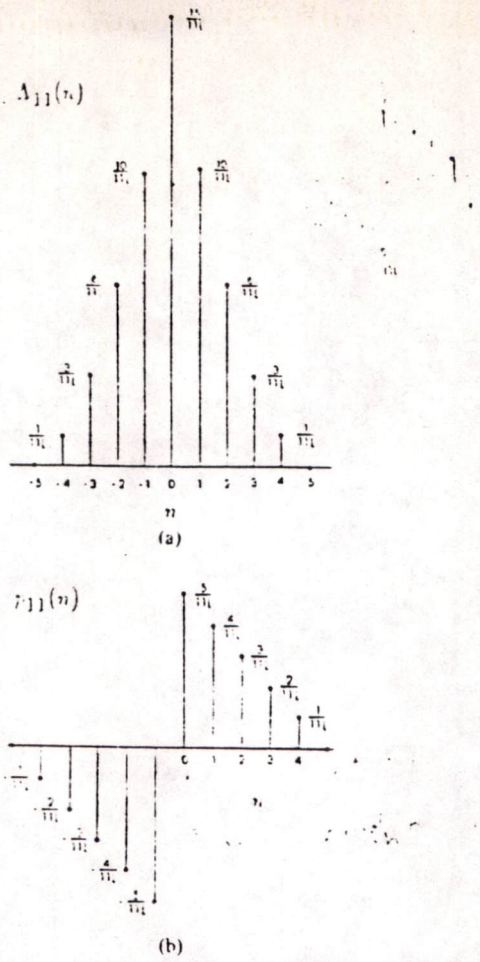
$$\Lambda_N(n) = \begin{cases} 1/NL & , n = -(N-1)/2 + 1 \\ (n + \frac{N-1}{2})\frac{1}{NL} + \Lambda_N(n-1) & , n = -\frac{(N-1)}{2} + 2, \ldots, 0 \\ \Lambda_N(-n) & , n = 1,2,\ldots,(N-1)/2 - 1 \end{cases}$$

ที่ L เป็นจำนวนจุดทั้งหมดในภาพ และให้ตัวแปร N เป็นจำนวนคี่ (ค่าของΛ_N จะเพิ่มขนาดเท่ากับ N-2) ดังนั้น เคอร์นัลของการตรวจสอบพีก (peak detection kernal) กำหนดดังนี้

$$P_N(n) = \delta \otimes \Lambda_N \qquad (5.1.3)$$

$$= \begin{cases} -[(N-1)/2 + n + 1]1/NL & , -(N-1)/2 \le n \le -1 \\ [(N-1)/2]1/NL & , n = 0 \\ [(N-1)/2 - n]1/NL & , 1 \le n \le (N-1)/2 - 1 \end{cases}$$

การทำการคอนโวโลชั่นด้วยดิฟฟีเรนเซอร์ (differencer) เป็นการประมาณการหาอนุพันธ์ (differentiation) ในกรณีของการทำฮิสโตแกรมให้เรียบ (smooth histrogram) ค่าพีกจะหาได้จากเครื่องหมายบวกลบทิศทาง และการผ่านจุดตัดแกนศูนย์(Zero-crossing) ของการทำคอนโวโลชั่นของ n กับ จุดตัดผ่านแกนศูนย์สามารถใช้ประมาณตำแหน่งของจุดสูงสุดและจุดเว้าของฮิสโตแกรม ในรูปที่ 5.1.1(a) จะแสดงตัวอย่างของฮิสโตแกรม โดยที่ตัวทำให้เรียบ (smoother) Λ_N มีขนาด N=11



รูป 5.1.1 แสดงตัวทำให้เรียบ (smoother Λ_N)และเคอร์เนลการตรวจสอบ P_N สำหรับ N=11 (a) Λ_N , (b) P_{11}

ในการพยายามที่จะแสดงผลของการทำคอนโวโลชั่นของฮิสโตแกรม กับเคอร์นัลของการตรวจสอบพีก จะแสดงจากการทำคอนโวโลชั่นของตัวทำให้เรียบ (smoother)กับตัวดิฟฟีเรนเซอร์ (differencer), P_N สำหรับ N_{11} แสดงในรูป 5.1.1(b) ในทางปฏิบัติกำหนดให้ค่าระดับเทา n_0 เป็นค่าศูนย์กลาง การคอนโวโลชั่นของเคอร์นัลการตรวจสอบพีก(peak detection kernal) กับฮิสโตแกรมผลลัพธ์ที่ได้ ในรูปของผลต่างของผลรวมค่าถ่วงน้ำหนัก (weight sum) ของค่าฮิสโตแกรมที่จุด -(N-1)/2 และผลรวมค่าถ่วงน้ำหนัก (N-1)/2 -1 ดังในรูป 5.1.1 ดังนั้นถ้าค่าพารามิเตอร์ N ลดลง ผลลัพธ์ของคอนโวโลชั่นก็จะสามารถหาค่าพีกที่มีขนาดเล็กได้ดี ถ้า N มีขนาดใหญ่ ก็จะใช้หาพีกที่มีขนาดใหญ่ (Global Peak)ได้ดีเราให้ N เป็นพารามิเตอร์ที่ใช้วิธีการนี้ การกำหนดขนาดของ N นั้นจะขึ้นอยู่กับการทดลองทำจากตัวอย่างว่าค่าใดที่เหมาะสม

ค่าสัญญาณการตรวจสอบ (detection signal) ในสมการที่(5.1.1) สามารถจะหาได้จากฟังก์ชั่นความหนาแน่นสะสม (cumulative density function : cdf) สามารถแสดงโดยสมการข้างล่าง

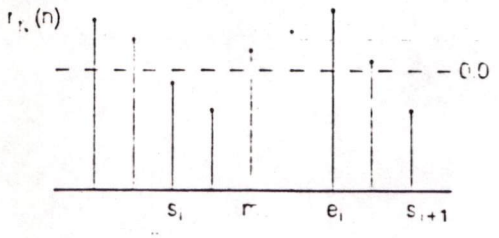
$$r_N = c - \overline{c}_N \qquad (5.1.4)$$

ที่ c เป็นนอร์มอลไรซ์ (normalize)cdf สำหรับภาพขนาด L จุด M ระดับเทา

$$c(n) = 1/L \sum_{l=0}^{n} h(l) \qquad , n = 0,1,...,M-1 \qquad (5.1.5)$$

และ $\bar{c_N}$ หาได้จากการคอนโวโลชั่น c กับหน้าต่างสี่เหลี่ยมแบบยูนิฟอร์ม (uniform)$W_N(n)$ $=1/N, -(N-1)/2 \leq n \leq (N-1)/2$ โดยจะทำการเฉลี่ยโดยเลื่อนหน้าต่างไปในจุดที่ต้องการหา

การหาตำแหน่งพีก (peak) จะกำหนดกฎขึ้นมาใช้เพื่อพิจารณาค่าสัญญาณ การตรวจสอบ r_n (ลักษณะของแต่ละพีกจะต้องประกอบด้วย จุดเริ่มต้น,จุดสิ้นสุด,จุดสุดยอด)



รูปที่ 5.1.2 แสดงลักษณะของพารามิเตอร์ในพีก

1.1) การผ่านจุดตัดแกนศูนย์ (Zero-crossing) ไปยังค่าลบของสัญญาณการตรวจสอบแสดงว่าจุดนั้นเป็นจุดเริ่มต้นของพีก สำหรับอันดับที่ i ของพีก จุดนี้จะให้เป็น s_i ในลักษณะเดียวกัน การตัดแกนศูนย์ไปยังลบ ค่าถัดไปที่ระดับเทา s_{i+1} แสดงว่าเป็นจุดเริ่มต้นของพีกถัดไป ค่าผ่านจุดตัดแกน 0 (zero-crossing) ของสัญญาณตรวจสอบไปยังค่าบวกที่เกิดขึ้นหลังจากจุดตัดผ่านแกนศูนย์ไปยังค่าลบ แสดงว่าได้พบจุดสูงสุดของพีกไปแล้ว จุดที่ผ่านจุดตัดแกนศูนย์ไปยังค่าบวกกำหนดเป็น m_i

1.2) ค่าที่สูงที่สุดระหว่างค่าผ่านจุดตัดแกนศูนย์ไปยังลบ (s_i กับ s_{i+1})แสดงว่าจุดนั้นเป็นจุดปลายสุดของพีก e_i โดยทั่วไปแล้วค่าพีกจะแสดงด้วยพารามิเตอร์ 3 ตัว (s_i,e_i,m_i)แสดงในรูปที่ 5.1.2 สำหรับในการหาพีกนั้นจะใช้พารามิเตอร์เพียงสองตัวคือ (s_i,e_i) ก็เพียงพอสำหรับอัลกอริทึม

2) รูปแบบการแบ่งพีก (Forming the Peak Clusters)

หลังจากหาพีกแต่ละตัวได้แล้วจะต้องมาทดสอบว่า ค่าพีกแต่ละอันนั้นมีค่าใกล้กันเพียงพอที่จะรวมกันเป็นพีกเดียวกันได้หรือไม่จะทำการทดสอบพีกที่ติดกันสำหรับ(s_k,e_k)และ (s_{k+1},e_{k+1}) ถ้า

$$s_{k+1} - e_k \leq d \qquad (5.1.6)$$

แล้วค่าพีกเหล่านี้จะรวมเป็นส่วนเดียวกันการแบ่งใหม่จะถูกแสดงดังนี้ (s_k, e_{k+1}) ถัดไปทำการตรวจสอบค่า(s_k,e_{k+1})กับ(s_{k+2},e_{k+2}) ว่ามีค่าชิด (closeness test) ชิดกันที่จะรวมกันได้หรือไม่ทำตามวิธีข้างต้นจนหมดจำนวนพีก หรือกรณีที่ค่าพีกไม่ได้ทำการรวมกันจะทำการตรวจสอบการชิดกันระหว่าง (s_{k+1},e_{k+1}) กับ (s_{k+2},e_{k+2}) เป็นค่าถัดไปโดยที่ค่าของ d เป็นพารามิเตอร์ทดสอบความชิดของแต่ละพีก จะถูกกำหนดจากการทำการทดลอง และศึกษาภาพรังสีเอ็กซ์เพื่อจะแยกส่วนที่เป็นปอดกับอวัยวะฉากหลัง หลังจากทำการตรวจสอบค่าพีกเสร็จเรียบร้อยแล้วจะทำการเรียงลำดับพีกที่ทำการรวมกับพีกอื่น และพีกที่ไม่ได้ทำการรวมกับพีกอื่นใหม่ลำดับเซตของการแบ่งชั้นของ (cluster) พีกจะแสดงเป็น$\{(S_i,E_i)\}$ ที่ S_i และ E_i เป็นจุดเริ่มต้นและจุดสิ้นสุดของการแบ่งพีกตามลำดับ

3. การเลือกเทรสโฮวด์ของปอดกับฉากหลัง (Lung/Mediastinum Thershold Selection)

ลำดับถัดมานี้จะเป็นการเลือกระดับเทาที่เป็นเทรสโฮวด์ จากการแบ่ง(cluster) ค่าพีกในหัวข้อที่แล้ว โดยจะกำหนดกฎขึ้นมาใช้ ซึ่งกฎนี้ได้มาจากการศึกษาภาพรังสีเอ็กซ์ 50 ภาพ

1) ถ้าแบ่งพีกได้ออกเป็น 2 ส่วน ค่าระดับเทาเทรสโฮวด์จะอยู่ที่ตำแหน่งระหว่างสองส่วนที่ทำการแบ่งได้

$$n_t = \text{int}\,[\,\mu E_1 + (1-\mu)S_2\,] \qquad (5.1.7)$$

ที่ n_t เป็นค่าเทรสโฮวด์ ,$0 \leq \mu \leq 1$ และ int[ ] เป็นค่าหาค่าจำนวนเต็มที่ใกล้ที่สุดโดยการตัดทศนิยม (nearest-integer truncation) ในที่นี้ให้ $\mu=0$

2) ถ้าจำนวนของการแบ่งส่วน (cluster) มากกว่าสอง,จะทำการหาค่าผลต่างของระดับเทาที่มากที่สุดของส่วนที่ทำการแบ่งที่ติดกัน ค่าเทรสโฮวด์จะอยู่ระหว่างการแบ่งส่วนของพีกเหล่านี้ แสดงการตรวจสอบการแบ่งส่วนเป็นคู่ ๆ เพื่อหาค่าที่มากที่สุดโดย(S_j,E_j) และ (S_{j+1},E_{j+1}) ตัวอย่างถ้าขนาดของ S_{j+1}-E_j มากที่สุดในทุก ๆ คู่ลำดับค่าเทรสโฮวด์ n_t จะหาได้ดังนี้

$$n_t = \text{int}\,[\,\mu E_j + (1-\mu)S_{j+1}\,] \qquad (5.1.8)$$

จะให้ μ=0 เหมือนในข้อข้างบน

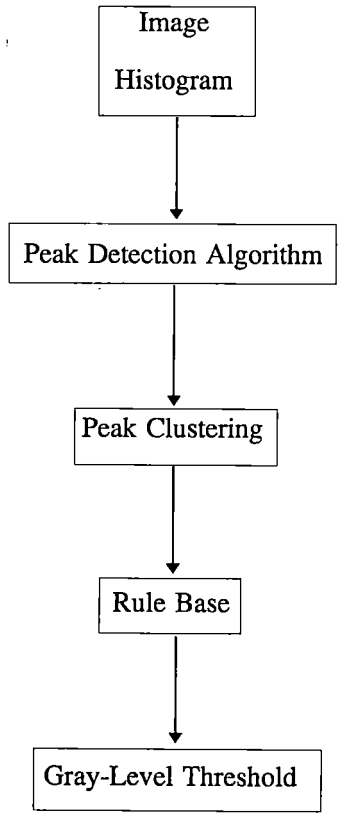
3) ถ้าการตรวจสอบพบว่ามีเพียงพีกเดียว และถ้ามีจำนวนของค่าระดับเทาอย่างน้อย P เปอร์เซนต์อยู่ระหว่างค่าระดับเทาที่มีจำนวนมากที่สุด n_{max} และ e_1 ดังสมการข้างล่าง

$$100\ (c(n_{max}) - c(e_1)) \geq P \qquad (5.1.9)$$

แล้วค่าเทรสโฮวด์ $n_t=e_1$ ถ้าไม่เป็นเช่นนั้นจะทำการตรวจสอบพีก และทำการแบ่ง (clustering) ใหม่การคำนวณโดยใช้พารามิเตอร์ N'(N'<N) ที่มีขนาดเล็กลงการทำการตรวจสอบพีก ที่ลดขนาดNลงจะทำให้ไวต่อการตรวจพีกเล็ก ๆ ได้ดีขึ้น ทำให้สามารถเพิ่มจำนวนพีกที่พบได้ ในทางปฏิบัติจะเปลี่ยนค่า N โดย (N+1)/2 กำหนดให้ K เป็นจำนวนครั้งที่จะทำซ้ำ ถ้าทำซ้ำไปKครั้งแล้วยังมีค่าพีกเพียงค่าเดียว และไปอยู่ในขอบเขตของ P เปอร์เซนต์แสดงว่าฮิสโตแกรมเป็นแบบยูนิโมเดอร์(unimodal histogram) ดังนั้น ค่าเทรสโฮวด์จะไม่สามารถหาได้ ในกรณีจะต้องทำกระบวนการต่าง ๆ กับภาพทั้งภาพ โดยกำหนดให้ n_t=0

4) ถ้าทำการรวมพีกแล้วค่าพีกทั้งหมดรวมเป็นส่วนเดียว(single cluster) และถ้ามีจำนวนระดับเทาอย่างน้อย P เปอร์เซนต์ อยู่ระหว่าง n_{max} กับ E_1 แล้ว $n_t=E_1$ ถ้าไม่ใช่จะทำการตรวจสอบและแบ่งส่วน (clustering) ใหม่โดยกำหนดให้มีพารามิเตอร์ที่ใหญ่ขึ้น (การเพิ่มค่าพารามิเตอร์ของการตรวจพีกเพื่อลดความไวในการตรวจสอบพีกให้ลดลง เมื่อจะแยกจำนวนพีกที่ใหญ่ขึ้นและน้อยลง) จะเปลี่ยนค่าพารามิเตอร์ N เป็น 2N-1 ถ้ากระทำไปจน K รอบแล้วยังรายได้เพียงส่วนเดียว (one cluster) แล้วจะกำหนดให้แต่ละพีกที่รวมกันเป็นส่วน (cluster) แยกออกมาเป็นแต่ละพีกอิสระเป็นส่วนหนึ่ง (cluster)ตามลำพัง และค่าเทรสโฮวด์จะหาได้จากกฎข้อ 2

จากการหาค่าเทรสโฮวด์อัตโนมัติ จะต้องระบุค่าพารามิเตอร์ 4 ตัว คือ Nพารามิเตอร์การตรวจสอบพีก,d พารามิเตอร์ของการตรวจสอบการชิดกัน, P ขอบเขตเปอร์เซนต์ K จำนวนรอบที่ให้ทำซ้ำ



รูปที่ 5.1.3 แสดงการหาค่าเทรสโฮวด์

5.1.2 การหาค่าเทรสโฮวด์ของภาพระดับเทาโดยใช้ค่าเอนโทรปีของฮิสโตแกรม

(Grey Level Picture Thresholding using the entropy of the Histogram)

เป็นการหาค่าเทรสโฮวด์โดยอัตโนมัติในการแบ่งแยกภาพ ด้วยค่าเอนโทรปีของฮิสโตแกรม หลักการคือการหาค่าสูงสุดของเอนโทรปีก่อนแบ่งภาพเป็นขาวกับดำด้วยค่าเอนโทรปีหลังการแปลงเป็นตัวกำหนด ถ้าภาพถูกแยกส่วนออกเป็นวัตถุกับฉากหลังของฮิสโตแกรมที่แบ่งออกเป็นสองส่วน (bimodal) ค่าของเทรสโฮวด์จะอยู่ระหว่างค่าส่วนเว้าของทั้งสองพีก (peak) ในภาพจริง ๆ แล้วการตรวจสอบส่วนเว้าที่ต่ำที่สุดของฮิสโตแกรมของทั้งสองพีกสามารถทำได้ยาก เนื่องจากฮิสโตแกรมอาจจะมีหลายพีก (peak)หรือไม่มีค่าพีกเลยก็ได้ สำหรับวิธีการนี้ไม่จำเป็นต้องอาศัยความรู้เกี่ยวกับลักษณะของภาพอื่นนอกจาก ค่าฮิสโตแกรมของระดับเทา

- เอนโทปรีของฮิสโตแกรม

1) นิยาม ให้ค่าจุดในภาพที่มีระดับเทาเท่ากับ n ระดับ [1,2,...,n] N_i จะแสดงจำนวนของจุดที่ระดับเทา i ดังนั้นจำนวนจุดทั้งหมดแสดงโดย N ที่ $N=N_1+N_2+...+N_n$ ค่าความน่าจะเป็นที่ระดับเทา i คือ $P_i=N_i/N$ จะได้

$$P_i = N_i/N \geq 0, \quad \sum_{i=1}^{n} P_i=1 \qquad (5.1.2\text{-}1)$$

ค่าระดับเทาของฮิสโตแกรมสามารถพิจารณาด้วย n ระดับเทาเป็นอิสระจากภาพนั้นเราสามารถตั้งสมมุติฐานตัวแปรเหล่านี้จะมีค่าสถิติที่เป็นอิสระจากกัน หรือจะกล่าวอีกนัยหนึ่งว่าจำนวนของจุดที่ระดับเทาหนึ่งไม่มีความสัมพันธ์กับจำนวนของจุดของระดับเทาใกล้ๆเพื่อที่จะทำการหาค่าเทรสโฮวด์ได้ง่ายขึ้น

ค่า information ที่ระดับเทา i ของฮิสโตแกรมเขียนได้ดังนี้

$$I_i = -\log_2(P_i) = -\mathrm{lb}(P_i) \qquad (5.1.2\text{-}2)$$

และค่าเอนโทปรี ดังสมการข้างล่างนี้

$$H = E[I_i] = -\sum_{i=1}^{n} P_i \cdot \mathrm{lb}(P_i) \qquad (5.1.2\text{-}3)$$

หลังจากการทำเทรสโฮวดิ้งภาพจะถูกแบ่งออกเป็นสองระดับคือขาว (w) กับดำ(b) จำนวนของจุดแต่ละระดับแสดงโดย N'_b และ N'_w ตามลำดับ และ P'_w กับ P'_bแสดงความน่าจะเป็นของระดับขาวกับดำ ซึ่งค่าของขาวกับดำเป็นอิสระต่อกัน ดังนั้นค่าเอนโทรปี ของภาพสองระดับจะเขียนได้ดังข้างล่างนี้

$$H' = -P'_w.lb(P'_w) - P'_b.lb_b(P'_b) \qquad (5.1.2\text{-}4)$$

2) <u>เอนโทรปีก่อนและหลัง</u> (A priori and posterior entropies)

ให้ s เป็นค่าของเทรสโฮวด์

$$N'_w = \sum_{i=1}^{s} N_i \quad , \quad N'_b = \sum_{i=s+i}^{n} N_i \qquad (5.1.2\text{-}5)$$

และ

$$P'_w = N'_w / N \quad , \quad P'_b = N_b / N \qquad (5.1.2\text{-}6)$$

สามารถกำหนดเป็นค่าเอนโทปรีย่อย ๆ ที่มีสมการคณิตศาสตร์ดังข้างล่าง

$$H_w = - \sum_{i=1}^{s} P_i.lb(P_i) \qquad (5.1.2\text{-}7)$$

$$H_b = - \sum_{i=s+1}^{n} P_i.lb(P_i) \qquad (5.1.2\text{-}8)$$

$$H'_w = - P'_w.lb(P'_w) \qquad (5.1.2\text{-}9)$$

$$H'_b = - P'_b.lb_b(P'_b) \qquad (5.1.2\text{-}10)$$

สองสมการแรกให้ความหมายแตกต่างจากสองสมการหลังซึ่งใช้ในการวัดปริมาณของ information ก่อน (priori) ที่จะทำเทรสโฮวด์ของจุดที่จะเป็นขาวและดำ โดยใช้ค่าเทรสโฮวด์ s (priori หมายถึง จุดที่ยังไม่ได้มีการแปลงเป็นระดับขาวหรือดำ) สำหรับเอนโทปรี 2 ค่าสุดท้าย H'_w และ H'_b ใช้วัดค่า information ของจุดขาวและดำ หลังจากทำเทรสโฮว จาก (3) และ (4) เราจะได้

$$H = H_w + H_b \ , \ H' = H'_w + H'_b \qquad (5.1.2\text{-}11)$$

จากสมการไม่มีความสัมพันธ์กันโดยตรงระหว่างH_w, H_b, H'_w, H'_b แต่มีความสัมพันธ์โดยอ้อมจากสมการที่ (5.1.2-5) และสมการที่ (5.1.2-7) ถึง (5.1.2-10)แทนที่จะใช้ค่าความน่าจะเป็นทางสถิติของรูปภาพ เราตั้งสมมุติฐานว่าฮิสโตแกรมสามารถเพียงพอที่จะใช้หาค่าเทรสโฮว์

3) <u>การเลือกค่าเอนโทปรีเทรสโฮวด์</u>

3.1) <u>หลักการ</u>

ในการหาค่าเทรสโฮวด์ที่จะทำภาพเป็นสองระดับเทานั้น ค่าเทรสโฮวด์นี้ะต้องทำให้เกิดความสัมพันธ์สูงสุดของภาพทั้งสองระดับ อันดับแรกจะต้องหาค่าเอนโทรปีหลังการแปลงสูงสุด H' โดยไม่คำนึงถึงความสัมพันธ์กับฮิสโตแกรมที่ยังไม่ได้ทำการเทรสโฮวด์ดังนี้

$$P'_b = P'_w \text{ ดังนั้น } N'_b = N'_w \qquad (5.1.2\text{-}12)$$

จากสมการ จำนวนจุดขาวและดำที่เท่ากันของภาพที่ทำการเทรสโฮวด์ จะให้ค่าเอนโทรปีสูงสุด ภาพที่มีค่าขาวครึ่งหนึ่งและดำครึ่งหนึ่งจะมีค่าเอนโทปรีเหมือนกัน

ค่าเอนโทรปรีก่อนทำเทรสโฮวด์ H สามารถหาได้จากฮิสโตแกรม เราพยายามที่จะกำหนดขอบเขตล่างสำหรับ H' เป็นพารมิเตอร์ในฟังก์ชั่นของของระดับเทา s ของค่าเทรสโฮวด์ จากจุดนี้เราะได้ผลลัพธ์ของเอนโทรปีก่อนการแปลงสูงสุดจากค่าเอนโทรปีหลังการแปลง H'

ในการทำเทรสโฮวดิ้ง จะแบ่งค่าเอนโทปรีก่อนเทรสโฮวด์ออกเป็น 2 ส่วนคือH_w และ H_b กำหนดให้ a เป็นค่าตัวแปร

$$H_w = \alpha H \quad , \quad H_b = (1 - \alpha)H , \qquad 0 \le \alpha \le 1 \qquad (5.1.2\text{-}13)$$

แสดงอัตราส่วนปริมาณของ informationของภาพก่อนเทรสโฮวด์ในจุดที่จะขาวและดำ

จากสมการ (5.1.2-7) และ (5.1.2-8) ให้

$$\sum_{i=1}^{s} P_i.lb(P_i) = \alpha \sum_{i=1}^{n} P_i.lb(P_i) \qquad (5.1.2\text{-}14)$$

$$= -\alpha H$$

ขอบเขตบนของสมการทางด้านซ้ายเขียนดังข้างล่าง

$$\sum_{i=1}^{s} P_i.lb(P_i) \le \sum_{i=1}^{s} [\, P_i.lb(\max\{P_1 \ldots P_s\}) \,] \qquad (5.1.2\text{-}15)$$

$$= lb(\max\{ P_1 .... P_s\}) . (\sum_{i=1}^{s} P_i)$$

สำหรับทุก ๆ i,Pi<1 ดังนั้น $lb(maxP_{1...}P_S)$ และส่วนซ้ายมือของสมการที่ 15 เป็นลบ เราจะไม่รวมกรณีที่ $lb(maxp_{1...}p_S)=0$ เพราะว่าจะเป็นการบอกว่าฮิสโตแกรมมีเพียงระดับเทาเดียวที่มีค่า เพราะ lb(1)=0 เมื่อมี Pi ตัวใดเป็น 1 ตัวที่เหลือเป็น 0 ดังนั้น

$$\sum_{i=1}^{s} P_i \leq \frac{\sum_{i=1}^{s}\{ P_i.lb(P_i) \}}{lb(\max\{ P_1 .... P_s\})} \qquad (5.1.2\text{-}16)$$

$$= \frac{- \alpha H}{lb(\max\{ P_1 .... P_s\})}$$

ในทางคล้าย ๆ กัน

$$\sum_{i=s+1}^{n} P_i \leq \frac{\sum_{i=s+1}^{n}\{ P_i.lb(P_i) \}}{lb(\max\{ P_{s+1} .... P_n\})} \qquad (5.1.2\text{-}17)$$

จากนิยามสมการที่ 4 ได้ค่าเอนโทปรีหลัง (posteriori) เทรสโฮวด์

$$-H' = P'_w.lb(P'_w) + P'_b.lb(P'_b) \qquad (5.1.2\text{-}18)$$

$$= (\sum_{i=1}^{s} P_i)lb(\sum_{i=1}^{s} P_i) + (\sum_{i=s+1}^{n} P_i)lb(\sum_{i=s+1}^{n} P_i)$$

ด้วยค่าขอบเขตบนใน (5.1.2-16) และ (5.1.2-17) จะได้ว่า

$$H' \leq \frac{- \alpha H.lb(\sum_{I=1}^{s} P_i)}{lb(\max\{ P_1 .... P_s\})} + \frac{(1 - \alpha)H.lb(\sum_{I=s+1}^{n} P_i)}{lb(\max\{ P_{s+1} .... P_n\})} \qquad (5.1.2\text{-}19)$$

ดังนั้น

$$H' \geq H \left[\frac{- \alpha . lb(\sum_{j=1}^{s} P_j)}{lb(\max\{ P_1 .... P_s \})} + \frac{(1 - \alpha) . lb(\sum_{j=s+1}^{n} P_j)}{lb(\max\{ P_{s+1} .... P_n \})} \right] \qquad (5.1.2\text{-}20)$$

$$H' \geq Fe(\alpha).H \qquad (5.1.2\text{-}21)$$

ที่ Fe(α) เป็นฟังก์ชั่นที่จะต้องหาค่าของ H' สำหรับอัลกอริทึมที่ใช้ในการหาค่าS ที่ดีที่สุด โดยใช้เอนโทปรีเทรสโฮวด์ จะกล่าวดังข้างล่าง (พยายามทำค่าเอนโทรปีก่อนเทรสโฮว์ เป็นตัวกำหนดเอนโทรปี H')

3.3 วิธีการเลือกเอนโทปรีเทรสโฮวด์

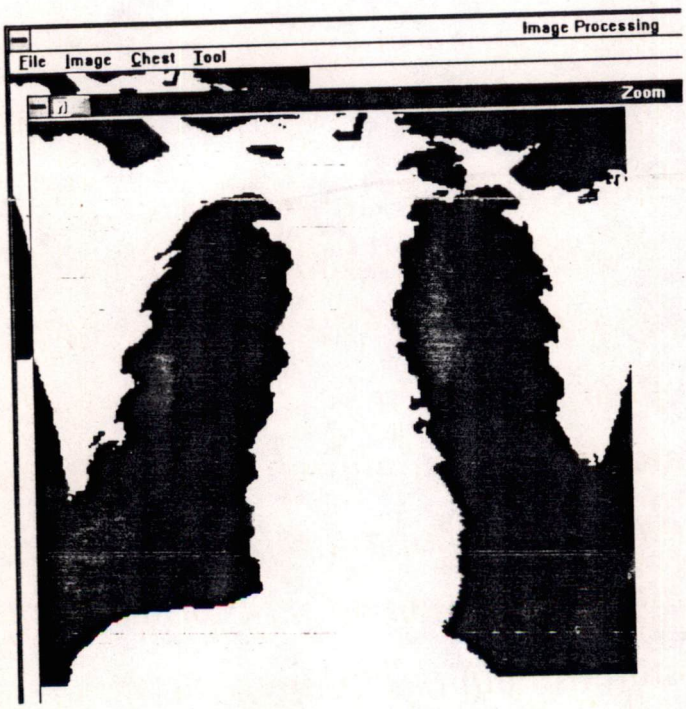
(a) คำนวณค่า α โดยใช้ค่า α เริ่มใกล้ 0

(b) หาค่า s จากสมการที่ (14)

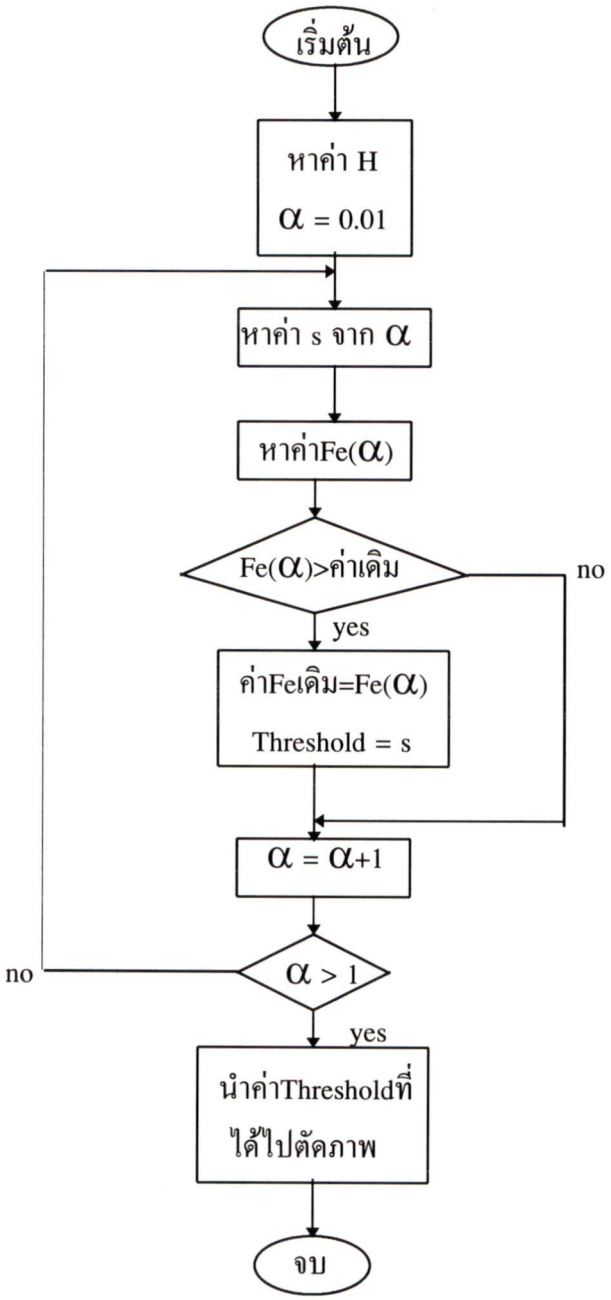
(c) คำนวณหาค่า Fe(α) ในสมการที่ (20)

(d) เพิ่มค่า α และไปที่ขั้นตอน b จนกว่า α เข้าใกล้ 1 ที่สุด

(e) หาค่า α ที่ให้ค่า Fe(α)สูงสุดค่าเอนโทรปีเทรสโฮวด์(entropy threshold) จะคำนวณได้จากสมการที่ (14)



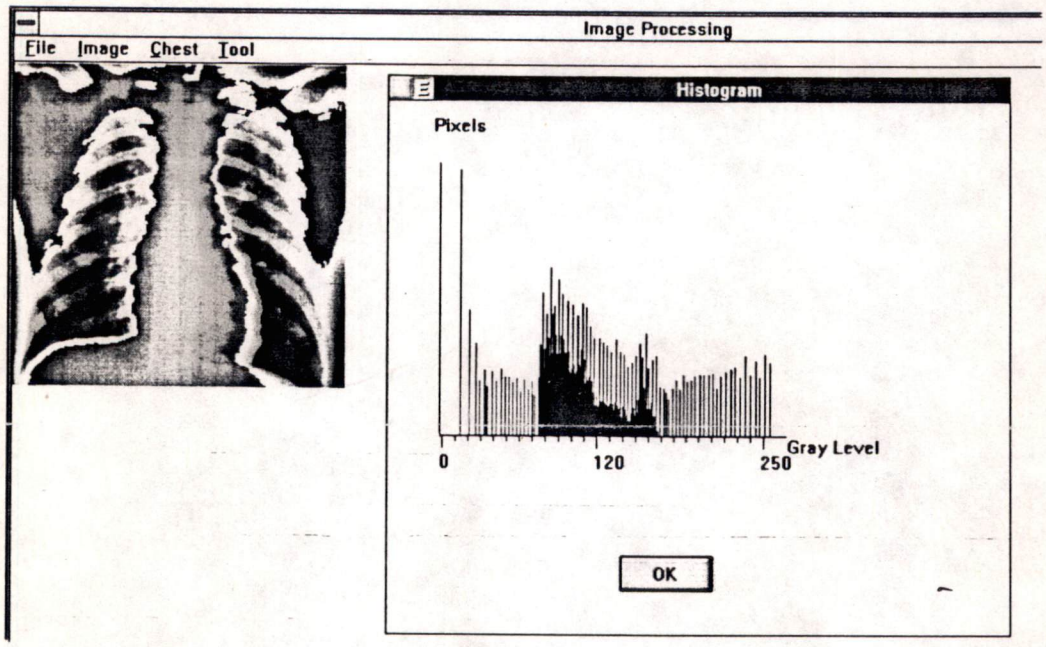
รูปที่ 5.1.2-1 ภาพที่หาค่าเทรสโฮวด์โดยใช้เอนโทรปี



5.1.2-2 แสดงโฟลว์ชาร์ตการทำงาน

5.2 การปรับปรุงภาพด้วยวิธีฮิสโตแกรมอิควอไรเซชั่นแบบแยกส่วนภาพ

เมื่อได้ค่าเทรสโฮวด์แล้วจะทำการแบ่งภาพออกเป็น 2 ส่วน ในส่วนที่มืดและส่วนที่สว่าง จากภาพที่ใช้ในการทำการทดลองส่วนที่เป็นบริเวณปอดจะเป็นภาพในส่วนที่มืด เราจะทำการปรับปรุงภาพในส่วนปอดนี้ด้วยวิธีฮิสโตแกรมอิควอไรเซชั่นในรูปที่ 5.2-1 จะเห็นได้ว่าฮิสโตแกรมของภาพในส่วนที่เป็นปอดจะสามารถกระจายระดับได้มากกว่าการคำนวณกับทั้งภาพเนื่องจากไม่นำค่าฮิสโตแกรมในส่วนอื่นมาใช้คำนวณด้วย การปรับปรุงภาพในส่วนที่เป็นปอดนี้สามารถใช้วิธีการปรับขยายค่าคอนทราส (Contrast Streching) สำหรับภาพที่ทำการปรับปรุงแล้วแสดงดังภาพ 5.2-1



รูปที่ 5.2-1 แสดงภาพที่ทำการปรับปรุงภาพในส่วนปอดโดยวิธีฮิสโตแกรมอิควอไรเซชั่น

5.3 การทำอันชาร์ปมาส์กแบบปรับได้แปรตามค่าระดับเทาของภาพ

การนำวิธีอันชาร์ปมาส์กิ้ง (Unsharp Masking) ในการปรับปรุงภาพที่แยกส่วนนั้นจะใช้วิธีอันชาร์ปมาส์กิ้งแบบปรับได้ดังได้กล่าวมาแล้วในบทที่ 4 พารามิเตอร์เน้น (Emphasis

Parameter) β จะแปรตามค่าระดับเทาของภาพ และนำค่าเทรสโฮวด์ที่ได้มาช่วยในการคำนวณ ดังในรูปที่ 5.3-1 สำหรับสมการการทำอันชาร์ปมาส์กิ้งแสดงในสมการที่ (5.3-1)

$$n'(i,j) = n(i,j) + \beta(n)[\, n(i,j) - \bar{n}(i,j) \,] \qquad (5.3\text{-}1)$$

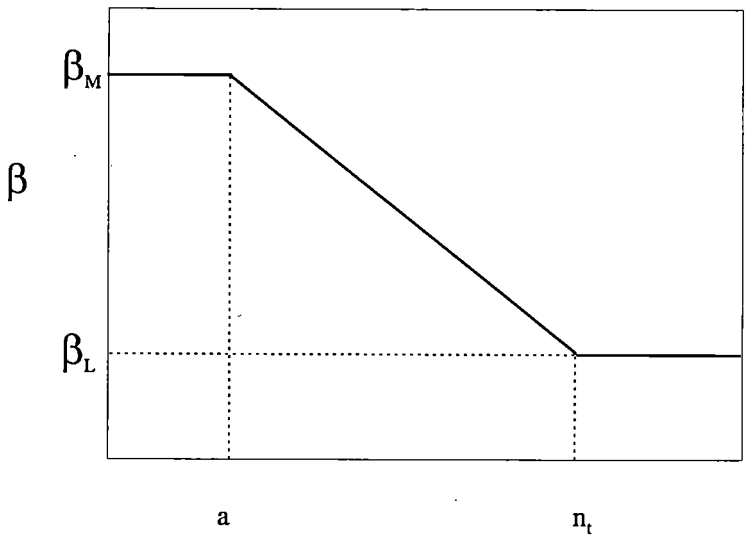
ที่ $n'(i,j)$ = ค่าระดับเทาที่จุด i,j หลังการคำนวณ

$n(i,j)$ = ค่าระดับเทาที่จุด i,j

$\beta(n)$ = ค่าพารามิเตอร์เน้นแปรตามค่าระดับเทาของภาพ

$\bar{n}(i,j)$ = ค่าเฉลี่ยรอบจุด i,j

ในหัวข้อนี้จะนำค่าเทรสโฮวด์ที่ได้มาแยกส่วนอวัยวะส่วนที่เป็นปอดกับอวัยวะส่วนอื่น และจะนำค่านี้มาเป็นตัวกำหนดค่าของตัวแปรที่ β ที่จะมาเน้นขอบภาพ จะกำหนดให้พารามิเตอร์นี้มีค่ามากในส่วนที่มืดของภาพเพื่อที่พยายามเห็นรายละเอียดภาพในส่วนนี้จึงต้องให้มีค่ามากในส่วนที่สว่าง สำหรับรูปแบบของการเปลี่ยนแปลงค่าพารามิเตอร์ $\beta(n)$ ในรูปที่ 5.3-1

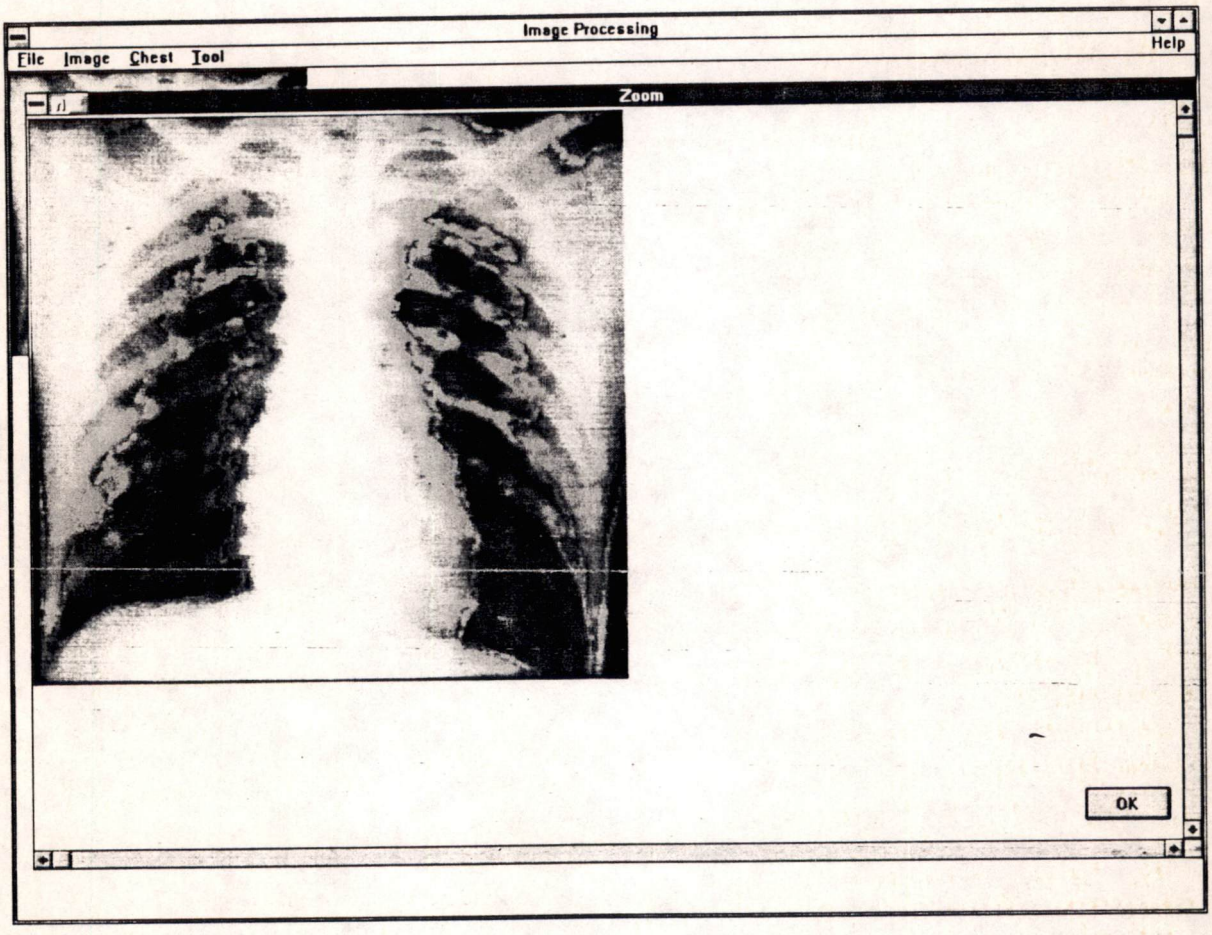


รูปที่ 5.3-1 รูปแบบของพารามิเตอร์เน้น (Emphasis Parameter) จะมีค่ามากในส่วนที่มืด ($n \leq n_t$) และมีค่าน้อยในส่วนที่สว่าง ($n > n_t$) ดังนั้น $\beta_M > \beta_L$ พารามิเตอร์ a จะเป็นค่าระดับเทาที่ใช้ในการคำนวณหาค่า β

ค่า n_t คือค่าเทรสโฮวด์ที่ใช้แบ่งแยกภาพ ค่าตัวแปร a ($a < n_t$) เป็นระดับเทาที่ต้องหาค่า

$\beta(n)$ ที่อยู่ในช่วง β_M และ β_L สามารถหาได้ดังสมการ (5.3-2)

$$\beta(n) = \begin{cases} \beta_M & , \ 0 \le n \le a \\ -n(\beta_M - \beta_L)/(n_t - a) + (n_t - a)/(n_t \beta_M - a \beta_L) & , \ a \le n \le n_t \\ \beta_L & , \ n > n_t \end{cases} \quad (5.3\text{-}2)$$



รูปที่ 5.3-2 ภาพที่ทำการอันชาร์ปมาส์กด้วยβ_M = 3และβ_L = 1

บทที่ 6
บทสรุปและข้อเสนอแนะ

6.1 สรุปงานวิจัย

ในการจัดทำโปรแกรมฐานข้อมูลภาพทางการแพทย์ที่สามารถนำเอาอัลกอริทึมต่างๆ ทางการประมวลผลข้อมูลภาพ มาใช้ทำให้แพทย์ผู้ทำการวินิจฉัยภาพสามารถปรับปรุงภาพที่มีคุณสมบัติไม่ดีให้สามารถมองเห็นรายละเอียดส่วนใหญ่ของภาพได้ดีขึ้น ซึ่งการต้องการในการปรับปรุงภาพส่วนใหญ่แล้วภาพจะมีปัญหาทางด้านค่าคอนทราสของภาพต่ำ ทำให้ไม่สามารถเห็นรายละเอียดของภาพได้ชัดเจนเท่าที่ควร หรือภาพที่ได้อยู่ในโทนที่ค่อนข้างมืด ดังนั้นจุดประสงค์หลักคือพยายามที่จะหาอัลกอริทึมที่สามารถปรับปรุงภาพให้มีคอนทราสเพิ่มขึ้น และสามารถวินิจฉัยภาพได้ดีขึ้น ในบทที่ 5 นั้นได้กล่าวถึงการทดลองการทำฮิสโตแกรมอิควอไรเซชั่นโดยแยกภาพออกเป็นสองส่วนด้วยค่าเทรสโฮวด์ และใช้ฮิสโตแกรมในแต่ละส่วนของภาพมาสร้างฟังก์ชั่นในการแปลงเพื่อที่จะปรับปรุงเฉพาะในส่วนที่เป็นปอด จากผลการทดลองในภาพที่ 5.2-1 จะทำการปรับปรุงเฉพาะในส่วนที่เป็นปอด เพื่อดูผลของวิธีการจะเห็นว่าในส่วนที่เป็นปอดจะเห็นรายละเอียดได้ดียิ่งขึ้น และจากฮิสโตแกรมของภาพจจะเห็นได้ว่าค่าระดับเทาในส่วนที่เป็นปอดสามารถกระจายไปอยู่ในช่วง 0-255 ได้ทั้งหมด นอกจากนั้นยังได้ทำการทดลองใช้วิธีอันชาร์ปมาส์กิ้งแบบปรับได้ ภาพที่ได้จากวิธีการนี้จะเห็นรายละเอียดภาพในส่วนที่มืดได้ดีขึ้น

เนื่องจากในภาพรังสีเอ็กซ์บางชนิดมีความละเอียดสูงจจึงไม่สามารถที่จะนำมาจัดเก็บได้ ตัวอย่างเช่น ภาพรังสีเอ็กซ์เรย์ทรวงอก แพทย์ผู้ทำการวินิจฉัยต้องการความละเอียดที่ 2048x2048 จุดภาพขึ้นไป ทำให้ไม่สามารถจัดหาอุปกรณ์การถ่ายภาพชนิดนี้ได้เพราะมีราคาค่อนข้างสูง ดังนั้นภาพที่ใช้ในบทที่ 5 จึงเป็นเพียงการนำตัวอย่างภาพมาทดลอง แต่ในปัจจุบันเทคโนโลยีทางด้านนี้ได้พัฒนาไปเร็วมาก ภายในเวลาอันใกล้นี้อุปกรณ์เหล่านี้จะมีราคาถูกลงและสามารถซื้อหานำมาใช้ได้

จากการนำเอาอัลกอริทึมต่างๆมาใช้ในการทำงาน สามารถที่จะให้ภาพที่มีคุณสมบัติดีขึ้นกว่าเดิม แต่ในการที่จะวัดว่าภาพที่ได้นั้นสามารถให้ความเที่ยงตรงในการวินิจฉัยของแพทย์นั้นยังทำได้ยาก อาจจะต้องอาศัยจากการทำการทดลองแล้วนำผู้แพทย์ที่วินิจฉัยภาพหลายๆคนมาทดลองดูผลลัพธ์ที่ได้

เอกสารอ้างอิง

1) Digital Image Processing , Rafael C. Gonzalez/Paul Wintz , Second Edition , Addison-Wesley Publishing Company,1987

2) Introduction Computer Vision Image Processing , Adrian Low ,McGrawHill,1991

3) Fundamental of Digital Image Processing , Anil K. Jain , Prentice Hall,1989

4) Digital Image Processing , William K. Pratt , Second Edition , A Wiley-Interscience Publication John Wiley & Son ,1991

5) Automic Anatomically Selective Image Enhancement in Digital Chest Radiography , M. Ibrahim Sezan,IEEE Transaction of Medical Imaging ,Vol. 8, No.2, June 1989 , p. 154-162

6) A New Method For Gray-Level Picture Thresholding Using The Entropy of Histogram , Thierry PUN , Signal Processing 2 (1980) , p 223-237

7) Regionally Adaptive Histogram Equalization of the Chest , Robert H. Sherrier and G. A. Johnson,IEEE Transaction of Medical Imaging ,Vol. MI-6 , No. 1, March 1987 , p 1-7

8) การปรับปรุงภาพรังสีเอ็กซ์ (Enhancement of Radiography) , บริสุทธิ์ พันธ์มา , กวิน สนธิเพิ่มพูน, ดร. ไพรัช ธัชยพงษ์, วิศวกรรมศาสตร์บัณฑิต , พระจอมเกล้าลาดกระบัง , 2532

9) การปรับปรุงรายละเอียดของขอบในภาพสีด้วยการแปลงภาพผลลบที่ถูกทำให้เรียบ (Edge Enhancement in Color Image Using Subtracted Smoothing Image) , อาโมทย์ สมบูรณ์แก้ว , ดร. ฟูศักดิ์ ชีวสุวิทย์ , วิศวกรรมศาสตร์มหาบัณฑิต , พระจอมเกล้าลาดกระบัง , 2537

ภาคผนวก ก

```
#include <windows.h>
#include <memory.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <io.h>
#include <conio.h>
#include <math.h>

#include "imagepro.h"

#define PALETTESIZE 256
#define  RGB_RED                0
#define  RGB_GREEN      1
#define  RGB_BLUE       2
#define  RGB_SIZE       3

typedef struct {
        int width,depth,bytes,bits;
        int background;
        unsigned char palette[768];
        int (*setup)();
        int (*closedown)();
        } FILEINFO;

typedef struct {
        char id[2];
        long filesize;
        int reserved[2];
        long headersize;
        long infoSize;
        long width;
        long depth;
        int biPlanes;
        int bits;
        long biCompression;
```

```
        long biSizeImage;
        long biXPelsPerMeter;
        long biYPelsPerMeter;
        long biClrUsed;
        long biClrImportant;
        } BMPHEAD;

FILEINFO fi;

HDC hDC;
HWND hInst;
HWND hScroll0;
FARPROC lpScroll0Standard;
BOOL FlagBright;
int test =0 ;
FILE *fp,*fp2;
LOGPALETTE *pPal;
HPALETTE hpal = NULL;
PAINTSTRUCT PtStr;
WORD nNumColors;
BMPHEAD bmp;
// Allocate Memory for Image
HGLOBAL hglb1,hglb2,hglb3,hglb4,hglb5;
unsigned char far *buffer1;
unsigned char far *buffer2;
unsigned char far *buffer3;
unsigned char far *buffer4;
unsigned char far *buffer5;
int RowPerPage,TotalPage,RowRemainder;
HGLOBAL handle1;
unsigned char far *image;
int Position;                          // for scroll bar


int FirstFlag = 0;
```

```
long FAR PASCAL WndProc (HWND hWnd, WORD iMessage,
                         WORD wParam, LONG lParam);

long FAR PASCAL Scroll0Proc(HWND,WORD,WORD,LONG);
void colorpalette(HWND hWnd,unsigned char *p);
void ChangeFormat(HWND);
void WriteBmp(HWND hWnd);
void AllocMem(HWND);
void ReadMem(unsigned char far *,int,int);
void WriteMem(unsigned char far *,int,int);
void LoadImage(void);
void Brightness(int);
void Contrast(HWND,int,unsigned char *);
void AutoContrast(void);
void Sharp(void);
void Smooth(void);
void Weight(void);
void Edge(void);
void HistEqualize(void);
int  Entropy(void);
void Write2Level(void);
int  CalEmphasis(int gray,int Threshold);
void AnatomicChest(void);
void AnatomicEqualize(void);
BOOL FAR PASCAL OpenNewDlgProc(HWND,unsigned,WORD,LONG);
BOOL FAR PASCAL ScrollDlgProc(HWND,unsigned,WORD,LONG);
BOOL FAR PASCAL GraphDlgProc(HWND,unsigned,WORD,LONG);
BOOL FAR PASCAL ZoomDlgProc(HWND,unsigned,WORD,LONG);

static char szNameText[15];


int PASCAL WinMain (HANDLE hInstance, HANDLE hPrevInstance,
                    LPSTR lpszCmdParam, int nCmdShow)
{
HWND hWnd;
```

```
MSG Message;
WNDCLASS WndClass;

if (!hPrevInstance)
        {
  memset(&WndClass,0x00,sizeof(WNDCLASS));
        WndClass.cbClsExtra = 0;
        WndClass.cbWndExtra = 0;
        WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
        WndClass.hCursor =  LoadCursor(NULL, IDC_ARROW);
        WndClass.hIcon = LoadIcon (NULL, "END");
        WndClass.hInstance = hInstance;
        WndClass.lpfnWndProc =    WndProc;
        WndClass.lpszClassName = "BRIGHT";
        WndClass.lpszMenuName = "MENU";
        WndClass.style = CS_HREDRAW | CS_VREDRAW;

        RegisterClass (&WndClass);
        }

hInst = hInstance;
hWnd  = CreateWindow ("BRIGHT",         /* class name    */
                "Image Processing",             /* Caption       */
                WS_OVERLAPPEDWINDOW,            /* Style         */
                0,                              /* x position    */
                0,                                  /* y position    */
                1024,                           /* cx - size     */
                768,                                /* cy - size     */
                NULL,                               /* Parent window */
                NULL,                               /* Menu          */
                hInstance,                      /* Program Instance */
                NULL);                          /* Parameters    */

ShowWindow (hWnd, nCmdShow);
while (GetMessage (&Message, 0, 0, 0))
        {
```

```
        TranslateMessage(&Message);
        DispatchMessage(&Message);
        }
return Message.wParam;
}

/*********************************************************/
/*              Window Procedure: WndProc                */
/*********************************************************/

long FAR PASCAL WndProc (HWND hWnd, WORD iMessage,
                         WORD wParam, LONG lParam)
{


//PAINTSTRUCT PtStr;
int i,x, y;



BYTE red,green,blue;
char str[40];



char ch;
int maxx,maxy,wchunck;
double NumChunck;
int xScreen,yScreen,bPos;
// xms
static FARPROC lpScroll0Proc;
unsigned char color;
float value;
int j,n,nalloc;
long off;   // offset of target
```

```
FARPROC lpprocDia;
//static HANDLE hInstance;



switch (iMessage)
        {
        case WM_COMMAND:
           switch(wParam)
                {
                case ID_LOADIMAGE:
                        UpdateWindow(hWnd);
                        LoadImage();
                        return 0;

                case ID_SAVE :
      memset(szNameText,' ',15);
                        lpprocDia = MakeProcInstance(OpenNewDlgProc,hInst);
                        if (DialogBox(hInst,"OPENNEW",hWnd,lpprocDia)){
      WriteBmp(hWnd);
    }
                        FreeProcInstance(lpprocDia);
    break;

                case ID_EXIT:
                        SendMessage(hWnd, WM_CLOSE, 0, 0L);
                        return 0;

                case ID_BRIGHT :
                        lpprocDia = MakeProcInstance(ScrollDlgProc,hInst);
                        if (DialogBox(hInst,"SCRLBOX",hWnd,lpprocDia)){
                                Brightness(Position);
    }
                        return 0;

                case ID_CONTRAST1 :
```

```
                lpprocDia = MakeProcInstance(ScrollDlgProc,hInst);
                if (DialogBox(hInst,"SCRLBOX",hWnd,lpprocDia)){
                        Contrast(hWnd,Position,fi.palette);
                }
                return 0;

        case ID_CONTRAST2 :
                UpdateWindow(hWnd);
                AutoContrast();
return 0;

        case ID_GRAPH :
                lpprocDia = MakeProcInstance(GraphDlgProc,hInst);
                DialogBox(hInst,"GRAPH",hWnd,lpprocDia);

return 0;

        case ID_SHARP :
                UpdateWindow(hWnd);
                Sharp();
                return 0;

        case ID_AVERAGE :
                UpdateWindow(hWnd);
                Smooth();
                return 0;

        case ID_WEIGHT :
                UpdateWindow(hWnd);
                Weight();
                return 0;

        case ID_HISTEQ :
                UpdateWindow(hWnd);
                HistEqualize();
                return 0;
```

```
            case ID_ADHIST :
                    UpdateWindow(hWnd);
//      AdaptiveHist();
        return 0;

            case ID_EDGE :
                    UpdateWindow(hWnd);
                    Edge();
                    return 0;

            case ID_ENTROPY :
                    UpdateWindow(hWnd);
    Write2Level();
                    return 0;

            case ID_CHEST1 :
                    UpdateWindow(hWnd);
    AnatomicChest();
                    return 0;

            case ID_CHEST2 :
                    UpdateWindow(hWnd);
                    AnatomicEqualize();
                    return 0;

            case ID_ZOOM2 :
                    lpprocDia = MakeProcInstance(ZoomDlgProc,hInst);
                    DialogBox(hInst,"ZOOM",hWnd,lpprocDia);
        return 0;

            case ID_ABOUT :
                    MessageBox(hWnd,
                      "Sample Application",
                      "Windows Graphics",
                      MB_ICONINFORMATION|MB_OK);
```

```
                return 0;
                }
                return 0;


        case WM_PAINT:
                hDC = BeginPaint(hWnd, &PtStr);
                ////
         if (FirstFlag ==0){
                ChangeFormat(hWnd);
    FirstFlag = 1;
         }

 //**************************************
          FlagBright = 0;
                pPal = (LOGPALETTE*) LocalAlloc(LPTR,sizeof(LOGPALETTE)+nNumColors*sizeof
(PALETTEENTRY));
                if (!pPal){
                        TextOut(hDC,10,10,"Allocate pPal Failed ",21);
        exit(1);
                }
                pPal->palNumEntries = nNumColors;
     pPal->palVersion = 0x300;
                for (i=0;i<nNumColors;i++){
                        pPal->palPalEntry[i].peRed   = fi.palette[(i*3)];
                        pPal->palPalEntry[i].peGreen = fi.palette[(i*3)+1];
                        pPal->palPalEntry[i].peBlue  = fi.palette[(i*3)+2];
                        pPal->palPalEntry[i].peFlags = (BYTE) 0;
                }
                hpal = CreatePalette(pPal);
                LocalFree((HLOCAL) pPal);
                hDC  = GetDC(hWnd);
                SelectPalette(hDC,hpal,0);
                RealizePalette(hDC);

                for (y=0;y<fi.depth;y++){
```

```
        ReadMem(image,fi.width,y);
                        for (x=0;x<fi.width;x++){
          color = *(image+x);
                          SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(color));
                        }
                }

                EndPaint(hWnd, &PtStr);
            return 0;

          case WM_DESTROY:
                PostQuitMessage(0);
                SetWindowLong(hScroll0,GWL_WNDPROC,(DWORD)lpScroll0Standard);
                FreeProcInstance(lpScroll0Proc);
                DeleteObject(hpal);
                GlobalUnlock(hglb1);
                GlobalFree(hglb1);
                GlobalUnlock(hglb2);
                GlobalFree(hglb2);
                GlobalUnlock(hglb3);
                GlobalFree(hglb3);
                GlobalUnlock(handle1);
      GlobalFree(handle1);
                fclose(fp);
                fclose(fp2);
                return 0;

          default:
                return(DefWindowProc(hWnd, iMessage, wParam,
                        lParam));
          }
   //return 0;
}


void colorpalette(HWND hWnd,unsigned char *p)
```

```
{
 int i,x,y;
 unsigned char color;


                for (i=0;i<256;i++){
                        pPal->palPalEntry[i].peRed   = *(p+i*3);
                        pPal->palPalEntry[i].peGreen =  *(p+i*3+1);
                        pPal->palPalEntry[i].peBlue  =  *(p+i*3+2);
                        pPal->palPalEntry[i].peFlags = (BYTE) 0;
                }
                hpal = CreatePalette(pPal);
                LocalFree((HLOCAL) pPal);
                hDC  = GetDC(hWnd);
                SelectPalette(hDC,hpal,0);
                RealizePalette(hDC);

}

void ChangeFormat(HWND hWnd)
{
 int x,y,i,j,n;
 unsigned char color,color2,R,G,B,change[256];
 unsigned char rgb[1024];


        if((fp=fopen("temp.BMP","rb"))== NULL){
                MessageBox(hWnd,"Error Message","Error open file temp.bmp",
                MB_ICONINFORMATION|MB_OK);
           exit('F');
         }
         if((fp2=fopen("work.BMP","w+b"))== NULL){
                MessageBox(hWnd,"Error Message","Error open file work.bmp",
                MB_ICONINFORMATION|MB_OK);
           exit('F');
```

```
}

fseek(fp,77,0);
if(fread((char *)&bmp,1,sizeof(BMPHEAD),fp)==sizeof(BMPHEAD)){
        fi.width = (int)bmp.width;
        fi.depth = (int)bmp.depth;
 fi.bits  = bmp.bits;
 if (fi.bits != 8)
        {
         MessageBox(hWnd,"Error Message","Error file not 8 bit image",
         MB_ICONINFORMATION|MB_OK);
  exit(1);
        }else{
          nNumColors = 256;
        }
        // get the pallete
        n=1<<fi.bits;
 for(i=0;i<n;i++)
 {
                fi.palette[(i*RGB_SIZE)+RGB_RED]   = fgetc(fp);
   fi.palette[(i*RGB_SIZE)+RGB_GREEN] = fgetc(fp);
   fi.palette[(i*RGB_SIZE)+RGB_BLUE]  = fgetc(fp);
   fgetc(fp);
 }
}else{
        TextOut(hDC,10,10,"Error reading Heading File ..",29);
 exit(1);
}

//***** set color palette ******

//**** change PALETTE to all gray level ****//
        fseek(fp,77,0);
          for(i=0;i<256;i++){
                R = fi.palette[(i*RGB_SIZE)+RGB_RED];
                G = fi.palette[(i*RGB_SIZE)+RGB_GREEN];
```

```
                B = fi.palette[(i*RGB_SIZE)+RGB_BLUE];

                if((R==G)&&(R==B))
                        change[i] = R;
                else
                        change[i] = i;
        } // end for

        for(i=0;i<256;i++)
    {
                fi.palette[(i*RGB_SIZE)+RGB_RED]   = i;
      fi.palette[(i*RGB_SIZE)+RGB_GREEN]  = i;
                fi.palette[(i*RGB_SIZE)+RGB_BLUE]   = i;
        }
/****** write file *******/
        for(i=0;i<256;i++)
        {
          rgb[(i*4)+0] = i;
      rgb[(i*4)+1] = i;
      rgb[(i*4)+2] = i;
      rgb[(i*4)+3] = 0;
                }

       fwrite((char *)&bmp,sizeof(BMPHEAD),1,fp2);

    for(i=0;i<1024;i++)
          fputc(rgb[i],fp2);

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

   //write to new file
        fseek(fp,1078,1);    //size of BMP Header
        for (y=0;y<fi.depth;y++)
                for (x=0;x<fi.width;x++){
                        color  = fgetc(fp);
                        color2 = change[color];
```

```
                                fputc(color2,fp2);
                        }
//*******  Read to New file && Mem  *********
        fseek(fp2,0L,0);
        if(fread((char *)&bmp,1,sizeof(BMPHEAD),fp2)==sizeof(BMPHEAD)){
          fi.width = (int)bmp.width;
          fi.depth = (int)bmp.depth;
                fi.bits  = bmp.bits;

         if (fi.bits != 8)
                {
                 MessageBox(hWnd,"This File is not 8 bit Image","Error Message",
                 MB_ICONINFORMATION|MB_OK);
                 exit(1);
                }else{
                  nNumColors = 256;
                }
          // get the pallete
                n=1<<fi.bits;
          for(i=0;i<n;i++)
                {
                        fi.palette[(i*RGB_SIZE)+RGB_BLUE]  = fgetc(fp2);
             fi.palette[(i*RGB_SIZE)+RGB_GREEN] = fgetc(fp2);
             fi.palette[(i*RGB_SIZE)+RGB_RED]   = fgetc(fp2);
                        fgetc(fp2);
          }
        }else{
                TextOut(hDC,10,10,"Error reading Heading File ..",29);
          exit(1);
        }


  //***** set color palette ******


                AllocMem(hWnd);
```

```
                handle1 = GlobalAlloc(GPTR,fi.width);
                image   = GlobalLock(handle1);

                for (y=0;y<fi.depth;y++){
                        for (x=0;x<fi.width;x++){
                                color      = fgetc(fp2);
                                *(image+x) = color;
                        }
                  WriteMem(image,fi.width,y);
                }
}


void WriteBmp(HWND hWnd)
{
int i,j,x,y;
FILE *fp1;
HGLOBAL wHandle;
unsigned char far *wBuf;
unsigned char color;
unsigned char rgb[1024];


        wHandle = GlobalAlloc(GPTR,fi.width);
        wBuf    = GlobalLock(wHandle);

        if((fp1=fopen(szNameText,"w+b"))== NULL){
                MessageBox(hWnd,"Error Message","Error open file work.bmp",
                MB_ICONINFORMATION|MB_OK);
          exit('F');
        }
        for (i=0;i<256;i++){
                rgb[(i*4)+0] = i;
            rgb[(i*4)+1] = i;
            rgb[(i*4)+2] = i;
                        rgb[(i*4)+3] = 0;
```

```
        }
        fwrite((char *)&bmp,sizeof(BMPHEAD),1,fp1);

        for(i=0;i<1024;i++)
                fputc(rgb[i],fp1);

                for (y=0;y<fi.depth;y++){
    ReadMem(wBuf,fi.width,y);
                        for (x=0;x<fi.width;x++){
                                color = *(wBuf+x);
                                fputc(color,fp1);
                        }
                }
        GlobalUnlock(wHandle);
        GlobalFree(wHandle);
}


void AllocMem(HWND hWnd)
{
  long TotalByte;
  double temp;

        TotalByte = (long)fi.width*fi.depth;

        if(TotalByte > 65536L){
                RowPerPage = floor((long)65536L/fi.width);
                TotalPage  = floor((long)fi.depth/RowPerPage);
                RowRemainder = fmod(fi.depth,RowPerPage);
                if (RowRemainder > 0)
    TotalPage += 1;
        }else{
   TotalPage = 1;
        }

        switch(TotalPage){
```

```
        case 1 : hglb1   = GlobalAlloc(GPTR,(long)fi.width*fi.depth);
    if (!hglb1){
                        MessageBox(hWnd,"Allocate Memory Fail at hglb1","Error Message",
                MB_ICONINFORMATION|MB_OK);
                                exit(1);
                            }
                            buffer1 = GlobalLock(hglb1);
    break;

        case 2 : if((hglb1   = GlobalAlloc(GPTR,(long)RowPerPage*fi.width))==NULL){
                        MessageBox(hWnd,"Allocate Memory Fail at hglb1","Error Message",
                MB_ICONINFORMATION|MB_OK);
                                exit(1);
                             }
                            buffer1 = GlobalLock(hglb1);

                            if(RowRemainder == 0){
                                if((hglb2   =
GlobalAlloc(GPTR,(long)RowPerPage*fi.width))==NULL){
                        MessageBox(hWnd,"Allocate Memory Fail at hglb2","Error Message",
                MB_ICONINFORMATION|MB_OK);
                                exit(1);
                              }
                            }else{
                                hglb2   =
GlobalAlloc(GPTR,(long)RowRemainder*fi.width);
          if (!hglb2){
                        MessageBox(hWnd,"Allocate Memory Fail at hglb2","Error Message",
                MB_ICONINFORMATION|MB_OK);
                                exit(1);
                                }
                            }
                            buffer2 = GlobalLock(hglb2);

    break;
```

```
        case 3 : if((hglb1   = GlobalAlloc(GPTR,(long)RowPerPage*fi.width))==NULL){
                        MessageBox(hWnd,"Allocate Memory Fail at hglb1","Error Message",
                MB_ICONINFORMATION|MB_OK);
                        exit(1);
        }
                        buffer1 = GlobalLock(hglb1);

                        if((hglb2   = GlobalAlloc(GPTR,(long)RowPerPage*fi.width))==
NULL){
                        MessageBox(hWnd,"Allocate Memory Fail at hglb2","Error Message",
                MB_ICONINFORMATION|MB_OK);
                        exit(1);
        }
                        buffer2 = GlobalLock(hglb2);

                        if(RowRemainder == 0){
                                if((hglb3   =
GlobalAlloc(GPTR,(long)RowPerPage*fi.width))==NULL){
                        MessageBox(hWnd,"Allocate Memory Fail at hglb3","Error Message",
                MB_ICONINFORMATION|MB_OK);
                                exit(1);
                          }
                        }else{
                                if((hglb3   =
GlobalAlloc(GPTR,(long)RowRemainder*fi.width))!=NULL){
                        MessageBox(hWnd,"Allocate Memory Fail at hglb3","Error Message",
                MB_ICONINFORMATION|MB_OK);
                                exit(1);
                                }
                        }
                        buffer3 = GlobalLock(hglb3);
                        break;
        }

}
```

```
void ReadMem(unsigned char far *buffer,int BufSize,int RowOffset)
{
        int i,j;
        int BufHglb;
  int PageNo,RowAtPage;

                if(TotalPage == 1){
                        for(i=0;i<BufSize;i++)
                          *(buffer+i) =  *(buffer1+fi.width*RowOffset+i); // not exceed 64K
                }else{
                  PageNo    = floor((long)RowOffset/RowPerPage);
                  RowAtPage = (long)RowOffset - (long)RowPerPage*(PageNo-1);

                        if(RowOffset<RowPerPage){
                                for(i=0;i<BufSize;i++)
                                        *(buffer+i)=*(buffer1+BufSize*RowOffset+i);
                        }else{
                                if((RowOffset>RowPerPage)&&(RowOffset<RowPerPage*2)){
                                        for(i=0;i<BufSize;i++)
                                                *(buffer+i)= *(buffer2+BufSize*RowAtPage+i);
                                }else{
                                if((RowOffset>RowPerPage*2)&&(RowOffset<RowPerPage*3))
                                        for(i=0;i<BufSize;i++)
                                                *(buffer+i) = *(buffer3+fi.width*RowAtPage+i);
                                }
                        }
                } // end total page
}

void WriteMem(unsigned char far *buffer,int BufSize,int RowOffset)
{
        int i,j;
        int BufHglb;
        int PageNo,RowAtPage;
```

```
                if(TotalPage == 1){
                        for(i=0;i<BufSize;i++){
                                *(buffer1+BufSize*RowOffset+i)= *(buffer+i);
        }
                }else{
                        PageNo    = ceil((long)(RowOffset+1)/RowPerPage);
                        RowAtPage = RowOffset - RowPerPage*(PageNo-1);

                        if(RowOffset<RowPerPage){
                                for(i=0;i<BufSize;i++)
                                        *(buffer1+BufSize*RowOffset+i) = *(buffer+i);
                        }else{
                                if((RowOffset>RowPerPage)&&(RowOffset<RowPerPage*2)){
                                        for(i=0;i<BufSize;i++)
                                                                *(buffer2+BufSize*RowAtPage+i) = *
(buffer+i);
                                }else{
                                if((RowOffset>RowPerPage*2)&&(RowOffset<RowPerPage*3))
                                        for(i=0;i<BufSize;i++)
                                                                *(buffer3+fi.width*RowAtPage+i) = *
(buffer+i);
                                }
                        }
                } //endif TotalPage
}

BOOL FAR PASCAL OpenNewDlgProc(hDlg,message,wParam,lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
        static HWND hName;


        switch(message)
```

```
    {
        case WM_INITDIALOG :
            hName = GetDlgItem(hDlg,ID_NAME);
            SendMessage(hName,EM_LIMITTEXT,12,0L);
            SetFocus(hName);
            return(FALSE);


        case WM_COMMAND :
            switch(wParam)
            {
                case IDOK :
                    GetDlgItemText(hDlg,ID_NAME,szNameText,15);
                    EndDialog(hDlg,1);
                    return(TRUE);

                case IDCANCEL :
                    EndDialog(hDlg,0);
                    return(TRUE);

                default :
        return(FALSE);
            }
        default :
    return(FALSE);
    }

}

BOOL FAR PASCAL GraphDlgProc(hDlg,message,wParam,lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
    HWND hGraph;
```

```
PAINTSTRUCT Gps;
RECT Grect;
HDC  Ghdc;
int StartX,StartY,StrLen;
int x,y,i,j;
float hist[256],ValueMax;
unsigned int Percent[256];
unsigned char color;
HGLOBAL Hhandle;
unsigned char far *hBuf;


switch(message)
{
        case WM_INITDIALOG :
                hGraph = GetDlgItem(hDlg,ID_PAINT);
                SetFocus(hGraph);
                return(FALSE);


        case WM_COMMAND :
                switch(wParam)
                {
                        case IDOK :
                                EndDialog(hDlg,1);
                                return(TRUE);


                        default :
        return(FALSE);
                }
        case WM_PAINT :
                Ghdc = BeginPaint(hDlg,&Gps);
                Hhandle = GlobalAlloc(GPTR,fi.width);
                hBuf    = GlobalLock(Hhandle);

                for(i=0;i<256;i++)
```

```
                    hist[i]=0;

                for (y=0;y<fi.depth;y++){
                    ReadMem(hBuf,fi.width,y);
                    for (x=0;x<fi.width;x++){
                        color = *(hBuf+x);
  hist[color] += 1;
                    }
                }
ValueMax = 0;
                for(i=0;i<256;i++){ //search maximum value
                    if(hist[i]>ValueMax)
                        ValueMax = hist[i];
                }

                for(i=0;i<256;i++){
  Percent[i] = (hist[i]*200)/ValueMax;
                }
                StartX = 40;
                StartY = 250;
                MoveTo(Ghdc,StartX,StartY);
                LineTo(Ghdc,StartX+270,StartY);
                MoveTo(Ghdc,StartX,StartY);
                LineTo(Ghdc,StartX,StartY-205);
                TextOut(Ghdc,StartX-5,10,"Pixels",6);
                TextOut(Ghdc,StartX+270,StartY,"Gray Level",10);
                TextOut(Ghdc,StartX,StartY+12,"0",1);
TextOut(Ghdc,StartX+120,StartY+12,"120",3);
                TextOut(Ghdc,StartX+250,StartY+12,"250",3);
                StartX += 2;
                for (i=0;i<=25;i++){        // Scale Graph
                    MoveTo(Ghdc,StartX+(i*10),StartY);
  if(i==0 || i==12 || i==25)
                        LineTo(Ghdc,StartX+(i*10),StartY+10);
  else
                        LineTo(Ghdc,StartX+(i*10),StartY+5);
```

```
                    }

                    MoveTo(Ghdc,StartX,StartY);
                    for(i=0;i<256;i++){
                            MoveTo(Ghdc,StartX+i,StartY);
                            LineTo(Ghdc,StartX+i,StartY-Percent[i]);
                    }

                    EndPaint(hDlg,&Gps);
                    GlobalUnlock(Hhandle);
                    GlobalFree(Hhandle);
        return(FALSE);
                    default :
          return(FALSE);
          }

}

BOOL FAR PASCAL ZoomDlgProc(hDlg,message,wParam,lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
        HWND hZoom,hWnd;
        PAINTSTRUCT Gps;
        RECT Grect;
        HDC  Ghdc;
        int x,y,i,j,count;
   float value,x1,x2,x3,x4,x5,x6,x7,x8,x9;
        unsigned char color;
        HGLOBAL Hhandle,Ihandle1,Ihandle2,Ihandle3;
        unsigned char far *Row1;
        unsigned char far *Row2;
        unsigned char far *Row3;
        unsigned char far *Lbuf2;
```

```
        HGLOBAL Lhandle;
        unsigned char far *Lbuf;
BOOL FlagRead = FALSE;
        BOOL FlagRead2 = FALSE;

        Lhandle  = GlobalAlloc(GPTR,fi.width);
        Lbuf     = GlobalLock(Lhandle);
        Hhandle  = GlobalAlloc(GPTR,fi.width*2);
        Lbuf2    = GlobalLock(Hhandle);
        Ihandle1 = GlobalAlloc(GPTR,fi.width*2);
Row1    = GlobalLock(Ihandle1);
        Ihandle2 = GlobalAlloc(GPTR,fi.width*2);
        Row2     = GlobalLock(Ihandle2);
        Ihandle3 = GlobalAlloc(GPTR,fi.width*2);
        Row3     = GlobalLock(Ihandle3);

        switch(message)
        {
                case WM_INITDIALOG :
                        hZoom = GetDlgItem(hDlg,ID_PAINT);
                        SetFocus(hZoom);
                        return(FALSE);


                case WM_COMMAND :
                        switch(wParam)
                        {
                                case IDOK :
                                        EndDialog(hDlg,1);
                                        return(TRUE);

                                default :
                return(FALSE);
                        }
                case WM_PAINT :
                        Ghdc = BeginPaint(hDlg,&PtStr);
```

```
            pPal = (LOGPALETTE*) LocalAlloc(LPTR,sizeof(LOGPALETTE)+nNumColors*sizeof
(PALETTEENTRY));
            if (!pPal){
                  TextOut(Ghdc,10,10,"Allocate pPal Failed ",21);
      exit(1);
            }
            pPal->palNumEntries = nNumColors;
    pPal->palVersion = 0x300;
            for (i=0;i<nNumColors;i++){
                        pPal->palPalEntry[i].peRed   = fi.palette[(i*3)];
                        pPal->palPalEntry[i].peGreen = fi.palette[(i*3)+1];
                        pPal->palPalEntry[i].peBlue  = fi.palette[(i*3)+2];
                        pPal->palPalEntry[i].peFlags = (BYTE) 0;
            }
            hpal = CreatePalette(pPal);
            Ghdc  = GetDC(hWnd);
            SelectPalette(Ghdc,hpal,0);
            RealizePalette(Ghdc);

           ReadMem(Lbuf,fi.width,0); //Read line 0

            for (i=0;i<fi.width*2;i++){ // init 3 Rows to Zoom
     if (FlagRead==FALSE){
                        *(Row1+i) = *(Lbuf+(i/2));
                        FlagRead  = TRUE;
                  }else{
                        *(Row1+i) = 0;
       FlagRead  = FALSE;
                  }
                  *(Row2+i)      = 0;
            }

            ReadMem(Lbuf,fi.width,1);   // Read line 1
            FlagRead = FALSE;
            for (i=0;i<fi.width*2;i++){
```

```
if (FlagRead == FALSE){
                        *(Row3+i) = *(Lbuf+(i/2));
                        FlagRead  = TRUE;
                }else{
*(Row3+i) = 0;
FlagRead  = FALSE;
                }

        }

        for(x=1;x<(fi.width*2)-1;x++){ // Calculate first line
                x4 = *(Row1+(x-1));
                x5 = *(Row1+x);
                x6 = *(Row1+(x+1));
                value = x4/2 + x5 + x6/2;
                SetPixel(Ghdc,x,fi.depth*2,PALETTEINDEX(value));
        }
        for(x=1;x<(fi.width*2)-1;x++){ //Calculate for second line
                        x1 = *(Row1+(x-1));
                        x2 = *(Row1+x);
                        x3 = *(Row1+(x+1));
                        x4 = *(Row2+(x-1));
                        x5 = *(Row2+x);
                        x6 = *(Row2+(x+1));
                        x7 = *(Row3+(x-1));
                        x8 = *(Row3+x);
                        x9 = *(Row3+(x+1));
                        value = x1/4 + x2/2 + x3/4 + x4/2 + x5 + x6/2 + x7/4 + x8/2 + x9/4;
                        if(value>255)
                                value = 255;
                        else
                                if(value<0)
                                        value = 0;
                        SetPixel(Ghdc,x,(fi.depth*2)-1,PALETTEINDEX(value));
        }
```

```
//********** Main Loop for Calculate *****************
        count = 1;
        FlagRead  = FALSE;

        for(y=2;y<(fi.depth*2)-1;y++){      // Main Loop
                if (FlagRead == FALSE){    // Read Memory Image & Insert Line 0
                        for(i=0;i<fi.width*2;i++)
*(Lbuf2+i) = 0;
       FlagRead  = TRUE;
                }else{
                        count += 1;
                        FlagRead2 = FALSE;
                        ReadMem(Lbuf,fi.width,count);
                        for(i=0;i<fi.width*2;i++){
                                if(FlagRead2 == FALSE){
                                        *(Lbuf2+i) = *(Lbuf+(i/2));
                                        FlagRead2  = TRUE;
                                }else{
                                        *(Lbuf2+i) = 0;
   FlagRead2 = FALSE;
                                }
                        }

FlagRead = FALSE;
                }

                for(x=0;x<fi.width*2;x++){     //Scroll Line up
                        *(Row1+x)         = *(Row2+x);
                        *(Row2+x)         = *(Row3+x);
*(Row3+x)   = *(Lbuf2+x);
                }

                for(x=1;x<(fi.width*2)-1;x++){         //Loop for putpixel 1 Line
                        x1 = *(Row1+(x-1));
                        x2 = *(Row1+x);
```

```
                    x3 = *(Row1+(x+1));
                    x4 = *(Row2+(x-1));
                    x5 = *(Row2+x);
                    x6 = *(Row2+(x+1));
                    x7 = *(Row3+(x-1));
                    x8 = *(Row3+x);
                    x9 = *(Row3+(x+1));
                    value = x1/4 + x2/2 + x3/4 + x4/2 + x5 + x6/2 + x7/4 + x8/2 + x9/4;
                    if(value>255)
                            value = 255;
                    else
                            if(value<0)
                                    value = 0;
                    SetPixel(Ghdc,x,(fi.depth*2)-y,PALETTEINDEX(value));
                //  SetPixel(Ghdc,x,(fi.depth*2)-y,PALETTEINDEX(x5));
                }

        }

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

                EndPaint(hDlg,&Gps);

                GlobalUnlock(Lhandle);
                GlobalFree(Lhandle);
                GlobalUnlock(Hhandle);
                GlobalFree(Hhandle);
                GlobalUnlock(Ihandle1);
                GlobalFree(Ihandle1);
                GlobalUnlock(Ihandle2);
                GlobalFree(Ihandle2);
                GlobalUnlock(Ihandle3);
                GlobalFree(Ihandle3);

return(FALSE);
```

```
            default :
        return(FALSE);
        }

}

BOOL FAR PASCAL ScrollDlgProc(hDlg,message,wParam,lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
HWND hScroll;

        switch(message)
        {
                case WM_INITDIALOG :
                  hScroll = GetDlgItem(hDlg,ID_SCRL);
      SetFocus(hScroll);
                  SetScrollRange(hScroll,SB_CTL,0,200,TRUE);
                  SetScrollPos(hScroll,SB_CTL,100,TRUE);
                  return(FALSE);

                case WM_HSCROLL :
                        SetFocus(HIWORD(lParam));
                        if(wParam != SB_THUMBTRACK){
        Position = (BYTE)GetScrollPos(HIWORD(lParam),SB_CTL);
        switch(wParam)
                                {
                                        case SB_LINEDOWN :
                                                Position += 10;
                                                break;
                                        case SB_LINEUP :
                                                Position -= 10;
                                                break;
                                        case SB_PAGEDOWN :
```

```
                                        Position += 10;
                                        break;
                                case SB_PAGEUP:
                                        Position -= 10;
                                        break;
                                case SB_THUMBPOSITION : // current position is specify by the
nPos
                                        Position = (BYTE)LOWORD(lParam);
                        }
                        SetScrollPos(HIWORD(lParam),SB_CTL,Position,TRUE);
        }
        return(TRUE);

                case WM_COMMAND :

                        switch(wParam)
                        {
                        case IDOK :
        EndDialog(hDlg,1);
        return(TRUE);

                        case IDCANCEL :
                                EndDialog(hDlg,0);
                                return(FALSE);

                        default :
                                return(FALSE);
                        }
                default :
        return(FALSE);
        }

}
```

```
void LoadImage(void)
{
        HGLOBAL Lhandle;
        unsigned char far *Lbuf;
  unsigned char color;
  int x,y;

                Lhandle = GlobalAlloc(GPTR,fi.width);
     Lbuf    = GlobalLock(Lhandle);

                fseek(fp2,1078,0);

                for (y=0;y<fi.depth;y++){
                        for (x=0;x<fi.width;x++){
                                color      = fgetc(fp2);
                                //SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(color));
                                *(Lbuf+x) = color;
                        }
                  WriteMem(Lbuf,fi.width,y);
                }

                for (y=0;y<fi.depth;y++){
                        ReadMem(Lbuf,fi.width,y);
                        for (x=0;x<fi.width;x++){
          color = *(Lbuf+x);
                                SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(color));
                        }
                }

                GlobalUnlock(Lhandle);
     GlobalFree(Lhandle);
}

void Brightness(int count)
{
int x,y,temp;
```

```
unsigned char color;
int array[768];
HGLOBAL Bhandle,Dhandle;
unsigned char far *bright;
unsigned char far *MemBuf;
char ch;
int num;


        num = count - 100;
        Bhandle = GlobalAlloc(GPTR,fi.width);
        bright  = GlobalLock(Bhandle);
        Dhandle = GlobalAlloc(GPTR,fi.width);
        MemBuf  = GlobalLock(Dhandle);

        for (y=0;y<fi.depth;y++){
    ReadMem(bright,fi.width,y);
            for (x=0;x<fi.width;x++){
                color     = *(bright+x);
                temp      = color+num;
                temp      = max(temp,0.0);
                temp      = min(temp,255.0);
              *(MemBuf+x) = temp;
                SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(temp));
            }
            WriteMem(MemBuf,fi.width,y);
        }
      GlobalUnlock(Bhandle);
      GlobalFree(Bhandle);
      GlobalUnlock(Dhandle);
      GlobalFree(Dhandle);
}

void Contrast(HWND hWnd,int count,unsigned char *p)
{
int x,y,temp;
```

```
int i,j,step;
HGLOBAL Chandle,Dhandle;
unsigned char far *Cbuf;
unsigned char far *MemBuf;
unsigned char color[768];
int icolor;
unsigned char tcolor;
int array[768];
char ch;
float num=1;

                Chandle = GlobalAlloc(GPTR,fi.width);
                Cbuf    = GlobalLock(Chandle);
                Dhandle = GlobalAlloc(GPTR,fi.width);
    MemBuf  = GlobalLock(Dhandle);

                count = count/4;
                count = count - 25;      //- center scroll position
                if (count >= 0){
      for(i=0;i<count;i++)
                        num *= 1.1;
                }else{
                        step = abs(count);
      for(i=0;i<step;i++)
        num *= 0.9;
                }
                // MessageBox(hWnd,"00","Message",
                //                                  MB_ICONINFORMATION|MB_OK);

                for(i=0;i<768;i++)
        array[i] = *(p++);

                for(i=0;i<768;i++){
                        array[i] = (array[i]-128)*num+128;
      temp     = max(array[i],0.0);
                        temp     = min(temp,255.0);
```

```
                color[i] = temp;
            }

            for (y=0;y<fi.depth;y++){
        ReadMem(Cbuf,fi.width,y);
                for (x=0;x<fi.width;x++){
                        icolor    = *(Cbuf+x);
                        tcolor    = color[icolor*3];
            *(MemBuf+x) = tcolor;
                        SetPixel(hDC,x,fi.depth-y+1,PALETTEINDEX(tcolor));
                 }
                 WriteMem(MemBuf,fi.width,y);
            }

            GlobalUnlock(Chandle);
            GlobalFree(Chandle);
            GlobalUnlock(Dhandle);
            GlobalFree(Dhandle);
}

void AutoContrast(void)
{
 int i,j,x,y;
 float Xmin,Xmax,Ymin,Ymax;
 float TempMin,TempMax,hist[256];
 float Temp;
 double PercentCut;
 float Slope;
 unsigned char color;
 HGLOBAL Hhandle,Dhandle;
 unsigned char far *hBuf;
 unsigned char far *MemBuf;
 int LookTable[256];
 int ii;
 BOOL Flag=FALSE;
```

```
                Hhandle = GlobalAlloc(GPTR,fi.width);
                hBuf    = GlobalLock(Hhandle);
                Dhandle = GlobalAlloc(GPTR,fi.width);
                MemBuf  = GlobalLock(Dhandle);

                for(i=0;i<256;i++){
                        hist[i]=0;
  LookTable[i]=0;
}
                for (y=0;y<fi.depth;y++){
                        ReadMem(hBuf,fi.width,y);
                        for (x=0;x<fi.width;x++){
                                color = *(hBuf+x);
   hist[color] += 1;
                        }
                }
                TempMin = 0;
                TempMax = 0;
                Xmin = 0;
                Xmax = 255;
                Ymin = 0;
                Ymax = 255;
                PercentCut = fi.width*0.01*fi.depth;    // set to 1%
ii=0;
                do{
                        TempMax += hist[255-ii];
                        if(TempMax > PercentCut){
                                Xmax = 255-ii;
        Flag =TRUE;
                        }
  ii++;

                }while (Flag==FALSE);
                Flag=FALSE;
ii=0;
```

```
                do{
                        TempMin += hist[ii];
                        if(TempMin > PercentCut){
                                Xmin = ii;
                                Flag = TRUE;
                        }
                        ii++;
                }while (Flag==FALSE);

                Slope = (Ymax-Ymin)/(Xmax-Xmin);

                for(i=0;i<256;i++){
                  if( i < Xmin )
                                LookTable[i] = 0;
else
                  if ((i >= Xmin)&&(i <= Xmax)){
                                Temp = (i - Xmin)*Slope;
                                if(Temp<0)
                                        Temp=0;
    if(Temp>255)
      Temp=255;
                                LookTable[i] = Temp;
}
                   else
    if(i>Xmax)
                                        LookTable[i] =255 ;

  //IndexArr++;
                }

                for (y=0;y<fi.depth;y++){
ReadMem(hBuf,fi.width,y);
                for (x=0;x<fi.width;x++){
          color     = *(hBuf+x);
                                *(MemBuf+x)= LookTable[color];
                                color = LookTable[color];
```

```
                                        SetPixel(hDC,x,fi.depth-y+1,PALETTEINDEX(color));
                        }
                        WriteMem(MemBuf,fi.width,y);
                }

        GlobalUnlock(Hhandle);
        GlobalFree(Hhandle);
        GlobalUnlock(Dhandle);
        GlobalFree(Dhandle);
}

void Sharp(void)
{
HGLOBAL hglb,Shandle,Dhandle;
unsigned char far *sharp;
unsigned char far *Sbuf;
unsigned char far *MemBuf;
int x,y,temp;
int i,j,step;
unsigned char color;
float value;
int icolor;
char ch;


                hglb    = GlobalAlloc(GPTR,(long)fi.width*3); // allocate 3 row
                sharp   = (unsigned char far *)GlobalLock(hglb);
                Shandle = GlobalAlloc(GPTR,fi.width);
                Sbuf    = GlobalLock(Shandle);
                Dhandle = GlobalAlloc(GPTR,fi.width);
                MemBuf  = GlobalLock(Dhandle);


                for(j=0;j<3;j++){
        ReadMem(Sbuf,fi.width,j);
                  for(i=0;i<fi.width;i++)      // read first 3 rows using 3x3
```

```
                    //*(sharp+i) = fgetc(fp2);
        *(sharp+fi.width*j+i) = *(Sbuf+i);
    }


        // write first 2 line
        for(x=0;x<fi.width;x++){        // write first line to screen
            SetPixel(hDC,x,fi.depth,PALETTEINDEX(*(sharp+x)));
            value =(*(sharp+fi.width*1+x))*5 - (*(sharp+fi.width*2+x)+
                                (*(sharp+fi.width*0+x))  + (*(sharp+fi.width*1+x+1))+
                                (*(sharp+fi.width*1+x-1)));
            SetPixel(hDC,x,fi.depth-1,PALETTEINDEX(value));
    }
////////////////////////////////////////////////////////////////


        for(y=2;y<fi.depth-1;y++){    // Calculate Sharpening loop
    ReadMem(Sbuf,fi.width,y);
            for(i=0;i<fi.width;i++){
                *(sharp+i)                              = *(sharp+fi.width*1+i);  //
row 0 = row 1
                *(sharp+fi.width*1+i) = *(sharp+fi.width*2+i);// row 1 = row 2
                //*(sharp+fi.width*2+i) = fgetc(fp2);
        *(sharp+fi.width*2+i) = *(Sbuf+i);
            }

            for(x=1;x<fi.width;x++){
                value =(*(sharp+fi.width*1+x))*5 - (*(sharp+fi.width*2+x)+
                                (*(sharp+fi.width*0+x))  + (*(sharp+fi.width*1+x+1))+
                                (*(sharp+fi.width*1+x-1)));
                if(value <0)
                    value = 0;
                else if (value>255)
                    value = 255;

        *(MemBuf+x) = value;
                SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(value));
```

```
                    }
                    WriteMem(MemBuf,fi.width,y);
                }
                    ////////////////////////////////////////// Scroll Row up


            GlobalUnlock(hglb);
            GlobalFree(hglb);
            GlobalUnlock(Shandle);
            GlobalFree(Shandle);
            GlobalUnlock(Dhandle);
    GlobalFree(Dhandle);
}

void Smooth(void)
{
HGLOBAL hglb,hglb1,Smhandle,Dhandle;
unsigned char far *smooth;
unsigned char far *Smbuf;
unsigned char far *MemBuf;
int x,y,temp;
int i,j,step;
unsigned char color;

float value;
int icolor;
char ch;


            hglb     = GlobalAlloc(GPTR,(long)fi.width*3); // allocate 1 row
            smooth   = (unsigned char far *)GlobalLock(hglb);
            Smhandle = GlobalAlloc(GPTR,fi.width);
            Smbuf    = GlobalLock(Smhandle);
            Dhandle  = GlobalAlloc(GPTR,(long)fi.width);
            MemBuf = GlobalLock(Dhandle);
```

```
            //fseek(fp2,sizeof(BMPHEAD)+1024,0);
            for(j=0;j<3;j++){
        ReadMem(Smbuf,fi.width,j);
                    for(i=0;i<fi.width;i++)        // read first 3 rows using 3x3
        *(smooth+fi.width*j+i) = *(Smbuf+i);
                        //*(smooth+i) = fgetc(fp2);
    }
            // write first 2 line
            for(x=0;x<fi.width;x++){          // write first line to screen
                SetPixel(hDC,x,fi.depth,PALETTEINDEX(*(smooth+x)));
                value =(*(smooth+fi.width*1+x))*5 - (*(smooth+fi.width*2+x)+
                                        (*(smooth+fi.width*0+x))   +
(*(smooth+fi.width*1+x+1))+
                                        (*(smooth+fi.width*1+x-1)));
                SetPixel(hDC,x,fi.depth-1,PALETTEINDEX(value));
    }
////////////////////// using average ////////////////////////////////////

            for(y=2;y<fi.depth-1;y++){                  // Calculate Smoothing loop
       ReadMem(Smbuf,fi.width,y);
                    for(i=0;i<fi.width;i++){
                        *(smooth+i)                         = *(smooth+fi.width*1+i);   // row 0
= row 1
                        *(smooth+fi.width*1+i) = *(smooth+fi.width*2+i);       // row 1 = row 2
        *(smooth+fi.width*2+i) = *(Smbuf+i);
                        //*(smooth+fi.width*2+i) = fgetc(fp2);
                    }

                    for(x=1;x<fi.width-1;x++){
                        value =*(smooth+x-1)          + *(smooth+x) +
                                    *(smooth+x+1)          + *(smooth+fi.width+x-1)+
                                    *(smooth+fi.width+x)   + *(smooth+fi.width+x+1)+
                                    *(smooth+fi.width*2+x-1)  + *(smooth+fi.width*2+x)+
                                    *(smooth+fi.width*2+x+1);
```

```
                    value = value/9;
                    if (value<0)
                        value = 0;
                    else if (value > 255)
                                        value = 255;
        *(MemBuf+x) = value;
                    SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(value));
                }
      WriteMem(MemBuf,fi.width,y);
            }
                ////////////////////////////////////////// Scroll Row up
          GlobalUnlock(hglb);
          GlobalFree(hglb);
          GlobalUnlock(Smhandle);
          GlobalFree(Smhandle);
          GlobalUnlock(Dhandle);
          GlobalFree(Dhandle);
}

void Weight(void)
{
HGLOBAL hglb,hglb1,Smhandle,Dhandle;
unsigned char far *smooth;
unsigned char far *Smbuf;
unsigned char far *MemBuf;
int x,y,temp;
int i,j,step;
unsigned char color;

float value;
float x1,x2,x3,x4,x5,x6,x7,x8,x9;
int icolor;
char ch;


          hglb    = GlobalAlloc(GPTR,(long)fi.width*3); // allocate 1 row
```

```
        smooth   = (unsigned char far *)GlobalLock(hglb);
        Smhandle = GlobalAlloc(GPTR,fi.width);
        Smbuf    = GlobalLock(Smhandle);
        Dhandle  = GlobalAlloc(GPTR,(long)fi.width);
        MemBuf = GlobalLock(Dhandle);



        //fseek(fp2,sizeof(BMPHEAD)+1024,0);
        for(j=0;j<3;j++){
    ReadMem(Smbuf,fi.width,j);
                for(i=0;i<fi.width;i++)        // read first 3 rows using 3x3
    *(smooth+fi.width*j+i) = *(Smbuf+i);
  }
        // write first 2 line
        for(x=0;x<fi.width;x++){          // write first line to screen
                SetPixel(hDC,x,fi.depth,PALETTEINDEX(*(smooth+x)));
                value =(*(smooth+fi.width*1+x))*5 - (*(smooth+fi.width*2+x)+
                                        (*(smooth+fi.width*0+x))   +
(*(smooth+fi.width*1+x+1))+
                                        (*(smooth+fi.width*1+x-1)));
                SetPixel(hDC,x,fi.depth-1,PALETTEINDEX(value));
  }
////////////////////// using weight ////////////////////////////////////

        for(y=2;y<fi.depth-1;y++){                  // Calculate Smoothing loop
    ReadMem(Smbuf,fi.width,y);
                for(i=0;i<fi.width;i++){
                        *(smooth+i)                         = *(smooth+fi.width*1+i);   // row 0
= row 1
                        *(smooth+fi.width*1+i) = *(smooth+fi.width*2+i);       // row 1 = row 2
    *(smooth+fi.width*2+i) = *(Smbuf+i);
                }

                for(x=1;x<fi.width-1;x++){
                        x1 = *(smooth+x-1);
                        x2 = *(smooth+x);
```

```
                        x3 = *(smooth+x+1);
                        x4 = *(smooth+fi.width+x-1);
                        x5 = *(smooth+fi.width+x);
                        x6 = *(smooth+fi.width+x+1);
                        x7 = *(smooth+fi.width*2+x-1);
                        x8 = *(smooth+fi.width*2+x);
                        x9 = *(smooth+fi.width*2+x+1);
                        value = (x1/16)+(x2/8)+(x3/16)+(x4/8)+(x5/4)+(x6/8)+(x7/16)+(x8/8)+
(x9/16);

                        if (value<0)
                                value = 0;
                        else if (value > 255)
                                                value = 255;
        *(MemBuf+x) = value;
                        SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(value));
                    }
      WriteMem(MemBuf,fi.width,y);
                }
                    ////////////////////////////////////////// Scroll Row up
            GlobalUnlock(hglb);
            GlobalFree(hglb);
            GlobalUnlock(Smhandle);
            GlobalFree(Smhandle);
            GlobalUnlock(Dhandle);
            GlobalFree(Dhandle);
}


void Edge(void)
{
HGLOBAL hglb,Ehandle,Dhandle;
unsigned char far *edge;
unsigned char far *Ebuf;
unsigned char far *MemBuf;
int x,y,temp;
```

```
int i,j,step;
unsigned char color;

float value;
int icolor;
char ch;


        hglb    = GlobalAlloc(GPTR,(long)fi.width*3); // allocate 1 row
        edge    = (unsigned char far *)GlobalLock(hglb);
        Ehandle  = GlobalAlloc(GPTR,fi.width);
        Ebuf       = GlobalLock(Ehandle);
        Dhandle  = GlobalAlloc(GPTR,(long)fi.width);
        MemBuf = GlobalLock(Dhandle);

         for (j=0;j<3;j++){
    ReadMem(Ebuf,fi.width,j);
                for(i=0;i<fi.width;i++)       // read first 3 rows using 3x3
                        *(edge+fi.width*j+i) = *(Ebuf+i);
                        //*(edge+i) = fgetc(fp2);
    }
        // write first 2 line
        for(x=0;x<fi.width;x++){          // write first line to screen
                SetPixel(hDC,x,fi.depth,PALETTEINDEX(*(edge+x)));
                value =(*(edge+fi.width*1+x))*5 - (*(edge+fi.width*2+x)+
                                        (*(edge+fi.width*0+x))   + (*(edge+fi.width*1+x+1))+
                                        (*(edge+fi.width*1+x-1)));
                SetPixel(hDC,x,fi.depth-1,PALETTEINDEX(value));
    }
////////////////////////////////////////////////////////////////////

        for(y=2;y<fi.depth-1;y++){
                ReadMem(Ebuf,fi.width,y);             // Calculate Edge loop
                for(i=0;i<fi.width;i++){
                        *(edge+i)                                   = *(edge+fi.width*1+i);   //
row 0 = row 1
```

```
                    *(edge+fi.width*1+i) = *(edge+fi.width*2+i);  // row 1 = row 2
                    *(edge+fi.width*2+i) = *(Ebuf+i);
                }

                for(x=1;x<fi.width-1;x++){
                    value = abs(*(edge+x+1)             + *(edge+fi.width+x+1)*2
                                                + *(edge+fi.width*2+x+1)    - *(edge+x-1)
                                                - *(edge+fi.width*1+x-1)*2   - *
(edge+fi.width*2+x-1))
                                            +abs(*(edge+fi.width*2+x-1)    + *(edge+fi.width*2+x)
*2
                                                + *(edge+fi.width*2+x+1)    - *(edge+x-1)
                                                - *(edge+x)*2               - *(edge+x+1));

                    *(MemBuf+x) = value;
                    SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(value));
                }
        WriteMem(MemBuf,fi.width,y);
            }
                ////////////////////////////////////////// Scroll Row up
            GlobalUnlock(hglb);
            GlobalFree(hglb);
            GlobalUnlock(Ehandle);
            GlobalFree(Ehandle);
            GlobalUnlock(Dhandle);
    GlobalFree(Dhandle);
}



void HistEqualize(void)
{

HGLOBAL hglb11,hglb22,hglb33,hglb55,Hhandle;
unsigned char far *Hbuf;
unsigned char far *MemBuf;
```

```
int i,x,y;
char ch;
unsigned char color;
long int far *hist;
unsigned char far *table;
long int far *Cumulative;
int temp,temp2;

double temp11,TotalPixel;
float temp1;


        hglb11  = GlobalAlloc(GPTR,256*sizeof(long int));
                hist    = (long int far *)GlobalLock(hglb11);
                hglb33  = GlobalAlloc(GPTR,256*sizeof(long int));
                Cumulative  = (long int far *)GlobalLock(hglb33);
    hglb55  = GlobalAlloc(GPTR,256*sizeof(unsigned char));
                table   = (unsigned char far *)GlobalLock(hglb55);
                Hhandle = GlobalAlloc(GPTR,fi.width);
                Hbuf    = GlobalLock(Hhandle);
                hglb22  = GlobalAlloc(GPTR,fi.width);
                MemBuf  = GlobalLock(hglb22);
        /************* HISTOGRAM EQUALIZATION *************/

        TotalPixel = (long)fi.width*(long)fi.depth;
        for(i=0;i<256;i++)
                *(hist+i) = 0;



        for(y=0;y<fi.depth;y++){    // create histogram
        ReadMem(Hbuf,fi.width,y);
                for(x=0;x<fi.width;x++){
                        temp        = *(Hbuf+x);
                        hist[temp] += 1;
                }
        }
```

```
temp11 = 0;
        for(i=0;i<256;i++){
        temp11 += *(hist+i);
                *(Cumulative+i) = temp11;
        }

        for(i=0;i<256;i++){
                        //temp1 = *(hist+i);
                        temp11 =        ceil(*(Cumulative+i)*256/TotalPixel)-1;
                        if(temp11>255)
                                temp11 = 255;
                        else
                                if (temp11<0)
                                        temp11 = 0;
                        *(table+i) = temp11;
        }



        for(y=0;y<fi.depth;y++){
  ReadMem(Hbuf,fi.width,y);
                for(x=0;x<fi.width;x++){
                        temp  = *(Hbuf+x);
                        temp2 = *(table+temp);
                        *(MemBuf+x) = temp2;
                        SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(temp2));
                }
                WriteMem(MemBuf,fi.width,y);
        }

      GlobalUnlock(hglb11);
      GlobalFree(hglb11);
      GlobalUnlock(hglb22);
      GlobalFree(hglb22);
      GlobalUnlock(hglb33);
      GlobalFree(hglb33);
```

```
        GlobalUnlock(Hhandle);
        GlobalFree(Hhandle);
        GlobalUnlock(hglb55);
        GlobalFree(hglb55);
}
/*
void AdaptiveHist(void)
{
HGLOBAL hglb11,hglb22,hglb33,hglb55,Hhandle;
unsigned char far *Hbuf;
unsigned char far *MemBuf;
int i,J,x,y,RectSizeX,RectSizeY,XStart,YStart;
char ch;
unsigned char color;
long int far *hist;
unsigned char far *table;
long int far *Cumulative;
int temp,temp2;

double temp11,TotalPixel;
float temp1;

            Hhandle = GlobalAlloc(GPTR,fi.width);
            Hbuf    = GlobalLock(Hhandle);
            hglb22  = GlobalAlloc(GPTR,fi.width);
            MemBuf  = GlobalLock(hglb22);
          hglb11  = GlobalAlloc(GPTR,4096*sizeof(long int)); //256*4
            hist    = (long int far *)GlobalLock(hglb11);

            hglb33  = GlobalAlloc(GPTR,256*sizeof(long int));
            Cumulative  = (long int far *)GlobalLock(hglb33);
    hglb55  = GlobalAlloc(GPTR,256*sizeof(unsigned char));
            table   = (unsigned char far *)GlobalLock(hglb55);

        //********* ADAPTIVE HISTOGRAM EQUALIZATION *********
```

```
        TotalPixel = (long)fi.width*(long)fi.depth;
        for(i=0;i<4096;i++)
                *(hist+i) = 0;


        for(y=0;y<fi.depth;y++){    // create histogram
        ReadMem(Hbuf,fi.width,y);
                for(x=0;x<fi.width;x++){
                        temp        = *(Hbuf+x);
                        hist[temp] += 1;
                }
        }
temp11 = 0;
        for(i=0;i<256;i++){
       temp11 += *(hist+i);
                *(Cumulative+i) = temp11;
        }


        for(i=0;i<256;i++){
                        //temp1 = *(hist+i);
                        temp11 =            ceil(*(Cumulative+i)*256/TotalPixel)-1;
                        if(temp11>255)
                                temp11 = 255;
                        else
                                if (temp11<0)
                                        temp11 = 0;
                        *(table+i) = temp11;
        }


        for(y=0;y<fi.depth;y++){
  ReadMem(Hbuf,fi.width,y);
                for(x=0;x<fi.width;x++){
                        temp  = *(Hbuf+x);
                        temp2 = *(table+temp);
```

```
                    *(MemBuf+x) = temp2;
                    SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(temp2));
                }
                WriteMem(MemBuf,fi.width,y);
            }

        GlobalUnlock(hglb11);
        GlobalFree(hglb11);
        GlobalUnlock(hglb22);
        GlobalFree(hglb22);
        GlobalUnlock(hglb33);
        GlobalFree(hglb33);
        GlobalUnlock(Hhandle);
        GlobalFree(Hhandle);
        GlobalUnlock(hglb55);
        GlobalFree(hglb55);

}
*/
int Entropy(void)
{
        double H=0,alpha,fealpha,max1pi,max2pi;
        double old_fe=0,lbmax1pi,lbmax2pi,lb1pi,lb2pi;
        double temp,temp2,temp3,temp4;
        double TotalPixel;
  double Tprob,lg2;
        int x,y,i,s,thr1;
  unsigned char color;
        int far *hist;
        double far *Prob;
        double far *lb;
        HGLOBAL Hhandle,hglb11,Hprob,Hlb;
  unsigned char far *hBuf;


  Hhandle = GlobalAlloc(GPTR,fi.width);
```

```
hBuf    = GlobalLock(Hhandle);
        hglb11  = GlobalAlloc(GPTR,256*sizeof(int));
        hist    = (int far *)GlobalLock(hglb11);
        Hprob   = GlobalAlloc(GPTR,256*sizeof(double));
        Prob    = (double far *)GlobalLock(Hprob);
        Hlb         = GlobalAlloc(GPTR,256*sizeof(double));
        lb              = (double far *)GlobalLock(Hlb);

for(i=0;i<256;i++){
                        *(hist+i)=0;
                        *(Prob+i)=0;
    lb[i]=0;
}

        for (y=0;y<fi.depth;y++){
                ReadMem(hBuf,fi.width,y);
                for (x=0;x<fi.width;x++){
                        color = *(hBuf+x);
    hist[color] += 1;
                }
        }

        TotalPixel = (long)fi.width*(long)fi.depth;
        lg2 = log10(2);

        for(i=0;i<256;i++){
          *(Prob+i) = *(hist+i);
          *(Prob+i) = *(Prob+i)/TotalPixel;
          Tprob = *(Prob+i);
  if (Tprob > 0)
                        *(lb+i)   = log10(Tprob)/lg2;
          else
           *(lb+i)= 0;
          H += -(*(Prob+i)*(*(lb+i)));
        }
```

```
alpha = 0.01;
//      for(alpha = 0.01;alpha<1.00001;alpha=alpha+0.01){
do{
        temp = -(alpha*H);
                temp2=0.0;
                for(i=0;i<256;i++){
                        temp2 += *(Prob+i)*(*(lb+i));
                        if(temp2<=temp){
                                s = i;          // Find s
        break;
                        }
                }
                temp3=0.0;
                max1pi = *(Prob+0);
                for(i=1;i<s;i++){
                        temp3 += *(Prob+i);
                        if(*(Prob+i) > max1pi)
        max1pi = *(Prob+i);
                }
                temp4=0.0;
                max2pi = *(Prob+s+1);
                for(i=s+1;i<256;i++){
                        temp4 += *(Prob+i);
                        if(*(Prob+i)>max2pi)
        max2pi = *(Prob+i);
                }
                if(max1pi!=0 && max2pi!=0 && temp3!=0 && temp4!=0){
                        lbmax1pi = log10(max1pi)/lg2;
                        lbmax2pi = log10(max2pi)/lg2;
                        lb1pi           = log10(temp3)/lg2;
                        lb2pi           = log10(temp4)/lg2;
                        fealpha  = ((alpha*lb1pi)/lbmax1pi)+(((1-alpha)*lb2pi)/lbmax2pi);
                        if(fealpha >= old_fe){
                                thr1 = s;
        old_fe = fealpha;
                        }
```

```
                }
                alpha = alpha + 0.01;
  }while(alpha <= 1.000);
        //}// end loop alpha
        GlobalUnlock(Hhandle);
        GlobalFree(Hhandle);
        GlobalUnlock(hglb11);
        GlobalFree(hglb11);
        GlobalUnlock(Hprob);
        GlobalFree(Hprob);
        GlobalUnlock(Hlb);
        GlobalFree(Hlb);

    return(thr1);
}

void Write2Level(void)
{
int x,y,Threshold;
unsigned char color;
HGLOBAL Hhandle,Dhandle;
unsigned char far *hBuf;
unsigned char far *MemBuf;


        Hhandle = GlobalAlloc(GPTR,fi.width);
        hBuf    = GlobalLock(Hhandle);
        Dhandle  = GlobalAlloc(GPTR,(long)fi.width);
        MemBuf          = GlobalLock(Dhandle);

    Threshold = Entropy();
        for (y=0;y<fi.depth;y++){
                ReadMem(hBuf,fi.width,y);
                for (x=0;x<fi.width;x++){
                        color = *(hBuf+x);
                        if (color >= Threshold)
```

```
                    color = 250;
                else
                    color = 0;

        *(MemBuf+x) = color;
                    SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(color));
                }
                WriteMem(MemBuf,fi.width,y);
            }
            GlobalUnlock(Hhandle);
            GlobalFree(Hhandle);
            GlobalUnlock(Dhandle);
            GlobalFree(Dhandle);
}

int CalEmphasis(int gray,int Threshold)
{
        float Beta,a,n,Bm,Bl;
    int B;

        a = 5;
        Bm = 3;
    Bl = 1;
        n = gray;    // n=Grey Level at Pixel


        if(n >=0 && n <= a)
            Beta = Bm;
        else
            if(gray > a && gray <= Threshold)
                Beta = -(n*(Bm-Bl)/(Threshold - a))+((Threshold*Bm - a*Bl)/(Threshold-a));
            else
                Beta = Bl;

    B = Beta;
        return(B);
```

```
}

void AnatomicChest(void) //use adaptive unsharp masking
{
int i,j,x,y,B,Point,Threshold;
int far *Btable;
float value,temp,temp1,temp2,temp3,temp4;
unsigned char color;
unsigned char far *sharp;
HGLOBAL Hhandle,hglb,Dhandle,Bhandle;
unsigned char far *hBuf;
unsigned char far *MemBuf;


        Hhandle = GlobalAlloc(GPTR,fi.width);
        hBuf    = GlobalLock(Hhandle);
        hglb    = GlobalAlloc(GPTR,(long)fi.width*3); // allocate 3 row
        sharp   = (unsigned char far *)GlobalLock(hglb);
        Dhandle = GlobalAlloc(GPTR,fi.width);
        MemBuf  = GlobalLock(Dhandle);
        Bhandle = GlobalAlloc(GPTR,256*sizeof(int));
        Btable  = (int far *)GlobalLock(Bhandle);

        Threshold = Entropy();

                for(i=0;i<256;i++)
       *(Btable+i) = CalEmphasis(i,Threshold);
  // Point = CalEmphasis(240,Threshold);

                for(j=0;j<3;j++){
    ReadMem(hBuf,fi.width,j);
                  for(i=0;i<fi.width;i++)     // read first 3 rows using 3x3
       *(sharp+fi.width*j+i) = *(hBuf+i);
    }
```

```
        // write first 2 line
        for(x=0;x<fi.width;x++){          // write first line to screen
                SetPixel(hDC,x,fi.depth,PALETTEINDEX(*(sharp+x)));// Line 1
                temp  = *(sharp+fi.width*1+x);  // Center Point
                temp1 = *(sharp+fi.width*2+x);
                temp2 = *(sharp+fi.width*0+x);
                temp3 = *(sharp+fi.width*1+x+1);
                temp4 = *(sharp+fi.width*1+x-1);
Point = temp;
                B = *(Btable+Point);
                value = temp + B*(temp1+temp2+temp3+temp4)/4;
                if(value <0)
                        value = 0;
                else if (value>255)
                        value = 255;
                SetPixel(hDC,x,fi.depth-1,PALETTEINDEX(value)); // Line 2
        }

//*************************************************************//
    for(y=2;y<fi.depth-1;y++){     // Calculate Sharpening loop
                ReadMem(hBuf,fi.width,y);
                for(i=0;i<fi.width;i++){
                        *(sharp+i)                              = *(sharp+fi.width*1+i);   //
row 0 = row 1
                        *(sharp+fi.width*1+i) = *(sharp+fi.width*2+i);// row 1 = row 2
                        *(sharp+fi.width*2+i) = *(hBuf+i);
                }

                for(x=1;x<fi.width;x++){
                 temp  = *(sharp+fi.width*1+x);  // Center Point
                 temp1 = *(sharp+fi.width*2+x);
                 temp2 = *(sharp+fi.width*0+x);
                 temp3 = *(sharp+fi.width*1+x+1);
                 temp4 = *(sharp+fi.width*1+x-1);
```

```
        Point = temp;
        B = *(Btable+Point);
                        value = temp + B*(temp1+temp2+temp3+temp4)/4;
                                if(value <0)
                                        value = 0;
                                else if (value>255)
                                        value = 255;

                                *(MemBuf+x) = value;
                                SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(value));
                        }
                        WriteMem(MemBuf,fi.width,y);
                }
        //*********************************************// Scroll Row up



        GlobalUnlock(Hhandle);
        GlobalFree(Hhandle);
        GlobalUnlock(hglb);
        GlobalFree(hglb);
        GlobalUnlock(Dhandle);
        GlobalFree(Dhandle);
        GlobalUnlock(Bhandle);
        GlobalFree(Bhandle);
}

void AnatomicEqualize(void)
{
HGLOBAL hglb11,hglb22,hglb33,hglb55,Hhandle;
unsigned char far *Hbuf;
unsigned char far *MemBuf;
int i,x,y,NumPart2;
char ch;
unsigned char color;
long int far *hist;
```

```
unsigned char far *table;
long int far *Cumulative;
int temp,temp2,Threshold;
double temp11;TotalPixel,TotalPart1,TotalPart2;
float temp1;


        hglb11  = GlobalAlloc(GPTR,256*sizeof(long int));
                hist    = (long int far *)GlobalLock(hglb11);
                hglb33  = GlobalAlloc(GPTR,256*sizeof(long int));
                Cumulative  = (long int far *)GlobalLock(hglb33);
    hglb55  = GlobalAlloc(GPTR,256*sizeof(unsigned char));
                table   = (unsigned char far *)GlobalLock(hglb55);
                Hhandle = GlobalAlloc(GPTR,fi.width);
                Hbuf    = GlobalLock(Hhandle);
                hglb22  = GlobalAlloc(GPTR,fi.width);
                MemBuf  = GlobalLock(hglb22);
        /************* HISTOGRAM EQUALIZATION *************/
   Threshold  = Entropy();
        TotalPixel = (long)fi.width*(long)fi.depth;
        NumPart2        = 256-Threshold;


        for(i=0;i<256;i++)
                *(hist+i) = 0;



        for(y=0;y<fi.depth;y++){    // create histogram
   ReadMem(Hbuf,fi.width,y);
                for(x=0;x<fi.width;x++){
                        temp        = *(Hbuf+x);
                        hist[temp] += 1;
                }
        }
        TotalPart1 = 0;
   TotalPart2 = 0;
        for(i=0;i<256;i++){
```

```
if (i<Threshold){
        TotalPart1 += *(hist+i);
            *(Cumulative+i) = TotalPart1;
        }else{
            TotalPart2 += *(hist+i);
  *(Cumulative+i)= TotalPart2;
        }
    }

    for(i=0;i<256;i++){
  if(i<Threshold){
//                  temp11 =    ceil(*(Cumulative+i)*Threshold/TotalPart1)-1;
                    temp11 =    ceil(*(Cumulative+i)*256/TotalPart1)-1;
                    if(temp11>255)
                        temp11 = 255;
                    else
                        if (temp11<0)
                            temp11 = 0;
                    *(table+i) = temp11;
                }else{
//                  temp11 =    ceil(*(Cumulative+i)*NumPart2/TotalPart2)-1;
                   //   temp11 =    ceil(*(Cumulative+i)*256/TotalPart2)-1;
                   temp11 = i;
                    if(temp11>255)
                        temp11 = 255;
                    else
                        if (temp11<0)
                            temp11 = 0;
                    *(table+i) = temp11;
                }
    }

    for(y=0;y<fi.depth;y++){
  ReadMem(Hbuf,fi.width,y);
        for(x=0;x<fi.width;x++){
```

```
                    temp  = *(Hbuf+x);
                    temp2 = *(table+temp);
                    *(MemBuf+x) = temp2;
                    SetPixel(hDC,x,fi.depth-y,PALETTEINDEX(temp2));
                }
                WriteMem(MemBuf,fi.width,y);
            }

        GlobalUnlock(hglb11);
        GlobalFree(hglb11);
        GlobalUnlock(hglb22);
        GlobalFree(hglb22);
        GlobalUnlock(hglb33);
        GlobalFree(hglb33);
        GlobalUnlock(Hhandle);
        GlobalFree(Hhandle);
        GlobalUnlock(hglb55);
        GlobalFree(hglb55);

}
/*****************end program***********************************/
```