



การบีบอัดและขยายไฟล์ข้อมูลรูปภาพ
(Image Data Files Compression and Expansion)



โดย

นางสาววิภารัตน์

แจ้ฉิน

นายอดิรุจน์

เยี่ยมเจริญ

นายอังกะ

สิทธิชัย

วัน เดือน ปี..... 15 ส.ค. 2560

เลขทะเบียน..... 037239

เลขเรียกหนังสือ..... 138332 ก 654

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้เผยแพร่ไปใช้ในการค้า

037239

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การบีบอัดและขยายไฟล์ข้อมูลรูปภาพ
(Image Data Files Compression and Expansion)

โดย

นางสาววิภารัตน์	แจ้งถิ่น	35104388
นายอดิรุจน์	เยี่ยมเจริญ	35104516
นายอังคาร	สิทธิชัย	35104549

อาจารย์ที่ปรึกษา

ผศ.ดร. สุวิพล	สิทธิวิชาค
อ. เกียรติกร	วงศ์โรจน์ภรณ์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การบีบอัดและขยายไฟล์ข้อมูลรูปภาพ

(Image Data Files Compression and Expansion)

ผู้จัดทำ

1. นางสาววิภาวรัตน์ แจ่มถื่น 35104388
2. นายอดิรุจน์ เยี่ยมเจริญ 35104516
3. นายอังคาร สิริชัย 35104549

สุวิมล

(ผศ.ดร. สุวิมล สิริชีวกาศ)

อาจารย์ที่ปรึกษา

อ. เกรียงไกร วงศ์โรจน์ภรณ์

(อ. เกรียงไกร วงศ์โรจน์ภรณ์)

อาจารย์ที่ปรึกษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การบีบอัดและขยายไฟล์ข้อมูลรูปภาพ
(Image Data Files Compression and Expansion)

โดย 1. นางสาววิภารัตน์ แจ่มถิ่น 35104388
2. นายอดิรุทธ์ เยี่ยมเจริญ 35104516
3. นายอังคาร สิทธิชัย 35104549

อาจารย์ที่ปรึกษา 1. ผศ.ดร. สุวิพล สิทธิชีวภาค
2. อ. เกรียงไกร วงศ์โรจน์ภรณ์

บทคัดย่อ

ปริญญานิพนธ์นี้ มีจุดประสงค์ เพื่อทำการลดขนาดไฟล์ข้อมูลภาพนิ่ง ที่จะทำให้การเก็บลงในอุปกรณ์เก็บข้อมูล โดยสามารถแบ่งออกได้เป็น การบีบอัดและขยายไฟล์ข้อมูลภาพ ด้วยวิธีสุ่มค่าความสว่าง (Luma-Subsampling) สำหรับภาพขาว-ดำแบบเกรย์สเกล , วิธีสุ่มค่าข้อมูลสี (Chroma - Subsampling) สำหรับภาพสี , วิธีตามความต่อเนื่องของข้อมูล (LZW) และ วิธีเจแพก (JPEG) สำหรับภาพทั้งสองชนิด ส่วนที่สอง การถอดข้อมูลรูปภาพสี ออกจากไฟล์ข้อมูลภาพชื่อไฟล์ .PCX และการเขียนเมนูเพื่อการใช้งานและวิเคราะห์ การบีบอัดและขยายข้อมูลภาพ ขาว-ดำ (เกรย์สเกล)

ABSTRACT

The purpose of this project are about still image and divided into 3 parts for storing.

Compression and expansion by Luma-Subsampling for Gray-scale image , Chroma-Subsampling for color image , LZW and JPEG for both of them. Reconstruction color image data from the image file , .PCX file. And menu for using and analyzing Gray-scale image compression.

สารบัญ

บทที่	หน้า
1. บทนำ	1
2. การถอดข้อมูลภาพออกจากไฟล์ .PCX	
2.1 โครงสร้างหลักของไฟล์ .PCX	5
2.2 การเข้ารหัสข้อมูลภาพ ไฟล์ .PCX	7
2.3 การถอดรหัสข้อมูลภาพ ไฟล์ .PCX	8
3. วิธีสุ่มค่าความสว่าง (Luma subsampling) และสุ่มค่าสี (Chroma subsampling)	
3.1 วิธีสุ่มค่าความสว่าง	9
3.2 วิธีสุ่มค่าสี	9
3.2.1 การสุ่ม 1 ทิศทาง	9
3.2.2 การสุ่ม 2 ทิศทาง	10
4. วิธีความต่อเนื่องของข้อมูล (LZW)	
4.1 แบบจำลองการเข้ารหัส	15
4.2 แบบจำลองการถอดรหัส	17
5. วิธีเจเพก (JPEG)	
ภาคที่ 1 เจเพก ภาพขาว - ดำ	20
5.1 การแปลงโคไซน์แบบไม่ต่อเนื่อง	21
5.2 การควอนไทซ์	28
5.3 การจัดลำดับแบบซิกแซก	30
5.4 การเข้ารหัส	30
5.5 การวัดผลการลดขนาดข้อมูล	34
ภาคที่ 2 เจเพก ภาพสี	36
6. ผังงาน (Flow Chart)	41
7. การทดลอง	
ภาคที่ 1 การประมวลผลภาพขาว - ดำ	58
ภาคที่ 2 การประมวลผลภาพสี	59
8. ผลการทดลอง	
ภาคที่ 1 การประมวลผลภาพขาว - ดำ	60
ภาคที่ 2 การประมวลผลภาพสี	69

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9. สรุปและวิเคราะห์ผลการทดลอง	82
ภาคที่ 1 การประมวลผลภาพขาว - ดำ	83
ภาคที่ 2 การประมวลผลภาพสี	84
10. แนวทางการพัฒนา	

- ภาคผนวก ก โปรแกรมการวิเคราะห์ผลการบีบข้อมูลภาพขาว-ดำในเมนู
- ภาคผนวก ข โปรแกรมการถอดข้อมูลภาพออกจากไฟล์ .PCX
- ภาคผนวก ค โปรแกรมวิธีสุ่มค่าความสว่างและสุ่มค่าสี
- ภาคผนวก ง โปรแกรมวิธีความต่อเนื่องของข้อมูล
- ภาคผนวก จ โปรแกรมวิธีเจเพก
- ภาคผนวก ฉ โปรแกรมการคำนวณค่าความผิดพลาดของภาพจากการบีบและขยายข้อมูล
- กิตติกรรมประกาศ
- เอกสารอ้างอิง

สารบัญผังงาน

	หน้า
ผังงานที่ 6.1 การถอดข้อมูลภาพออกจากไฟล์ .PCX	40
ผังงานที่ 6.2 การบีบข้อมูลแบบ LUMA 2 : 1	41
ผังงานที่ 6.3 การขยายข้อมูลแบบ LUMA 2 : 1	42
ผังงานที่ 6.4 การบีบข้อมูลแบบ LUMA 4 : 1	43
ผังงานที่ 6.5 การขยายข้อมูลแบบ LUMA 4 : 1	44
ผังงานที่ 6.6 การบีบข้อมูลแบบ Chroma 4 : 2 : 2	45
ผังงานที่ 6.7 การขยายข้อมูลแบบ Chroma 4 : 2 : 2	46
ผังงานที่ 6.8 การบีบข้อมูลแบบ Chroma 2 ทิศทาง 4 : 1 : 1	47
ผังงานที่ 6.9 การขยายข้อมูลแบบ Chroma 2 ทิศทาง 4 : 1 : 1	48
ผังงานที่ 6.10 การบีบข้อมูลแบบ LZW ภาพขาว-ดำ และภาพสี	49
ผังงานที่ 6.11 การขยายข้อมูลแบบ LZW ภาพขาว-ดำ และภาพสี	50
ผังงานที่ 6.12 การบีบข้อมูลแบบ JPEG ภาพขาว-ดำ	51
ผังงานที่ 6.13 กระบวนการ FDCT , ควอนไทซ์และการเข้ารหัส	52
ผังงานที่ 6.14 การขยายข้อมูลแบบ JPEG ภาพขาว-ดำ	53
ผังงานที่ 6.15 กระบวนการ IDCT , ดีควอนไทซ์และการถอดรหัส	54
ผังงานที่ 6.16 การบีบข้อมูลแบบ JPEG ภาพสี	55
ผังงานที่ 6.17 การขยายข้อมูลแบบ JPEG ภาพสี	56
ผังงานที่ 6.18 การวิเคราะห์ผลการบีบอัดข้อมูลภาพ ขาว-ดำ ในเมนู	57

สารบัญรูปภาพ (ผลการทดลอง)

	หน้า
ภาคที่ 1 การประมวลผลภาพขาว - ดำ	
รูปที่ 1.1 แสดงภาพ ROSE ดันฉบับ	62
รูปที่ 1.2 แสดงภาพ CLOWN ดันฉบับ	62
รูปที่ 1.3 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี LUMA 2 : 1	63
รูปที่ 1.4 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี LUMA 2 : 1	63
รูปที่ 1.5 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี LUMA 4 : 1	64
รูปที่ 1.6 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี LUMA 4 : 1	64
รูปที่ 1.7 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี LZW	65
รูปที่ 1.8 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี LZW	65
รูปที่ 1.9 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี JPEG 3	66
รูปที่ 1.10 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี JPEG 3	66
รูปที่ 1.11 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี JPEG 10	67
รูปที่ 1.12 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี JPEG 10	67
รูปที่ 1.13 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี JPEG 25	68
รูปที่ 1.14 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี JPEG 25	68
ภาคที่ 2 การประมวลผลภาพสี	
รูปที่ 2.1 แสดงภาพ LISA ดันฉบับ	71
รูปที่ 2.2 แสดงภาพ BIG ดันฉบับ	71
รูปที่ 2.3 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี CNO411 (Chroma 4:1:1 ไม่แปลง RGB เป็น YCrCb)	72
รูปที่ 2.4 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี CNO411	72
รูปที่ 2.5 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี C411 (Chroma 4:1:1 แปลง RGB เป็น YCrCb)	73
รูปที่ 2.6 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี C411	73
รูปที่ 2.7 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี CR4 (Chroma 4:1:1 แบบ 2 ทิศทาง)	74

รูปที่ 2.8 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี CR4	74
รูปที่ 2.9 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี CR5	75
(Chroma 16:1:1 แบบ 2 ทิศทาง)	
รูปที่ 2.10 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี CR5	75
รูปที่ 2.11 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี LZW	76
รูปที่ 2.12 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี LZW	76
รูปที่ 2.13 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี JPEG 1	77
รูปที่ 2.14 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี JPEG 1	77
รูปที่ 2.15 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี JPEG 10	78
รูปที่ 2.16 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี JPEG 10	78
รูปที่ 2.17 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี JPEG 25	79
รูปที่ 2.18 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี JPEG 25	79
รูปที่ 2.19 แสดงเมนูเพื่อช่วยวิเคราะห์ผลการบีบอัดข้อมูลภาพขาว - ดำ	80
รูปที่ 2.20 แสดงเมนูเพื่อเลือกวิธีการที่จะนำภาพที่มีอยู่มาทำการวิเคราะห์	80
รูปที่ 2.21 แสดงหน้าจอเพื่อรับค่าไฟล์อินพุต ไฟล์เอาต์พุต และค่าแฟกเตอร์คุณภาพ สำหรับกระบวนการแจก	81
รูปที่ 2.22 แสดงผลการวิเคราะห์ภาพอินพุต เมื่อทำการบีบอัดด้วยวิธีที่กำหนด	81

บทที่ 1

บทนำ

ปัจจุบันระบบทางดิจิทัลนับได้ว่ามีบทบาทต่อมนุษย์เป็นอย่างมาก เพราะข้อดีหลากหลายประการที่ได้เปรียบระบบทางอนาล็อกอยู่อย่างมากมาย อย่างเช่น การเก็บข้อมูลภาพ (Image) แสดงดังตารางที่ 1.1 จะเห็นว่าขนาดของข้อมูลมหาศาลอยู่เหมือนกัน จึงเป็นการสมควรอย่างยิ่งที่จะต้องมีการบีบอัดข้อมูล ไม่ว่า จะเพื่อเก็บไว้ในอุปกรณ์เก็บข้อมูล หรือเพื่อการส่งผ่านโครงข่ายต่างๆ และจำเป็นที่จะต้องมีการขยายข้อมูล กลับคืน เพื่อสร้างเป็นภาพที่ต้องการแสดงผล อีกด้วย

	IMAGE
OBJECT TYPE	-Bit-mapped graphics -Still photos -Faxes
SIZE and BANDWIDTH	-Simple (Gray-scale) 77 kB per image (320 x 240 x 8 bits) -Detailed (Color) 3 MB per image (1100 x 900 x 24 bits)

ตารางที่ 1.1 แสดงขนาดของข้อมูลภาพ

สมมุติภาพต้นฉบับมีขนาดถึง 3 GB (3000 ภาพที่เป็นภาพสี) ถ้าเราทำการบีบอัดข้อมูลด้วยอัตราส่วน 15 : 1 เราจะสามารถลดขนาดได้ เหลือเพียง 0.2 GB เท่านั้น ซึ่งอาจสรุปความสำคัญของการบีบอัดข้อมูลโดยอาศัยความสัมพันธ์กับระบบมัลติมีเดีย (Multimedia System) ที่กำลังเป็นที่นิยมกันอย่างแพร่หลายเป็นหลัก ได้ดังต่อไปนี้

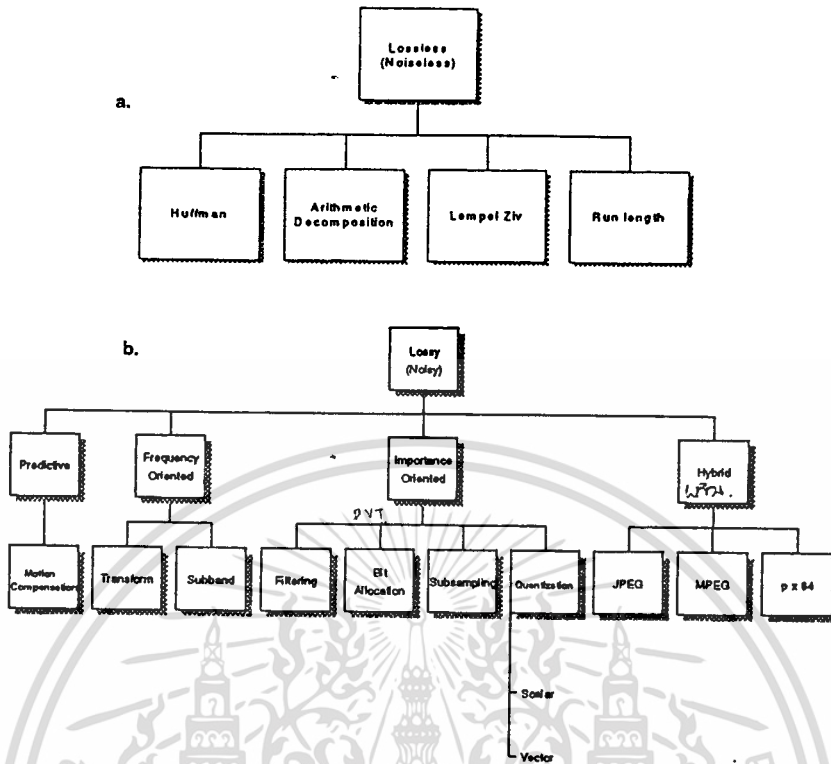
1. เนื่องจากข้อมูลภาพ เป็นข้อมูลี่ส่วนหนึ่งของระบบมัลติมีเดีย ซึ่งต้องการเนื้อที่ในการเก็บข้อมูลเป็นจำนวนมาก การบีบอัดข้อมูลจะช่วยให้สามารถประหยัดเนื้อที่ในการเก็บข้อมูลได้มากขึ้น
2. เพื่อรองรับการเล่นย้อนกลับ (Play Back) ข้อมูลมัลติมีเดียแบบเวลาจริง (Real time)
3. ช่วยให้เราสามารถส่งข้อมูลผ่านโครงข่ายที่มีแบนวิธ (Bandwidth) อย่างจำกัดได้แบบเวลาจริง

โดยพื้นฐานสามารถจำแนกการบีบอัดข้อมูลออกได้เป็น 2 ประเภท คือ

1. การลดขนาดข้อมูลภาพโดยไม่มีการสูญเสียของข้อมูล (Lossless)
2. การลดขนาดข้อมูลภาพโดยที่มีการสูญเสียของข้อมูลเกิดขึ้น (Lossy)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชื่อของกระบวนการต่างๆ อาจแสดงได้ตามรูปที่ 1.1



รูปที่ 1.1 แสดงชื่อของกระบวนการในการลดขนาดข้อมูล รูป (a) คือการลดขนาดข้อมูลแบบไม่มีการสูญเสีย และรูป (b) คือการลดขนาดข้อมูลแบบมีการสูญเสียของข้อมูล

การลดขนาดข้อมูลแบบมีการสูญเสียของข้อมูลนั้น อาศัยหลักความจริงที่ว่า บรรดาภาพที่ปรากฏบนจอภาพนั้น จะมีรายละเอียดของภาพมากมาย เกินกว่าที่ตาของคนเราจะมองเห็นแล้วแยกแยะได้ทั้งหมด เป็นต้นว่า สีเส้นที่มีเฉดโทนของสีที่ใกล้เคียงกันมากจนเราไม่อาจตัดสินได้ว่าเป็นสีเดียวกันหรือไม่ ด้วยเหตุนี้ ได้มีการออกแบบการลดขนาดข้อมูลโดยไม่คำนึงถึงข้อมูลภาพทั้งหมด ข้อมูลสีที่มีความใกล้เคียงกันมาก อาจถูกตัดทอนออกไปบ้าง แต่ก็ยังนับว่าเป็นการสูญเสียที่ยังยอมรับคุณภาพได้

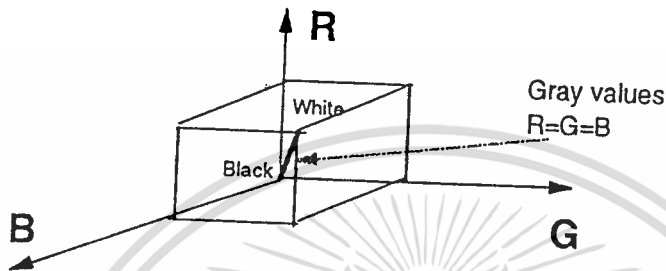
การลดขนาดข้อมูลด้วยวิธีต่างๆ จะให้อัตราการลดขนาดข้อมูลที่แตกต่างกันออกไป มากบ้างน้อยบ้าง ซึ่งจะมีสิ่งหนึ่งที่คล้ายกัน คือ ถ้าลดขนาดข้อมูลได้มากเท่าไร ความผิดเพี้ยนของภาพจากกระบวนการจะมากขึ้นตามไปด้วย ในขณะที่ภาพที่ต้องการคุณภาพดีเยี่ยม เราอาจไม่สามารถลดขนาดข้อมูลได้มากเท่าที่ควร

ระดับคุณภาพของการลดขนาดข้อมูลในหลายๆ รูปแบบ อาจมีการกำหนดอัตราการสูญเสียของข้อมูล เป็น 1 - 100 โดยที่ 100 หมายถึงข้อมูลจะเกิดการสูญเสียน้อยที่สุด และ 1 หมายถึงข้อมูลจะเกิดการสูญเสียมากที่สุด โดยปกติควรมีค่าไม่ต่ำกว่า 7.5 อย่างไรก็ตาม บางวิธีการจะมีการกำหนดค่าที่ใช้ในการกำหนดคุณภาพของภาพโดยเฉพาะเลยก็มี เช่น เจเพก เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

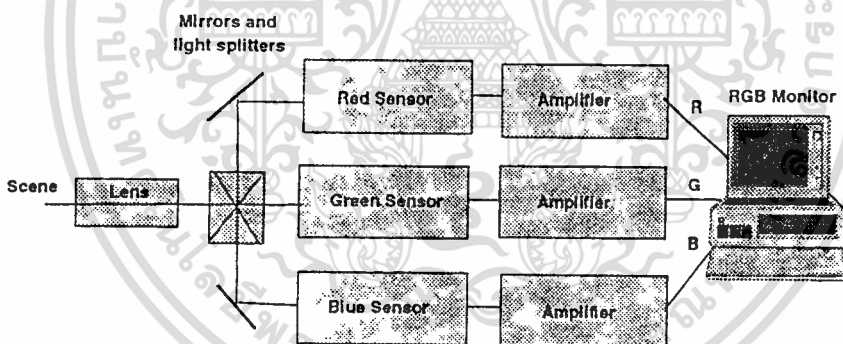
สีพื้นบนหน้านั้นเกิดจากการผสมกันของสีหลัก คือ สีแดง สีเขียวและสีน้ำเงิน (R , G , B) ในอัตราส่วนต่างๆ กัน ส่วนภาพขาว-ดำนั้นเกิดจากสีหลักผสมกันในอัตราส่วนที่เท่ากัน (R = G = B) ภาพขาว-ดำ มีอยู่หลายชนิด เช่น ภาพไบนารี (Binary image) แสดงข้อมูลด้วย 1 บิตต่อ 1 แซมเปิ้ล (Sample) ภาพคอมพิวเตอร์กราฟฟิค ที่แสดงผลแบบ 4 บิตต่อแซมเปิ้ลและภาพเกรย์สเกล (Grey - scale) แสดงผลด้วย 8 บิตต่อแซมเปิ้ล สำหรับภาพสีนั้น แสดงด้วย 16 , 24 หรือมากกว่า 24 บิตต่อแซมเปิ้ล

ลักษณะของข้อมูลสีแสดงตามรูปที่ 1.2



รูปที่ 1.2 แสดงส่วนประกอบของสี

การแสดงผลออกมาเป็นสีต่างๆ นั้น ถ้าพิจารณากับจอโหมด RGB แล้ว จะเป็นดังรูปที่ 1.3



รูปที่ 1.3 แสดงลักษณะการแสดงผลออกทางจอภาพโหมด RGB

สำหรับการแสดงผลในโหมดอื่นนั้น สมมติเป็น YUV (มาตรฐานของ CCIR 601) ก็อาศัยความสัมพันธ์กับ RGB กล่าวคือ

$$Y = 0.299R + 0.587G + 0.114B \quad \therefore \text{ข้อมูลความสว่าง (Luminance information)}$$

$$U = 0.564 (B - Y) \quad \therefore \text{ข้อมูลสี (Chrominance information)}$$

$$V = 0.713 (R - Y) \quad \therefore \text{ข้อมูลสี}$$

ส่วนโหมดอื่นๆ ที่ไม่ได้กล่าวถึงก็มีความสัมพันธ์กับ RGB ในลักษณะเดียวกัน แต่ต่างสมการ

หน่วยความจำ ในการแสดงผลภาพกราฟฟิค หรือเรียกอีกอย่างหนึ่งว่า เฟรมบัฟเฟอร์ (Frame

buffer) สามารถจำแนกออกได้เป็น 2 ประเภทด้วยกัน คือ

เอ็กสแรนเป็นเอ็กสแรนที่ส่งวินโดวสำหรับการใช้งานเพื่อการศึกษเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

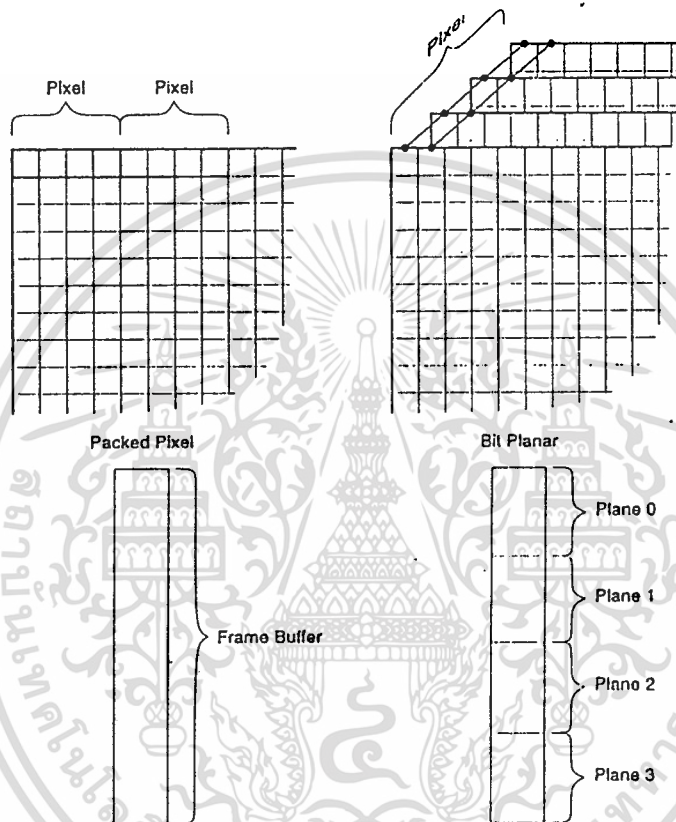
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1) แบบแพกพิกเซล (Packed pixel)

2) แบบระนาบบิต (Bit planar)

แบบระนาบบิตนี้ จะเป็นผลดี ในการใช้งานสำหรับข้อมูลภาพที่มีหลายๆ บิตต่อพิกเซล ในรูปที่

1.4 แสดงถึงตัวอย่างในการใช้งานแบบระนาบบิต ขนาด 4 บิตต่อพิกเซล (16 สี) เทียบกับชนิดแพกพิกเซล



รูปที่ 1.4 แสดงลักษณะของเฟรมบัฟเฟอร์ที่ทำการเก็บข้อมูลภาพ

บทที่ 2

ไฟล์ .PCX

2.1 โครงสร้างหลักของไฟล์ .PCX

ไฟล์ .PCX นั้น จะเริ่มต้นไฟล์ด้วยข้อมูลขนาด 128 ไบต์ (Bytes) ซึ่งจะทำให้การเก็บข้อมูลที่เป็นเฮดเดอร์ (Header) ต่าง ๆ เพื่อที่จะนำไปใช้ในการเก็บ (Restore) ภาพ ดังรูปที่ 2.1 ค่าของเฮดเดอร์ค่าใดที่ไม่ถูกใช้ มักจะทำการรีเซ็ตให้มีค่าเท่ากับ 0 (ศูนย์) เสมอ

โครงสร้างของฟอร์แมต (Format) ไฟล์ .PCX ทั้งหมด จะมีลักษณะเป็น

- 1) เฮดเดอร์ (Header) ขนาด 128 ไบต์
- 2) ข้อมูลบิตแมพ (Bit Mapped) ที่เข้ารหัสตามความต่อเนื่องของข้อมูล (RLE (Run-Length Encoding))

ส่วนเฮดเดอร์ที่เป็นพาเลท (Palette) จะเป็นตารางสำหรับตัวแปรสีหลัก คือ สีแดง สีเขียวและสีน้ำเงิน (RGB) ขนาด 3 ช่อง ซึ่งพาเลทจะมีขนาด 48 ช่อง จาก 16 ช่องของตารางสีหลักคูณกับ 3 ช่องที่เป็นสีหลัก

ไบต์ที่	ขนาด(ไบต์)	ชื่อข้อมูล	รายละเอียด
0	1	Password	ค่า '0aH' = 10 แสดงว่าไฟล์นี้เป็นไฟล์ .PCX
1	1	Version	เก็บค่ารุ่นของ PC Paintbrush ดังนี้ 0 = รุ่นที่ 2.5 2 = รุ่นที่ 2.8 พร้อมข้อมูลพาเลท 3 = รุ่นที่ 2.8 ไม่มีข้อมูลพาเลท 5 = รุ่นที่ 3.0
2	1	Encoding	ค่า = 1 แสดงว่าบิตแมพถูกเข้ารหัสด้วยวิธีความต่อเนื่องของข้อมูลหรือ RLE นั้นเอง เนื่องจากบิตแมพต้องถูกเข้ารหัสด้วยวิธีนี้เสมอ ดังนั้นค่านี้จึงถูกเซตให้เท่ากับ 1 ตลอด
3	1	Bits per Pixel	จำนวนของบิตที่ใช้แสดงอิมเมจ 1 พิกเซล (Pixel) จาก 1 Plane (ระนาบสี) มีค่า 1,2,4 หรือ 8 ภาพอิมเมจสีที่เป็น 24 บิตต่อพิกเซล โดยทั่วไปเป็น 3 ระนาบของแต่ละ 8 บิตต่อพิกเซล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไบต์ที่	ขนาด(ไบต์)	ชื่อข้อมูล	รายละเอียด
4	8	Windows - Dimension	ค่าจำนวนเต็ม (Integer) 4 ค่า (ค่าละ 2 ไบต์) ให้ค่าตำแหน่งที่มุมบนซ้ายและค่าตำแหน่งที่มุมล่างขวาของภาพ กำหนดให้มีรูปแบบเป็น x1, y1, x2, y2 โดยปกติถ้าเป็นภาพเต็มจอ ค่าของตำแหน่ง ณ มุมบนซ้ายของภาพ x1 , y1 จะเป็น 0,0 และค่าตำแหน่ง ณ มุมล่างขวาของภาพ สำหรับในโหมดวีดีโอ ที่มีขนาดจอเป็น W x H (กว้าง x ยาว) จะมีค่า x2 , y2 เป็น W-1 , H-1 ตามลำดับ
12	2	Horizontal - Resolution	ความละเอียดของการแสดงภาพในแนวนอน มีหน่วยเป็นจำนวนจุดต่อนิ้ว (Dots per Inch) 640 = EGA, VGA 320 = CGA 720 = HERCULES
14	2	Vertical - Resolution	ความละเอียดของการแสดงภาพในแนวตั้ง มีหน่วยเป็นจำนวนจุดต่อนิ้ว 480 = VGA 350 = EGA 200 = CGA 348 = HERC
16	48	Color Map	ข้อมูลพาเลทสี
64	1	Reserved	
65	1	No. of Plane	จำนวนของระนาบที่ใช้แสดงภาพ ค่านี้อาจเป็น 1 หรือ 4 แต่ต้องมีค่าเป็น 3 เมื่อเป็นภาพอิมเมจ 24 บิต แม้ว่าการแสดงภาพ 24 บิตนี้จะไม่ใช้แบบหลายระนาบก็ตาม ก็ยังต้องคงค่านี้เอาไว้
66	2	Bytes per - Line	จำนวนไบต์ต่อการสแกน 1 บรรทัดต่อระนาบ
68	2	Palette Info	ข่าวสารการแปลงข้อมูลพาเลทที่แสดงถึงชนิดของพาเลท ซึ่งจะมีค่าเป็น 1 เมื่อเป็นพาเลทสี และเป็น 2 เมื่อเป็นพาเลทขาว-ดำ
70	8	Maximum x	ข้อมูลที่ ใช้เฉพาะ Fractal File

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไบต์ที่	ขนาด(ไบต์)	ชื่อข้อมูล	รายละเอียด
78	8	Minimum x	ข้อมูลที่ใช้เฉพาะ Fractal File
86	8	Maximum y	ข้อมูลที่ใช้เฉพาะ Fractal File
94	8	Minimum y	ข้อมูลที่ใช้เฉพาะ Fractal File
102	8	P Value	ข้อมูลที่ใช้เฉพาะ Fractal File
110	8	Q Value	ข้อมูลที่ใช้เฉพาะ Fractal File
127	10	Not used	

ตารางที่ 2.1 โครงสร้างหลักของไฟล์ .PCX

ในวิธีเ็ข้อมูลพาเลทจะอยู่ที่ส่วนท้ายของไฟล์ .PCX การเข้าถึงข้อมูลเป็นดังนี้ ต้องตรวจไบต์ตรงตำแหน่งรุ่นของไฟล์ว่าเป็น 5 (รุ่นที่ 3.0) หรือไม่ และข้อมูลพาเลทจะต้องนับย้อนหลังจากท้ายไฟล์ไป 769 ไบต์ ในไบต์นั้นจะต้องเป็น 0cH (12 เลขฐาน 10) จากนั้น ข้อมูลที่ท้ายไฟล์จะเป็นข้อมูลพาเลทสี

2.2 การเข้ารหัสไฟล์ .PCX

ไฟล์นี้มีวิธีการเข้ารหัสข้อมูล ที่เรียกว่า การเข้ารหัสตามความต่อเนื่องของข้อมูล ซึ่งเป็นวิธีการลดขนาดของไฟล์ข้อมูลอย่างหนึ่ง โดยอาศัยการพิจารณาจากความแตกต่างของลำดับบิตของข้อมูล 2 บิตบนของไบต์ และ 6 บิตล่างของไบต์ โดยที่การอ่านฟอร์แมทจะทำการอ่านทีละไบต์ แล้วทำการตรวจสอบที่ตำแหน่ง 2 บิตบนก่อน ดังรายละเอียดดังนี้

ถ้าไบต์ใดๆ มีค่าไม่เหมือนกับไบต์อื่นๆ และถ้าค่า 2 บิตบนไม่เท่ากับ '11' ไบต์นั้นจะถูกเก็บลงไฟล์เลย นอกนั้นจะมีตัวนับ (Counter) อยู่เพื่อคอยนับว่าเหมือนกันกี่ไบต์ แต่ต้องไม่เกิน 63 เนื่องจากการซ้ำกันของข้อมูลพิจารณาที่ 6 บิตล่างของไบต์เท่านั้นจึงมีจำนวนซ้ำกันได้สูงสุดที่ต้องมีค่าไม่เกิน 63 (0011 1111)

ถ้าเกินให้นำค่าตัวนับที่ได้ OR กับ 'c0H' แล้วเก็บค่าตัวนับนั้นลงไฟล์ แล้วตามด้วยค่าของไบต์นั้นๆ จากนั้นจึงเริ่มนับใหม่เป็น 1 ต่อไป

ถ้าในกรณีของไบต์เดียวที่มีค่าบิตบนเป็น '11' ให้เขียนตัวนับเท่ากับ 1 (OR ด้วย c0H = 'c1H') ลงไปในไฟล์และตามด้วยค่าไบต์นั้น ดังแสดงตามตารางที่ 2.2

2.3 การถอดรหัสไฟล์ .PCX

การถอดรหัสใช้วิธีการตรงกันข้ามกับการเข้ารหัส โดย

เงื่อนไข	การกระทำ
ค่าที่เหมือนไบต์อื่น	นับ ค่าของตัวนับ +1
ค่า < 'c0H' ไม่เหมือนไบต์อื่น	เก็บลงไฟล์เลย
ค่า > 'c0H' ไม่เหมือนไบต์อื่น	เก็บ 'c1H' ลงไฟล์ ตามด้วยค่าไบต์นั้น
ค่าตัวนับ > 63	ค่าตัวนับ = 1 เก็บค่า 'FFH' และค่าไบต์ลงไฟล์

ตารางที่ 2.2 แสดงการถอดรหัสไฟล์ .PCX

อ่านข้อมูลมา 1 ไบต์แล้วตรวจดูว่ามากกว่า 'c0H' หรือไม่ (ค่า 2 บิตบนเป็น 11) ถ้าใช่ค่านี้จะเป็นค่าตัวนับทันที (XOR ด้วยค่า c0H จะได้ค่าจริงที่น้อยกว่า 63 ออกมา) ซึ่งในไบต์ต่อไปก็จะเป็นค่าของไบต์ (Value) แล้วทำการขยายข้อมูลออกมาตามจำนวนนับนั้นๆ แล้วนำมาเก็บในหน่วยความจำ

ถ้าค่าไบต์ที่อ่านน้อยกว่า c0H จะทำการเก็บข้อมูลไบต์นั้นตัวเดียวลงในหน่วยความจำที่มีอยู่ได้เลยโดยไม่ต้องทำการขยายข้อมูล

ทำเช่นนี้จนจบไฟล์ก็ได้ข้อมูลของไฟล์ทั้งหมด จากนั้นนำข้อมูลในหน่วยความจำที่ได้มาแสดงบนจอภาพ แล้วนำข้อมูลไปไว้ในหน่วยความจำแสดงผล (Display Memory) หรืออาจจะใช้ฟังก์ชันในการแสดงภาพออกมาก็ได้

บทที่ 3

มาตรฐานการสุ่มสัญญาณความสว่าง (Luma-Subsampling)

มาตรฐานการสุ่มสัญญาณสีของ CCIR (CCIR 601 Color-Subsampling)

3.1 มาตรฐานการสุ่มสัญญาณความสว่าง

การบีบอัดข้อมูลด้วยวิธีนี้เป็น การประมวลผลกับค่าความสว่างหรือความขาว - ดำ ของภาพนั่นเอง (Luminance information) ปริมาณนิพจน์ฉบับนี้ได้รวบรวมผล การบีบอัดข้อมูลภาพเกรย์สเกลที่ค่าของแต่ละพิกเซลแทนด้วยข้อมูลขนาด 8 บิต (0 - 255 ระดับความสว่างของภาพ)

กระบวนการแบบ 2 : 1 นั้น เริ่มจากการอ่านค่าของพิกเซล 2 x 2 พิกเซลที่อยู่ติดกัน นำมาหาค่าเฉลี่ยแล้วเก็บค่าเฉลี่ยที่ได้ จึงเหมือนว่าเราสามารถลดขนาดข้อมูลจาก 4 พิกเซล เหลือเพียง 1 พิกเซลที่จะทำการเก็บเท่านั้น ส่วนวิธีการแบบ 4 : 1 ก็คล้ายกัน ต่างกันที่ว่าจะทำการอ่านค่าของพิกเซลที่อยู่ติดกัน 4 x 4 พิกเซล จึงได้ว่า เราสามารถลดขนาด 16 พิกเซลเหลือเพียง 1 พิกเซล เท่านั้น

ในกระบวนการเขียนหรือขยายข้อมูลคืน ก็จะมีการคัดลอกค่าเฉลี่ยนั้นให้กับพิกเซลที่อยู่ข้างเคียงกัน 4 หรือ 2 x 2 พิกเซล สำหรับแบบ 2 : 1 หรือ 16 หรือ 4 x 4 พิกเซล สำหรับแบบ 4 : 1 ตามลำดับ รายละเอียดของกระบวนการพิจารณาได้จากผังงานในบทที่ 6

3.2 มาตรฐานการสุ่มสัญญาณสีของ CCIR

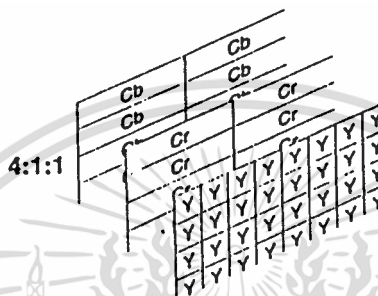
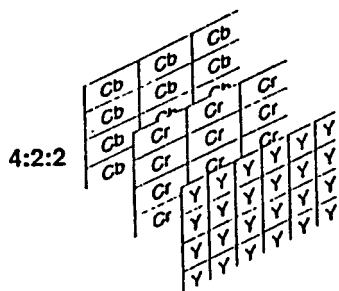
มาตรฐานของ CCIR 601-2 ได้กำหนดวิธีการของการสุ่มสีของข้อมูลดิจิทัลของภาพอิมเมจด้วยการค้นจำนวน 3 จำนวนด้วยโคลอน (Colon , :) โดยที่แต่ละค่าจะบอกถึงจำนวนของการแซมเปิ้ลสำหรับแต่ละคอมโพเนนท์ (Component) ของภาพ จำนวนแรกแสดงค่า Y (Luminance) หรือค่าความสว่างของภาพ จำนวนที่สองแทน C_r (R-Y) หรือผลต่างของค่าสีแดงกับค่าความสว่าง และค่า C_b (B-Y) หรือผลต่างของสีน้ำเงินกับความสว่างของภาพ โดยทั่วไปนิยมใช้อัตราส่วนการสุ่ม 2 ค่า คือ 4:2:2 และ 4:1:1

3.2.1 การสุ่มค่าข้อมูลสีแบบทิศทางเดียว (One - directional Chroma-Subsampling)

วิธีแบบ 4:2:2 หมายถึง ในขณะที่ ทำการสุ่มค่าของสัญญาณความสว่าง (Luminance) จำนวน 4 แซมเปิ้ลก็จะทำการสุ่มค่าสี (Chrominance) คือ C_r และ C_b อย่างละ 2 แซมเปิ้ล หรือกล่าวได้อีกอย่างว่า ทำการสุ่มความสว่าง 2 แซมเปิ้ลในขณะที่สุ่มสีอย่างละ 1 แซมเปิ้ลนั่นเอง

วิธีแบบ 4:1:1 หมายถึง ทำการสุ่มค่าความสว่าง 4 แซมเปิ้ลแล้วทำการสุ่มค่าสี 1 แซมเปิ้ล

ทั้งสองกรณีทีกล่าวนี้นี้ ข้อมูลต่างๆ จะถูกทำการสุ่มตามแนวนอน ซึ่งหมายถึง ทุกค่าความสว่างและข้อมูลสีจำเป็นที่จะต้องลดขนาดข้อมูลจากการสแกน 1 แถวของวิดีโอเฟรม เท่านั้น

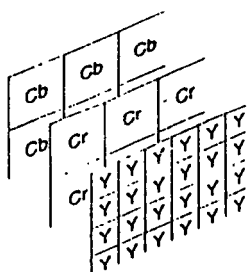


รูปที่ 3.1 แสดงคอมโพเนนต์ของอิมเมจที่สัมพันธ์กันตามแนวนอน สำหรับวิธีการสุ่มสี ทั้งแบบ วิธี 4:2:2 และ 4:1:1

สมมติว่าขนาดของ 1 แรมเบิลเป็น 8 บิตสำหรับแต่ละคอมโพเนนต์ของภาพ จำนวนเฉลี่ยของ บิตต่อพิกเซล สำหรับการสุ่มแบบ 4:2:2 จะเป็น $8 \times (4/2)$ หรือเท่ากับ 16 บิตต่อพิกเซล และสำหรับการสุ่มแบบ 4:1:1 เป็น $8 \times (6/4)$ หรือเท่ากับ 12 บิตต่อพิกเซลโดยเฉลี่ยที่ใช้ในการแสดงผลนั่นเอง

3.2.2 การสุ่มค่าข้อมูลสีแบบ 2 ทิศทาง (Bi-Directional Chroma Subsampling)

วิธีการนี้เป็นการสุ่มแซมเบิลทั้งตามแนวดิ่งและแนวนอน ซึ่งจะให้ผลการลดหรืออัตราการลดขนาดข้อมูลเท่ากับวิธี CCIR 601 4:1:1 ที่เป็น 1 ทิศทางข้างต้น วิธีการนี้แสดงตามรูปที่ 3.2

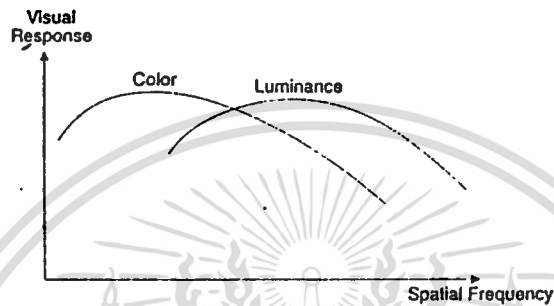


รูปที่ 3.2 แสดงวิธีการสุ่มข้อมูลสีแบบ 2 ทิศทาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หมายเหตุ

สาเหตุที่เราทำการแยกข้อมูลภาพสีจาก RGB มาเป็น YC_Cb เพราะว่า จากการวิจัยของผู้เชี่ยวชาญสรุปได้ว่า ผลตอบสนองต่อตาของมนุษย์เป็นตามรูปที่ 3.3 กล่าวคือ เมื่อความถี่สูงขึ้น มนุษย์จะตอบสนองความสว่างของภาพมากกว่าสีของภาพ หมายถึง มนุษย์สามารถรับรู้การเปลี่ยนแปลงของสีได้น้อยกว่าการเปลี่ยนแปลงของความสว่างของภาพ เราจึงได้นำหลักการนี้มาพิจารณาโดยทำการลดขนาดข้อมูลที่ส่วนที่เป็นข้อมูลสี และยังคงรักษาข้อมูลของความสว่างของภาพไว้ให้มากที่สุด



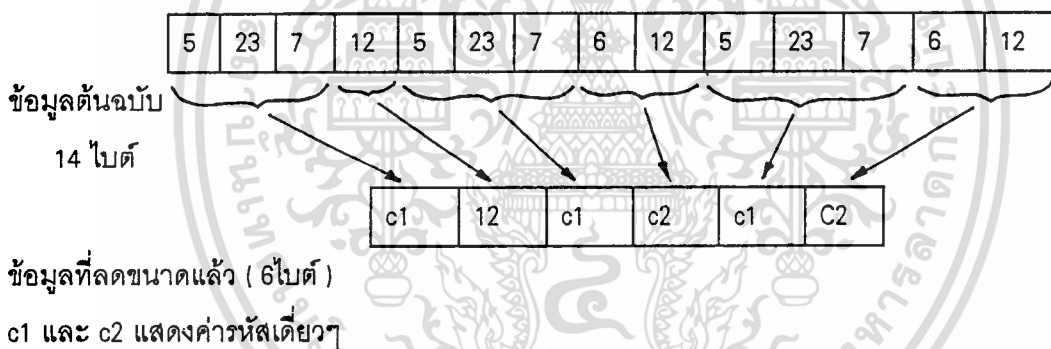
รูปที่ 3.3 แสดงผลตอบสนองต่อข้อมูลที่เป็นความเปลี่ยนแปลงของสีกับความเปลี่ยนแปลงของความสว่างของภาพ

บทที่ 4

LZW การเข้ารหัสด้วยวิธีความต่อเนื่องของข้อมูล

ถ้าเรามีลำดับของค่าของข้อมูล ที่ไม่เหมือนกัน ซึ่งเกิดขึ้นซ้ำ ๆ กันมากกว่า 1 ครั้ง ในสตรีม (Stream) ของข้อมูลแล้ว เราสามารถอาศัยการซ้ำกันของข้อมูลนี้มาทำการลดขนาดข้อมูลทั้งหมดลงได้ โดยที่รหัส (Code) เดียวใดๆ สามารถกำหนดให้กับกลุ่มของข้อมูลที่ซ้ำๆ กันนั้น แล้วทำการเก็บไว้ในเซตของข้อมูลที่ถูกรหัสแล้ว พิจารณาจากรูปที่ 4.1 ซึ่งประกอบด้วย 2 ลำดับของข้อมูลที่เกิดซ้ำกัน จะสังเกตเห็นว่ามีการซ้ำกันของข้อมูลอื่นอยู่ในข้อมูลชุดนี้ด้วย ซึ่งยังไม่ได้ทำการมาร์ค (Mark) และใส่รหัสเอาไว้ให้ ลำดับของ (12 , 5) เกิดซ้ำกันถึง 2 ครั้งแต่ไม่ได้มีการกำหนดรหัสเอาไว้เพราะมันซ้อนทับอยู่กับข้อมูลที่ซ้ำกันอยู่ก่อนหน้าแล้ว (Over Lap)

ลำดับของ (5 , 23 , 7 , 6 , 12) มีการเกิดซ้ำกันถึง 2 ครั้งเหมือนกันในตัวอย่าง แต่ก็ไม่ได้ใช้ในการเข้ารหัสอีกเช่นกัน ซึ่งตัวอย่างนี้แสดงให้เห็นถึงปัญหาจากการแบ่งกลุ่มของค่าของข้อมูล เพื่อทำการลดขนาดให้ได้มากที่สุด



รูปที่ 4.1 แสดงตัวอย่างการกำหนดแพทเทิร์น (Pattern) ที่ซ้ำกันให้กับค่าของรหัสเดียวกัน

การแบ่งกลุ่มของข้อมูลใดๆ ให้ดีที่สุด ควรจะเป็นการเก็บลำดับของข้อมูล เป็นข้อมูลที่ได้จากการสแกนเอาไว้ก่อน แล้วทำการค้นหว่า มีข้อมูลที่เหลือที่ตามมาซ้ำกันหรือไม่ อย่างไร ในตัวอย่างที่กล่าวข้างต้น ลำดับของ (5 , 23 , 7) จะถูกพบก่อนเป็นอันดับแรก จึงถูกกำหนดเป็นรหัสเมื่อมีการค้นพบลำดับข้อมูลชุดนี้ต่อไป มา จะเห็นว่าวิธีนี้จะไม่พบลำดับของข้อมูลที่ซ้ำกันยาวขึ้น คือ (5 , 23 , 7 , 6 , 12)

รหัสและลำดับของข้อมูลที่ตามหลังรหัสมานั้น (ส่วนที่เพิ่มเข้ามาภายหลังที่) จะถูกสร้างเป็นตารางรหัส (Code Book) ซึ่งจะถูกรหัสตามด้วยข้อมูลที่ถูกรหัสแล้ว แต่ในทางกลับกัน ตัวถอดรหัส (Decoder) จะไม่มีทางที่จะทราบว่า รหัสที่แสดงออกมาคืออะไร ถ้าตารางรหัสมีขนาดใหญ่มากจะทำให้ประสิทธิภาพการลดขนาดข้อมูลมีน้อยลงเพราะต้องการเนื้อที่ในการเก็บตารางรหัสมาก ลำดับการลด

ขนาดของข้อมูลด้วยวิธีนี้พัฒนาขึ้นโดย Lampel-Ziv และ Welch เรียกว่าเป็น LZW ซึ่งหลีกเลี่ยงความจำเป็นที่ต้องมีการจดจำตารางรหัส การลดขนาดด้วยวิธีนี้นับได้ว่าเป็นประโยชน์อย่างมากเพราะใช้การประมวลผลข้อมูลแบบเป็นลำดับและต้องการเนื้อที่ในหน่วยความจำเพียงเล็กน้อยเท่านั้น รวมทั้งความสามารถของอัลกอริทึมที่สามารถเข้ารหัสได้จำนวนข้อมูลที่น้อย

การลดขนาดด้วยวิธี LZW มีฐานมาจากความคิด ที่ตัวเข้ารหัสและถอดรหัส (Encode-Decoder) จะสร้างตารางรหัสเป็นสตริงของข้อมูลที่ออกมาเป็นลำดับ ตัวเข้ารหัสจะให้เอาต์พุต (Output) เป็นแพทเทิร์น ของรหัสเมื่อมีการตรวจพบแพทเทิร์นเดียวกันนี้มากกว่า 1 ครั้งเท่านั้น ขั้นแรกมันจะทำการโปรเซส (Process) ลำดับของข้อมูล 1 ชุดหรือ 1 ลำดับก่อน แล้วจัดวางลำดับนี้ลงในตารางสีแล้วจึงเอาต์พุตรหัสนี้โดยปราศจากการเข้ารหัสใดๆ ตัวถอดรหัสก็จะรับลำดับของข้อมูลนี้แล้วแทนลงในตารางสี ตัวเข้ารหัสเมื่อพบว่ามีแพทเทิร์นที่เกิดซ้ำกันเป็นครั้งที่ 2 ก็จะให้เอาต์พุตรหัสจากตารางสีสำหรับแพทเทิร์นนี้ ตัวถอดรหัสก็จะทราบความหมายของรหัสนี้เพราะมันได้ทำการสร้างรหัสที่เหมือนกันไว้ในตารางรหัสที่เป็นของแพทเทิร์นที่เกิดขึ้นเป็นอันดับแรกไว้

ฟอร์แมตของข้อมูล	ชื่อกระบวนการ	หมายเหตุ
GIF PostScript	การย่อข้อมูลด้วย LZW ตัวเข้ารหัส LZW	ยอมให้รหัสมีขนาดอยู่ในช่วง 2-12 บิต มักให้ขนาดของข้อมูลเดิมเป็น 8 บิต
TIFF	ตัวถอดรหัส LZW แบบที่ 5	มักใช้ 8 บิตต่อไบต์สำหรับข้อมูลหรือตามจำนวนบิตต่อพิกเซลของข้อมูลภาพ

ตารางที่ 4.1 สตริงของข้อมูลที่รองรับการเข้ารหัสแบบ LZW

หมายเหตุ

เพื่อทำความเข้าใจ อัลกอริทึมของการเข้ารหัสแบบวิธีนี้มากยิ่งขึ้น จึงต้องมีการกำหนดให้สตริงของค่าข้อมูล (String of Data Values) ขึ้นมา สตริงของค่าข้อมูลจะเป็นค่าของข้อมูลที่ถูกแสดงอย่างเป็นลำดับ ซึ่งมีส่วนนำหน้า (Prefix) และค่าข้อมูลสุดท้าย (Last Data Value) ตัวอย่างเช่น ถ้ามีสตริงของข้อมูลเป็น (12 , 34 , 18 , 11) จะหมายถึง มีส่วนนำหน้าเป็น (12 , 34 , 18) และค่าสุดท้ายเป็น 11 ซึ่งสตริงของค่าข้อมูลนี้จะเป็นแพทเทิร์นของข้อมูลที่ตัวเข้ารหัสและถอดรหัสจะจดจำไว้ในตารางรหัสที่ถูกสร้างไว้เหมือนกัน

อัลกอริทึมการเข้ารหัสแบบ LZW จะเป็นไปตามที่จะกล่าวต่อไปนี้ โดยที่โคลอน (Colon :) จะถูกใช้แสดงความต่อเนื่องของสตริงของข้อมูล เช่น (2 , 8 , 4) : (3) หมายถึงสตริง 2 , 8 , 4 , 3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- * สตริงของค่าข้อมูลจะถูกสร้างขึ้น จนกระทั่งสตริงนี้แตกต่างไปจากสตริงที่ถูกเข้ารหัสตัวก่อนหน้า นั่นคือจะไม่มีข้อมูลใดๆ เข้าไปในตารางรหัสสำหรับสตริงนี้
 - * สตริงข้อมูลใหม่ มักจะถูกสร้างมาจากบางสตริงที่เคยถูกพบมาก่อนหน้าแล้ว รวมกับค่าของข้อมูลใหม่ค่าหนึ่ง
 - * เมื่อมีการค้นพบแพทเทิร์นใหม่ขึ้น จะเกิด
 - 1) แพทเทิร์นใหม่นี้จะถูกรวมเข้าไปในตารางรหัสของตัวเข้ารหัสทันที
 - 2) รหัสสำหรับตัวนำหน้าของสตริงข้อมูลใหม่จะถูกส่งเป็นเอาต์พุต
 - 3) สตริงข้อมูลจะถูกกำหนดให้เป็นสตริงเริ่มต้นอีกครั้งหนึ่ง (ถือว่าเกิดเป็นครั้งแรก)
- แล้วจึงทำ การค้นหา (Search) สตริงข้อมูลใหม่ต่อไปอีก

initialize the CodeBook (one entry for each possible individual data value)

Prefix = empty string

repeat

DataValue = next data value in data stream

if Prefix + DataValue is already in the CodeBook

then Prefix = Prefix + DataValue

else {

add Prefix + DataValue to the CodeBook

output the Prefix code from the CodeBook

Prefix = DataValue

}

until all data values are processed;

output the code from the CodeBook for the last Prefix

รูปที่ 4.2 อัลกอริทึมสำหรับการเข้ารหัสข้อมูลด้วยวิธี LZW

4.1 แบบจำลองการเข้ารหัส

จะขอยกตัวอย่างสตรีมของค่าของข้อมูลและจำลองการเข้ารหัสด้วยวิธี LZW ในตัวอย่างนี้ สมมติว่าค่าของข้อมูลนี้ในการขยายขนาดของข้อมูลกลับคืน จะมีค่าที่เกิดขึ้นเป็น 1 ใน 3 ค่าที่เป็นไปได้ คือ 0 , 1 หรือ 2

สตรีมของข้อมูลเป็น : 0 1 0 1 1 1 1 2
อัลกอริทึมจำลองการเข้ารหัสด้วย LZW
กำหนดค่าเริ่มต้นให้กับตารางรหัส ดังนี้

รหัส	ค่าข้อมูล	ส่วนนำหน้า : ค่าข้อมูล
0	0	-- 0
1	1	-- 1
2	2	-- 2

ส่วนนำหน้า = สตริงว่าง
เริ่มจากการโปรเซสค่าของข้อมูลจากสตรีมของข้อมูล 1 ค่ำ 1 ครั้ง

ค่าข้อมูลอินพุต ส่วนนำหน้า : ค่าของข้อมูล		ตารางรหัสใหม่ที่ได้		เอาต์พุต ส่วนนำหน้า	
		รหัส	สตริงข้อมูล		
0	-- 0	ในตาราง	รหัสของ 0		1
1	0 : 1	3	0 , 1	0	1
0	1 : 0	4	1 , 0	1	0
1	0 : 1	ในตาราง	รหัสของ 3		3
1	3 : 1	5	0 , 1 , 1	3	1
1	1 : 1	6	1 , 1	1	1
1	1 : 1	ในตาราง	รหัสของ 6		6
2	6 : 2	ในตาราง	รหัสของ 6	6	2

เอาต์พุตรหัสส่วนนำหน้าตัวสุดท้าย 2

ตารางที่ 4.2 แบบจำลองอัลกอริทึมในการเข้ารหัส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สังเกตว่า แต่ละครั้งที่รหัสถูกส่งเป็นเอชท์พุท ตารางรหัสจะต้องมีการรับค่าตัวใหม่เข้ามา นั่นแสดงว่า ตัวถอดรหัสจะสร้างตารางรหัสตามข้อมูลที่ได้รับเข้ามาใหม่ทุกครั้งที่ทำกรอ่านรหัสใหม่เข้ามา เพราะฉะนั้น แสดงให้เห็นได้ว่า จะมีเพียงการเกิดสตริงข้อมูลครั้งที่สองเท่านั้น ที่ตัวเข้ารหัสจะส่งรหัสของสตริงข้อมูลนั้นออกไป

อัลกอริทึมของการถอดรหัสข้อมูลที่ถูกลดขนาดด้วยวิธีของ LZW แสดงได้ดังต่อไปนี้ คือ แต่ละครั้งที่ค่าของรหัสถูกส่งเข้ามาจากข้อมูลที่ถูกลดขนาดแล้ว สตริงของข้อมูลของรหัสนั้นก็จะต้องถูกส่งออกเป็นเอชท์พุทออกมาจากเซตของข้อมูลที่ทำการขยายกลับคืน แต่วรหัสใหม่จะสร้างอินพุทตัวใหม่ให้กับตารางรหัสของตัวถอดรหัส ซึ่งตัวถอดรหัสนี้จะทำการสร้างตารางรหัสเอาไว้เช่นเดียวกับตารางรหัสเริ่มต้นของตัวเข้ารหัสด้วยแล้ว (เป็นไปได้ว่าจะทำการอ่านรหัสหนึ่งๆ จากสตรีมของข้อมูลที่ทำการลดขนาดแล้ว แต่ไม่เสร็จสิ้นแค่ตารางรหัสของตัวถอดรหัสเท่านั้น ยังมีกรณีพิเศษอื่นๆ ที่ต้องนำมาพิจารณา ซึ่งจะได้อีกกล่าวในส่วนต่อไป)

```

initialize the CodeBook ( one entry for each possible individual data value )
Code = the first code value in the compressed data
output the " data string " from the CodeBook associated with the Code value
repeat
    OldCode = Code
    Code = the next code value in the compressed data stream
    if the Code already exists in the CodeBook
        then {
            output the data string associated with the Code
            Prefix = OldCode
            Suffix = 1st value from the Code data string
        }
    else {
        Prefix = OldCode
        Suffix = 1st value from the Prefix data string
        output the ( Prefix : Suffix ) data string
    }
    add ( Prefix : Suffix ) to the CodeBook
until all the Code values are processed;
```

รูปที่ 4.3 อัลกอริทึมการถอดรหัสด้วย LZW

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 แบบจำลองการถอดรหัส

กำหนดให้เอาข้อมูลทีลดขนาดแล้วมาจากตัวอย่างข้างต้น แล้วจำลองอัลกอริทึมการถอดรหัส

อินพุตไปยังตัวถอดรหัส (จากตัวอย่างที่แล้ว) : 0 1 3 1 6 2

การจำลองอัลกอริทึมการถอดรหัสด้วย LZW

กำหนดค่าเริ่มต้นให้กับตารางรหัส ดังนี้

รหัส	ค่าข้อมูล	ส่วนนำหน้า : ส่วนต่อท้าย
0	0	- : 0
1	1	- : 1
2	2	- : 2

รหัส - 0 (รหัสข้อมูลแรก)

เอาท์พุท 0 (สดริงข้อมูลจากตารางรหัสที่กำหนดค่าเริ่มต้นแล้ว)

รหัสแต่ละตัวในสตรีนอินพุทเป็น

รหัสเดิม	รหัสอินพุท	มีรหัสนี้อยู่ในตารางรหัสหรือไม่	ส่วนนำหน้า	ส่วนต่อท้าย	เอาท์พุท	ตารางรหัสใหม่		
						ดัชนีรหัส	ค่าข้อมูล	ส่วนนำหน้า : ส่วนต่อท้าย
0	1	มี	0	1	1	3	0,1	0:1
1	3	มี	1	0	0,1	4	1,0	1:0
3	1	มี	3	1	1	5	0,1,1	3:1
1	6	ไม่มี	1	1	1,1	6	1,1	1:1
6	2	มี	6	2	2	7	1,1,2	6:2

ตารางที่ 4.3 การจำลองอัลกอริทึมการถอดรหัสด้วย LZW

สังเกตว่าตัวถอดรหัสจะรวมรหัสใหม่เข้าไปในตารางรหัสของมัน ทุกๆ ครั้งที่รหัสใหม่ถูกอ่านออกมาจากสตรีมของข้อมูลที่ถูกลดขนาดแล้ว (ยกเว้นแต่รหัสแรกเท่านั้น) เพราะฉะนั้นจะเห็นได้ว่าตัวถอดรหัสมักจะช้ากว่าอยู่ 1 ขั้นตอน (1 Step Behind) ตัวเข้ารหัสในการรวมรหัสที่เข้ามาลงในตารางรหัสของมันถ้าทำการเปรียบเทียบ 2 วิธีข้างต้น (จากตัวอย่างการเข้ารหัสและถอดรหัสข้างต้น) เนื่องจากว่าตัวถอดรหัสจะทำงานช้ากว่าตัวเข้ารหัสอยู่ 1 ขั้นตอน จึงเป็นไปได้ที่รหัสถัดไปของสตรีมของข้อมูลที่ถูกลดขนาดแล้วอาจมีหรือไม่มีปรากฏอยู่ในตารางรหัสของตัวถอดรหัสก็ได้

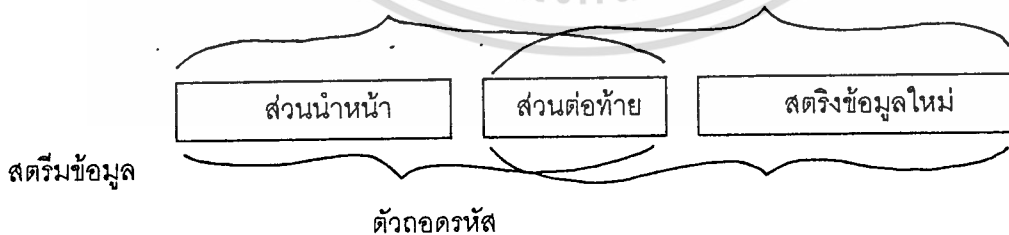
ถ้ารหัสหนึ่งๆ ถูกบรรจุอยู่ในตารางรหัสของตัวถอดรหัสเรียบร้อยแล้ว จากนั้นก็ถูกโปรเซสได้เป็นเอาต์พุตซึ่งเป็นค่าของข้อมูลของรหัสนั้น รหัสก่อนหน้าและค่าของข้อมูลอันดับแรกจากรหัสใหม่ ก็จะถูกรับเข้ามาในตารางรหัสเป็นแพทเทิร์นใหม่ต่อไป

ถ้ารหัสหนึ่งๆ ไม่ได้อยู่ในตารางรหัส คงมีอยู่เหตุผลเดียว คือ ค่าของข้อมูลที่มาก่อนถูกสร้างเป็นสตรีมของข้อมูลแล้ว กรณีนี้ค่าของข้อมูลแรกและข้อมูลสุดท้ายใน 2 สตรีมข้อมูลนั้นจะเหมือนหรือสมมูลกัน เพราะว่าข้อมูลส่วนหน้าและส่วนท้าย (Prefix-Suffix) สมมูลกันกับสตรีมข้อมูลที่ถูกสร้างขึ้นจากรหัสใหม่ และสตรีมข้อมูลใหม่ก็จะถูกเริ่มต้นด้วยค่าส่วนท้าย พิจารณาจากรูปที่ 4.6

ตัวเข้ารหัส

1. หาสตรีมที่ทราบส่วนนำหน้าและส่วนที่ต่อท้ายที่สร้างแพทเทิร์นใหม่
2. เอาท์พุทรหัสสำหรับส่วนนำหน้า
3. รวมเข้าไปในตารางรหัส สตรีมข้อมูลเป็น (ส่วนนำหน้า + ส่วนต่อท้าย)
4. เริ่มต้นหาสตรีมข้อมูลใหม่โดยใช้ส่วนต่อท้ายเท่านั้น

1. หาสตรีมข้อมูลที่เหมือนกับสตรีมข้อมูล ตัวที่แล้ว
 2. เอาท์พุทรหัสสำหรับสตรีมข้อมูลนี้
- หมายเหตุ รหัสนี้จะอยู่ในตารางรหัสของตัวเข้ารหัสเรียบร้อยแล้ว แต่อาจจะยังไม่อยู่ในตารางรหัสของตัวถอดรหัสก็เป็นได้

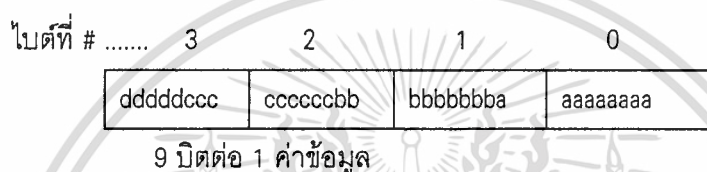


เมื่อตัวถอดรหัสได้รับรหัสที่ยังไม่มีอยู่ในตารางรหัส สตรีมข้อมูลทั้งสองนี้จะสมมูลกัน

รูปที่ 4.4 กรณีพิเศษเมื่อตัวถอดรหัสรับรหัส 1 ตัวที่ไม่ได้อยู่ในตารางรหัสของมัน ค่าข้อมูลส่วนต่อท้ายจึงสมมูลกันระหว่างตัวแรกกับส่วนนำหน้าของตัวที่ตามมา

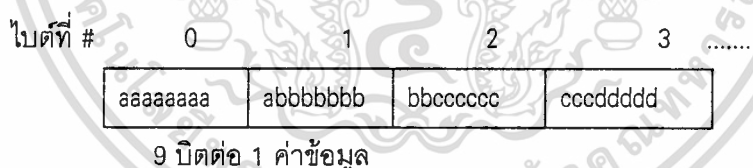
แต่ละขั้นตอนของการเข้ารหัสแบบ LZW ที่แสดงในตารางผู้ใช้ ถ้าค่าของรหัสที่เปลี่ยนแปลงความยาวได้ เพื่อใช้ในการลดขนาดของข้อมูล ในแต่ละกรณีตารางรหัสถูกจำกัดอยู่ที่การรับอินพุต 4096 ค่า (ค่าดัชนี 0-4095) ค่าของรหัส เริ่มจากจำนวนของบิตที่จำเป็นต้องใช้ในการแสดงค่าดัชนีที่มากที่สุด ในกรณีกำหนดตารางรหัสบวกกับอีก 1 บิต ถ้าในขณะที่ทำการสร้างตารางรหัส (ใส่ค่าดัชนี) ตารางได้ใช้ความยาวของรหัสปัจจุบันทั้งหมดแล้วต้องทำการเพิ่มอีก 1 บิต ยกตัวอย่างเช่น หลังจากที่ยอมรับอินพุตที่ 511 เข้ามาในตารางรหัสแล้ว ค่าของข้อมูลจะต้องเพิ่มขึ้นเป็น 10 บิตโดยอัตโนมัติ

การแพก (Packing) บิตเหล่านี้แตกต่างกันระหว่างการใช้งาน เช่น GIF การแพกบิตใช้การแพกตามลำดับของกำลัง (Low Order Portion) ในลักษณะของไบต์ โดยที่ข้อมูลจะถูกแพกเป็นสตริมที่มองจากขวาไปซ้าย ดังรูปที่ 4.5



รูปที่ 4.5 การแพกบิตเป็นไบต์ตามวิธี LZW ของ GIF

การใช้งานอื่นเช่น Postscript และ TIFF การแพกจะเป็นการจัดแบบอันดับสูงมาก่อน (High Order Portion) โดยข้อมูลจะถูกแพกเป็นสตริมโดยมองจากซ้ายไปขวา พิจารณาตามรูปที่ 4.6



รูปที่ 4.6 การแพกบิตเป็นไบต์ตามวิธี LZW ของ PostScript และ TIFF

ตารางรหัสอาจจะเต็มได้ ขึ้นอยู่กับขีดจำกัดของอินพุตขนาด 4096 ในการใช้งานบางชนิดหรือบางแอปพลิเคชัน (Application) จะกำหนดให้มีการล้างรหัส (ClearCode) เพื่อทำการกำหนดสถานะเริ่มต้นใหม่ให้กับตารางรหัสอีกครั้งหนึ่ง เมื่อตารางถูกใช้งานจนเต็มหรือเริ่มจะเต็ม ค่าของการล้างรหัสมักจะมีค่าตามหลังจากตารางของตารางรหัสที่เต็มแล้ว สำหรับ Postscript และ TIFF ซึ่งมักกำหนดขนาดของข้อมูลเป็น 8 บิตต่อไบต์ ค่าของการล้างรหัสจะเป็น 256 โดยที่ใน GIF ค่าการล้างรหัสนี้สามารถเปลี่ยนแปลงค่าได้ ขึ้นอยู่กับขนาดของตารางรหัสที่ได้กำหนดไว้ รวมทั้งในแต่ละการใช้งานจะกำหนดจุดสิ้นสุดข้อมูล (EndofData) ซึ่งมักมีค่าเป็น ค่าการล้างรหัส + 1 ซึ่งจุดสิ้นสุดข้อมูลจะถือเป็นรหัสตัวสุดท้ายที่ถูกส่งด้วยตัวเข้ารหัส

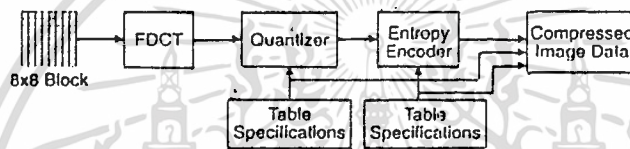
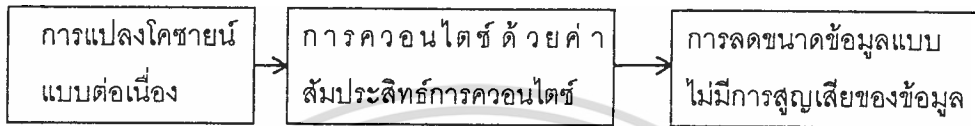
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้ง 037239

บทที่ 5

เจเพก [JPEG (Joint Photographics Experts Group)]

ภาคที่ 1 เจเพกสำหรับภาพขาว - ดำ (เกรย์สเกล)

เจเพกเป็นมาตรฐานการลดขนาดข้อมูลที่สร้างจากผู้เชี่ยวชาญจาก 2 องค์กรมาตรฐาน คือ CCITT และ OSI ซึ่งประกอบด้วยข้อกำหนดต่างๆ ที่เกี่ยวกับทั้งการเข้ารหัสที่มีการสูญเสียและไม่มีการสูญเสียของข้อมูล ขั้นตอนโดยประมาณคือ



รูปที่ 5.1 แสดงการเข้ารหัสแบบเจเพก

ลำดับของการเข้ารหัสแบบเจเพกเป็นไป ตามรูปที่ 5.1 โดยที่ ภาพต้นฉบับจะถูกแบ่งออกเป็น บล็อก (Block) เล็กๆ ขนาด 8×8 พิกเซล ถ้าเป็นภาพที่แสดงผลด้วยจำนวน n บิตต่อพิกเซลแล้ว ค่าข้อมูลของพิกเซลจะมีค่า ตั้งแต่ 0 ถึง $2^n - 1$ ก่อนที่จะผ่าน การแปลงโคไซน์แบบไม่ต่อเนื่อง (Discrete Cosine Transform หรือ DCT) จะมีการสร้างค่าของพิกเซลให้เป็นจำนวนเต็มแบบมีเครื่องหมาย ซึ่งมีค่าตั้งแต่ -2^{n-1} ถึง $2^{n-1} - 1$ ตัวอย่างเช่นภาพแบบเกรย์สเกลที่มี 8 บิตต่อพิกเซล (ความเข้ม $2^8 = 256$ ระดับ) มีค่า 0 - 255 เมื่อจะผ่านการแปลงต้องทำข้อมูลให้เป็น -128 ถึง 127 ก่อน

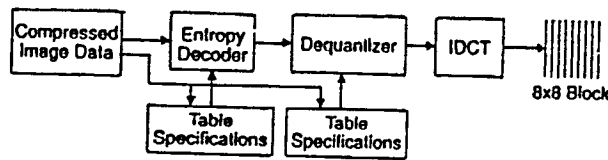
เมื่อผ่านการแปลงโคไซน์แบบไม่ต่อเนื่อง แล้วเอาที่พิกเซลที่ได้ จะเป็นค่าสัมประสิทธิ์ทางความถี่ (DCT Coefficient) ต่อมาจะเป็นการควอนไทซ์ค่าสัมประสิทธิ์เหล่านี้ ในขั้นตอนนี้อะไรที่จะมีการลดขนาดของข้อมูลและเกิดความผิดพลาดจากการควอนไทซ์ขึ้น (Quantizing Error) ในการควอนไทซ์นี้จะเป็นแบบไม่เป็นยูนิฟอร์ม (Non-Uniform) กล่าวคือ ค่าสัมประสิทธิ์แต่ละค่า จะถูกควอนไทซ์ด้วยระดับการควอนไทซ์ (Step Size) ที่ไม่เท่ากันเสมอ ขนาดของระดับขั้นหรือสเตป (Step) ในการควอนไทซ์จะถูกกำหนดโดยค่าในตารางที่ 1 เรียกว่าเป็นตารางค่าควอนไทซ์นั่นเอง ซึ่งค่าในตารางนี้จะเป็นค่าที่ใช้เฉพาะกับสัมประสิทธิ์ที่ได้จากการแปลงโคไซน์แบบไม่ต่อเนื่องแต่ละค่าเท่านั้น

ค่าสัมประสิทธิ์ที่ถูกควอนไทซ์แล้วจะถูกนำมาจัดตามลำดับซิกแซก เพื่อความสะดวกในการเข้ารหัสซึ่งอาจใช้ วิธีของฮัฟแมน (Huffman Encoding) หรือ วิธีการเข้ารหัสแบบความยาวของรหัสไม่คงที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(Variable Length Encoding) ซึ่งในตารางที่ 2 จะเป็นตารางที่ใช้คู่กับการเข้ารหัส (อธิบายในตอนท้าย)
เอาท์พุทจากกระบวนการนี้จะเป็นข้อมูลภาพที่ถูกลดขนาดเรียบร้อยแล้ว

การถอดรหัสของข้อมูลก็ทำในลักษณะตรงกันข้าม แสดงตามรูปที่ 5.2



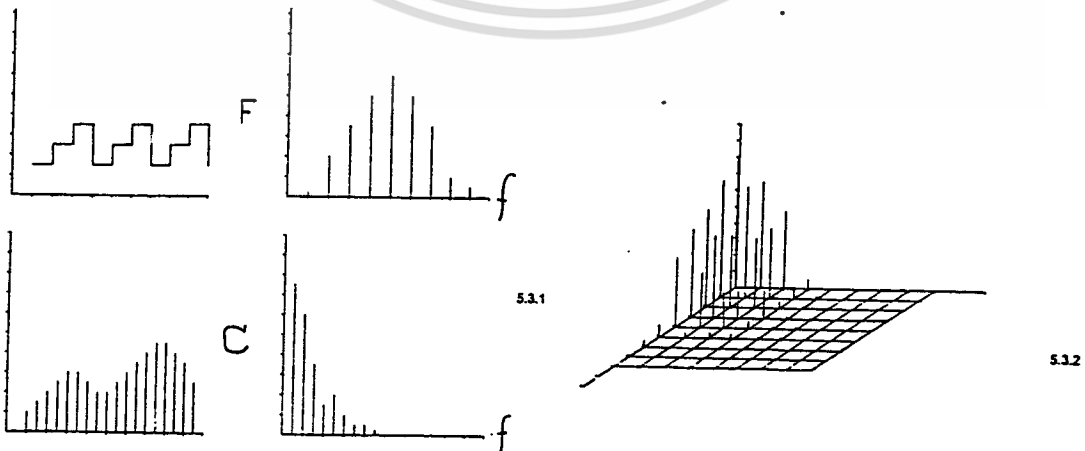
รูปที่ 5.2 แสดงการถอดรหัสข้อมูลด้วยวิธีการของเจเพก

หมายเหตุ

ภาพขาว-ดำที่เป็นเกรย์สเกล 256 ระดับ เมื่อผ่าน การแปลงโคไซน์แบบไม่ต่อเนื่องแล้ว ค่าของสัมประสิทธิ์ทางความถี่จะมีค่าตั้งแต่ -1024 ถึง 1023 จึงต้องใช้จำนวนบิตเพิ่มขึ้นจากเดิม 8 บิต เป็น ขนาด 11 บิตเลยทีเดียว

5.1 การแปลงโคไซน์แบบไม่ต่อเนื่อง

สิ่งสำคัญตามมาตรฐานที่เจเพกต้องการคือ การแปลงโคไซน์แบบไม่ต่อเนื่อง ซึ่งจะอธิบาย [REDACTED] ที่นี้ เริ่มจากพิจารณารูปที่ 5.3.1 ที่เป็นการเปรียบเทียบระหว่างการแปลงโดเมนของเวลาของสัญญาณใดๆ เป็นโดเมนของความถี่ โดยใช้การแปลงฟูเรียร์ (Fourier Transform) และ การแปลงโคไซน์แบบไม่ต่อเนื่อง จะเห็นได้อย่างชัดเจนว่า ทั้งสองวิธีจะมีการรวบรวม (Compact) ส่วนประกอบที่สำคัญของข้อมูลไว้ได้ เหมือนกัน แต่ที่ความถี่ต่างกัน กล่าวคือ การแปลงโคไซน์แบบไม่ต่อเนื่องจะสามารถเก็บส่วนประกอบส่วนใหญ่ของภาพไว้ที่ความถี่เท่ากับ 0 ส่วนการแปลงฟูเรียร์ ความถี่ที่รวบรวมข้อมูลส่วนใหญ่ที่สำคัญของภาพ จะไม่ได้อยู่ที่ความถี่เท่ากับ 0 และในรูปที่ 2 แสดงถึง การแปลงโคไซน์แบบไม่ต่อเนื่องแบบ 2 ทิศทางเมื่อ มีการแบ่งภาพออกเป็นบล็อกขนาด 8×8 พิกเซล

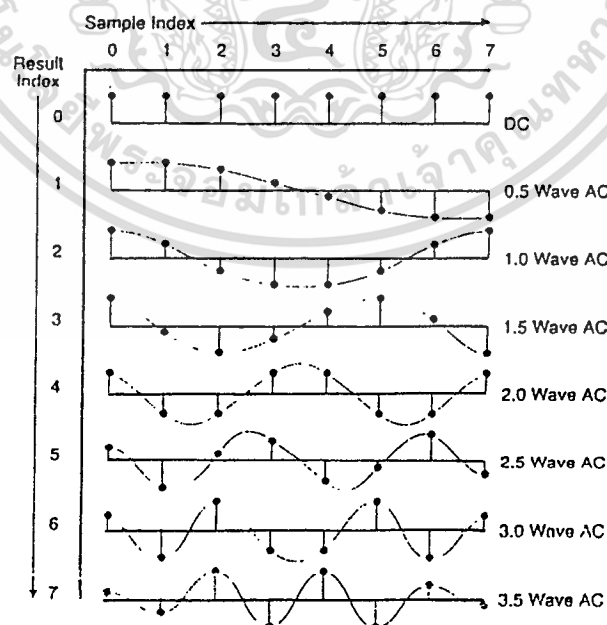


รูปที่ 5.3 แสดงการเปรียบเทียบการแปลงด้วย ฟูเรียร์กับการแปลงโคไซน์แบบไม่ต่อเนื่อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแปลงโคไซน์แบบไม่ต่อเนื่อง เป็นการประมวลผลทางคณิตศาสตร์ที่เป็นการคำนวณส่วนประกอบทางความถี่ ของแอมพลิจูดของสัญญาณที่อัตราการแซมปลิงใดๆ ซึ่ง การแปลงโคไซน์แบบไม่ต่อเนื่องนี้จะต้องประยุกต์ใช้กับจำนวนของแอมพลิจูดที่รู้ค่าแน่นอน ในทางคณิตศาสตร์สามารถพิจารณาได้ง่ายๆ คือ ถ้าจำนวนที่ใช้เป็นกำลังคู่ของ 2 หลักการของ การแปลงโคไซน์แบบไม่ต่อเนื่องจะอธิบายได้ง่ายขึ้น สำหรับ การแปลงโคไซน์แบบไม่ต่อเนื่องแบบทิศทางเดียวใช้ในการเปลี่ยน (Convert) อาร์เรย์ (Array) หนึ่งของจำนวนที่แสดงค่าแอมพลิจูด (Amplitude) ของสัญญาณ ณ จุดที่ทราบค่าใดๆ บนโดเมนของเวลาไปเป็นอีกอาร์เรย์หนึ่ง ซึ่งใช้แสดงค่าแอมพลิจูดที่เป็นคอมโพเนนต์ (Component) ทางความถี่ของสัญญาณที่เป็นอินพุต (Input) อาร์เรย์ผลลัพธ์ จะมีอีเลเมนต์ (Element) แรกเป็นค่าเฉลี่ยของทุกๆ แอมพลิจูดของอาร์เรย์อินพุต เรียกว่าเป็นสัมประสิทธิ์กระแสตรง (DC Coefficient) และอีเลเมนต์ที่เหลือก็เริ่มมีความถี่เข้ามาเกี่ยวข้อง จึงเรียกว่า มีสัมประสิทธิ์กระแสสลับ (AC Coefficient) ที่เป็นค่าเฉพาะเป็นของแต่ละอีเลเมนต์ (ค่าสัมประสิทธิ์ คือ แอมพลิจูดของคอมโพเนนต์ทางความถี่)

ความถี่ ที่ได้แสดงโดยแต่ละอีเลเมนต์ ในอาร์เรย์ผลลัพธ์จะเป็นฟังก์ชันของค่าดัชนีอาร์เรย์ (Array Index) สำหรับอีเลเมนต์นั้น สารบัญทางความถี่ของเซตของแอมพลิจูดที่แต่ละความถี่จะถูกคำนวณโดยการถ่วงน้ำหนักค่าเฉลี่ย (Weighted Average) ให้กับเซตที่นำมาคิด ค่าในการถ่วงน้ำหนักสำหรับแต่ละแอมพลิจูดอินพุต หาโดยการคูณดัชนีปัจจุบันของอาร์เรย์ผลลัพธ์กับค่าคงที่ π (pi) และดัชนีของแอมพลิจูดอินพุต วิธีนี้จะมี ผลต่อการสร้างลำดับของสัมประสิทธิ์การถ่วงน้ำหนัก โดยที่ประมาณค่าของคลื่นโคไซน์ (Cosine) ที่ความถี่เป็นสัดส่วนกับดัชนีของอาร์เรย์ผลลัพธ์ ในรูป 5.4 แสดงความสัมพันธ์ในการประมาณค่าระหว่างสัมประสิทธิ์การถ่วงน้ำหนักของแต่ละแอมพลิจูดอินพุต สำหรับแต่ละค่าดัชนีของอาร์เรย์ผลลัพธ์ ในการแปลงโคไซน์แบบไม่ต่อเนื่อง 1 ทิศทาง บนอาร์เรย์ของ 8 แอมพลิจูดอินพุต



รูปที่ 5.4 แสดงค่าสัมประสิทธิ์การถ่วงน้ำหนักสัมพัทธ์ของการแปลงโคไซน์แบบไม่ต่อเนื่อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่าถ่วงน้ำหนักสูงสุดสำหรับแอมเปิ้ลใดๆ จะมีค่าเป็น 1 ซึ่งจะมีค่าอยู่ในช่วง -1 ถึง 1 พิจารณาจากรูปที่ 5.4 สามารถพิจารณาได้ว่าแต่ละสเปค ของดัชนีอาร์เรย์ผลลัพธ์ จะขึ้นอยู่กับค่าที่เพิ่มขึ้นทีละครึ่งไซเคิล (Cycle) ในคอมโพเนนท์ทางความถี่ที่สัมพันธ์กัน สำหรับอาร์เรย์ของ 8 แอมเปิ้ล ความถี่สูงสุด 3.5 ไซเคิล ซึ่ง มีค่าใกล้เคียงกับความถี่สูงสุดที่สามารถแสดงออกมาได้ นั่นคือ 4 ไซเคิลหรือครึ่งเท่าของความถี่ในการแอมเปิ้ล ตารางที่ 5.1 แสดงค่าสัมประสิทธิ์ที่ใช้ในการคำนวณ ค่าเฉลี่ยถ่วงน้ำหนัก สำหรับแต่ละคอมโพเนนท์ทางความถี่ สมการที่ใช้ในการคำนวณแสดงตามสมการที่ 5.1

(8)
3)
0)

ดัชนีผลลัพธ์ ดัชนีของแอมเปิ้ล→ ↓	0	1	2	3	4	5	6	7
0	+0.707	+0.707	+0.707	+0.707	+0.707	+0.707	+0.707	+0.707
1	+0.981	+0.0831	+0.556	+0.195	-0.195	-0.556	-0.831	-0.981
2	+0.924	+0.383	-0.383	-0.924	-0.924	-0.383	+0.383	+0.924
3	+0.831	-0.195	-0.981	-0.556	+0.556	+0.981	+0.195	-0.831
4	+0.707	-0.707	-0.707	+0.707	+0.707	-0.707	-0.707	+0.707
5	+0.556	-0.981	+0.195	+0.831	-0.831	-0.195	+0.981	-0.556
6	+0.383	-0.924	+0.924	-0.383	-0.383	+0.924	-0.924	+0.383
7	+0.195	-0.556	+0.831	-0.981	+0.981	-0.831	+0.556	-0.195

ตารางที่ 5.1 แสดงค่าสัมประสิทธิ์การถ่วงน้ำหนักแอมเปิ้ลสำหรับการแปลงโคไซน์แบบไม่ต่อเนื่อง แบบ 1 ทิศทาง

$$\text{Coeff}(k,m) = C(k) \cos \left[\frac{(2m+1)k\pi}{2N} \right] \quad \dots\dots (5.1)$$

$$\begin{aligned} \text{เมื่อ } C(k) &= \frac{1}{\sqrt{2}} \quad \text{เมื่อ } k = 0 \\ &= 1 \quad \text{เมื่อ } k \neq 0 \\ \{ k &= \text{ดัชนีของอาร์เรย์ผลลัพธ์} \} \\ \{ m &= \text{ดัชนีของแอมเปิ้ลอาร์เรย์} \} \\ \{ n &= \text{ขนาดของแอมเปิ้ลอาร์เรย์} \} \end{aligned}$$

× 12.7

สมการที่ 5.1 นี้เป็นสมการการหาค่าสัมประสิทธิ์การถ่วงน้ำหนัก ซึ่งสามารถตรวจสอบได้โดยเมื่อดัชนีของอาร์เรย์ผลลัพธ์เป็น 0 ทำให้ $\cos 0 = 1$ แฟกเตอร์การถ่วงน้ำหนักจะมีค่าที่เท่ากันทุกแอมเปิ้ล ถ้าย้ายสมการนี้เพื่อสร้างค่าเฉลี่ยในการถ่วงน้ำหนักผลลัพธ์ของสมการแสดงตามสมการที่ 5.2 ซึ่งเป็นสมการสำหรับการแปลงโคไซน์แบบไม่ต่อเนื่อง 2 ทิศทางบนอาร์เรย์ของแอมเปิ้ลที่มีขนาด N

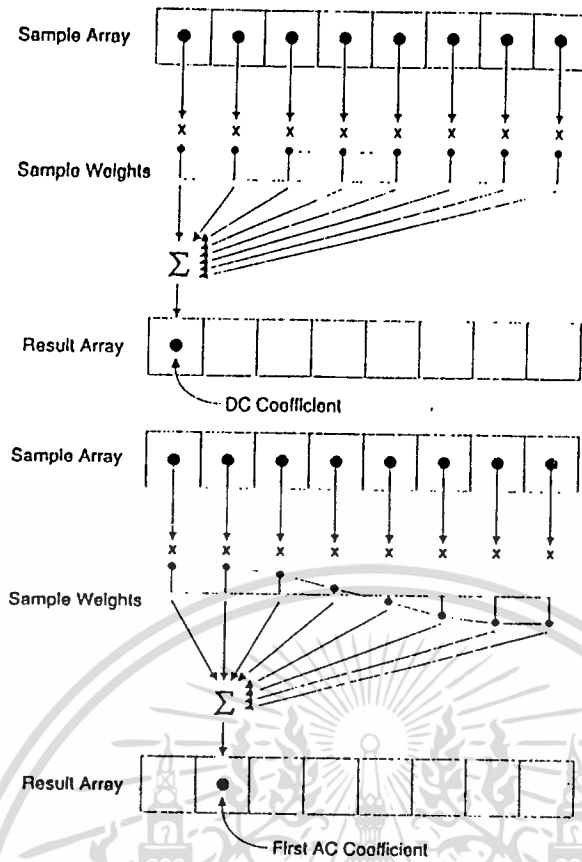
$$C(i,j) = \frac{1}{\sqrt{2}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \text{pixels}(x,y) \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right] \quad \text{.....(5.2)}$$

$$C(x) = \frac{1}{\sqrt{2}} \quad \text{เมื่อ } x = 0$$

$$= 1 \quad \text{เมื่อ } x > 0$$

ในสมการที่ 5.2 นี้เป็นสมการสำหรับการแปลงโคไซน์แบบไม่ต่อเนื่อง 2 ทิศทาง ถือได้ว่าอาจไม่มีความจำเป็น ที่จะต้องทำการคำนวณสมการดังกล่าว สำหรับแต่ละอีเลเมนต์ในเซตของแอมเปิ้ล ถ้า การแปลงโคไซน์แบบไม่ต่อเนื่องจะถูกคำนวณสำหรับหลายๆ แอมเปิ้ล เซตของแมตริกซ์ของค่าสัมประสิทธิ์ที่แสดงตามตารางที่ 5.1 สามารถคำนวณเพียงครั้งเดียว แล้วนำค่านี้ไปใช้กับเซตที่ตามมาดังในรูปที่ 5.5 แสดงวิธีการที่สัมประสิทธิ์การถ่วงน้ำหนัก ที่ใช้ในการคำนวณสัมประสิทธิ์กระแสตรงและสัมประสิทธิ์กระแสสลับค่าแรกเมื่อเป็นการคำนวณ การแปลงโคไซน์แบบไม่ต่อเนื่อง สำหรับเซตของ 8 แอมเปิ้ล

ครั้งหนึ่งที่เซตของแอมเปิ้ลจะถูกแปลงไปเป็นเซตของสัมประสิทธิ์ทางความถี่ ซึ่งสามารถเลือกทำการควอนไทซ์เพื่อลดขนาดของเซตของข้อมูล ในขณะที่ยังคงมีข้อมูลที่สำคัญไว้ นั่นคือถ้าสัมประสิทธิ์ทางความถี่ถูกควอนไทซ์ เซตของข้อมูลต้นฉบับจะไม่สามารถสร้างข้อมูลคืนได้อย่างสมบูรณ์ การสร้างข้อมูลจะถูกจัดอยู่ในรูปแบบที่ง่าย ๆ โดยการรวมผลของสัมประสิทธิ์ทางความถี่และสัมประสิทธิ์การถ่วงน้ำหนักของแต่ละแอมเปิ้ล สมการที่ใช้ในการสร้างข้อมูลคืน เรียกว่า การแปลงกลับโคไซน์แบบไม่ต่อเนื่อง (Inverse DCT) หาได้ตามรูปสมการที่ 5.3 สำหรับ การแปลงโคไซน์แบบไม่ต่อเนื่อง 1 ทิศทาง (อาจจะเรียกได้อีกอย่างหนึ่งว่า การแปลงโคไซน์แบบไม่ต่อเนื่องไปข้างหน้า (Forward DCT))



รูปที่ 5. การหาค่าสัมประสิทธิ์การถ่วงน้ำหนักของแฮมเบิล

$$X(m) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} X(k) C(k) \cos \left[\frac{(2m+1)k\pi}{2N} \right] \quad \text{.....(5.3)}$$

$$\text{เมื่อ } C(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{เมื่อ } k = 0 \\ 1 & \text{เมื่อ } k \neq 0 \end{cases}$$

สมการที่ 5.3 นี้เป็นสมการที่ใช้ในการแปลงกลับ โคไซน์ แบบไม่ต่อเนื่อง 1 ทิศทาง

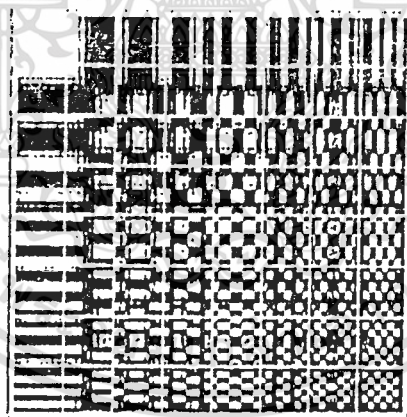
การแปลงโคไซน์แบบไม่ต่อเนื่องแบบ 2 ทิศทาง

จากการแปลงโคไซน์แบบไม่ต่อเนื่อง ที่ได้กล่าวถึงไปแล้วนั้นสามารถอธิบาย การแปลงโคไซน์แบบไม่ต่อเนื่อง 2 ทิศทางได้โดย เซตของข้อมูลโดยทั่วไปของโอเพอร์เรชั่น (Operation) นี้คืออาร์เรย์ของพิกเซล 2 มิติ ถ้าเราพิจารณาจากอาร์เรย์ของพิกเซลขนาด 8×8 พิกเซล การแปลงโคไซน์แบบไม่ต่อเนื่อง 1 ทิศทางจะถูกนำมาใช้แยกกันสำหรับแต่ละแถวและแต่ละหลัก ผลลัพธ์จะได้เป็น 8 แถวของสัมประสิทธิ์ทางความถี่ ถ้าค่าเหล่านี้ถูกนำไปใช้เช่นเดียวกันในลักษณะเป็นคอลัมน์ คอลัมน์แรกก็จะได้เป็นสัมประสิทธิ์กระแสดตรงล้วนๆ คอลัมน์ถัดมาก็จะได้สัมประสิทธิ์กระแสดลบค่าแรกของแต่ละแถว ไปเรื่อยๆ เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จึงได้ผลลัพธ์เป็นสัมประสิทธิ์ทางความถี่ 2 มิติ ขนาด 8 แถว x 8 หลัก และมีค่าเฉพาะสำหรับแต่ละค่าของบล็อกพิกเซลอินพุท จะเห็นได้ทันทีที่เลยว่าอีเลเมนต์ที่อยู่มุมบนซ้ายมือจะเป็นสัมประสิทธิ์กระแสตรงตัวเดียวเท่านั้น ส่วนอีเลเมนต์อื่นจะมีส่วนของความถี่เข้ามาเกี่ยวข้องด้วย แต่ละสัมประสิทธิ์ทางความถี่นี้เมื่อเรานำมาผ่าน การแปลงกลับโคไซน์ แบบไม่ต่อเนื่องแบบ 2 ทิศทางแล้ว จะได้ค่าแพทเทิร์นที่ต่างกันของพื้นที่ ที่มีความขาวดำด้วยอัตราความคมชัด (Contrast Ratio) ซึ่งกำหนดโดยแอมพลิจูดของสัมประสิทธิ์ในแสดงรูปที่ 5.6 แสดงถึงแพทเทิร์น ที่สร้างจากค่าสัมประสิทธิ์ทางความถี่ของ การแปลงโคไซน์แบบไม่ต่อเนื่องของ 8 x 8 พิกเซล สมมติว่าแต่ละค่าเป็นค่าสูงที่สุดแล้ว

หมายเหตุ

จะเห็นว่าบล็อกบนซ้ายมือจะเป็นสีเทาทึบ แสดงถึงค่าเฉลี่ยของความเข้มทั้งบล็อกขนาด 8 x 8 พิกเซล ส่วนบล็อกที่อยู่มุมล่างขวามือจะมีลักษณะเป็นตารางหมากรุก เพราะมีส่วนคอมโพเนนท์ทางความถี่ที่สูงสุดทั้งตามแนวตั้งและแนวนอน บล็อกที่มุมขวามือด้านบนและซ้ายมือด้านล่าง แสดงแถบตามแนวนอนและแนวตั้งตามลำดับ เพราะมีความถี่สูงสุดทางแนวนอนแต่ต่ำสุดตามแนวตั้งและความถี่สูงสุดทางแนวตั้งแต่ต่ำสุดทางแนวนอน ตามลำดับ สมการรวมของ การแปลงโคไซน์แบบไม่ต่อเนื่อง 2 ทิศทางแสดงตามสมการที่ 5.4



รูปที่ 5.6 แสดงแพทเทิร์นของความเข้มของสัมประสิทธิ์การถ่วงน้ำหนัก การแปลงโคไซน์แบบไม่ต่อเนื่อง

$$\text{Forward DCT } F[u,v] = \frac{1}{N} C_u C_v \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f[m,n] \cos\left[\frac{(2m+1)u\pi}{2N}\right] \cos\left[\frac{(2n+1)v\pi}{2N}\right] \dots\dots (5.4)$$

$$\text{Inverse DCT } f[m,n] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} C_u C_v F[u,v] \cos\left[\frac{(2m+1)u\pi}{2N}\right] \cos\left[\frac{(2n+1)v\pi}{2N}\right]$$

เมื่อ C_u และ $C_v = 1$ ที่ $u,v = 0$ (สัมประสิทธิ์กระแสดตรง)

C_u และ $C_v = 2$ ที่ $u,v \neq 0$

$F[u,v]$ = เซลเป้าหมายสำหรับสัมประสิทธิ์การแปลงโคไซน์แบบต่อเนื่อง

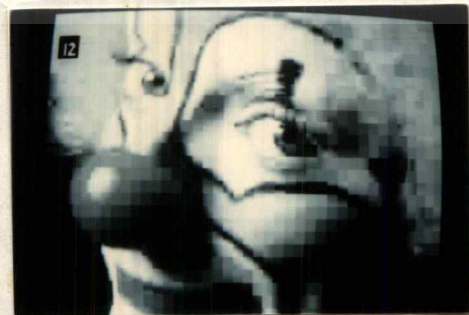
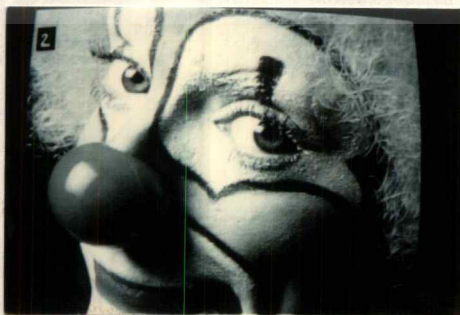
$f[m,n]$ = เซลเป้าหมายสำหรับพิกเซลต้นฉบับและที่สร้างขึ้น

สมการที่ 5.4 นี้เป็นการแปลงและการแปลงกลับโคไซน์แบบไม่ต่อเนื่อง

สมการนี้แสดงถึงผลรวมของเทอมของโคไซน์สำหรับทิศทางในแนวนอนและแนวตั้ง สำหรับแต่ละอีเลเมนต์ในอาร์เรย์ของผลลัพธ์ แสดงสมการรวมการแปลงโคไซน์แบบไม่ต่อเนื่อง 1 ทิศทาง 2 สมการตาม 2 แนว ทั้งในแนวตั้งและแนวนอน

ประสิทธิภาพในการคำนวณ

การแปลงพิกเซล 2 มิติไปเป็นคอมพิวเตอร์ทางความถี่ไม่ได้เกิดกระบวนการลดขนาดของข้อมูลแต่อย่างใด ซึ่งข้อดีที่จะได้จากการแปลงโคไซน์แบบไม่ต่อเนื่อง คือ สามารถเลือกทำการควอนไทซ์คอมพิวเตอร์ทางความถี่ภายในบล็อกได้ โดยที่พิกเซลที่เป็นอินพุตนั้น ไม่ได้มีกฎเกณฑ์หรือข้อจำกัดใดๆ ที่จะต้องกำหนดขนาดของอาร์เรย์ 2 มิติของพิกเซลที่จะนำมาแปลง จำนวนครั้งของการคูณกันสำหรับแต่ละอีเลเมนต์ ของอาร์เรย์ผลลัพธ์ในการแปลงแบบ 1 ทิศทางจะเป็นอัตราส่วนกำลังสองของจำนวนอีเลเมนต์ในแฮมเปิ้ลอาร์เรย์นั้น บนออร์เดอร์ (Order) ของ $[N \log_2(N)]^2$ สำหรับการแปลง 2 ทิศทาง จะเป็นอัตราส่วนกับผลรวมของกำลังสองของทั้งสองทิศทางของแฮมเปิ้ลอาร์เรย์ การแบ่งภาพต้นฉบับออกเป็นบล็อกของพิกเซลที่ขนาดเล็กลงจะทำให้จำนวนครั้งในการคำนวณลดลงเป็นอย่างมาก แต่ข้อเสียที่อาจได้รับคือ จากกระบวนการสร้างภาพกลับคืนอาจมองเห็นผลต่างของแต่ละบล็อกได้อย่างชัดเจน

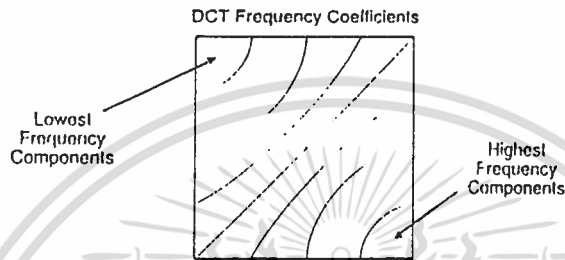


รูปที่ 5.7 แสดงตัวอย่างภาพต้นฉบับและภาพที่ได้จากกระบวนการของเจเพก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในทางปฏิบัติ ดีกรีในการควอนไทซ์ ที่ใช้กับคอมโพเนนต์ทางความถี่นั้น ขึ้นอยู่กับจำนวนของแฟกเตอร์ที่รวมไว้ในคุณลักษณะของภาพ อัตราการลดขนาดข้อมูลที่ต้องการ และคุณภาพของภาพที่ต้องการหลังจากทำการสร้างภาพกลับคืน สำหรับภาพสีจะทำการควอนไทซ์ส่วนที่เป็นสีมากกว่าส่วนของความสว่างและเมื่อมีการคำนวณ การแปลงโคไซน์แบบไม่ต่อเนื่องให้กับบล็อกของภาพมักใช้เลขจำนวนเต็มมากกว่าทศนิยมเพราะใช้เวลาในการคำนวณและความยุ่งยากในการคำนวณน้อยกว่า

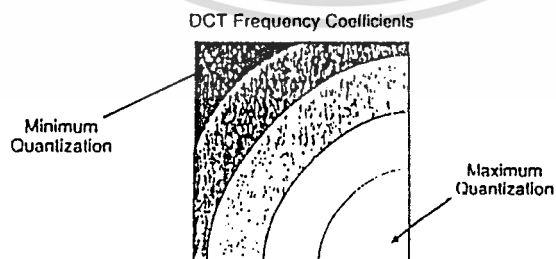
ในรูปที่ 5.8 จะแสดงถึงการกระจายของข้อมูล (คอมโพเนนต์ทางความถี่) ที่เป็นผลลัพธ์จากการแปลงโคไซน์แบบไม่ต่อเนื่อง 2 ทิศทาง



รูปที่ 5.8 การกระจายของส่วนประกอบทางความถี่ (ประมาณ) ของการแปลงโคไซน์แบบต่อเนื่อง 2 ทิศทาง

5.2 การควอนไทซ์

การควอนไทซ์ จะเป็นการลดจำนวนบิตที่จำเป็นที่จะใช้เก็บเลขจำนวนเต็ม ของค่าสัมประสิทธิ์ที่ได้จากการแปลงโคไซน์แบบไม่ต่อเนื่อง ซึ่งจะทำให้เป็นแมตริกซ์ โดยที่ทุกๆ อีเลเมนต์ในแมตริกซ์จะมีค่าที่ใช้ในการควอนไทซ์เป็นค่าเฉพาะ เรียกว่าค่าสัมประสิทธิ์การควอนไทซ์หรือค่าควอนตัม (Quantum) ลักษณะเป็นค่าที่มากขึ้นตามแนวเส้นทแยงมุม ตามรูปที่ 5.9 หรือเป็นแบบซิกแซกโดยประมาณ โดยที่ค่าควอนตัมจะเป็นค่าที่บอกขนาดของสเตป ในการควอนไทซ์ ซึ่งส่วนใหญ่มีค่า 1 - 256



รูปที่ 5.9 แสดงการกระจายค่าของสัมประสิทธิ์ที่ใช้ในการควอนไทซ์

อีเลเมนต์ที่อยู่ด้านมุมบนซ้ายมือ จะมีส่วนประกอบสำคัญของภาพ จะถูกควอนไทซ์ด้วยสเตปแคบๆ (ให้ค่าผิดพลาด (Error) น้อย) เพราะต้องการความถูกต้องและความละเอียดมาก ส่วนที่เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาต
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตำแหน่งที่ห่างออกไปจากจุดนี้ตามแนวทแยงมุมค่าสเต็ปจะกว้างขึ้น เพื่อปิดค่าส่วนที่ไม่สำคัญของภาพให้ใกล้เคียงหรือเท่ากับ 0 และที่มุมขวาล่างจะมีสเต็ปกว้างที่สุด เพราะค่าผิดพลาดที่เกิดขึ้นมีผลต่อภาพน้อย

$$\text{ค่าที่ได้จากการควอนไทซ์(Quantized value) (i,j)} = \frac{\text{ค่าที่ได้จากการแปลงโคไซน์แบบไม่ต่อเนื่อง(DCT(i,j))}}{\text{ค่าควอนตัม(i,j)}}$$

ในกระบวนการของการดีควอนไทซ์จะทำการหาค่าสัมประสิทธิ์ทางความถี่โดย

$$\text{ค่าที่ได้จากการแปลงโคไซน์แบบไม่ต่อเนื่อง (DCT (i,j))} = \text{ค่าที่ได้จากการควอนไทซ์ (i,j)} \times \text{ค่าควอนตัม (i,j)}$$

การเลือกเมตริกซ์ที่จะนำมาทำการควอนไทซ์

ค่าควอนตัมมีค่ามากจะสามารถลดขนาดข้อมูลได้มากแต่ภาพจะผิดเพี้ยนมาก ในขณะที่ค่าควอนตัมน้อยๆ จะลดขนาดได้น้อยแต่ผิดเพี้ยนน้อย เพื่อให้คุณภาพของภาพยืดหยุ่นได้ตามความต้องการของผู้ใช้จึงมีการกำหนดค่าแฟกเตอร์คุณภาพ (Quality Factor) ปกติมีค่า 1 - 25 (ค่าที่มากกว่านี้ภาพจะเพี้ยนมากเกินไปจะยอมรับได้) ซึ่งค่านี้จะถูกนำไปคำนวณเพื่อเลือกค่าควอนตัมที่เหมาะสมในการควอนไทซ์ ตามคุณภาพที่ต้องการ

ค่าแฟกเตอร์คุณภาพจะมีความสัมพันธ์กับค่าควอนตัมตามแนวทแยงมุมของเมตริกซ์แสดงตามรูปที่ 5.10 โดยที่สมการที่ใช้ในการคำนวณค่าควอนตัมจากค่าแฟกเตอร์คุณภาพคือ

$$\text{ค่าควอนตัม (i,j)} = 1 + [(1 + i + j) * \text{ค่าแฟกเตอร์คุณภาพ}]$$

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

รูปที่ 5.10 แสดงเมตริกซ์ของค่าควอนตัมที่แฟกเตอร์คุณภาพเท่ากับ 2

การเข้ารหัสด้วยวิธีความต่อเนื่องของข้อมูล

1. สัมประสิทธิ์กระแสตรง

เนื่องจากเราจะทำการใส่รหัส สัมประสิทธิ์กระแสตรงด้วยผลต่างของ สัมประสิทธิ์กระแสตรงปัจจุบันกับค่าก่อนหน้า ดังนั้นค่าที่บอกถึงผลต่างก็จะมีค่าตั้งแต่ - 2048 ถึง 2047 การเข้ารหัสจึงประกอบด้วย 2 ส่วน

1.1 SIZE หมายถึง จำนวนบิตที่จะต้องใช้ในการใส่ค่าข้อมูล (ผลต่าง)

1.2 AMPLITUDE หมายถึง ค่าของผลต่าง

รหัสเป็น (SIZE) (AMPLITUDE)

2. สัมประสิทธิ์กระแสสลับ

เนื่องจาก คำนี้นี้ได้จากการควอนไทซ์แล้วประมาณค่าได้ใกล้เคียงค่า 0 พอจัดลำดับซิกแซกแล้วก็จะได้ลำดับค่าที่มีค่าและไม่มีค่าบ้าง ส่วนท้ายๆ ของการจัดจะเป็น 0 ที่ต่อเนื่องกันเป็นส่วนใหญ่ การเข้ารหัสประกอบด้วย 3 ส่วนคือ

2.1 RUNLENGTH คือ จำนวนค่า 0 ที่ต่อเนื่องกัน

2.2 SIZE คือ จำนวนบิตที่จะต้องใช้ในการใส่ค่าข้อมูลที่ตามมา (ไม่เท่ากับ 0)

2.3 AMPLITUDE คือ ค่าข้อมูลที่ไม่ใช่ 0 นั้นเอง

รหัสเป็น (RUNLENGTH , SIZE) (AMPLITUDE)

หมายเหตุ

มีเงื่อนไขพิเศษคือ ถ้าเกิดมี 0 ต่อเนื่องกันมากกว่า 16 ตัวสมมติเป็น 22 ตัว จะได้รหัสเป็น

(15 , 0) (6 , 4) (13)

(15 , 0) ความหมายคือ มี 0 ต่อเนื่องกัน 16 ตัว (เป็นรูปแบบที่กำหนดตายตัว)

(6 , 4) (13) ความหมายคือ - มี 0 อีก 6 ตัว

- ใช้ 4 บิตในการแทนค่าข้อมูลที่มีค่าเท่ากับ 13

ตารางที่ 5.2 แสดงข้อมูลที่จำนวนบิตที่ใช้ต่างๆ กัน เรียกว่าเป็นรหัสแบบความยาวของรหัสไม่คงที่หรือ (Variable Length Code) ซึ่งได้จากการสร้างตารางของฮัฟแมนสำหรับแต่ละคอมโพเนนต์ของภาพ ซึ่งค่าของข้อมูลเป็น จำนวนเต็มแบบความยาวไม่คงที่ (Variable Length Integer)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SIZE	ช่วงของค่า AMPLITUDE
1	(-1 , 1)
2	(-3 , -2) (2 , 3)
3	(-7...-4) (4...7)
4	(-15...-8) (8...15)
5	(-31...-16) (16...31)
6	(-63...-32) (32...63)
7	(-127...-64) (64...127)
8	(-255...-128) (128...255)
9	(-511...-256) (256...511)
10	(-1023...-512) (512...1023)

ตารางที่ 5.2 แสดงช่วงของค่าข้อมูลที่จำนวนบิตต่างกัน

97	97	97	102	102	102	102	102
88	88	97	97	97	97	102	97
88	88	97	88	88	97	88	88
88	88	88	88	88	88	88	88
88	92	88	88	88	88	88	88
88	88	83	88	82	88	88	88
88	83	80	86	88	86	86	86
88	80	83	83	80	80	83	88

ตารางที่ 5.3 แสดงค่าพิทเชลของภาพต้นฉบับ

+2653	-47	-20	-16	-14	0	0	-10
+ 166	-4	-14	+10	-7	+11	+14	+12
+43	-23	+1	0	+21	+12	0	+6
+45	-13	-3	+2	+2	-7	-29	-16
+8	+12	+12	-6	+12	-7	-8	+1
+2	+11	+1	+21	+12	-4	+1	-4
-5	+15	+10	-3	-11	-1	+18	-13
+4	+7	-11	+3	-9	-14	+7	0

ตารางที่ 5.4 แสดงค่าสัมประสิทธิ์ทางความถี่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8
8	8	8	8	256	256	256	256
8	8	8	256	256	256	256	256
8	8	256	256	256	256	256	256
8	8	256	256	256	256	256	256
8	8	256	256	256	256	256	256
8	8	256	256	256	256	256	256

ตารางที่ 5.5 แสดงค่าสัมประสิทธิ์การควอนไตซ์หรือค่าควอนตัม

+2853	-5	-2	-2	-1	0	0	-1
+20	0	-1	+1	0	+1	+1	+1
+5	-2	0	0	0	0	0	0
+5	-1	0	0	0	0	0	0
+1	+1	0	0	0	0	0	0
0	+1	0	0	0	0	0	0
0	+1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

ตารางที่ 5.6 แสดงค่าสัมประสิทธิ์ทางความถี่ที่ผ่านการควอนไตซ์แล้ว

95	96	101	100	100	103	102	100
88	89	95	95	96	99	99	98
88	89	92	91	90	91	90	88
86	88	89	89	87	88	87	87
84	87	87	88	87	88	88	88
80	85	84	86	85	86	87	88
79	84	82	85	83	85	86	88
78	83	80	83	81	82	83	85

ตารางที่ 5.7 ค่าของพิกเซลที่ได้จากการสร้างภาพกลับคืน

+2	+1	-4	+2	+2	-1	0	+2
0	-1	+2	+2	+1	-2	+3	-1
0	-1	+5	-3	-2	+6	-2	0
+2	0	-1	-1	+1	0	+1	+1
-4	+5	+1	0	+1	0	0	0
0	+3	-1	+2	-3	+2	+1	0
+1	-1	-2	+1	+5	+1	0	-2
+2	-3	+3	0	-1	-2	0	+3

ตารางที่ 5.8 แสดงค่าความผิดพลาดที่เกิดขึ้น

รูปที่ 5.12: แสดงตัวอย่างการเข้ารหัสข้อมูลด้วยวิธีเจเพก ซึ่งถ้าจะคำนวณฮัฟแมนได้ (ที่แสดงผลตามข้อ (e) นั้น) ต้องหา Probability ของ (g) ก่อน แล้วสร้างตารางด้วยวิธีของฮัฟแมน แล้วใส่รหัสของฮัฟแมนคืนให้กับข้อมูลที่ได้จาก (e)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(a) Original 8x8 block

140	144	147	140	144	155	179	175
144	152	140	147	140	148	167	179
152	165	138	167	143	162	152	172
168	145	165	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

(b) Shifted block

12	16	19	12	11	27	51	47
16	24	12	19	12	20	39	51
24	27	8	39	35	34	24	44
40	17	28	32	24	27	8	32
34	20	28	32	12	8	19	34
19	39	12	27	27	12	8	34
8	28	-5	39	34	16	12	19
20	27	8	27	24	19	19	8

(c) Block after DCT Eq. (5.1)

185	17	14	8	123	-9	-13	-18
20	-34	26	-9	-10	10	15	6
-10	-23	-1	6	-18	3	20	0
-8	-5	14	-14	-8	-2	-3	8
-3	9	7	1	-11	17	18	15
3	-2	-16	8	8	-3	0	-6
8	0	-2	3	-1	-7	-1	-1
0	-7	-2	1	1	4	-6	0

(d) Quantization Table (quality=2)

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

(e) Block after quantization Eq. (5.2)

61	-3	2	0	2	0	0	-1
4	4	2	0	0	0	0	0
-1	-2	0	0	-1	0	-1	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	-1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(f) Zig-zag sequence

61,-3,4,-1,-4,2,0,2,-2,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,-1,0,0,-1,0,0,
0,0,-1,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

(g) Intermediate symbol sequence

{5},{61},{0,2}{-3},{0,3}{4},{0,1}{-1},{0,3}{-4},{0,2}{2},{1,2}{2},{0,2}{-2},
{5,2}{2},{3,1}{1},{6,1}{-1},{2,1}{-1},{4,1}{-1},{7,1}{-1},{0,0}

(e) Encoded bit sequence (total 98 bits)

(1101)(111101)(01)(00)(100)(100)(00)(100)(001)(01)(10)
(11011)(10)(01)(01)(1111110111)(10)(111010)(1)
(111011)(0)(11100)(0)(111011)(0)(1111010)(0)(1010)

รูปที่ 5.12 แสดงตัวอย่างของข้อมูลกับกระบวนการเจเพก

5.5 การวัดผลการลดขนาดข้อมูล

1) อัตราการลดขนาดข้อมูล (C_R) = $\frac{\text{ขนาดข้อมูลต้นฉบับ}}{\text{ขนาดข้อมูลที่ถูกลดขนาดแล้ว}}$

$$\text{เปอร์เซ็นต์การลดขนาดข้อมูล (\%)} = 100 - \left[\frac{\text{ขนาดข้อมูลที่ลดขนาดแล้ว}}{\text{ขนาดข้อมูลต้นฉบับ}} \times 100 \right]$$

2) ค่าความผิดพลาดของภาพ (Root Mean Square Error (RMSE))

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - x'_i)^2}$$

x_i = ค่าฟังก์ชันต้นทุน

x'_i = ค่าพิทเชลที่ได้จากการสร้างข้อมูลคืน

n = จำนวนพิกเซลทั้งหมดในการแสดงผล

3) จำนวนบิตเฉลี่ยที่ใช้ในการเข้ารหัสข้อมูล (N_b) (บิตต่อพิกเซล)

$$N_b = \text{จำนวนบิตที่ใช้ในการเข้ารหัสข้อมูล} / \text{จำนวนพิกเซลของภาพ}$$

ตารางที่ 5.9 จะเห็นว่า ถ้าเราต้องการภาพที่มีคุณภาพดีมาก ๆ ต้องใช้จำนวนบิตในการเข้ารหัสมากด้วย อัตราการลดขนาดข้อมูลก็จะน้อยลง แต่ถ้าต้องการภาพที่ไม่ต้องถูกต้องมากนัก คือเพี้ยนได้บ้างในระดับที่ยอมรับได้ เราก็อาจจะได้อัตราการลดขนาดที่มากขึ้น

จำนวนบิตต่อพิกเซล (N_b)	คุณภาพของภาพที่ได้
0.25 - 0.5	คุณภาพดีพอใช้
0.5 - 0.75	คุณภาพดีมาก
0.75 - 1.0	คุณภาพดีเยี่ยม
1.5 - 2.0	ไม่สามารถแยกความแตกต่างด้วยสายตาวาภาพเกิดการผิดเพี้ยน

ตารางที่ 5.9 แสดงจำนวนบิตต่อพิกเซลที่มีผลต่อคุณภาพของภาพที่ได้

หมายเหตุ สมการคำนวณค่าความผิดพลาดของการบีบอัดและขยายข้อมูลภาพสี คือ

$$\text{ERROR} = \sqrt{\frac{1}{n \times 3} \sum_{i=1}^n [(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2]}$$

n : จำนวนพิกเซลของภาพ

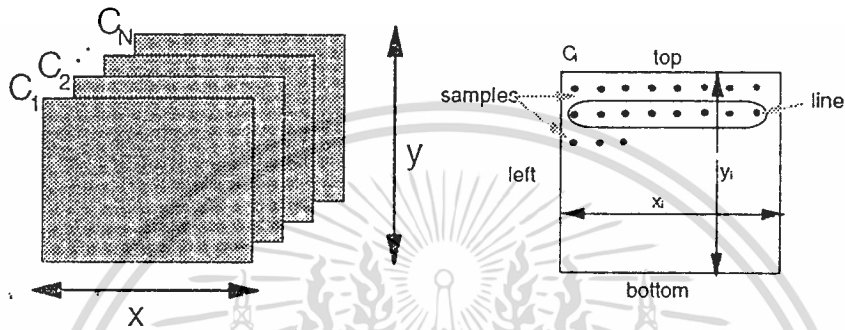
r : ค่าของคอมโพเนนต์สีแดง

g : ค่าของคอมโพเนนต์สีเขียว

b : ค่าของคอมโพเนนต์สีน้ำเงิน

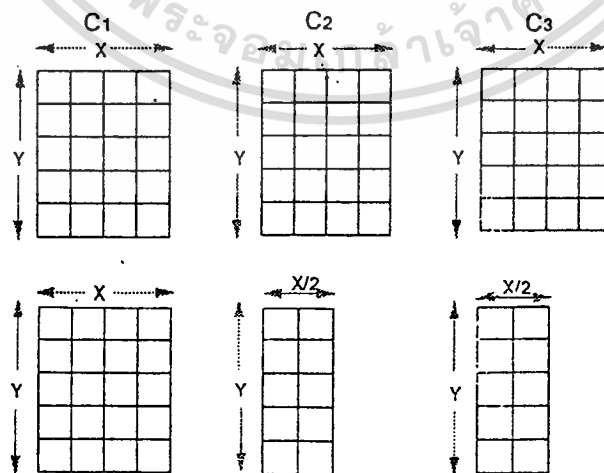
ภาคที่ 2 เจเพกสำหรับภาพสี

จากอัลกอริทึมของเจเพกสำหรับภาพเกรย์สเกล ถ้าเราจะนำมาบีบอัดข้อมูลภาพสี อาจกล่าวได้ว่า จะต้องเป็นการบีบอัดข้อมูลแบบหลายๆ คอมโพเนนต์ของภาพ จากโมเดล (Model) ของภาพต้นฉบับของเจเพกจะประกอบไปด้วย 1 ถึง 255 คอมโพเนนต์ของภาพที่เรียกกันว่า สีหรือแบนด์ ของสเปกตรัม (Spectral Band) ดังรูปที่ 5.13



รูปที่ 5.13 แสดงคอมโพเนนต์ของภาพสี

จากพื้นฐานของสีที่กล่าวว่า สีต่างๆ ประกอบขึ้นจากสีหลัก คือ สีแดง สีเขียว และสีน้ำเงิน ที่อัตราส่วนผสมต่างๆ กัน เราอาจพิจารณาได้ ตัวอย่างเช่น การแสดงผลแบบ RGB และ YCrCb ก็ประกอบไปด้วย 3 คอมโพเนนต์ ซึ่งแต่ละคอมโพเนนต์อาจจะมีจำนวนของพิกเซลในแนวนอง (X_i) และแนวนอน (Y_i) ที่ไม่เท่ากันก็เป็นได้ ดังในรูปที่ 5.14 แสดงให้เห็นถึง 2 กรณี สำหรับภาพสีที่มี 3 คอมโพเนนต์ กรณีแรกทุกๆ คอมโพเนนต์มีความละเอียดในการแสดงผลเท่ากัน แต่ในกรณีที่สอง แต่ละคอมโพเนนต์มีการแสดงผลที่ความละเอียดไม่เท่ากัน



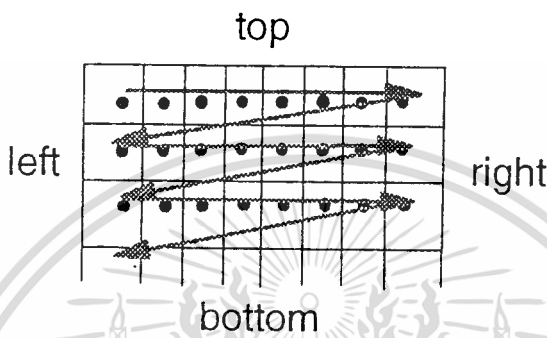
รูปที่ 5.14 แสดงความละเอียดของแต่ละคอมโพเนนต์ในการประมวลผลข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การประมวลผลคอมโพเนนต์ต่างๆ

คอมโพเนนต์ต่างๆ ของสีสามารถถูกประมวลผลได้ 2 ลักษณะ ด้วยกัน คือ

1. แบบลำดับการประมวลผลข้อมูลของแต่ละคอมโพเนนต์ตามลำดับ คือ จะทำการบีบอัดข้อมูลที่ละคอมโพเนนต์ จากซ้ายไปขวา และจากบนลงล่าง สมมติเป็นโหมด RGB เราก็จะทำการบีบอัด R (สีแดง) ก่อน G (สีเขียว) และ B (สีน้ำเงิน) ตามกันไปเป็นลำดับ แสดงตามรูปที่ 5.15 ซึ่งในการแสดงผลการขยายข้อมูลกลับคืนนั้น จะเห็นหน้าจอเป็น สีแดงจนเต็มจอก่อน แล้วค่อยแสดงสีเขียว ช้อนจนเต็มจอ และสีน้ำเงิน ช้อนทับเป็นลำดับสุดท้าย เป็นต้น



รูปที่ 5.15 แสดงลักษณะการประมวลผลข้อมูลของแต่ละคอมโพเนนต์แบบไม่ซ้อนทับกัน

2. แบบประมวลผลแต่ละคอมโพเนนต์ไปพร้อมๆ กัน โดยที่คอมโพเนนต์ต่างๆ เหล่านั้นจะถูกนำมารวมกัน เรียกว่า หน่วยที่ถูกเข้ารหัสแบบน้อยที่สุด (Minimum Coded Units หรือ MCUs) วิธีนี้เหมาะสำหรับงานที่จำเป็นต้องแสดงผลหรือพิมพ์ พร้อมๆ ไปกับกระบวนการขยายข้อมูลคืนในทันที เนื่องจากคอมโพเนนต์ต่างๆ จะต้องซ้อนทับเข้าด้วยกัน ดังนั้นจึงจำเป็นที่จะต้องทำการแบ่งแต่ละคอมโพเนนต์แบบสี่เหลี่ยม ตาม H_i และ V_i ตัวอย่างแสดง ตามรูปที่ 5.16 ซึ่งภาพนี้ประกอบด้วย 4 คอมโพเนนต์ คือ C_1, C_2, C_3, C_4 ซึ่งแต่ละคอมโพเนนต์ก็มีความละเอียด H_i, V_i ที่ไม่เท่ากัน

A: $H=2, V=2$

B: $H=2, V=1$

C: $H=1, V=2$

D: $H=1, V=1$

Data units	A	B	C	D
MCU1 = {	A00,A01,A10,A11,	B00,B01	C00,C10,	D00}
MCU2 = {	A02,A03,A12,A13,	B02,B03,	C01,C11,	D01}
MCU3 = {	A04,A05,A14,A15,	B04,B05	C02,C12,	D02}
MCU4 = {	A20,A21,A30,A31,	B10,B11	C20,C30,	D10}

รูปที่ 5.16 แสดงทิศทางการประมวลผลข้อมูลแต่ละคอมโพเนนต์ แบบที่ความละเอียดของแต่ละคอมโพเนนต์ไม่เท่ากัน

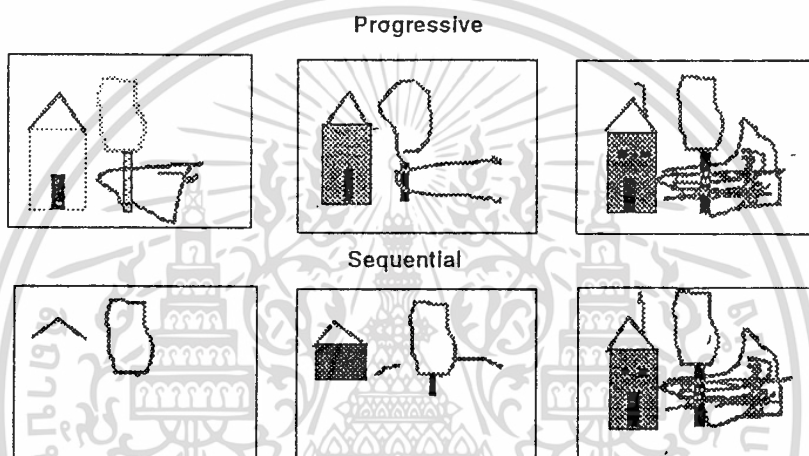
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแสดงผลข้อมูล

การแสดงผลข้อมูลผ่านการบีบอัดและขยายข้อมูลคืนแล้ว

1. แบบภาพหายไปแล้วเริ่มละเอียดขึ้น (Progressive) จะใช้การสแกนภาพครั้งแรกๆ แล้วทำการส่งข้อมูลภาพอย่างหายาบบๆ ออกมาก่อน เมื่อทำการสแกนครั้งต่อๆ มา จะเริ่มแสดงรายละเอียดของภาพมากขึ้น

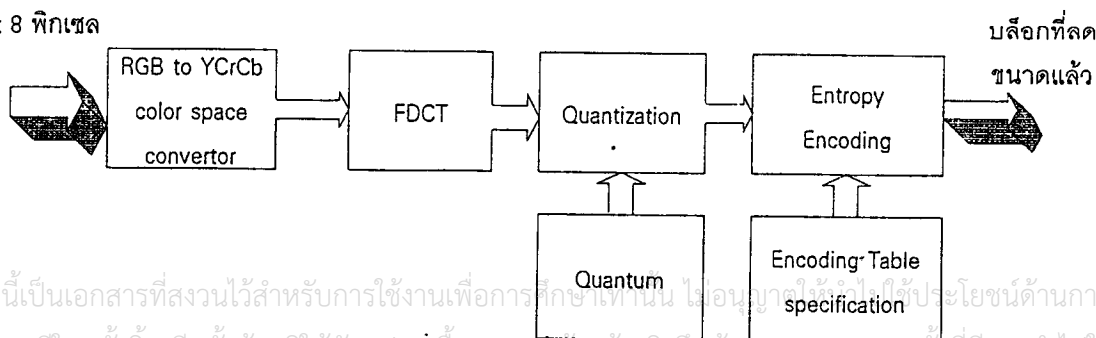
2. แบบซีควนเชียล (Sequential) เป็นการแสดงผลภาพอย่างชัดเจน แบบเรียงทีละพิกเซลไปเรื่อยๆ จนหมดภาพ ดังนั้นอาจจะทำให้เรารู้สึกว่าชักช้าหรือหงุดหงิด เพราะกว่าจะแสดงเต็มจอได้จะต้องใช้เวลา



รูปที่ 5.17 แสดงลักษณะการแสดงผลแบบหายไปแล้วจึงละเอียดขึ้นกับแบบซีควนเชียล

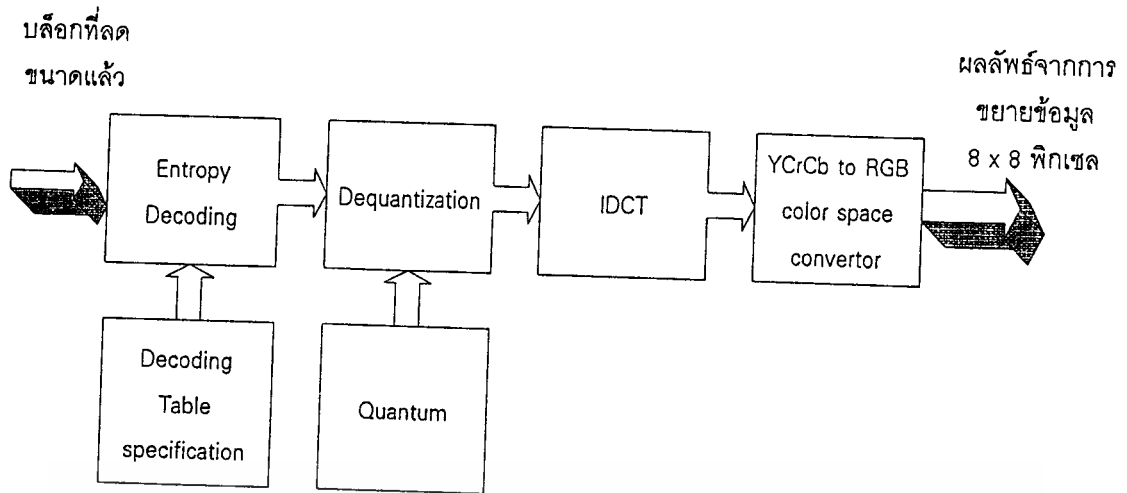
บล็อกไดอะแกรม (Block Diagram) ของตัวเข้ารหัสและถอดรหัสของ JPEG จะเหมือนกับภาพเกรย์สเกล แต่จะมีส่วนที่เพิ่มเติมเข้ามาคือ จะต้องเพิ่มบล็อกของการแปลงสี ตัวอย่างเช่นการแปลงจาก RGB เป็น YCrCb ก่อนเข้าทำการแปลงโคไซน์แบบไม่ต่อเนื่อง และในการถอดรหัสก็ต้องมีการแปลงกลับจาก YCrCb ไปเป็น RGB อีกครั้งหนึ่งหลังจากทำการแปลงกลับโคไซน์แบบไม่ต่อเนื่องเสร็จเรียบร้อยแล้วด้วย ดังนี้

ภาพต้นฉบับ
8 x 8 พิกเซล



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 5.18 แสดงการเข้ารหัสด้วยวิธี JPEG สำหรับภาพสี



รูปที่ 5.19 แสดงการถอดรหัสด้วยวิธี JPEG สำหรับภาพสี

สำหรับสมการที่ใช้ในการแปลงสี จาก RGB ไปเป็น YCrCb นั้น

$$Y = \{ (0.299 * R) + (0.587 * G) + (0.114 * B) \}$$

$$Cr = R - Y$$

$$Cb = B - Y$$

และในการแปลงกลับจาก YCrCb เป็น RGB ทำได้โดย

$$R = Cr + Y$$

$$B = Cb + Y$$

$$G = \{ (1.7 * Y) - (0.509 * R) - (0.194 * B) \}$$

หมายเหตุ

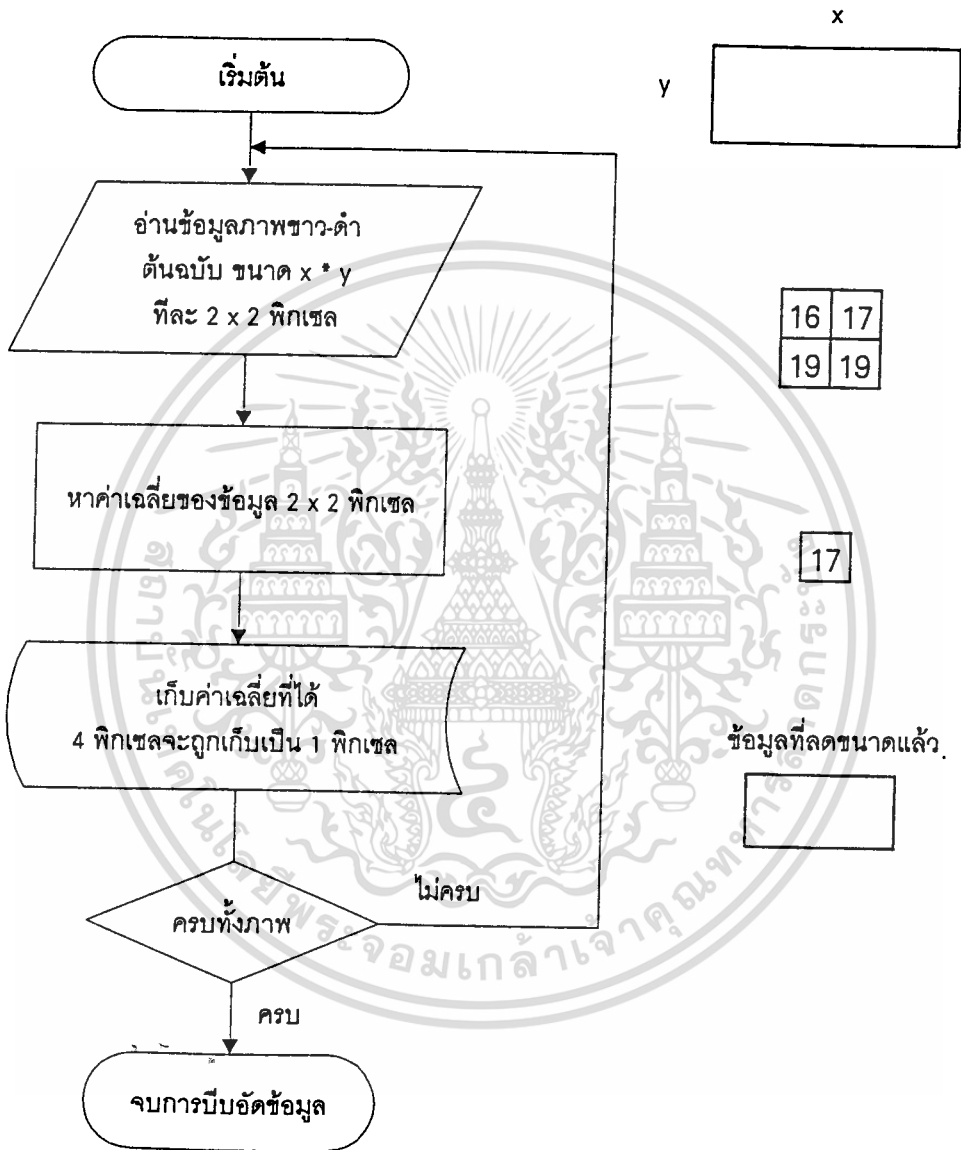
สาเหตุที่เราทำการเลือกที่จะแปลงจาก RGB ไปเป็น YCrCb เพราะว่า จากผลการวิจัยของผู้เชี่ยวชาญได้สรุปเอาไว้ว่า ตาของมนุษย์ไม่สามารถมองเห็นความแตกต่างของสีได้ดีเท่าความแตกต่างของความสว่างหรือความมืดของภาพ ดังนั้นถ้าเราทำการแปลงคอมโพเนนต์ของภาพ โดยอาศัยสมการที่เป็นฟังก์ชันของสีกับความสว่างของภาพ จะให้ภาพผลลัพธ์ที่ไม่ผิดเพี้ยนไปจากภาพต้นฉบับมากนัก ซึ่งต่างกับการบีบอัดและทำการขยายคืนด้วยคอมโพเนนต์ของ RGB โดยตรง

ในการทดสอบนี้ได้เลือกประมาณผลกับ YCrCb ที่มีความละเอียดเท่ากับภาพต้นฉบับ กล่าวคือ ถ้าภาพเป็นภาพขนาด 320 x 200 พิกเซล ความละเอียดของ Y, Cr, Cb จะเป็น 320 x 200 พิกเซลเช่นกัน และในการแสดงผลใช้การแสดงผลแบบเป็นลำดับหรือซีควเอนเชียล

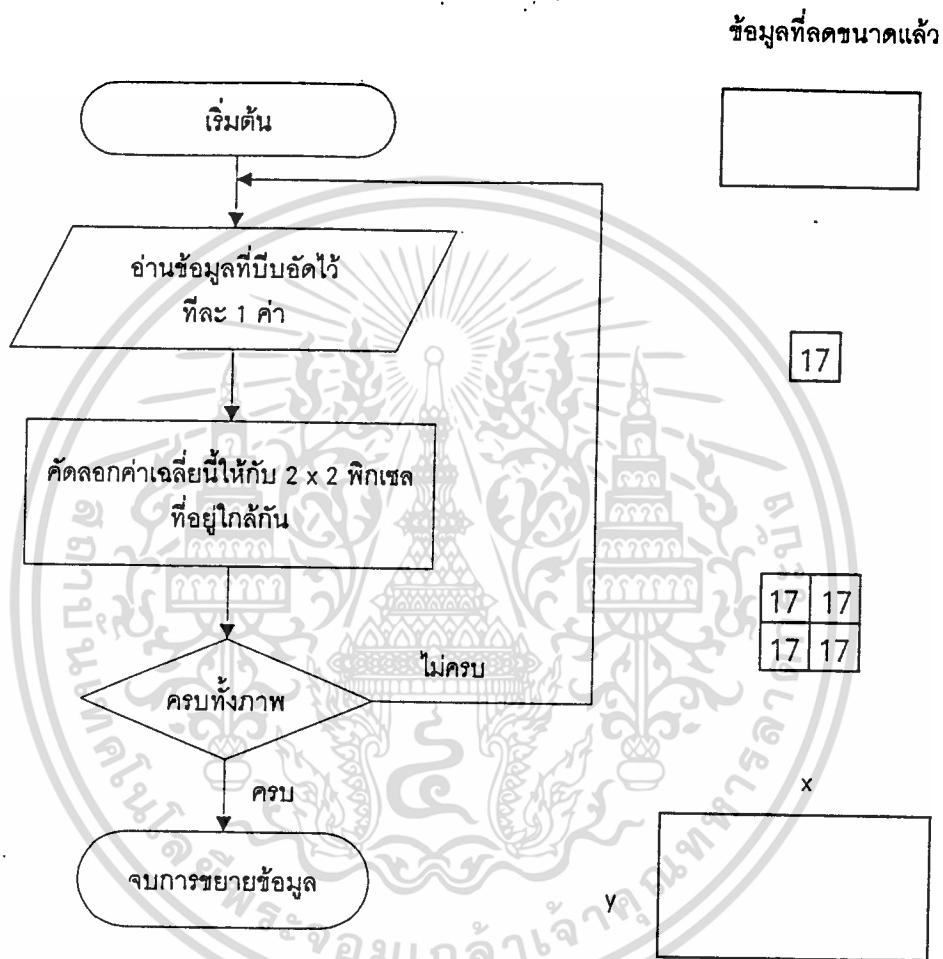


บทที่ 6
ผลงาน

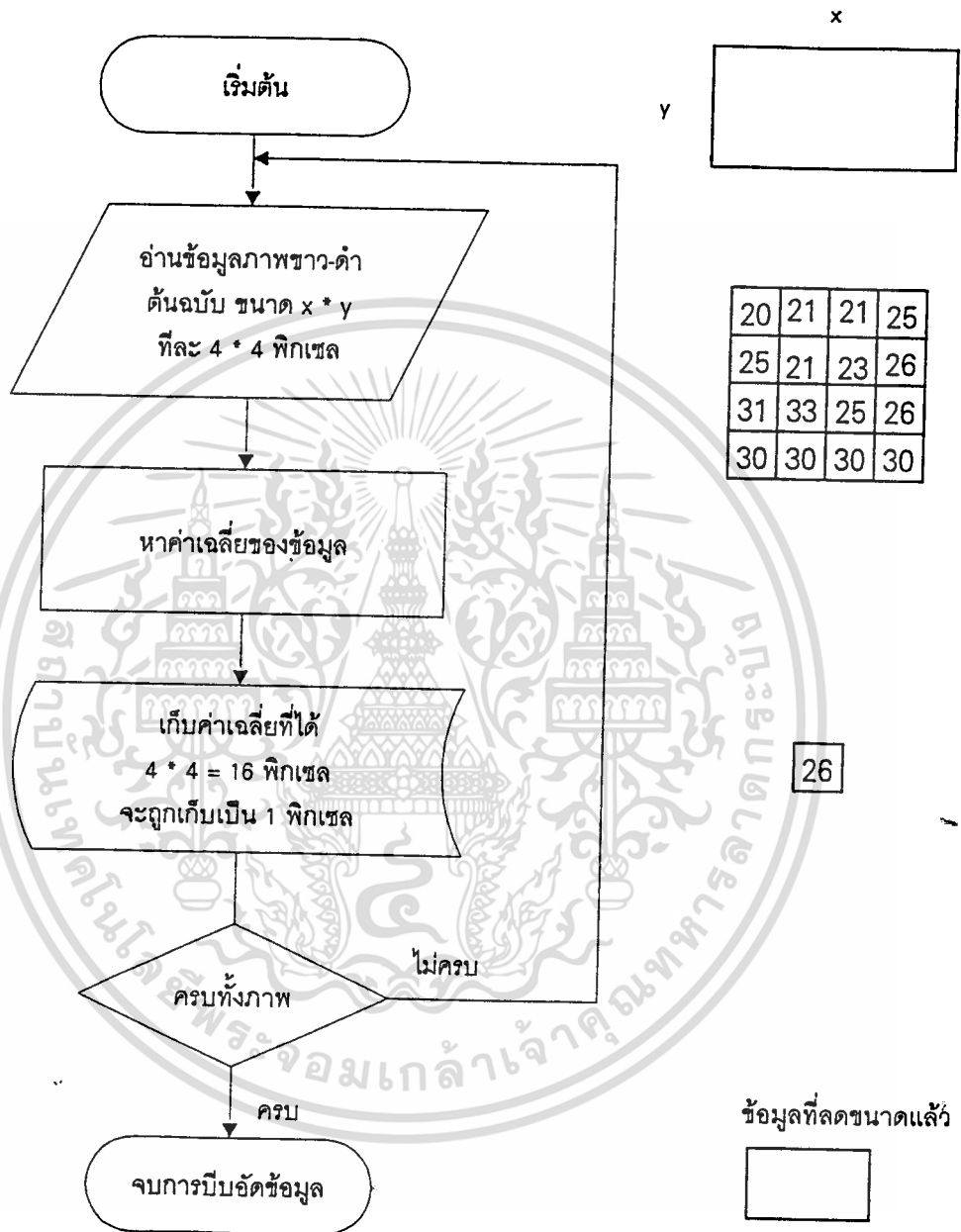
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



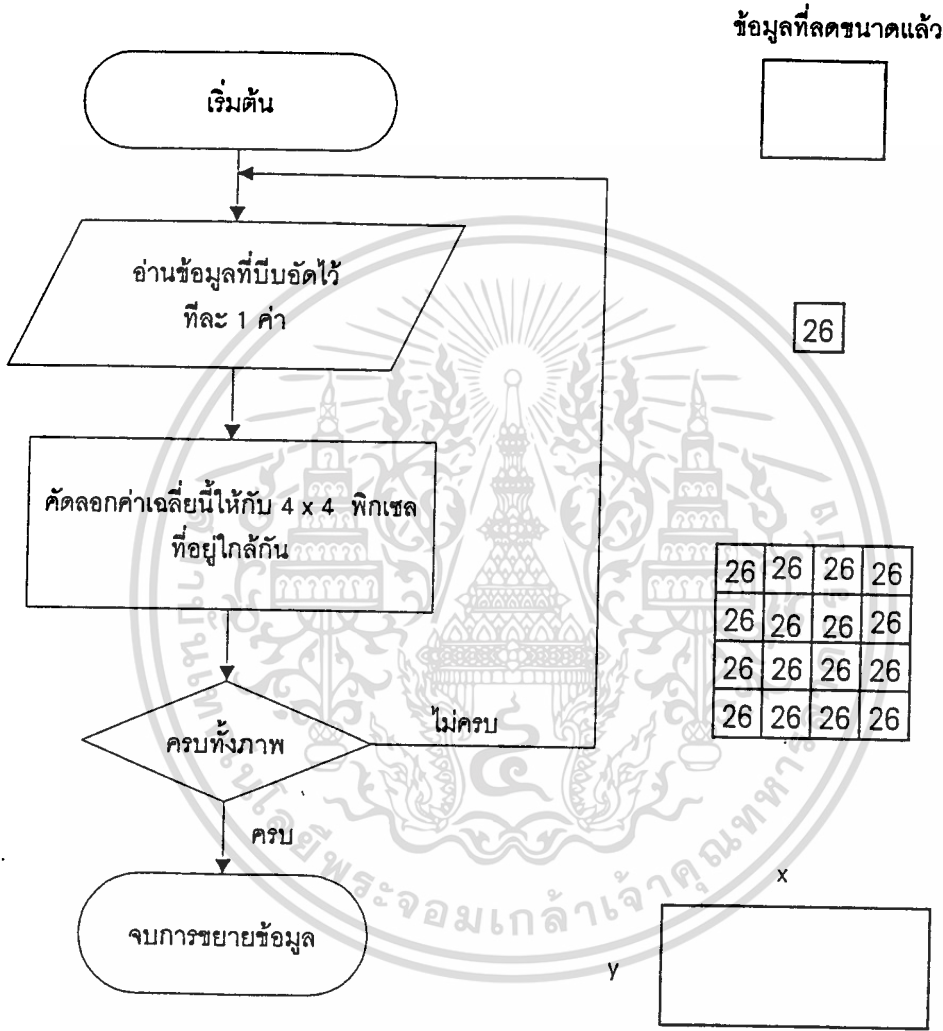
ผังงานที่ 6.2 การบีบข้อมูลแบบ LUMA 2:1



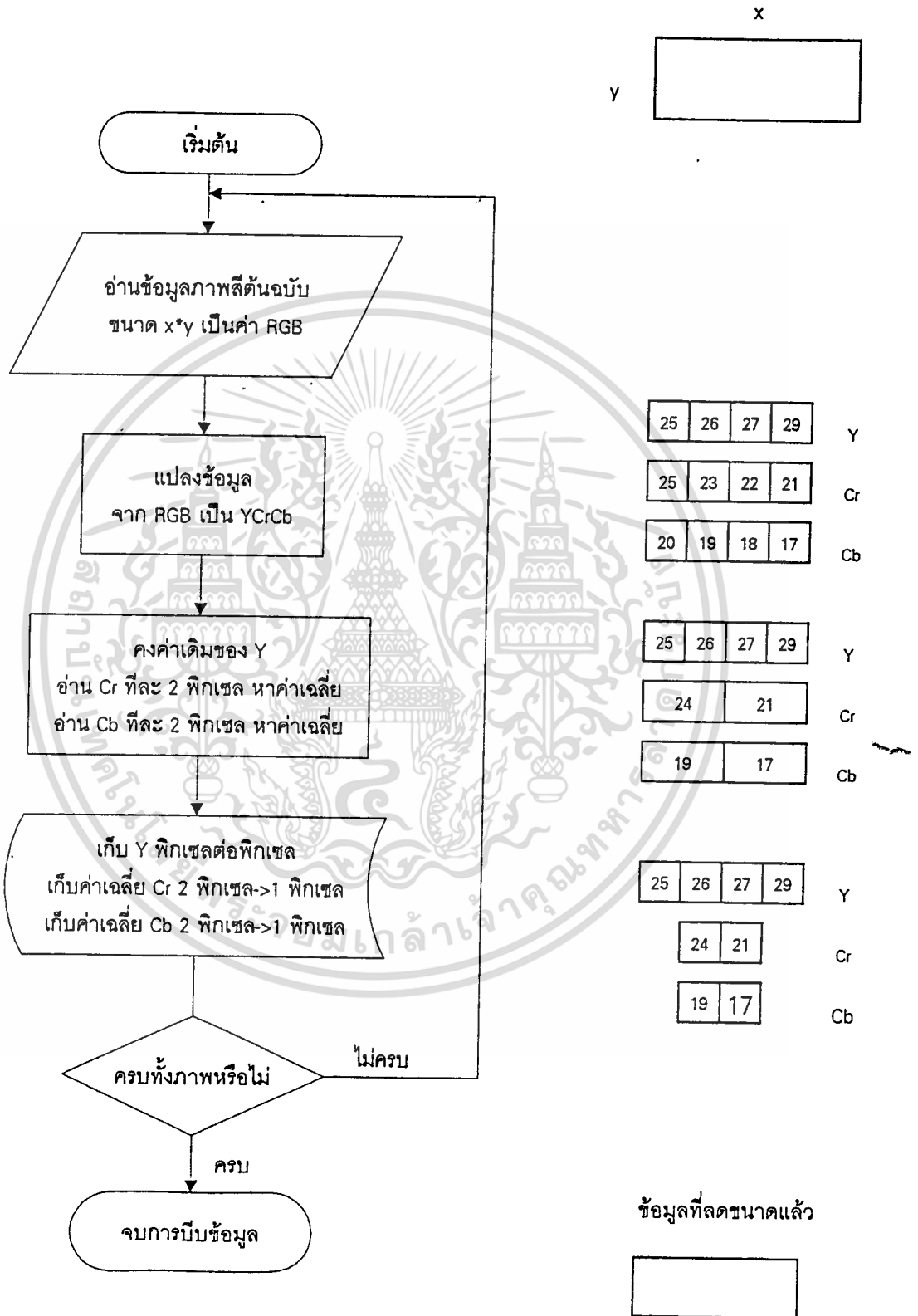
ผังงานที่ 6.3 การขยายข้อมูลแบบ LUMA 2 : 1



ผังงานที่ 6.4 การบีบข้อมูลแบบ LUMA 4 : 1



ผังงานที่ 6.5 การขยายข้อมูลแบบ LUMA 4 : 1

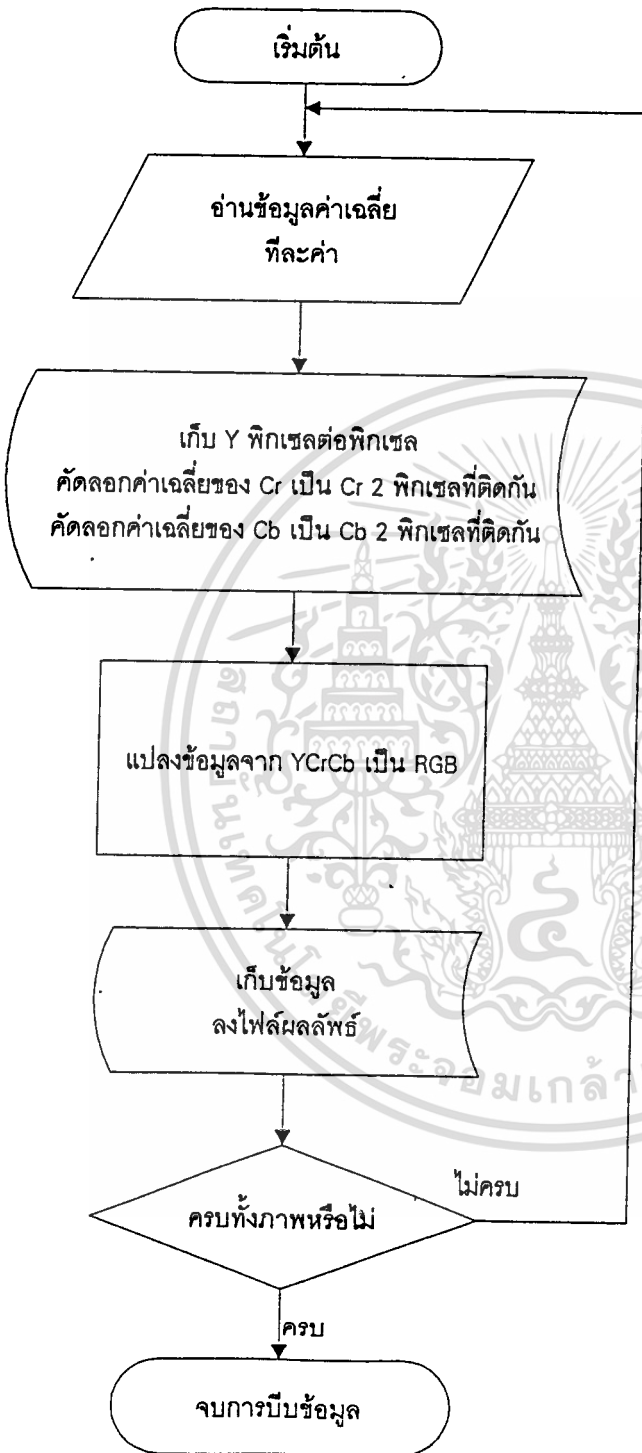


ข้อมูลทีลดขนาดแล้ว

--

25	26	27	29	Y
24	21			Cr
19	17			Cb

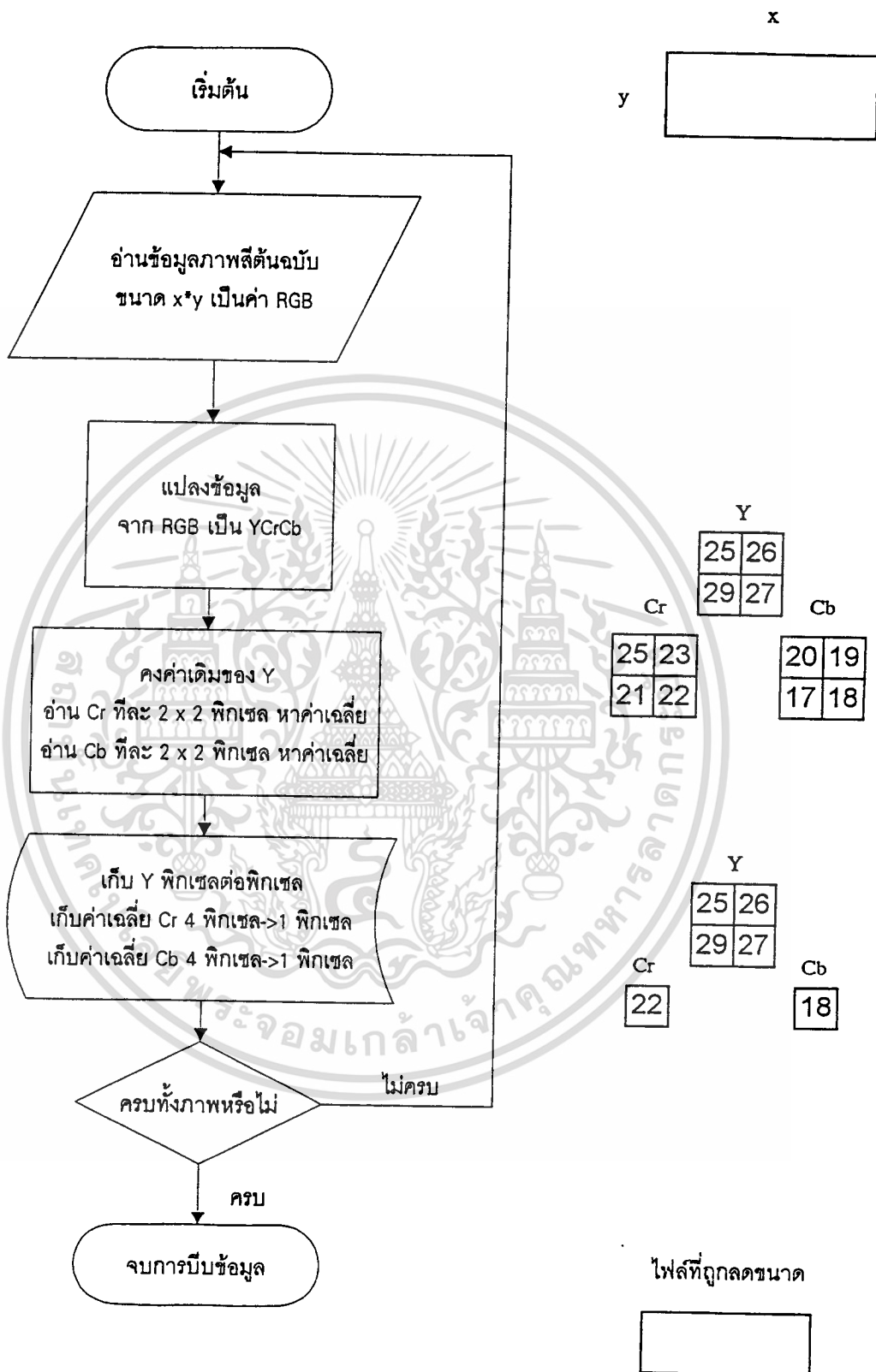
25	26	27	29	Y
24	24	21	21	Cr
19	19	17	17	Cb



	x
y	

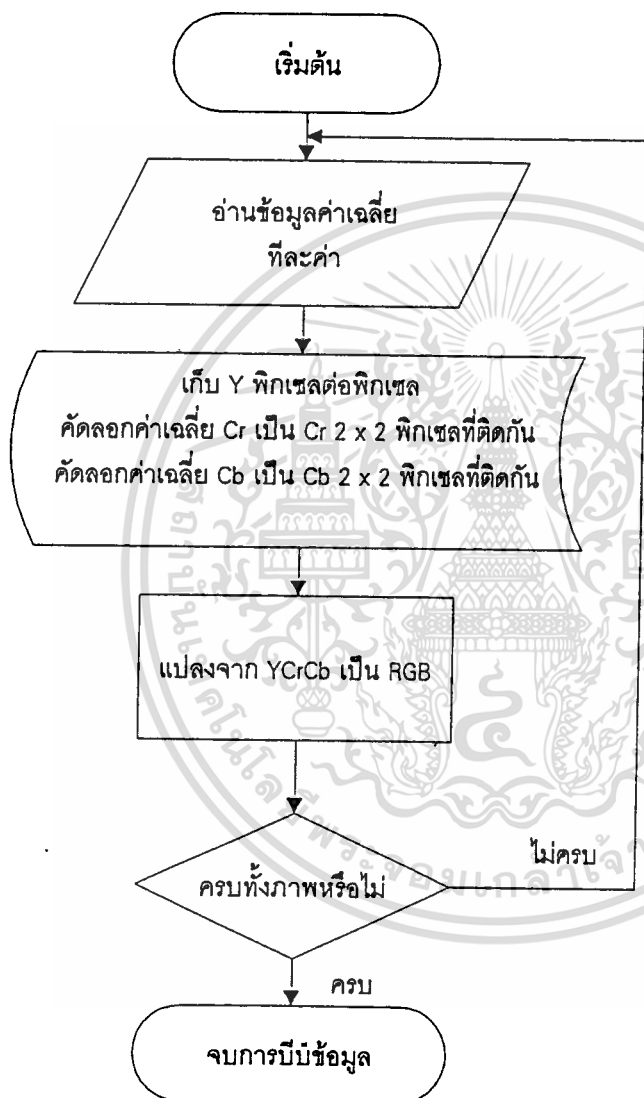
ผังงานที่ 6.7 การขยายข้อมูลแบบ Chroma 4 : 2 : 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ผังงานที่ 6.8 การบีบข้อมูลแบบ Chroma 2 ทิศทาง 4 : 1 : 1

ไฟล์ที่ลดขนาดแล้ว



25	26	Y
29	27	

22	Cr
----	----

18	Cb
----	----

25	26	Y
29	27	

22	22	Cr
22	22	

18	18	Cb
18	18	

x

y

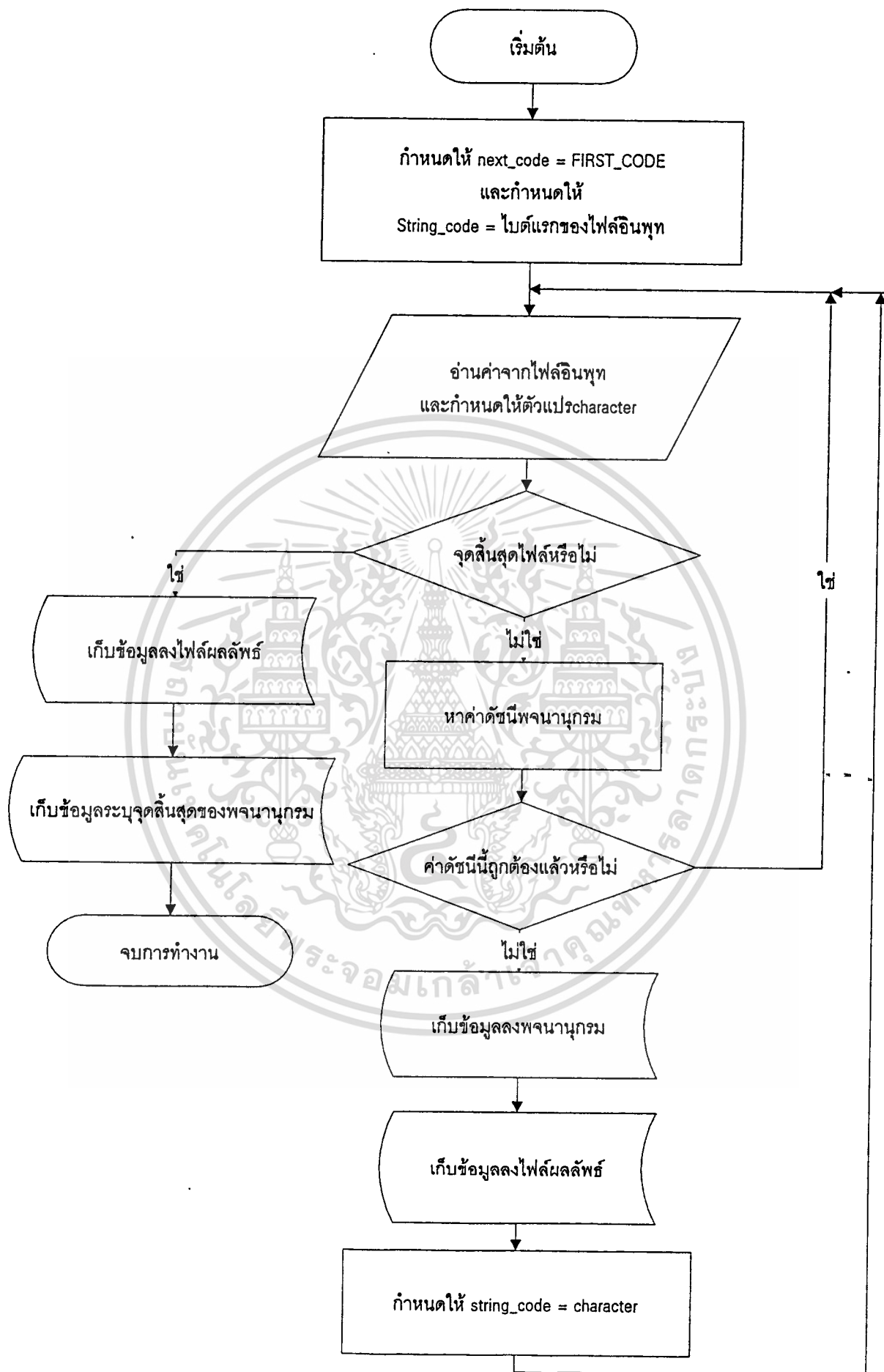


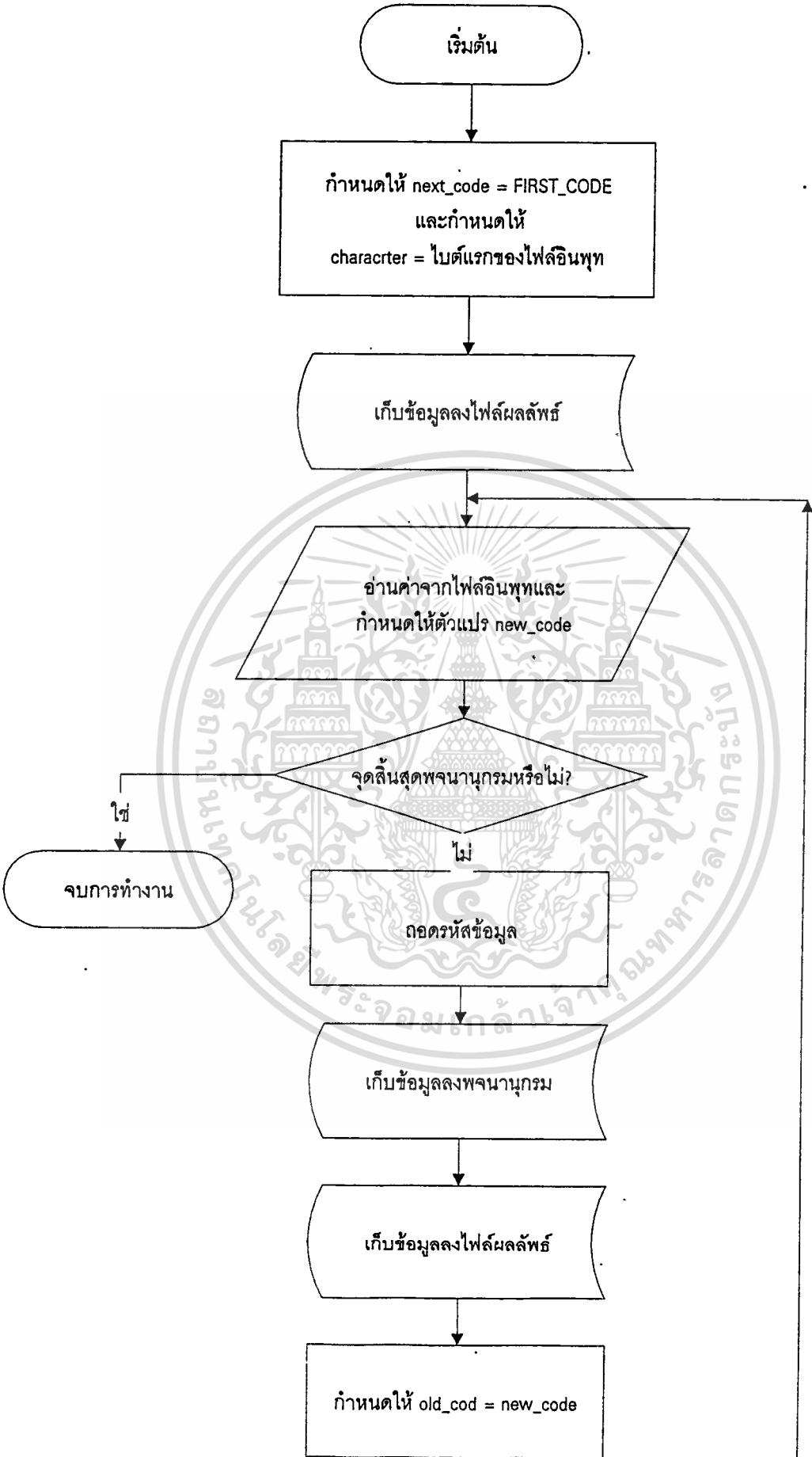
ผังงานที่ 6.9

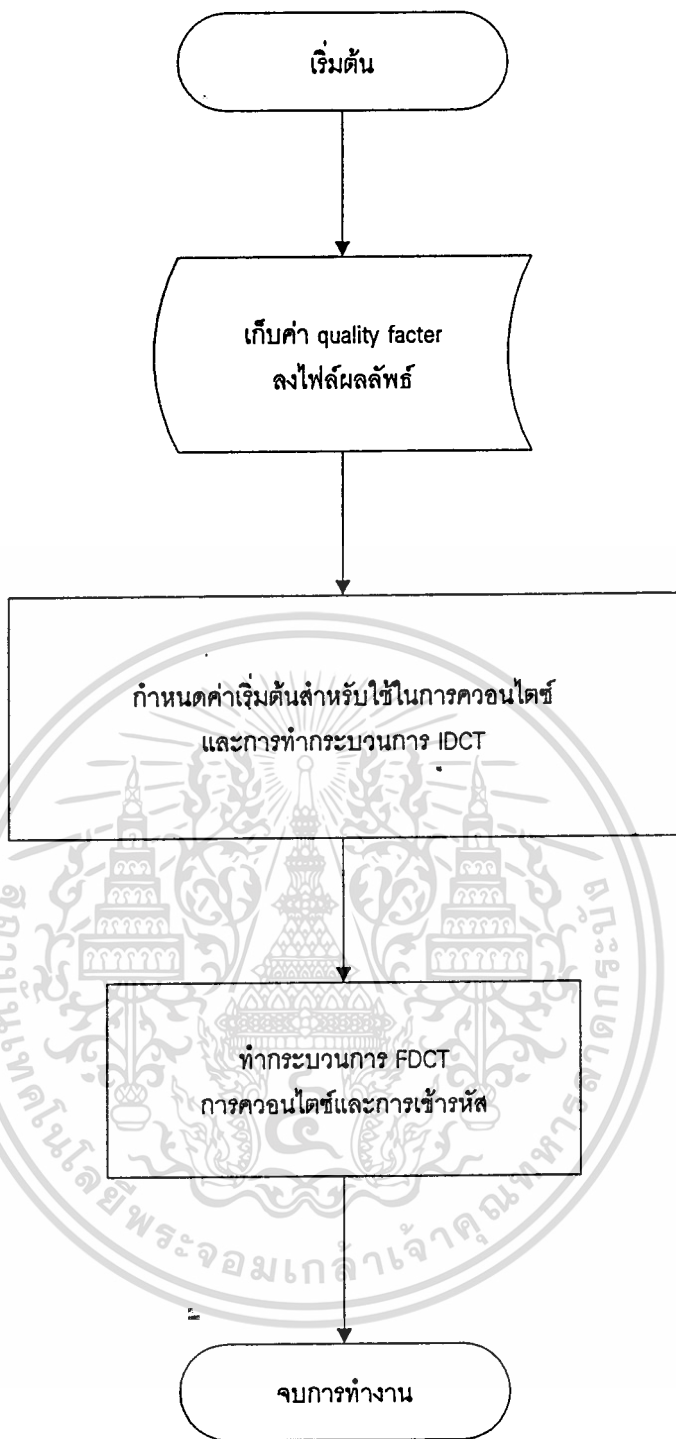
การขยายข้อมูลแบบ Chroma 2 ทิศทาง 4 : 1 : 1

เอกสารนี้เป็นเอกสารสงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

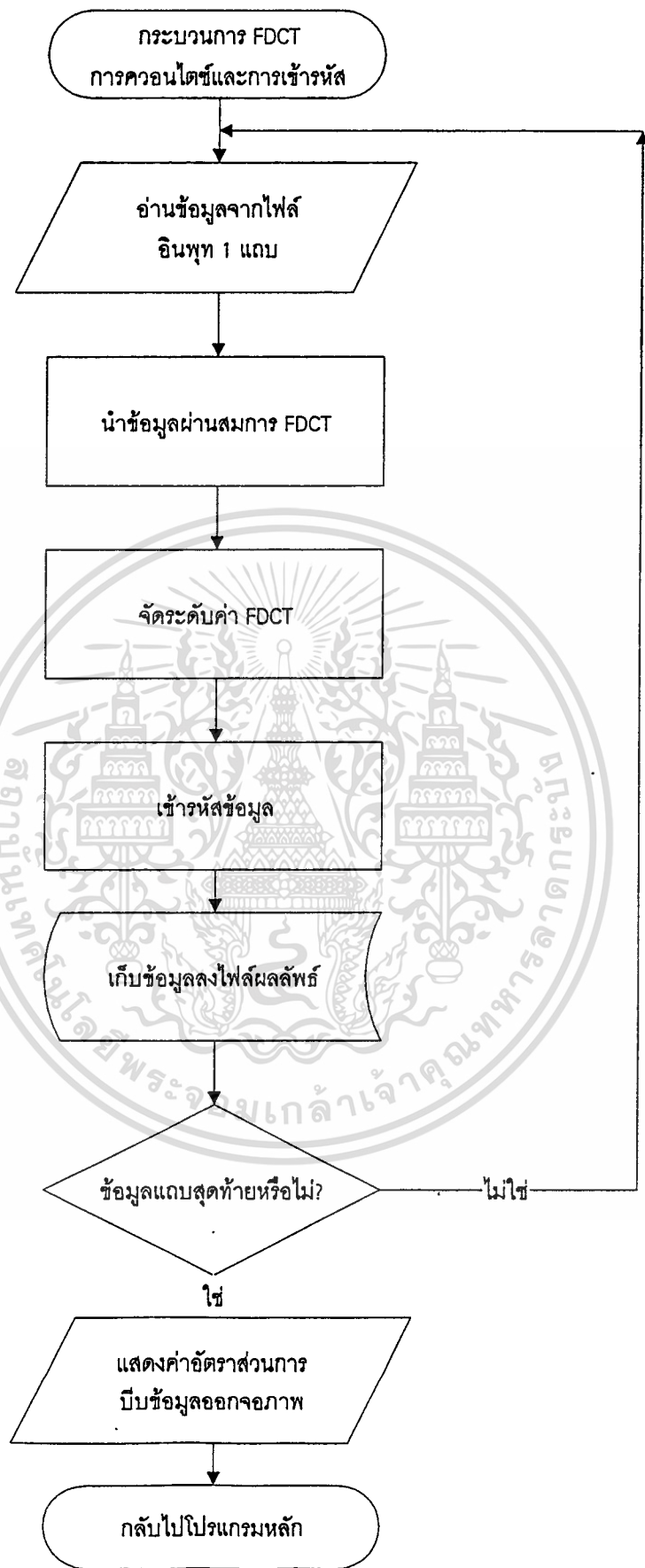
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้





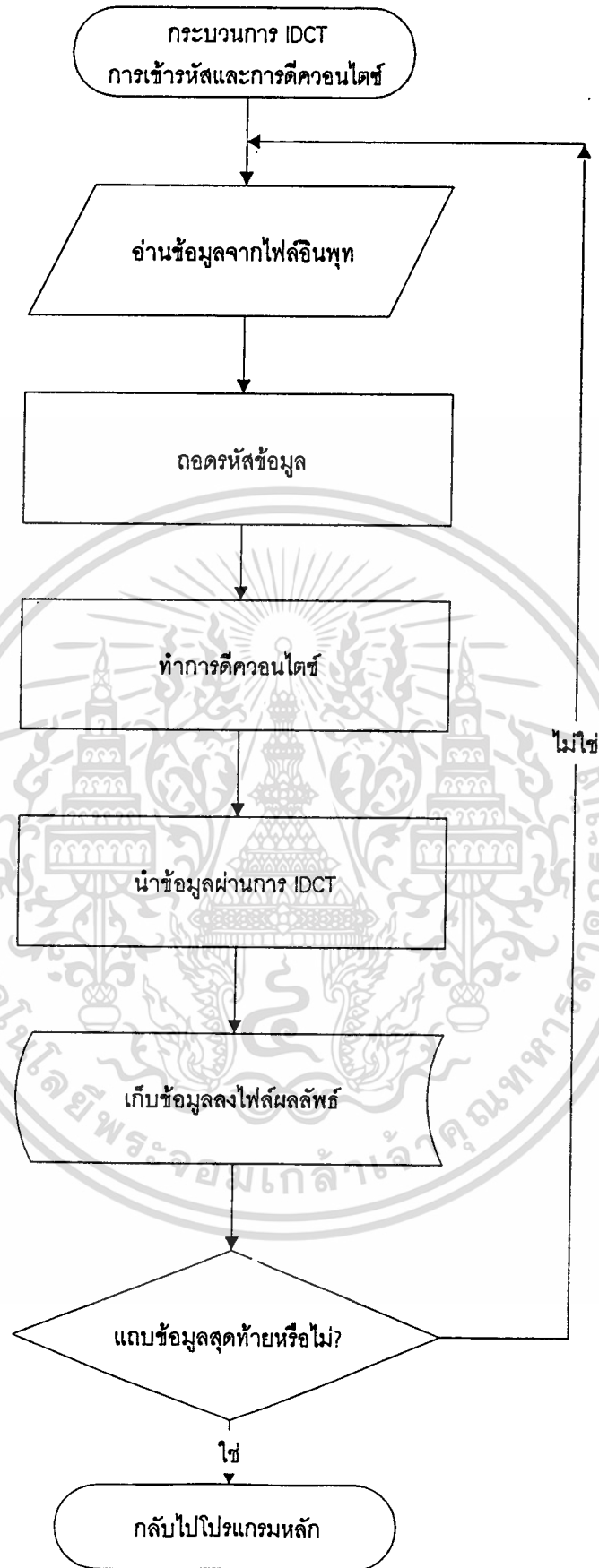


ผังงานที่ 6.12 การบีบข้อมูลแบบ JPEG ภาพขาว-ดำ





ผังงานที่ 6.14 การขยายข้อมูลแบบ JPEG ภาพขาว-ดำ

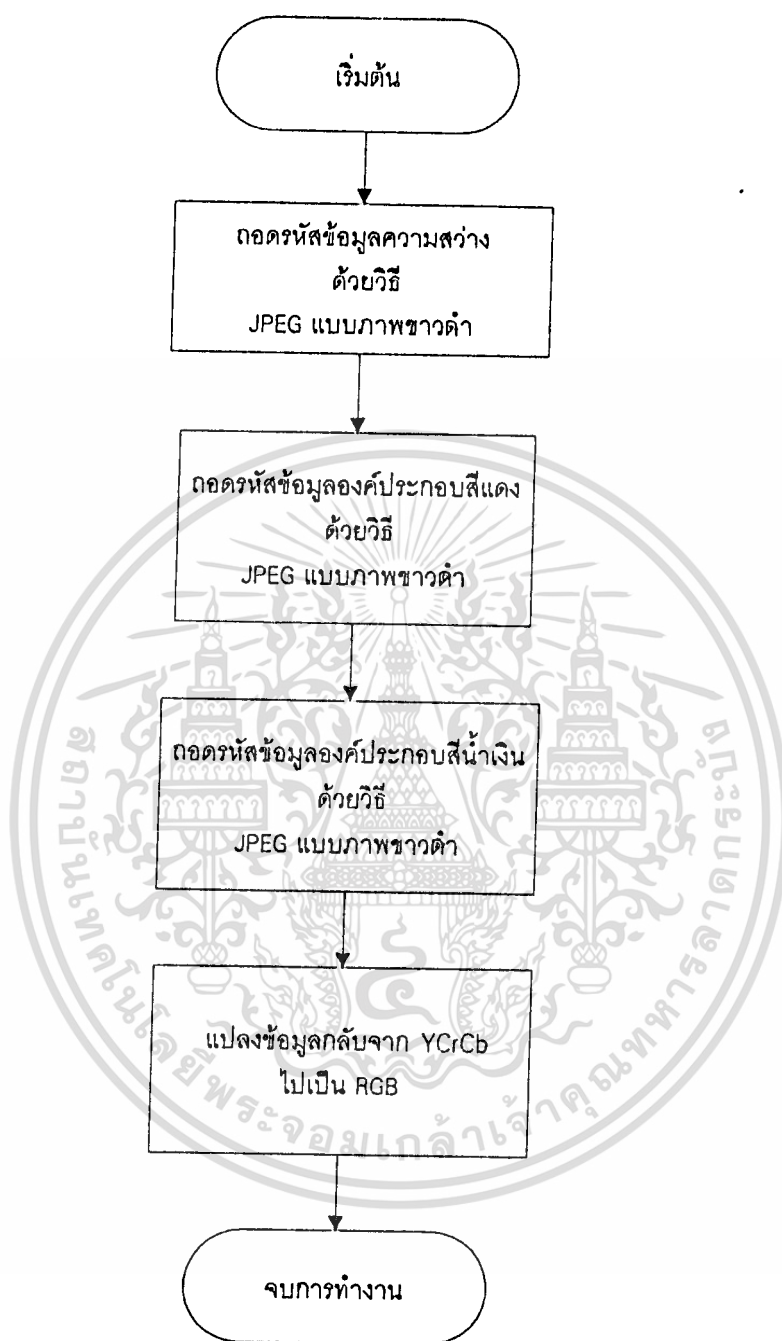


ผังงานที่ 6.15 กระบวนการ IDCT , ดีควอนไดซ์และการถอดรหัส

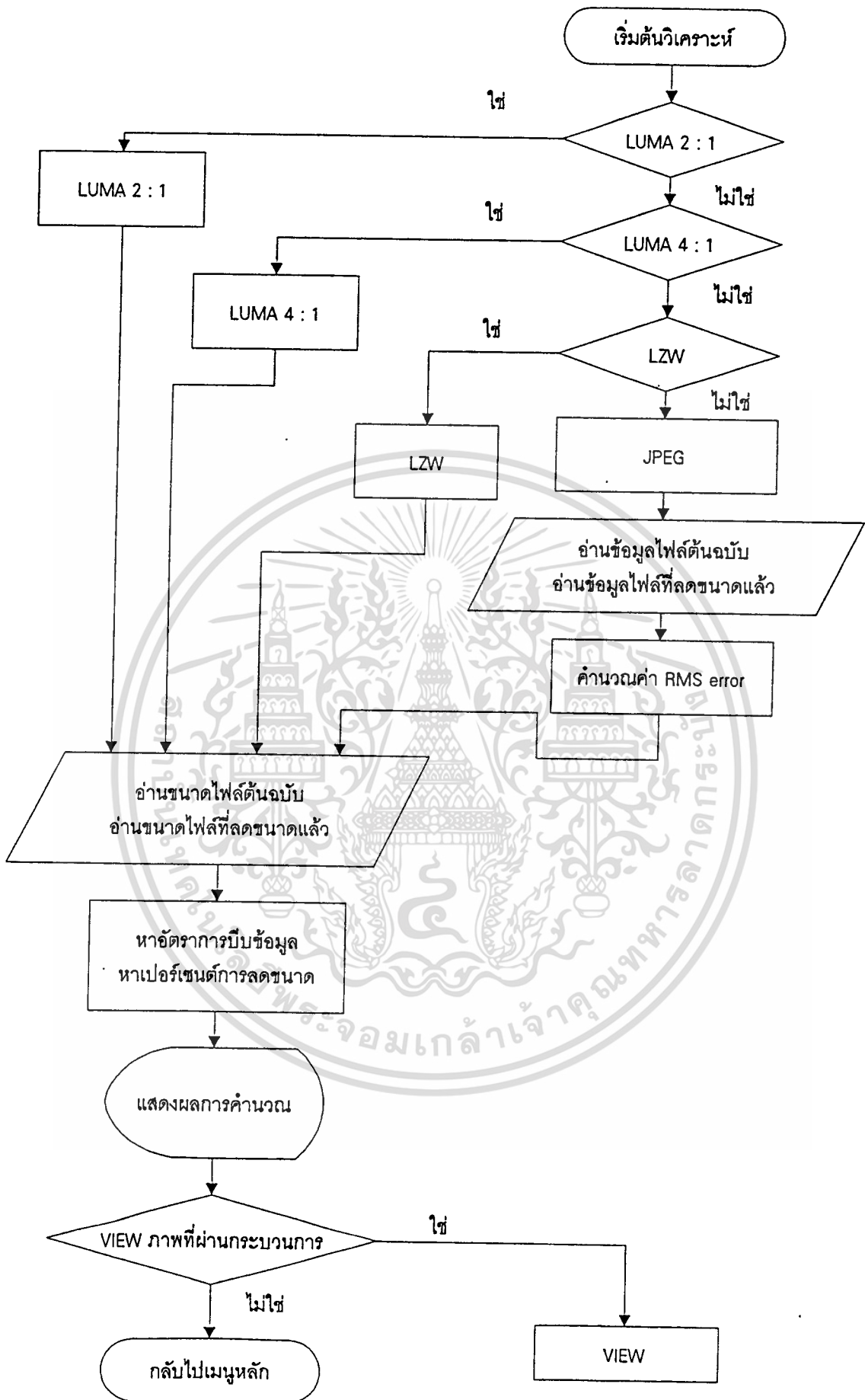
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้เผยแพร่หรือใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ผังงานที่ 6.16 การบีบข้อมูลแบบ JPEG ภาพสี



ผังงานที่ 6.17 การขยายข้อมูลแบบ JPEG ภาพสี



บทที่ 7

การทดลอง

ภาคที่ 1 การประมวลผลภาพขาว - ดำ

1. นำข้อมูลภาพจากไฟล์ข้อมูลภาพต้นฉบับ (ไฟล์ .GS) มาทำการบีบอัดและขยายไฟล์ข้อมูล ด้วยวิธี
 - 1.1 วิธีสุ่มค่าสัญญาณความสว่าง (Luma-Subsampling)
 - 1.1.1 LUMA 2 : 1
 - 1.1.2 LUMA 4 : 1
 - 1.2 วิธีตามความต่อเนื่องของข้อมูล (LZW)
 - 1.3 วิธีแจก ที่ค่าแฟกเตอร์คุณภาพต่าง ๆ
 - 1.3.1 JPEG 3 : ค่าแฟกเตอร์คุณภาพ เท่ากับ 3
 - 1.3.2 JPEG 10 : ค่าแฟกเตอร์คุณภาพ เท่ากับ 10
 - 1.3.3 JPEG 25 : ค่าแฟกเตอร์คุณภาพ เท่ากับ 25

2. เขียนโปรแกรมเมนูเพื่อช่วยในการแสดงผลข้อมูลภาพ

หมายเหตุ

การทดลองนี้ทำด้วยเครื่องคอมพิวเตอร์ รุ่น 486

ภาคที่ 2 การประมวลผลภาพสี

1. ถอดข้อมูลที่เป็นข้อมูลของภาพสีล้วนๆ ออกจากไฟล์ .PCX (มีการตัดเฮดเดอร์และถอดรหัสข้อมูล)

2. ทำการบีบอัดและขยายข้อมูลรูปภาพสี ด้วยวิธีการต่างๆ ดังต่อไปนี้

2.1 วิธีสุ่มค่าสัญญาณสี (Chroma subsampling) แบบที่ไม่มีการแปลงจาก RGB เป็น YCrCb

2.1.1 CNO411 : สุ่มตามแนวตั้ง แบบ 4 : 1 : 1

2.2 วิธีสุ่มค่าสัญญาณสี (Chroma subsampling) มีการแปลงจาก RGB เป็น YCrCb

2.2.1 C411 : สุ่มตามแนวตั้ง แบบ 4 : 1 : 1

2.2.2 C422 : สุ่มตามแนวตั้ง แบบ 4 : 2 : 2

2.2.3 CR4 : สุ่มแบบ 2 ทิศทาง แบบ 4 : 1 : 1

2.2.4 CR5 : สุ่มแบบ 2 ทิศทาง แบบ 16 : 1 : 1

2.3 วิธีตามความต่อเนื่องของข้อมูล (LZW)

2.4 วิธีเจเพกที่ค่าแฟกเตอร์คุณภาพต่างๆ

2.4.1 JPEG 1 : ค่าแฟกเตอร์คุณภาพ เท่ากับ 1

2.4.2 JPEG 10 : ค่าแฟกเตอร์คุณภาพ เท่ากับ 10

2.4.3 JPEG 25 : ค่าแฟกเตอร์คุณภาพ เท่ากับ 25

3. เขียนโปรแกรมเมนูเพื่อการใช้งานในการบีบอัดข้อมูลภาพขาว - ดำ

หมายเหตุ

การทดลองนี้ ได้ทำการประมวลผลด้วยเครื่องคอมพิวเตอร์ รุ่น 486

บทที่ 8

ผลการทดลอง

ภาคที่ 1 ผลการประมวลผลภาพขาว - ดำ

ตารางที่ 8.1 แสดงการเปรียบเทียบขนาดของภาพต้นฉบับกับภาพที่ทำการลดขนาด และเปอร์เซ็นต์การลดขนาดข้อมูล ภาพต่างๆ

วิธี / ชื่อภาพ	ROSE (64000 ไบต์)	CLOWN (64000 ไบต์)	LISAW (64000 ไบต์)	MOUSE (64000 ไบต์)
LUMA 2:1	16000	16000	16000	16000
	75%	75%	75%	75%
LUMA 4:1	4000	4000	4000	4000
	93.75%	93.75%	93.75%	93.75%
LZW	30905	73361	52554	56274
	52%	-14%	18%	13%
JPEG 3	10452	11323	6971	6403
	84%	83%	90%	90%
JPEG 10	6277	6355	4968	4917
	91%	91%	93%	93%
JPEG 25	4807	4811	4170	4337
	93%	93%	94%	94%

ตารางที่ 8.2 แสดงผลการจับเวลาที่ใช้ในการคำนวณ โดยเริ่มนับเมื่อสั่งรันโปรแกรมและสิ้นสุดเมื่อรันโปรแกรมเสร็จ โดยเลือกภาพ ROSE และ CLOWN เมื่อรันบนเครื่อง 486 หน่วยเป็นวินาทีโดยประมาณ

ชื่อภาพ/วิธี	LUMA 2:1	LUMA 4:1	LZW	JPEG 3	JPEG 10	JPEG 25
ROSE	4	3	6	13	12	12
CLOWN	4	3	6	13	15	14

ตารางที่ 8.3 แสดงค่าผิดพลาด เป็นค่ากำลังสองของค่าเฉลี่ยรากที่สอง (RMS error)

วิธี/ ชื่อภาพ	ROSE	CLOWN	LISAW	MOUSE
JPEG 3	6.7	6.9	3.2	3.0
JPEG 10	10.7	10.9	4.7	4.5
JPEG 25	14.3	14.1	6.6	6.3

ตารางที่ 8.4 แสดงจำนวนบิต (เฉลี่ย) ที่ใช้ในการเข้ารหัสข้อมูล (บิตต่อพิกเซล)

วิธี / ชื่อภาพ	ROSE	CLOWN	LISAW	MOUSE
LUMA 2:1	2	2	2	2
LUMA 4:1	0.5	0.5	0.5	0.5
LZW	3.8	9.2	6.6	7
JPEG 3	1.3	1.4	0.9	0.8
JPEG 10	0.7	0.8	0.6	0.6
JPEG 25	0.6	0.6	0.6	0.6

ภาคที่ 1 การประมวลผลภาพขาว - ดำ



รูปที่ 1.1 แสดงภาพ ROSE ต้นฉบับ



รูปที่ 1.2 แสดงภาพ CLOWN ต้นฉบับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

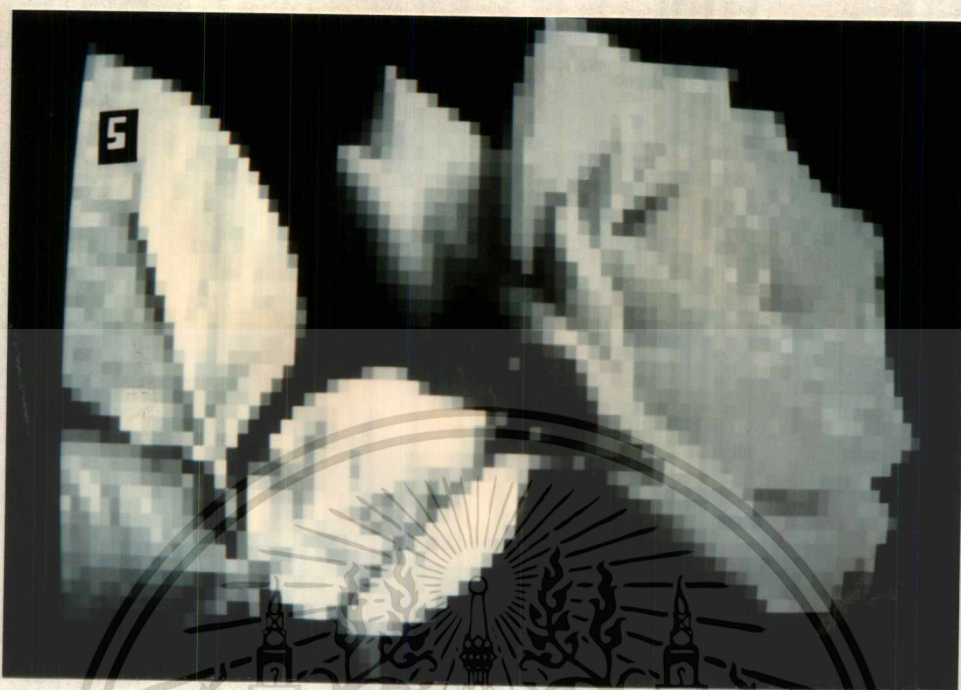


รูปที่ 1.3 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี LUMA 2 : 1



รูปที่ 1.4 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี LUMA 2 : 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.5 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี LUMA 4:1



รูปที่ 1.6 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี LUMA 4 : 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.7 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี LZW



รูปที่ 1.8 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี LZW

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.9 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี JPEG 3



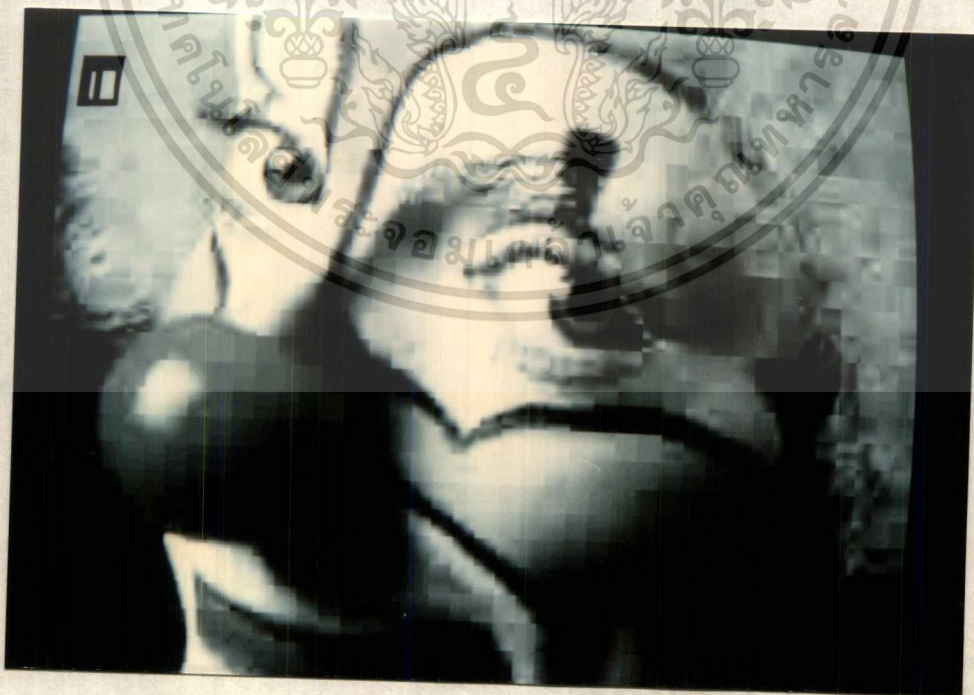
รูปที่ 1.10 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี JPEG 3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.11 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี JPEG 10



รูปที่ 1.12 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี JPEG 10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.13 แสดงผลการบีบอัดข้อมูล รูป ROSE ด้วยวิธี JPEG 25



รูปที่ 1.14 แสดงผลการบีบอัดข้อมูล รูป CLOWN ด้วยวิธี JPEG 25

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคที่ 2 ผลการประมวลผลภาพสี

ตารางที่ 8.5 แสดงผลการถอดข้อมูลจากไฟล์ .PCX

ชื่อแฟ้มข้อมูล	ขนาดแฟ้มข้อมูลก่อนทำการถอดข้อมูลภาพ (ไบต์)	ขนาดข้อมูลภาพที่ถอดได้ (ไบต์)	ชื่อไฟล์ผลลัพธ์
l.pcx	978085	921600	l.dat
g.pcx	1070833	921600	g.dat
big.pcx	923214	921600	big.dat
castle.pcx	943996	921600	castle.dat

ตารางที่ 8.6 แสดงผลการบีบอัดข้อมูลภาพสี

วิธี \ ชื่อภาพ	l.dat	g.dat	castle.dat	big.dat
CNO411	230400 75%	230400 75%	230400 75%	230400 75%
C411	460800 50%	460800 50%	460800 50%	460800 50%
C422	614400 33%	614400 33%	614400 33%	614400 33%
CR4 2 ทิศทาง 4:1:1	460800 50%	460800 50%	460800 50%	460800 50%
CR5 2 ทิศทาง 16:1:1	345600 62.5%	345600 62.5%	345600 62.5%	345600 62.5%
LZW	1038500 - 12%	934410 - 1%	1019348 -10%	802977 13%
JPEG 1	138961 85.9%	155716 83.1%	195912 78.7%	141752 84.6%
JPEG 10	52556 94.2%	57252 93.7%	51784 94.3%	51784 94.3%
JPEG 25	37659 95.9%	38386 95.8%	42996 95.3%	37396 95.9%

ตารางที่ 8.7 แสดงค่าความผิดพลาดของข้อมูล สำหรับวิธีเจเพก

วิธี \ ชื่อภาพ	l.dat	g.dat	castle.dat	big.dat
JPEG 1	4.71	4.47	10.50	5.84
JPEG 10	17.40	20.59	24.76	18.38
JPEG 25	23.99	27.40	33.19	23.76

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 8.8 แสดงเวลาที่ใช้ในการบีบข้อมูลโดยประมาณ (จากการจับเวลาตั้งแต่สิ้นวัน
โปรแกรมจนกระทั่งวันเสร็จ) หน่วยเป็น วินาที

วิธี \ ชื่อภาพ	l.dat	g.dat	castle.dat	big.dat
CNO411	5	5	5	5
C411	21	21	22	21
C422	28	27	28	28
CR4	19	19	20	20
CR5	19	19	19	19
LZW	71	34	46	34
JPEG 1	70	72	73	69
JPEG 10	69	69	69	68
JPEG 25	69	67	68	67

ตารางที่ 8.9 แสดงเวลาที่ใช้ในการขยายข้อมูลโดยประมาณ (จากการจับเวลาตั้งแต่สิ้นวัน
โปรแกรมจนกระทั่งวันเสร็จ) หน่วยเป็น วินาที

วิธี \ ชื่อภาพ	l.dat	g.dat	castle.dat	big.dat
CNO411	4	5	4	4
C411	19	20	20	19
C422	21	22	21	21
CR4	21	21	21	21
CR5	20	20	20	20
LZW	45	30	46	30
JPEG 1	65	72	72	66
JPEG 10	63	65	69	62
JPEG 25	65	67	67	62

ตารางที่ 8.10 แสดงจำนวนบิตเฉลี่ยที่ใช้ในการเข้ารหัสข้อมูล (บิตต่อพิกเซล)

วิธี \ ชื่อภาพ	l.dat	g.dat	castle.dat	big.dat
CNO411	6	6	6	6
C411	12	12	12	12
C422	15.9	15.9	15.9	15.9
CR4	12	12	12	12
CR5	9	9	9	9
LZW	27	24.3	26.4	21
JPEG 1	3.6	4.2	5.1	3.6
JPEG 10	1.5	1.5	1.5	1.5
JPEG 25	0.9	0.9	0.9	0.9

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.1 แสดงภาพ LISA ต้นฉบับ

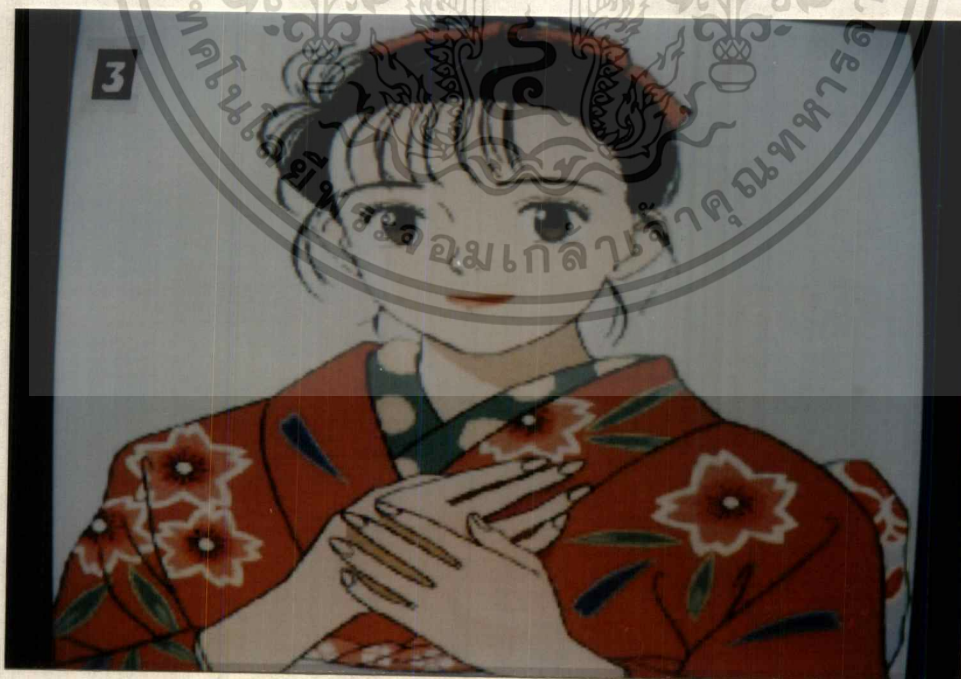


รูปที่ 2.2 แสดงภาพ BIG ต้นฉบับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี CNO411
(Chroma 4:1:1 ไม่แปลง RGB เป็น YCrCb)



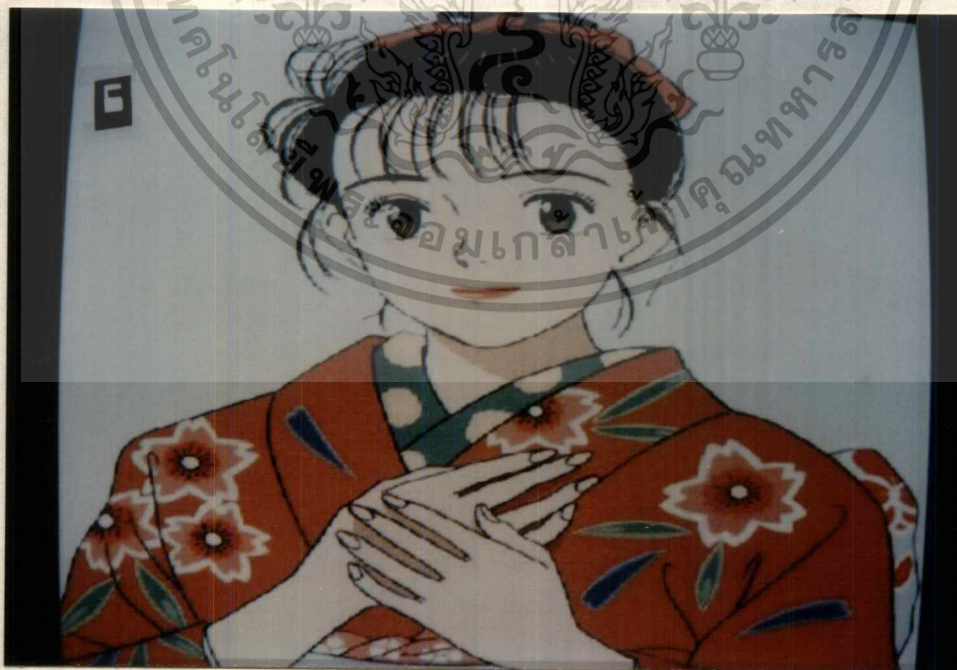
รูปที่ 2.4 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี CNO411

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



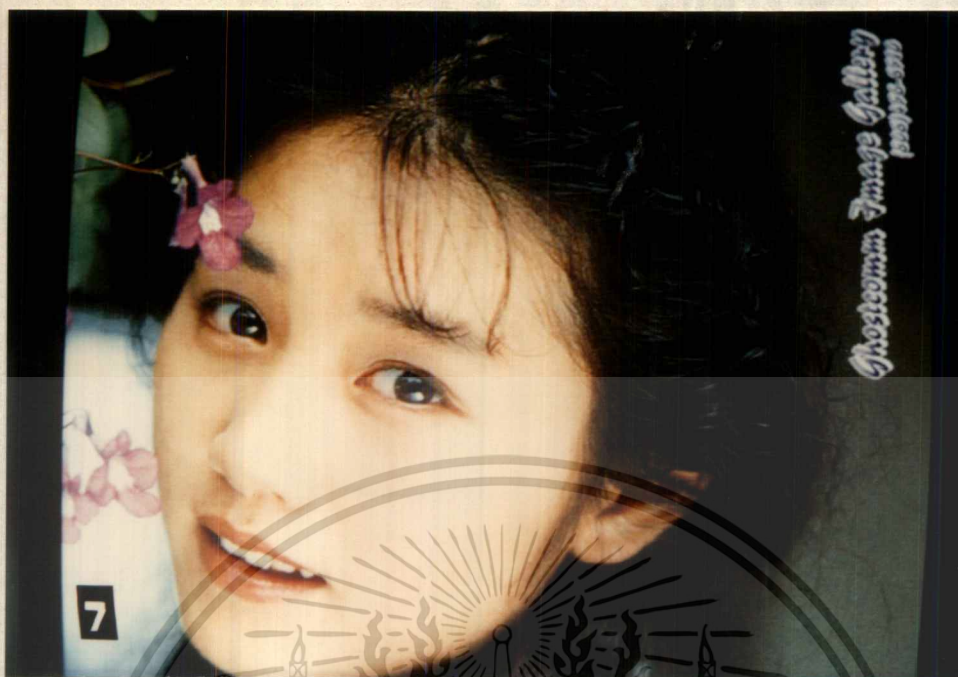
รูปที่ 2.5 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี C411
(Chroma 4:1:1 แปลง RGB เป็น YCrCb)



รูปที่ 2.6 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี C411

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.7 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี CR4
(Chroma 4:1:1 แบบ 2 ทิศทาง)



รูปที่ 2.8 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี CR4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



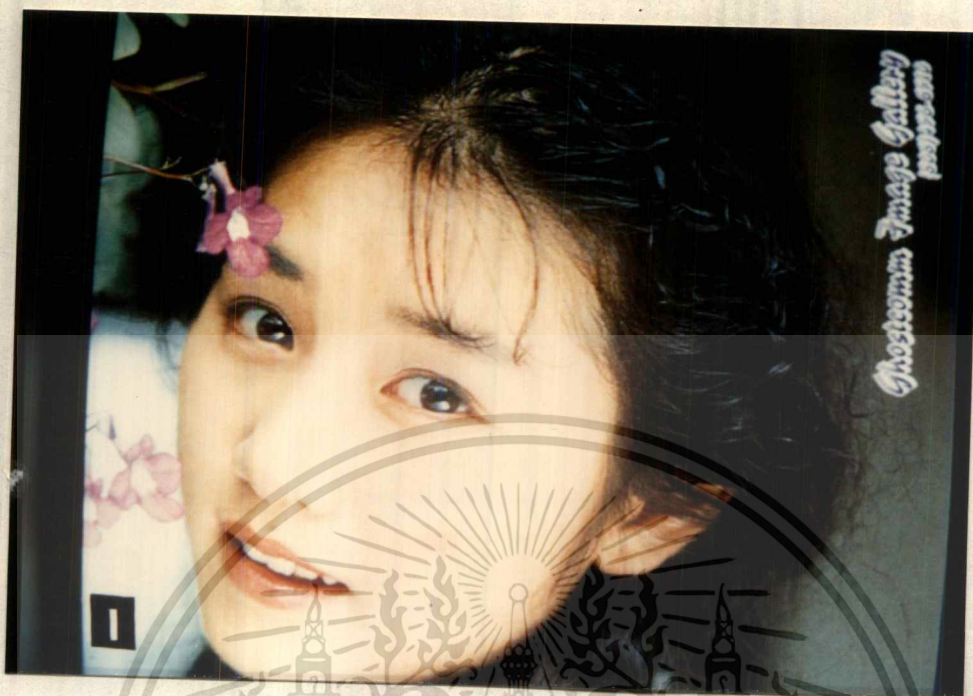
รูปที่ 2.9 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี CR5
(Chroma 16:1:1 แบบ 2 ทิศทาง)



รูปที่ 2.10 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี CR5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.11 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี LZW

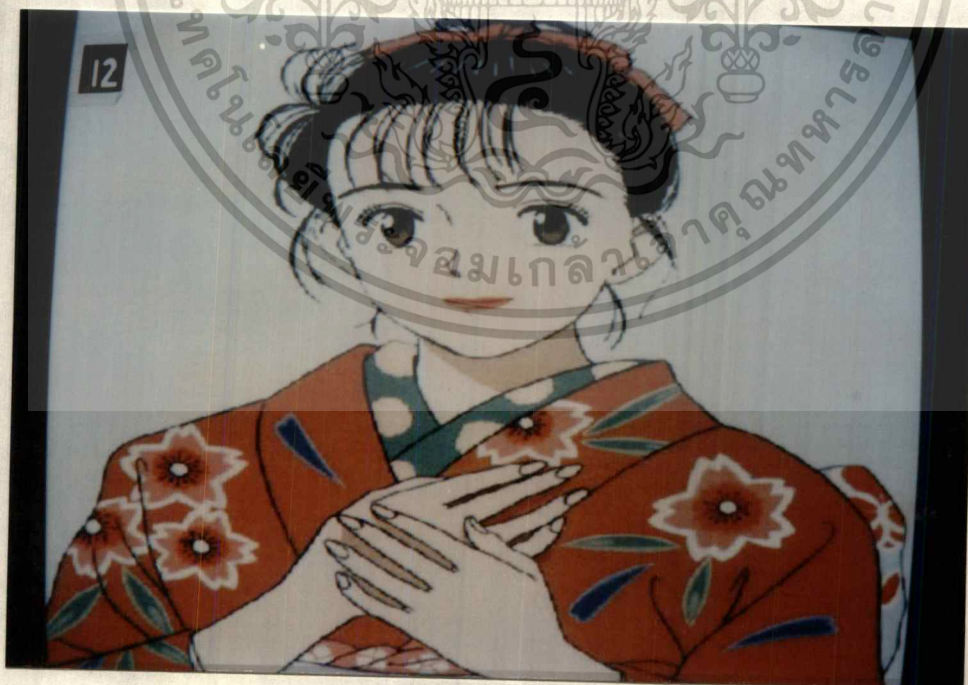


รูปที่ 2.12 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี LZW

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.13 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี JPEG 1



รูปที่ 2.14 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี JPEG 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.15 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี JPEG 10



รูปที่ 2.16 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี JPEG 10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

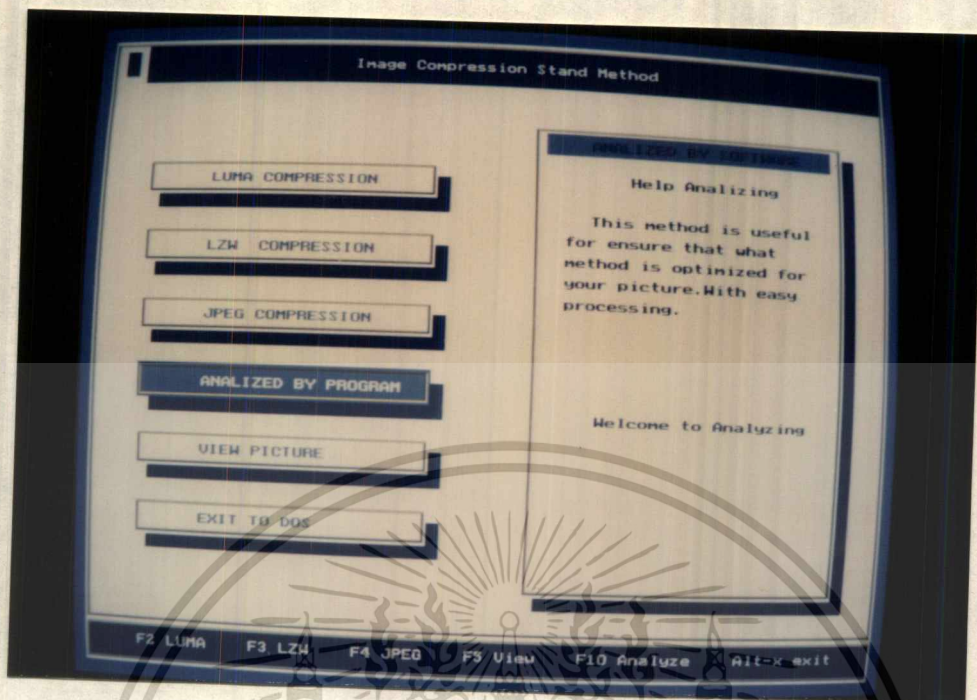


รูปที่ 2.17 แสดงผลการบีบอัดข้อมูล รูป LISA ด้วยวิธี JPEG 25

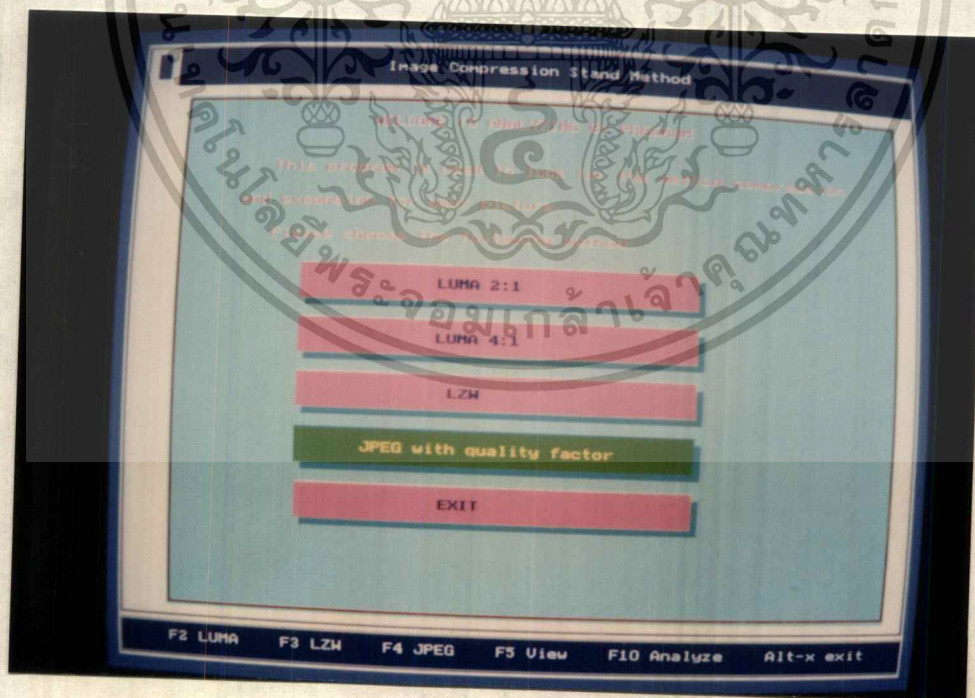


รูปที่ 2.18 แสดงผลการบีบอัดข้อมูล รูป BIG ด้วยวิธี JPEG 25

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

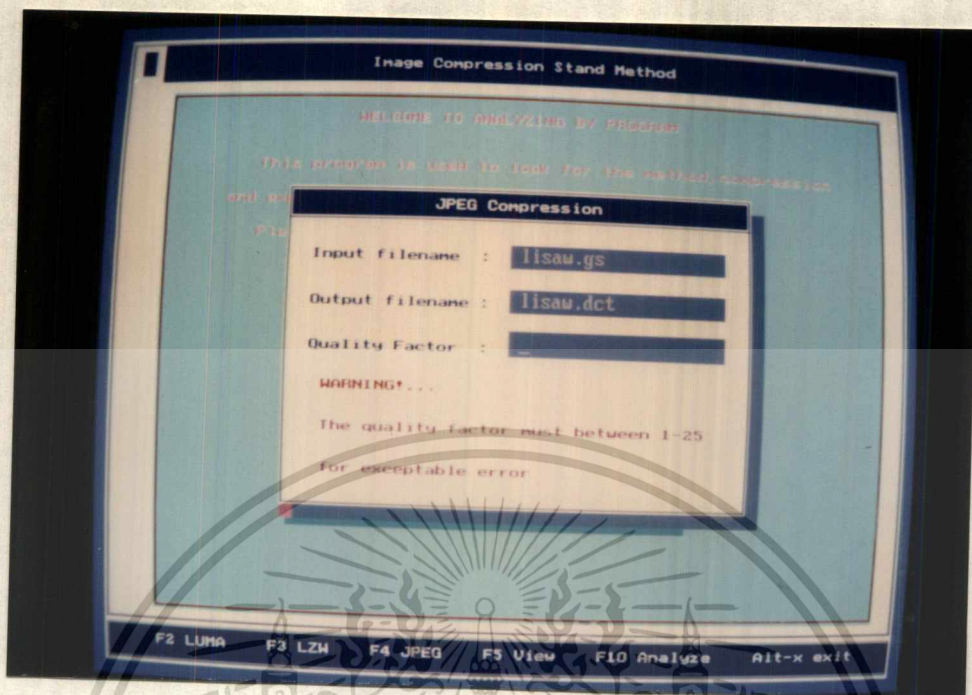


รูปที่ 2.19 แสดงการเขียนเมนูเพื่อช่วยวิเคราะห์ผลการบีบอัดข้อมูลภาพขาว - ดำ

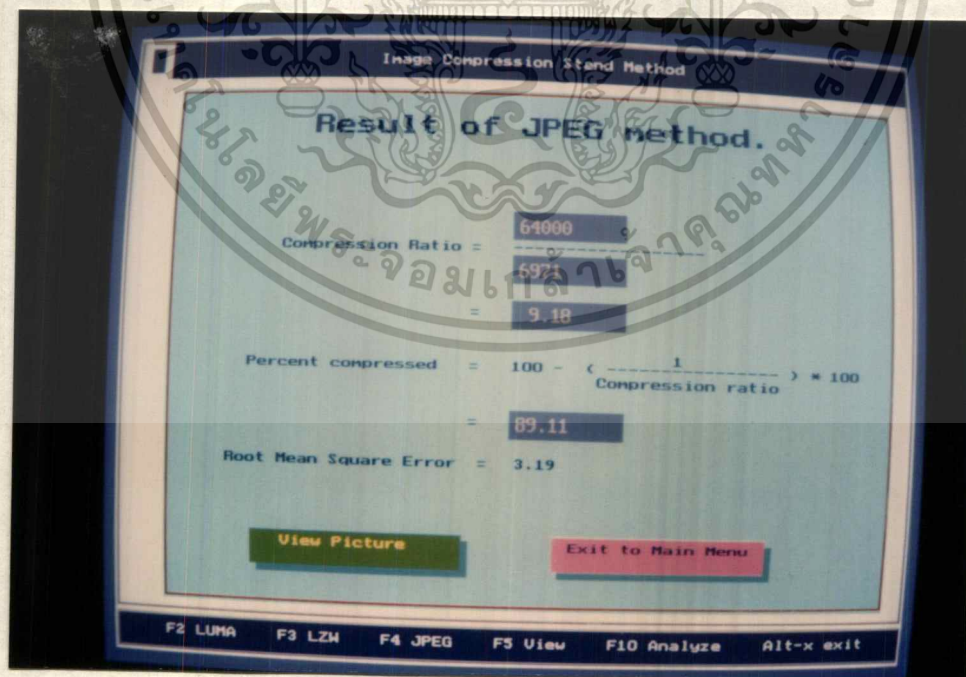


รูปที่ 2.20 แสดงเมนูเพื่อเลือกวิธีการที่จะนำภาพที่มีอยู่มาทำการวิเคราะห์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.21 แสดงหน้าจอเพื่อรับค่าไฟล์อินพุต ไฟล์เอาต์พุต และค่าแฟกเตอร์คุณภาพ สำหรับกระบวนการเจเพก



รูปที่ 2.22 แสดงผลการวิเคราะห์ภาพอินพุต เมื่อทำการบีบอัดด้วยวิธีที่กำหนด เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 9

สรุปและวิเคราะห์ผลการทดลอง

ภาคที่ 1 ภาพขาว - ดำ

1. อัตราการบีบอัดข้อมูล

1.1 วิธีที่สามารถลดขนาดข้อมูลได้มากที่สุดคือ LUMA 4 : 1 รองลงมาคือ JPEG 25 , 10 , 3 ตามลำดับ น้อยที่สุด คือ LZW

วิธี LZW จากลักษณะของภาพ CLOWN ที่มีความแตกต่าง ของระดับความเข้มกระจายอย่างสม่ำเสมอตลอดทั้งภาพ นอกจากจะไม่สามารถลดขนาดของข้อมูลแล้ว ยังเป็นการเพิ่มขนาดของข้อมูลเนื่องมาจากกระบวนการเสียด้วยซ้ำ สังเกตจากค่าเปอร์เซ็นต์การลดขนาดเป็นค่าลบ ในขณะที่ภาพ ROSE เป็นภาพที่มีสีใกล้เคียงกันอยู่ในตำแหน่งที่ใกล้ๆ กัน ทำให้สามารถลดขนาดด้วยวิธี LZW ได้มาก ดังนั้นจะเห็นได้ว่า วิธีของ LZW เหมาะสมสำหรับที่มีสีต่อเนื่องกัน อย่างเช่น ภาพการ์ตูน เป็นต้น และจะไม่เหมาะสมอย่างยิ่งกับภาพที่หลากหลายสีสัน กระจายอยู่ทั่วภาพ

1.2 วิธี JPEG ที่มีค่าแฟกเตอร์คุณภาพสูง จะช่วยลดขนาดข้อมูลได้มากขึ้น

2. คุณภาพของภาพ

2.1 วิธีที่ไม่ทำให้ข้อมูลเกิดการผิดเพี้ยนเลย คือ LZW รองลงมา คือ JPEG 3 , 10 , 25 ตามลำดับ ข้อมูลเพี้ยนมากที่สุด เมื่อใช้วิธี LUMA

วิธี LUMA ให้คุณภาพที่ไม่ค่อยดีนัก จึงเหมาะกับภาพที่ไม่ต้องการรายละเอียดมากนัก ในขณะที่วิธี LZW ข้อมูลไม่ผิดไปจากเดิมเลย เหมาะกับภาพที่ต้องการรายละเอียดสูง เช่น ภาพทางการแพทย์ ส่วนวิธี JPEG ใช้ได้กับภาพที่ต้องการรายละเอียดพอสมควร

2.2 วิธี JPEG มีข้อดีที่สามารถเลือกค่าแฟกเตอร์คุณภาพได้ แต่ค่าแฟกเตอร์คุณภาพที่มีค่ามาก จะทำให้ข้อมูลผิดพลาดมากขึ้นตามไปด้วย และไม่ว่าลักษณะของภาพจะมีลักษณะการกระจายตัวของสีต่างกันอย่างไร ก็สามารถลดขนาดได้อย่างเป็นที่น่าพอใจ

3. ความเร็วในการประมวลผลข้อมูล

เนื่องจาก LUMA เป็นวิธีการแบบบิตแมพ (Bit Mapped) จึงใช้เวลาในการคำนวณน้อยมาก ในขณะที่ JPEG ต้องใช้กระบวนการทางคณิตศาสตร์มากมาย จึงเสียเวลามาก

4. จำนวนบิตเฉลี่ยที่ใช้ในการเข้ารหัส โดยรวมแล้ววิธีเจเพกที่ค่าแฟกเตอร์คุณภาพมากๆ จะเข้ารหัสด้วยจำนวนบิตที่น้อยกว่าเจเพกที่ค่าน้อย เป็นผลให้ความละเอียดของข้อมูลหรือความถูกต้องของภาพมีน้อยกว่า ในขณะที่วิธีสุ่มค่าความสว่าง แบบ 2:1 จะเข้ารหัสด้วยจำนวนบิตที่มากกว่า 4:1 เสมอ โดยที่วิธี LZW นั้นจำนวนบิตของรหัสไม่แน่นอนขึ้นอยู่กับความสามารถในการลดขนาดภาพในขณะที่ได้ภาพไม่เพี้ยน

จะเห็นได้ว่า JPEG เป็นวิธีที่เหมาะสมกับภาพทุกชนิด ในขณะที่วิธีอื่นต้องพิจารณาผลที่จะเกิดขึ้นประกอบไปด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคที่ 2 ภาพสี

1. อัตราการบีบอัดข้อมูล

วิธีที่สามารถลดขนาดข้อมูลได้มากที่สุด คือ วิธีสุ่มค่าสี วิธี เจเพก และ LZW ตามลำดับ

1.1 การสุ่มค่าสีโดยที่ไม่มีการแปลงข้อมูลภาพ จาก RGB เป็น YCrCb จะลดขนาดได้มาก แต่ภาพจะเกิดการผิดเพี้ยนมากกว่าภาพที่มีการแปลงข้อมูลเสียก่อน ในขณะที่วิธี 4 : 1 : 1 ทั้ง 1 และ 2 ทิศทางจะให้ขนาดข้อมูลที่ลดได้เท่ากัน และเมื่อเปรียบเทียบกับ CR4 กับ CR5 จะเห็นว่า ยิ่งกำหนดจำนวนพิกเซลที่ติดกันมากๆ เพื่อหาค่าเฉลี่ย ก็จะสามารถลดขนาดลงได้มากขึ้น

1.2 วิธี LZW ก็ยังคงไม่สามารถลดขนาดของภาพส่วนใหญ่ได้ โดยทำได้ก็แต่เพียงภาพที่มีสีต่อเนื่องกันมากๆ เท่านั้น

1.3 วิธีเจเพก ที่มีค่าแฟกเตอร์คุณภาพมาก จะสามารถลดขนาดได้มาก

2.คุณภาพของภาพที่ได้

LZW ให้คุณภาพที่ดีเหมือนเดิม ในขณะที่วิธีเจเพก จะเริ่มผิดเพี้ยนไปจากเดิมมากขึ้น เมื่อค่าแฟกเตอร์คุณภาพเพิ่มขึ้น (จากการทดลอง เราพบว่าเจเพกที่ค่าแฟกเตอร์คุณภาพเท่ากับ 1 ภาพเริ่มผิดเพี้ยน แล้วพอเพิ่มค่าแฟกเตอร์คุณภาพเป็น 3 ภาพจะผิดไปจากเดิมอย่างมากเลยทีเดียว จนกระทั่งค่าแฟกเตอร์คุณภาพที่เท่ากับ 10 ภาพเริ่มผิดจนเกือบยอมรับไม่ได้)

วิธีสุ่มค่าสีของภาพ ความผิดพลาดของข้อมูลที่ถูกระงับตามแนวตั้งกับตามแนวนอนใกล้เคียงกัน เมื่อคิดแบบ 1 ทิศทางเทียบกับ 2 ทิศทางแล้ว แบบ 2 ทิศทางจะให้ภาพที่คล้ายต้นฉบับมากกว่า และเมื่อทำวิธีที่คิดจากพิกเซลที่ติดกันมากๆ ถึงแม้จะลดขนาดได้มาก แต่คุณภาพก็จะแย่ตามไปอีกด้วย

3.เวลาที่ใช้ในการคำนวณ

3.1 วิธีสุ่มค่าความสว่างใช้เวลาอย่างน้อยที่สุดเมื่อไม่มีการแปลงข้อมูล แล้วเริ่มใช้เวลามากขึ้นเมื่อต้องทำการคำนวณแบบ 1 และ 2 ทิศทาง ตามลำดับ

3.2 วิธี LZW ยังคงต้องใช้เวลา ในการสร้างตารางรหัส เข้ารหัสข้อมูล

3.3 วิธีเจเพก ยังคงใช้เวลาในการคำนวณต่างๆ มากมายอยู่เหมือนเดิม ซึ่งเมื่อค่าแฟกเตอร์คุณภาพถึงแม้จะมีค่าที่ต่างกันแต่เวลาที่ใช้ก็ไม่ต่างกันนัก เกือบจะเท่ากันอีกด้วย แต่จะพอเห็นได้ว่าภาพการ์ตูนจะใช้ เวลาในการประมวลผลที่น้อยกว่า เพราะเป็นสีที่ต่อเนื่อง สามารถคำนวณได้ง่าย

4.จำนวนบิตเฉลี่ยที่ใช้ในการเข้ารหัสแปรไปตามขนาดไฟล์ผลลัพธ์ และจะมีผลต่อความถูกต้องของภาพ

4.1 วิธีสุ่มค่าสีจะมีอัตราที่คงที่สำหรับภาพที่มีขนาดเท่ากัน แต่ความเพี้ยนของภาพยังขึ้นอยู่กับลักษณะของภาพด้วย ภาพที่สีใกล้เคียงกันจะเพี้ยนน้อย แต่ภาพที่หลากสีจะเพี้ยนได้มาก

4.2 วิธี LZW นั้น จำนวนบิตที่เข้ารหัส ขึ้นกับความสามารถในการลดขนาดข้อมูล เพราะจะให้ภาพที่ไม่ผิดเพี้ยนเสมอ ถ้าข้อมูลเหมือนกันมาก ก็เข้ารหัสด้วยจำนวนบิตน้อย แต่ถ้าเป็นภาพหลากสี ข้อมูลจะกระจาย ต้องเข้ารหัสด้วยจำนวนบิตมาก

4.3 วิธี เจเพก จำนวนบิต และคุณภาพของภาพที่ได้ ขึ้นกับค่าแฟกเตอร์คุณภาพที่เลือกใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 10

แนวทางพัฒนา

ภาคที่ 1 ภาพขาว - ดำ

1. ศึกษาและทำการเขียนซอฟต์แวร์ที่ช่วยวิเคราะห์เพื่อการตัดสินใจ ว่า ควรใช้วิธีการบีบอัดข้อมูลวิธีใด กับภาพต่างๆ ที่มีอยู่
2. ศึกษาและทำการหาข้อมูลรูปภาพอื่น ๆ ที่ไม่ใช่ไฟล์ข้อมูลรูปภาพ (.GS)
3. ศึกษาและทำการบีบอัดและขยายข้อมูลภาพสี

ภาคที่ 2 ภาพสี

แนวทางที่เป็นที่น่าสนใจ สำหรับผู้ที่สนใจทั่วไป คือ

1. ศึกษาและทำการเขียนเป็นโปรแกรมสำเร็จรูป เพื่อการใช้งานให้สะดวกขึ้น
2. จากการศึกษาการบีบอัดข้อมูลภาพเพื่อการเก็บลงอุปกรณ์เก็บข้อมูล ก็อาจศึกษาเพื่อใช้ประยุกต์ในการส่งข่าวสารระหว่างเครื่องคอมพิวเตอร์ 2 เครื่อง หรือ ผ่านโครงข่าย



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

/* โปรแกรมหลักของเมนูสำหรับการบีบข้อมูล, ส่วนวิเคราะห์การบีบ

ข้อมูล, ส่วนแสดงผลข้อมูลรูปภาพออกจอภาพ */

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <graphics.h>
#include <io.h>
#include "lzw12.c"
#include "newdct.c"
#include "luma2.c"
#include "luma4.c"
#include "bitio.c"
#include "errhand.c"
```

/* กำหนดค่าคงที่ของรหัสแทนโค้ดของคีย์บอร์ด */

```
#define UP      0x4800
#define DOWN    0x5000
#define LEFT    0x4B00
#define RIGHT   0x4D00
#define RETURN  0x1C0D
#define ESC     0x011B
#define F1      0x3b00
#define F2      0x3c00
#define F3      0x3d00
#define F4      0x3e00
#define F5      0x3f00
#define F10     0x4400
#define Altx    0x2d00
```

/* กำหนดตัวแปรชนิดโกลบอลของโปรแกรม */

```
long      lzwsz, jpegsz, luma2sz, luma4sz;
char      *lzwratio, *jpegratio, *luma2ratio, *luma4ratio;
char      *lzwsizestr, *jpegsizestr;
char      *luma2sizestr, *luma4sizestr;
int        x, y, i, j, length, quality;
float      rmserr;
unsigned long fsz;
char        distortchk, dispchk;
char        fsizestr[10], rmsstr[5], qstr[5];
char        lzwnam[20];
char        jpegnam[20];
char        luma2nam[20], luma4nam[20];
char        inputnam[20], outputnam[20];
char        *extension;
char        com[50];
int         chk, chkopen;
int         view_type = 0;
int         choice, anachoice, num, filtype;
int         exitview = 0;
```

```
BIT_FILE   *output;
FILE        *input, *outputfile;
FILE        *lzwfile;
FILE        *jpegfile;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

FILE          *luma2;
FILE          *luma4;

void          *saveview[10];
void          *mem[10];

/* กำหนดโปรโตไทป์ */

void initialize_graphics_mode();
void close_graphics_mode();
void drawmain();
void drawblock(int xs,int ys,int xl,int yl);
void drawblock1(int xs,int ys,int xl,int yl);
void drawblock2(int xs,int ys,int xl,int yl);
void drawblock3(int xs,int ys,int xl,int yl,char *text);
void saveblock(int xs,int ys,int xl,int yl,int num);
void saveblockmain();
void selectmain();
void menuhelp(int choice);
void exitmain();
void viewpic();
void Luma();
void Lumasub2();
void Lumasub4();
void Lzw();
void Jpeg();
void getstr(int x,int y,int quality);
void getstr1(int x,int y,int length,char text[]);
void LUMA2();
void LUMA4();
void LZW();
void JPEGq();
void EXIT();
void selectanalyze();
void drawmainy();
void drawblockly(int xs,int ys,int xl,int yl);
void drawblocky(int xs,int ys,int xl,int yl);
void Getparameter(char inputname[20]);
int openinput(char *inputname);
void openoutput(char *outputname);
void drawblockly(int xs,int ys,int xl,int yl);
void drawblocky(int xs,int ys,int xl,int yl);
void drawmainy();
void Exit();
void selectview();
void view();

/* โปรแกรมหลัก */

void main()
{
    clrscr();
    initialize_graphics_mode();
    drawmain();
    saveblockmain();
    selectmain();
    closegraph();
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* รูทีนที่ใช้ในการปรับจอภาพเข้าสู่โหมดกราฟิก */
```

```
void initialize_graphics_mode()
{
    int gdriver =DETECT,gmode,errorcode;
    initgraph(&gdriver,&gmode,"d:\\turbo\\source");
    errorcode = graphresult();
    if (errorcode != grOk)
    {
        printf(" Graphics error : %s\n",grapherrormsg(errorcode));
        printf(" Press any key halt : ");
        getch();
        exit(1);
    }
}
```

```
/* ฟังก์ชันสำหรับปรับจอภาพเข้าสู่โหมดข้อความ */
```

```
void close_graphics_mode()
{
    closegraph();
}
```

```
/* รูทีนนี้ใช้สำหรับแสดงภาพของเมนูหลัก */
```

```
void drawmain()
{
    int x,y;

    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(LIGHTBLUE);
    setcolor(LIGHTGRAY);
    drawblock1(10,10,getmaxx()-10,getmaxy()-40);
    drawblock1(10,getmaxy()-40,getmaxx()-10,getmaxy()-10);
    setfillstyle(1,1);
    bar(40,14,getmaxx()-14,41);
    bar(15+30/4,18,10+30*3/4,36);
    bar(14,getmaxy()-36,getmaxx()-12,getmaxy()-14);
    settextstyle(0,0,1);
    setcolor(WHITE);
    outtextxy(160,24,"      Image Compression Stand Method  ");
    outtextxy(45,getmaxy()-30," F2 LUMA      F3 LZW      F4 JPEG\
F5 View      F10 Analyze      Alt-x exit");
    drawblock(50,100,275,125);
    drawblock(50,150,275,175);
    drawblock(50,200,275,225);
    drawblock(50,250,275,275);
    drawblock(50,300,275,325);
    drawblock(50,350,275,375);
    setcolor(BLACK);
    outtextxy(70,110,"      LUMA COMPRESSION  ");
    outtextxy(70,160,"      LZW  COMPRESSION  ");
    outtextxy(70,210,"      JPEG COMPRESSION  ");
    outtextxy(70,260,"      ANALIZED BY PROGRAM ");
    outtextxy(70,310,"      VIEW PICTURE      ");
    outtextxy(70,360,"      EXIT TO DOS      ");
    drawblock(350,70,getmaxx()-50,getmaxy()-70);
    setcolor(BLUE);
    setfillstyle(1,BLACK);
    bar(360,75,getmaxx()-60,90);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

outtextxy(380,80,"    LUMA COMPRESSION");
outtextxy(425,105,"Help Luma");
outtextxy(380,135,"This method can compress");
outtextxy(375,150,"with 75% for Luma 2:1");
outtextxy(375,165,"and 93.5% for Luma 4:1");
outtextxy(395,180,"If you does not serious");
outtextxy(375,195,"about error and the time");
outtextxy(375,210,"it has spent,you might");
outtextxy(375,225,"choose this method.");
outtextxy(395,240,"Error from Luma 2:1");
outtextxy(375,255,"are less than error from");
outtextxy(375,270,"Luma 4:1");
outtextxy(400,300,"Welcome to Luma method.");
}

```

/* รูทีนสำหรับแสดงรูปบล็อก */

```

void drawblock(int xs,int ys,int xl,int yl)
{
    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(LIGHTBLUE);
    setfillstyle(1,1);
    bar(xs+10,ys+10,xl+10,yl+10);
    setfillstyle(1,WHITE);
    bar(xs,ys,xl,yl);
    setcolor(BLUE);
    rectangle(xs+2,ys+2,xl-2,yl-2);
    line(xs+3,ys+3,xl-3,ys+3);
    setcolor(LIGHTGRAY);
    rectangle(xs+3,ys+3,xl-3,yl-3);
    line(xs+2,ys+2,xl-2,ys+2);
}

```

/* รูทีนในการตรวจสอบการกดปุ่มคีย์บอร์ดสำหรับการเลือกทำงานตามทาง
เลือกในเมนูหลัก */

```

void selectmain()
{
    int i;
    int xs = 100;

    choice = 0;
    do {
        i = bioskey(0);

        switch(i)
        {
            case UP : if(choice == 0)
                        {
                            putimage(50,100,saveview[choice],0);
                            choice =5;
                            putimage(50,350,saveview[choice],4);
                            menuhelp(choice);
                        }
                    else
                    {
                        switch(choice)
                        {
                            case 1:xs=150;break;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        case 2:xs=200;break;
        case 3:xs=250;break;
        case 4:xs=300;break;
        case 5:xs=350;break;
    }
    putimage(50,xs,saveview[choice],0);
    choice =choice-1;
    xs = xs-50;
    putimage(50,xs,saveview[choice],4);
    menuhelp(choice);
}break;
case DOWN : if(choice == 5)
{
    putimage(50,350,saveview[choice],0);
    choice =0;
    putimage(50,100,saveview[choice],4);
    menuhelp(choice);
}
else
{
    switch(choice)
    {
        case 0:xs=100;break;
        case 1:xs=150;break;
        case 2:xs=200;break;
        case 3:xs=250;break;
        case 4:xs=300;break;
        case 5:xs=350;break;
    }
    putimage(50,xs,saveview[choice],0);
    choice =choice+1;
    xs = xs+50;
    putimage(50,xs,saveview[choice],4);
    menuhelp(choice);
} break;
case RETURN : switch(choice)
{
    case 0:Luma();
        drawmain();
        putimage(50,100,saveview[0],4);
        menuhelp(choice);
        break;
    case 1:Lzw();
        drawmain();
        putimage(50,150,saveview[1],4);
        menuhelp(choice); break;
    case 2:Jpeg();
        drawmain();
        putimage(50,200,saveview[2],4);
        menuhelp(choice);
        break;
    case 3:analyze();
        drawmain();
        putimage(50,250,saveview[3],4);
        menuhelp(choice);
        break;
    case 4:viewpic();
        drawmain();
        putimage(50,300,saveview[4],4);
        menuhelp(choice);
        break;
    case 5:exitmain();
        drawmain();
        putimage(50,350,saveview[5],4);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาค้นคว้าเท่านั้น ไม่อนุญาตให้เผยแพร่โดยไม่ได้รับอนุญาตจากผู้นิพนธ์

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        menuhelp(choice);
        break;
    }break;
case F2 :putimage(50,100+50*choice,saveview[choice],0);
        putimage(50,100,saveview[0],4);
        choice=0;
        menuhelp(choice);
        Luma();
        drawmain();
        putimage(50,100,saveview[0],4);
        menuhelp(choice);
        break;
case F3 :putimage(50,100+50*choice,saveview[choice],0);
        putimage(50,150,saveview[1],4);
        choice=1;
        menuhelp(choice);
        Lzw();
        drawmain();
        putimage(50,150,saveview[1],4);
        menuhelp(choice);
        break;
case F4 :putimage(50,100+50*choice,saveview[choice],0);
        putimage(50,200,saveview[2],4);
        choice=2;
        menuhelp(choice);
        Jpeg();
        drawmain();
        putimage(50,200,saveview[2],4);
        menuhelp(choice);
        break;
case F5 :putimage(50,100+50*choice,saveview[choice],0);
        putimage(50,300,saveview[4],4);
        choice=4;
        menuhelp(choice);
        viewpic();
        drawmain();
        putimage(50,300,saveview[4],4);
        menuhelp(choice);
        break;
case F10 :putimage(50,100+50*choice,saveview[choice],0);
        putimage(50,250,saveview[3],4);
        choice=3;
        menuhelp(choice);
        analyze();
        drawmain();
        putimage(50,250,saveview[3],4);
        menuhelp(choice);
        break;
case AltX :putimage(50,100+50*choice,saveview[choice],0);
        putimage(50,350,saveview[5],4);
        choice=5;
        menuhelp(choice);
        exitmain();
        drawmain();
        putimage(50,350,saveview[5],4);
        menuhelp(choice);
        break;
    } /*switch*/
} while( i != ESC );
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* รูทีนสำหรับเก็บบล็อกข้อมูลบนจอภาพลงบนตัวแปร saveview */
```

```
void saveblock(int xs,int ys,int xl,int yl,int num )
{
    unsigned int size;

    size =imagesize(xs,ys,xl,yl);
    saveview[num] = malloc(size);
    getimage(xs,ys,xl,yl,saveview[num]);
}
```

```
/* รูทีนสำหรับเก็บบล็อกข้อมูลบนจอภาพลงบนตัวแปร mem */
```

```
void saveblock1(int xs,int ys,int xl,int yl,int num )
{
    unsigned int size;

    size =imagesize(xs,ys,xl,yl);
    mem[num] = malloc(size);
    getimage(xs,ys,xl,yl,mem[num]);
}
```

```
/* รูทีนสำหรับเก็บบล็อกข้อมูลทางเลือกของเมนูหลัก */
```

```
void saveblockmain()
{
    saveblock(50,100,275,125,0);
    saveblock(50,150,275,175,1);
    saveblock(50,200,275,225,2);
    saveblock(50,250,275,275,3);
    saveblock(50,300,275,325,4);
    saveblock(50,350,275,375,5);
    putimage(50,100,saveview[0],4);
    setcolor(BLUE);
    setfillstyle(1,BLACK);
    bar(360,75,getmaxx()-60,90);
    outtextxy(380,80,"    LUMA COMPRESSION");
    outtextxy(425,105,"Help Luma");
    outtextxy(380,135,"This method can compress");
    outtextxy(375,150,"with 75% for Luma 2:1");
    outtextxy(375,165,"and 93.5% for Luma 4:1");
    outtextxy(395,180,"If you does not serious");
    outtextxy(375,195,"about error and the time");
    outtextxy(375,210,"it has spent,you might");
    outtextxy(375,225,"choose this method.");
    outtextxy(395,240,"Error from Luma 2:1");
    outtextxy(375,255,"are less than error from");
    outtextxy(375,270,"Luma 4:1");
    outtextxy(400,300,"Welcome to Luma method.");
}
```

```
/* รูทีนแสดงคำอธิบายคร่าว ๆ ของแต่ละทางเลือก */
```

```
void menuhelp(int choice)
{
    setfillstyle(1,WHITE);
    bar(370,100,getmaxx()-60,getmaxy()-80);
    setfillstyle(1,BLACK);
    bar(360,75,getmaxx()-60,90);
    setcolor(BLUE);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

switch(choice)
{
case 0: outtextxy(380,80,"    LUMA COMPRESSION");
        outtextxy(425,105,"Help Luma");
        outtextxy(380,135,"This method can compress");
        outtextxy(375,150,"with 75% for Luma 2:1");
        outtextxy(375,165,"and 93.5% for Luma 4:1");
        outtextxy(395,180,"If you does not serious");
        outtextxy(375,195,"about error and the time");
        outtextxy(375,210,"it has spent,you might");
        outtextxy(375,225,"choose this method.");
        outtextxy(395,240,"Error from Luma 2:1");
        outtextxy(375,255,"are less than error from");
        outtextxy(375,270,"Luma 4:1");
        outtextxy(400,300,"Welcome to Luma method.");
        outtextxy(380,80,"    LUMA COMPRESSION");
        break;
case 1: outtextxy(380,80,"    LZW COMPRESSION");
        outtextxy(425,105,"Help Lzw");
        outtextxy(388,130,"This method is dangerous");
        outtextxy(375,150,"because it's optimized");
        outtextxy(375,165,"for no more difference");
        outtextxy(375,179,"color like a cartoon.");
        outtextxy(375,195,"But if your picture has");
        outtextxy(375,210,"many difference color");
        outtextxy(375,225,"you ought to look for");
        outtextxy(375,240,"JPEG.");
        outtextxy(395,255,"Because if you use LZW");
        outtextxy(375,270,"You might recieve bigger");
        outtextxy(375,285,"memory than original.");
        outtextxy(395,300,"This method decompress");
        outtextxy(375,315,"with no error");
        outtextxy(400,350,"Welcome to LZW method.");
        break;
case 2: outtextxy(380,80,"    JPEG COMPRESSION");
        outtextxy(425,105,"Help Jpeg");
        outtextxy(395,135,"This method is useful");
        outtextxy(375,150,"for every picture");
        outtextxy(375,165,"The quality factor is");
        outtextxy(375,180,"the method to take you to");
        outtextxy(375,195,"many values of error.");
        outtextxy(375,210,"The acceptable range is");
        outtextxy(375,225,"from 1 to 25,the minimum");
        outtextxy(375,240,"will has minimize error.");
        outtextxy(375,255,"You can choose quality");
        outtextxy(375,270,"by yourself or 3 if not.");
        outtextxy(395,285,"If you can wait this");
        outtextxy(375,300,"processing,use this method.");
        outtextxy(400,350,"Welcome to JPEG method.");
        break;
case 3: outtextxy(380,80,"    ANALIZED BY SOFTWARE");
        outtextxy(425,105,"Help Analizing");
        outtextxy(395,135,"This method is useful");
        outtextxy(375,150,"for ensure that what");
        outtextxy(375,165,"method is optimized for");
        outtextxy(375,180,"your picture.With easy");
        outtextxy(375,195,"processing.");
        outtextxy(400,280,"Welcome to Analyzing");
        break;
case 4: outtextxy(380,80,"    VIEW PICTURE");
        outtextxy(425,105,"Help View");
        outtextxy(395,135,"Welcome for showing");
        outtextxy(375,150,"your original or");

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        outtextxy(375,165,"decompressed picture,");
        outtextxy(375,180,"what you want");
        break;
    case 5: outtextxy(380,80,"      EXIT TO DOS");
            outtextxy(425,105,"Help Exit to Dos");
            outtextxy(395,135,"You can exit to DOS");
            outtextxy(375,150,"by 1) This way");
            outtextxy(375,165,"    2) Alt-X");
            outtextxy(395,220,"Good Luck");
            break;
    }
}

```

/* รุกี้นในการสอบถามผู้ใช้เพื่อความแน่ใจ ก่อนออกจากโปรแกรมนี้ */

```

void exitmain()
{
    int i,k;

    k=1;

    drawblock(170,130,450,250);
    drawblock2(200,190,300,215);
    drawblock2(330,190,430,215);
    setcolor(BLUE);
    outtextxy(200,150,"      Exit      ");
    outtextxy(195,170,"Are you sure you want to quit?");
    outtextxy(210,200,"    Quit");
    outtextxy(340,200,"    Cancel");
    saveblock(200,190,300,215,6);
    saveblock(330,190,430,215,7);
    putimage(330,190,saveview[7],4);
    do {
        i =bioskey(0);
        switch(i)
        {
            case LEFT : if(k == 0)
                        {
                            k=1;
                            putimage(330,190,saveview[7],4);
                            putimage(200,190,saveview[6],0);
                        }
                        else
                        {
                            k=0;
                            putimage(200,190,saveview[6],4);
                            putimage(330,190,saveview[7],0);
                        }
                        break;
            case RIGHT : if(k==1)
                        {
                            k=0;
                            putimage(200,190,saveview[6],4);
                            putimage(330,190,saveview[7],0);
                        }
                        else
                        {
                            k=1;
                            putimage(200,190,saveview[6],0);
                            putimage(330,190,saveview[7],4);
                        }
                        break;
        }
    }
}

```

```

        case RETURN : switch(k)
        {
            case 0:closegraph();exit(0);
            case 1: i = ESC;
        }
    } /*switch*/
} while(i != ESC);
}

```

/* รูทีนสำหรับแสดงรูปบล็อกแบบที่ 1 */

```

void drawblock1(int xs,int ys,int xl,int yl)
{
    setviewport(0,0,getmaxx(),getmaxy(),1);
    setfillstyle(1,WHITE);
    bar(xs,ys,xl,yl);
    setcolor(BLUE);
    rectangle(xs+2,ys+2,xl-2,yl-2);
    line(xs+3,ys+3,xl-3,ys+3);
    setcolor(LIGHTGRAY);
    rectangle(xs+3,ys+3,xl-3,yl-3);
    line(xs+2,ys+2,xl-2,ys+2);
}

```

/* รูทีนสำหรับแสดงรูปบล็อกแบบที่ 2 */

```

void drawblock2(int xs,int ys,int xl,int yl)
{
    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(LIGHTBLUE);
    setfillstyle(1,1);
    bar(xs+4,ys+4,xl+4,yl+4);
    setfillstyle(1,WHITE);
    bar(xs,ys,xl,yl);
    setcolor(BLUE);
    rectangle(xs+2,ys+2,xl-2,yl-2);
    line(xs+3,ys+3,xl-3,ys+3);
    setcolor(LIGHTGRAY);
    rectangle(xs+3,ys+3,xl-3,yl-3);
    line(xs+2,ys+2,xl-2,ys+2);
}

```

/* รูทีนสำหรับแสดงรูปบล็อกแบบที่ 3 */

```

void drawblock3(int xs,int ys,int xl,int yl,char *text)
{
    int xmid,ymid,l;

    xmid = (xs+xl)/2;
    ymid = (ys+20)-(20/2)-2;

    setcolor(BLUE);
    setfillstyle(1,LIGHTBLUE);
    bar(xs+10,ys+10,xl+10,yl+10);
    setfillstyle(1,WHITE);
    bar(xs,ys,xl,yl);
    setfillstyle(1,BLUE);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

bar(xs+3,ys+3,xl-3,ys+20);
rectangle(xs+1,ys+1,xl-1,yl-1);
rectangle(xs+1,ys+1,xl-1,ys+22);
settextjustify(1,2);
settextstyle(0,0,1);
setcolor(WHITE);
outtextxy(xmid,ymid,text);
settextjustify(0,2);
}

```

/* รูทีนที่ใช้ในการแสดงรูปภาพของไฟล์ข้อมูลรูปภาพต้นฉบับ
และไฟล์ที่ถูกบีบข้อมูลด้วยวิธีต่าง ๆ */

```

void viewpic()
{
    char inputname[20];
    char *extension;
    char com[50];
    int ext;

    drawblock3(150,120,500,350,"View picture");
    setcolor(1);
    setfillstyle(1,LIGHTBLUE);
    outtextxy(170,165,"Input filename : ");
    bar(320,160,480,176);
    getstrl(42,11,18,inputname);
    extension = strstr(inputname,".");
    extension++;
    if(!strcmp(extension,"gs")) ext = 1;
    if(!strcmp(extension,"lum")) ext = 2;
    if(!strcmp(extension,"lu4")) ext = 2;
    if(!strcmp(extension,"lzw")) ext = 3;
    if(!strcmp(extension,"dct")) ext = 4;
    switch(ext)
    {
        case 1 : view_gs(com,inputname);
                 break;
        case 2 : view_luma(com,inputname);
                 break;
        case 3 : view_lzw(com,inputname);
                 break;
        case 4 : view_jpeg(com,inputname);
                 break;
    }
    initialize_graphics_mode();
}

```

/* รูทีนแสดงทางเลือกของการบีบข้อมูลด้วยวิธีการ Luma subsampling */

```

void Luma()
{
    int i,k;int m;

    k=1;
    drawblock(170,130,450,250);
    drawblock2(200,190,300,215);
    drawblock2(330,190,430,215);
    setcolor(BLUE);
    outtextxy(200,150,"      Welcome to LUMA method.  ");
    outtextxy(200,170,"      What method do you want ?");
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

outtextxy(210,200,"LUMA 2:1");
outtextxy(340,200,"LUMA 4:1");
saveblock(200,190,300,215,8);
saveblock(330,190,430,215,9);
putimage(330,190,saveview[9],4);
do {
    i =bioskey(0);
    switch(i)
    {
        case LEFT : if(k == 0)
            {
                k=1;
                putimage(330,190,saveview[9],4);
                putimage(200,190,saveview[8],0);
            }
            else
            {
                k=0;
                putimage(200,190,saveview[8],4);
                putimage(330,190,saveview[9],0);
            }break;
        case RIGHT : if(k==1)
            {
                k=0;
                putimage(200,190,saveview[8],4);
                putimage(330,190,saveview[9],0);
            }
            else
            {
                k=1;
                putimage(200,190,saveview[8],0);
                putimage(330,190,saveview[9],4);
            } break;
        case RETURN : switch(k)
            {
                case 0:Lumasub2();
                i = ESC;
                break;
                case 1:Lumasub4();
                i = ESC;
                break;
            }
    } /*switch*/
} while(i != ESC);
}

```

/* รุ้ทีนหลักของการบีบข้อมูลด้วยวิธีการ Luma subsampling ชนิด 2 : 1 */

```

void Lumasub2()
{
do{
    drawblock3(150,120,500,350,"Luma subsampling 2:1 Compression");
    setcolor(BLUE);
    setfillstyle(1,LIGHTBLUE);
    outtextxy(170,165,"Input filename : ");
    bar(320,160,480,176);
    getstr1(41,11,18,inputname);
    extension = strstr(inputname,".");
    extension++;
    chk = strcmp(extension,"gs");
    if( chk != 0 )

```

```

    {
        drawblock1(170,275,470,325);
        setcolor(RED);
        outtextxy(185,295," Warning!..you must enter .gs file.");
        getch();
    }
    if( chk == 0 )
        chkopen = openinput(inputname);
    else chkopen = -1;
    } while( chk != 0 || chkopen != 0);

do{
    outtextxy(170,198,"Output filename : ");
    bar(320,192,480,208);
    getstr1(42,13,18,outputname);
    extension = strstr(outputname,".");
    extension++;
    chk = strcmp(extension,"lum");
    if( chk != 0 )
    {
        drawblock1(170,275,470,325);
        setcolor(RED);
        outtextxy(185,295," Warning!..you must enter .lum file");
        getch();
        setfillstyle(1,WHITE);
        bar(165,270,475,330);
        setfillstyle(1,LIGHTBLUE);
        setcolor(BLUE);
    }
    } while( chk != 0 );

    strcpy(luma2name,outputname);
    if( (outputfile = fopen(luma2name,"wb")) == NULL )
    {
        closegraph();
        printf("Error opening output file.");
        exit(1);
    }

    setcolor(BLUE);
    drawblock3(150,120,500,350,"Luma subsampling 2:1 Compression");
    setcolor(RED);
    outtextxy(280,240,"Compressing...");
    settextstyle(0,0,1);
    setcolor(BLUE);
    Luma2com(input,outputfile);
    fclose(input);
    fclose(outputfile);
}

```

/* รุทินหลักของการบีบข้อมูลด้วยวิธีการ Luma subsampling ชนิด 4 : 1 */

```

void Lumasub4()
{
do{
    drawblock3(150,120,500,350,"Luma subsampling 4:1 Compression");
    setcolor(BLUE);
    setfillstyle(1,LIGHTBLUE);
    outtextxy(170,165,"Input filename : ");
    bar(320,160,480,176);
    getstr1(42,11,18,inputname);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

extension = strstr(inputname, ".");
extension++;
chk = strcmp(extension, "gs");
if( chk != 0 )
{
    drawblock1(170,275,470,325);
    setcolor(RED);
    outtextxy(185,295, " Warning!..you must enter .gs file.");
    getch();
}
if( chk == 0 )
    chkopen = openinput(inputname);
else chkopen = -1;
} while( chk != 0 || chkopen != 0 );

do{
    outtextxy(170,198,"Output filename : ");
    bar(320,192,480,208);
    getstr1(42,13,18,outputname);
    extension = strstr(outputname, ".");
    extension++;
    chk = strcmp(extension, "lu4");
    if( chk != 0 )
    {
        drawblock1(170,275,470,325);
        setcolor(RED);
        outtextxy(185,295, " Warning!..you must enter .lu4 file");
        getch();
        setfillstyle(1,WHITE);
        bar(165,270,475,330);
        setfillstyle(1,LIGHTBLUE);
        setcolor(BLUE);
    }
    } while( chk != 0 );

    strcpy(luma4name,outputname);
    if( (outputfile = fopen(luma4name,"wb")) == NULL )
    {
        closegraph();
        printf("Error opening output file.");
        exit(1);
    }

    setcolor(BLUE);
    drawblock3(150,120,500,350,"Luma subsampling 4:1 Compression");
    setcolor(RED);
    outtextxy(280,240, "Compressing...");
    setcolor(BLUE);
    Luma4com(input,outputfile);
    fclose(input);
    fclose(outputfile);
}

/* รุ้ทันหลักของการบีบข้อมูลด้วยวิธีการ LZW */
void Lzw()
{
    do{
        drawblock3(150,120,500,350,"LZW Compression");
        setcolor(BLUE);
        setfillstyle(1,LIGHTBLUE);
        outtextxy(170,165,"Input filename : ");
        bar(320,160,480,176);
        getstr1(42,11,18,inputname);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    extension = strstr(inputname, ".");
    extension++;
    chk = strcmp(extension, "gs");
    if( chk != 0 )
    {
        drawblock1(170,275,470,325);
        setcolor(RED);
        outtextxy(185,295, " Warning!..you must enter .gs file.");
        getch();
    }
    if( chk == 0 )
        chkopen = openinput(inputname);
    else chkopen = -1;
} while( chk != 0 || chkopen != 0 );

do{
    outtextxy(170,198,"Output filename : ");
    bar(320,192,480,208);
    getstr1(42,13,18,outputname);
    extension = strstr(outputname, ".");
    extension++;
    chk = strcmp(extension, "lzw");
    if( chk != 0 )
    {
        drawblock1(170,275,470,325);
        setcolor(RED);
        outtextxy(185,295, " Warning!..you must enter .lzw file");
        getch();
        setfillstyle(1,WHITE);
        bar(165,270,475,330);
        setfillstyle(1,LIGHTBLUE);
        setcolor(BLUE);
    }
} while( chk != 0 );

strcpy(lzwname,outputname);
openoutput(lzwname);
setcolor(BLUE);
drawblock3(150,120,500,350,"LZW Compression");
setcolor(RED);
outtextxy(280,240,"Compressing...");
setcolor(BLUE);
CompressFile( input, output, 0, inputname );
CloseOutputBitFile( output );
fclose( input );
lzwfile = fopen(lzwname,"rb");
lzwsize = (long)filelength(fileno(lzwfile));
}

/* รูทีนหลักของการบีบข้อมูลด้วยวิธีการ JPEG */
void Jpeg()
{
    do{
        drawblock3(150,120,500,350,"JPEG Compression");
        setcolor(BLUE);
        setfillstyle(1,LIGHTBLUE);
        outtextxy(170,165,"Input filename : ");
        bar(320,160,480,176);
        getstr1(42,11,18,inputname);
        extension = strstr(inputname, ".");
        extension++;
        chk = strcmp(extension, "gs");
        if( chk != 0 )

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    drawblock1(170,275,470,325);
    setcolor(RED);
    outtextxy(185,295," Warning!..you must enter .gs file.");
    getch();
}
if( chk == 0 )
    chkopen = openinput(inputname);
else chkopen = -1;
} while( chk != 0 || chkopen != 0 );

do{
    outtextxy(170,198,"Output filename : ");
    bar(320,192,480,208);
    getstr1(42,13,18,outputname);
    extension = strstr(outputname,".");
    extension++;
    chk = strcmp(extension,"dct");
    if( chk != 0 )
    {
        drawblock1(170,275,470,325);
        setcolor(RED);
        outtextxy(185,295," Warning!..you must enter .dct file");
        getch();
        setfillstyle(1,WHITE);
        bar(165,270,475,330);
        setfillstyle(1,LIGHTBLUE);
        setcolor(BLUE);
    }
} while( chk != 0 );

strcpy(jpegname,outputname);
openoutput(jpegname);

outtextxy(170,231,"Quality Factor : ");
setcolor(RED);
outtextxy(180,260,"WARNING!...");
setcolor(MAGENTA);
outtextxy(180,290,"The quality factor must between 1-25");
outtextxy(180,320,"for exceptable error");
do
{
    bar(320,222,480,240);
    quality=3;
    getstr1(42,15,2,qstr);
    quality = atoi(qstr);
    if(strcmp(qstr,"") == 0) quality = 3;
    if(quality < 0 || quality > 25)
    {
        gotoxy(42,15);
        printf("Silly");
        delay(500);
    }
}while(quality < 0 || quality > 25);

setcolor(BLUE);
drawblock3(150,120,500,350,"JPEG Compression");
setcolor(RED);
outtextxy(280,240,"Compressing...");
setcolor(BLUE);
DctCompressFile( input, output, quality );
CloseOutputBitFile( output );
fclose( input );
jpegfile = fopen(jpegname,"rb");

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    jpegsize = (long)filelength(fileno(jpegfile));
}

/* ฟังก์ชันในการรับข้อความสำหรับเก็บในตัวแปร */

void getstr1(int x,int y,int length,char text[])
{
    char ch;
    int i,j,t;

    i = 0; ch = 0;
    t = x;

    if( length < 0 || length > 18 ) length = 18;
    length++;

    gotoxy(x,y);
    printf(" ");
    while( ch != 27 && ch != 13 && i < length)
    {
        ch = getch();
        if ( ch != 8 )
        {
            if ( i < length && i > -2 )
            {
                text[i] = ch;
                i++;
            }
            if ( ch == 27)
                while( i > 0) text[--i] = 0;
            gotoxy(x++,y);
            if ( i < length && ch != 13 ) printf("%c",ch);
            if ( i < length-1) printf("_");
        }
        else
        {
            if ( i < length && i > 0 )
            {
                i--;
                text[i] = 0;
                gotoxy(x--,y); printf(" ");
                gotoxy(x,y); printf("_");
            }
        }
    }

    if (ch == 27)
    {
        gotoxy(t,y);
        for(j=0;j<length;j++)
            printf(" ");
    }
    if( i != 0 && ch == 13 )
    {
        gotoxy(--x,y);
        printf(" ");
        text[--i] = 0;
    }

    if( i == length ) text[--i] = 0;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* รูทีนสำหรับแสดงภาพของไฟล์ข้อมูลภาพต้นฉบับ */
```

```
view_gs(com,inputname)
char inputname[20];
char com[60];
{
    strcpy(com,"gs ");
    strcat(com,inputname);
    closegraph();
    system(com);
}
```

```
/* รูทีนสำหรับแสดงภาพของไฟล์ข้อมูลภาพที่ถูกบีบด้วยวิธีการ
```

```
Luma subsampling */
```

```
view_luma(com,inputname)
char inputname[20];
char com[60];
{
    strcpy(com,"vll ");
    strcat(com,inputname);
    closegraph();
    system(com);
}
```

```
/* รูทีนสำหรับแสดงภาพของไฟล์ข้อมูลภาพที่ถูกบีบด้วยวิธีการ LZW */
```

```
view_lzw(com,inputname)
char inputname[20];
char com[60];
{
    strcpy(com,"mvlzw ");
    strcat(com,inputname);
    closegraph();
    system(com);
}
```

```
/* รูทีนสำหรับแสดงภาพของไฟล์ข้อมูลภาพที่ถูกบีบด้วยวิธีการ JPEG */
```

```
view_jpeg(com,inputname)
char inputname[20];
char com[60];
{
    strcpy(com,"mvdct ");
    strcat(com,inputname);
    closegraph();
    system(com);
}
```

```
/* รูทีนหลักสำหรับช่วยในการวิเคราะห์การบีบข้อมูลด้วยวิธีการต่าง ๆ */
```

```
analyze()
{
    int x,y;

    setviewport(0,0,getmaxx(),getmaxy(),1);
    drawblockly(50,50,getmaxx()-30,getmaxy()-50);
    setcolor(LIGHTMAGENTA);
    outtextxy(200,65,"WELCOME TO ANALYZING BY PROGRAM");
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    outtextxy(125,100,"This program is used to look for the\
        method,compression");
    outtextxy(100,125,"and expansion for your picture.");
    outtextxy(125,150,"Please choose the following method.");
    drawblocky(150,175,450,200);
    drawblocky(150,215,450,240);
    drawblocky(150,255,450,280);
    drawblocky(150,295,450,320);
    drawblocky(150,335,450,360);
    setcolor(BLUE);
    outtextxy(255,185,"LUMA 2:1");
    outtextxy(255,225,"LUMA 4:1");
    outtextxy(265,265,"LZW");
    outtextxy(200,305,"JPEG with quality factor");
    outtextxy(260,345,"EXIT");
    saveblock1(150,175,450,200,0);
    saveblock1(150,215,450,240,1);
    saveblock1(150,255,450,280,2);
    saveblock1(150,295,450,320,3);
    saveblock1(150,335,450,360,4);
    putimage(150,175,mem[0],4);
    setcolor(BLUE);
    setfillstyle(1,BLACK);
    outtextxy(255,185,"LUMA 2:1");
    selectanalyze();
}
/* รูทีนแสดงทางเลือกของส่วนการวิเคราะห์การบีบข้อมูล */

```

```

void selectanalyze()
{
    int i;
    int xs =175;
    anachoice = 0;
    i = 0;
    while( i != ESC && exitview != 1 )
    {
        i = bioskey(0);
        switch(i)
        {
            case UP : if (anachoice == 0)
            {
                putimage(150,175,mem[anachoice],0);
                anachoice = 4;
                putimage(150,335,mem[anachoice],4);
            }
            else
            {
                switch(anachoice)
                {
                    case 1 : xs = 215;break;
                    case 2 : xs = 255;break;
                    case 3 : xs = 295;break;
                    case 4 : xs = 335;break;
                }
                putimage(150,xs,mem[anachoice],0);
                anachoice = anachoice-1;
                xs = xs-40;
                putimage(150,xs,mem[anachoice],4);
            }
            break;
            case DOWN : if( anachoice == 4)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        {
            putimage(150,335,mem[anachoice],0);
            anachoice = 0;
            putimage(150,175,mem[anachoice],4);
        }
    else
    {
        switch(anachoice)
        {
            case 0 : xs = 175;break;
            case 1 : xs = 215;break;
            case 2 : xs = 255;break;
            case 3 : xs = 295;break;
        }
        putimage(150,xs,mem[anachoice],0);
        anachoice = anachoice+1;
        xs = xs+40;
        putimage(150,xs,mem[anachoice],4);
    }
    break;
case RETURN : switch(anachoice)
{
    case 0 :Lumasub2();
            LUMA2();
            if( !exitview )
            {
                drawmain();
                drawmainy();
            }
            putimage(150,175,mem[0],4);
            break;
    case 1 :Lumasub4();
            LUMA4();
            if( !exitview )
            {
                drawmain();
                drawmainy();
            }
            putimage(150,215,mem[1],4);
            break;
    case 2 :Lzw();
            LZW();
            if( !exitview )
            {
                drawmain();
                drawmainy();
            }
            putimage(150,255,mem[2],4);
            break;
    case 3 :Jpeg();
            JPEGq();
            if( !exitview )
            {
                drawmain();
                drawmainy();
            }
            putimage(150,295,mem[3],4);
            break;
    case 4 :i=ESC;
            break;
}
}
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        exitview = 0;
    }

/* รูทีนแสดงเมนูหลักของส่วนการวิเคราะห์ */

void drawmainy()
{
    int x,y;

    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(LIGHTBLUE);
    setcolor(LIGHTBLUE);
    drawblockly(50,50,getmaxx()-30,getmaxy()-50);
    setfillstyle(4,1);
    settextstyle(0,0,1);
    setcolor(LIGHTMAGENTA);
    outtextxy(200,65,"WELCOME TO ANALIZING BY PROGRAM");
    outtextxy(125,100,"This program is used to look for the\
        method,compression");
    outtextxy(100,125,"and expansion for your picture.");
    outtextxy(125,150,"Please choose the following method.");
    drawblocky(150,175,450,200);
    drawblocky(150,215,450,240);
    drawblocky(150,255,450,280);
    drawblocky(150,295,450,320);
    drawblocky(150,335,450,360);
    setcolor(BLUE);
    outtextxy(255,185,"LUMA 2:1");
    outtextxy(255,225,"LUMA 4:1");
    outtextxy(265,265,"LZW");
    outtextxy(200,305,"JPEG with quality factor");
    outtextxy(260,345,"EXIT");
}

/* รูทีนแสดงรูปบล็อกแบบที่ 1 สำหรับใช้ในส่วนการวิเคราะห์ */

void drawblockly(int xs,int ys,int xl,int yl)
{
    setviewport(0,0,getmaxx(),getmaxy(),1);
    setfillstyle(1,LIGHTCYAN);
    bar(xs,ys,xl,yl);
    setcolor(RED);
    rectangle(xs+2,ys+2,xl-1,yl-2);
    line(xs+3,ys+3,xl-3,ys+3);
    setcolor(RED);
    rectangle(xs+3,ys+3,xl-2,yl-3);
    line(xs+2,ys+2,xl-2,ys+2);
}

/* รูทีนแสดงรูปบล็อกสำหรับใช้ในส่วนการวิเคราะห์ */

void drawblocky(int xs,int ys,int xl,int yl)
{
    setviewport(0,0,getmaxx(),getmaxy(),1);
    setfillstyle(1,CYAN);
    bar(xs+5,ys+5,xl+5,yl+5);
    setfillstyle(1,LIGHTMAGENTA);
    bar(xs,ys,xl,yl);
}

```

```
/* รูทีนสำหรับเก็บบล็อกข้อมูลของเมนูหลักการวิเคราะห์ */
```

```
void saveblockmainy()  
{  
    saveblock1(150,175,450,200,0);  
    saveblock1(150,215,450,240,1);  
    saveblock1(150,255,450,280,2);  
    saveblock1(150,295,450,320,3);  
    saveblock1(150,335,450,360,4);  
    putimage(150,175,mem[0],4);  
}
```

```
/* รูทีนแสดงทางเลือกในส่วนเมนูหลักของการวิเคราะห์ */
```

```
void selectmainy()  
{  
    int i;  
    int xs = 175;  
    choice = 0;  
    do  
    {  
        i = bioskey(0);  
        switch(i)  
        {  
            case UP : if(choice == 0)  
            {  
                putimage(150,175,mem[choice],0);  
                choice = 4;  
                putimage(150,335,mem[choice],4);  
            }  
            else  
            {  
                switch(choice)  
                {  
                    case 1: xs = 215;break;  
                    case 2: xs = 255;break;  
                    case 3: xs = 295;break;  
                    case 4: xs = 335;break;  
                }  
                putimage(150,xs,mem[choice],0);  
                choice = choice-1;  
                xs = xs-40;  
                putimage(150,xs,mem[choice],4);  
            }  
            break;  
            case DOWN : if (choice == 4)  
            {  
                putimage(150,335,mem[choice],0);  
                choice = 0;  
                putimage(150,175,mem[choice],4);  
            }  
            else  
            {  
                switch(choice)  
                {  
                    case 0 :xs = 175;break;  
                    case 1 :xs = 215;break;  
                    case 2 :xs = 255;break;  
                    case 3 :xs = 295;break;  
                }  
                putimage(150,xs,mem[choice],0);  
                choice = choice+1;  
            }  
        }  
    }  
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        xs = xs+40;
        putimage(150,xs,mem[choice],4);
    }
    break;
case RETURN : switch(choice)
    {
        case 0: Lumasub2();
            break;
        case 1: Lumasub4();
            break;
        case 2: Lzw();
            break;
        case 3: Jpeg();
            break;
        case 4: EXIT();
            break;
    }
    break;
}
}while(i != ESC);
}

/* รูทีนแสดงผลการทำงานของบีบข้อมูลด้วยวิธีการ Luma subsampling 2 : 1 */
void LUMA2()
{
    int x,y;
    FILE *file;

    file = fopen(luma2name,"rb");
    luma2size = filelength(fileno(file));
    fclose(file);

    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(LIGHTBLUE);
    drawblockly(50,50,getmaxx()-30,getmaxy()-50);
    setcolor(BLUE);
    settextstyle(0,0,2);
    outtextxy(160,70,"Result of LUMA 2:1 method.");
    settextstyle(0,0,1);
    setfillstyle(1,LIGHTBLUE);
    bar(315,140,400,160);
    gotoxy(41,10);
    printf("64000");
    outtextxy(100,163," Compression Ratio = -----");
    bar(315,172,400,192);
    gotoxy(41,12);
    printf("%lu",luma2size);
    bar(315,205,400,225);
    outtextxy(100,210,"
                                     = ");
    gotoxy(41,14);
    printf("%5.2f",64000.0/luma2size);
    setcolor(BLUE);
    outtextxy(100,250," Percent compressed = 100 -");
    outtextxy(100,243,"
                                     1");
    outtextxy(100,250,"
                                     ( ----- ) * 100");
    outtextxy(100,260,"
                                     Compression ratio");
    outtextxy(100,290,"
                                     =");
    bar(315,285,400,305);
    gotoxy(41,19);
    printf("%5.2f",100-((1/(64000.0/luma2size))*100));
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    outtextxy(110,330,"You can view picture for error decision\
    by View in this.");
    view_type = 0;
    selectview();
}

/* รูทีนแสดงผลการบีบข้อมูลด้วยวิธีการ Luma subsampling 4 : 1 */

void LUMA4()
{
    int x,y;
    FILE *file;

    file = fopen(luma4name,"rb");
    luma4size = filelength(fileno(file));
    fclose(file);

    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(LIGHTBLUE);
    drawblockly(50,50,getmaxx()-30,getmaxy()-50);
    setcolor(BLUE);
    settextstyle(0,0,2);
    outtextxy(160,70,"Result of LUMA 4:1 method.");
    settextstyle(0,0,1);
    setfillstyle(1,LIGHTBLUE);
    bar(315,140,400,160);
    gotoxy(41,10);
    printf("64000");
    outtextxy(100,163,"    Compression Ratio = -----");
    bar(315,172,400,192);
    gotoxy(41,12);
    printf("%lu",luma4size);
    bar(315,205,400,225);
    outtextxy(100,210,"    = ");
    gotoxy(41,14);
    printf("%5.2f",64000.0/luma4size);
    setcolor(BLUE);
    outtextxy(100,250,"    Percent compressed = 100 -");
    outtextxy(100,243,"    1");
    outtextxy(100,250,"    ( ----- ) * 100");
    outtextxy(100,260,"    Compression ratio");
    outtextxy(100,290,"    =");
    bar(315,285,400,305);
    gotoxy(41,19);
    printf("%5.2f",100-((1/(64000.0/luma4size))*100));
    outtextxy(110,330,"You can view picture for error decision\
    by View in this.");
    view_type = 1;
    selectview();
}

```

/* รูทีนแสดงผลการบีบข้อมูลด้วยวิธีการ LZW */

```

void LZW()
{

```

```

    int x,y;
    FILE *file;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

file = fopen(lzwname,"rb");
lzwsz = filelength(fileno(file));
fclose(file);

setviewport(0,0,getmaxx(),getmaxy(),1);
setbkcolor(LIGHTBLUE);
drawblockly(50,50,getmaxx()-30,getmaxy()-50);
setcolor(BLUE);
settextstyle(0,0,2);
outtextxy(160,70,"Result of LZW method.");
settextstyle(0,0,1);
setfillstyle(1,LIGHTBLUE);
bar(315,140,400,160);
gotoxy(41,10);
printf("64000");
outtextxy(100,163," Compression Ratio = -----");
bar(315,172,400,192);
gotoxy(41,12);
printf("%lu",lzwsz);
bar(315,205,400,225);
outtextxy(100,210," = ");
gotoxy(41,14);
printf("%5.2f",64000.0/lzwsz);
setcolor(BLUE);
outtextxy(100,250," Percent compressed = 100 -");
outtextxy(100,243," 1");
outtextxy(100,250," ( ----- ) * 100");
outtextxy(100,260," Compression ratio");
outtextxy(100,290," =");
bar(315,285,400,305);
gotoxy(41,19);
printf("%5.2f",100-((1/(64000.0/(float)lzwsz))*100));
view_type = 2;
selectview();
}

/* รูทีนแสดงผลการบีบข้อมูลด้วยวิธีการ JPEG */

void JPEGq()
{
    int x,y;
    FILE *file;
    int i,j;
    int digit,dec,sign;
    int diff,c1,c2;
    char tmp[5];
    double error;
    BIT_FILE *input;
    FILE *original;
    FILE *output;

    file = fopen(jpegname,"rb");
    jpegsize = filelength(fileno(file));
    fclose(file);

    setviewport(0,0,getmaxx(),getmaxy(),1);
    setbkcolor(LIGHTBLUE);
    drawblockly(50,50,getmaxx()-30,getmaxy()-50);
    setcolor(BLUE);
    settextstyle(0,0,2);
    outtextxy(160,70,"Result of JPEG method.");
    settextstyle(0,0,1);
    setfillstyle(1,LIGHTBLUE);

```

```

bar(315,140,400,160);
gotoxy(41,10);
printf("64000");
outtextxy(100,163," Compression Ratio = -----");
bar(315,172,400,192);
gotoxy(41,12);
printf("%lu",jpegsize);
bar(315,205,400,225);
outtextxy(100,210,"                      = ");
gotoxy(41,14);
printf("%5.2f",64000.0/jpegsize);
setcolor(BLUE);
outtextxy(100,250," Percent compressed = 100 -");
outtextxy(100,243,"                      1");
outtextxy(100,250,"                      ( ----- ) * 100");
outtextxy(100,260,"                      Compression ratio");
outtextxy(100,290,"                      =");
bar(315,285,400,305);
gotoxy(41,19);
printf("%5.2f",100-(((float)jpegsize/64000.0)*100));
input = OpenInputBitFile(jpegname);
output = fopen("temp","wb");

DctExpandFile( input, output );
CloseInputBitFile( input );
fclose( output );

original = fopen(inputname,"rb");
output = fopen("temp","rb");

error = 0.0;
for ( j = 0 ; j < 200 ; j++ ) {
    for ( i = 0 ; i < 320 ; i++ ) {
        c1 =getc( original );
        c2 =getc( output );
        diff = c1 - c2;
        error += diff*diff;
        if ( diff < 0 )
            diff *= -1;
        if ( diff > 63 )
            diff = 63;
    }
}

fclose(original);
fclose(output);

error /= 320.0 * 200.0;
error = sqrt(error);
jpegratio = fcvt(error,4,&dec,&sign);
if( error < 10.0 )
{
    strncpy(tmp,jpegratio,1);
    tmp[1] = 0;
    strcat(tmp,".");
    strcat(tmp,++jpegratio);
    tmp[4] = 0;
}
else
{
    strncpy(tmp,jpegratio,2);
    tmp[2] = 0;
    strcat(tmp,".");
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    jpegratio+=2;
    strcat(tmp,jpegratio);
    tmp[5] = 0;
}

```

```

outtextxy(100,320,"Root Mean Square Error  = ");
outtextxy(320,320,tmp);
view_type = 3;
selectview();

```

```

}

```

/* รุ้ทันของการสอบถามความแน่ใจของผู้ใช้ ก่อนออกจากโปรแกรม */

```

void Exit()
{
    exitmain();
}

```

/* รุ้ทันแสดงทางเลือกในการแสดงภาพหลังการวิเคราะห์ */

```

void selectview()
{
    int i,k;

    k = 0;
    drawblocky(120,375,280,400);
    drawblocky(350,375,510,400);
    setcolor(BLUE);
    outtextxy(143,380,"View Picture");
    outtextxy(363,380,"Exit to Main Menu");
    saveblock1(120,375,280,400,5);
    saveblock1(350,375,510,400,6);
    putimage(120,375,mem[5],4);
    do
    {
        i=bioskey(0);
        switch(i)
        {
            case LEFT : if(k==0)
                        {
                            k =1;
                            putimage(350,375,mem[6],4);
                            putimage(120,375,mem[5],0);
                        }
                        else
                        {
                            k=0;
                            putimage(120,375,mem[5],4);
                            putimage(350,375,mem[6],0);
                        }break;
            case RIGHT : if(k ==1)
                        {
                            k=0;
                            putimage(120,375,mem[5],4);
                            putimage(350,375,mem[6],0);
                        }
                        else
                        {
                            k=1;
                            putimage(350,375,mem[6],4);
                            putimage(120,375,mem[5],0);
                        }break;
        }
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        case RETURN : switch(k)
        {
            case 0:view();
                i = ESC;
                break;
            case 1:exitview = 1;
                i = ESC;
                break;
        }
    }while(i != ESC);
}

/* รوتينสำหรับตัดสินใจเลือกวิธีการแสดงภาพตามชนิดของไฟล์ข้อมูลภาพ */

void view()
{
    switch(view_type)
    {
        case 0 : view_luma(com,luma2name);
            break;
        case 1 : view_luma(com,luma4name);
            break;
        case 2 : view_lzw(com,lzwname);
            break;
        case 3 : view_jpeg(com,jpegname);
    }
    initialize_graphics_mode();
}

/* รوتينของการสอบถามความแน่ใจของผู้ใช้ ก่อนออกจากโปรแกรม */

void EXIT()
{
    exitmain();
}

/* รوتينในการตรวจสอบการเปิดไฟล์ข้อมูลอินพุต */

int openinput(char *inputname)
{
    input = fopen(inputname,"rb");
    if (input ==NULL)
    {
        drawblock1(200,275,450,325);
        setcolor(RED);
        outtextxy(215,295,"Error opening for input file");
        getch();
        setfillstyle(1,WHITE);
        bar(195,270,455,330);
        setcolor(BLUE);
        setfillstyle(1,LIGHTBLUE);
        return(-1);
    }
    return(0);
}

```

```
/* รุ้ทึนใ้การตรวจสอบการเปิดไฟล์ข้อมูลผลลัพธ์ */
```

```
void openOutput(char *outputname)
{
    output = OpenOutputBitFile(outputname);
    if( output== NULL)
    {
        drawblock1(200,275,450,325);
        setcolor(RED);
        outtextxy(215,295,"Error opening output file");
        getch();
        setfillstyle(1,WHITE);
        bar(195,270,455,330);
        setcolor(BLUE);
    }
}
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาคผนวก ข

โปรแกรมการถอดข้อมูลภาพออกจากไฟล์ .PCX

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

#include "vesa.c"
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>

unsigned char picture[3][3000];
int          _24bit;
char         imagename[50];
char         dataname[50];
char         extension[10];
char         length;
int picx,picy;

typedef struct {
    char manufacturer;
    char version;
    char encoding;
    char bits_per_pixel;
    int  xmin,ymin;
    int  xmax,ymax;
    int  hres;
    int  vres;
    char __palette[48];
    char reserved;
    char color_planes;
        int  bytes_per_line;
        int  palette_type;
        char filler[58];
    } PCXHEAD;
PCXHEAD header;

int chartohex(char ch);
void bigrealdisplay(void);

/*****/

int main(int argc,char *argv[])
{
    int b,mode = 0;

    clrscr();

    if( argc < 3 )
    {
        printf("Usage pcx24i <mode> <inputfile.pcx> <outputfile.dat>\n");
        printf("256 colors pcx file format mode = 101,103,105\n");
        printf("16m colors pcx file format mode = 111,112,113");
        exit(1);
    }

    for(b=0;argv[1][b]!=0;b++)
        mode=(mode<<4)+chartohex(argv[1][b]);

    strcpy(imagename,argv[2]);
    strcpy(extension,strchr(imagename,46));

    if(strlen(imagename) < 5)
    {
        printf("Filename mustn't less than 5 characters.");
        exit(1);
    }
    else if(strcmp(extension,".pcx") != 0 )
    {
        printf("Filename must have extension ( *.pcx )");

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        exit(1);
    }

    if( strcmp(argv[3],"") == 0 )
    {
        length = strlen(argv[2]);
        strncpy(dataname,argv[2],length-4);
        strcat(dataname,".dat");
    }
    else
    {
        strcpy(dataname,argv[3]);
    }
    /* open VESA mode */
    if(!opengraph(mode))
    {
        printf("Error while opening VESA mode\n");
        exit(1);
    }

    if(getbitperpix() < 8)
    {
        closegraph();
        printf("This program runs on 256 colors or upper\n");
        exit(1);
    }

    /* display picture */
    bigrealdisplay();
    while(bioskey(1)!=0) getch();
    getch();
    closegraph();
    return 0;
}

void bigrealdisplay(void)
{
    FILE *imagefile,*imagedata;
    unsigned int width,depth;
    unsigned int bytes,i,j;
    int n,c,ij;
    char _palette[768];
    int x,xoff,yoff;

    if ((imagedata = fopen(dataname,"wb")) == NULL)
    {
        closegraph();
        printf("Error opening file %s",dataname);
        exit(1);
    }
    if((imagefile = fopen(imagefile,"rb"))!=NULL)
    {
        if(fread((char *)&header,1,sizeof(PCXHEAD),imagefile)
            == sizeof(PCXHEAD))
        {
            if(header.bits_per_pixel == 8 && header.color_planes == 3)
                _24bit = 1; else
            {
                if(!fseek(imagefile,-769L,SEEK_END))
                {
                    if(fgetc(imagefile) == 0x0c &&
                        fread(_palette,1,768,imagefile) == 768)
                    {
                        _24bit = 0;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        for(x= 0;x<768;x+=3)
            setdac((int)(x/3), (unsigned char)_palette[x]>>2,
                    (unsigned char)_palette[x+1]>>2,
                    (unsigned char)_palette[x+2]>>2);
        fseek(imagefile,128L,SEEK_SET);
    }
    else
        { picx = picy = depth = i = 0; }
}

bytes = header.bytes_per_line;
width = header.xmax-header.xmin+1;
depth = header.ymax-header.ymin+1;

if((_24bit)^(getbitperpix()!=8))
{
    closegraph();
    printf("Please use 8bit PCX file with 256 color \n");
    printf(" ..or 24bit PCX file with true color \n");
    exit(1);
}

picx = width;
picy = depth;

if(picx%2) picx++;

for (i=0;i<depth;i++)
{
    if(_24bit)
    {
        for (ij=2;ij>=0;ij--)
        {
            n = 0;
            do{
                c = fgetc(imagefile) &0xff;
                if((c & 0xc0) == 0xc0)
                {
                    j = c & 0x3f;
                    c = fgetc(imagefile);
                    while(j-->0)
                    {
                        picture[ij][n] = (unsigned char)c;
                        n++;
                    }
                }
            } while(n<bytes);
        }
    }
    else
    {
        picture[ij][n] = (unsigned char)c;
        n++;
    }
}

else
{
    n = 0;
    do{
        c = fgetc(imagefile) &0xff;
        if((c & 0xc0) == 0xc0)
        {
            j = c & 0x3f;
            c = fgetc(imagefile);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        while(j--)
        {
            picture[0][n] = (unsigned char)c;
            n++;
        }
    }
    else
    {
        picture[0][n] = (unsigned char)c;
        n++;
    }
}while(n<bytes);
}

xoff = (getxres()-picx)>>1;
yoff = (getyres()-picy)>>1;
if (xoff<0)xoff = 0;
if (yoff<0)yoff = 0;

/* store image data of 1 lines onto output file */
if( _24bit )
{
    for(ij=2;ij>=0;ij--)
        for(n=0;n<bytes;n++)
            fputc(picture[ij][n],imagedata);
}
else
{
    for(n=0; n < bytes; n++)
        fputc(picture[0][n],imagedata);
}
for(x = 0;x<(picx-1);x++)
{
    if(_24bit)
    {
        directpixel(x+xoff,i+yoff,(unsigned int)picture[2][x],
                    (unsigned int)picture[1][x],
                    (unsigned int)picture[0][x]);
    }
    else
    {
        putpixel(x+xoff,i+yoff,(unsigned int)picture[0][x]);
    }
}
}
}
else
{
    closegraph();
    printf("Error reading file %s",imagename);
    exit(1);
}

fclose(imagefile);
fclose(imagedata);

}
else
{
    closegraph();
    printf("Error opening file %s",imagename);
    exit(1);
}

printf("\007");

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

getch();

xoff = ((getxres()-picx)>>1)+50;
yoff = ((getyres()-picy)>>1)+50;
if (xoff<0)xoff = 0;
if (yoff<0)yoff = 0;

if((imagedata = fopen(dataname,"rb")) == NULL)
{
    closegraph();
    printf("Error while readding from output file\n");
    exit(1);
}

for(i=0; i < depth; i++)
{
    /* read image data of 1 lines for displaying onto the screen */

    if( _24bit )
    {
        for(ij=2;ij>=0;ij--)
            for(n=0;n<bytes;n++)
                picture[ij][n] = fgetc(imagedata);
    }
    else
    {
        for(n=0;n<bytes;n++)
            picture[0][n] = fgetc(imagedata);
    }

    /* display image data onto the screen 1 scan line */
    for(x = 0;x<(picx-1);x++)
    {
        if(_24bit)
        {
            directpixel(x+xoff,i+yoff,(unsigned int)picture[2][x],
                        (unsigned int)picture[1][x],
                        (unsigned int)picture[0][x]);
        }
        else
        {
            putpixel(x+xoff,i+yoff,(unsigned int)picture[0][x]);
        }
    }
    printf("\007");
}

int chartohex(char ch)
{
    switch(ch)
    {
        case '0':return 0;
        case '1':return 1;
        case '2':return 2;
        case '3':return 3;
        case '4':return 4;

        case '5':return 5;
        case '6':return 6;
        case '7':return 7;
        case '8':return 8;
        case '9':return 9;
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    case 'a':
    case 'A':return 0xa;
    case 'b':
    case 'B':return 0xb;
    case 'c':
    case 'C':return 0xc;
    case 'd':
    case 'D':return 0xd;
    case 'e':
    case 'E':return 0xe;
    case 'f':
    case 'F':return 0xf;
}
printf("Please use 0 to F to select Mode\n");
return 0;
}

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

/* โปรแกรมนี้ใช้ในการบีบและขยายขนาดข้อมูลแบบ Luma Subsampling

ชนิด 2:1 */

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
```

```
int row,size,pel,tmp;
unsigned char image[2][320],coded_image[160];
```

/* ส่วนการบีบข้อมูล */

```
Luma2com(FILE *fi,FILE *fo)
{
    for(row=0; row < 100; row++)
    {
        /* อ่านข้อมูล 320 คอลัมน์ */
        fread(&image[0][0],320,1,fi);
        fread(&image[1][0],320,1,fi);
        for(pel=0; pel < 160; pel++)
        {
            tmp = (int)image[0][2*pel];
            tmp += (int)image[0][2*pel+1];
            tmp += (int)image[1][2*pel];
            tmp += (int)image[1][2*pel+1];
            coded_image[pel] = (unsigned char) (tmp/4);
        }
        /* เก็บข้อมูล 160 คอลัมน์ */
        fwrite(coded_image,160,1,fo);
    }
}
```

/* ส่วนการขยายขนาดข้อมูล */

```
Luma2decom(FILE *fi,FILE *fo)
{
    for(row=0; row < 100; row++)
    {
        /* อ่านข้อมูล 160 คอลัมน์ */
        fread(coded_image,160,1,fi);
        for(pel=0; pel < 320; pel+= 2)
        {
            /* ข้อมูลภาพ 1 จุดของอินพุตไฟล์ถูกกำหนดให้กับข้อมูลภาพ
            4 จุดของเอาท์พุตไฟล์ */
            image[0][pel] = coded_image[pel/2];
            image[0][pel+1] = coded_image[pel/2];
            image[1][pel] = coded_image[pel/2];
            image[1][pel+1] = coded_image[pel/2];
        }
        /* เขียนข้อมูล 320 คอลัมน์ */
        fwrite(&image[0][0],320,1,fo);
        fwrite(&image[1][0],320,1,fo);
    }
}
```

/* โปรแกรมนี้ใช้ในการบีบและขยายขนาดข้อมูลแบบ LUMASUBSAMPLING

ชนิด 4:1 */

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
```

```
int row,size,pel,tmp;
unsigned char data[4][320],coded_data[80];
```

/* ส่วนของการบีบข้อมูล */

```
Luma4com(FILE *fi,FILE *fo)
{
```

```
    for(row=0; row < 50; row++)
```

```
    {
```

```
        /* อ่านข้อมูล 320 คอลัมน์ */
```

```
        fread(&data[0][0],320,1,fi);
```

```
        fread(&data[1][0],320,1,fi);
```

```
        fread(&data[2][0],320,1,fi);
```

```
        fread(&data[3][0],320,1,fi);
```

```
        for(pel=0; pel < 80; pel++)
```

```
        {
```

```
        /* หาค่าเฉลี่ยของข้อมูลภาพ 16 จุด ที่อยู่ติดกัน */
```

```
        tmp = (int)data[0][4*pel];
```

```
        tmp += (int)data[0][4*pel+1];
```

```
        tmp += (int)data[0][4*pel+2];
```

```
        tmp += (int)data[0][4*pel+3];
```

```
        tmp += (int)data[1][4*pel];
```

```
        tmp += (int)data[1][4*pel+1];
```

```
        tmp += (int)data[1][4*pel+2];
```

```
        tmp += (int)data[1][4*pel+3];
```

```
        tmp += (int)data[2][4*pel];
```

```
        tmp += (int)data[2][4*pel+1];
```

```
        tmp += (int)data[2][4*pel+2];
```

```
        tmp += (int)data[2][4*pel+3];
```

```
        tmp += (int)data[3][4*pel];
```

```
        tmp += (int)data[3][4*pel+1];
```

```
        tmp += (int)data[3][4*pel+2];
```

```
        tmp += (int)data[3][4*pel+3];
```

```
        coded_data[pel] = (unsigned char)(tmp/16);
```

```
    }
```

```
/* เขียนข้อมูล 80 คอลัมน์ */
```

```
    fwrite(coded_data,80,1,fo);
```

```
    }
```

```
}
```

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

/* ส่วนของการขยายขนาดข้อมูล */

```
Luma4decom(FILE *fi,FILE *fo)
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

for(row=0; row < 50; row++)
{
/* อ่านข้อมูล 160 คอลัมน์ */
fread(coded_data,80,1,fi);
for(pel=0; pel < 320; pel+= 4)
{
/* ข้อมูลภาพ 1 จุดของอินพุตไฟล์ถูกกำหนดให้กับข้อมูลภาพ
16 จุดของเอาต์พุตไฟล์ */
data[0][pel] = coded_data[pel/4];
data[0][pel+1] = coded_data[pel/4];
data[0][pel+2] = coded_data[pel/4];
data[0][pel+3] = coded_data[pel/4];
data[1][pel] = coded_data[pel/4];
data[1][pel+1] = coded_data[pel/4];
data[1][pel+2] = coded_data[pel/4];
data[1][pel+3] = coded_data[pel/4];
data[2][pel] = coded_data[pel/4];
data[2][pel+1] = coded_data[pel/4];
data[2][pel+2] = coded_data[pel/4];
data[2][pel+3] = coded_data[pel/4];
data[3][pel] = coded_data[pel/4];
data[3][pel+1] = coded_data[pel/4];
data[3][pel+2] = coded_data[pel/4];
data[3][pel+3] = coded_data[pel/4];
}
/* เขียนข้อมูล 320 คอลัมน์ */
fwrite(&data[0][0],320,1,fo);
fwrite(&data[1][0],320,1,fo);
fwrite(&data[2][0],320,1,fo);
fwrite(&data[3][0],320,1,fo);
}
}

```

```

/* โปรแกรมหลักของการบีบข้อมูลด้วยวิธี CCIR-601 แบบ 4:2:2 */

#include<stdio.h>
#include<conio.h>
#include<dos.h>

/* แมคโครในการปรับเศษ */

#define ROUND( a )(( (a)< 0 ) ? (int)((a) - 0.5):\
                    (int)((a) + 0.5))

int    y_map[4][640],cr_map[4][640],cb_map[4][640];
int    red[4][640],green[4][640],blue[4][640];
int    cr[2][640],cb[2][640];
int    r,g,b,n;
float  y_val,rf,gf,bf;

FILE    *imagedata;
FILE    *chromadata;
FILE    *output;

char    imagename[20];
char    chromaname[20];

/* โปรแกรมหลัก */

main(int argc,char *argv[])
{
    clrscr();

    /* แนะนำการเรียกใช้โปรแกรม */

    if( argc < 4 )
    {
        printf("Usage %s <inputfile> <outputfile> \
/c or /u",argv[0]);
        exit(1);
    }

    if( argv[3][1] != 'c' && argv[3][1] != 'u' )
    {
        printf("Please specify /c for compress \
or /u for uncompress.\n");
        exit(1);
    }

    if( argv[3][1] == 'c' )
    {
        if( (imagedata = fopen(argv[1],"rb")) == NULL )
        {
            printf("Error reading input file.");
            exit(1);
        }

        if( (output = fopen(argv[2],"wb")) == NULL )
        {
            printf("Error opening output file.");
            exit(1);
        }
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        M422();
    }

    if( argv[3][1] == 'u' )
    {
        if( ( chromadata = fopen(argv[1],"rb")) == NULL )
        {
            printf("Error reading input file.");
            exit(1);
        }

        if( (output = fopen(argv[2],"wb")) == NULL )
        {
            printf("Error opening output file.");
            exit(1);
        }
        M422U();
    }
}

```

/* ภูินหลักของการบีบข้อมูลแบบ 4:2:2 */

```

M422()
{
    int i,j,k;

    clrscr();
    printf("Compressing...\n\n");
    textcolor(1);
    for(i=0;i<40;i++)
        cputs("0");
    gotoxy(1,3);
    textcolor(3);
    n = 0;

    for(k=0;k<120;k++)
    {
        for(i=0;i<4;i++)
        {
            /* อ่านข้อมูล 4 แถวของแต่ละองค์ประกอบ */

            for(j=0;j<640;j++)
                red[i][j] = (int)fgetc(imagedata);
            for(j=0;j<640;j++)
                green[i][j] = (int)fgetc(imagedata);
            for(j=0;j<640;j++)
                blue[i][j] = (int)fgetc(imagedata);
        }

        /* แปลงข้อมูลจากรูปแบบ RGB ไปเป็น YCrCb */

        for(i=0;i<4;i++)
            for(j=0;j<640;j++)
            {
                rf = (float)red[i][j];
                gf = (float)green[i][j];
                bf = (float)blue[i][j];
                y_val = 0.299*rf + 0.587*gf + 0.114*bf;
                y_map[i][j] = (int)ROUND(y_val);
                cr_map[i][j] = (int)ROUND(rf-y_val);
            }
        }
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        cb_map[i][j] = (int)ROUND(bf-y_val);
    }
    CCIR_422();
}
gotoxy(1,3);
printf("\n\nSucessful !");
fclose(imagedata);
fclose(output);
}

/* รูทีนการบีบข้อมูลแบบ 4:2:2 */

int CCIR_422()
{
    int i,j;

    for(i=0;i<2;i++)
        for(j=0;j<640;j++)
        {
            cr[i][j] = ROUND((cr_map[i*2][j] + cr_map[i*2+1][j])/2);
            cb[i][j] = ROUND((cb_map[i*2][j] + cb_map[i*2+1][j])/2);
        }

    for(i=0;i<4;i++)
        for(j=0;j<640;j++)
        {
            fputc(y_map[i][j],output);
        }

    for(i=0;i<2;i++)
        for(j=0;j<640;j++)
        {
            fputc(cr[i][j],output);
        }

    for(i=0;i<2;i++)
        for(j=0;j<640;j++)
        {
            fputc(cb[i][j],output);
        }

    gotoxy((n/3)+1,3);
    cputs("0");
    gotoxy(20,1);
    printf("processed %4d%%", (int)((n/3)+1)*2.5);
    n++;
}

```

/* รูทีนหลักของการขยายข้อมูลแบบ 4:2:2 */

```

M422U()
{
    int i,j,k;

    clrscr();
    printf("Uncompressing...\n\n");
    textcolor(1);
    for(i=0;i<40;i++)
        cputs("0");
    gotoxy(1,3);
    textcolor(3);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

n = 0;

for(k=0;k<120;k++)
{
    for(i=0;i<4;i++)
    {
        CCIR_422U();

        for(i=0;i<4;i++)
        {

/* เก็บข้อมูลของแต่ละองค์ประกอบลงไฟล์ */

            for(j=0;j<640;j++)
            {
                r = saturate(red[i][j]);
                fputc(r,output);
            }
            for(j=0;j<640;j++)
            {
                g = saturate(green[i][j]);
                fputc(g,output);
            }
            for(j=0;j<640;j++)
            {
                b = saturate(blue[i][j]);
                fputc(b,output);
            }
        }
    }
    gotoxy(1,3);
    printf("\n\nSucessful !");
    fclose(chromadata);
    fclose(output);
}

/* routineของการขยายข้อมูลแบบ 4:2:2 */

int CCIR_422U()
{
    int i,j;

    for(i=0;i<4;i++)
        for(j=0;j<640;j++)
            y_map[i][j] = (int)fgetc(chromadata);

    for(i=0;i<2;i++)
        for(j=0;j<640;j++)
            cr[i][j] = (char)fgetc(chromadata);

    for(i=0;i<2;i++)
        for(j=0;j<640;j++)
            cb[i][j] = (char)fgetc(chromadata);

    for(i=0;i<4;i++)
        for(j=0;j<640;j++)
        {
            red[i][j] = cr[i/2][j] + y_map[i][j];
            blue[i][j] = cb[i/2][j] + y_map[i][j];
            green[i][j] = (int)ROUND(((float)y_map[i][j]*1.703
                - (float)red[i][j]*0.5093
                - (float)blue[i][j]*0.1942));
        }
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

gotoxy((n/3)+1,3);
cputs("0");
gotoxy(20,1);
printf("processed %4d%%", (int)((n/3)+1)*2.5);
n++;
}

```

/* ฟังก์ชันสำหรับการปรับค่าให้อยู่ในช่วง 0 - 255 */

```

int saturate(int a)
{
    if(a > 255) a = 255;
    if(a < 0 ) a = 0;
    return (a);
}

```



```
/* โปรแกรมหลักของการบีบข้อมูลด้วยวิธี CCIR-601 แบบ 4:1:1 */
```

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
```

```
/* แมคโครในการบีบพิเศษ */
```

```
#define ROUND( a )(((a) < 0 ) ? (int)((a) - 0.5):\
                    (int)((a) + 0.5))
```

```
int    y_map[4][640],cr_map[4][640],cb_map[4][640];
int    red[4][640],green[4][640],blue[4][640];
int    cr[1][640],cb[1][640];
int    r,g,b,n;
float  y_val,rf,gf,bf;
```

```
FILE   *imagedata;
FILE   *chromadata;
FILE   *output;
```

```
char    imagename[20];
char    chromaname[20];
```

```
/* โปรแกรมหลัก */
```

```
main(int argc,char *argv[])
{
```

```
    clrscr();
```

```
/* แนะนำการเรียกใช้โปรแกรม */
```

```
if( argc < 4 )
{
    printf("Usage %s <inputfile> <outputfile> \
          /c or /u",argv[0]);
    exit(1);
}
```

```
if( argv[3][1] != 'c' && argv[3][1] != 'u')
{
    printf("Please specify /c for compress \
          or /u for uncompress.\n");
    exit(1);
}
```

```
if( argv[3][1] == 'c' )
{
```

```
    if( (imagedata = fopen(argv[1],"rb")) == NULL )
    {
        printf("Error reading input file.");
        exit(1);
    }
```

```
    if( (output = fopen(argv[2],"wb")) == NULL )
    {
        printf("Error opening output file.");
        exit(1);
    }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        M411();
    }

    if( argv[3][1] == 'u' )
    {

        if( ( chromadata = fopen(argv[1],"rb")) == NULL )
        {
            printf("Error reading input file.");
            exit(1);
        }

        if( (output = fopen(argv[2],"wb")) == NULL )
        {
            printf("Error opening output file.");
            exit(1);
        }
        M411U();
    }
}

```

/* รูทีนหลักของการบีบข้อมูลแบบ 4:1:1 */

```

M411()
{
    int i,j,k;

    clrscr();
    printf("Compressing...\n\n");
    textcolor(1);
    for(i=0;i<40;i++)
        cputs("0");
    gotoxy(1,3);
    textcolor(3);
    n = 0;

    for(k=0;k<120;k++)
    {
        for(i=0;i<4;i++)
        {
            /* อ่านข้อมูล 4 แถวของแต่ละองค์ประกอบ */

            for(j=0;j<640;j++)
                red[i][j] = (int)fgetc(imagedata);
            for(j=0;j<640;j++)
                green[i][j] = (int)fgetc(imagedata);
            for(j=0;j<640;j++)
                blue[i][j] = (int)fgetc(imagedata);
        }
    }
}

```

/* แปลงข้อมูลจากรูปแบบ RGB ไปเป็น YCrCb */

```

for(i=0;i<4;i++)
    for(j=0;j<640;j++)
    {
        rf = (float)red[i][j];
        gf = (float)green[i][j];
        bf = (float)blue[i][j];
        y_val = 0.299*rf + 0.587*gf + 0.114*bf;
        y_map[i][j] = (int)ROUND(y_val);
        cr_map[i][j] = (int)ROUND((rf-y_val));
    }
}

```

เอกสารนี้เป็นเอกสารของบริษัทเอกชน ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        cb_map[i][j] = (int)ROUND(bf-y_val);
    }
    CCIR_411();
}

gotoxy(1,3);
printf("\n\nSucessful !");
fclose(imagedata);
fclose(output);
}

/* รุ้ที่ในการบีบข้อมูลแบบ 4:1:1 */

int CCIR_411()
{
    int i,j;

    for(i=0;i<1;i++)
        for(j=0;j<640;j++)
        {
            cr[i][j] = (cr_map[i*4][j] + cr_map[i*4+1][j]+
                        cr_map[i*4+2][j] + cr_map[i*4+3][j])/4;
            cb[i][j] = (cb_map[i*4][j] + cb_map[i*4+1][j]+
                        cb_map[i*4+2][j] + cb_map[i*4+3][j])/4;
        }

    for(i=0;i<4;i++)
        for(j=0;j<640;j++)
            fputc(y_map[i][j],output);

    for(i=0;i<1;i++)
        for(j=0;j<640;j++)
            fputc(cr[i][j],output);

    for(i=0;i<1;i++)
        for(j=0;j<640;j++)
            fputc(cb[i][j],output);

    gotoxy((n/3)+1,3);
    cputs("0");
    gotoxy(20,1);
    printf("processed %4d%%", (int)((n/3)+1)*2.5);
    n++;
}

```

/* รุ้ที่หลักของการขยายข้อมูลแบบ 4:1:1 */

```

M411U()
{
    int i,j,k;

    clrscr();
    printf("Uncompressing...\n\n");
    textcolor(1);
    for(i=0;i<40;i++)
        cputs("0");
    gotoxy(1,3);
    textcolor(3);
    n = 0;
}

```

```

for(k=0;k<120;k++)
{
    for(i=0;i<4;i++)
    {
        CCIR_411U();

        for(i=0;i<4;i++)
        {
            /* เก็บข้อมูลของแต่ละองค์ประกอบลงไฟล์ */

```

```

                for(j=0;j<640;j++)
                {
                    r = saturate(red[i][j]);
                    fputc(r,output);
                }

```

```

            for(j=0;j<640;j++)
            {
                g = saturate(green[i][j]);
                fputc(g,output);
            }

```

```

            for(j=0;j<640;j++)
            {
                b = saturate(blue[i][j]);
                fputc(b,output);
            }
        }
    }
}

```

```

gotoxy(1,3);
printf("\n\nSucessful !");
fclose(chromadata);
fclose(output);
}

```

/* routineของการขยายข้อมูลแบบ 4:1:1 */

```

int CCIR_411U()
{
    int i,j;

    for(i=0;i<4;i++)
        for(j=0;j<640;j++)
            y_map[i][j] = (int)fgetc(chromadata);

    for(i=0;i<1;i++)
        for(j=0;j<640;j++)
            cr[i][j] = (char)fgetc(chromadata);

    for(i=0;i<1;i++)
        for(j=0;j<640;j++)
            cb[i][j] = (char)fgetc(chromadata);

    for(i=0;i<4;i++)
        for(j=0;j<640;j++)
        {
            red[i][j] = cr[i/4][j] + y_map[i][j];
            blue[i][j] = cb[i/4][j] + y_map[i][j];
        }
}

```

เอกสารนี้เป็นเอกสารลิขสิทธิ์สงวนไว้สำหรับใช้ภายในเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        green[i][j] = (int)ROUND(((float)y_map[i][j]*1.703
                                - (float)red[i][j]*0.5093
                                - (float)blue[i][j]*0.1942));
    }

    gotoxy((n/3)+1,3);
    cputs("0");
    gotoxy(20,1);
    printf("processed %4d%%", (int)(((n/3)+1)*2.5));
    n++;
}

```

/* ฟังก์ชันสำหรับการปรับค่าให้อยู่ในช่วง 0 - 255 */

```

int saturate(int a)
{
    if(a > 255) a = 255;
    if(a < 0 ) a = 0;
    return (a);
}

```





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

/*โปรแกรมนี้แสดงรวมฟังก์ชันต่าง ๆ ที่ใช้ในการบีบและขยายข้อมูลแบบ LZW12 */

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "errhand.h"
#include "bitio.h"
```

/* ค่าคงที่ต่าง ๆ ที่ใช้ในโปรแกรมโดยที่ตัวแปร BITS ใช้กำหนดค่าจำนวนบิต
ของรหัสและตัวแปร TABLE_SIZE กำหนดค่าขนาดของพจนานุกรม */

```
#define BITS 12
#define MAX_CODE ( ( 1 << BITS ) - 1 )
#define TABLE_SIZE 5021
#define END_OF_STREAM 256
#define FIRST_CODE 257
#define UNUSED -1
```

/* กำหนดโปรโตไทป์ */

```
#ifdef __STDC__
unsigned int find_child_node( int parent_code, int child_character );
unsigned int decode_string( unsigned int offset, unsigned int code );
#else
unsigned int find_child_node();
unsigned int decode_string();
#endif
```

```
char *CompressionName = "LZW 12 Bit Encoder";
char *Usage = "in-file out-file\n\n";
```

```
unsigned long count;
int x,y;
```

/* โครงสร้างของพจนานุกรมประกอบด้วยค่ารหัสข้อมูลซึ่งประกอบด้วยข้อมูลจริง ๆ อยู่
2 ส่วน คือ ค่ารหัสแม่และค่ารหัสตัวอักษร ซึ่งจะมีค่าน้อยกว่า 256 */

```
struct dictionary {
    int code_value;
    int parent_code;
    char character;
} dict[ TABLE_SIZE ];
```

```
char decode_stack[ TABLE_SIZE ];
```

/* การบีบข้อมูลจะอาศัยหลักการอ่านข้อมูลเข้ามาทีละไบต์และเปรียบเทียบค่าจาก
การรวมกันของรหัสปัจจุบันกับค่าตัวอักษรปัจจุบันว่ามีในพจนานุกรมหรือไม่ถ้า
ไม่มีก็ถูกเพิ่มเข้าไปในพจนานุกรมเป็นรหัสใหม่ การเก็บข้อมูลลงไฟล์ผลลัพธ์จะ
เก็บข้อมูลจำนวน 12 บิตแทนแต่ละรหัสในพจนานุกรม */

```
void CompressFile( input, output, argc, argv )
FILE *input;
BIT_FILE *output;
int argc;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

char *argv[];
{
    int next_code;
    int character;
    int string_code;
    unsigned int index;
    unsigned int i;

    count = 1L;
    next_code = FIRST_CODE;

    for ( i = 0 ; i < TABLE_SIZE ; i++ )
        dict[ i ].code_value = UNUSED;
    if ( ( string_code = getc( input ) ) == EOF )
        string_code = END_OF_STREAM;
    while ( ( character = getc( input ) ) != EOF ) {
        count++;
        index = find_child_node( string_code, character );
        if ( dict[ index ].code_value != -1 )
            string_code = dict[ index ].code_value;
        else {
            if ( next_code <= MAX_CODE ) {
                dict[ index ].code_value = next_code++;
                dict[ index ].parent_code = string_code;
                dict[ index ].character = (char) character;
            }
            OutputBits( output, (unsigned long) string_code, BITS );
            string_code = character;
        }
    }
    OutputBits( output, (unsigned long) string_code, BITS );
    OutputBits( output, (unsigned long) END_OF_STREAM, BITS );
    while ( argc-- > 0 )
        printf( "Unknown argument: %s\n", *argv++ );
}

```

/*การขยายข้อมูล เริ่มต้นด้วยการอ่านค่ารหัสเข้ามาและแปลงกับไปเป็นอาเรย์

ของตัวอักษรของรหัสนั้น ๆ และเก็บลงไฟล์ผลลัพธ์ */

```

void ExpandFile( input, output, argc, argv )
BIT_FILE *input;
FILE *output;
int argc;
char *argv[];
{
    unsigned int next_code;
    unsigned int new_code;
    unsigned int old_code;
    int character;
    unsigned int count;

    next_code = FIRST_CODE;
    old_code = (unsigned int) InputBits( input, BITS );
    if ( old_code == END_OF_STREAM )
        return;
    character = old_code;
    putc( old_code, output );

    while ( ( new_code = (unsigned int) InputBits( input, BITS ) )
        != END_OF_STREAM ) {

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

if ( new_code >= next_code ) {
    decode_stack[ 0 ] = (char) character;
    count = decode_string( 1, old_code );
}
else
    count = decode_string( 0, new_code );
character = decode_stack[ count - 1 ];
while ( count > 0 )
    putc( decode_stack[ --count ], output );

if ( next_code <= MAX_CODE ) {
    dict[ next_code ].parent_code = old_code;
    dict[ next_code ].character = (char) character;
    next_code++;
}
old_code = new_code;
}
while ( argc-- > 0 )
    printf( "Unknown argument: %s\n", *argv++ );
}

/* รุทีนนี้ใช้ในการหาค่าดัชนีในพจนานุกรมจากการรวมกันของค่ารหัสปัจจุบันกับค่า
ตัวอักษรปัจจุบัน โดยตารางค่าดัชนีจะถูกสร้างโดยการเอกคลูซีฟออร์ ของรหัส
ตัวหน้ากับตัวอักษรตัวหลัง */

unsigned int find_child_node( parent_code, child_character )
int parent_code;
int child_character;
{
    int index;
    int offset;

    index = ( child_character << ( BITS - 8 ) ) ^ parent_code;
    if ( index == 0 )
        offset = 1;
    else
        offset = TABLE_SIZE - index;
    for ( ; ; ) {
        if ( dict[ index ].code_value == UNUSED )
            return( index );
        if ( dict[ index ].parent_code == parent_code &&
            dict[ index ].character == (char) child_character )
            return( index );
        index -= offset;
        if ( index < 0 )
            index += TABLE_SIZE;
    }
}

/* รุทีนนี้ใช้ในการถอดรหัสข้อมูลจากพจนานุกรม และเก็บค่าอาเรย์ของตัวอักษรที่ได้จาก
การถอดรหัสไว้ในตัวแปรโครงสร้าง decode_stack และคำนวณค่าจำนวนตัวอักษร
ของรหัสนั้น */

```

```

unsigned int decode_string( count, code )
unsigned int count;
unsigned int code;
{

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

/* โมดูล DCT ใช้ร่วมกับโปรแกรมการบีบข้อมูล ซึ่งใช้หลักการของ Discrete Cosine Transform */

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "bitio.h"
#include "errhand.h"
```

/* กำหนดค่าขนาดของไฟล์รูปภาพ โดยตัวแปร ROWS เป็นค่าจำนวนแถวและ COLS เป็นค่า
จำนวนคอลัมน์และค่า N ใช้กำหนดขนาดของ DCT บล็อก */

```
#define ROWS          200
#define COLS          320

#define ROWS1         480
#define COLS1         640

#define ROWS2         240
#define COLS2         320

#define N              8
```

/* แมคโครในการปัดเศษของเลขจำนวนเต็ม */

```
#define ROUND( a ) ( ( (a) < 0 ) ? (int) ( (a) - 0.5 ) : \
                    (int) ( (a) + 0.5 ) )
```

/* กำหนดโปรโตไทป์ */

```
void Initialize(int quality);
void ReadPixelStrip(FILE *input,unsigned char strip[N][COLS]);
void ReadPixelStrip1(FILE *input,unsigned char strip1[N][COLS1]);
void ReadPixelStrip2(FILE *input,unsigned char strip1[N][COLS2]);
int InputCode(BIT_FILE *input);
void ReadDCTData(BIT_FILE *input,int input_data[N][N]);
void OutputCode(BIT_FILE *output_file,int code );
void WriteDCTData(BIT_FILE *output_file,int output_data[N][N]);
void WritePixelStrip(FILE *output,unsigned char strip[N][COLS]);
void WritePixelStrip1(FILE *output,unsigned char strip1[N][COLS1]);
void WritePixelStrip2(FILE *output,unsigned char strip1[N][COLS2]);
void ForwardDCT(unsigned char *input[ N ],int output[N][N]);
void InverseDCT(int input[N][N],unsigned char *output[N]);
void DctCompressFile(FILE *input,BIT_FILE *output,int quality);
void DctCompressFile1(FILE *input,BIT_FILE *output,int quality);
void DctCompressFile2(FILE *input,BIT_FILE *output,int quality);
void DctExpandFile(BIT_FILE *input,FILE *output);
void DctExpandFile1(BIT_FILE *input,FILE *output);
void DctExpandFile2(BIT_FILE *input,FILE *output);
```

/* ตัวแปรโกลบอลที่ใช้ได้ในทุกส่วนของโปรแกรม */

```
unsigned char PixelStrip[N][COLS];
unsigned char PixelStrip1[N][COLS1];
unsigned char PixelStrip2[N][COLS2];
double C[N][N];
double Ct[N][N];
int InputRunLength;
int OutputRunLength;
int Quantum[N][N];
int quality;
```

เอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

struct zigzag {
    int row;
    int col;
} ZigZag[ N * N ] =
{
    {0, 0},
    {0, 1}, {1, 0},
    {2, 0}, {1, 1}, {0, 2},
    {0, 3}, {1, 2}, {2, 1}, {3, 0},
    {4, 0}, {3, 1}, {2, 2}, {1, 3}, {0, 4},
    {0, 5}, {1, 4}, {2, 3}, {3, 2}, {4, 1}, {5, 0},
    {6, 0}, {5, 1}, {4, 2}, {3, 3}, {2, 4}, {1, 5}, {0, 6},
    {0, 7}, {1, 6}, {2, 5}, {3, 4}, {4, 3}, {5, 2}, {6, 1}, {7, 0},
    {7, 1}, {6, 2}, {5, 3}, {4, 4}, {3, 5}, {2, 6}, {1, 7},
    {2, 7}, {3, 6}, {4, 5}, {5, 4}, {6, 3}, {7, 2},
    {7, 3}, {6, 4}, {5, 5}, {4, 6}, {3, 7},
    {4, 7}, {5, 6}, {6, 5}, {7, 4},
    {7, 5}, {6, 6}, {5, 7},
    {6, 7}, {7, 6},
    {7, 7}
};

```

/* โคไซน์ทรานฟอร์มเมตริกซ์และค่าทรานโพสเมตริกซ์ซึ่งทั้งสองเมตริกซ์จะใช้ขณะที่มีการคำนวณในกระบวนการ FDCT และการทำอินเวอร์ต กำหนดค่าควอนไทซ์เมตริกซ์ซึ่งขึ้นอยู่กับค่าควอลิตี้ด้วยและค่าตัวแปรที่ใช้ในเข้ารหัสแบบรันเลนจ์เอ็นโคดดิ้ง ก็ถูกกำหนดให้มีค่าเป็นศูนย์ด้วย */

```

void Initialize( quality )
int quality;
{
    int i;
    int j;
    double pi = atan( 1.0 ) * 4.0;

    for ( i = 0 ; i < N ; i++ )
        for ( j = 0 ; j < N ; j++ )
            Quantum[ i ][ j ] = 1 + ( ( 1 + i + j ) * quality );
    OutputRunLength = 0;
    InputRunLength = 0;
    for ( j = 0 ; j < N ; j++ ) {
        C[ 0 ][ j ] = 1.0 / sqrt( (double) N );
        Ct[ j ][ 0 ] = C[ 0 ][ j ];
    }
    for ( i = 1 ; i < N ; i++ ) {
        for ( j = 0 ; j < N ; j++ ) {
            C[ i ][ j ] = sqrt( 2.0 / N ) *
                cos( pi * ( 2 * j + 1 ) * i / ( 2.0 * N ) );
            Ct[ j ][ i ] = C[ i ][ j ];
        }
    }
}

```

/* รูทีนนี้ถูกใช้เมื่อเริ่มกระบวนการบีบข้อมูล โดยทำการอ่านข้อมูล 1 แถบ นั่นคือ ค่าจำนวน 8 แถว และ 320 คอลัมน์ ของไฟล์รูปภาพ จากนั้นทำการแบ่งข้อมูลออกเป็นบล็อก ๆ สำหรับทำการกระบวนการ FDCT */

```

void ReadPixelStrip( input, strip )
FILE *input;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

unsigned char strip[ N ][ COLS ];
{
    int row;
    int col;
    int c;

    for ( row = 0 ; row < N ; row++ )
        for ( col = 0 ; col < COLS ; col++ ) {
            c = getc( input );
            if ( c == EOF )
                fatal_error( "Error reading input grey scale file" );
            strip[ row ][ col ] = (unsigned char) c;
        }
}

```

/* รูทีนนี้ถูกใช้เมื่อเริ่มกระบวนการบีบข้อมูล โดยทำการอ่านข้อมูล 1 แถบ นั่นคือ ค่าจำนวน 8 แถว และ 640 คอลัมน์ ของไฟล์รูปภาพ จากนั้นทำการแบ่งข้อมูลออกเป็นบล็อก ๆ สำหรับทำการกระบวนการ FDCT */

```

void ReadPixelStrip1( input, strip1 )
FILE *input;
unsigned char strip1[ N ][ COLS1 ];
{
    int row;
    int col;
    int c;

    for ( row = 0 ; row < N ; row++ )
        for ( col = 0 ; col < COLS1; col++ ) {
            c = getc( input );
            if ( c == EOF )
                fatal_error( "Error reading input grey scale file" );
            strip1[ row ][ col ] = (unsigned char) c;
        }
}

```

```

void ReadPixelStrip2( input, strip2 )
FILE *input;
unsigned char strip2[ N ][ COLS2 ];
{
    int row;
    int col;
    int c;

    for ( row = 0 ; row < N ; row++ )
        for ( col = 0 ; col < COLS2; col++ ) {
            c = getc( input );
            if ( c == EOF )
                fatal_error( "Error reading input grey scale file" );
            strip2[ row ][ col ] = (unsigned char) c;
        }
}

```

/* รูทีนนี้ใช้สำหรับอ่านข้อมูลที่บีบอัดเป็นรหัส DCT จากไฟล์ที่ถูกบีบข้อมูลแล้ว โดยที่ค่ารหัสประกอบด้วย 2 ส่วน คือ ค่าจำนวนบิตของรหัสและ ค่าที่ถูกเข้ารหัสแล้ว โดยค่าจำนวนบิตนี้ถูกกำหนดด้วยค่าเลขฐานสองดังนี้ */

*	จำนวนบิต	รหัสเลขฐานสอง
*		
*	0	00
*	1	010
*	2	011
*	3	1000
*	4	1001
*	5	1010
*	6	1011
*	7	1100
*	8	1101
*	9	1110
*	10	1111

- * ค่าจำนวนบิตของค่าศูนย์จะถูกตามด้วยค่าขนาด 4 บิต เพื่อบอกให้ทราบถึงจำนวนของค่าศูนย์
- * ที่ต่อเนื่องกัน ส่วนค่า 1 - 10 นั้นจะบอกให้ทราบว่าค่าอื่น ๆ ตามมา ซึ่งค่าของการเข้ารหัสจะมี
- * ลักษณะเป็นดังข้างล่าง

*	จำนวนบิต	ขนาด
*		
*	1	-1, 1
*	2	-3 to -2, 2 to 3
*	3	-7 to -4, 4 to 7
*	4	-15 to -8, 8 to 15
*	5	-31 to -16, 16 to 31
*	6	-63 to -32, 32 to 64
*	7	-127 to -64, 64 to 127
*	8	-255 to -128, 128 to 255
*	9	-511 to -256, 256 to 511
*	10	-1023 to -512, 512 to 1023

```
int InputCode( input_file )
BIT_FILE *input_file;
{
    int bit_count;
    int result;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

if ( InputRunLength > 0 ) {
    InputRunLength--;
    return( 0 );
}
bit_count = (int) InputBits( input_file, 2 );
if ( bit_count == 0 ) {
    InputRunLength = (int) InputBits( input_file, 4 );
    return( 0 );
}
if ( bit_count == 1 )
    bit_count = (int) InputBits( input_file, 1 ) + 1;
else
    bit_count = (int) InputBits( input_file, 2 ) +
        ( bit_count << 2 ) - 5;
result = (int) InputBits( input_file, bit_count );
if ( result & ( 1 << ( bit_count - 1 ) ) )
    return( result );
return( result - ( 1 << bit_count ) + 1 );
}

/* รูทีนนี้จะทำการอ่านข้อมูลเป็นบล็อกและทำการดีควอนไดซ์โดยใช้ควอนไดซ์เมตริกซ์ */

```

```

void ReadDCTData( input_file, input_data )
BIT_FILE *input_file;
int input_data[ N ][ N ];
{
    int i;
    int row;
    int col;

    for ( i = 0 ; i < ( N * N ) ; i++ ) {
        row = ZigZag[ i ].row;
        col = ZigZag[ i ].col;
        input_data[ row ][ col ] = InputCode( input_file ) *
            Quantum[ row ][ col ];
    }
}

```

/* รูทีนนี้ทำหน้าที่เก็บค่ารหัสข้อมูลลงไฟล์ผลลัพธ์ */

```

void OutputCode( output_file, code )
BIT_FILE *output_file;
int code;
{
    int top_of_range;
    int abs_code;
    int bit_count;

    if ( code == 0 ) {
        OutputRunLength++;
        return;
    }
    if ( OutputRunLength != 0 ) {
        while ( OutputRunLength > 0 ) {
            OutputBits( output_file, 0L, 2 );
            if ( OutputRunLength <= 16 ) {
                OutputBits( output_file,
                    (unsigned long) ( OutputRunLength - 1 ), 4 );
                OutputRunLength = 0;
            } else {
                OutputBits( output_file, 15L, 4 );
                OutputRunLength -= 16;
            }
        }
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    }
}
}
if ( code < 0 )
    abs_code = -code;
else
    abs_code = code;
top_of_range = 1;
bit_count = 1;
while ( abs_code > top_of_range ) {
    bit_count++;
    top_of_range = ( ( top_of_range + 1 ) * 2 ) - 1;
}
if ( bit_count < 3 )
    OutputBits( output_file, (unsigned long) ( bit_count + 1 ), 3 );
else
    OutputBits( output_file, (unsigned long) ( bit_count + 5 ), 4 );
if ( code > 0 )
    OutputBits( output_file, (unsigned long) code, bit_count );
else
    OutputBits( output_file, (unsigned long) ( code + top_of_range ),
                bit_count );
}

```

/* รุทีนนี้ทำการเก็บค่ารหัสข้อมูลที่ผ่านการจัดระดับลงไฟล์ผลลัพธ์ */

```

void WriteDCTData( output_file, output_data )
BIT_FILE *output_file;
int output_data[ N ][ N ];
{
    int i;
    int row;
    int col;
    double result;

    for ( i = 0 ; i < ( N * N ) ; i++ ) {
        row = ZigZag[ i ].row;
        col = ZigZag[ i ].col;
        result = output_data[ row ][ col ] / Quantum[ row ][ col ];
        OutputCode( output_file, ROUND( result ) );
    }
}

```

/* รุทีนนี้ทำการเขียนแถบข้อมูลลงไฟล์ผลลัพธ์ให้อยู่ในรูปแบบไฟล์รูปภาพ */

```

void WritePixelStrip( output, strip )
FILE *output;
unsigned char strip[ N ][ COLS ];
{
    int row;
    int col;

    for ( row = 0 ; row < N ; row++ )
        for ( col = 0 ; col < COLS ; col++ )
            putc( strip[ row ][ col ], output );
}

```

```

void WritePixelStrip1( output, strip1 )
FILE *output;
unsigned char strip1[ N ][ COLS1 ];

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    int row;
    int col;

    for ( row = 0 ; row < N ; row++ )
        for ( col = 0 ; col < COLS1 ; col++ )
            putc( strip1[ row ][ col ], output );
}

void WritePixelStrip2( output, strip2 )
FILE *output;
unsigned char strip2[ N ][ COLS2 ];
{
    int row;
    int col;

    for ( row = 0 ; row < N ; row++ )
        for ( col = 0 ; col < COLS2 ; col++ )
            putc( strip2[ row ][ col ], output );
}

/*
 *          สมการ FDCT
 *
 *          DCT = C * pixels * Ct
 */
void ForwardDCT( input, output )
unsigned char *input[ N ];
int output[ N ][ N ];
{
    double temp[ N ][ N ];
    double temp1;
    int i;
    int j;
    int k;

    /* MatrixMultiply( temp, input, Ct ); */
    for ( i = 0 ; i < N ; i++ ) {
        for ( j = 0 ; j < N ; j++ ) {
            temp[ i ][ j ] = 0.0;
            for ( k = 0 ; k < N ; k++ )
                temp[ i ][ j ] += ( (int) input[ i ][ k ] - 128 ) *
                                   Ct[ k ][ j ];
        }
    }

    /* MatrixMultiply( output, C, temp ); */
    for ( i = 0 ; i < N ; i++ ) {
        for ( j = 0 ; j < N ; j++ ) {
            temp1 = 0.0;
            for ( k = 0 ; k < N ; k++ )
                temp1 += C[ i ][ k ] * temp[ k ][ j ];
            output[ i ][ j ] = ROUND( temp1 );
        }
    }
}

/*
 *          สมการ IDCT
 *
 *          pixels = C * DCT * Ct
 */

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

void InverseDCT( input, output )
int input[ N ][ N ];
unsigned char *output[ N ];
{
    double temp[ N ][ N ];
    double templ;
    int i;
    int j;
    int k;

/* MatrixMultiply( temp, input, C ); */
    for ( i = 0 ; i < N ; i++ ) {
        for ( j = 0 ; j < N ; j++ ) {
            temp[ i ][ j ] = 0.0;
            for ( k = 0 ; k < N ; k++ )
                temp[ i ][ j ] += input[ i ][ k ] * C[ k ][ j ];
        }
    }

/* MatrixMultiply( output, Ct, temp ); */
    for ( i = 0 ; i < N ; i++ ) {
        for ( j = 0 ; j < N ; j++ ) {
            templ = 0.0;
            for ( k = 0 ; k < N ; k++ )
                templ += Ct[ i ][ k ] * temp[ k ][ j ];
            templ += 128.0;
            if ( templ < 0 )
                output[ i ][ j ] = 0;
            else if ( templ > 255 )
                output[ i ][ j ] = 255;
            else
                output[ i ][ j ] = (unsigned char) ROUND( templ );
        }
    }
}

/* รุทีนหลักของการบีบข้อมูลซึ่งต้องมีการส่งผ่านพารามิเตอร์มาให้ คือชื่อไฟล์อินพุต เอาท์พุต และ
ค่าคุณภาพของการบีบข้อมูล */

```

```

void DctCompressFile( input, output, quality)
FILE *input;
BIT_FILE *output;
int quality;
{
    int row;
    int col;
    int i;
    unsigned char *input_array[ N ];
    int output_array[ N ][ N ];

    if ( quality < 0 || quality > 25 )
        fatal_error( "Illegal quality factor of %d\n", quality );
    Initialize( quality );
    OutputBits( output, (unsigned long) quality, 8 );
    for ( row = 0 ; row < ROWS ; row += N ) {
        ReadPixelStrip( input, PixelStrip );
        for ( col = 0 ; col < COLS ; col += N ) {
            for ( i = 0 ; i < N ; i++ )
                input_array[ i ] = PixelStrip[ i ] + col;
            ForwardDCT( input_array, output_array );
            WriteDCTData( output, output_array );
        }
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    }
}
OutputCode( output, 1 );
}

void DctCompressFile1( input, output, quality)
FILE *input;
BIT_FILE *output;
int quality;
{
    int row;
    int col;
    int i;
    unsigned char *input_array[ N ];
    int output_array[ N ][ N ];

    if ( quality < 0 || quality > 25 )
        fatal_error( "Illegal quality factor of %d\n", quality );
    Initialize( quality );
    OutputBits( output, (unsigned long) quality, 8 );
    for ( row = 0 ; row < ROWS1 ; row += N ) {
        ReadPixelStrip1( input, PixelStrip1 );
        for ( col = 0 ; col < COLS1 ; col += N ) {
            for ( i = 0 ; i < N ; i++ )
                input_array[ i ] = PixelStrip1[ i ] + col;
            ForwardDCT( input_array, output_array );
            WriteDCTData( output, output_array );
        }
    }
    OutputCode( output, 1 );
}

```

```

void DctCompressFile2( input, output, quality)
FILE *input;
BIT_FILE *output;
int quality;
{
    int row;
    int col;
    int i;
    unsigned char *input_array[ N ];
    int output_array[ N ][ N ];

    if ( quality < 0 || quality > 25 )
        fatal_error( "Illegal quality factor of %d\n", quality );
    Initialize( quality );
    OutputBits( output, (unsigned long) quality, 8 );
    for ( row = 0 ; row < ROWS2 ; row += N ) {
        ReadPixelStrip2( input, PixelStrip2 );
        for ( col = 0 ; col < COLS2 ; col += N ) {
            for ( i = 0 ; i < N ; i++ )
                input_array[ i ] = PixelStrip2[ i ] + col;
            ForwardDCT( input_array, output_array );
            WriteDCTData( output, output_array );
        }
    }
    OutputCode( output, 1 );
}

```

/* รูทีนหลักของการขยายขนาดข้อมูลในรูปแบบ DCT และขยายกลับไปเป็นไฟล์รูปภาพเหมือนเดิม */

```

void DctExpandFile( input, output )
BIT_FILE *input;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

FILE *output;
{
    int row;
    int col;
    int i;
    int input_array[ N ][ N ];
    unsigned char *output_array[ N ];
    int quality;

    quality = (int) InputBits( input, 8 );
    Initialize( quality );
    for ( row = 0 ; row < ROWS ; row += N ) {
        for ( col = 0 ; col < COLS ; col += N ) {
            for ( i = 0 ; i < N ; i++ )
                output_array[ i ] = PixelStrip[ i ] + col;
            ReadDCTData( input, input_array );
            InverseDCT( input_array, output_array );
        }
        WritePixelStrip( output, PixelStrip );
    }
}

```

```

void DctExpandFile1( input, output )
BIT_FILE *input;
FILE *output;
{
    int row;
    int col;
    int i;
    int input_array[ N ][ N ];
    unsigned char *output_array[ N ];
    int quality;

    quality = (int) InputBits( input, 8 );
    Initialize( quality );
    for ( row = 0 ; row < ROWS1 ; row += N ) {
        for ( col = 0 ; col < COLS1 ; col += N ) {
            for ( i = 0 ; i < N ; i++ )
                output_array[ i ] = PixelStrip1[ i ] + col;
            ReadDCTData( input, input_array );
            InverseDCT( input_array, output_array );
        }
        WritePixelStrip1( output, PixelStrip1 );
    }
}

```

```

void DctExpandFile2( input, output )
BIT_FILE *input;
FILE *output;
{
    int row;
    int col;
    int i;
    int input_array[ N ][ N ];
    unsigned char *output_array[ N ];
    int quality;

    quality = (int) InputBits( input, 8 );
    Initialize( quality );
    for ( row = 0 ; row < ROWS2 ; row += N ) {
        for ( col = 0 ; col < COLS2 ; col += N ) {
            for ( i = 0 ; i < N ; i++ )
                output_array[ i ] = PixelStrip2[ i ] + col;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        ReadDCTData( input, input_array );
        InverseDCT( input_array, output_array );
    }
    WritePixelStrip2( output, PixelStrip2 );
}
}

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* โปรแกรมหลักของการบีบและขยายข้อมูลภาพสี ด้วยวิธีการ JPEG */
```

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <io.h>
#include <string.h>
#include <stdlib.h>
#include "bitio.c"
#include "errhand.c"
#include "newdct.c"
```

```
/* แมคโครในการบิตเศษ */
```

```
#define ROUND( a ) ((a) < 0 ) ? (int)((a) - 0.5):\
                        (int)((a) + 0.5))
```

```
/* กำหนดตัวแปรชนิดโกลบอลของโปรแกรม */
```

```
int      y_map[4][640], cr_map[4][640], cb_map[4][640];
int      red[4][640], green[4][640], blue[4][640];
int      cr[2][320], cb[2][320];
int      r, g, b, n, quality;
unsigned char  c;
long     count, ysize, crsize, cbsize;
float    y_val, rf, gf, bf;
```

```
BIT_FILE      *output;
BIT_FILE      *input;
FILE          *data;
FILE          *outjpeg;
FILE          *imagedata;
FILE          *ydata;
FILE          *crdata;
FILE          *cbdata;
FILE          *jpegfile;
```

```
char  imagename[20];
char  jpegname[20];
char  yname[20];
char  crname[20];
char  cbname[20];
```

```
int    length;
long   filesize;
```

```
/* โปรแกรมหลัก */
```

```
main(int argc, char *argv[])
{
```

```
    clrscr();
```

```
/* แนะนำวิธีการเรียกใช้งานโปรแกรม */
```

```
if( argc < 4 )
{
    printf("For Compression\n");
    printf("Usage : %s <inputfile> <outputfile> \
            quality(1-25) /c", argv[0]);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    printf("For Uncompression\n");
    printf("Usage : %s <inputfile> <outputfile> /u");
    exit(1);
}

if( argc == 5 && argv[4][1] != 'c')
{
    printf("Please specify /c");
    exit(1);
}

if( argc == 4 && argv[3][1] != 'u' )
{
    printf("Please specify /u");
    exit(1);
}

if( argv[4][1] == 'c' )
{
    if( ( imagedata = fopen(argv[1],"rb")) == NULL )
    {
        printf("Error reading input file.");
        exit(1);
    }

    strcpy(jpegname,argv[2]);
    length = strlen(argv[1]);

    strncpy(yname,argv[1],length-4);
    strcat(yname, ".y");

    strncpy(crname,argv[1],length-4);
    strcat(crname, ".cr");

    strncpy(cbname,argv[1],length-4);
    strcat(cbname, ".cb");

    if( ( ydata = fopen(yname,"wb")) == NULL )
    {
        printf("Error openning output file.");
        exit(1);
    }
    if( ( crdata = fopen(crname,"wb")) == NULL )
    {
        printf("Error openning output file.");
        exit(1);
    }

    if( ( cbdata = fopen(cbname,"wb")) == NULL )
    {
        printf("Error openning output file.");
        exit(1);
    }

    quality = atoi(argv[3]);
    printf("Compressing %s to %s\n",argv[1],argv[2]);
    printf("quality factor = %d\n",quality);

    /* แปลงข้อมูลภาพจากรูปแบบ RGB ไปเป็นรูปแบบ YCrCb
    และบีบข้อมูลด้วยวิธีการ CCIR-601 */

    Mbidir4();

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* บีบข้อมูลค่าความสว่าง */

Ycompress();

/* บีบข้อมูลค่าความสว่าง */

Crcompress();

/* บีบข้อมูลองค์ประกอบสีน้ำเงิน */

Cbcompress();

/* รวมข้อมูลทั้งหมดที่ถูกบีบเข้าในไฟล์ผลลัพธ์เดียวกัน */

Sum();

/* แสดงผลลัพธ์การบีบข้อมูล */

Result();
}

if( argv[3][1] == 'u' )
{
    if( ( jpegfile = fopen(argv[1],"rb")) == NULL )
    {
        printf("Error opening input file.");
        exit(1);
    }

    /* อ่านข้อมูลส่วนหัวของไฟล์เพื่อใช้ในแยกแยะส่วนต่าง ๆ ของข้อมูลได้ */

    fread(&ysize,sizeof(ysize),1,jpegfile);
    fread(&crsize,sizeof(crsize),1,jpegfile);
    fread(&cbsize,sizeof(cbsize),1,jpegfile);
    fclose(jpegfile);

    strcpy(jpegname,argv[1]);
    length = strlen(argv[2]);

    strncpy(yname,argv[2],length-4);
    strcat(yname, ".y");

    strncpy(crname,argv[2],length-4);
    strcat(crname, ".cr");

    strncpy(cbname,argv[2],length-4);
    strcat(cbname, ".cb");

    if( ( ydata = fopen(yname,"wb") ) == NULL )
    {
        printf("Error opening output file.");
        exit(1);
    }
    if( ( crdata = fopen(crname,"wb") ) == NULL )
    {
        printf("Error opening output file.");
        exit(1);
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

if( ( cbdata = fopen(cbname,"wb")) == NULL )
{
    printf("Error openning output file.");
    exit(1);
}

printf("Uncompressing %s to %s\n",argv[1],argv[2]);

/* ขยายข้อมูลค่าความสว่างด้วยวิธีการ JPEG */

Yuncompress();

/* ขยายข้อมูลองค์ประกอบสีแดงด้วยวิธีการ JPEG */

Cruncompress();

/* ขยายข้อมูลองค์ประกอบสีน้ำเงินด้วยวิธีการ JPEG */

CBuncompress();

if( ( imagedata = fopen(argv[2],"wb")) == NULL )
{
    printf("Error openning output file.");
    exit(1);
}

if( ( ydata = fopen(yname,"rb")) == NULL )
{
    printf("Error openning output file.");
    exit(1);
}

if( ( crdata = fopen(crname,"rb")) == NULL )
{
    printf("Error openning output file.");
    exit(1);
}

if( ( cbdata = fopen(cbname,"rb")) == NULL )
{
    printf("Error openning output file.");
    exit(1);
}

/* ขยายข้อมูลกลับด้วยวิธีการ CCIR-601 */

Mbidir4U();

/* ลบข้อมูลไฟล์ชั่วคราวทิ้ง */

Clear();
}

/* รุทีนหลักของการบีบข้อมูลด้วยวิธี CCIR-601 */

```

```

Mbidir4()
{

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

int i,j,k;

printf("CCIR-601 Compressing...\n");
n = 0;

for(k=0;k<120;k++)
{
    for(i=0;i<4;i++)
    {
        /* อ่านข้อมูล 4 แถว ของทั้ง 3 องค์ประกอบ */

        for(j=0;j<640;j++)
            red[i][j] = (int)fgetc(imagedata);
        for(j=0;j<640;j++)
            green[i][j] = (int)fgetc(imagedata);
        for(j=0;j<640;j++)
            blue[i][j] = (int)fgetc(imagedata);
    }
}

```

/* แปลงข้อมูลจากรูปแบบ RGB ไปเป็น YCrCb */

```

for(i=0;i<4;i++)
    for(j=0;j<640;j++)
    {
        rf = (float)red[i][j];
        gf = (float)green[i][j];
        bf = (float)blue[i][j];
        y_val = 0.299*rf + 0.587*gf + 0.114*bf;
        y_map[i][j] = (int)ROUND(y_val);
        cr_map[i][j] = (int)ROUND(rf-y_val);
        cb_map[i][j] = (int)ROUND(bf-y_val);
    }
Bidir4();
}

```

```

fclose(imagedata);
fclose(ydata);
fclose(crddata);
fclose(cbdata);
}

```

/* ฟังก์ชันในการปรับข้อมูลให้อยู่ในช่วง 0 - 255 */

```

int saturate(int a)
{
    if(a > 255) a = 255;
    if(a < 0 ) a = 0;
    return (a);
}

```

/* รุทีนสำหรับการบีบข้อมูลด้วยวิธี CCIR-601 */

```

int Bidir4()
{

```

```

    int i,j;

```

```

    for(i=0;i<2;i++)

```

```

        for(j=0;j<320;j++)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    cr[i][j] = (cr_map[i*2][j*2] + cr_map[i*2+1][j*2] +
                cr_map[i*2][j*2+1] + cr_map[i*2+1][j*2+1] )/4;
    cb[i][j] = (cb_map[i*2][j*2] + cb_map[i*2+1][j*2] +
                cb_map[i*2][j*2+1] + cb_map[i*2+1][j*2+1] )/4;
}

for(i=0;i<4;i++)
    for(j=0;j<640;j++)
        fputc(y_map[i][j],ydata);

for(i=0;i<2;i++)
    for(j=0;j<320;j++)
        fputc(cr[i][j],crdata);

for(i=0;i<2;i++)
    for(j=0;j<320;j++)
        fputc(cb[i][j],cbdata);

gotoxy(35,3);
printf("processed %4d%%", (int)((n/3)+1)*2.5);
n++;
}

```

/* รุทีนหลักของการขยายข้อมูลด้วยวิธี CCIR-601 */

```

Mbidir4U()
{
    int i,j,k;

    printf("CCIR-601 Uncompressing...\n\n");
    n = 0;

    for(k=0;k<120;k++)
    {
        for(i=0;i<4;i++)
        {
            Bidir4U();
        }

        for(i=0;i<4;i++)
        {
            /* เก็บข้อมูลของแต่ละองค์ประกอบลงไฟล์ */

            for(j=0;j<640;j++)
            {
                r = saturate(red[i][j]);
                fputc(r,imagedata);
            }

            for(j=0;j<640;j++)
            {
                g = saturate(green[i][j]);
                fputc(g,imagedata);
            }

            for(j=0;j<640;j++)
            {
                b = saturate(blue[i][j]);
                fputc(b,imagedata);
            }
        }
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    }
}
}

fclose(imagedata);
fclose(ydata);
fclose(crdata);
fclose(cbdata);

}

/*  routines for the exchange of data with CCIR-601 */

int Bidir4U()
{
    int i,j;

    for(i=0;i<4;i++)
        for(j=0;j<640;j++)
            y_map[i][j] = (int)fgetc(ydata);

    for(i=0;i<2;i++)
        for(j=0;j<320;j++)
            cr[i][j] = (char)fgetc(crdata);

    for(i=0;i<2;i++)
        for(j=0;j<320;j++)
            cb[i][j] = (char)fgetc(cbdata);

    for(i=0;i<4;i++)
        for(j=0;j<640;j++)
        {
            red[i][j] = cr[i/2][j/2] + y_map[i][j];
            blue[i][j] = cb[i/2][j/2] + y_map[i][j];
            green[i][j] = (int)ROUND(((float)y_map[i][j]
                *1.703 - (float)red[i][j]*0.5093
                - (float)blue[i][j]*0.1942));
        }

    gotoxy(35,5);
    printf("processed %4d%%", (int)((n/3)+1)*2.5);
    n++;
}

```

/* routines for the exchange of data with JPEG */

```

Ycompress()
{
    printf("\nCompressing luminance...\n");
    data = fopen(yname , "rb" );
    if ( data == NULL )
        fatal_error( "Error opening %s for input\n", yname );
    output = OpenOutputBitFile("l.dct");
    if ( output == NULL )
        fatal_error("Error opening %s for output\n", "l.dct" );
    DctCompressFile1( data, output, quality);
    CloseOutputBitFile( output );
    fclose( data );
    return( 0 );
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* รูทีนสำหรับการขยายข้อมูลค่าความสว่างด้วยวิธี JPEG */
```

```
Yuncompress()  
{  
  
    printf("Uncompressing luminance...\n");  
    input = OpenInputBitFile(jpegname);  
    fseek(input->file, 12L, SEEK_SET);  
    DctExpandFile1(input, ydata);  
    CloseInputBitFile(input);  
    fclose(ydata);  
  
}
```

```
/* รูทีนสำหรับการบีบข้อมูลองค์ประกอบสีแดงด้วยวิธี JPEG */
```

```
CRcompress()  
{  
  
    printf("Compressing red component...\n");  
    data = fopen(crname, "rb");  
    if ( data == NULL )  
        fatal_error( "Error opening %s for input\n", crname );  
    output = OpenOutputBitFile("cr.dct");  
    if ( output == NULL )  
        fatal_error("Error opening %s for output\n", "cr.dct" );  
    DctCompressFile2( data, output, quality);  
    CloseOutputBitFile( output );  
    fclose( data );  
    return( 0 );  
  
}
```

```
/* รูทีนสำหรับการขยายข้อมูลองค์ประกอบสีแดงด้วยวิธี JPEG */
```

```
CRuncompress()  
{  
  
    printf("Uncompressing red component...\n");  
    input = OpenInputBitFile(jpegname);  
    fseek(input->file, (long) (ysize+12), SEEK_SET);  
    DctExpandFile2(input, crdata);  
    CloseInputBitFile(input);  
    fclose(crdata);  
  
}
```

```
/* รูทีนสำหรับการบีบข้อมูลองค์ประกอบสีน้ำเงินด้วยวิธี JPEG */
```

```
CBcompress()  
{  
  
    printf("Compressing blue component...\n");  
    data = fopen(cbname, "rb");  
    if ( data == NULL )  
        fatal_error( "Error opening %s for input\n", cbname );  
    output = OpenOutputBitFile("cb.dct");  
    if ( output == NULL )  
        fatal_error("Error opening %s for output\n", "cb.dct" );  
    DctCompressFile2( data, output, quality);
```

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

CloseOutputBitFile( output );
fclose( data );
return( 0 );
}

```

/* ภูทึนสำหรับการขยายข้อมูลองค์ประกอบสีน้ำเงินด้วยวิธี JPEG */

```

CBuncompress()
{
    printf("Uncompressing blue component...\n");
    input = OpenInputBitFile(jpegname);
    fseek(input->file, (long) (ysize+crsize+12), SEEK_SET);
    DctExpandFile2(input, cbdata);
    CloseInputBitFile(input);
    fclose(cbdata);
}

```

/* ภูทึนสำหรับรวมไฟล์รวมไฟล์ที่ถูกบีบแต่ละองค์ประกอบเข้าด้วยกัน */

```

Sum()
{
    if( ( ydata = fopen("l.dct","rb")) == NULL )
    {
        printf("Error openning input file.");
        exit(1);
    }

    if( ( crdata = fopen("cr.dct","rb")) == NULL )
    {
        printf("Error openning input file.");
        exit(1);
    }

    if( ( cbdata = fopen("cb.dct","rb")) == NULL )
    {
        printf("Error openning input file.");
        exit(1);
    }

    remove(jpegname);

    if( ( jpegfile = fopen(jpegname,"ab")) == NULL )
    {
        printf("Error openning input file.");
        exit(1);
    }

    ysize = filelength(fileno(ydata));
    crsize = filelength(fileno(crdata));
    cbsize = filelength(fileno(cbdata));

    fwrite(&ysize, sizeof(ysize), 1, jpegfile);
    fwrite(&crsize, sizeof(crsize), 1, jpegfile);
    fwrite(&cbsize, sizeof(cbsize), 1, jpegfile);
}

```

```

for(count=0;count<ysize;count++)
{
    c = fgetc(ydata);
    fputc(c, jpegfile);
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

fclose(ydata);

crsize = filelength(fileno(crdata));

for(count=0;count<crsize;count++)
{
    c = fgetc(crdata);
    fputc(c, jpegfile);
}

fclose(crdata);

cbsize = filelength(fileno(cbdata));

for(count=0;count<cbsize;count++)
{
    c = fgetc(cbdata);
    fputc(c, jpegfile);
}

fclose(cbdata);
fclose(jpegfile);

remove("l.dct");
remove("cr.dct");
remove("cb.dct");
remove(ynname);
remove(crname);
remove(cbname);
}

Clear()
{
    remove(ynname);
    remove(crname);
    remove(cbname);
}

/* รวบรวมผลลัพธ์จากการบีบข้อมูล */

Result()
{
    jpegfile = fopen(jpegname, "rb");
    filesize = filelength(fileno(jpegfile));
    fclose(jpegfile);
    printf("\nResult of Compression\n");
    printf("-----\n");
    printf("Input file size      921600 bytes\n");
    printf("Output file size      %8lu bytes\n", filesize);
    printf("Percent compressed    \
    %8.2f\n", 100 - (((float) filesize / 921600.0) * 100));
    printf("Compression ratio    \
    %8.2f", 921600.0 / filesize);
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก จ

โปรแกรมการคำนวณค่าความผิดพลาดของภาพจากการบีบอัดและขยายข้อมูล



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include <stdio.h>
#include <math.h>

FILE *data,*uncom;
long float error;
unsigned char r1[640],b1[640],g1[640];
unsigned char r2[640],b2[640],g2[640];
int diff1[640],diff2[640],diff3[640];
int i,j,k;
```

```
main(int argc,char *argv[])
```

```
{
    clrscr();

    if( argc != 3 )
    {
        printf("Usage : %s <file1> <file2>");
        exit(1);
    }

    if( (data = fopen(argv[1],"rb")) == NULL )
    {
        printf("Error reading %s",argv[1]);
        exit(1);
    }

    if( (uncom = fopen(argv[2],"rb")) == NULL )
    {
        printf("Error reading %s",argv[2]);
        exit(1);
    }

    printf("Processing...");
    for(j=0;j<480;j++)
    {
        fread(r1,640,1,data);
        fread(g1,640,1,data);
        fread(b1,640,1,data);

        fread(r2,640,1,uncom);
        fread(g2,640,1,uncom);
        fread(b2,640,1,uncom);

        for(i=0;i<640;i++)
        {
            diff1[i] = r1[i] - r2[i];
            diff2[i] = g1[i] - g2[i];
            diff3[i] = b1[i] - b2[i];
        }

        for(i=0;i<640;i++)
            error += ( diff1[i]*diff1[i] + diff2[i]*diff2[i]
                    + diff3[i]*diff3[i] );
        gotoxy(1,2);
        printf("Calculating line %5d",j);
        printf("      Error = %15.2lf",error);
    }

    error = error/921600.0;
    error = sqrt(error);
    printf("\nRMSE = %5.2lf",error);
    fclose(data);
    fclose(uncom);
    getch();
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ขอขอบพระคุณ ผศ.ดร. สุวิพล สิริชีวะภาค และ อ. เกรียงไกร วงศ์โรจน์ภรณ์ ในฐานะ
อาจารย์ที่ปรึกษา ที่กรุณาให้คำแนะนำและช่วยเหลือในการทำปริญญานิพนธ์ฉบับนี้ จนสำเร็จลุล่วงด้วยดี
ขอขอบพระคุณ อาจารย์ กฤษณ์ วงศ์รุจิระ ที่ให้ความรู้เรื่องการใช้ภาษา C
และขอบคุณ คุณธงชัย มณีชูเกศ ที่ให้ความช่วยเหลือในการถ่ายภาพ บันทึกผลการทดลอง
รวมทั้งเพื่อนๆ นักศึกษาทุกท่าน ที่ให้ความช่วยเหลือด้วยดีตลอดมา

คณะผู้จัดทำ

27 มีนาคม 2539



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หนังสืออ้างอิง

- [1] RISC , “ PC MAGAZINE THAILAND “ , กรุงเทพฯ , ปีที่ 1 ฉบับที่ 3 มิถุนายน 2536
- [2] C. Wayne Brown and Barry J. Shepherd , “ Graphics File Format reference and guide “ ,
U.S.A. Prentice Hall , 1995
- [3] Mark Nelson , “ The Data Compression Book “ , U.S.A. , M&T Publishing,Inc. , 1992
- [4] Phillip E. Mattison , “ Practical Digital Video with Programming Example in C “ , U.S.A. ,
John Wilry & Sons,Inc. , 1994
- [5] ดร. ยุทธพงศ์ รังสรรค์เสรี , “ Lecture Notes วิชา Multimedia Communication “ ,
สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง , 2539

