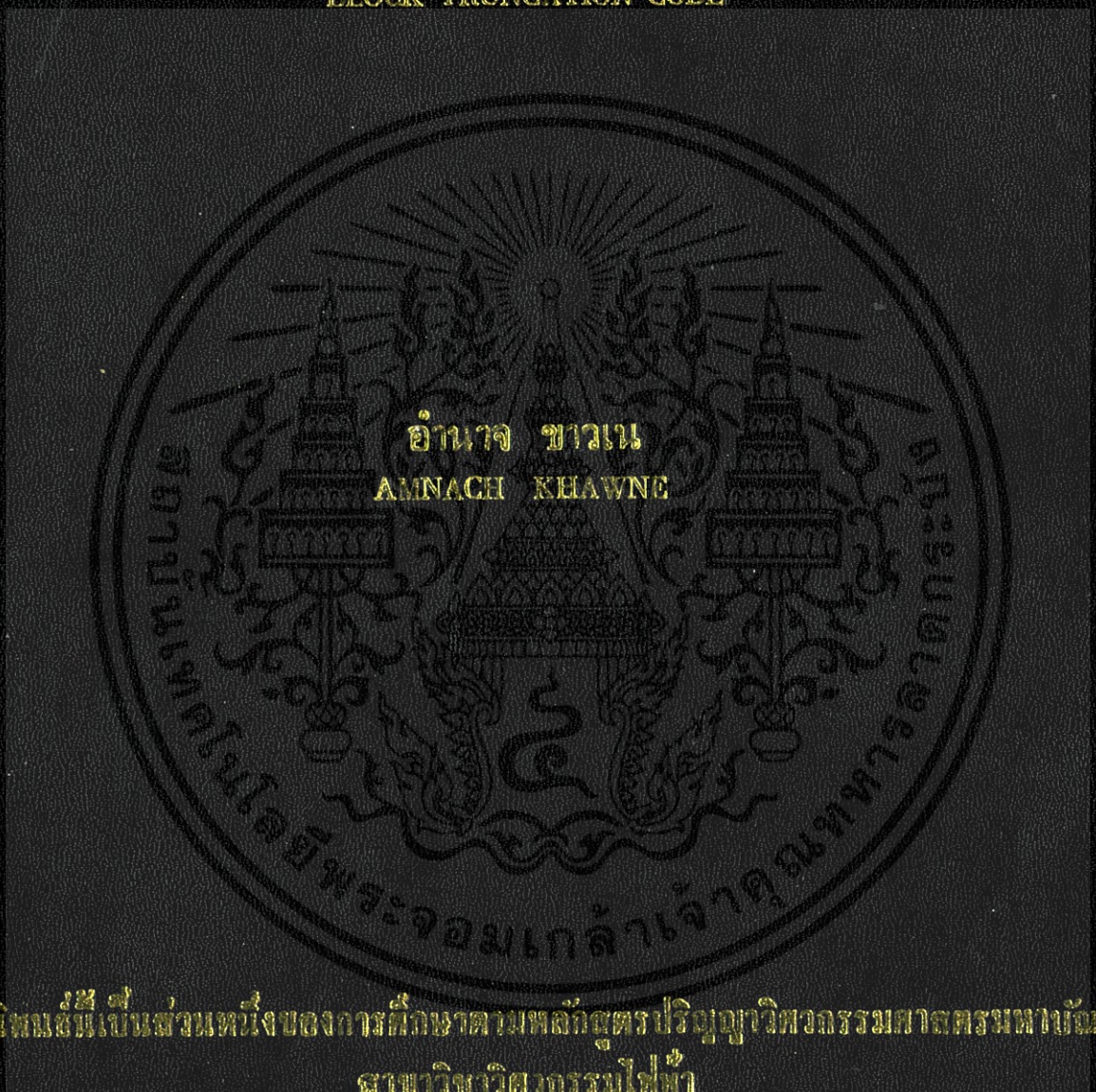
การบีบอัดข้อมูลภาพโดยการเข้ารหัสแบบบล็อกหลายขนาด

IMAGE COMPRESSION BASED ON VARIABLE BLOCK TRUNCATION CODE



อำนาจ ขาวเน

AMNACH KHAWNE

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2545

ISBN 974-648-944-5

การบีบอัดข้อมูลภาพโดยการเข้ารหัสแบบบล็อกหลายขนาด

IMAGE COMPRESSION BASED ON VARIABLE BLOCK TRUNCATION CODE

อำนาจ ขาวเน

AMNACH KHAWNE

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2545

ISBN 974-648-944-5

IMAGE COMPRESSION BASED ON VARIABLE BLOCK TRUNCATION CODE

AMNACH KHAWNE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2002

ISBN 974-648-944-5

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การบีบอัดข้อมูลภาพโดยการเข้ารหัสแบบบล็อกหลายขนาด
IMAGE COMPRESSION BASED ON VARIABLE BLOCK TRUNCATION CODE

ชื่อนักศึกษา นายอำนาจ ขาวเน

รหัสประจำตัว 42061037

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ผู้ควบคุมวิทยานิพนธ์ ผศ.ดร.สุทธิชัย นพนาคีพงษ์

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
รศ.ดร.ยุทธพงษ์	รังสรรค์เสรี	
รศ.ดร.ปัญญา	ฐิติมัชฌิมา	
รศ.นิภา	ลีลารุจิ	
รศ.ดร.สุวิพล	สิทธิชีวภาค	
ผศ.ดร.สุทธิชัย	นพนาคีพงษ์	

วัน/เดือนปี ที่สอบ 23 พฤษภาคม 2545 **เวลา** 10.30-12.30 น.

สถานที่สอบ ณ อาคาร 12 ชั้น ชั้น 4 (ห้อง E12-404)

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.บุญชัย อัตชู)

คณบดีบัณฑิตวิทยาลัย

วันที่ 20 **เดือน** พฤษภาคม **พ.ศ.** 2545

หัวข้อวิทยานิพนธ์ การบีบอัดข้อมูลภาพโดยการเข้ารหัสแบบบล็อกหลายขนาด

นักศึกษา นายอำนาจ ขาวเน

รหัสประจำตัว 42061037

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมไฟฟ้า

พ.ศ. 2545

อาจารย์ผู้ควบคุมวิทยานิพนธ์ ผศ.ดร.สุทธิชัย นพนาคีพงษ์

บทคัดย่อ

วิทยานิพนธ์เล่มนี้ได้นำเสนอวิธีการบีบอัดข้อมูลภาพ โดยการเข้ารหัสแบบบล็อกหลายขนาด ซึ่งใช้วิธีการแยกองค์ประกอบควอดทรีแบบใหม่ โดยอาศัยหลักการของลากราเกียนมัลติพลายเออร์ การแยกองค์ประกอบควอดทรีเป็นวิธีการที่ใช้แสดงภาพในลักษณะที่เป็นลำดับชั้น แต่อย่างไรก็ตามการที่จะแบ่งระดับชั้นของภาพได้นั้นจำเป็นต้องกำหนดค่าเทรชโฮล ซึ่งเป็นเรื่องยากที่จะหาค่าเทรชโฮลที่เหมาะสมในการแบ่งภาพทั้งภาพ ในที่นี้ได้นำเสนอวิธีการตัดสินใจในการแบ่งบล็อกโดยใช้ค่าของลา กราเกียนมาเป็นตัวกำหนดการเติบโตของโครงสร้างต้นไม้ ซึ่งในวิธีการที่ได้นำเสนอนั้นจะให้โครงสร้างต้นไม้เติบโตทีละโหนดแตกต่างกับการเติบโตของโครงสร้างต้นไม้แบบเดิม ซึ่งเติบโตทีละชั้น และผลที่ได้ของวิธีการที่ได้นำเสนอมีข้อดีกว่าวิธีการแบบเดิม

Thesis Title	Image Compression Based on Variable Block Truncation Code
Student	Mr.Amnach Khawne
Student ID.	42061037
Degree	Master of Engineering
Programme	Electrical Engineering
Year	2002
Thesis Advisor	Asst. Dr. Suthichai Noppanakeepong

ABSTRACT

In this thesis, we proposes image coding by using variable block truncation code, which uses a new quadtree growing decided by the Lagrangian function. Quadtree decomposition (QT) is a technique for representing an image data as hierarchical data structure. However, QT technique requires the threshold for the test-block criterion, it is difficult to define the optimum threshold value which is covered the whole image data. In our algorithm, we use Lagrangian cost instead of the threshold for decision the tree grows. In our algorithm, the tree grows one node at a time that differ from the method of the quadtree growing which grows tree one layer at the time, and our results shown that the algorithm obtained a good quality of image reconstruction.

กิตติกรรมประกาศ

การจัดทำวิทยานิพนธ์ฉบับนี้ได้รับความสำเร็จลุล่วงไปด้วยดีก็เพราะการช่วยเหลือและให้ความอนุเคราะห์จากบุคคลหลายท่าน ขอขอบพระคุณแม่ ขอขอบคุณพี่ๆและน้องๆห้องวิจัยการสื่อสารดาวเทียมที่ให้ความช่วยเหลือและให้กำลังใจเป็นอย่างมาก ขอขอบพระคุณ ผศ.ดร.สุทธิชัย นพนาคีพงษ์ ในฐานะที่เป็นอาจารย์ที่ปรึกษาที่ให้ความดูแลและเอาใจใส่อย่างสม่ำเสมอ ขอขอบพระคุณ ผศ. กฤษณ์ วงศ์รุจิระและรศ.ดร. ปัญญา ฐิติมัฌชิมา ที่ให้คำปรึกษา

ขอขอบพระคุณ รศ.ดร. ซุโตมุ คาวาบาตะ (Assoc. Prof. Tsutomu KAWABATA) ที่ให้คำแนะนำและให้คำปรึกษา พร้อมทั้งเอาใจใส่ดูแลเสมอมาในการทำวิจัย

ขอขอบพระคุณห้องวิจัยของ ศาสตราจารย์ ทาเคดะ มิทซูโอะ (Prof. Takeda MITSUO) ที่ได้เอื้อเฟื้อสถานที่ในการทำวิจัย

อำนาจ ขาวเน

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....I

บทคัดย่อภาษาอังกฤษ.....II

กิตติกรรมประกาศ.....III

สารบัญ.....IV

สารบัญตาราง.....VII

สารบัญรูป.....VIII

บทที่ 1 บทนำ.....1

1.1 ความเป็นมาและความสำคัญของปัญหา.....1

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษาวิจัย.....2

1.3 สมมติฐานของการศึกษาวิจัย.....2

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย.....3

1.5 ขอบเขตของการวิจัย.....3

1.6 ขั้นตอนของการศึกษาวิจัย.....3

บทที่ 2 การบีบอัดข้อมูลภาพเบื้องต้น.....4

2.1 ทฤษฎีการควอนไทซ์.....5

2.2 การควอนไทซ์แบบสเกลาร์ที่ใช้ระดับคงที่และไม่คงที่.....6

2.3 เวกเตอร์ควอนไทซ์.....8

2.4 การควอนไทซ์โดยการเก็บรักษาโมเมนต์.....11

2.5 อัตราข่าวสาร.....13

2.5.1 ฟังก์ชันความผิดเพี้ยน- อัตราบิต.....14

2.5.2 คำนิยามของความผิดเพี้ยน.....14

2.5.3 อัตราความผิดเพี้ยนของข้อมูลแบบเกาส์เซียน.....16

2.5.4 การประมาณค่าที่ไม่ต่อเนื่องของฟังก์ชันความผิดเพี้ยน- อัตราบิต.....19

สารบัญ (ต่อ)

หน้า

2.6 ความรู้เกี่ยวกับการบีบอัดข้อมูลภาพเบื้องต้น.....20
2.6.1 การบีบอัดข้อมูลภาพแบบที่มีการสูญเสีย.....22
2.6.1.1 การบีบอัดข้อมูลภาพโดยการแปลง.....22
2.6.1.2 การเข้ารหัสโดยการทำนาย.....26
2.6.2 การบีบอัดข้อมูลแบบที่ไม่มีการสูญเสีย.....27
2.7 การวัดประสิทธิภาพของการบีบอัดข้อมูลภาพ.....38
2.7.1 การวัดค่าผิดพลาดกำลังสองเฉลี่ย.....39
2.7.2 การค่าสัดส่วนสัญญาณต่อสัญญาณรบกวน และค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด.....39
2.7.3 อัตราบิตและอัตราการลดข้อมูล.....40

บทที่ 3 การเข้ารหัสแบบบล็อก.....41
3.1 การเข้ารหัสแบบบล็อกที่มีขนาดคงที่.....41
3.2 การเข้ารหัสแบบบล็อกที่มีขนาดคงที่โดยใช้ค่าสมบูรณ์โมเมนต์.....45
3.3 การเข้ารหัสแบบบล็อกที่มีขนาดคงที่โดยการใช้การเข้ารหัสที่มีสามระดับ.....46
3.4 การเข้ารหัสแบบแบ่งข้อมูลภาพออกเป็นบล็อกหลายขนาด.....50
3.5 การเข้ารหัสแบบบล็อกที่มีหลายขนาดโดยใช้การแยกองค์ประกอบแบบควอดทรี.....50

บทที่ 4 วิธีดำเนินการวิจัย.....53
4.1 ขั้นตอนการเข้ารหัสแบบบล็อกหลายขนาดโดยการใช้ค่าลากรานจ์.....53
4.1.1 การจัดสรรบิตรหัสของโครงสร้างต้นไม้.....57
4.1.2 การคำนวณอัตราบิตและความผิดเพี้ยน.....58
4.2 ขั้นตอนการถอดรหัสแบบบล็อกหลายขนาดโดยการใช้ค่าลากรานจ์.....62

บทที่ 5 ผลการศึกษาวิจัย.....64

สารบัญ (ต่อ)

หน้า

5.1 ผลของการเปรียบเทียบค่าการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองและค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด.....65
5.2 ผลของการเปรียบเทียบอัตราการบีบอัด.....68
5.3 ผลของการเปรียบเทียบจำนวนบล็อกขนาดต่างๆ.....69
5.4 การเปรียบเทียบค่าความผิดพลาดกำลังสองของวิธีการต่างๆ 72
5.5 การเปรียบเทียบค่าค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของวิธีการต่างๆ.....76

บทที่ 6 สรุปผลและแนวทางการพัฒนา.....82
6.1 สรุปผลการวิจัย.....82
6.2 แนวทางการพัฒนา.....83

เอกสารอ้างอิง.....84
ภาคผนวก.....87
ภาคผนวก ก วิธีการของตัวคูณลากรานจ์.....88
ภาคผนวก ข โปรแกรม MATLAB ที่ใช้ในการทดลอง.....92
ภาคผนวก ค ผลงานที่ได้รับการตีพิมพ์.....111
ประวัติผู้เขียน.....112

สารบัญตาราง

ตารางที่ **หน้า**

2.1 ความน่าจะเป็นของสัญลักษณ์ในการเข้ารหัสด้วยวิธีฮัฟแมน.....31

2.2 แสดงการแบ่งขอบเขตตามความน่าจะเป็นสะสมในการเข้ารหัสเลขคณิต.....33

2.3 ปทานุกรมสำหรับวิธีการของเลมเปล-ซิป.....36

4.1 การจัดสรรบิตให้กับโครงสร้างต้นไม้.....57

5.1 แสดงค่าความผิดพลาดเฉลี่ยกำลังสองที่ค่าเทรชโฮลเท่ากับ 10 และค่าระดับการควอนไทซ์เท่ากับ 8.....64

5.2 แสดงค่าความผิดพลาดเฉลี่ยกำลังสองที่ค่าเทรชโฮลเท่ากับ 20 และค่าระดับการควอนไทซ์เท่ากับ 8.....65

5.3 แสดงค่าความผิดพลาดเฉลี่ยกำลังสองที่ค่าเทรชโฮลเท่ากับ 50 และค่าระดับการควอนไทซ์เท่ากับ 8.....65

5.4 แสดงค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดที่ค่าเทรชโฮลเท่ากับ 10 และค่าระดับการควอนไทซ์เท่ากับ 8.....66

5.5 แสดงค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดที่ค่าเทรชโฮลเท่ากับ 20 และค่าระดับการควอนไทซ์เท่ากับ 8.....66

5.6 แสดงค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดที่ค่าเทรชโฮลเท่ากับ 50 และค่าระดับการควอนไทซ์เท่ากับ 8.....67

5.7 การเปรียบเทียบวิธีการที่ได้นำเสนอกับวิธีการอื่น.....79

5.8 การเปรียบเทียบจำนวนบล็อกของวิธีการที่ได้นำเสนอกับวิธีการอื่น.....79

สารบัญรูป

รูปที่ **หน้า**

2.1 การประมวลภาพด้วยคอมพิวเตอร์ 5
2.2 การแบ่งเส้นจำนวนจริงออกเป็นช่วงๆในการควอนไทซ์ข้อมูล..... 6
2.3 การควอนไทซ์แบบจัดระดับคงที่.....7
2.4 การควอนไทซ์แบบที่ระดับไม่คงที่.....7
2.5 การเข้ารหัสและการถอดรหัสสัญญาณ 8
2.6 เซลล์ของเวกเตอร์ควอนไทซ์.....9
2.7 การกระจายสัมประสิทธิ์ของการแปลงโคซายน์เต็มหน่วย.....21
2.8 การบีบอัดข้อมูลภาพแบบต่างๆ.....21
2.9 วิธีการเข้าและถอดรหัสแบบทำนาย.....26
2.10 การเข้ารหัสในระบบ PCM.....28
2.11 การสร้างรหัสฐานสองด้วยวิธีฮัฟแมน.....31
2.12 การแบ่งขอบเขตตามข้อมูลที่เพิ่มขึ้นในการเข้ารหัสเลขคณิต.....34
2.13 การบีบอัดข้อมูลภาพแบบ JPEG.....38
3.1 การเข้ารหัสแบบบล็อกขนาดคงที่.....43
3.2 การแทนที่แต่ละพิกเซลด้วยจุดภาพที่มีสองระดับ.....43
3.3 ขั้นตอนการเข้ารหัสแบบบล็อกคงที่.....44
3.4 ขั้นตอนการถอดรหัสแบบบล็อกคงที่.....45
3.5 ขั้นตอนการเข้ารหัสแบบบล็อกโดยการควอนไทซ์สามระดับ.....48
3.6 ขั้นตอนการถอดรหัสแบบบล็อกโดยการควอนไทซ์สามระดับ..... 49
3.7 แสดงการแบ่งบล็อกของภาพ Lena ออกเป็นบล็อกหลายขนาด.....50
3.8 แสดงภาพที่ถูกแบ่งเป็นบล็อกที่มีหลายขนาดด้วยวิธีการแยกองค์ประกอบ
แบบควอดทรี.....51
4.1 แสดงฟังก์ชันอัตราบิต-ความผิดเพี้ยนในทางทฤษฎี.....54
4.2 ฟังก์ชันของอัตราบิต-ความผิดเพี้ยนที่ได้ทางปฏิบัติโดยทำตามขั้นตอนการ
แบ่งบล็อกด้วยวิธีการหาค่าลากรานจ์.....55
4.3 แสดงรูปโครงสร้างต้นไม้ของการแบ่งภาพออกเป็นบล็อกหลายขนาด.....57
4.4 การแบ่งภาพของบล็อกขนาด 16×16 และรหัสต้นไม้ของบล็อก.....58
4.5 ตำแหน่งของโหนดใบไม้และโหนดภายใน.....59

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

4.6 การเข้ารหัสแบบบล็อกหลายขนาดโดยการใช้ค่าลากรานจ์ (LVBTC).....60
4.7 การถอดรหัสแบบบล็อกหลายขนาดโดยการใช้ค่าลากรานจ์ (LVBTC).....61
4.8 การอ้างอิงข้อมูลของจากตำแหน่งของต้นไม้รหัส.....62
4.9 แสดงรูปบล็อกไดอะแกรมของการเข้ารหัสแบบ LVBTC.....63
4.10 แสดงรูปบล็อกไดอะแกรมของการถอดรหัสแบบ LVBTC.....63
5.1 แสดงรูปที่ใช้ในงานวิจัยทั้งหมด.....64
5.2 แสดงอัตราการบีบอัดที่ค่าลากรานจ์ต่างๆ โดยที่เทรชโฮลเท่ากับ 10
และระดับการควอนไทซ์เท่ากับ 8.....68
5.3 แสดงอัตราการบีบอัดที่ค่าลากรานจ์ต่างๆ โดยที่เทรชโฮลเท่ากับ 10
และระดับการควอนไทซ์เท่ากับ 16.....69
5.4 แสดงอัตราการบีบอัดที่ค่าลากรานจ์ต่างๆ โดยที่เทรชโฮลเท่ากับ 10
และระดับการควอนไทซ์เท่ากับ 32.....69
5.5 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Lena ขนาด512x512
โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8.....70
5.6 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Jet ขนาด512x512
โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8..... 70
5.7 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Boat ขนาด512x512
โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8..... 70
5.8 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Peppers ขนาด512x512
โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8.....71
5.9 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Lena ขนาด256x256
โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8..... 71
5.10 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Baboon ขนาด256x256
โดยเทรชโฮล เท่ากับ 10 และระดับควอนไทซ์เท่ากั 8.....71
5.11 แสดงการเปรียบเทียบจำนวนบล็อกของภาพที่เข้าสู่ขบวนการเข้ารหัส
แบบ AMBTC.....72
5.12 แสดงการเปรียบเทียบจำนวนบล็อกของภาพที่เข้าสู่ขบวนการเข้ารหัส
แบบ DCT.....72

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

5.13 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Lena ขนาด 512x512.....73

5.14 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Peppers ขนาด 512x512.....73

5.15 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Jet ขนาด 512x512.....74

5.16 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Boat ขนาด 512x512.....74

5.17 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Lena ขนาด 256x256.....75

5.18 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Baboon ขนาด 256x256.....75

5.19 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Lena ขนาด 512x512.....76

5.20 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Peppers ขนาด 512x512.....77

5.21 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Jet ขนาด 512x512.....77

5.22 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Boat ขนาด 512x512.....78

5.23 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Lena ขนาด 256x256.....78

5.24 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Baboon ขนาด 256x256.....79

5.25 แสดงภาพที่ได้จากการสร้างกลับของวิธีการต่างๆ.....79

บทที่ 1
บทนำ

ปัจจุบันการบีบอัดข้อมูลภาพหรือการลดขนาดของข้อมูลภาพเข้ามามีบทบาทในการลดขนาดของข้อมูลภาพเพื่อการจัดเก็บหรือการส่งข้อมูลและการประยุกต์ใช้ในด้านอื่นๆ ในเรื่องของการส่งข้อมูลนั้นด้วยเหตุที่อุปกรณ์สื่อสารมีข้อจำกัดของแบนด์วิดท์ ดังนั้นจึงมีความจำเป็นในการลดข้อมูลของภาพเพื่อให้สามารถส่งข้อมูลโดยที่ข้อมูลที่ส่งนั้นมีจำนวนน้อยที่สุดเท่าที่จะเป็นไปได้ เพื่อให้ใช้ช่องสัญญาณในการส่งข้อมูลได้อย่างมีประสิทธิภาพมากที่สุด นอกจากเรื่องของแบนด์วิดท์แล้วในเรื่องของการเก็บข้อมูลก็เป็นส่วนสำคัญเช่นเดียวกันเพราะว่าปัจจุบันอุปกรณ์การเก็บข้อมูลมีราคาแพง การการลดขนาดของข้อมูลภาพถือเป็นส่วนที่ช่วยให้การใช้พื้นที่ในการเก็บข้อมูลได้ประหยัดเนื้อที่ในการเก็บ ถ้าการลดข้อมูลทำได้มากเท่าไรก็ยิ่งทำให้พื้นที่การใช้งานมีเพิ่มขึ้นมากเท่านั้น แต่ในที่นี้การลดขนาดของข้อมูลภาพควรคำนึงถึงคุณภาพของภาพหลังจากที่ได้มีการสร้างกลับด้วย เพราะถ้าอัตราการลดขนาดของข้อมูลภาพมีค่าสูงย่อมทำเกิดความผิดเพี้ยนหลังจากการสร้างกลับภาพมีมากด้วยเช่นกัน(ในกรณีการลดขนาดของข้อมูลภาพที่มีการสูญเสีย) ดังนั้นวิธีการลดขนาดของข้อมูลภาพที่ดีควรที่จะมีอัตราการลดขนาดของข้อมูลภาพสูงและมีคุณภาพที่ดีหลังจากการสร้างกลับภาพ

1.1 ความเป็นมาและความสำคัญของปัญหา

การเข้ารหัสแบบบล็อกซึ่งแต่เดิมเป็นการเข้ารหัสที่มีขนาดคงที่ [1, 2, 3, 4, 5] และการเข้ารหัสแบบดังกล่าวข้อมูลภาพ นั้นบางส่วนประกอบไปด้วยข้อมูลที่มีรายละเอียดของภาพสูงหรือข้อมูลภาพที่มีการเปลี่ยนแปลงค่าของระดับของจุดภาพอย่างรวดเร็ว [6] ดังนั้นการแบ่งข้อมูลภาพออกเป็นบล็อกที่มีขนาดเท่าๆกัน จึงเป็นวิธีการแบ่งที่ไม่เหมาะสมกับการที่จะนำมาใช้แบ่งข้อมูลภาพที่มีลักษณะดังกล่าว ดังนั้นการเข้ารหัสแบบบล็อกได้มีการพัฒนา โดยแบ่งบล็อกให้มีหลายขนาดโดยพิจารณาแต่ละบล็อกด้วยค่าเทรชโฮล [7] ว่าบล็อกดังกล่าวควรที่จะแบ่งให้บล็อกมีขนาดเล็กลง หรือว่าบล็อกดังกล่าวนั้นไม่จำเป็นต้องแบ่งอีกต่อไป แต่วิธีการดังกล่าวนี้ยังคงต้องใช้ค่าเทรชโฮล ซึ่งการหาค่าเทรชโฮลที่เหมาะสมของแต่ละภาพนั้น เป็นเรื่องยากสำหรับค่าที่เหมาะสมเพื่อนำมาใช้ในการแบ่งภาพ เพราะค่าเทรชโฮลนั้นจะต้องเป็นค่าเทรชโฮลที่ใช้แบ่งบล็อกของทั้งภาพ ซึ่งค่าที่ใช้จะเป็นในลักษณะที่เป็นโกลบอล (global) ดังนั้นเมื่อมีการกำหนดค่าเทรชโฮลแต่ละครั้งนั้นถ้าค่าที่ใช้ไม่เหมาะสมกับการแบ่งจะทำให้จำนวนบล็อกมากหรือน้อยไป

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

ในการศึกษาวิจัยของการทำวิทยานิพนธ์นี้ ได้มีจุดมุ่งหมายเพื่อศึกษาวิธีการเข้ารหัสภาพและวิธีการบีบอัดข้อมูลภาพ โดยเริ่มศึกษาหลักการบีบอัดข้อมูลภาพเบื้องต้นรวมทั้งวิธีการประยุกต์ของการบีบอัดข้อมูล ทั้งนี้ผู้วิจัยได้ศึกษาถึงการลดขนาดข้อมูลภาพโดยได้ทำการแบ่งภาพถูกแบ่งออกเป็นบล็อกย่อยหลายๆขนาด โดยสังเกตว่าแต่ละบล็อกที่แบ่งออกมานั้นมีความเป็นเนื้อเดียวกันของข้อมูลหรือไม่แล้วถ้าข้อมูลที่แบ่งออกมาไม่มีความเป็นเนื้อเดียวกันนั้นจะทำการแบ่งต่อไปอย่างไร ทั้งนี้ได้พิจารณาผลกระทบที่เกิดขึ้นกับการแบ่งดังนั้นในวิทยานิพนธ์นี้จึงได้นำเสนอวิธีการที่จะทำการแยกองค์ ประกอบแบบควอดทรี [8] โดยใช้ค่าลากรานจ์ (Lagrange Cost) [9,10,11] จากรูปแบบของลากรานจ์มัลติพลายเออร์ (Lagrange Multiplier) ดังนั้นการแยกองค์ประกอบในแต่ละครั้งจึงมีการพิจารณาถึงผลที่จะเกิดขึ้นเมื่อได้ทำการแยกองค์ประกอบไป ถึงผลกระทบของค่าความผิดเพี้ยนและอัตราบิตทั้งหมดของภาพเพื่อให้ลดขนาดของข้อมูลภาพให้ได้มากที่สุด และทำให้ความผิดเพี้ยนน้อยหลังจากการสร้างกลับภาพ

1.3 สมมติฐานของการศึกษา

ในการบีบอัดข้อมูลภาพแบบแบ่งข้อมูลออกเป็นบล็อกมีด้วยกันหลายวิธี ในที่นี้ได้นำเสนอวิธีการเข้ารหัสแบบบล็อกที่แบ่งบล็อกออกเป็นหลายๆ ขนาด (Vaiable Block Truncation Code) [3, 7, 12, 13, 14, 15, 16, 17] เพื่อบีบอัดข้อมูลภาพต้นแบบให้ได้มากที่สุด อีกทั้งยังแก้ปัญหาของการแบ่งข้อมูลภาพที่มีขนาดเท่ากันทั้งหมด ซึ่งวิธีการดังกล่าว ถ้าข้อมูลภาพในส่วนที่มีการเปลี่ยนแปลงระดับสีมากภายในนั้นไม่มีความเป็นเนื้อเดียวกัน (non-homogeneous) ในการแบ่งบล็อกที่มีขนาดใหญ่เกินไปจะทำให้การสร้างกลับภาพมีคุณภาพด้อยลง (ซึ่งขนาดของบล็อกไม่ควรมีขนาดใหญ่กว่า 32x32 [13]) อีกทั้งวิธีการแบ่งบล็อกด้วยวิธีการของควอดทรีนั้นไม่ได้พิจารณาถึงอัตราบิต (bit rate) และค่าความผิดเพี้ยน (distortion) ดังนั้นควรพิจารณาโครงสร้างต้นไม้แบบควอดทรีที่มีการเติบโต (grow) ทีละน้อย [18] ซึ่งการเติบโตแต่ละครั้งนั้นจะพิจารณาถึงอัตราบิตและความผิดเพี้ยนเป็นหลักในการเลือกโหนดที่จะให้โครงสร้างแบบ ควอดทรีสามารถที่จะเติบโตได้ดีที่สุดนั่นหมายถึงว่าบล็อกที่ทำการแบ่งนอกจากจะมีความเป็นเนื้อเดียวกันของข้อมูลภายในบล็อกแล้วยังช่วยให้ประสิทธิภาพของสร้างกลับของภาพมีความผิดเพี้ยนน้อยอีกด้วย

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

ในทฤษฎีอัตราบิต-ความผิดเพี้ยน (Rate-Distortion Theory) เป็นทฤษฎีที่กล่าวถึงการจัดสรรบิตให้กับข้อมูลที่ไม่มีความสัมพันธ์กันหรือที่เรียกกว่าข้อมูลแบบไม่มีเมมโมรี (Memoryless source) ซึ่งในการกำหนดบิตให้กับข้อมูลมีความจำเป็นอย่างยิ่ง เพราะว่าในแต่ละบิตที่จัดสรรให้นั้นสามารถทำให้ความผิดเพี้ยนของข้อมูลจะเพิ่มขึ้นตามอัตราบิตเป็นค่าประมาณ 6.0206 dB ต่อบิต [19] ในกรณีข้อมูลแบบเกาส์เซียน (Gaussian source) นั่นหมายความว่าถ้าเราสามารถจัดสรรบิตที่เหมาะสมให้กับข้อมูลเราสามารถที่จะลดความผิดเพี้ยนลงได้

1.5 ขอบเขตของการวิจัย

ในการวิจัยผู้วิจัยได้ศึกษาเฉพาะการเข้ารหัส โดยที่ทำการแบ่งภาพออกเป็นบล็อกหลายขนาด (Variable Block Truncation Coding) ซึ่งเป็นวิธีการเข้ารหัสแบบที่มีการสูญเสียข้อมูลหลังจากที่ได้มีการสร้างกลับภาพ โดยศึกษาถึงวิธีการเข้ารหัสแบบบล็อกหลายวิธีที่มีประสิทธิภาพดีและเหมาะสมที่จะนำมาใช้ในการเข้ารหัส โดยในที่นี้ผู้วิจัยได้เลือกการเข้ารหัสแบบบล็อกโดยใช้ค่าสมบูรณ์โมเมนต์ร่วมกับการแยกองค์ประกอบควอดทรีแบบใหม่โดยใช้การหาค่าลากรานจ์เพื่อนำมาใช้ในการแบ่งบล็อกข้อมูลภาพและได้นำการแปลงโคซายน์ต็มหน่วยมาใช้ในการแปลงข้อมูลภาพในส่วนที่มีความละเอียดสูงเพื่อช่วยเพิ่มประสิทธิภาพในการสร้างกลับภาพ

1.6 ขั้นตอนของการศึกษา

ในการศึกษาวิจัยได้เริ่มการวิจัยออกเป็นขั้นตอนดังต่อไปนี้

1. ศึกษาวิธีการเข้ารหัสแบบแบ่งบล็อก
    1.1 แบบบล็อกขนาดคงที่
    1.2 แบบบล็อกหลายขนาด
2. ศึกษาวิธีการของลากรานจ์มัลติพลายเออร์กรณีที่นำมาประยุกต์ใช้กับทฤษฎีอัตราบิต-ความผิดเพี้ยน
3. การออกแบบโค๊ตบุคของแต่ละบล็อกภาพ
4. การออกแบบต้นไม้รหัสเพื่อใช้ในการสร้างกลับภาพ
6. การเปรียบเทียบผลที่ได้กับวิธีการอื่น

บทที่ 2
การบีบอัดข้อมูลภาพเบื้องต้น

การบีบอัดข้อมูลภาพคือการลดจำนวนของข้อมูลภาพต้นแบบ วิธีที่ง่ายที่สุดและรวดเร็วของการบีบอัดข้อมูลภาพคือการแซมเปิลภาพ โดยให้ตัวเลขมีจำนวนจำกัดของจุดภาพต่อหนึ่งพื้นที่ของการแซมเปิล ด้วยเหตุนี้จำนวนของการแซมเปิลต่อหนึ่งหน่วยพื้นที่จะถูกลดลง ในขบวนการประมวลผลภาพแต่ละจุดภาพ (pixel) ซึ่งจะถูกควอนไทซ์โดยแทนค่าด้วยตัวเลขจำนวนบิตที่จำกัดและจัดเก็บข้อมูล(หรือทำการส่ง)ในรูปแบบดิจิตอล ประโยชน์ของการบีบอัดข้อมูลภาพมีเพิ่มมากขึ้นสำหรับการเก็บข้อมูลและการส่งข้อมูลภาพเพื่อมุ่งเน้นถึงการทำให้หน่วยความจำและแบนด์วิดท์ของการส่งมีจำนวนน้อยโดยทั่วไปภาพที่ถูกลดข้อมูลเมื่อทำการถอดรหัสเพื่อสร้างภาพกลับมักจะเกิดความผิดเพี้ยนขึ้น (ในกรณีที่มีการสูญเสีย) ประสิทธิภาพของวิธีการบีบอัดภาพวัดได้จากความสามารถในการบีบอัด ความผิดเพี้ยนที่เกิดขึ้นและความยุ่งยากของวิธีการบีบอัด ความยุ่งยากของวิธีการลดข้อมูลมีความสำคัญที่ควรพิจารณาเพื่อใช้ในการปรับปรุงฮาร์ดแวร์

ในการแปลงสัญญาณภาพในรูปแบบของอนาล็อกไปเป็นสัญญาณดิจิตอล ซึ่งเป็นผลให้ความต้องการแบนด์วิดท์มีสูงขึ้นเพื่อใช้ในการส่งข้อมูล วิธีการลดข้อมูลพยายามเพื่อที่จะลดค่าใช้จ่ายส่วนนี้ลงและโอกาสที่ขบวนการมีความยืดหยุ่นเพื่อลดแบนด์วิดท์ของสัญญาณดิจิตอลให้มีค่าน้อยกว่าความต้องการแบนด์วิดท์ของสัญญาณอนาล็อกของสัญญาณดังกล่าว

ประโยชน์ของการลดข้อมูลดังที่กล่าวมาแล้วนั้นสำหรับทางด้านของการส่งข้อมูล ซึ่งวิธีการบีบอัดข้อมูลภาพจะเน้นถึงการทำงานในแบบเวลาจริง (real-time) และแบบออนไลน์ (on-line) ซึ่งมุ่งเน้นเพื่อจำกัดข้อมูลและความยุ่งยากในการทำฮาร์ดแวร์ สำหรับประโยชน์ทางด้านการเก็บข้อมูลนั้นจะไม่เน้นการทำงานที่เป็นแบบเวลาจริงและแบบออนไลน์ โดยสามารถทำงานในลักษณะออฟ-ไลน์(off-line) แต่อย่างไรก็ตามการถอดรหัสหรือการกู้ข้อมูลที่ถูกบีบอัดข้อมูลภาพควรจะมีความรวดเร็วและมีประโยชน์เพื่อให้ใช้เวลาน้อยลง ประโยชน์ของการส่งข้อมูลภาพถูกใช้ในหลายด้านเช่น การแพร่ภาพโทรทัศน์ การสำรวจระยะไกล (remote sensing) ระบบเรดาร์ระบบโซนาระบบการประชุมทางไกล การสื่อสารคอมพิวเตอร์ การส่งแฟกซ์ เป็นต้น ในส่วนของการเก็บข้อมูลภาพมีความต้องการอย่างมาก สำหรับการเก็บเอกสารทางด้านการศึกษาและธุรกิจและภาพทางการแพทย์ เป็นต้น

2.1 ทฤษฎีการควอนไทซ์ [10]

ในการประมวลผลด้วยคอมพิวเตอร์ โดยส่วนใหญ่ข้อมูลภาพจะอยู่ในรูปแบบของอะเรย์ (Array) และตำแหน่งของข้อมูลภาพแต่ละจุดภาพในระบบ 2 มิติจะเรียกว่าเพล (pel) หรือพิกเซล (picture element "pixel") หรือจุดภาพ ซึ่งตำแหน่งพิกัดจะบอกได้ด้วยฟังก์ชัน $f(x,y)$ โดยทำการดิจิไทซ์(การแซมเปิลกับการควอนไทซ์)ทั้งระนาบ (spatial) และขนาด (amplitude) การดิจิไทซ์ของพิกัดในระนาบ (x,y) ถูกเรียกว่าการแซมปลิ้ง (sampling) ภาพ และถ้าเป็นการทำดิจิไทซ์ทางขนาดจะเรียกว่าการควอนไทซ์ในระดับเทา (gray-level quantization) [20] โดยส่วนมากภาพสองมิติที่มีขนาด $N \times M$ จะแสดงดังสมการที่ 2.1

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,1) & & f(1,M-1) \\ \vdots & & & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix} \tag{2.1}$$

ในทางด้านขวามือของสมการ 2.1 ปกติจะเรียกว่าภาพดิจิตอล (digital image) ความต้องการเบื้องต้นสำหรับการประมวลภาพด้วยคอมพิวเตอร์แสดงดังรูปที่ 2.1 นั่นคือภาพควรจะอยู่ในรูปแบบดิจิตอล การประมวลผลภาพเริ่มจากภาพถูกแซมเปิลหรือการสุ่มสัญญาณให้อยู่ในรูปแบบจุดภาพในแต่ละแซมเปิลหรือจุดภาพจะถูกควอนไทซ์โดยใช้จำนวนบิตที่จำกัด ภาพที่ถูกควอนไทซ์แล้วสามารถประมวลผลได้ด้วยคอมพิวเตอร์ เพื่อที่จะแสดงผลภาพในรูปแบบที่เป็นดิจิตอลนั้น อันดับแรกภาพจะถูกแปลงไปเป็นสัญญาณอนาลอก ซึ่งภาพที่แสดงบนจอภาพนั้นได้จากการสแกนของหลอดภาพ

รูปที่ 2.1 การประมวลภาพด้วยคอมพิวเตอร์

การควอนไทซ์มีอยู่สองแบบด้วยกันคือการควอนไทซ์แบบสเกลาร์ ซึ่งทำการควอนไทซ์ข้อมูลที่ทำการสุ่มโดยการทดลองตัวอย่าง ส่วนการควอนไทซ์แบบเวกเตอร์จะทำการควอนไทซ์

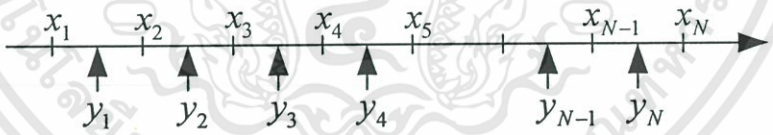
บล็อกของข้อมูลโดยใช้บล็อก (ที่มีความยาวบล็อกเท่าๆกัน) การควอนไทซ์แบบสเกลาร์มีข้อดีตรงที่มีความง่าย แต่ความผิดเพี้ยนมากกว่าการควอนไทซ์แบบเวกเตอร์ที่มีความยุ่งยากมากกว่า

2.2 การควอนไทซ์แบบสเกลาร์ที่ใช้ระดับคงที่และไม่คงที่ (Uniform and Non-Uniform Scalar Quantization)

เวกเตอร์ข้อมูลที่จะถูกลดคือ $X=(X_1,X_2,\ldots,X_n)$ สมมติว่าข้อมูลตัวอย่าง x_i เป็นค่าจริงซึ่งในทางปฏิบัติมีความเป็นไปได้หลายค่าสำหรับเวกเตอร์ข้อมูล X วัตถุประสงค์ของการควอนไทซ์คือเพื่อเตรียมพร้อมข้อมูลไว้สำหรับการเข้ารหัสของเวกเตอร์ $\hat{X}=(\hat{X}_1,\hat{X}_2,\ldots,\hat{X}_n)$ ซึ่งเวกเตอร์นี้ได้มากจากเซตจำกัดของเวกเตอร์ C ที่มีขนาดไม่เกิน 2^{nR} เมื่อ R คืออัตราบิตในการบีบอัดข้อมูลภาพที่พิจารณาหรือที่กำหนดให้ในหน่วยบิตต่อแซมเปิลเซต C ถูกเรียกว่าโค๊ตบุค (codebook) ของการควอนไทซ์ และความผิดเพี้ยนในการควอนไทซ์ถูกกำหนดโดย

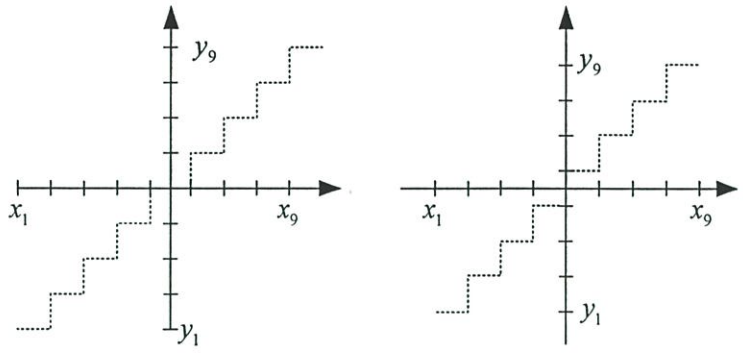
$$D \triangleq n^{-1}\left[\left(X_1-\hat{X}_1\right)^2+\left(X_2-\hat{X}_2\right)^2+\ldots+\left(X_n-\hat{X}_n\right)^2\right] \qquad (2.2)$$

ถ้าเราแสดงข้อมูลในรูปเส้นจำนวนจริงดังรูปที่ 2.2 โดยให้แกน x จะแสดงถึงค่าอินพุทและแกน

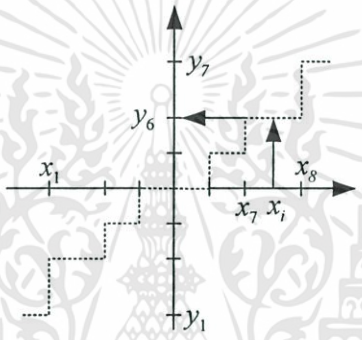


รูปที่ 2.2 การแบ่งเส้นจำนวนจริงออกเป็นช่วงๆในการควอนไทซ์ข้อมูล

y แสดงถึงค่าเอาท์พุทที่มีค่าไม่ต่อเนื่อง (discrete) เส้นแบ่งแต่ละค่าบนแกน x คือระดับการตัดสินใจ ส่วนบนแกน y คือค่าเอาท์พุท ในรูปที่ 2.3 นี้เป็นการควอนไทซ์แบบจัดระดับคงที่หรือการควอนไทซ์แบบลิเนียร์ (linear) และในรูปที่ 2.4 เป็นการควอนไทซ์แบบที่ระดับไม่คงที่หรือการควอนไทซ์แบบที่ไม่ลิเนียร์ (non-linear)



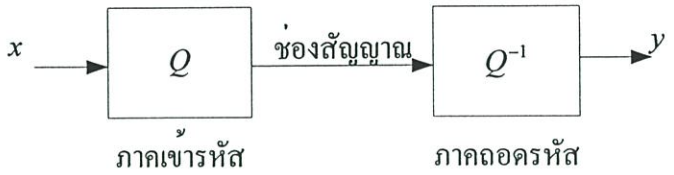
รูปที่ 2.3 การควอนไทซ์แบบจัดระดับคงที่



รูปที่ 2.4 การควอนไทซ์แบบที่ระดับไม่คงที่

ในรูปที่ 2.4 เป็นการควอนไทซ์แบบที่ระดับไม่คงที่(non-linear)สมมติว่าการควอนไทซ์ในรูปที่ 2.4 มีอินพุทคือ x_i ซึ่งอินพุทค่าดังกล่าวอยู่ในช่วงระหว่าง x_7 และ x_8 ดังนั้นเอาท์พุทคือ y_6

ในทางอุดมคติค่าเอาท์พุทที่ถูกกำหนดไว้ให้กับเซลล์จะเป็นค่าที่ดีที่สุด (optimal) สำหรับเซลล์นั้นๆแล้ว ค่าที่ดีที่สุดของเซลล์ถูกเรียกว่าเซนทรอยด์ (centriod) [12] ของมัน เซนทรอยด์ถูกกำหนดเป็นค่าเฉลี่ยของอินพุททั้งหมดซึ่งตกอยู่ในช่วงเซลล์ การนิยามดังกล่าวนี้จำเป็นอย่างยิ่งที่ว่าเซนทรอยด์นั้นจะต้องวางเรียงอยู่ภายในขอบเขตของเซลล์ ถ้าทราบถึงเส้นแบ่งเซลล์และการกระจายความน่าจะเป็นของอินพุท เซนทรอยด์สามารถกำหนดได้อย่างถูกต้อง สำหรับกรณีพิเศษของการกระจายแบบยูนิฟอร์ม (uniform distribution) เซนทรอยด์จะมีตำแหน่งที่อยู่ตรงกลาง Euclidean ของเซลล์ เพราะว่ามีค่าเอาท์พุทที่เป็นไปได้มีจำนวนจำกัดและค่าเอาท์พุทนั้นไม่จำเป็นต้องส่งไปยังภาครับ โดยค่าเอาท์พุทแต่ละค่าจะถูกกำหนดดัชนีให้และดัชนีของเอาท์พุทจะถูกส่งไปแทนที่จะส่งค่าเอาท์พุทนั้นไปโดยตรง ที่ทางด้านรับดัชนีที่รับมาได้สามารถเลือกหรือจับคู่กับค่าเอาท์พุทที่เหมาะสมโดยดูจากตารางที่เก็บข้อมูลไว้ รูปการเข้ารหัสและการถอดรหัสแสดงดังรูปที่ 2.5



รูปที่ 2.5 การเข้ารหัสและการถอดรหัสสัญญาณ

การออกแบบการควอนไทซ์แบบสเกลาร์

การควอนไทซ์ที่มีประสิทธิภาพดีควรมีคุณสมบัติดังต่อไปนี้

ความยุ่งยากในการอธิบายหรือแจกแจงส่วนต่างๆน้อย : การควอนไทซ์ควรที่จะมีความง่ายในการบ่งบอกส่วนต่างๆ ดังนั้นส่วนหัว (overhead) จึงเป็นส่วนเพื่อบ่งชี้การควอนไทซ์สำหรับบอกทางด้านถอดรหัสซึ่งควรจะมีค่าน้อยกว่า nR บิต

ความยุ่งยากในการคำนวณน้อย : กฎของการควอนไทซ์ควรที่จะสามารถยอมรับการพัฒนาที่เกิดขึ้นภายหลังที่มีความเร็วมากขึ้นได้

ความผิดเพี้ยนต่ำ : ในทางอุดมคติความผิดเพี้ยนที่เกิดจากการควอนไทซ์ควรจะมีความผิดพลาด (arising) ต่ำที่สุดจากการควอนไทซ์แบบอื่นที่มีความยุ่งยากเท่าๆกัน นั่นหมายถึงที่อัตราบิตที่เท่ากันด้วย การออกแบบการควอนไทซ์ที่มีประสิทธิภาพและมีความผิดเพี้ยนน้อยได้กล่าวไว้ใน [21]

2.3 เวกเตอร์ควอนไทซ์ [10, 22]

เวกเตอร์ควอนไทซ์เป็นวิธีที่คล้ายกับการสเกลาร์ควอนไทซ์โดยมีมิติที่มากกว่า เวกเตอร์ควอนไทซ์โดยส่วนมากถูกนำมาใช้กับสัญญาณที่ได้ถูกทำการดิจิไทซ์เรียบร้อยแล้ว เวกเตอร์ควอนไทซ์เป็นวิธีการที่เริ่มต้นนำมาใช้กับการลดข้อมูลและการจดจำลักษณะรูปแบบ (pattern recognition) เวกเตอร์ ควอนไทซ์ Q ที่มีมิติเท่ากับ k และมีขนาดเท่ากับ N นิยามว่าเป็นการจับคู่จากเวกเตอร์ในปริภูมิยูคริดีน(Euclidean space) ที่มี k มิติ R^k ไปเป็นเซตจำกัดของเวกเตอร์หรือคำรหัส (codeword) นั้นคือ

$$Q : R^k \rightarrow C \tag{2.3}$$

เมื่อ $C = (y_1, y_2, \ldots, y_N)$ เป็นโค๊ตบุคโดยคำรหัส $y_i \in R^k$ สำหรับ $i \in \{1,2,\ldots,N\}$ อัตราบิตของเวกเตอร์ควอนไทซ์สามารถนิยามได้ดังนี้

$$r = \frac{(\log_2 N)}{k} \tag{2.4}$$

จากคำนิยามของเวกเตอร์ควอนไทซ์คือการจับคู่จากปริภูมิ R^k ไปเป็นโค๊ตบุค C ที่มีขนาด N ซึ่งเสมือนว่าปริภูมิ R^k นั้นถูกแบ่งออกเป็น N ส่วนหรือเรียกว่าเซลล์(cell)ส่วนต่างๆ กำหนดให้เป็น R_i ซึ่งก็คือ

$$R_i = \{x \in R^k : Q(x) = y_i\} \tag{2.5}$$

ซึ่งมีลักษณะเช่นเดียวกับเซลล์หรือระดับการตัดสินใจของวิธีการสเกลาร์ควอนไทซ์ R_i บางครั้งถูกเรียกว่าภาพที่ตรงกันข้าม (Inverse image) ของ y_i

$$R_i = Q^{-1}(y_i) \tag{2.6}$$

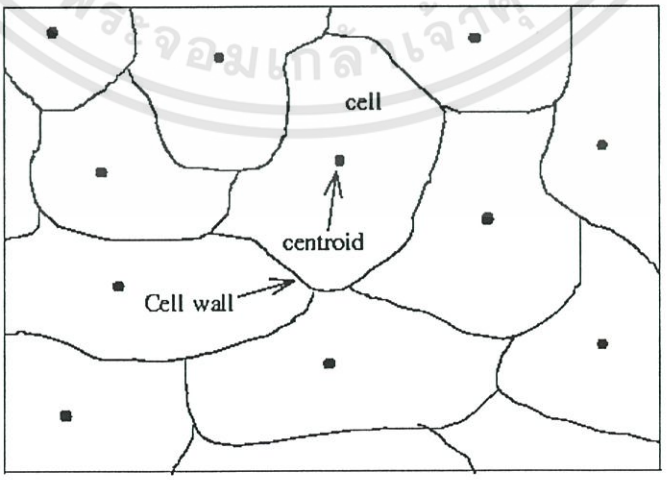
แล้ว

$$\bigcup_i R_i = R^k \tag{2.7}$$

และ

$$R_i \cap R_j = 0 \quad \text{สำหรับ } i \neq j$$

ดังนั้นเซลล์คือการแบ่งส่วนของ R^k ในรูปที่ 2.6 แสดงรูปของเซลล์และเซนทรอยด์ของส่วนของเวกเตอร์ควอนไทซ์สองมิติ ซึ่งเซนทรอยด์นั้นแสดงด้วยจุดสีดำ



รูปที่2.6 เซลล์ของเวกเตอร์ควอนไทซ์

เวกเตอร์ควอนไทซ์ทางด้านภาคเข้ารหัส E จะทำการจับคู่อินพุท R^k ไปเป็นเซต I ที่มีดัชนี

$$E : R^k \rightarrow I \tag{2.8}$$

ผลที่ได้ก็คือภาคเข้ารหัสไม่จำเป็นต้องรู้โค๊ตบุค เพียงแต่ทำการแบ่งเซลล์ของปริภูมิ R^k ส่วนทางด้านถอดรหัส D จะทำการจับคู่เซตของดัชนี I ไปเป็นเซตของการสร้างกลับคืนข้อมูล C เขียนเป็นสมการได้ดังนี้

$$D : I \rightarrow C \tag{2.9}$$

การถอดรหัสทำได้โดยมีตารางไว้เปรียบเทียบ (lookup table) ซึ่งภาครับไม่จำเป็นต้องทราบสิ่งต่างๆในการแบ่งปริภูมิ R^k เลย เวกเตอร์ควอนไทซ์สามารถมองได้ว่าเป็นการต่อกันระหว่างภาคการเข้ารหัสและภาคถอดรหัสหรือที่เรียกกันว่า codec ในรูปที่ 2.5 แสดง codec ของเวกเตอร์ควอนไทซ์ ซึ่งในความเป็นจริงจะเหมือนกับว่าเป็นสเกลาร์ควอนไทซ์ที่เป็นกรณีพิเศษ

โวโรนอย (Voronoi) หรือพื้นที่ข้างเคียงที่สุดในเวกเตอร์ควอนไทซ์ นั้นเป็นกรณีพิเศษของเวกเตอร์ควอนไทซ์ซึ่งการแบ่งปริภูมิถูกกำหนดได้สมบูรณ์แบบโดยโค๊ตบุคและการวัดความผิดเพี้ยนในเวกเตอร์ควอนไทซ์ ในความเป็นจริงพื้นที่ข้างเคียงที่สุดของเวกเตอร์ควอนไทซ์เป็นรูปแบบธรรมดามากของเวกเตอร์ควอนไทซ์ในทางปฏิบัติ การวัดความผิดเพี้ยนนั้นเป็นที่นิยมอย่างสูงและนำมาใช้ในการวัดพื้นที่ข้างเคียงที่สุดของเวกเตอร์ควอนไทซ์การวัดดังกล่าวคือการวัดความผิดพลาดกำลังสองซึ่งถูกนิยามว่าคือระยะทางยูคลิเดียน (Euclidean distance) ระหว่างเวกเตอร์นั่นเอง

$$d(x, y) = \sum_{i=1}^{k} (x_i - y_i)^2 \tag{2.10}$$

ซึ่งการวัดความผิดเพี้ยนแบบอื่นๆก็สามารถนำมาใช้ได้เช่นกัน การแบ่งเซลล์สำหรับพื้นที่ข้างเคียงที่สุดในเวกเตอร์ควอนไทซ์นิยามโดย

$$R_i = \{x : d(x, y_i) \leq d(x, y_j) \quad all \quad j \in I\} \tag{2.11}$$

พื้นที่ข้างเคียงที่สุดนั้นสามารถทำได้ตามวิธีการที่จะได้อธิบายต่อไปนี้ สำหรับโค๊ตบุคที่มีขนาด N

วิธีการหาพื้นที่ข้างเคียงที่สุด

ขั้นที่ 1 กำหนดให้ $d = d_{initial}, j = 1, i = 1.$

ขั้นที่ 2 ทำการคำนวณ $D_j = d(x, y_i)$

ขั้นที่ 3 ถ้า $D_j < d, d = D_j, i = j.$

ขั้นที่ 4 ถ้า $j < N, j++,$ ไปยังขั้นที่ 2.

ขั้นที่ 5 จบการทำงาน

เมื่อ $d_{initial}$ มักจะมีค่ามากกว่าความผิดเพี้ยนที่คาดไว้และ i คือดัชนีของคำรหัสซึ่งยอมให้มีค่าความผิดเพี้ยนน้อยที่สุดซึ่งมีค่าเท่ากับค่าสุดท้ายของ d เน้นว่าการเข้ารหัสไม่ต้องการอธิบายที่เป็นส่วนของเลขาคณิตของส่วนต่างๆแต่เพียงต้องการโค๊ตบุคและวัดค่าความผิดเพี้ยน

2.4 การควอนไทซ์โดยการเก็บรักษาโมเมนต์ (Moment Preserving Quantization)

เนื่องจากวิธีการของ PCM (Pulse Code Modulation) เป็นวิธีการออกแบบการควอนไทซ์ที่นิยมใช้กันมาก ซึ่งการควอนไทซ์แบบดังกล่าวได้ถูกออกแบบเพื่อใช้สำหรับข้อมูลอินพุทที่มีฟังก์ชั่นความหนาแน่นของความน่าจะเป็น (Probability Density Function) เฉพาะอย่างของรูปแบบอินพุท โดยที่มีสัมพันธ์กับดัชนี (index) หรือความระเอียดสูง มาตรฐานของบรรทัดฐานซึ่งใช้โดยที่ค่าความผิดพลาดกำลังสอง (Mean Square Error) ระหว่างอินพุทและเอาท์พุทซึ่งการออกแบบการควอนไทซ์เพื่อทำให้ค่าความผิดพลาดกำลังสองต่ำลงหรือเพื่อจุดประสงค์อื่นคือในการวัดค่าความผิดพลาดกำลังสอง

วิธีการเข้ารหัสเช่น BTC ใช้ระดับการควอนไทซ์ที่ใช้ระดับเพียงไม่กี่ระดับและใช้รูปแบบที่เป็นนอนพาราเมทริก (non-parametric) ของการควอนไทซ์แบบที่มีการเก็บรักษาโมเมนต์ (Moment preserving Quantization) [2] ซึ่งในที่นี้นอนพาราเมทริกหมายถึงการควอนไทซ์ที่ถูกออกแบบให้กระชับกับข้อมูลจริง

การควอนไทซ์แบบที่มีการเก็บรักษาโมเมนต์นั้นกำหนดให้ตัวแปรสุ่ม X หมายถึงอินพุทที่เข้ามายังภาคควอนไทซ์ซึ่งมีฟังก์ชั่นการกระจายคือ $F(x), x \in [a,b]$ ซึ่ง $[a,b]$ เป็นจำนวนจำกัดไม่จำกัด หรือกึ่งไม่จำกัด ให้ Y หมายถึงตัวแปรสุ่มที่เอาท์พุทของภาคควอนไทซ์ สำหรับการควอนไทซ์ที่มีเพียงสองระดับ ตัวแปรสุ่ม Y นั้นเป็นค่าที่ไม่ต่อเนื่อง (discrete) และเกิดขึ้นมีค่าเป็น $\{y_1, y_2\}$ โดยที่มีความน่าจะเป็น $P_1 = \Pr(Y = y_1)$ และ $P_2 = \Pr(Y = y_2)$ เอาท์พุท Y เกิดขึ้นมีค่าเป็น y_1 ก็ต่อเมื่ออินพุท X นั้นมีค่าต่ำกว่าค่าเทรชโฮลบางค่าของ x_1 หรือไม่เช่นนั้นเอาท์พุทคือ y_2 ดังนั้นการออกแบบโดยทั่วไปเพื่อการควอนไทซ์สองระดับ สิ่งหนึ่งที่ต้องพิจารณาในการ

เลือกระดับเอาท์พุท y_1 และ y_2 และค่าเทรชโฮล x_1 เมื่อใช้บรรทัดฐานของการออกแบบเพื่อที่จะให้การควอนไทซ์สองระดับเก็บรักษาค่าโมเมนต์ของข้อมูลอินพุทจำเป็นอย่างยิ่งที่ภาคควอนไทซ์ จะต้องเก็บรักษาโมเมนต์ทั้งหมดสามโมเมนต์แรกของอินพุทหรืออีกอย่างหนึ่งคือจะต้องทราบค่าหนึ่งในสามของพารามิเตอร์ในที่นี้สามารถกำหนดสามสมการเพื่อเก็บรักษาค่าโมเมนต์ได้ดังนี้

$$\begin{aligned} E[Y] &= E[X] = y_1P_1 + y_2P_2 \\ E[Y^2] &= E[X^2] = y_1^2P_1 + y_2^2P_2 \\ E[Y^3] &= E[X^3] = y_1^3P_1 + y_2^3P_2 \end{aligned} \tag{2.12}$$

$$P_1 + P_2 = 1$$

เมื่อ $E[\bullet]$ ถูกกำหนดได้โดยการอินทิเกรตในรูปแบบของเลเบสคิว-สไทล์จี (Lebesque-Stieltjes)

$$E[X^i] = \int_a^b x^i dF(x), \text{ โดยที่ } y_1 \le x_1 \le y_2$$

เราสมมติว่าโมเมนต์นั้นมีค่าอยู่จริงและเป็นจำนวนจำกัดจาก (2.12) สามารถเขียนใหม่ได้เป็น

$$\begin{aligned} m_1 &= y_1F(x_1) + y_2(1 - F(x_1)) \\ m_2 &= y_1^2F(x_1) + y_2^2(1 - F(x_1)) \\ m_3 &= y_1^3F(x_1) + y_2^3(1 - F(x_1)) \end{aligned} \tag{2.13}$$

เมื่อ $m_i = E[X^i]$

$$\begin{aligned} P_1 &= Prob(X \le x_1) = F(x_1) \\ P_2 &= Prob(X > x_1) = 1 - F(x_1) \end{aligned}$$

โดยการพิสูจน์สมการที่ (2.13) เพื่อหาค่า y_1, y_2 และ x_1 การควอนไทซ์ที่ได้คือสามโมเมนต์แรกของ X ซึ่งจะเหมือนกันกับของ Y เพื่อการหา x_1 จะสมมติให้ค่า $F^{-1}(\cdot)$ นั้นมีอยู่จริง

ในกรณีที่ไม่มีการสูญเสียเราจะสมมติให้ $m_1 = 0$ และ $m_2 = 1$ (คือ X มีค่าเฉลี่ยเป็นศูนย์และมีค่าความแปรปรวนเป็นหนึ่ง) ดังนั้นสมการที่ (2.13) เขียนใหม่ได้เป็น

$$\begin{aligned} 0 &= y_1F(x_1) + y_2(1 - F(x_1)) \\ 1 &= y_1^2F(x_1) + y_2^2(1 - F(x_1)) \\ m_3 &= y_1^3F(x_1) + y_2^3(1 - F(x_1)) \end{aligned} \tag{2.14}$$

โดยการหาค่า y_1, y_2 จากสองสมการแรกใน (2.14) ในเทอมของ $F(x_1)$ และใช้คำตอบที่ได้แทนลงในสมการที่สามซึ่งจะได้ดังนี้

$$y_1 = -\sqrt{\frac{1-F(x_1)}{F(x_1)}} = -\sqrt{\frac{P_2}{P_1}}$$
$$y_2 = \sqrt{\frac{F(x_1)}{1-F(x_1)}} = \sqrt{\frac{P_1}{P_2}} \qquad (2.15)$$
$$F(x_1) = \frac{1}{2} - \frac{m_3}{2}\sqrt{\frac{1}{4+m_3^2}}$$

จากสมการที่ (2.15) แสดงให้เห็นว่าค่าเทรชโฮล x_1 โดยทั่วไปแล้วเป็นค่ามัธยฐาน (median) ของ X และไม่ได้เป็นค่าเฉลี่ย และ m_3 โดยทั่วไปแล้วหมายถึงตัวเลขสัญลักษณ์ (signed number) และสามารถแปรความหมายเป็นการวัดค่าโมเมนต์อันดับที่สาม (skewness) ในฟังก์ชันของความหนาแน่นได้ และผลที่ได้จากสมการที่ (2.15) แสดงให้เห็นว่าค่าเทรชโฮลจะมีอิทธิพลกับค่าที่อยู่เหนือหรือต่ำกว่าค่ามัธยฐาน (Median) ที่เป็นไปตามเครื่องหมายและขนาดของค่าโมเมนต์อันดับที่สามนี้

2.5 อัตราข่าวสาร (Information rate)

อัตราข้อมูลภาพที่เป็นข้อมูลดิบไม่ได้เป็นสิ่งบ่งบอกถึงอัตราข่าวสารเฉลี่ยของมัน สำหรับแหล่งข้อมูลซึ่งมีสัญลักษณ์ที่แตกต่างกันที่เป็นไปได้ L สัญลักษณ์โดยมีค่าความน่าจะเป็น p_i โดยที่ $i = 0,\ldots,L-1$ โดยจะมีเอ็นโทรปีดังนี้

$$H = -\sum_{i=0}^{L-1} p_i \log_2 p_i \qquad (2.16)$$

โดยที่เอ็นโทรปีจะหมายถึงปริมาณข่าวสารเฉลี่ยที่แหล่งกำเนิดข่าวสารให้กำเนิดอยู่ [23] เนื่อง จากว่า H เป็นค่าคาดคะเนของปริมาณข่าวสารที่ส่งออกจากแหล่งกำเนิดข่าวสารแต่ละครั้งหรือแต่ละสัญลักษณ์และมีหน่วยเป็นบิตต่อสัญลักษณ์

ด้วยทฤษฎีการเข้ารหัสที่ไม่มีการรบกวนของแชนแนล ข้อมูลข่าวสารสามารถเข้ารหัสโดยปราศจากความผิดเพี้ยน แหล่งข้อมูลของเอ็นโทรปี H บิตต่อซิมบอล เปลี่ยนเป็น $H+\varepsilon$ บิตต่อซิมบอล เมื่อ ε คือปริมาณค่าบวกที่มีจำนวนน้อยๆแล้วอัตราการบีบอัดข้อมูลภาพที่เป็นไปสูงสุด C นิยามได้โดย

$$C = \frac{\text{อัตราเฉลี่ยของข้อมูลภาพต้นแบบ(B)}}{\text{อัตราเฉลี่ยของข้อมูลที่ถูกเข้ารหัส}(H+\varepsilon)} \tag{2.17}$$

โดย $$\frac{B}{(H+\varepsilon)} \cong \frac{B}{H}$$

ถ้าแหล่งกำเนิดข่าวสารที่พิจารณาอยู่นี้ให้กำเนิดสัญลักษณ์เป็นจำนวน r สัญลักษณ์ต่อวินาที แหล่งกำเนิดข่าวสารนั้นจะกำเนิดข่าวสารเป็นปริมาณข่าวสารเฉลี่ยต่อหนึ่งวินาทีเป็นดังนี้

$$R_i \triangleq rH \qquad \text{บิตต่อวินาที} \tag{2.18}$$

โดย R_i นี้เรียกว่า อัตราข่าวสาร (information rate)

การเข้ารหัสข่าวสารที่ดีที่สุดนั้นจะต้องสามารถทำการเข้ารหัสข่าวสารโดยที่อัตราข้อมูลของสัญญาณไบนารีที่สร้างขึ้นมีขนาดเท่ากับอัตราข่าวสารที่เข้ามา

2.5.1 ฟังก์ชันความผิดเพี้ยน- อัตราบิต (Distortion-Rate function)[11, 19, 24]

การเข้ารหัสแบบที่ไม่มีเมมโมรี (Memoryless coding) หรือข้อมูลที่นำมาเข้ารหัสไม่มีความเกี่ยวข้องกัน [23] นั้นมีการผลิตเอาท์พุทลำดับของตัวแปรสุ่มที่มีการกระจายแบบอิสระจากกันตามลักษณะของฟังก์ชันความหนาแน่นความน่าจะเป็นบางอย่าง ในที่นี้เราสนใจในแหล่งข้อมูลแบบไม่มีเมมโมรี่ สมมติว่าเราต้องการเข้ารหัสแบบที่มีการสูญเสียข้อมูล (lossless encoding)โดยมีข้อมูลเป็นรูปแบบของเวกเตอร์ที่กำเนิดจากข้อมูลที่ไม่มีเมมโมรี โดยที่อัตราการบีบอัดข้อมูลมีค่าไม่มากไปกว่า R บิตต่อหนึ่งหน่วยข้อมูล ดังนั้นเราจะดูว่าความผิดเพี้ยนหลังจากที่ทำการสร้างกลับข้อมูลแล้วจะมีค่าเป็นอย่างไร

2.5.2 คำนิยามของความผิดเพี้ยน $D(R)$

ให้ $f(x)$ เป็นฟังก์ชันความหนาแน่นของความน่าจะเป็น ซึ่งข้อมูลที่ไม่มีเมมโมรี (memoryless source) เป็นแหล่งกำเนิดข้อมูล โดยที่เป็นค่าคงที่จำนวนจริงบวก k การเข้ารหัสตัวแปรสุ่มจะก่อให้เกิดข้อมูลเวกเตอร์ของความยาวที่มีค่าเป็นจำนวนผลคูณของค่า k โดยใช้การควอนไทซ์แบบเวกเตอร์ที่มีมิติเท่ากับ k ของอัตราการบีบอัดข้อมูลภาพที่มีค่า $\leq R$ ซึ่งจะได้ความผิดเพี้ยนที่น้อยที่สุดสำหรับข้อมูลเวกเตอร์ดังกล่าวนั้น โดยกฎของจำนวนที่มีค่ามากๆนั้นความผิดเพี้ยนที่น้อยที่สุดนี้จะประมาณได้ว่าใกล้เคียงกับค่าที่จำกัด $D_k(R)$ โดยความยาวของ

ข้อมูลเวกเตอร์มีค่าไปสู่ค่าอนันต์ ด้วยค่าความน่าจะเป็นเท่ากับหนึ่ง ค่าจำกัด $D_k(R)$ เป็นฟังก์ชันที่มีอยู่เพียงฟังก์ชันความหนาแน่น $f(x)$ เท่านั้นคือ

$$D_k(R) = k^{-1} \min_{Q} \left[\iint \cdots \int_{R^k} \|X - Q(X)\|^2 f_k(x) dx \right] \tag{2.19}$$

เมื่อ R^k คือปริภูมิยูคลิเดียน (Euclidean Space) , $\|\cdot\|$ หมายถึงค่านอร์มของยูคลิเดียน (Euclidean norm), $f(x)$ คือผลคูณความหนาแน่น (Density product)

$$f_k(x) = f_k(x_1, x_2, \ldots, x_k) = f(x_1) f(x_2) \ldots f(x_k) = \prod_{n=1}^{k} f(x_i) \tag{2.20}$$

และค่าที่น้อยที่สุดได้มาจากการควอนไทซ์แบบเวกเตอร์ Q ที่มี k มิติ ซึ่งเซตของเวกเตอร์รหัส $Q(R^k)$ มีขนาดเป็น 2^{kR}

ฟังก์ชันความผิดเพี้ยน-อัตราบิต $D(R)$ สำหรับแหล่งข้อมูลที่ไม่มีเมมโมรี ถูกกำหนดได้เป็น

$$D(R) \triangleq \inf_{k} D_k(R) \tag{2.21}$$

สำหรับ $R > 0$ ในสมการที่ (2.21) ไม่สามารถที่จะคำนวณค่า $D(R)$ ได้ ดังนั้นจึงมีการพิสูจน์หาค่าของ $D(R)$ โดยที่

$$D(R) = \min_{g(y|x)} \left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - y)^2 f(x) g(y \mid x) dx dy \right] \tag{2.22}$$

เมื่อค่าที่ต่ำที่สุดเป็นค่าความหนาแน่นที่มีเงื่อนไข $g(y \mid x)$ ทั้งหมดนี้ซึ่ง

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \log \left[\frac{g(y \mid x)}{\int_{-\infty}^{\infty} f(u) g(y \mid u) du} \right] f(x) g(y \mid x) dx dy \le R \tag{2.23}$$

การทำให้มีค่าความผิดเพี้ยนน้อยที่สุด สามารถอธิบายได้กะทัดรัดกว่าโดยใช้แนวความคิดที่เรียกว่าปริมาณข่าวสารร่วมเฉลี่ย (mutual information) ให้ X, Y เป็นตัวแปรสุ่มซึ่ง $f(x)$ คือ

ความหนาแน่นของข้อมูล X และ $g(y|x)$ คือความหนาแน่นที่มีเงื่อนไขสำหรับ Y ที่ทำให้ $X=x$ แล้วข้อมูลปริมาณข่าวสารร่วมระหว่าง X และ Y คือปริมาณ $I(X;Y)$ สามารถนิยามได้ดังนี้

$$I(X;Y)\triangleq\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\log_2\left[\frac{g(y|x)}{\int_{-\infty}^{\infty}f(u)g(y|u)du}\right]f(x)g(y|x) \tag{2.24}$$

สิ่งหนึ่งสามารถพิจารณา ปริมาณข่าวสารร่ วมเฉลี่ย $I(X;Y)$ เป็นการวัดค่าของตัวแปรสุ่ม 2 ตัวคือ X และ Y ว่ามีความสัมพันธ์ซึ่งกันและกันอย่างไร ถ้า X และ Y อิสระจากกันทางสถิติ แล้ว $I(X;Y)=0$ แต่ถ้า X และ Y มีความสัมพันธ์กันแล้ว $I(X;Y)>0$ ปริมาณข่าวสารร่วมเฉลี่ย $I(X;Y)$ สามารถแสดงได้ว่าค่าของมันที่มีขนาดมากที่สุดเท่าที่จะเป็นไปได้เมื่อ X เป็นฟังก์ชันของ Y ใช้แนวคิดดังกล่าวของ ปริมาณข่าวสารร่วมเฉลี่ย $I(X;Y)$ สามารถเขียนสมการที่ (2.22) ใหม่ได้ดังนี้

$$D(R)=\min\left\{E[X-Y]^2 : I(X;Y)\le R\right\} \tag{2.25}$$

หรือสามารถอธิบายได้อีกอย่างหนึ่งได้ว่าถ้าค่าตัวแปรสุ่ม X เป็นค่าคงที่ที่มีการกระจายตามลักษณะความหนาแน่น $f(x)$ และมีค่าน้อยที่สุด ค่าของความผิดเพี้ยนที่คาดไว้ $E[(X-Y)^2]$ บนตัวแปรสุ่ม Y ทั้งหมดที่มีการกระจายโดยที่มีความสัมพันธ์กับ X ซึ่ง ปริมาณข่าวสารร่วมเฉลี่ย $I(X;Y)$ เป็นค่าที่ไม่มากกว่าค่าของ R ตัวแปรสุ่ม Y แต่ละตัวที่มีความสัมพันธ์กับ X สามารถอธิบายได้ในเทอมของความหนาแน่นที่มีเงื่อนไข $g(y|x)$ ซึ่งนิยามถึง "ช่องสัญญาณทดสอบ" ตัวแปรสุ่ม Y สามารถนำมาคิดเป็นเอาท์พุทจากช่องสัญญาณทดสอบเมื่ออินพุทคือ X เราสามารถมอง $D(R)$ ได้ว่าเป็นความผิดเพี้ยนกำลังสองที่คาดเดาว่ามีค่าน้อยที่สุดระหว่างอินพุทของช่องสัญญาณกับเอาท์พุทของช่องสัญญาณทั้งหมดของช่องสัญญาณทดสอบซึ่ง X เป็นอินพุทและ ปริมาณข่าวสารร่วมเฉลี่ย ระหว่างอินพุทและเอาท์พุทมีค่าไม่มากไปกว่าค่าของ R

2.5.3 อัตราความผิดเพี้ยนของข้อมูลแบบเกาส์เซียน

เราสามารถคำนวณความผิดเพี้ยน $D(R)$ สำหรับข้อมูลที่ไม่มีเมมโมรีและมีการกระจายแบบเกาส์เซียน ซึ่งแหล่งข้อมูลแต่ละตัวมีการกระจายในรูปแบบความหนาแน่นแบบเกาส์เซียนด้วยค่าเฉลี่ยมีค่าเป็นศูนย์ และความแปรปรวนมีค่าเป็น σ^2 แล้วจะได้ว่า

$$D(R)=\sigma^2 2^{-2R} \tag{2.26}$$

ให้ $R(D)$ เป็นส่วนกลับของ $D(R)$ ดังนั้นฟังก์ชันอัตราบิต-ความผิดเพี้ยนจากสมการที่ (2.26) แสดงให้เห็นว่า

$$R(D)=\begin{cases}\frac{1}{2}\log_2\left(\frac{\sigma^2}{D}\right) & ,D<\sigma^2\\ 0 & ,D\geq\sigma^2\end{cases} \tag{2.27}$$

ให้ $f(x)$ เป็นความหนาแน่นแบบเกาส์เซียนแสดงได้ดังนี้

$$f(x)=\left(\frac{1}{\sqrt{2\pi\sigma}}\right)\exp\left[\frac{-x^2}{2\sigma^2}\right] \tag{2.28}$$

เราจะได้

$$R(D)=\min_{f(x,y)}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\log_2\left[\frac{f(x,y)}{f(x)f(y)}\right]f(x,y)dxdy \tag{2.29}$$

เมื่อค่าต่ำสุดเป็นค่าความหนาแน่นร่วม (joint densities) ทั้งหมดของ $f(x,y)$ ซึ่งเป็นไปตาม

$$f(x)=\int_{-\infty}^{\infty}f(x,y)dy \tag{2.30}$$

และ

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-y)^2 f(x,y)dxdy\leq D \tag{2.31}$$

และเมื่อ $g(y)$ เป็นความหนาแน่นที่อยู่บริเวณขอบ (marginal density)

$$g(y)=\int_{-\infty}^{\infty}f(x,y)dx \tag{2.32}$$

สำหรับแหล่งข้อมูลที่ไม่มีเมมโมรีของข้อมูลซึ่งแต่ละค่าเอาท์พุทมีค่าความแปรปรวน σ^2, $R(D)=0$ เมื่อ $D\geq\sigma^2$ ดังนั้นเราอาจจะสมมติได้ว่า $D<\sigma^2$ ให้ $f(x,y)$ เป็นค่าความหนาแน่นร่วมซึ่ง

$$f(x,y)=\left(\frac{1}{2\pi\sqrt{D(\sigma^2-D)}}\right)\exp\left[-\frac{(x-y)^2}{2D}-\frac{y^2}{2(\sigma^2-D)}\right] \tag{2.33}$$

และจาก (2.30) และ (2.31) จะได้ว่า

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\log_2\left[\frac{f(x,y)}{f(x)f(y)}\right]f(x,y)dxdy=\frac{1}{2}\log_2\left(\frac{\sigma^2}{D}\right) \tag{2.34}$$

ดังนั้น

$$R(D)\le\frac{1}{2}\log_2\left(\frac{\sigma^2}{D}\right) \tag{2.35}$$

เพื่อหาค่าความไม่เท่ากันแบบรีเวอร์ส (reverse inequality) โดยนำ $f(x,y)$, (2.30)และ (2.31) มาเขียนใหม่ได้ว่า

$$R(D)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\log_2\left[\frac{f(x,y)}{f(x)f(y)}\right]f(x,y)dxdy \tag{2.36}$$

ให้ D^* เป็นสมการทางด้านซ้ายมือของสมการ (2.31) ให้ $f(x\mid y)$ เป็นความหนาแน่นที่มีเงื่อนไข $f(x,y)/g(y)$ และให้ $g(x\mid y)$ เป็นความหนาแน่นที่มีเงื่อนไขดังนี้

$$g(x\mid y)=\left(\frac{1}{\sqrt{2\pi D^*}}\right)\exp\left[-\frac{(x-y)^2}{2D^*}\right] \tag{2.37}$$

ดังนั้นสามารถเขียนสมการทางด้านขวามือของสมการที่ (2.36)ในรูปผลบวกได้เป็น

$$\left\{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\log_2\left[\frac{f(x,y)}{f(x)f(y)}\right]f(x,y)dxdy+\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\log_2\left[\frac{g(x\mid y)}{f(x)}\right]f(x,y)dxdy\right\} \tag{2.38}$$

เทอมทางด้านซ้ายมือของสมการที่ (2.38) เป็นค่าที่ไม่เป็นลบ เนื่องจากคุณสมบัติของเอ็นโทรปีที่มีความสัมพันธ์กัน ทางขวามือของสมการ (2.38) ซึ่งพิสูจน์ได้ว่ามันเท่ากับ $1/2\log_2\left(\sigma^2/D^*\right)$ ซึ่งมีค่ามากกว่าหรือเท่ากับ $1/2\log_2\left(\sigma^2/D\right)$ เราได้แสดงให้เห็นว่า

$$R(D)\ge 1/2\log_2\left(\sigma^2/D^*\right) \tag{2.39}$$

ในสมการที่ (2.26) สำหรับฟังก์ชันความผิดเพี้ยน-อัตราบิต $D(R)$ แหล่งข้อมูลแบบเกาส์ที่ไม่มีเมมโมรีกันถูกแปลงไปอยู่ในรูปแบบของเดซิเบล ซึ่งจะได้ว่า

$$[D(R)]_{decibels} = (20\log_{10} 2)R \approx (6.0206)R \tag{2.40}$$

สมการที่ (2.40) บอกถึงว่า R เป็นจำนวนจริงบวก SQNR (Signal to Quantization Noise Ratio) ที่มีค่าสูงสุดสำหรับการควอนไทซ์แบบเวกเตอร์ของข้อมูลแบบเกาส์เซียนที่ไม่มีเมมโมรีที่อัตราบิตเท่ากับ R บิตต่อแซมเปิล SNR (Signal to Noise Ratio) ที่มีค่าสูงสุดของการควอนไทซ์แบบเวกเตอร์นั้นเหมือนกับกรณีของการควอนไทซ์แบบสเกลาร์ของข้อมูลที่มีการกระจายแบบยูนิฟอร์มที่ไม่มีเมมโมรีที่อัตราบิตเท่ากัน

จุดที่ว่าอัตราบิต R ของการบีบอัดข้อมูลภาพที่มีค่าคงที่ แหล่งข้อมูลที่ไม่มีเมมโมรีแบบเกาส์เซียน (Gaussian M emoryless s oruce) ซึ่งเป็นข้อมูลที่มีความผิดเพี้ยนมากที่สุดหรือกล่าวได้อีกอย่างหนึ่งว่าถ้า $D^*(R)$ เป็นฟังก์ชันความผิดเพี้ยน-อัตราบิตในหน่วยเดซิเบลสำหรับแหล่งข้อมูลแบบแบบเกาส์เซียนที่ไม่มีเมมโมรีและถ้า $D(R)$ เป็นฟังก์ชันความผิดเพี้ยน-อัตราบิตในหน่วยเดซิเบลสำหรับ แหล่งข้อมูลที่ไม่เป็นแบบแบบเกาส์เซียนและไม่มีเมมโมรี(Non-Gaussian Memoryless source)แล้วนั้นจะได้ว่า $D^*(R) \le D(R)$ สำหรับ $R > 0$ ทุกๆค่า

2.5.4 การประมาณค่าที่ไม่ต่อเนื่องของฟังก์ชันความผิดเพี้ยน- อัตราบิต $D(R)$

เพื่อที่จะคำนวณหาค่า $D(R)$ เราจะใช้การประมาณค่าที่ไม่ต่อเนื่องของ $D(R)$ ดังนั้นกำหนดให้ $D_f(R)$ เป็นฟังก์ชันความผิดเพี้ยน-อัตราบิตของแหล่งข้อมูลแบบที่ไม่มีเมมโมรี ที่มีผลบังคับโดยฟังก์ชันความหนาแน่น $f(x)$ และให้ $Q(x)$ เป็นการควอนไทซ์แบบสเกลาร์ ผลของความผิดพลาดของการควอนไทซ์จากการใช้ $Q(x)$ เพื่อควอนไทซ์ข้อมูลแบบที่ไม่มีเมมโมรีมีค่าดังนี้

$$\int_{-\infty}^{\infty}(x - Q(x))^2 f(x)dx \tag{2.41}$$

กำหนดให้ $\{x_1, x_2, \ldots, x_N\}$ เป็นเซตของระดับการควอนไทซ์ของ $Q(x)$ (เซตของระดับของการควอนไทซ์คือเซตและ $Q(R)$ เมื่อ R เป็นเส้นตรงจำนวนจริง) ให้ $p_1, p_2, \ldots, p_N$ เป็นความน่าจะเป็นซึ่งกำหนดได้ดังนี้

$$p_i \triangleq \int_{a_i}^{b_i} f(x)dx \tag{2.42}$$

เมื่อ a_i, b_i คือจุดปลายทางซ้ายและขวามือของเส้นจำนวนจริงตามลำดับ ที่มีช่วงอยู่ระหว่าง

$$\{x \in R : Q(x) = x_i\} \tag{2.43}$$

ให้ $\tilde{f}$ เป็นฟังก์ชันความหนาแน่นแบบไม่ต่อเนื่อง (discrete density function)

$$\tilde{f} \triangleq \sum_{i=1}^{N} p_i \delta(x - x_i) \tag{2.44}$$

ให้ $D_{\tilde{f}}(R)$ ฟังก์ชันความผิดเพี้ยน-อัตราบิตของแหล่งข้อมูลแบบที่ไม่มีเมมโมรีที่มีผลบังคับโดยความหนาแน่น $\tilde{f}$ เป็นที่ทราบว่าถ้าความผิดพลาดของการควอนไทซ์ของสมการที่ (2.41) มีค่าน้อยมากๆแล้ว $D_f(R)$ จะเป็นค่าที่ประมาณได้ดีโดยฟังก์ชันความผิดเพี้ยน-อัตราบิต $D_{\tilde{f}}(R)$

$$D_f(R) \approx D_{\tilde{f}}(R) \quad , R > 0 \tag{2.45}$$

นั่นหมายถึงถ้าเราสร้างกราฟ $D_f(R)$ เป็นฟังก์ชันของ R และ สร้าง $D_{\tilde{f}}(R)$ เป็นฟังก์ชันของ R จะได้ผลการสร้างกราฟที่ออกมาสอดคล้องกัน

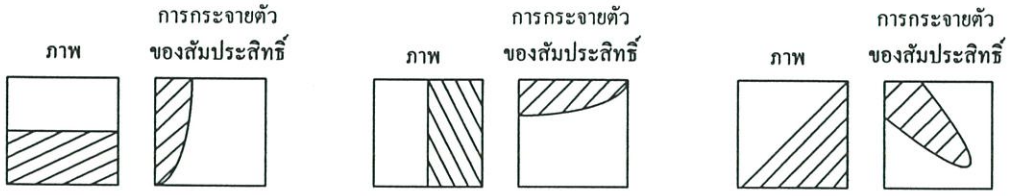
ฟังก์ชันความผิดเพี้ยน-อัตราบิต $D_{\tilde{f}}(R)$ ถูกเรียกว่าการประมาณค่าแบบดิสครีสของฟังก์ชันความผิดเพี้ยน-อัตราบิต $D(R)$

2.6 ความรู้เกี่ยวกับการบีบอัดข้อมูลภาพเบื้องต้น

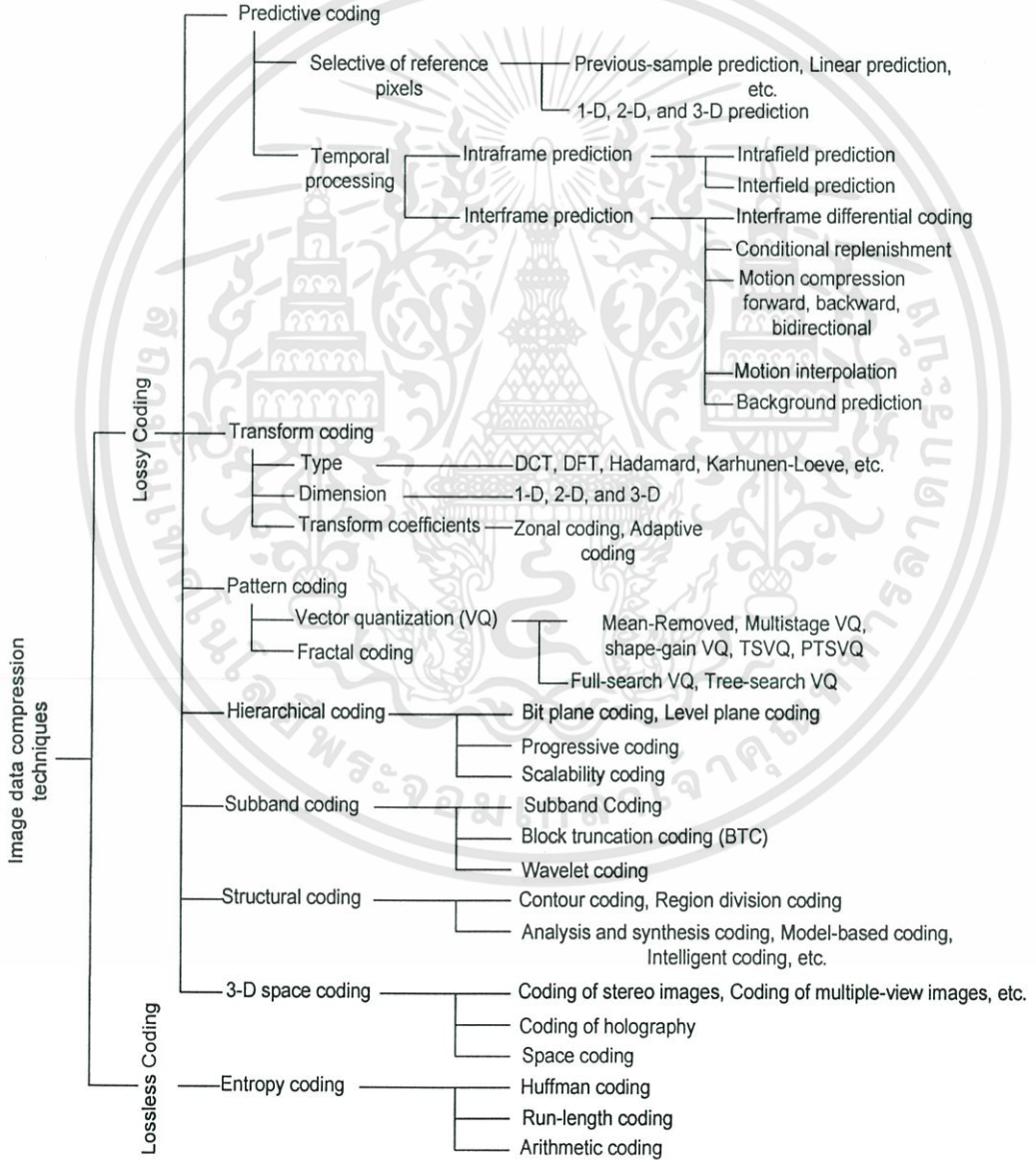
วิธีการบีบอัดข้อมูลภาพสามารถแบ่งออกได้เป็น 2 ประเภท ประเภทแรกเป็นวิธีการที่กำจัดความซ้ำซ้อน (redundancy) ในข้อมูลภาพ ความซ้ำซ้อนคือคุณลักษณะซึ่งสัมพันธ์กับการทำนาย การสุ่ม(sample) การทำนาย (predict) หรือการทำให้ราบเรียบ (smoothness) ในข้อมูลเช่น ภาพที่มีระดับเทาคงที่ สามารถทำการทำนายจุดภาพได้เพราะเราทราบจุดภาพแรกแล้ว หรือพูดอีกอย่างหนึ่งได้ว่าไวท์น้อยส์ (white noise)ไม่สามารถทำการทำนายได้และทุกๆจุดภาพจะต้องถูกเก็บเพื่อทำการสร้างภาพกลับ ดังนั้นวิธีการบีบอัดข้อมูลภาพหลายๆ วิธีพยายามที่จะแสดงภาพที่ถูกแซมเปิลมาในรูปแบบของอะเรย์ $\{u_i, j\}$ ให้อยู่ในรูปอะเรย์แบบอื่นคือ $\{\xi_i, j\}$ ซึ่งไม่มีการซ้ำซ้อนข้อมูลและด้วยเหตุนี้ $\{u_i, j\}$ สามารถกำหนดให้มีรูปแบบที่แน่นอนและไม่เหมือนกับแบบอื่น แต่อย่างไรก็ตามยังคงมีความผิดเพี้ยนเกิดขึ้นหลังจากได้ทำการสร้างกลับภาพ $\{u_i, j\}$ วิธีการบีบอัดข้อมูลภาพที่มีประสิทธิภาพนั้นมุ่งเน้นที่จะทำให้ความผิดเพี้ยนมีค่าน้อยที่สุด เทคนิคเช่น DPCM (Differential Pulse Code Modulation) และวิธีการเข้ารหัสแบบใช้การทำนายแบบต่างนั้นจัดอยู่ในประเภทนี้ด้วย

ประเภทที่สองของวิธีการบีบอัดข้อมูลคือวิธีการบีบอัดโดยอาศัยการแปลงข้อมูล (transform) โดยเก็บรักษาพลังงานของข้อมูลภาพไว้ ซึ่งหมายถึงการทำให้พลังงานไปอยู่ส่วนใดส่วนหนึ่ง

ของภาพและสามารถที่จะใช้จำนวนบิตจำนวนน้อยในการเข้ารหัสได้ ตัวอย่างเช่นการแปลงโคซายน์เต็มหน่วยดังรูปที่ 2.7



รูปที่ 2.7 การกระจายสัมประสิทธิ์ของการแปลงโคซายน์เต็มหน่วย



รูปที่ 2.8 วิธีการบีบอัดข้อมูลภาพแบบต่างๆ

2.6.1 การบีบอัดข้อมูลภาพแบบที่มีการสูญเสีย (Lossy image compression)

การบีบอัดข้อมูลภาพแบบที่มีการสูญเสียคือเมื่อข้อมูลที่ได้มีการบีบอัดข้อมูลแล้วไม่สามารถที่จะทำการสร้างกลับข้อมูลภาพให้เหมือนกับภาพต้นแบบได้ หรือพูดอีกอย่างได้ว่าข้อมูลมีการสูญหายไปขณะทำการบีบอัดข้อมูล แต่อัตราการบีบอัดข้อมูลจะสูงกว่าการบีบอัดข้อมูลแบบที่ไม่มีการสูญเสีย วิธีการบีบอัดข้อมูลภาพที่มีการสูญเสียมีดังต่อไปนี้

2.6.1.1 การบีบอัดข้อมูลภาพโดยการแปลง (Transform image coding)

วิธีการแปลงข้อมูลภาพ [20, 25] มีด้วยกันหลายวิธีซึ่งในที่นี้ได้นำเสนอวิธีการที่ใช้กันโดยทั่วไปประโยชน์ของการแปลงข้อมูลภาพคือการทำให้ข้อมูลมีความเหมาะสม หรือการที่ทำให้ข้อมูล ภาพไปอยู่รวมกันโดยที่มีความหนาแน่น (ความหนาแน่นของพลังงาน) อยู่เพียงขอบเขตใดขอบเขตหนึ่ง เพราะว่าข้อมูลภาพซึ่งปกติจะอยู่ในสปาเชียลโดเมน (spatial domain) ที่ยังไม่ได้ทำการแปลงจะมีความสัมพันธ์ระหว่างจุดภาพที่อยู่ใกล้ๆกันมาก แต่ถ้าข้อมูลดังกล่าวถูกแปลงไปอยู่อีกโดเมนหนึ่งซึ่งความหนาแน่นอยู่ในขอบเขตหนึ่งที่จำกัดซึ่งจะทำให้จำนวนบิตหรือช่วงของข้อมูลน้อยลง ดังนั้นจำนวนบิตที่จะนำมาเข้ารหัสจะมีจำนวนน้อยด้วยเช่นกัน

การแปลงฟูเรียร์เต็มหน่วย (Discrete Fourier Transform)

การแปลงข้อมูลภาพด้วยการแปลงฟูเรียร์เต็มหน่วย โดยการแปลงฟูเรียร์เต็มหน่วยของลำดับข้อมูล $\{u(n), n = 0,\ldots,N-1\}$ นิยามได้โดย

$$v(k) = \sum_{n=0}^{N-1} u(n) W_N^{kn}, k = 0,1,\ldots,N-1 \qquad (2.46)$$

เมื่อ $W_N \triangleq \exp\left\{-\frac{j2\pi}{N}\right\}$ การแปลงกลับทำได้โดย

$$\mu(n) = \frac{1}{N}\sum_{k=0}^{N-1} v(k) W_N^{-kn} \quad ,n = 0,1,\ldots,N-1 \qquad (2.47)$$

ในสมการที่หนึ่งนั้นเป็นการแปลงภาพเพียงหนึ่งมิติ แต่สำหรับการแปลงข้อมูลภาพสองมิติสำหรับการแปลงเต็มหน่วยสำหรับภาพที่มีขนาด $N \times N$ นิยามได้โดย

$$v(k,l)=\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}u(m,n)W_N^{km}\,W_N^{ln} \qquad 0\le k,l\le N-1 \tag{2.48}$$

และการแปลงกลับสามารถทำได้โดย

$$u(m,n)=\frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}v(k,l)W_N^{-km}W_N^{-ln}, \quad 0\le m,n\le N-1 \tag{2.49}$$

การแปลงโคซายน์เต็มหน่วย (Discrete Cosine Transform)

การแปลงโคซายน์ของเมทริกซ์ $C=\{c(k,n)\}$ ที่มีขนาด $N\times N$ เราเรียกวิธีนี้ว่าเป็นการแปลงโคซายน์ที่ไม่ต่อเนื่องหรือเบสิสเวกเตอร์ของการแปลงโคซายน์เต็มหน่วย (Discrete Cosine Transform) ซึ่งนิยามได้ดังนี้

$$c(k,n)=\begin{cases}\dfrac{1}{\sqrt{N}}, & k=0, 0\le n\le N-1\\[2ex] \sqrt{\dfrac{2}{N}}\cos\dfrac{\pi(2n+1)k}{2N}, & 1\le k\le N-1,\ 0\le n\le N-1\end{cases} \tag{2.50}$$

ในการแปลงแบบหนึ่งมิติของลำดับข้อมูล $\{u(n),\ 0\le n\le N-1\}$ สามารถนิยามได้ดังนี้

$$v(k)=\alpha(k)\sum u(n)\cos\left[\frac{\pi(2n+1)k}{2N}\right], \quad 0\le k\le N-1 \tag{2.51}$$

เมื่อ

$$\alpha(0)\triangleq\sqrt{\frac{1}{N}}, \quad \alpha(k)\triangleq\sqrt{\frac{2}{N}} \text{ สำหรับ } 1\le k\le N-1 \tag{2.52}$$

และการแปลงกลับของการแปลงแบบหนึ่งมิติคือ

$$u(n)=\sum_{k=0}^{N-1}\alpha(k)v(k)\cos\left[\frac{\pi(2n+1)k}{2N}\right], \qquad 0\le n\le N-1 \tag{2.53}$$

ส่วนการแปลงโคซายน์ที่ไม่ต่อเนื่องในสองมิติสามารถทำได้โดย

$$v(k,n) = \alpha(k)\alpha(n)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} u(x,y)\cos\left[\frac{(2x+1)k\pi}{2N}\right]\cos\left[\frac{(2y+1)n\pi}{2N}\right] \quad (2.54)$$

$$u(x,y) = \sum_{k=0}^{N-1}\sum_{n=0}^{N-1} \alpha(k)\alpha(n)v(k,n)\cos\left[\frac{(2x+1)k\pi}{2N}\right]\cos\left[\frac{(2y+1)n\pi}{2N}\right] \quad (2.55)$$

$\alpha(k),\alpha(n) = \frac{1}{2}$ เมื่อ $k,n = 0$ และ $\alpha(k),\alpha(n) = 1$ ในกรณีอื่นๆ

คุณสมบัติของการแปลงโคซายน์

1. การแปลงโคซายน์ที่เป็นจริงและออร์ทอกอนัล(orthogonal) หมายถึง

$$C = C^* \Rightarrow C^{-1} = C^T$$

2. การแปลงโคซายน์เป็นการแปลงอย่างเร็ว การแปลงโคซายน์ของเวกเตอร์ขององค์ประกอบ N ตัวสามารถคำนวณในการทำงานได้โดยเป็นจำนวนครั้ง $O(N \log_2 N)$ ครั้งบนจำนวน N จุดของการแปลงฟูเรียร์แบบที่ไม่ต่อเนื่อง

3. การแปลงโคซายน์ในการทำให้พลังงานมีขนาดกะทัดรัด สำหรับข้อมูลที่มีความสัมพันธ์กันมากๆได้

4. เบสิสเวกเตอร์ (Basis vector) ของการแปลงโคซายน์คือแกนเวกเตอร์ของเมทริกซ์สมมาตรสามแกนหลัก Q_C ซึ่ง

$$Q_C = \begin{bmatrix} 1-\alpha & -\alpha & 0 & \\ -\alpha & 1 & \ddots & \\ 0 & \ddots & 1 & -\alpha \\ & & -\alpha & 1-\alpha \end{bmatrix}$$

5. การแปลงโคซายน์ของเมทริกซ์ขนาด $N \times N$ นั้นมีความใกล้เคียงกับการแปลงแบบ Karhunen-Loeve(KL) ของลำดับข้อมูลมาร์คอฟอันดับที่ 1 ที่มีความยาว N

การแปลงข้อมูลโดยการใช้การแปลงโคซายน์เป็นวิธีที่ใช้เป็นมาตรฐานในการแปลงข้อมูลภาพของในช่วงแรกๆของการบีบอัดข้อมูลด้วยวิธีการ JPEG(Joint Photographic Experts Group) [26] ซึ่งเป็นวิธีการบีบอัดข้อมูลที่มีประสิทธิภาพวิธีหนึ่ง ต่อมาได้มีการพัฒนาวิธีการแปลงข้อมูลภาพที่สามารถสร้างกลับภาพได้ดี

การแปลงแบบซายน์เต็มหน่วย (Discrete Sine Transform)

การแปลงแบบซายน์ของเมทริกซ์ขนาด $N \times N$ จะได้เป็น $\psi = \{\psi(k,n)\}$ ซึ่งจะถูกเรียกว่า การแปลงแบบซายน์ที่ไม่ต่อเนื่องและสามารถเขียนเป็นสมการได้ดังนี้

$$\psi(k,n) = \sqrt{\frac{2}{N+1}} \sin \frac{\pi(k+1)(n+1)}{N+1} \quad , \quad 0 \le k,n \le N-1 \tag{2.56}$$

การแปลงแบบซายน์ในข้อมูลแบบหนึ่งมิตินั้นจะได้ว่า

$$v(k) = \sqrt{\frac{2}{N+1}} \sum_{n=0}^{N-1} u(n) \sin \frac{\pi(k+1)(n+1)}{N+1}, \quad 0 \le k \le N-1 \tag{2.57}$$

$$u(n) = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N-1} u(n) v(k) \sin \frac{\pi(k+1)(n+1)}{N+1}, \quad 0 \le k \le N-1 \tag{2.58}$$

การแปลงแบบฮาดามาร์ด (Hadamard Transform)

องค์ประกอบของเบสิสเวกเตอร์ของการแปลงแบบฮาดามาร์ดจะใช้เพียงแค่ค่า ± 1 และพบว่าองค์ประกอบนี้มีความเหมาะสมสำหรับการประมวลผลสัญญาณดิจิตอล เมทริกซ์ของการแปลงแบบฮาดามาร์ด H_n คือเมทริกซ์ที่มีขนาด $N \times N$ เมื่อ $N \triangleq 2^n, n = 1,2,3$ ซึ่งเขียน H_1 ได้ดังนี้

$$H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.59}$$

และผลคูณคอนเนคเคอร์ (Connecker product) ของเมทริกซ์ H_n จะได้

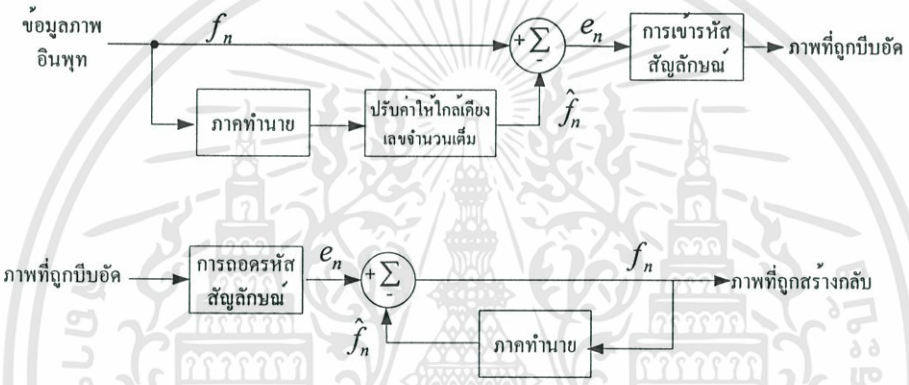
$$H_n = H_{n-1} \otimes H_1 = H_1 \otimes H_{n-1} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix} \tag{2.60}$$

เมทริกซ์เวกเตอร์ของการแปลงแบบฮามาดาร์ด สามารถสร้างโดยการสุ่มเอาประเภทของฟังก์ชันที่เรียกว่า Walsh function ซึ่งฟังก์ชันดังกล่าวนี้จะนำเอาค่าเพียงแค่ค่า ± 1 มาใช้และ

จัดรูปแบบให้เบสิสตั้งฉากกันอย่างสมบูรณ์สำหรับฟังก์ชันที่รวมกันของเมทริกซ์จัตุรัสหลายๆ เมทริกซ์ซึ่งนิยามการแปลงแบบนี้ว่าการแปลงแบบ Walsh-Hadamard

2.6.1.2 การเข้ารหัสโดยการทำนาย (Predictive Coding)

วิธีการเข้ารหัสโดยการทำนาย [21] ทำได้โดยการกำจัดจุดภาพที่ซ้ำซ้อนของจุดภาพใกล้เคียงโดยการแยกและการเข้ารหัสเพียงข้อมูลที่ได้มาใหม่ในแต่ละจุดภาพ ข้อมูลใหม่ของจุดภาพจะถูกกำหนดเป็นความแตกต่างระหว่างค่าจริงและค่าที่ทำการทำนายของบิตนั้นๆ



รูปที่ 2.9 วิธีการเข้าและถอดรหัสโดยการทำนาย

จากรูปเป็นส่วนประกอบเบื้องต้นของการเข้ารหัสโดยการทำนายที่มีการสูญเสียของข้อมูล ระบบการทำงานประกอบด้วยภาคเข้ารหัสและภาคถอดรหัสทั้งสองส่วนที่ทำการทำนายข้อมูล ข้อมูลภาพอินพุทภาพแสดงด้วย f_n ภาคที่ทำการทำนายจะทำการสร้างค่าหนึ่งขึ้นมาจากอินพุทบางส่วนก่อนหน้านี้ ซึ่งเอาท์พุทของภาคที่ทำการทำนายแสดงด้วย $\hat{f}_n$ ซึ่งเป็นค่าที่ถูกทำให้มีค่าใกล้เคียงกับจำนวนจริง(อินพุท)มากที่สุด โดยเอาท์พุทดังกล่าวถูกใช้เพื่อนำมาทำการหาความผิดพลาดเนื่องจากการทำนายโดยสามารถหาได้ดังนี้

$$e_n = f_n - \hat{f}_n \tag{2.61}$$

การเข้ารหัสทำได้โดยใช้การเข้ารหัสแบบความยาวรหัสที่เปลี่ยนแปลงได้ เพื่อกำเนิดจุดภาพถัดไปของลำดับข้อมูลที่เข้ารหัส ภาคถอดรหัสทำการสร้างข้อมูล e_n กลับมาจากรหัสคำที่มีความยาวไม่เท่ากันและทำการทำกลับกับขั้นตอนของการเข้ารหัสโดย

$$f_n = e_n + \hat{f}_n \tag{2.62}$$

มีหลายวิธีที่สามารถนำมาใช้เพื่อสร้าง $\hat{f}_n$ แต่โดยส่วนใหญ่การทำนายจะสร้างโดยการรวมกันแบบเชิงเส้นของจุดภาพก่อนหน้านั้น m จุดภาพ นั่นคือ

$$\hat{f}_n = round\left[\sum_{i=1}^{m} \alpha_i f_{n-i}\right] \tag{2.63}$$

เมื่อ m คือลำดับของการทำนายแบบเชิงเส้นซึ่งทำให้เป็นจำนวนเต็ม(round) คือฟังก์ชันที่ใช้เพื่อทำให้ค่านั้นใกล้จำนวนเต็ม α_i ซึ่ง $i = 1,2,\ldots,m$ คือสัมประสิทธิ์ของการทำนาย

ในกรณีอื่นเมื่อใช้ n เป็นดัชนีในพิกัดระนาบและหรือจำนวนเฟรม(ลำดับภาพ)ของภาพในการเข้ารหัสที่มีการทำนายแบบเชิงเส้นหนึ่งมิติสามารถเขียนเป็นสมการได้ดังนี้

$$\hat{f}_n(x,y) = round\left[\sum_{i=1}^{m} \alpha_i f(x,y-i)\right] \tag{2.64}$$

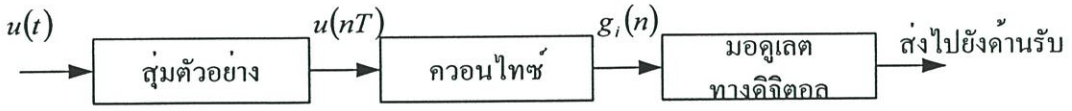
เมื่อแต่ละตัวแปรนั้นแสดงถึงฟังก์ชันของระนาบมิติของ x และ y จากสมการ (2.64) นั้น การทำนายแบบเชิงเส้นของ $\hat{f}(x,y)$ ในหนึ่งมิติคือฟังก์ชันของจุดภาพก่อนหน้านั้นบนเส้นตรงปัจจุบันเท่านั้น ในการเข้ารหัสแบบใช้การทำนายในสองมิติ การทำนายจะเป็นฟังก์ชันของจุดภาพก่อนหน้าที่กวาดไปทางขวาและซ้ายและจากบนลงล่างของภาพ ในสมการ (2.64) นั้นไม่สามารถที่จะคำนวณจุดภาพแรกของแต่ละแถวในข้อมูลภาพได้ซึ่งจุดภาพดังกล่าวนั้นจะต้องใช้การเข้ารหัสแบบอื่น

2.6.2 การบีบอัดข้อมูลแบบที่ไม่มีการสูญเสีย (Lossless image compression)

การบีบอัดข้อมูลแบบที่ไม่มีการสูญเสียข้อมูล หมายถึงการบีบอัดข้อมูลภาพโดยที่ข้อมูลภาพหลังจากสร้างภาพกลับมาแล้วนั้นไม่มีการสูญเสียข้อมูล ดังนั้นการบีบอัดข้อมูลภาพแบบนี้ทำให้การเข้าลดข้อมูลภาพในลักษณะที่ลดบิตข้อมูลที่ซ้ำซ้อนของข้อมูลภาพออกไป

การเข้ารหัสจุดภาพ (Pixel Encoding)

วิธีการนี้จะทำการเข้ารหัสแต่ละจุดภาพอิสระจากกัน ซึ่งไม่พิจารณาความสัมพันธ์ของจุดภาพ ในระบบ PCM สัญญาณวิดีโอจะถูกแซมเปิลควอนไทซ์ และทำการเข้ารหัสโดยผลที่ได้จะเรียกว่าคำรหัส (code word) ที่เหมาะสม



รูปที่ 2.10 การเข้ารหัสในระบบ PCM

เอาท์พุทของการควอนไทซ์จะถูกเข้ารหัสโดยคำรหัสที่มีความยาวของเลขไบนารีจำนวนจำกัด B บิต โดยทั่วไปจะใช้ 8 บิต (สำหรับภาพสีเดียว(monochrome)) หรือการสัมมนาทางไกล (video conference) แต่สำหรับภาพทางการแพทย์หรือสัญญาณวิดีโอภาพสีจะใช้ 10-12 บิต

อัตราบิตของการเข้ารหัสแบบ PCM ที่ใช้บิตน้อยที่สุดจะได้โดยการคำนวณอัตราบิต-ความผิดเพี้ยน ดังนี้

$$R_{PCM} = \frac{1}{2}\log\frac{\sigma_u^2}{\sigma_q^2} \quad ,\sigma_q^2 < \sigma_u^2 \tag{2.65}$$

เมื่อ σ_u^2 คือค่าความแปรปรวนของทางด้านอินพุทของข้อมูลของภาคควอนไทซ์และ σ_q^2 คือความผิดเพี้ยนเฉลี่ยกำลังสองของภาคควอนไทซ์

การเข้ารหัสเอ็นโทรปี (Entropy Encoding)

ถ้าจุดภาพที่จะทำการควอนไทซ์มีการกระจายที่เป็นแบบปกติ (uniform) แล้วเอ็นโทรปีของการกระจายจะมีค่าน้อยกว่า B (B จำนวนบิตของภาพต้นแบบ) การเข้ารหัสเอ็นโทรปีเป็นการเข้ารหัสบล็อกของข้อมูลที่มีขนาด M จุดภาพซึ่งมีข้อมูล MB บิตโดยมี $p_i \quad ,i = 0,1,\ldots,L-1, L = 2^{MB}$ เป็นความน่าจะเป็น ดังนั้นอัตราบิตเฉลี่ยคือ

$$\sum_i p_i\left(-\log_2 p_i\right) = H \tag{2.66}$$

การเข้ารหัสดังกล่าวจะจัดสรรให้แต่ละบล็อกมีความยาวรหัสที่แตกต่างกันไป เมื่อบล็อกที่มีความน่าจะเป็นสูงจะแทนด้วยรหัสความยาวรหัสจำนวนน้อย และมากขึ้นไปตามลำดับ ถ้า $-\log_2 p_i$ ไม่เป็นจำนวนเต็ม อัตราบิตที่ได้จะเกิน H สำหรับเทคนิคที่เรียกว่าการเข้ารหัสแบบฮัฟแมน (Huffman Coding) มีประสิทธิภาพมากกับวิธีการเข้ารหัสที่มีความยาวรหัสที่เปลี่ยนแปลงได้

การเข้ารหัสแบบรัน-เลนจ์ (Run-Length coding)

พิจารณาแหล่งข้อมูลที่มีตัวเลขที่เหมือนกันและเกิดขึ้นติดต่อกัน การเข้ารหัสแบบรัน-เลนจ์ [5, 12, 17] มีประโยชน์อย่างยิ่งกับข้อมูลที่เกิดขึ้นซ้ำๆกันเป็นจำนวนมากๆ เช่นในเอกสารข้อมูลภาพ หรือ แผนที่ ถ้าให้ p เป็นความน่าจะเป็นของการเกิดจำนวนเลข 0 (เป็นส่วนที่แสดงด้วยสีขาวในภาพ) มีค่าใกล้เคียงหนึ่ง (ในที่นี้การเข้ารหัสแต่ละช่วงที่มีเลขจำนวนติดต่อกันเราจะเรียกว่า RUN) แล้วสมมติว่าการเข้ารหัสสามารถเข้ารหัสได้ความยาวสูงที่สุดของรหัสเป็น M และให้ $M = 2^m - 1$ แล้วการเข้ารหัสดังกล่าวนี้จะเป็นการนำเอาข้อมูลขนาด m บิตมาทำการเข้ารหัสโดยที่รหัสมีความยาวคงที่ ถ้าจำนวนเลขศูนย์เกิดขึ้นกระจัดกระจายไม่ต่อเนื่องกันแล้วความน่าจะเป็นของความยาวของ RUN จะกลายเป็นการกระจายในลักษณะเป็นพื้นที่ (geometric distribution)

$$g(l) = \begin{cases} p^l(1-p), & 0 \le l \le M-1 \\ p^M, & l = M \end{cases} \tag{2.67}$$

เพราะว่าความยาวของ RUN ของ $l \le M-1$ เสมือนเป็นลำดับของ l โดยที่เป็นเลขศูนย์ 10 จำนวนแล้วตามมาด้วยเลข 1 นั่นคือจะมีจำนวนทั้งหมด $l+1$ จำนวนเฉลี่ยของซิมบอล (symbol) ต่อ RUN จะเป็น

$$\mu = \sum_{l=0}^{M-1}(l+1)p^l(1-p) + Mp^M = \left(\frac{1-p^M}{1-p}\right) \tag{2.68}$$

จากสมการนี้จะได้ว่าเป็นการนำเอาตัวเลขจำนวน m บิตมาสร้างรหัสที่มีความยาว RUN หนึ่งค่าสำหรับลำดับข้อมูล μ ที่เป็นซิมบอลแบบไบนารีหลายๆตัว อัตราการบีบอัดข้อมูลที่เป็นไปได้คือ

$$C = \frac{\mu}{m} = \frac{\left(1-p^M\right)}{m(1-p)} \tag{2.69}$$

เช่นถ้าให้ $p = 0.9$ และ $M = 15$ เราจะได้ $m = 4, \mu = 0.79$ และ $C = 1.985$ อัตราบิตจะได้เป็น $B_a = m/\mu = 0.516$ บิตต่อจุดภาพ และประสิทธิภาพของการเข้ารหัสจะได้ว่า $H/B_a = 0.469/0.615 = 91\%$

ตัวอย่าง ของการเข้ารหัสแบบรัน-เลนจ์

อินพุทคือ R T A A A A S D E E E E E

เอาท์พุทหลังจากทำการบีบอัดข้อมูล R T *4A S D *5E

ตัวอย่างข้างบนแต่ละสตรีมของสัญลักษณ์ที่เหมือนกัน จะถูกแทนด้วยตัวเลขจำนวนของสัญลักษณ์ เครื่องหมาย * เป็นส่วนที่บอกว่าส่วนถัดไปจะเป็นส่วนที่สัญลักษณ์เกิดขึ้นซ้ำกัน

การเข้ารหัสแบบฮัฟแมน (Huffman Coding Algorithm)

ถ้าให้แหล่งกำเนิดข่าวสารกำเนิดข่าวสารที่แตกต่างกัน L องค์ประกอบ แต่ละองค์ประกอบมีความน่าจะเป็นของการเกิดเป็น $p_1, p_2, \ldots, p_{L-1}$ การเข้ารหัสด้วยรหัสฐาน D วิธีการของฮัฟแมนมีขั้นตอนดังนี้

ขั้นที่ 1 เรียงลำดับองค์ประกอบของข้อมูลข่าวสารจากความน่าจะเป็นสูงไปหาต่ำ

ขั้นที่ 2 คำนวณตัวเลข l ที่เป็นไปตามเงื่อนไข 2 ประการต่อไปนี้

$$2 \le l \le D \,,\, \frac{L-l}{D-1} = \text{เลขจำนวนเต็มที่มีค่าบวก}$$

ขั้นที่ 3 จากนั้นนำค่าความน่าจะเป็นขององค์ประกอบข่าวสาร l ที่มีค่าความน่าจะเป็นต่ำที่สุดสองตัวแรกมารวมกัน แล้วให้ถือว่าเป็นองค์ประกอบข่าวสารใหม่ 1 ตัว โดยมีความน่าจะเป็นใหม่ที่ได้จากผลรวมของความน่าจะเป็นของทั้งสององค์ประกอบรวมกัน

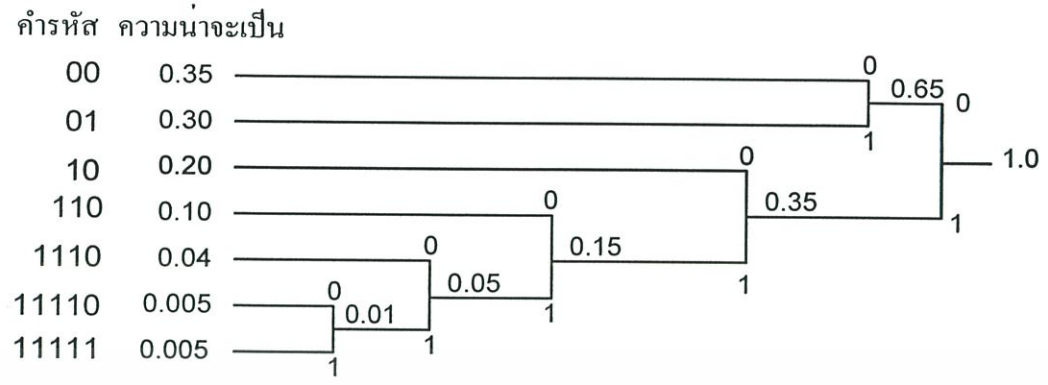
ขั้นที่ 4 นำองค์ประกอบใหม่มาพิจารณาร่วมกับองค์ประกอบที่เหลือ แล้วเรียงลำดับความน่าจะเป็นใหม่แล้วทำตามขั้นตอนที่ 2 จนกว่าความน่าจะเป็นที่รวมได้มีค่าเป็น 1 เมื่อทำการคำนวณความยาวเฉลี่ยและเอ็นโทรปีจะได้ดังนี้

$$n = \sum_{i=1}^{L-1} n_i p_i \tag{2.70}$$

$$H = \sum_i p_i \log_2 \frac{1}{p_i}$$

ตัวอย่างการเข้ารหัสแบบฮัฟแมน

สมมติให้องค์ประกอบข่าวสารมี 7 องค์ประกอบคือ $x_1, x_2, x_3, x_4, x_5, x_6, x_7$ โดยที่มีความน่าจะเป็นของการเกิดเป็น 0.35, 0.30, 0.20, 0.10, 0.04, 0.005, 0.005 ตามลำดับ เมื่อเข้ารหัสด้วยวิธีการเข้ารหัสฮัฟแมนจะได้ดังนี้



รูปที่ 2.11 การสร้างรหัสฐานสองด้วยวิธีฮัฟแมน

ตารางที่ 2.1 ความน่าจะเป็นของสัญลักษณ์ในการเข้ารหัสด้วยวิธีฮัฟแมน

สัญลักษณ์	ความน่าจะเป็น	คำรหัส	ความยาวรหัส
x_1	0.35	00	2
x_2	0.30	01	2
x_3	0.20	10	2
x_4	0.10	110	3
x_5	0.04	1110	4
x_6	0.005	11110	5
x_7	0.005	11111	5

เมื่อทำการคำนวณความยาวเฉลี่ยและเอ็นโทรปีจะได้ดังนี้

$$n = 2\times 0.35 + 2\times 0.30 + 2\times 0.20 + 3\times 0.10 + 4\times 0.04 + 5\times 0.005 + 5\times 0.005$$

$$n = 2.21$$

$$H = -\begin{bmatrix} 0.35\log_2 0.35 + 0.30\log_2 0.30 + 0.2\log_2 0.2 + 0.1\log_2 0.1 + 0.04\log_2 0.04 \\ + 0.005\log_2 0.005 + 0.005\log_2 0.005 \end{bmatrix}$$

$$H = 2.11$$

การเข้ารหัสเลขคณิต (Arithmetic Coding)

การเข้ารหัสเลขคณิต [19, 24, 27, 28] เป็นวิธีการเข้ารหัสที่รู้จักกันทั่วไปซึ่งเป็นวิธีการสำหรับการเข้ารหัสแบบที่ไม่มีการสูญเสียข้อมูล (lossless) การเข้ารหัสเลขคณิตจะทำการแทนที่ลำดับข้อมูลสัญลักษณ์ด้วยช่วงระยะห่างของจำนวนจริงในที่นี้คือ $[0,1)$ เมื่อแต่ละสัญลักษณ์ถูก

เข้ารหัสช่วงระยะห่างดังกล่าวจะถูกแทนที่ด้วยช่วงย่อยลงไป หลังจากที่ได้มีการเข้ารหัสลำดับสัญลักษณ์แล้วลำดับดังกล่าวสามารถที่จะสร้างกลับคืนมาได้อย่างถูกต้อง

ขั้นตอนการเข้ารหัสเลขคณิต

จัดสรรค่าความน่าจะเป็นให้กับแต่ละสัญลักษณ์และวิธีการเข้ารหัสสามารถทำได้เป็นสามขั้นตอนดังต่อไปนี้

ขั้นที่ 1 การเข้ารหัสเริ่มด้วยการกำหนด "ช่วงปัจจุบัน" $[H, L)$ กำหนดให้มีค่าเป็น $[0,1)$

ขั้นที่ 2 สำหรับแต่ละเหตุการณ์ในอินพุทไฟล์การเข้ารหัสทำได้เป็นสองขั้นตอนคือ **ก.** และ **ข.** ดังนี้

ก. จัดการแบ่งช่วงย่อยปัจจุบันไปเป็นช่วงย่อยตามแต่ละสัญลักษณ์อินพุท

ข. ขนาดของช่วงย่อยจะเป็นสัดส่วนตามค่าความน่าจะเป็นของสัญลักษณ์และเลือกช่วงขอบเขตย่อยที่สอดคล้องกับสัญลักษณ์ที่พบในข้อมูลแล้วนำช่วงของขอบเขตดังกล่าวมาเป็นช่วงของขอบเขตปัจจุบันแทน ทำในขั้นตอนที่สองจนกว่าจะสิ้นสุดชุดของข้อมูลที่จะนำมาเข้ารหัสหรือเมื่อพบอักษรที่แสดงถึงจุดสิ้นสุดของข้อมูล

ขั้นที่ 3 เอาท์พุทที่ได้ก็คือช่วงขอบเขตสุดท้ายหลังจากที่ได้มีการแบ่งเรียบร้อยแล้วจากขั้นตอนที่ 2 แล้วนำค่าของขอบเขตดังกล่าวมาแปลงเป็นรหัสไบนารี

จำนวนบิตของรหัสไบนารีที่ใช้แทนช่วงของขอบเขตพิจารณาได้จากผลคูณของค่าความน่าจะเป็นของแต่ละสัญลักษณ์ให้มีค่าเท่ากับ p ที่เกิดจากลำดับของสัญลักษณ์ข้อมูล จำนวนบิตของรหัสจะเท่ากับ $-\log_2 p$ บิต การคำนวณขนาดของขอบเขตของสัญลักษณ์ที่เกิดขึ้นโดยการทำการหาช่วงขอบเขตใหม่ได้ดังนี้

$$[L + R_L(H - L), L + R_H(H - L)) \tag{2.71}$$

เมื่อ $L =$ ขอบเขตล่างของข้อมูลทั้งหมด เช่นช่วงแรกคือ " 0 "

$H =$ ขอบเขตบนของข้อมูลทั้งหมด เช่นช่วงแรกคือ " 1 "

$R_L =$ ขอบเขตล่างของข้อมูลของสัญลักษณ์ที่กำลังทำการเข้ารหัส เช่น สัญลักษณ์ I มีขอบเขตล่างเป็น 0.35

$R_H =$ ขอบเขตบนของข้อมูลของสัญลักษณ์ที่กำลังทำการเข้ารหัส เช่น สัญลักษณ์ I มีขอบเขตบนเป็น 0.5

ตัวอย่างของการแปลงช่วงขอบเขตเป็นเลขไบนารีเช่นดังตัวอย่างข้างบน $S = \{E, I, O, U, !\}$ และมีความน่าจะเป็น $\{0.15 \quad 0.35 \quad 0.05 \quad 0.25 \quad 0.20\}$ ตามลำดับ การแบ่งช่วงระยะระหว่าง 0 กับ 1 แสดงไว้ดังตารางที่ 2.2

ตารางที่ 2.2 แสดงการแบ่งขอบเขตตามความน่าจะเป็นสะสมในการเข้ารหัสเลขคณิต

สัญลักษณ์ข้อมูลต้นแบบ	ความน่าจะเป็น	ความน่าจะเป็นสะสม	ขอบเขต
E	0.15	0.15	[0 , 0.15)
I	0.35	0.5	[0.15 , 0.5)
O	0.05	0.55	[0.5 , 0.55)
U	0.25	0.8	[0.55 , 0.8)
!	0.20	1	[0.8 , 1)

ตัวอย่างการเข้ารหัสสัญลักษณ์ *EEIU*! แสดงไว้ดังรูปที่ 2.12 ซึ่งการเข้ารหัสจะทำการแบ่งดังนี้

1. สัญลักษณ์ E จะลดขอบเขตของระยะห่างระหว่าง 0 กับ 1 ลงเป็น $[0,0.15)$
2. สัญลักษณ์ E ตัวที่สองจะลดขอบเขตจาก $[0,0.15)$ เป็น $[0,0.0225)$
3. สัญลักษณ์ I จะลดขอบเขตจาก $[0,0.0225)$ เป็น $[0.0034,0.0113)$
4. สัญลักษณ์ U จะลดขอบเขตจาก $[0.0034,0.0113)$เป็น $[0.0077,0.0097)$
5. สัญลักษณ์ !จะลดขอบเขตจาก $[0.0077,0.0097)$ เป็น $[0.0093,0.0097)$

ดังนั้นสัญลักษณ์ *EEIU*! สามารถเข้ารหัสโดยการแทนค่าด้วยช่วงระยะห่างหรือตัวเลขจำนวนใดๆที่อยู่ในช่วง $[0.0093,0.0097)$ ค่าความน่าจะเป็นของการเข้ารหัสของข้อมูลนี้คือ $(0.15)^2 \times 0.35 \times 0.25 \times 0.20 = 0.00039375$ และจำนวนบิตของข้อมูลที่จะใช้แทนกลุ่มของสัญลักษณ์ข้อมูลนี้จะเท่ากับ $\lceil -\log_2(0.00039375) \rceil = \lceil 11.3104 \rceil = 11$ บิต จากนั้นทำการเลือกข้อมูลในช่วง $\{0.0093,0.0097\}$มาแปลงเป็นเลขไบนารี โดยเลือกค่ากลางในช่วงนี้ซึ่งก็คือ 0.0095 นำมาแปลงเป็นเลขไบนารีจะได้ $(0.00000010011\ldots)_2$ แต่จะนำข้อมูลมาเพียง 11 บิตเท่านั้นเพื่อใช้แทนข้อมูลที่นำมาเข้ารหัส ดังนั้นจะได้รหัสที่แทนข้อมูล *EEIU*! เป็น 0000 0010 011

ขั้นตอนการถอดรหัส

สมมติว่ารับข้อมูลเข้ามาเป็น 0.0095

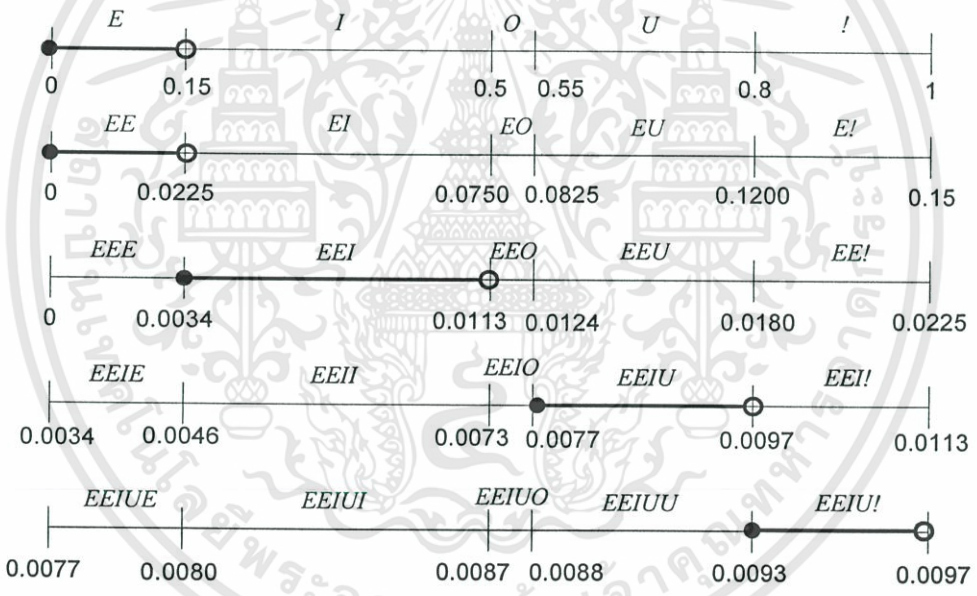
1. เริ่มโดยการแบ่งช่วง $[0,1)$ ตามความน่าจะเป็น พบว่า 0.0095 อยู่ในช่วง $[0,0.15)$ซึ่งเป็นช่วงของสัญลักษณ์ E ซึ่งจะได้ตัวอักษรตัวแรกคือ E จากนั้นทำการลดขอบเขตลงเป็น $[0,0.15)$

2. ทำการแบ่งช่วง $[0,0.15)$ ตามความน่าจะเป็น พบว่า 0.0095 อยู่ในช่วง $[0,0.0225)$ ซึ่งเป็นช่วงของสัญลักษณ์ E ซึ่งจะได้ตัวอักษรตัวที่สองคือ E จากนั้นทำการลดขอบเขต $[0,0.15)$ ลงเป็น $[0,0.0225)$

3. ทำการแบ่งช่วง $[0,0.0225)$ ตามความน่าจะเป็น พบว่า 0.0095 อยู่ในช่วง $[0.0034,0.0113)$ ซึ่งเป็นช่วงของสัญลักษณ์ I ซึ่งจะได้ตัวอักษรตัวที่สามคือ I จากนั้นทำการลดขอบเขต $[0,0.0225)$ ลงเป็น $[0.0034,0.0113)$

4. ทำการแบ่งช่วง $[0.0034,0.0113)$ ตามความน่าจะเป็น พบว่า 0.0095 อยู่ในช่วง $[0.0077,0.0097)$ ซึ่งเป็นช่วงของสัญลักษณ์ U ซึ่งจะได้ตัวอักษรตัวที่สี่คือ U จากนั้นทำการลดขอบเขต $[0.0034,0.0113)$ ลงเป็น $[0.0077,0.0097)$

5. ทำการแบ่งช่วง $[0.0077,0.0097)$ ตามความน่าจะเป็น พบว่า 0.0095 อยู่ในช่วง $[0.0077,0.0097)$ ซึ่งเป็นช่วงของสัญลักษณ์ !ซึ่งจะได้ตัวอักษรตัวที่ห้าคือ !



รูปที่ 2.12 การแบ่งขอบเขตตามข้อมูลที่เพิ่มขึ้นในการเข้ารหัสเลขคณิต

การเข้ารหัสแบบเลมเปล-ซิป (Lempel-Ziv Code)

การเข้ารหัสแบบฮัฟแมนเป็นวิธีหนึ่งที่มีประสิทธิภาพ แต่ใช้ได้ดีกับข้อมูลแบบที่ไม่มีความจำที่ไม่ต่อเนื่อง (Discrete Memoryless Source) ซึ่งจำเป็นที่ต้องทราบค่าของความน่าจะเป็นของข้อมูลเหล่านั้นด้วย แต่ถ้าในกรณีของข้อมูลแบบที่มีหน่วยความจำ (Discrete Memory Source) นั้นเราจำเป็นต้องทราบความน่าจะเป็นร่วม (joint probabilities) แต่อย่างไรในทางปฏิบัติไม่สามารถที่จะทราบได้ค่าเหล่านี้ได้

การเข้ารหัสแบบเลมเปล-ซิป (Lempel-Ziv) เป็นวิธีการหนึ่งในรูปแบบของวิธีการเข้ารหัสยูนิเวอร์แซล (Universal Source Code Algorithm) ซึ่งเป็นการบีบอัดข้อมูลโดยที่ไม่รู้ข้อมูลทางสถิติ การเข้ารหัสแบบเลมเปล-ซิปนั้นลำดับข้อมูลของแหล่งข้อมูลไม่ต่อเนื่องจะถูกแบ่งออกเป็นบล็อกที่มีความยาวหลายๆขนาดและเรียกบล็อกข้อมูลนี้ว่ากลุ่มรหัส (phrases) [19] กลุ่มรหัสใหม่สร้างขึ้นทุกๆช่วงเวลา ซึ่งบล็อกใดบล็อกหนึ่งของจำนวนตัวอักษรจากแหล่งข้อมูลจะแตกต่างจากกลุ่มรหัสก่อนหน้านั้นในอักษรตัวสุดท้าย กลุ่มรหัสจะถูกจัดให้เป็นลำดับรายการในรูปแบบของปทานุกรม (dictionary) ในการเข้ารหัสกลุ่มรหัสตัวใหม่เราสามารถระบุตำแหน่งของกลุ่มรหัสที่มีอยู่แล้วในปทานุกรมแต่นำตัวอักษรใหม่มาต่อได้เลย ตัวอย่างเช่นมีข้อมูลดังนี้

101011010010011101010000110011101011000110 11

เมื่อทำการแบ่งออกเป็นกลุ่มจะได้ดังนี้

1, 0, 10, 11, 01, 00, 100, 111, 010, 1000, 011, 001, 110, 101, 10001, 1011

สังเกตได้ว่าแต่ละกลุ่มจะมีความต่อเนื่องกัน เพื่อทำการเข้ารหัสกลุ่มรหัสเหล่านี้ เราจะต้องสร้างปทานุกรมดังตารางที่ 2.3 ในที่นี้ตำแหน่งของปทานุกรมจะถูกกำหนดให้ตัวเลขมีความต่อเนื่องกันเป็นลำดับซึ่งเริ่มจาก 1 และเพิ่มขึ้นเรื่อยๆ จนถึง 16 ซึ่งก็คือจำนวนของกลุ่มรหัสนั่นเอง ในลำดับข้อมูลนั้น คำรหัส (codewor d) ถูกกำหนดโดยรายการของตำแหน่งของปทานุกรมก่อนหน้าซึ่งตรงกับกลุ่มรหัสกลุ่มใหม่ในทุกๆตำแหน่งสุดท้ายของคำรหัส แล้วตัวอักษรเอาท์พุทค่าใหม่นำมาต่อท้ายเข้ากับตำแหน่งของปทานุกรมของกลุ่มรหัสก่อนหน้า ค่าเริ่มต้น 0000 ถูกนำมาใช้ในการเข้ารหัสเป็นอันดับแรก

ทางด้านรับจะมีปทานุกรมเหมือนกับทางด้านส่ง และทางด้านรับสามารถถอดรหัสตามลำดับข้อมูลที่ส่งมา จากขั้นตอนการเข้ารหัสจะเห็นได้ว่าจำนวนบิตที่นำมาเข้ารหัสมีทั้งหมด 44 บิต และเมื่อทำการแบ่งกลุ่มจะได้เป็น 16 กลุ่ม โดยแต่ละคำรหัสจะใช้บิตทั้งหมด 5 บิตในที่นี้จะมีรหัสทั้งหมด 80 บิต ดังนั้นวิธีการดังกล่าวนี้เสมือนว่าไม่ได้เป็นวิธีการลดจำนวนข้อมูลแต่อย่างใด เนื่องจากเพราะว่าข้อมูลที่เราพิจารณามีจำนวนน้อยแต่ถ้าในกรณีที่ข้อมูลมีจำนวนมากวิธีการเข้ารหัสจะมีประสิทธิภาพมาก การเข้ารหัสแบบเลมเปล-ซิป เป็นวิธีการที่ได้นำมาใช้ในการเข้ารหัสไฟล์ข้อมูลในคอมพิวเตอร์

ตารางที่ 2.3 ปทานุกรมสำหรับวิธีการของเลมเปล-ซิป

ลำดับที่	ตำแหน่งปทานุกรม	กลุ่มข้อมูล	คำรหัส
1	0001	1	00001
2	0010	0	00000
3	0011	10	00010
4	0100	11	00011
5	0101	01	00101
6	0110	00	00100
7	0111	100	00100
8	1000	111	01001
9	1001	010	01010
10	1010	1000	01110
11	1011	011	01011
12	1100	001	01101
13	1101	110	01000
14	1110	101	00111
15	1111	10001	10101
16		1011	11101

การเข้ารหัสระนาบบิต (Bit-Plane coding)

ความสำคัญของการลดข้อมูลคือการกำจัดความซ้ำซ้อนของข้อมูล วิธีการเข้ารหัสระนาบบิต[20] เป็นวิธีการหนึ่งที่อยู่บนพื้นฐานของแนวความคิดของการแยกองค์ประกอบที่มีหลายระดับ (ภาพสีหรือภาพขาวดำ) ออกเป็นลำดับของภาพที่เป็นเลขไบนารี และทำการลดข้อมูลแต่ละภาพไบนารีดังกล่าวด้วยวิธีการลดข้อมูลทั่วๆไปที่เป็นการลดข้อมูลไบนารี

การแยกองค์ประกอบของระนาบบิต

ภาพที่มีระดับสีเทาและมีขนาด m บิตสามารถที่จะแสดงในรูปแบบของโพลีโนเมียลเลขฐาน สองได้ดังนี้

$$a_{m-1}2^{m-1}, a_{m-2}2^{m-2}, \ldots, a_1 2^1, a_0 2^0 \tag{2.72}$$

บนพื้นฐานของคุณสมบัติดังกล่าวนี้ วิธีที่ง่ายที่สุดของการแยกองค์ประกอบภาพให้เป็นภาพในรูปแบบไบนารีคือการแยกสัมประสิทธิ์ของโพลีโนเมียล m ไปเป็นระนาบบิตในระดับที่ m โดยที่ระนาบบิตศูนย์ถูกกำหนดขึ้นโดยเลือกเอาบิตที่ a_0 ของแต่ละจุดภาพ ขณะที่ระดับระนาบบิต $(m-1)$ ประกอบไปด้วยบิตหรือสัมประสิทธิ์ที่ $a_{(m-1)}$ โดยทั่วไปแต่ละระนาบบิตจะกำหนดตัวเลขไว้จาก 0 ถึง $(m-1)$และถูกสร้างขึ้นโดยกำหนดให้จุดภาพของระนาบนั้นๆเท่ากับค่าของบิตที่เหมาะสมหรือสัมประสิทธิ์โพลีโนเมียลจากแต่ละจุดภาพในภาพต้นแบบ

วิธีการแยกองค์ประกอบเริ่มด้วยให้รหัสของภาพเป็นรหัสเกรย์ (gray code) ที่มี m บิตคือ $g_{m-1}\cdots g_2 g_1 g_0$ ซึ่งสอดคล้องกับโพลีโนเมียลตามสมการที่ (2.72) ซึ่งคำนวณได้ดังนี้

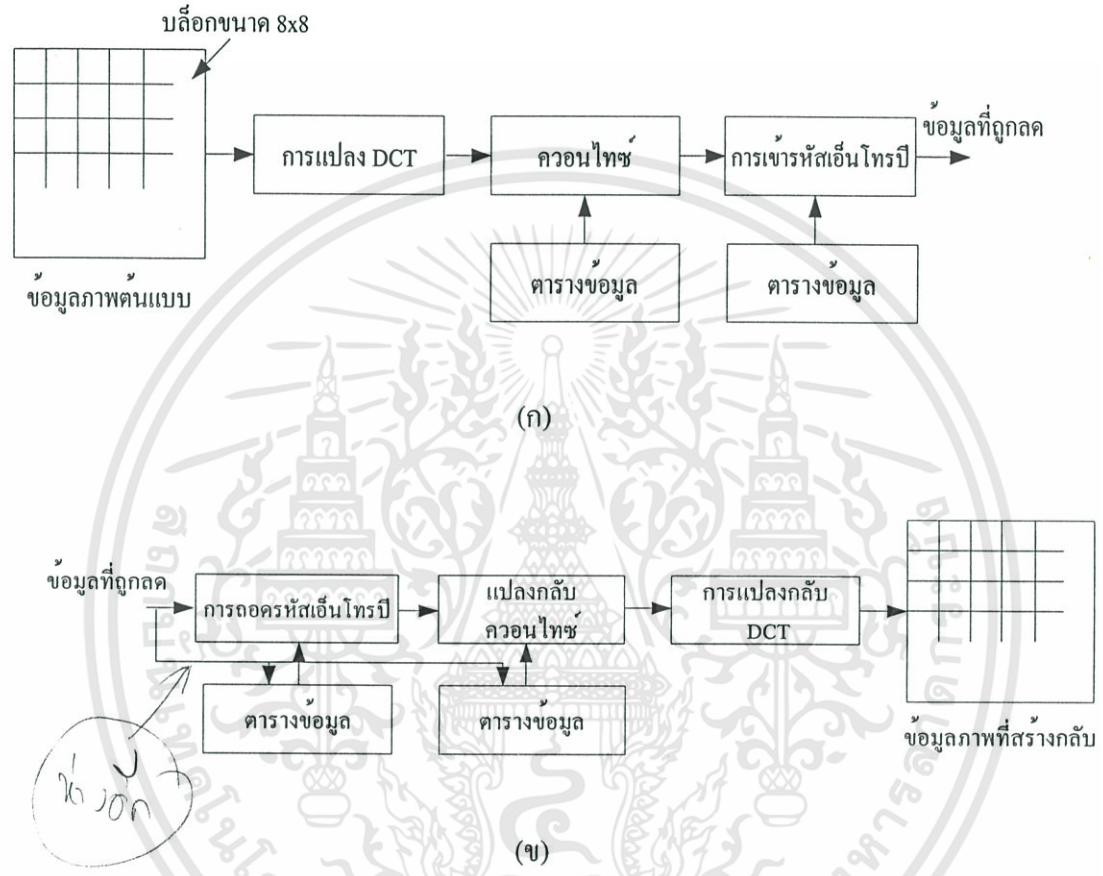
$$\begin{aligned} g_i &= a_i \oplus a_{i+1}, \qquad 0 \le i \le m-2 \\ g_{m-1} &= a_{m-1} \end{aligned} \tag{2.73}$$

โดยที่ $\oplus$ คือการกระทำเอ็กครูซีฟออ exclusive OR รหัสนี้มีคุณสมบัติอย่างหนึ่งคือคำรหัส (เลขไบนารี) จะมีความแตกต่างกันเพียงแค่หนึ่งตำแหน่ง ซึ่งทำให้มีผลกระทบน้อยเมื่อแบ่งออกเป็นภาพไบนารี เช่น ในระดับสีเทา 127 (0111111) และ 128 (1000000) อยู่ติดกันซึ่งปกติแล้วจะแตกต่างกันทุกตำแหน่ง แต่ถ้าใช้วิธีนี้มีเพียงระนาบในระดับที่ 7 เท่านั้นที่มีบิต 0 และ 1 เพราะว่ามีบิตที่แตกต่างกันเพียงตำแหน่งเดียวคือรหัสเกรย์ (gray code) ของข้อมูลของ 127 และ 128 มีค่าเป็น 11000000 และ 01000000 ตามลำดับ

การบีบอัดข้อมูลภาพแบบมาตรฐาน JPEG

การบีบอัดข้อมูลภาพแบบ JPEG [26] เป็นการเข้ารหัสแบบมาตรฐานและเป็นวิธีที่รับรองโดยคณะกรรมการกำหนดมาตรฐานสากล ISO และ CCITT การบีบอัดข้อมูลภาพแบบJPEG ใช้ขั้นตอนในการบีบอัดข้อมูลภาพโดยขั้นแรกข้อมูล สีแดง สีเขียว และ สีน้ำเงิน ในระบบ RGB ของภาพที่ต้องการบีบอัดข้อมูลภาพ จะถูกแปลงให้อยู่ในค่าของระบบ YUV ประกอบด้วยค่าระดับความสว่างของสี (luminance) หนึ่งค่าและค่าข้อมูลของแม่สีที่ประกอบกันเป็นสีนั้นๆ 2 ค่า (Chrominance) ในการเก็บค่า ระดับความสว่างของสีจะเก็บข้อมูลระดับของสีเทา (Gray-Scale) อยู่ในค่า Y ส่วนข้อมูลของ chrominance อยู่ในค่า U และ V และใช้วิธีการแปลงโคซายน์เต็มหน่วย (Discrete Cosine Transform) ในการแปลงข้อมูลค่าเฉลี่ย YUV ของบล็อกแต่ละบล็อกที่แบ่ง

ออกเป็นขนาด 8×8 จุดภาพให้อยู่ในรูปของอะเรย์ขนาด 2 มิติ ขั้นตอนต่อมาจะใช้เทคนิคการแซมเปิลในการแปลงอะเรย์ 2 มิติไปสู่อะเรย์ 1 มิติและใช้เทคนิคการเข้ารหัสแบบฮัฟแมนที่ได้กล่าวไปแล้ว ในการบีบอัดข้อมูลภาพอีกทีหนึ่งเป็นขั้นตอนสุดท้าย การเข้ารหัส JPEG แสดงไว้ดังรูปที่ 2.13



รูปที่ 2.13 การบีบอัดข้อมูลภาพแบบ JPEG

2.7 การวัดประสิทธิภาพของการบีบอัดข้อมูลภาพ

ในการบีบอัดข้อมูลภาพนั้นวิธีการหนึ่งที่จะนำมาเปรียบเทียบว่าวิธีการใดมีประสิทธิภาพมาก กว่าอีกวิธีการหนึ่ง นอกจากจะเป็นในเรื่องของความเร็วแล้วยังมีปัจจัยอื่นๆอีก โดยทั่วไปแล้วมีวิธีการวัดประสิทธิภาพของภาพที่ได้ทำการสร้างกลับมาด้วยกันดังนี้

2.7.1 การวัดค่าผิดพลาดกำลังสองเฉลี่ย

การวัดประสิทธิภาพด้วยวิธีนี้ จะใช้ค่าความผิดพลาดกำลังสองเฉลี่ยมาเป็นตัวเปรียบเทียบประสิทธิภาพของการลดข้อมูล โดยที่ถ้าค่าความผิดพลาดกำลังสองเฉลี่ยมีค่ามากหมายความว่าภาพที่ได้ทำการสร้างกลับมามีความผิดเพี้ยนมาก แต่ในทางกลับกันถ้าค่าความผิดพลาดกำลังสองเฉลี่ยมีค่าน้อยซึ่งหมายความว่าภาพที่สร้างกลับมามีประสิทธิภาพดี ในที่นี่เราสามารถหาค่าความผิดพลาดกำลังสองเฉลี่ยได้ดังนี้

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[x(i,j)-\hat{x}(i,j)]^2 \tag{2.74}$$

โดยที่ M และ N คือ จำนวนจุดภาพในแนวนอนและแนวตั้งทั้งหมดของภาพ $x(i,j)$ และ $\hat{x}(i,j)$ คือจุดภาพที่ตำแหน่ง (i,j) ของภาพต้นแบบและภาพที่ได้ทำการสร้างกลับตามลำดับ

2.7.2 การค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนและค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด

SNR และ *PSNR* (signal to noise ratio and peak signal to noise ratio)

SNR คือ ค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนบางครั้งเรียกว่าค่าสัญญาณต่อสัญญาณรบกวนต่อการควอนไทซ์

$$SNR = 10\log_{10}\frac{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[x(i,j)]^2}{MSE} \tag{2.75}$$

โดยที่ *MSE* (Mean Square Error) คือค่าผิดพลาดกำลังสองเฉลี่ยหาได้จากสมการที่ (2.74)

PSNR คือค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดมีหน่วยเป็นเดซิเบล(dB)

$$PSNR = 10\log_{10}\frac{P^2}{MSE} = 10\log_{10}\frac{(2^n-1)^2}{MSE} \tag{2.76}$$

โดยที่ P ในที่นี้คือแทนค่าสูงสุดของขอบเขตของข้อมูล

n คือจำนวนบิตที่ใช้แทนจุดภาพ เช่น ภาพที่ระดับเป็นสีเทาจะเท่ากับ 8 บิตต่อจุดภาพ

2.7.3 อัตราบิตและอัตราการลดข้อมูล

อัตราบิต (bit rate) คือค่าเฉลี่ยของจำนวนบิตต่อจุดภาพ (bit per pixel) ของภาพที่ถูกลดข้อมูลโดยที่

$$\text{อัตราบิต} = \frac{\text{จำนวนบิตทั้งหมดของภาพที่ถูกลดข้อมูล}}{\text{จำนวนจุดภาพทั้งหมดของภาพต้นแบบ}} \tag{2.77}$$

อัตราการลดข้อมูล (Compression ratio) คืออัตราส่วนระหว่างจำนวนบิตข้อมูลที่ใช้แทนภาพต้นแบบกับจำนวนบิตข้อมูลที่ใช้แทนภาพที่ถูกทำการลดข้อมูลโดย

$$\text{อัตราการลดข้อมูล} = \frac{\text{จำนวนบิตข้อมูลของภาพต้นแบบ}}{\text{จำนวนบิตข้อมูลภาพที่ถูกลดข้อมูล}} \tag{2.78}$$

บทที่ 3
การเข้ารหัสแบบบล็อก

3.1 วิธีการเข้ารหัสแบบบล็อกที่มีขนาดคงที่ (Block Truncation Coding "BTC")

วิธีการเข้ารหัสแบบแบ่งข้อมูลภาพเป็นบล็อกที่มีขนาดคงที่ [1, 2, 3, 4, 5] เป็นวิธีการที่ใช้ในการลดข้อมูลภาพวิธีหนึ่ง ซึ่งวิธีดังกล่าวนี้เป็นวิธีการการลดข้อมูลแบบที่มีการสูญเสียข้อมูล (lossy) การเข้ารหัสแบบบล็อกนี้จุดสำคัญคือการเก็บรักษาค่าโมเมนต์ (moment preserving [MP] [2]) ของการควอนไทซ์ของบล็อก วิธีการเริ่มจากแบ่งภาพที่มีขนาด $N \times N$ ออกเป็นบล็อกสี่เหลี่ยมจัตุรัสที่มีขนาด $n \times n$ (ปกติจะเป็นขนาด 4×4)และแต่ละจุดภาพภายในบล็อกข้อมูลที่แบ่งจะถูกควอนไทซ์โดยใช้การควอนไทซ์เพียงสองระดับ ("0" และ "1") วิธีการแบ่งข้อมูลออกเป็นสองส่วนทำได้โดยกำหนดค่าเทรชโฮล (threshold) X_{th} วิธีการของเข้ารหัสแบบแบ่งข้อมูลเป็นบล็อกนี้ หลังจากที่ได้มีการแบ่งบล็อกเรียบร้อยแล้วจากนั้นจะทำการหาค่าเฉลี่ย $\overline{X}$ และค่าความเบี่ยงเบนมาตรฐาน $\bar{\sigma}$ ของแต่ละบล็อกที่แบ่งออกมา ถ้าสมมติให้จุดภาพเป็น x_i เมื่อ $i = 1, \ldots, m$ นั่นคือ

$$\overline{X} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3.1}$$

$$\overline{X^2} = \frac{1}{m}\sum_{i=1}^{m} x_i^2 \tag{3.2}$$

และสามารถหาค่าความเบี่ยงเบนมาตรฐานได้โดย

$$\bar{\sigma} = \sqrt{\overline{X^2} - \overline{X}^2} \tag{3.3}$$

ขั้นตอนของการเข้ารหัสแสดงดังรูปที่ 3.1 หลังจากที่ทำการเข้ารหัสแล้วทางด้านส่งจะส่งข้อมูลบิตที่เข้ารหัสค่าเฉลี่ยรวมถึงบิตที่เข้ารหัสค่าเบี่ยงเบนมาตรฐานและบิตที่แสดงค่าของแต่ละจุดภาพ (ในที่นี้คือบิต "0" และบิต "1" ที่ใช้แทนแต่ละจุดภาพภายในบล็อกข้อมูล) ในที่นี้เราสามารถคำนวณค่าอัตราบิตที่ใช้ในการเข้ารหัสได้ดังนี้

$$\frac{b + b + m}{m} = 1 + \frac{2b}{m} \text{ บิตต่อจุดภาพ} \tag{3.4}$$

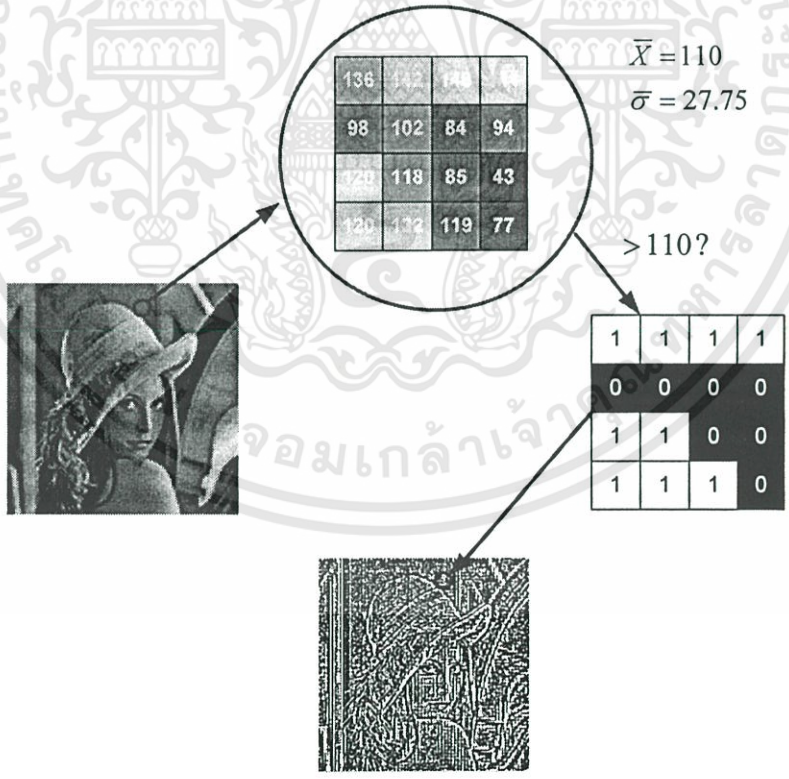
เมื่อ b คือจำนวนบิตที่ใช้ในการเข้ารหัสค่าของค่าเฉลี่ยและค่าความเบี่ยงเบนมาตรฐาน m คือจำนวนบิตทั้งหมดภายในบล็อกภาพ ในที่นี้สามารถคำนวณอัตราบิตที่เข้ารหัสข้อมูลของการเข้ารหัสแบบบล็อกคงที่โดยปกติแบ่งออกเป็นบล็อกขนาด 4×4 ได้ดังนี้

$$\frac{8+8+16}{16}=1+\frac{2*8}{16}=2 \text{ บิตต่อจุดภาพ}$$

การเข้ารหัสแบบบล็อกนี้อัตราบิตจะขึ้นอยู่กับจำนวนบิตที่เข้ารหัสค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานรวมไปถึงขนาดของบล็อกที่แบ่ง ทางด้านรับจะทำการคำนวณค่าของแต่ละจุดภาพเพื่อทำการถอดรหัสและสร้างกลับภาพได้ดังนี้

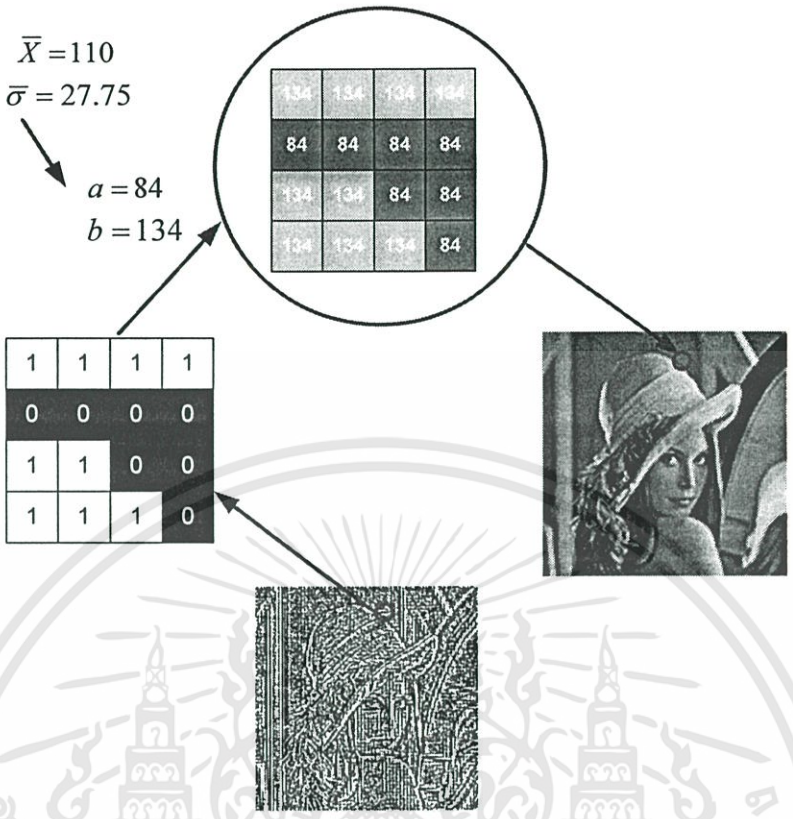
$$a=\overline{X}-\overline{\sigma}\sqrt{\frac{q}{m-q}} \quad \text{สำหรับจุดภาพ } x_i < X_{th} \tag{3.5}$$

$$b=\overline{X}+\overline{\sigma}\sqrt{\frac{m-q}{q}} \quad \text{สำหรับจุดภาพ } x_i \geq X_{th} \tag{3.6}$$



(ก)

รูปที่ 3.1 การเข้ารหัสแบบบล็อกขนาดคงที่ : (ก) ขั้นตอนการเข้ารหัสแบบบล็อก, (ข) ขั้นตอนการถอดรหัสแบบบล็อก

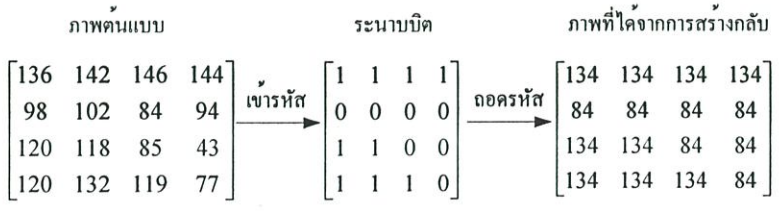


(ข)

รูปที่ 3.1 (ต่อ)

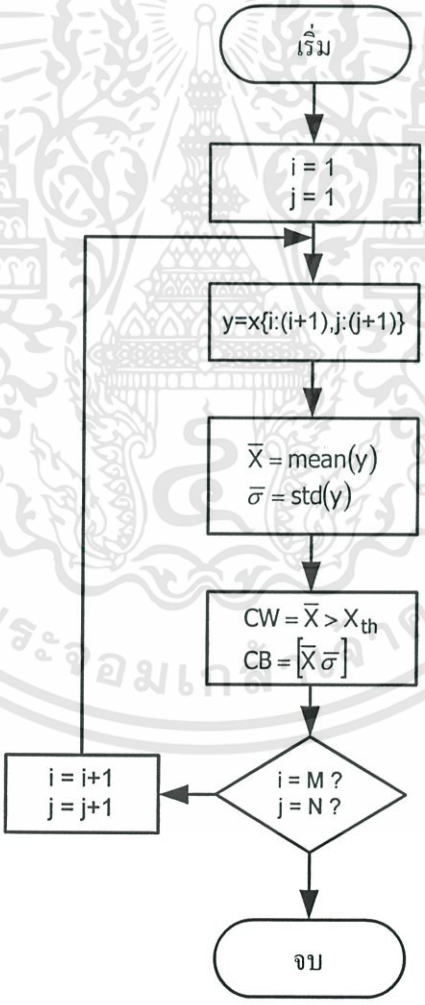
เมื่อ a และ b คือค่าระดับที่ได้จากการสร้างกลับภาพสำหรับแทนแต่ละบิตที่เป็น " 0 " หรือ " 1 " ของรหัสที่ส่งมาจากด้านส่งโดยระดับของ a จะแทนค่าเมื่อรหัสที่ส่งมาเป็น " 0 " และระดับของ b จะแทนค่าเมื่อรหัสที่ส่งมาเป็น "1 " ของข้อมูลแต่ละบล็อก ส่วนค่าของ q คือจำนวนของจุดภาพที่มีค่ามากกว่าหรือเท่ากับค่าเทรชโฮล X_{th} ซึ่งในที่นี้ก็คือค่าตัวเลขที่เป็นเลข "1 " นั่นเอง และค่าของ m คือค่าของจำนวนจุดภาพทั้งหมดภายในแต่ละบล็อก

สำหรับตัวอย่างสมมติว่าเราต้องการที่จะทำการเข้ารหัสโดยใช้วิธีนี้กับรูปภาพที่มีขนาดระดับสีเป็นระดับทั้งหมด 256 ระดับ และแต่ละบล็อกจะแบ่งออกเป็นขนาด 4x4 ในที่นี้เราสมมติว่าเราเลือกเอาบล็อกๆหนึ่งขึ้นมาโดยเป็นดังนี้

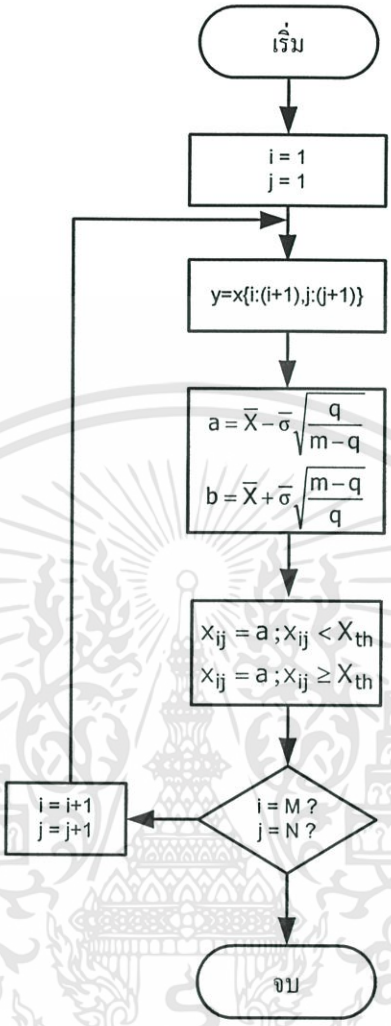


รูปที่ 3.2 การแทนที่แต่ละพิกเซลด้วยจุดภาพที่มีสองระดับ

ในขั้นตอนแรกนั้นจะต้องแบ่งข้อมูลภาพออกเป็นสองกลุ่ม สมมติว่าให้กลุ่มที่หนึ่งที่ทำการเลือกมานี้ประกอบไปด้วยค่าของแต่ละจุดภาพที่มีค่ามากกว่าค่าเทรชโฮลที่เราได้กำหนดและให้กลุ่มที่สองเป็นกลุ่มที่ประกอบด้วยจุดภาพที่มีค่าของจุดภาพนั้นน้อยกว่าค่าเทรชโฮลที่ได้กำหนด ค่าเทรชโฮลที่เราใช้นี้จะได้มาจากการหาค่าเฉลี่ยของจุดภาพทั้งหมดภายในบล็อกที่เราได้เลือกมา (ในที่นี้คือมีทั้งหมด 16 จุดภาพ) ในที่นี้ค่าเทรชโฮลจะได้เป็น 110 จากจุดภาพทั้งหมดที่แสดงข้างบน ในที่นี้ถ้าเรามองจากบล็อกของข้อมูลที่แสดงจะเห็นว่าที่ตำแหน่งของจุดภาพที่ 1, 2, 3, 4, 9, 10, 13, 14 และ15 จะอยู่ในกลุ่มที่หนึ่งซึ่งมีค่าที่มากกว่าค่าเทรชโฮลและจากนั้นเราจะนำเอาค่าที่แต่ละจุดภาพที่มีค่ามากกว่าค่าเทรชโฮลจะถูกแทนด้วยเลข “1” และค่าที่น้อยกว่าค่าเทรชโฮลจะถูกแทนด้วยเลข “0” จากนั้นทำการหาค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานเพื่อส่งไปยังด้านรับโดยที่ขั้นตอนการเข้ารหัสและการถอดรหัสเป็นดังรูปที่ 3.3 และ 3.4 ตามลำดับ



รูปที่ 3.3 ขั้นตอนการเข้ารหัสแบบบล็อกคงที่



รูปที่ 3.4 ขั้นตอนการถอดรหัสแบบบล็อกคงที่

3.2 การเข้ารหัสแบบบล็อกที่มีขนาดคงที่โดยใช้ค่าสมบูรณ์โมเมนต์ (Absolute Moment Block Truncation Coding)

การเข้ารหัสแบบบล็อกที่มีขนาดคงที่โดยใช้ค่าสมบูรณ์โมเมนต์ (AMBTC) นี้ได้นำเสนอโดย Lema, M. D. and Mitchell [5] การเข้ารหัสสามารถทำได้เช่นเดียวกันกับการเข้ารหัสแบบบล็อกคงที่ (BTC) เพียงแต่คำนวณหาค่าเฉลี่ยและตัวอย่างแรกของค่าสมบูรณ์ของโมเมนต์ (first absolute central moment) แทนที่จะคำนวณหาค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐาน โดยค่าสมบูรณ์ของโมเมนต์สามารถคำนวณได้ดังนี้

$$\overline{\alpha} = \frac{1}{m}\sum_{i=1}^{m}\left|x_i - \overline{X}\right| \tag{3.7}$$

และการสร้างภาพกลับสามารถคำนวณหาค่าของ a และ b ได้ดังนี้

$$a = \overline{X} - \frac{\overline{\alpha}}{2}\cdot\frac{m}{m-q} \tag{3.8}$$

$$b = \overline{X} + \frac{\overline{\alpha}}{2}\cdot\frac{m}{q} \tag{3.9}$$

ค่าของ m และ q จะเหมือนกันกับที่ใช้ในการคำนวณในสมการที่ (3.5) และ (3.6) โดยประสิทธิภาพของวิธีการนี้สามารถลดอัตราบิตที่นำมาเข้ารหัสได้ในที่นี้คือ ลดจำนวนบิตที่นำมาเข้ารหัสค่าของ $\overline{\alpha}$ อีกทั้งวิธีนี้ยังลดเวลาในการเข้ารหัสและถอดรหัสได้เพราะวิธีการนี้ไม่ได้ใช้ส่วนของการคำนวณทางคณิตศาสตร์ที่เป็นส่วนของราก และ การคูณเหมือนในการเข้ารหัสแบบบล็อกดั้งเดิม [2]

3.3 การเข้ารหัสแบบบล็อกโดยการควอนไทซ์สามระดับ(Three Level Block Truncation Code)

การเข้ารหัสแบบบล็อกโดยการควอนไทซ์สามระดับ [1] ในที่นี้การเข้ารหัสจะใช้การควอนไทซ์สามระดับแทนที่จะใช้สองระดับ เนื่องจากว่าการควอนไทซ์สองระดับนั้นถ้าข้อมูลมีความแตกต่างกันมากๆ หรือช่วงของข้อมูลมีมากทำให้การควอนไทซ์ที่ใช้เพียงสองระดับอาจทำให้เกิดผลของการเหลื่อมล้ำของสีของบล็อกที่อยู่ใกล้เคียง (Blocking effect) และนอกจากนั้นยังทำให้การสร้างกลับภาพมีความผิดเพี้ยนสูงด้วย แต่ในการเข้ารหัสแบบบล็อกโดยการควอนไทซ์สามระดับนี้มีข้อด้อยกว่าการใช้การควอนไทซ์สองระดับคือการใช้บิตเพิ่มขึ้นหรือพูดอีกอย่างได้ว่าจำนวนบิตต่อจุดภาพที่ใช้สูงนั่นเอง

การเข้ารหัสเริ่มโดยการแบ่งข้อมูลภาพออกเป็นบล็อกขนาด $N \times N$ เช่นเดียวกับการเข้ารหัสแบบบล็อกทั่วไปดังนี้

$$\begin{bmatrix} 128 & 255 & 255 & 255 \\ 0 & 128 & 255 & 255 \\ 0 & 0 & 128 & 255 \\ 0 & 0 & 0 & 128 \end{bmatrix}$$

จากนั้นทำการหาค่าช่วง R ของข้อมูลภายในบล็อก ค่าเฉลี่ยที่มากที่สุดภายในบล็อกและค่าเฉลี่ยที่น้อยที่สุดคือ M_h และ M_l ตามลำดับโดยสามารถคำนวณได้ดังนี้

$$R = \max\{f(i,j)\} - \min\{f(i,j)\} \tag{3.10}$$

$$M_h = \max\{f(i,j)\} - R/3 \tag{3.11}$$

$$M_l = \min\{f(i,j)\} - R/3 \tag{3.12}$$

$\max\{f(i,j)\}$ = ค่าของพิกเซลที่มีค่ามากที่สุดภายในบล็อก

$\min\{f(i,j)\}$ = ค่าของพิกเซลที่มีค่าน้อยที่สุดภายในบล็อก

$$R = 255 - 0 = 255 \tag{3.13}$$

$$M_h = 255 - 255/3 = 170 \tag{3.14}$$

$$M_l = 0 - 255/3 = 85 \tag{3.15}$$

จากนั้นทำการแทนค่าข้อมูลภายในบล็อกดังนี้

- พิกเซลที่มีค่ามากกว่าหรือเท่ากับ M_h แทนค่าด้วยเลข 2
- พิกเซลที่มีค่าน้อยกว่าหรือเท่ากับ M_l แทนค่าด้วยเลข 0
- ช่วงระหว่าง M_h และ M_l แทนค่าด้วยเลข 1

หลังจากที่แทนด้วยตัวเลขดังกล่าวแล้วจะได้เป็น

$$\begin{bmatrix} 1 & 2 & 2 & 2 \\ 0 & 1 & 2 & 2 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

บล็อกที่ได้ถูกเข้ารหัสเรียบร้อยแล้วจะส่งเพียงค่า $\max\{f(i,j)\}$ และ $\min\{f(i,j)\}$ ไปยังด้านรับ ที่ทางด้านรับจะทำการคำนวณ R , M_h และ M_l ใหม่อีกครั้ง ซึ่งคำนวณได้จาก (3.10) (3.11)และ (3.12) และสามารถสร้างภาพกลับได้โดย

$$b = \lfloor (M_h + \max\{f(i,j)\})/2 \rfloor \tag{3.16}$$

$$a = \lfloor (M_l + \min\{f(i,j)\})/2 \rfloor \tag{3.17}$$

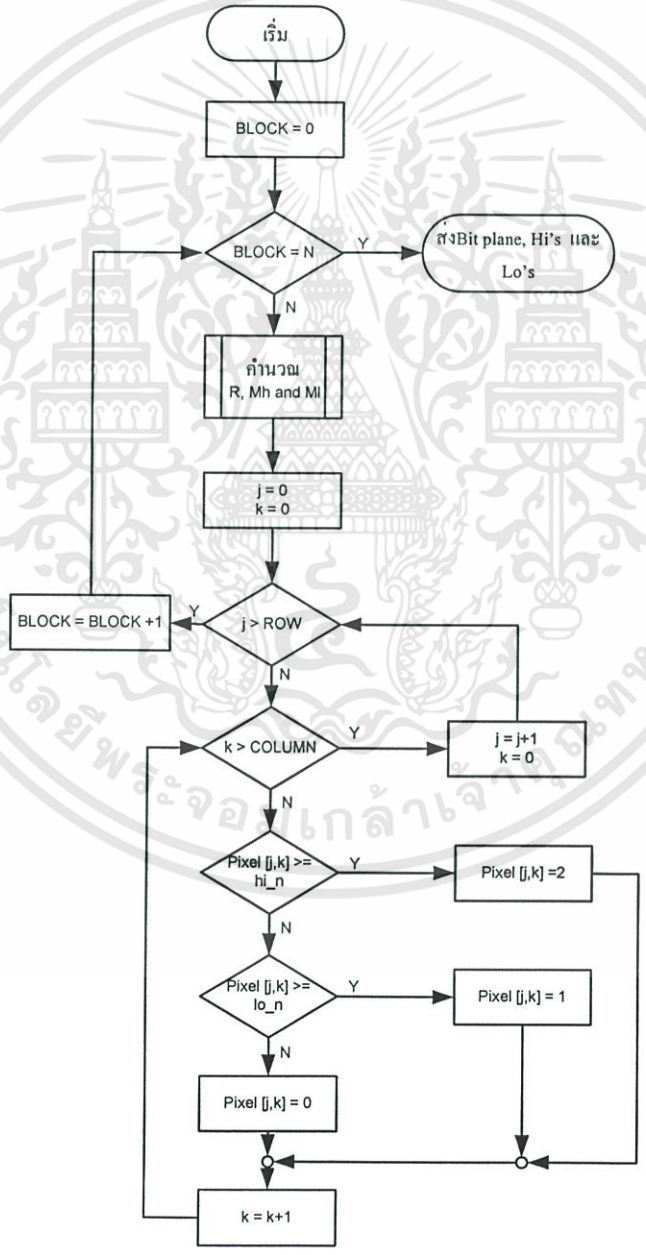
เมื่อ b และ a คือค่าพิกเซลที่ถูกกำหนดให้สำหรับพิกเซลที่ถูกควอนไทซ์แล้วแทนด้วยเลข 2 และ 0 ตามลำดับ ส่วนพิกเซลที่ถูกแทนด้วย "1" นั้นสามารถคำนวณเพื่อสร้างกลับข้อมูลภาพได้โดย

$$mid = \lfloor (M_h + M_l)/2 \rfloor \tag{3.18}$$

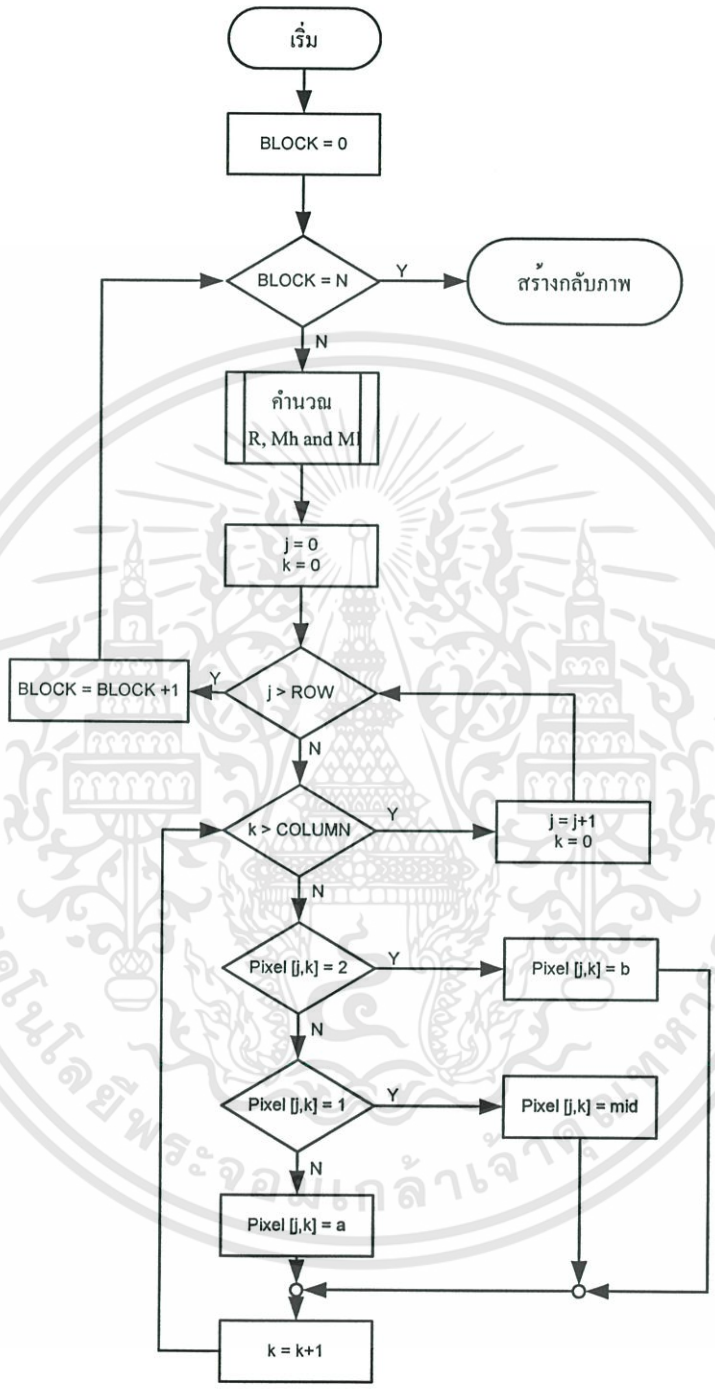
ข้อมูลภาพที่ได้จากการสร้างกลับจะได้เป็น

$$\begin{bmatrix} 127 & 212 & 212 & 212 \\ 42 & 127 & 212 & 212 \\ 42 & 42 & 127 & 212 \\ 42 & 42 & 42 & 127 \end{bmatrix}$$

ขั้นตอนการเข้ารหัสแบบบล็อกโดยการควอนไทซ์สามระดับสามารถเขียนเป็นแผนภาพแสดงขั้นตอนการเข้ารหัสได้ดังนี้



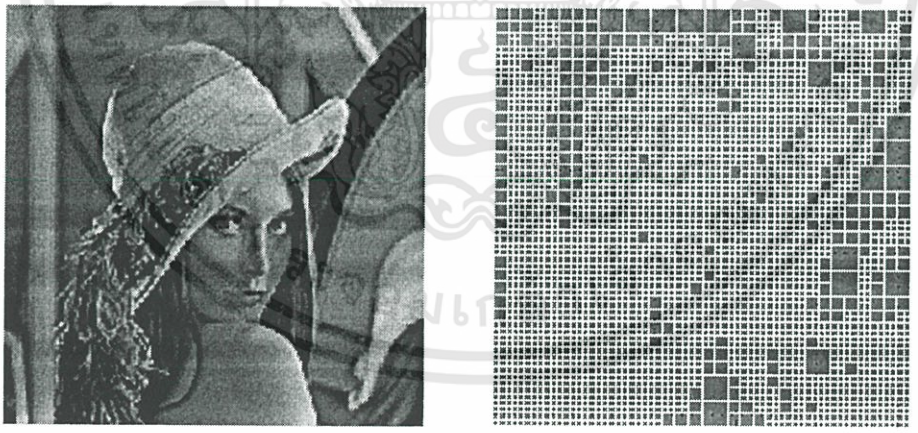
รูปที่ 3.5 ขั้นตอนการเข้ารหัสแบบบล็อกโดยการควอนไทซ์สามระดับ



รูปที่ 3.6 ขั้นตอนการถอดรหัสแบบบล็อกโดยการควอนไทซ์สามระดับ

3.4 การเข้ารหัสแบบแบ่งข้อมูลภาพออกเป็นบล็อกหลายขนาด (Variable Block Truncation Coding)

การเข้ารหัสแบบแบ่งข้อมูลภาพออกเป็นหลายขนาด [3, 7, 12, 13, 14, 15, 16, 17] โดยการประยุกต์นำเอาวิธีการของการแยกองค์ประกอบแบบควอดทรี [8] และการเข้ารหัสแบบบล็อกซึ่งได้กล่าวมาแล้วในหัวข้อที่ 3.1 ซึ่งวิธีการนี้เริ่มจากการแบ่งบล็อกออกเป็นหลายขนาดด้วยวิธีการแยกองค์ประกอบแบบควอดทรี จากนั้นจะได้บล็อกที่มีหลายขนาดพร้อมด้วยต้นไม้รหัสที่บ่งบอกถึงโครงสร้างทั้งหมดขององค์ประกอบภาพ ในที่นี้ส่วนของภาพถูกแสดงโดยส่วนที่เป็นโหนดใบไม้นั่นเอง ข้อมูลภายในแต่ละบล็อกนั้นจะมีความเป็นเนื้อเดียวกันเมื่อเปรียบเทียบกับการแบ่งบล็อกแบบที่มีขนาดคงที่ การกำหนดว่าบล็อกจะต้องแบ่งต่อไปหรือไม่นั้นขึ้นอยู่กับการกำหนดค่าของเทรชโฮล ซึ่งใช้ค่าของความเบี่ยงเบนมาตรฐานเปรียบเทียบกับค่าโทรชโฮล [7] โดยถ้าค่าเบี่ยงเบนมาตรฐานน้อยกว่าค่าเทรชโฮล บล็อกนั้นก็ไม่จำเป็นที่ต้องแบ่งบล็อกนั้นต่อไป แต่ถ้าค่าเบี่ยงเบนมาตรฐานมากกว่าค่าโทรชโฮลบล็อกดังกล่าวจะต้องแบ่งออกเป็นบล็อกเท่าๆกันสี่บล็อก วิธีการจะทำต่อเนื่องไปจนกว่าทุกๆบล็อกมีค่าเบี่ยงเบนมาตรฐานน้อยกว่าค่าเทรชโฮล หลังจากทำการแบ่งบล็อกได้แล้วส่วนของภาพที่แสดงด้วยองค์ประกอบควอดทรีที่เป็นลีฟโหนดจะทำการเข้ารหัสแบบบล็อกต่อไป

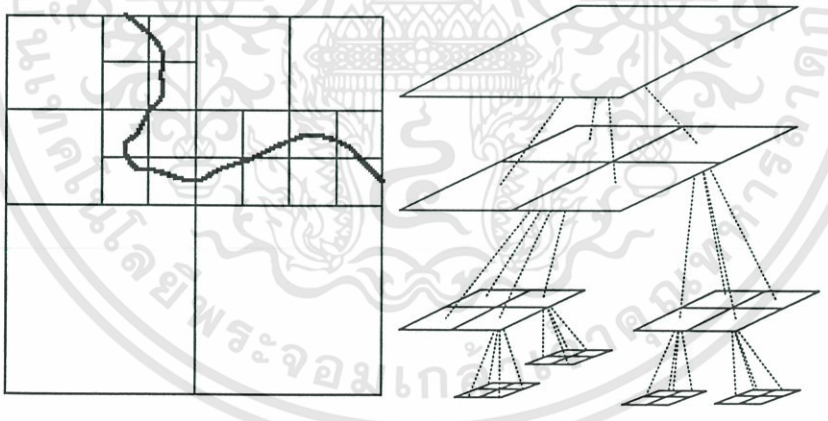


รูปที่ 3.7 แสดงการแบ่งบล็อกของภาพ Lena ออกเป็นบล็อกหลายขนาด

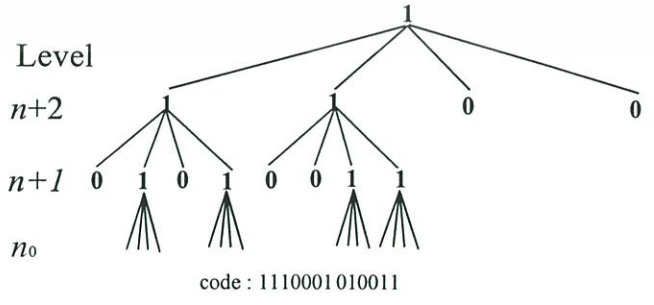
3.5 การเข้ารหัสแบบบล็อกหลายขนาดโดยการแยกองค์ประกอบแบบควอดทรี

การแยกองค์ประกอบแบบควอดทรี (Quadtree Decomposition) วิธีการทำได้โดยกำหนดค่าเทรชโฮลเพื่อทำการแบ่งข้อมูลภาพออกเป็นบล็อกย่อยๆที่มีหลายขนาดแสดงดังรูปที่3.4 การแยก

องค์ประกอบควอดทรีเป็นเทคนิคการวิเคราะห์ภาพอย่างหนึ่งสามารถทำได้โดยแบ่งข้อมูลภาพออกเป็นบล็อกซึ่งในแต่ละบล็อกนั้นภายในจะประกอบด้วยข้อมูลที่มีลักษณะใกล้เคียงกันมากที่สุดหรือเรียกได้อีกอย่างหนึ่งว่ามีความเป็นเนื้อเดียวกันนั่นเอง และด้วยเหตุนี้เองการแยกองค์ประกอบแบบควอดทรี จึงเป็นวิธีการที่สามารถบ่งบอกถึงลักษณะของข้อมูลภาพได้เป็นอย่างดี การแยกองค์ประกอบแบบนี้เป็นรูปแบบของโครงสร้างแบบต้นไม้ [29] ซึ่งในแต่ละโหนดที่ไม่เป็นโหนดใบไม้ (Non leaf node) หรือโหนดภายใน (inner node) นั้นจะมี 4 สาขาย่อยแยกออกมาจากโหนดนั้น การแยกองค์ประกอบภาพนั้นในแต่ละโหนดในควอดทรี จะตรงกันกับบล็อกข้อมูลภาพย่อยๆของภาพที่ทำการแยกองค์ประกอบโดยที่มีการกำหนดขนาดและตำแหน่งที่แน่นอน ซึ่งโหนดลูก (child node) ทั้ง 4 จะได้มากจากการแบ่งจากบล็อกข้อมูลที่อยู่ในระดับของโหนดพ่อ (father node) ออกมาเป็นสี่ส่วนโดยที่แต่ละส่วนมีรูปสี่เหลี่ยมจัตุรัสเท่าๆกันอยู่ในรูปของบล็อกข้อมูลย่อยๆจำนวนสี่บล็อก การแบ่งบล็อกข้อมูลภาพเริ่มจากข้อมูลภาพที่มีขนาด $2^N \times 2^N$ ถูกแบ่งออกได้เป็นระดับทั้งหมด $(N - n_0 + 1)$ เมื่อ n_0 คือเลขยกกำลังของระดับระดับที่เป็นโหนดใบไม้ เมื่อทุกๆบล็อกข้อมูลที่ระดับ n ใดๆมีขนาดบล็อกเท่ากับ $2^n \times 2^n$ โดยที่ $0 \le n_0 \le n \le N$ และแต่ละบล็อกขนาด $2^n \times 2^n$ อาจจะเป็นโหนดใบไม้ (leaf node) หรือไม่นั้นขึ้นอยู่กับถ้าสามารถแบ่งออกเป็นบล็อกขนาด $2^{n-1} \times 2^{n-1}$ ได้แสดงว่ายังไม่เป็นโหนดใบไม้ หรือถ้าไม่สามารถแบ่งต่อไปได้นั้นจะถือว่าเป็นโหนดใบไม้ ที่มีขนาดเท่ากับ $2^{n_0} \times 2^{n_0}$



(ก) (ข)



(ค)

รูปที่ 3.8 แสดงภาพที่ถูกแบ่งเป็นบล็อกที่มีหลายขนาดด้วยวิธีการแยกองค์ประกอบแบบควอดทรี

วิธีการแบ่งบล็อกด้วยการแยกองค์ประกอบควอดทรีมีขั้นตอนดังต่อไปนี้

ขั้นที่ 1 : กำหนดให้ $i=1; N=2^{n-1}$;

ขั้นที่ 2 : สำหรับ $k,l=0,\ldots,N-1$;

ถ้า $j,m=0,1$ all $x_{i-1}(2k+j,2l+m)$ คือโหนดใบไม้

- คำนวณ $x_i(k,l)$ ตามสมการดังนี้

$$x_i(k,l)=\frac{1}{4}\sum_{j=0}^{1}\sum_{m=0}^{1}x_{i-1}(2k+j,2l+m)$$

- ทำการทดสอบดังสมการข้างล่างนี้

$$\bigcap_{j,m=0}^{1}\left|x_i(k,l)-x_{i-1}(2k+j,2l+m)\right|\le T=True$$

ถ้าการทดสอบเป็นจริง , $x_i(k,l)$ คือโหนดใบไม้

ถ้าการทดสอบไม่เป็นจริง $x_i(k,l)$ ไม่ใช่โหนดใบไม้

เพิ่มค่า k,l

ขั้นที่ 3: ถ้าไม่มีโหนดใบไม้จากขั้นตอนที่ 2 ให้หยุด

ถ้ามีโหนดใบไม้จากขั้นตอนที่ 2 ให้ $N=\frac{N}{2}, i=i+1;$

ขั้นที่ 4 ไปยังขั้นที่ 2.

บทที่ 4
วิธีดำเนินการวิจัย

4.1 ขั้นตอนการเข้ารหัสแบบบล็อกหลายขนาดโดยการใช้ค่าลากรานจ์ (LVBTC)

การเข้ารหัสแบบบล็อกหลายขนาดดังที่ได้กล่าวไว้ในหัวข้อที่ 3.4 เนื่องจากว่าการดำเนินการวิจัยนี้ได้ทำการแบ่งภาพเริ่มแรกให้มีขนาดเป็น 16×16 เนื่องจากว่าการแบ่งบล็อกภาพที่มีขนาดมากกว่า 32×32 นั้นข้อมูลภาพภายในบล็อกจะไม่มีความเป็นเนื้อเดียวกัน [13] ดังนั้นเพื่อเป็นการกำหนดว่าจะให้ส่วนใดเป็นส่วนที่จะทำการแปลงข้อมูลภาพด้วยวิธีการแปลงโคซายน์เต็มหน่วยหรือไม่นั้นได้กำหนดค่าเทรชโฮลขึ้นตามรายละเอียดที่กล่าวไว้ใน [30] โดยนำค่าเทรชโฮลมาทำการเปรียบเทียบการค่าความแปรปรวนโดยค่าที่ใช้ในงานวิจัยเป็นดังนี้ คือ

รายละเอียดของภาพต่ำ $\sigma^2 < 10$

รายละเอียดของภาพปานกลาง $10 < \sigma^2 < 50$

รายละเอียดของภาพสูง $\sigma^2 > 50$

ในที่นี้ได้ใช้ค่าเทรชโฮลเพื่อนำมาทำการแยกประเภทบล็อกดังกล่าวด้วยกันสามค่าคือ 10, 20 และ 50 จากนั้นเมื่อได้ทำการแยกประเภทของรายละเอียดภาพเรียบร้อยแล้วในขั้นตอนของการแบ่งบล็อกออกเป็นหลายขนาดมีขั้นตอนการแบ่งบล็อกเป็นดังต่อไปนี้

ขั้นที่ 1. กำหนดให้ $i = 1:16:M$ และ $j = 1:16:N$

ขั้นที่ 2. แบ่งบล็อกภาพออกเป็น 16×16 ในที่นี้ให้เป็นบล็อกขนาด $m\times n$

ขั้นที่ 3. จากนั้นทำการหาค่าลากรานจ์ (Lagrange Cost) จากรูปแบบของลากรานจ์มัลติพลายเออร์ (Lagrange Multiplier) ด้วยวิธีการของตัวคูณลากรานจ์ในภาคผนวก ก. จากนั้นทำการจำลองการแบ่งภาพย่อยลงเป็นขนาด $m/2\times n/2$

ขั้นที่ 4. ทำการหาค่าลากรานจ์ของทุกบล็อกที่ทำการแบ่งมาจากบล็อกขนาด $m\times n$ จากนั้นทำการเปรียบเทียบค่าลากรานจ์ของบล็อกขนาด $m\times n$ กับผลรวมของค่าลากรานจ์ของบล็อกขนาด $m/2\times n/2$

4.1 ถ้าค่าลากรานจ์ของบล็อก $m\times n$ มากกว่าผลรวมของค่าลากรานจ์ของบล็อกขนาด $m/2\times n/2$ กำหนดให้ต้นไม้รหัส (code tree) ให้มีค่าเป็น "1"

4.2 ถ้าค่าลากรานจ์ของบล็อก $m \times n$ มากกว่าผลรวมของค่าลากรานจ์ของบล็อกขนาด $m/2 \times n/2$ ทำการเข้ารหัสและวิธีการกำหนดให้ต้นไม้รหัส (code tree) ให้มีค่าเป็น "0"

ขั้นที่ 5. ถ้าเป็นไปตามข้อ 4.1 ให้ $m = m/2$ และ $n = n/2$

ขั้นที่ 6. ทำซ้ำ ในขั้นตอนที่ 3 จนกระทั่งบล็อกย่อยที่มีขนาดเล็กที่สุดคือขนาด 2×2

ขั้นที่ 7. กำหนดให้ $i = i + 1$ และ $j = j + 1$ กลับไปยังขั้นตอนที่ 2

การหาค่าลากรานจ์ของแต่ละบล็อกสามารถเขียนเป็นสมการได้ดังนี้

$$J = D + \lambda R \tag{4.1}$$

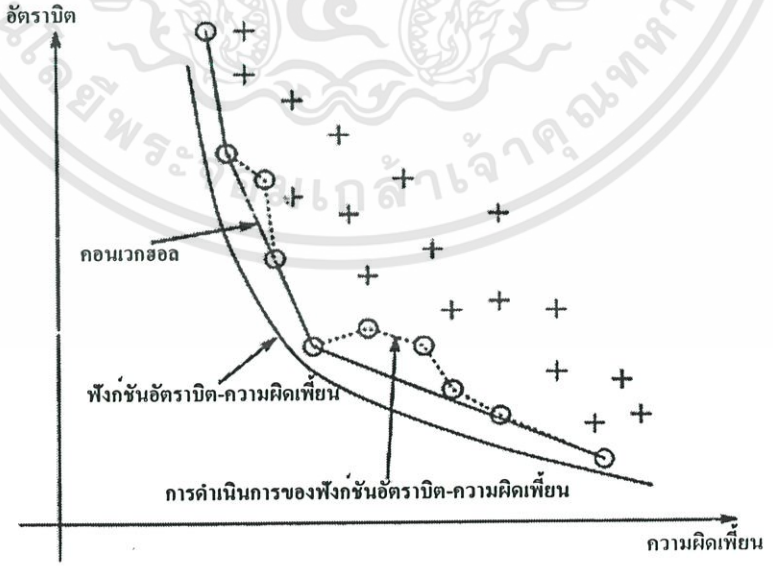
เมื่อ J = ค่าลากรานจ์

D = ค่าความผิดเพี้ยน

R = อัตราบิต

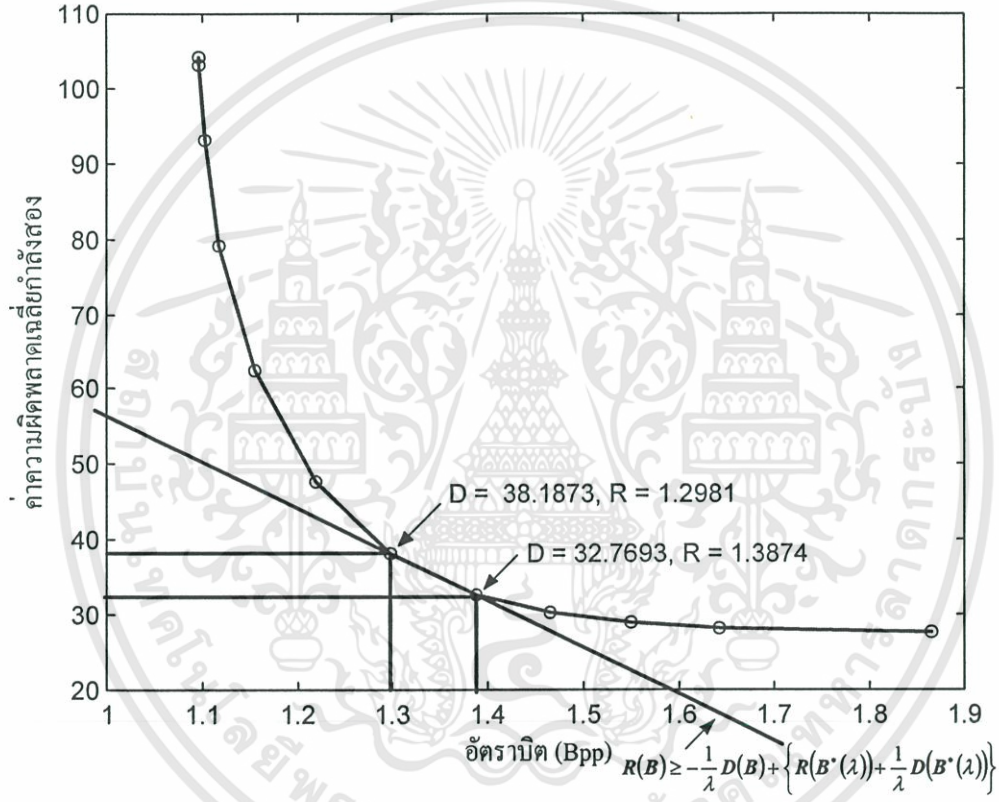
λ = ลากรานจ์มัลติพลายเออร์

ค่า λ ในที่นี้จะใช้ค่าเป็นเลขจำนวนเต็มโดยเริ่มจากค่า "1" และเพิ่มขึ้นเรื่อยๆ โดยในการทดลองจะใช้ค่า λ ด้วยกันทั้งหมด 12 ค่าซึ่งได้แก่ 1, 5, 10, 20, 40, 80, 160, 320, 640, 1280, 2560 และ 5120 และเมื่อเริ่มขั้นตอนการทำงานตามรูปที่ 4.2



รูปที่ 4.1 แสดงฟังก์ชันอัตราบิต-ความผิดเพี้ยนในทางทฤษฎี

สามารถวาดกราฟของค่าอัตราบิตและความผิดเพี้ยนที่ค่า λ ต่างๆกันได้ดังรูปที่ 4.2 การเข้ารหัสในส่วนของการเข้ารหัสแบบบล็อกโดยใช้ค่าสมบูรณ์โมเมนต์ทำการควอนไทซ์โดยการใช้ 2 ระดับได้กล่าวในหัวข้อ 3.2 และในส่วนที่เป็นการเข้ารหัสโดยใช้การแปลง DCT นั้นจะใช้ระดับการควอนไทซ์ทั้งหมดสามระดับคือ 8, 16 และ 32 โดยที่ในแต่ละครั้งที่มีการแบ่งบล็อกนั้นจะทำการเก็บค่าของรหัสต้นไม้ไว้เช่นเดียวกับการเข้ารหัสโดยใช้การแยกองค์ประกอบควอดทรีในหัวข้อ 3.4.1 ซึ่งการจัดสรรบิตให้กับแต่ละบล็อกจะได้กล่าวในหัวข้อถัดไป



รูปที่ 4.2 ฟังก์ชันของอัตราบิต-ความผิดเพี้ยนที่ได้ทางปฏิบัติโดยทำตามขั้นตอนการแบ่งบล็อกด้วยวิธีการหาค่าลากรานจ์

จากรูปที่ 4.2 จะเป็นค่าความผิดเพี้ยนที่เกิดขึ้นเมื่อกำหนดอัตราบิตค่าต่างๆให้ซึ่งจะเป็นไปตามทฤษฎีความผิดเพี้ยน [11, 19, 24, 27] ดังรูปที่ 4.1 และเป็นไปตาม วิธีการของวิธีการของตัวคูณลา กรานจ์ที่ได้กล่าวได้ในภาคผนวก ก. โดยสามารถคำนวณได้ดังนี้

$$R(B) \geq -\frac{1}{\lambda} D(B) + \left\{ R(B^*(\lambda)) + \frac{1}{\lambda} D(B^*(\lambda)) \right\} \tag{4.2}$$

จากรูปที่ 1 ในภาคผนวก ก. นำค่าในตำแหน่ง $R(B_1^*(\lambda))$และ $D(B_1^*(\lambda))$ มาแทนในสมการที่ (4.2) จะได้ดังนี้

$$R(B) \geq -\frac{1}{\lambda} D(B) + \left\{ R(B_1^*(\lambda)) + \frac{1}{\lambda} D(B_1^*(\lambda)) \right\} \tag{4.3}$$

และจากรูปที่ 4.3 เราสามารถหาค่า λ โดยหาจากสมการที่ (ก.7) ของภาคผนวก ก.ได้ดังนี้

$$\lambda = -\frac{dD}{dR} = -\left(\frac{38.1873 - 32.7693}{1.2981 - 1.3874} \right) = 60.6719 \tag{4.4}$$

นำค่าของ λ มาแทนลงใน (4.2) ซึ่งจะได้

$$R(B) \geq -\frac{1}{60.6719} D(B) + \left\{ 1.2981 + \frac{1}{60.6719} 38.1873 \right\} \tag{4.5}$$

$$R(B) \geq -0.0165 D(B) + 1.9275 \tag{4.6}$$

ในกรณีนี้ถ้าเราให้ $D(B) = 35$ จะได้ $R(B)$ ดังนี้

$$R(B) = -0.0165(35) + 1.9275$$
$$R(B) = 1.35$$

ค่า $R(B)$ ดังกล่าวสามารถบอกได้ว่าถ้าเรารู้ค่าของ $D(B)$ เราสามารถหาค่าได้ว่าความต้องการบิตที่จะกำหนดให้กับข้อมูลภาพมีจำนวนเท่าใดจึงจะเพียงพอนอกจากนั้นก็สามารถนำไปใช้ได้กับช่วงอื่นภายในกราฟได้ด้วยเช่นเดียวกัน

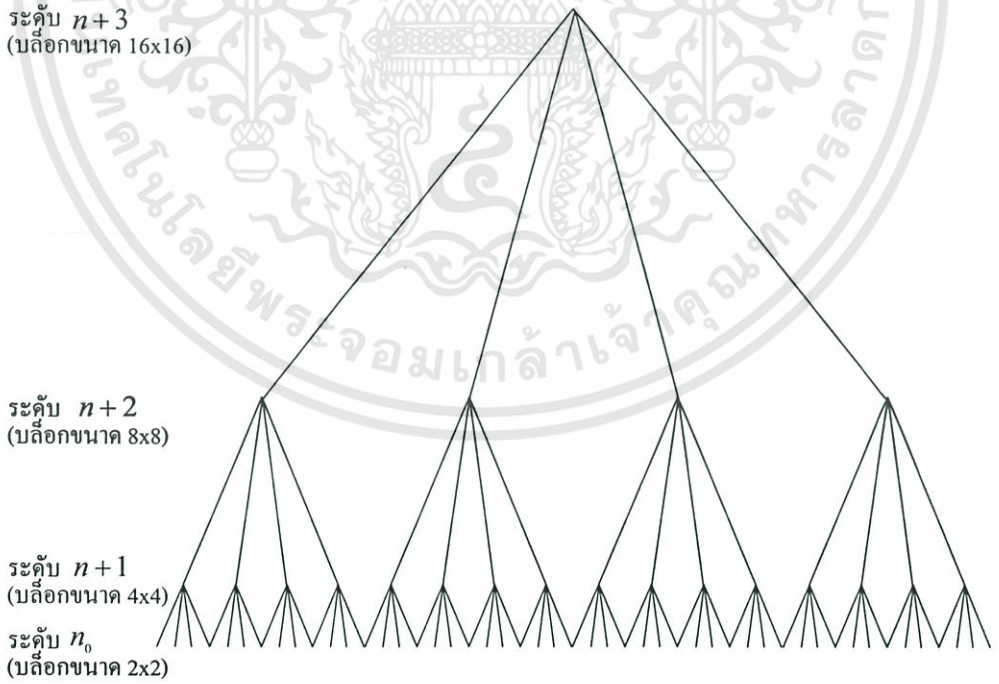
และในกรณีดังกล่าวนี้ จะเห็นได้ว่ามีความสอดคล้องของค่า λ ที่ได้เป็นไปตามสมการที่ (2) ที่ในภาคผนวก ก. ที่ว่า

$$\lambda_2 \geq -\frac{D(B^*(\lambda_1)) - D(B^*(\lambda_2))}{R(B^*(\lambda_1)) - R(B^*(\lambda_2))} \geq \lambda_1 \tag{4.7}$$

ซึ่งในที่นี้ λ มีค่าเท่ากับ 60.6719 และ $\lambda_1 = 60$ และ $\lambda_2 = 120$

4.1.1 การจัดสรรบิตรหัสของโครงสร้างต้นไม้

การจัดสรรบิตให้กับบล็อกเริ่มจากบล็อกที่มีขนาดใหญ่ที่สุดมีขนาด 16×16 และบล็อกที่เล็กที่สุดมีขนาด 2×2 ในที่นี้จำนวนบิตทั้งหมดที่ให้แสดงส่วนของโครงสร้างต้นไม้จะใช้ทั้งหมด 22 บิตและโดยที่จะกำหนดให้บิตแรกเป็นบิตที่บ่งบอกว่าบล็อกดังกล่าวผ่านขั้นตอนวิธีการใด โดยที่กำหนดให้บิตเป็น " 1 " เพื่อบอกว่าบล็อกดังกล่าวได้ใช้การเข้ารหัสแบบบล็อกที่มีขนาดคงที่โดยใช้ค่าสมบูรณ์โมเมนต์และถ้าเป็น " 0 " เพื่อบอกว่าบล็อกดังกล่าวได้ทำการเข้ารหัสโดยการใช้การแปลงโคซายน์เต็มหน่วย และบิตที่สองจะกำหนดให้กับการแบ่งจากบล็อกขนาด 16×16 เป็นขนาด 8×8 คือถ้าเป็น "0" หมายถึงบล็อก 16×16 ไม่ถูกแบ่งเป็นขนาด 8×8 แต่ถ้าเป็น "1" หมายถึงทำการแบ่งบล็อกจากขนาด 16×16 เป็นขนาด 8×8 จากนั้นเป็นบิตที่บ่งบอกว่าบล็อกขนาด 8×8 แบ่งเป็น 4×4 หรือไม่อีก 5 บิตโดยที่บ่งชี้ในลักษณะเดียวกันคือถ้าเป็น "0" หมายถึงไม่แบ่งบล็อกจาก 8×8 เป็น 4×4 และถ้าเป็น "1" คือแบ่งบล็อกจาก 8×8 เป็น 4×4 และจำนวนบิตอีก 16 บิตเพื่อบ่งบอกว่าภาพถูกแบ่งออกจากบล็อกขนาด 4×4 ออกเป็น 2×2 หรือไม่โดยลักษณะของการกำหนดบิตเช่นเดียวกับการแบ่งดังที่ได้กล่าวมา เมื่อสร้างโครงสร้างต้นไม้ในกรณีที่แบ่งบล็อกออกเป็นขนาดที่เล็กที่สุดคือ 2×2 ทั้งหมดจะได้โครงสร้างต้นไม้ดังรูปที่ 4.3

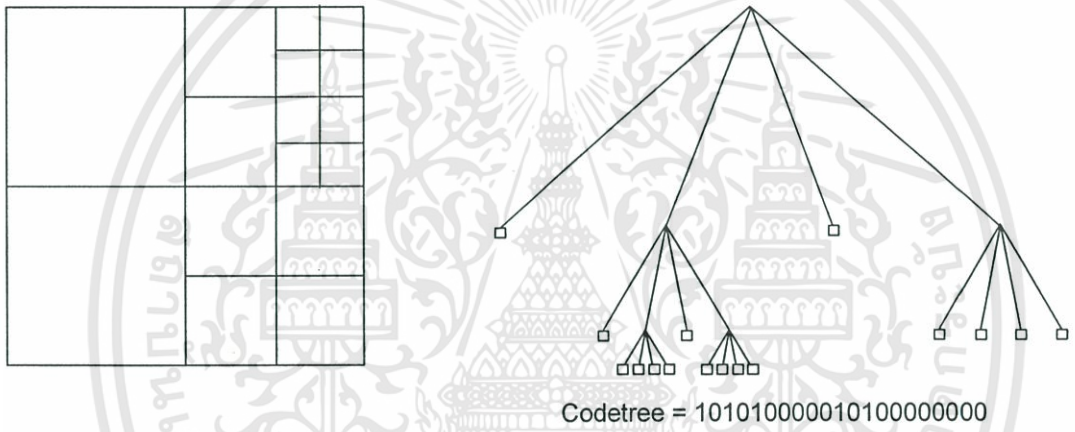


รูปที่ 4.3 แสดงรูปโครงสร้างต้นไม้ของการแบ่งภาพออกเป็นบล็อกหลายขนาด

ตารางที่ 4.1 การจัดสรรบิตให้กับโครงสร้างต้นไม้

ระดับ	ขนาดบล็อก	จำนวนบิต
n+3	16x16	1
n+2	8x8	5
n+1	4x4	21

จากรูปที่ 4.3 จะมีบิตที่แสดงโครงสร้างต้นไม้เป็นบิต "1" ทั้ง 21 บิต และในกรณีที่ภาพขนาด16×16 ถูกแบ่งออกเป็นดังรูปที่ 4.4 จะมีโครงสร้างต้นไม้และบิตที่แสดงโครงสร้างต้นไม้เป็นดังต่อไปนี้



รูปที่ 4.4 การแบ่งภาพของบล็อกขนาด 16×16 และรหัสต้นไม้ของบล็อก

4.1.2 การคำนวณอัตราบิตและความผิดเพี้ยน

การคำนวณอัตราบิตของแต่ละบล็อกของการเข้ารหัสแบบบล็อกโดยใช้ค่าสมบูรณ์โมเมนต์สามารถคำนวณได้โดย

$$R_B = B_{bitplane} + B_{\bar{X}} + B_{\bar{\alpha}} + T_{all} \tag{4.2}$$

R_B คือจำนวนบิตทั้งหมดของบล็อกของข้อมูลภาพที่กำลังพิจารณา

$B_{bitplane}$ คือ บิตที่ใช้แสดงแทนการควอนไทซ์สองระดับ(เลข 0,1)

$B_{\bar{X}}$ คือ บิตที่ใช้แทนเฉลี่ยของข้อมูลภายในบล็อก

$B_{\bar{\alpha}}$ คือ บิตที่ใช้แสดงค่าสมบูรณ์โมเมนต์ของข้อมูลภายในบล็อก

T_{all} คือ บิตที่ใช้แสดงโครงสร้างต้นไม้ทั้งหมดของบล็อกของข้อมูลภาพที่กำลังพิจารณา

การคำนวณหาจำนวนโหนดภายใน (inner node) และและโหนดใบไม้ (leave node) สามารถทำการได้ตามสมการดังต่อไปนี้

$$T_{all} = \text{จำนวนของ } \widetilde{T}_{leave} + \text{จำนวนของ } T_{inner} \tag{4.3}$$

$$\text{จำนวนของ } \widetilde{T}_{leave} = 3(T_{inner}) + 1 \tag{4.4}$$

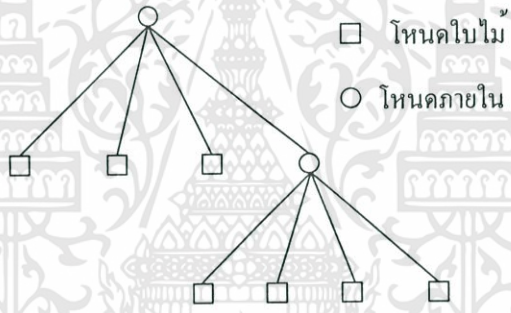
$$\text{จำนวนของ } T_{inner} = \frac{\widetilde{T} - 1}{3} \tag{4.5}$$

$$T_{all} = 4(T_{inner}) + 1 \tag{4.6}$$

หรือ

$$T_{all} = 4\left(\frac{\widetilde{T}_{leave} - 1}{3}\right) + 1 \tag{4.7}$$

แทน (4.4) และ (4.5) ลงใน (4.3) จะได้ดังสมการที่ (4.6) และ (4.7)



รูปที่ 4.5 ตำแหน่งของโหนดใบไม้และโหนดภายใน

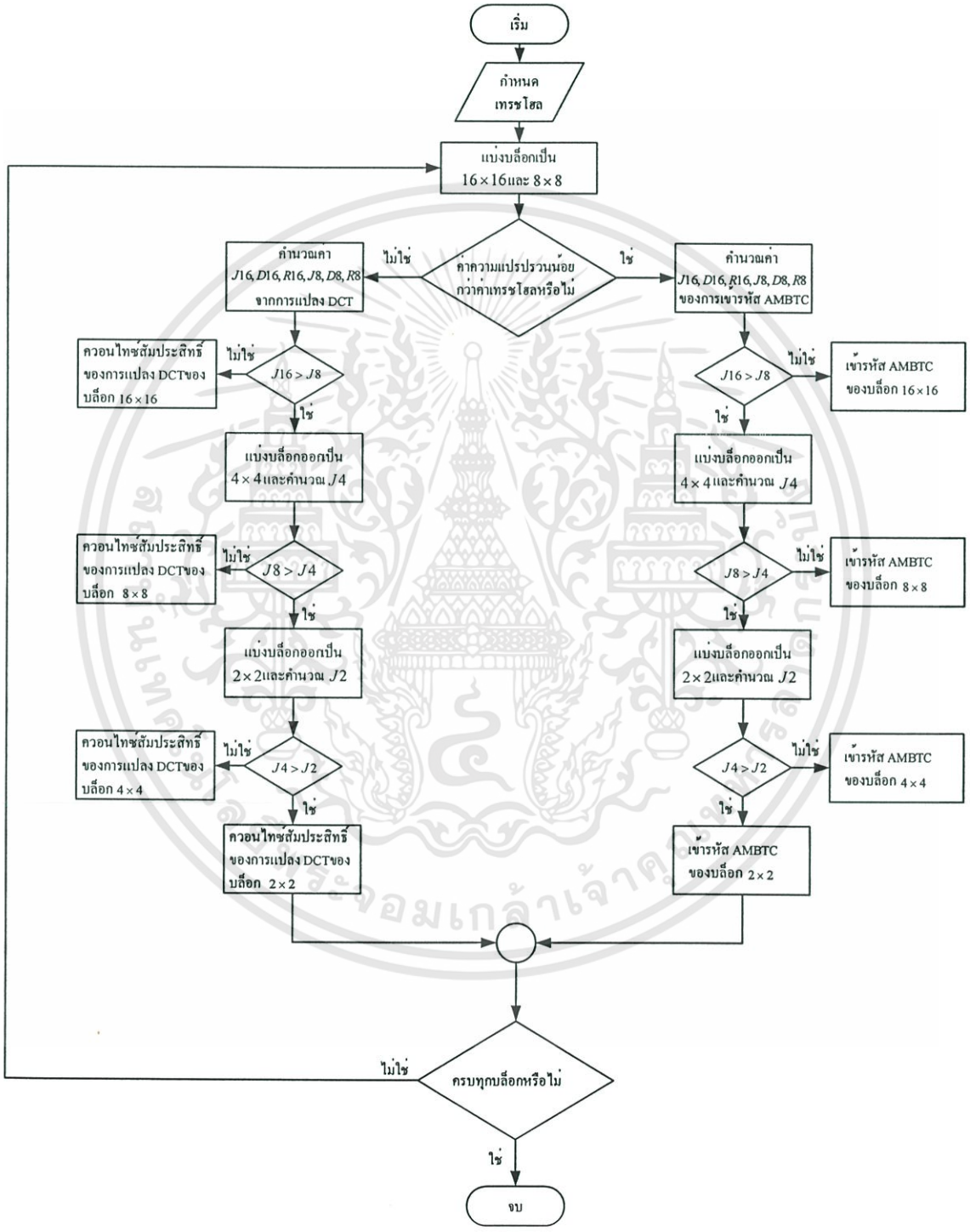
ความผิดเพี้ยนของการเข้ารหัสจะได้มาจากการคิดคำนวณความผิดเพี้ยนที่เกิดขึ้นจากการเข้า รหัสแบบบล็อกที่มีขนาดคงที่โดยใช้ค่าสมบูรณ์โมเมนต์ซึ่งสามารถคำนวณความผิดเพี้ยนหรือความผิดพลาดกำลังสองที่เกิดขึ้นหลังจากการเข้ารหัสแล้วได้ดังนี้

$$D_B = \sum_{i=1}^{m-q-1} (x_i - a)^2 + \sum_{i=m-q}^{m} (x_i - b)^2 \tag{4.8}$$

เมื่อ D_B คือ ความผิดเพี้ยนของการเข้ารหัสแบบ AMBTC

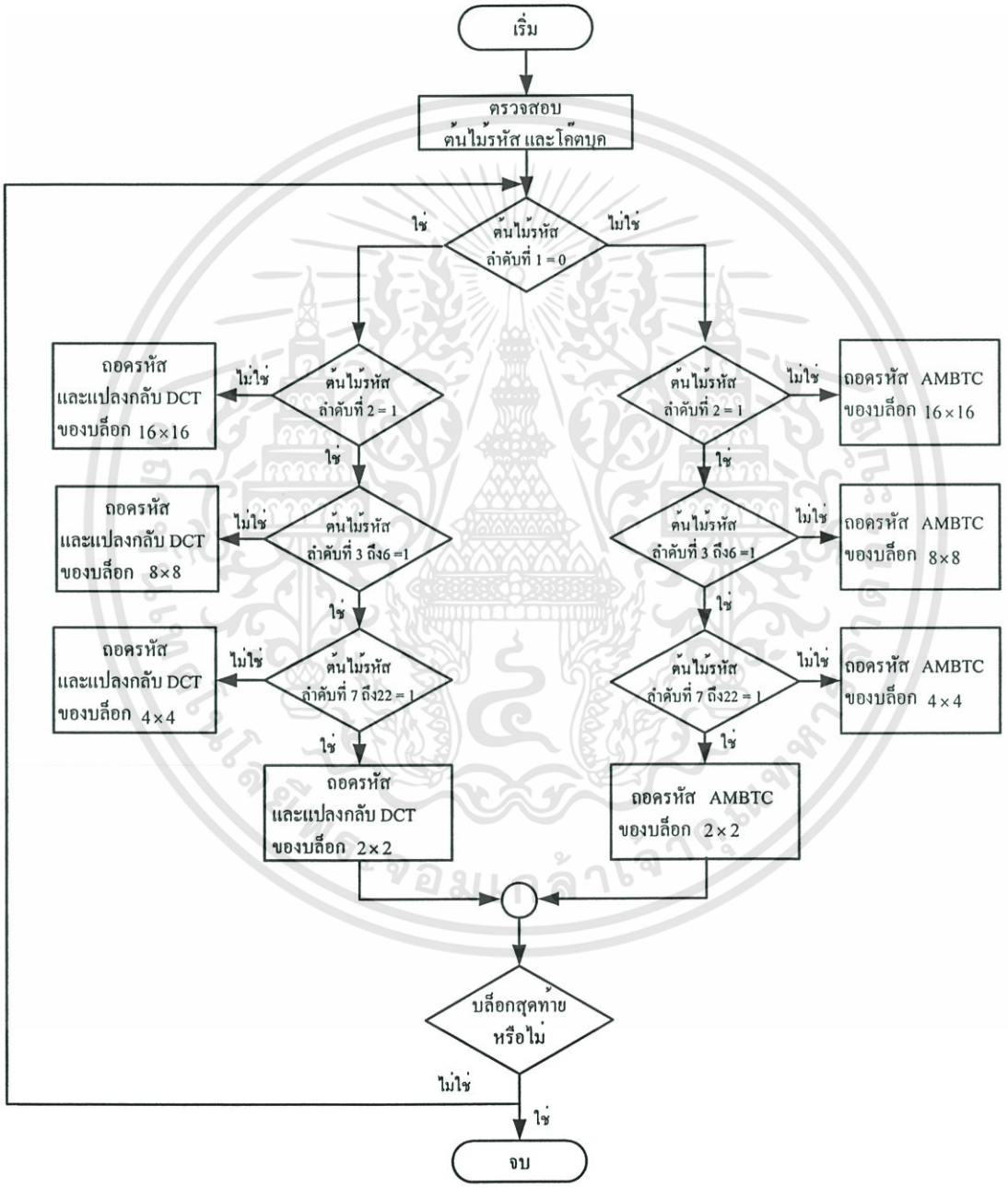
a, b คือบิตที่ได้จากการทำการสร้างกลับภาพของเข้ารหัสแบบบล็อกที่มีขนาดคงที่โดยใช้ค่าสมบูรณ์โมเมนต์

จำนวนบิตของขบวนการที่ใช้การแปลงโคซายน์เต็มหน่วยนั้นสามารถหาได้โดยหาผลรวมของบิตที่ถูกเข้ารหัสแบบความยาวหลายขนาด (Variable length code)และความผิดเพี้ยนของขบวนการที่ใช้การแปลงโคซายน์เต็มหน่วยหาได้โดยใช้สมการที่ 2.74



รูปที่ 4.6 การเข้ารหัสแบบบล็อกหลายขนาดโดยการใช้ค่าลากรานจ์ (LVBTC)

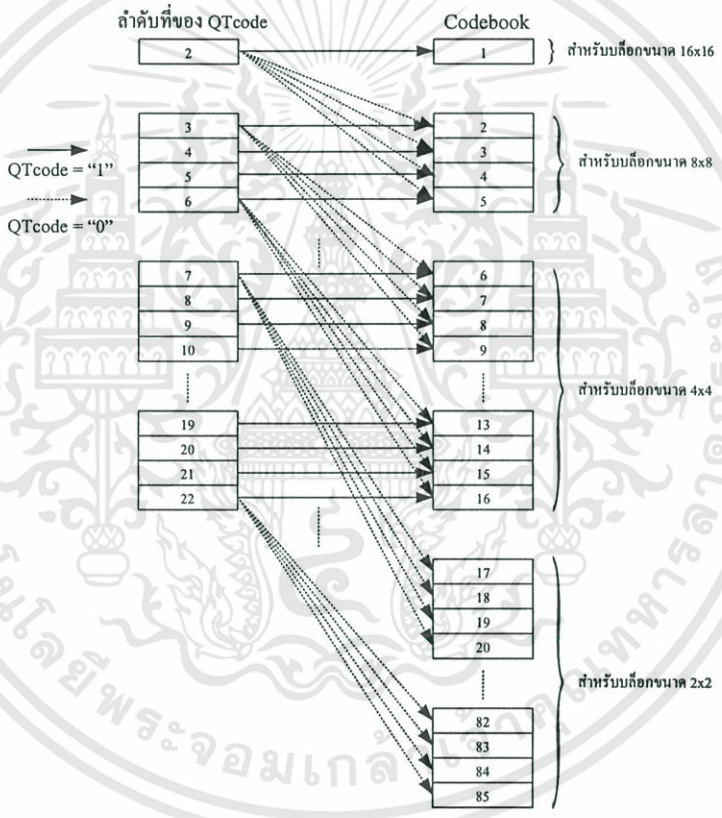
วิธีการเข้ารหัสเป็นดังรูปที่ 4.6 ในแต่ละครั้งเมื่อมีการแบ่งบล็อกเกิดขึ้นจะทำการปรับปรุงและเก็บค่าต้นไม้รหัสไว้และเมื่อใดที่บล็อกนั้นไม่ได้ถูกกระทำอีกต่อไปซึ่งหมายความว่าบล็อกว่าบล็อกดังกล่าวมีข้อมูลที่มีความใกล้เคียงกันและไม่จำเป็นที่จะต้องแบ่งบล็อกอีกต่อไปหรือเป็นไปตามข้อ 4.2 ของขั้นตอนการแบ่งบล็อกนั่นเอง แต่ถ้าทำการแบ่งบล็อกต่อไปนั้นหมายถึงว่าบล็อกดังกล่าวมีข้อมูลที่ไม่มีความใกล้เคียงกันและจำเป็นที่จะต้องแบ่งบล็อกต่อไป



รูปที่ 4.7 การถอดรหัสแบบบล็อกหลายขนาดโดยการใช้ค่าลากรานจ์ (LVBTC)

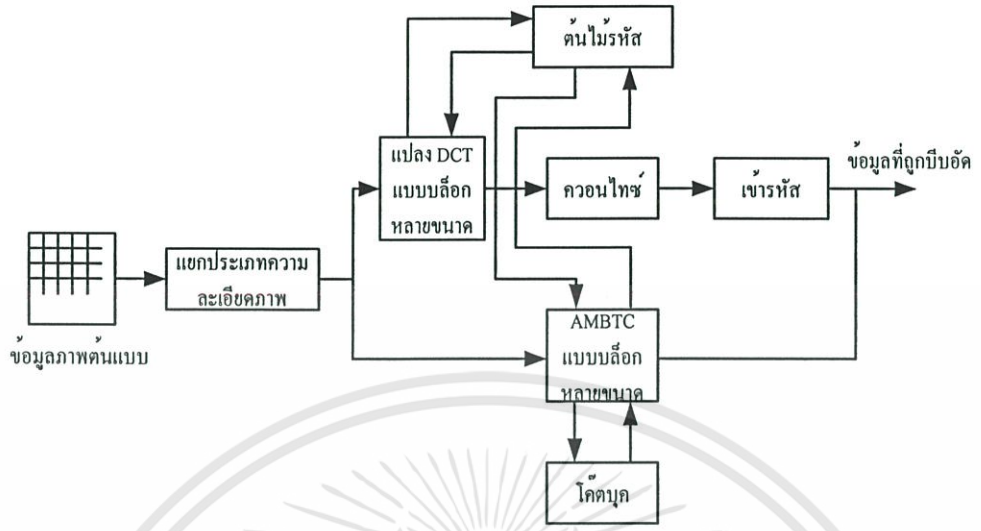
4.2 ขั้นตอนการถอดรหัสแบบบล็อกหลายขนาดโดยการใช้ค่าลากรานจ์

ขั้นตอนการถอดรหัสเริ่มด้วยการนำข้อมูลบิตไบนารีของแต่ละบล็อกรวมถึงรหัสโครงสร้างต้นไม้ ของบล็อกนั้นๆที่ได้จากการเข้ารหัสมาถอดรหัสจากนั้นนำรหัสต้นไม้มาสร้างทำการแบ่งบล็อกให้เหมือนกับการแบ่งบล็อกของทางด้านเข้ารหัส จากนั้นทำการถอดรหัสจากที่เข้ารหัสด้วยการเข้ารหัสแบบบล็อกโดยใช้ค่าสมบูรณ์โมเมนต์และเช่นเดียวกันก็ทำการถอดรหัสทำกลับการควอนไทซ์และการแปลงกลับ DCT และสุดท้ายคือการรวมภาพกลับให้เท่ากับภาพต้นแบบ ขั้นตอนการเข้าและถอดรหัสได้แสดงดังรูปที่ 4.9 และ 4.10 ตามลำดับ

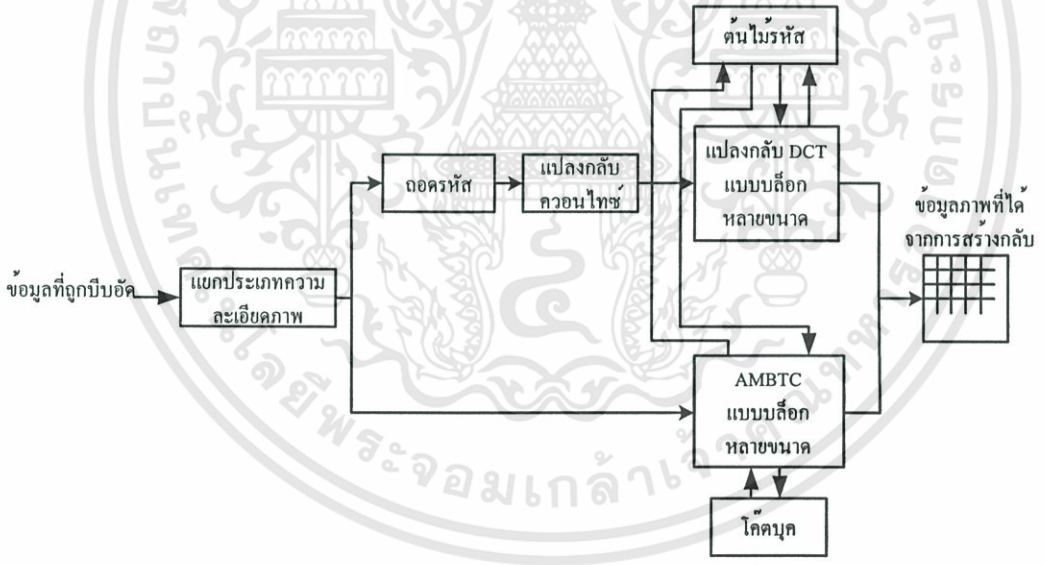


รูปที่ 4.8 การอ้างอิงข้อมูลของจากตำแหน่งของต้นไม้รหัส

ในรูปที่ 4.8 เป็นรูปของการอ้างตำแหน่งข้อมูลที่อยู่ในต้นไม้รหัสกับตำแหน่งของโค๊ตบุคที่เก็บค่าเฉลี่ยและค่าสมบูรณ์โมเมนต์ของการเข้ารหัสโดยใช้ค่าสมบูรณ์โมเมนต์ (AMBTC) การอ้างอิงดังกล่าวจะใช้ทางด้านการถอดรหัสเพื่อบอกตำแหน่งของบล็อกต่างๆภายในภาพ



รูปที่ 4.9 แสดงรูปบล็อกไดอะแกรมของการเข้ารหัสแบบ LVBTC



รูปที่ 4.10 แสดงรูปบล็อกไดอะแกรมของการถอดรหัสแบบ LVBTC

บทที่ 5
ผลการวิจัย

ในการทดลองของวิธีการที่ได้นำเสนอ (LVBTC) ซึ่งในที่นี้จะใช้ภาพที่มีขนาด 512x512 ทั้งหมด 4 ภาพและภาพขนาด 256x256 ทั้งหมด 2 ภาพ รวมด้วยกันทั้งหมด 6 ภาพ โดยที่ภาพที่ใช้นั้นเป็นภาพระดับสีเทา (Gray level) ซึ่งเป็นภาพมาตรฐาน [31] ซึ่งแสดงดังรูปที่ 5.1 และในที่นี้ใช้โปรแกรมคำนวณทางคณิตศาสตร์ MATLAB เวอร์ชั่น 5.3.1 ทำงานบนเครื่องไมโครคอมพิวเตอร์ที่มีหน่วยประมวลผล "Celeron Processor" ความเร็ว 500 MHz และมีหน่วยความจำขนาด 192 Mbytes

(ก) ภาพLena ขนาด 512x512 พิกเซล

(ข) ภาพ Pepper ขนาด 512x512 พิกเซล

(ค) ภาพ Jet ขนาด 512x512 พิกเซล

(ง) ภาพ Boat ขนาด 512x512 พิกเซล

(จ) ภาพ Lena ขนาด 256x256 พิกเซล

(ฉ) ภาพ baboon ขนาด 256x256 พิกเซล

รูปที่ 5.1 แสดงรูปที่ใช้ในงานวิจัยทั้งหมด

5.1 ผลของการเปรียบเทียบค่าการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสอง (MSE) และค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด (PSNR)

ในผลของการวิจัยส่วนแรกจะเป็นผลของการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสอง (MSE) และค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด (PSNR) ของภาพต่างๆที่มีการเปลี่ยนค่าลากรานจ์มัลติพลายเออร์ (λ) ตารางที่ 5.1 ถึง 5.6 เป็นการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองและค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพต่างๆที่ใช้ในการทำวิจัยตามลำดับ เมื่อเพิ่มค่าลากรานจ์มัลติพลายเออร์ (λ) มากขึ้นจะพบว่าค่าความผิดพลาดจะมากขึ้นตามไปด้วยจากสมการที่ 4.1 เป็นสมการเส้นตรงที่มีการเปลี่ยนแปลงค่าลากรานจ์มัลติพลายเออร์ดังที่กล่าวไว้ในในบทที่ 4 ซึ่งค่าดังกล่าวเปรียบเสมือนค่าความชันของสมการเส้นตรง โดยที่การเปลี่ยนแปลงแต่ละค่าจะทำให้ความชันเปลี่ยนแปลงไปโดยที่จะมีผลทำให้ค่าของความผิดเพี้ยนและอัตราบิตเปลี่ยนแปลงไปด้วย ผลของการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสอง (MSE) และค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด (PSNR) ของภาพต่างๆ มีดังต่อไปนี้

ตารางที่ 5.1 แสดงค่าความผิดพลาดเฉลี่ยกำลังสองที่ค่าเทรชโฮลเท่ากับ 10 และค่าระดับการควอนไทซ์เท่ากับ 8

ลากรานจ์มัลติพลายเออร์ (λ)	ความผิดพลาดเฉลี่ยกำลังสอง					
	512x512				256x256	
	Lena	Peppers	Jet	Boat	Lena	Baboon
1	4.6438	5.3867	3.6558	5.7535	2.4368	5.4985
5	5.1337	5.8810	4.0887	6.1446	2.8293	5.7564
10	5.6261	6.3784	4.4018	6.5406	3.1682	5.8575
20	6.3245	7.0846	4.8519	7.2288	3.4838	6.0372
40	7.0201	7.8758	5.3316	7.9227	3.9568	6.1411
80	7.8728	8.9329	5.8815	8.6661	4.5382	6.5312
160	8.3705	9.7164	6.2507	9.0868	4.8460	6.5349
320	8.5078	9.7190	6.3969	9.0888	4.8460	6.5349
640	8.5092	9.7213	6.3969	9.0893	4.8460	6.5349
1280	8.5099	9.7213	6.3969	9.0895	4.8460	6.5349
2560	8.5101	9.7214	6.3969	9.0898	4.8460	6.5349
5120	8.5068	9.7214	6.3969	9.0901	4.8460	6.5349

ตารางที่ 5.2 แสดงค่าความผิดพลาดเฉลี่ยกำลังสองที่ค่าเทรชโฮลเท่ากับ 20 และค่าระดับการควอนไทซ์เท่ากับ 8

ลากรานจ์มัลติพลายเออร์ (λ)	ความผิดพลาดเฉลี่ยกำลังสอง					
	512x512				256x256	
	Lena	Peppers	Jet	Boat	Lena	Baboon
1	7.6086	7.6260	6.0991	12.331	6.3688	13.348
5	8.2159	8.1764	6.5691	12.780	6.7808	13.659
10	8.8201	8.8490	7.0214	13.361	7.3200	13.837
20	9.8745	9.9962	7.7530	14.745	8.2067	14.715
40	11.381	11.632	8.8515	17.359	10.059	16.413
80	14.029	14.012	10.483	20.873	12.427	19.007
160	17.028	16.814	12.736	25.365	14.475	21.305
320	20.511	20.514	15.975	29.969	20.764	23.415
640	23.295	23.522	17.698	32.393	21.441	23.941
1280	23.741	23.806	19.281	33.534	22.791	23.941
2560	23.741	23.806	19.522	33.534	22.791	23.941
5120	23.741	23.806	19.522	33.534	22.791	23.941

ตารางที่ 5.3 แสดงค่าความผิดพลาดเฉลี่ยกำลังสองที่ค่าเทรชโฮลเท่ากับ 50 และค่าระดับการควอนไทซ์เท่ากับ 8

ลากรานจ์มัลติพลายเออร์ (λ)	ความผิดพลาดเฉลี่ยกำลังสอง					
	512x512				256x256	
	Lena	Peppers	Jet	Boat	Lena	Baboon
1	27.5705	24.207	32.588	42.7361	50.6645	177.36
5	28.2002	24.786	33.063	43.1631	51.0578	177.93
10	29.0171	25.559	33.576	43.8780	51.5757	178.53
20	30.2887	27.033	34.509	45.6738	53.0215	180.11
40	32.7693	29.625	36.768	49.3101	55.5547	186.12
80	38.1873	34.164	40.917	56.4317	62.1403	199.28
160	47.6597	41.962	50.356	68.1905	79.8209	223.65
320	62.5242	56.056	66.095	85.3862	100.9897	249.64
640	79.2036	72.130	83.635	104.5359	119.6328	267.74
1280	93.0422	90.947	99.470	122.0321	145.6080	282.36
2560	103.2364	110.55	107.30	137.1021	167.2464	283.36
5120	104.2951	122.53	109.18	140.4932	171.3022	283.36

ตารางที่ 5.4 แสดงค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดที่ค่าเทรชโฮลเท่ากับ 10 และค่าระดับการควอนไทซ์เท่ากับ 8

ลากรานจ์มัลติพลายเออร์ (λ)	ค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด (dB)					
	512x512				256x256	
	Lena	Peppers	Jet	Boat	Lena	Baboon
1	41.462	40.818	42.501	40.531	41.681	40.728
5	41.026	40.436	42.015	40.246	41.386	40.529
10	40.629	40.084	41.6945	39.975	41.126	40.454
20	40.121	39.628	41.2717	39.54	40.864	40.322
40	39.667	39.168	40.8622	39.142	40.523	40.248
80	39.170	38.621	40.4359	38.753	40.083	39.981
160	38.903	38.256	40.1715	38.547	39.83	39.978
320	38.833	38.255	40.0711	38.546	39.83	39.978
640	38.833	38.254	40.0711	38.546	39.83	39.978
1280	38.832	38.254	40.0711	38.545	39.83	39.978
2560	38.832	38.254	40.0711	38.545	39.83	39.978
5120	38.831	38.254	40.0711	38.545	39.83	39.978

ตารางที่ 5.5 แสดงค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดที่ค่าเทรชโฮลเท่ากับ 20 และค่าระดับการควอนไทซ์เท่ากับ 8

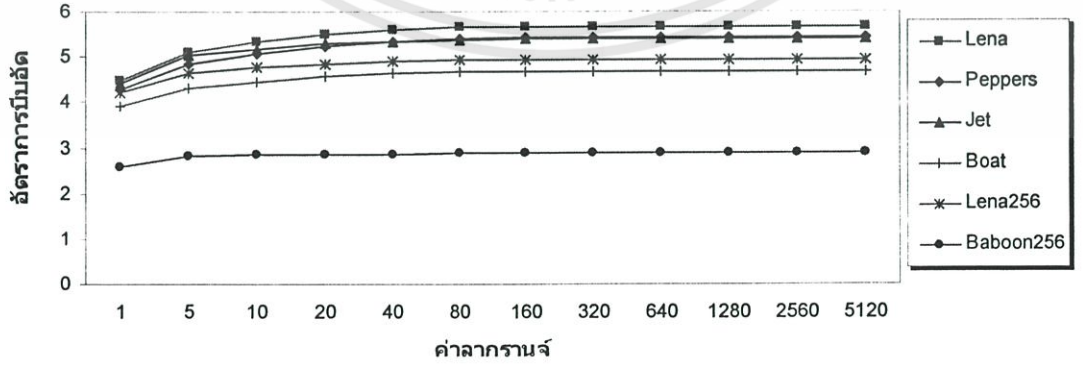
ลากรานจ์มัลติพลายเออร์ (λ)	ค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด (dB)					
	512x512				256x256	
	Lena	Peppers	Jet	Boat	Lena	Baboon
1	39.318	39.308	40.278	37.221	39.158	36.877
5	38.984	39.005	39.956	37.066	38.971	36.777
10	38.676	38.662	39.667	36.873	38.763	36.720
20	38.186	38.132	39.236	36.444	38.441	36.453
40	37.569	37.474	38.661	35.736	37.822	35.979
80	36.660	36.666	37.926	34.935	37.045	35.342
160	35.819	35.874	37.081	34.088	36.137	34.846
320	35.011	35.010	36.096	33.364	34.985	34.436
640	34.458	34.416	35.651	33.026	34.529	34.339
1280	34.376	34.364	35.280	32.876	34.285	34.339
2560	34.376	34.364	35.226	32.876	34.285	34.339
5120	34.376	34.364	35.226	32.876	34.285	34.339

ตารางที่ 5.6 แสดงค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดที่ค่าเทรชโฮลเท่ากับ 50 และค่าระดับการควอนไทซ์เท่ากับ 8

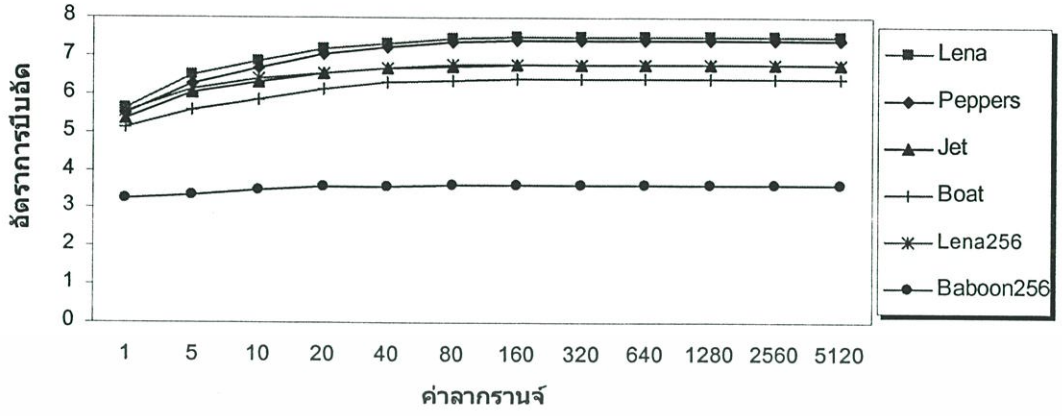
ลากรานจ์มัลติพลายเออร์ (λ)	ค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด (dB)					
	512x512				256x256	
	Lena	Peppers	Jet	Boat	Lena	Baboon
1	33.7264	34.291	33.000	31.8229	31.0838	25.642
5	33.6283	34.189	32.937	31.7797	31.0502	25.628
10	33.5043	34.055	32.871	31.7083	31.0064	25.614
20	33.3180	33.812	32.752	31.5341	30.8863	25.575
40	32.9761	33.414	32.476	31.2014	30.6836	25.433
80	32.3116	32.795	32.012	30.6156	30.1971	25.136
160	31.3493	31.902	31.110	29.7936	29.1096	24.635
320	30.1703	30.645	29.929	28.8169	28.0880	24.158
640	29.1434	29.550	28.907	27.9382	27.3523	23.854
1280	28.4440	28.543	28.154	27.2661	26.4990	23.623
2560	27.9925	27.695	27.825	26.7604	25.8972	23.607
5120	27.9482	27.248	27.750	26.6543	25.7932	23.607

5.2 ผลของการเปรียบเทียบอัตราการบีบอัด

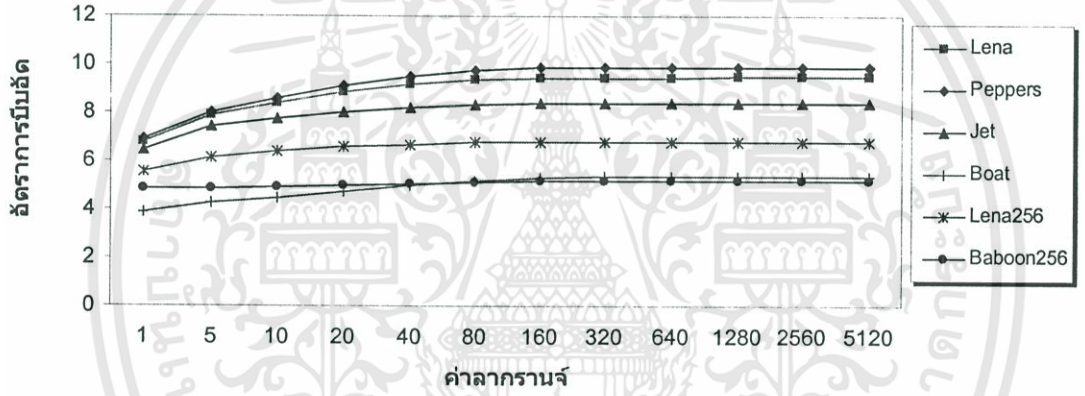
การทดลองได้ใช้ค่าเทรชโฮล ซึ่งเป็นค่าที่นำมาแยกประเภทของบล็อกที่มีรายละเอียดสูงและบล็อกที่มีรายละเอียดต่ำดังที่กล่าวไว้ในบทที่ 4 โดยได้เปรียบเทียบค่าเทรชโฮลค่าต่างพบว่าที่ค่าเทรชโฮลค่าต่ำจะมีค่า ความผิดพลาดเฉลี่ยกำลังสองน้อยกว่ากรณีที่ใช้ค่าเทรชโฮลค่ามากและเช่นเดียวกันค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดจะมีค่ามากเมื่อใช้ค่าเทรชโฮลค่าต่ำและจะมีค่าน้อยเมื่อใช้ค่าเทรชโฮลสูง ส่วนต่อไปจะเป็นการเปรียบเทียบอัตราการบีบอัดข้อมูลภาพโดยมีผลการวิจัยดังต่อไปนี้



รูปที่ 5.2 แสดงอัตราการบีบอัดที่ค่าลากรานจ์ต่างๆ โดยที่เทรชโฮลเท่ากับ 10 และระดับการควอนไทซ์เท่ากับ 8



รูปที่ 5.3 แสดงอัตราการบีบอัดที่ค่าลากรานจ์ต่างๆโดยที่เทรชโฮลเท่ากับ 10 และระดับการควอนไทซ์เท่ากับ 16

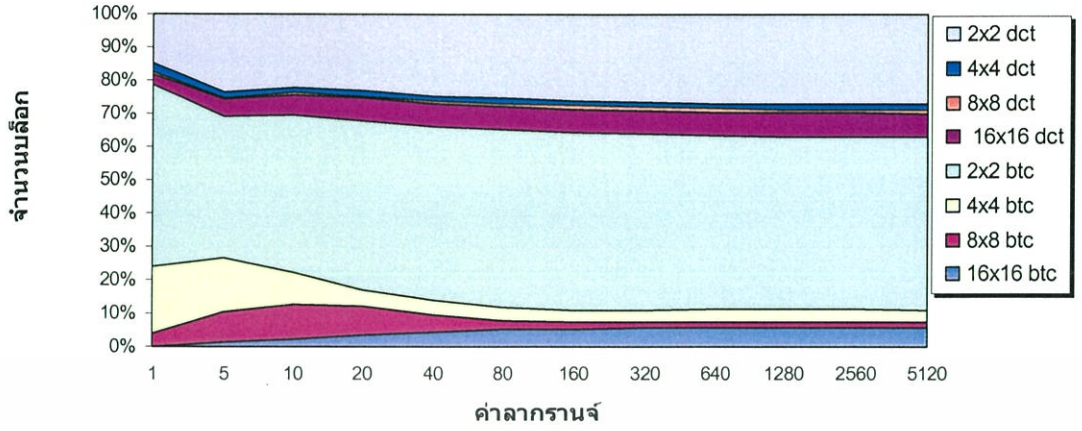


รูปที่ 5.4 แสดงอัตราการบีบอัดที่ค่าลากรานจ์ต่างๆโดยที่เทรชโฮลเท่ากับ 10 และระดับการควอนไทซ์เท่ากับ 32

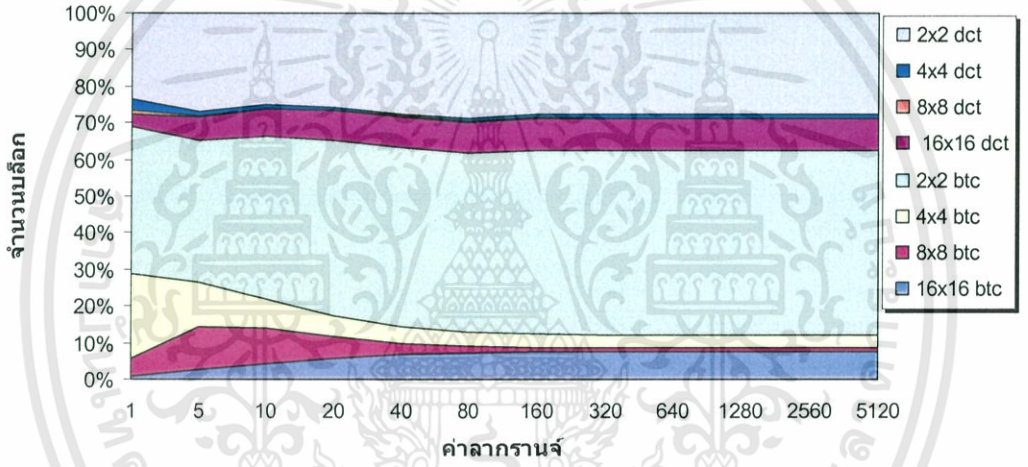
จากรูปที่ 5.2 ถึง 5.4 เป็นการเปรียบเทียบอัตราการบีบอัดโดยที่มีการเปลี่ยนระดับการควอนไทซ์ที่มีค่า 8, 16 และ 32 ตามลำดับ และเปลี่ยนค่าลากรานจ์มัลติพลายเออร์ (λ) ในการวิจัยพบว่าเมื่อใช้ระดับการควอนไทซ์สูงขึ้นจะทำให้ค่าอัตราการบีบอัดสูงขึ้นด้วย และเช่นเดียวกันที่ค่าลา กรานจ์มัลติพลายเออร์สูงขึ้นก็ทำให้อัตราการบีบอัดสูงขึ้นเช่นกัน

5.3 ผลของการเปรียบเทียบจำนวนบล็อกขนาดต่างๆ

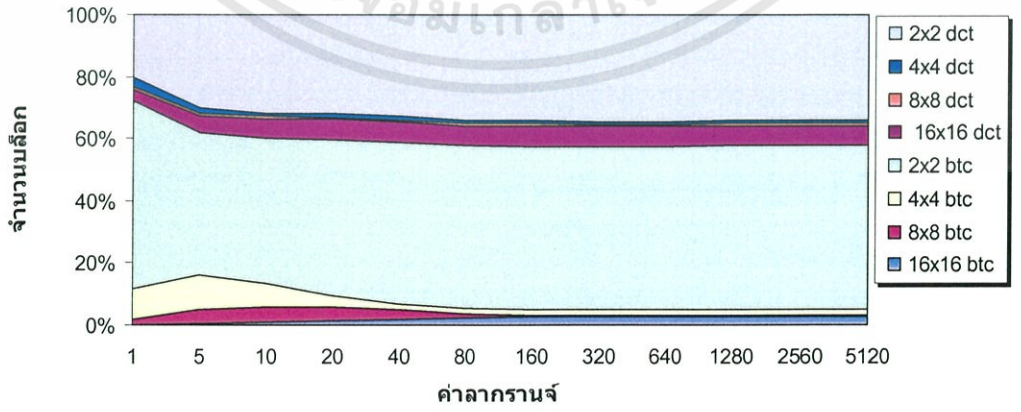
ในส่วนถัดไปจะเป็นการเปรียบเทียบจำนวนบล็อกที่มีการแบ่งออกเป็นบล็อกย่อยโดยที่มีการเปลี่ยนแปลงค่าลากรานจ์มัลติพลายเออร์ แต่ให้ระดับควอนไทซ์และค่าเทรชโฮลคงที่ โดยมีผลการวิจัยเป็นดังต่อไปนี้



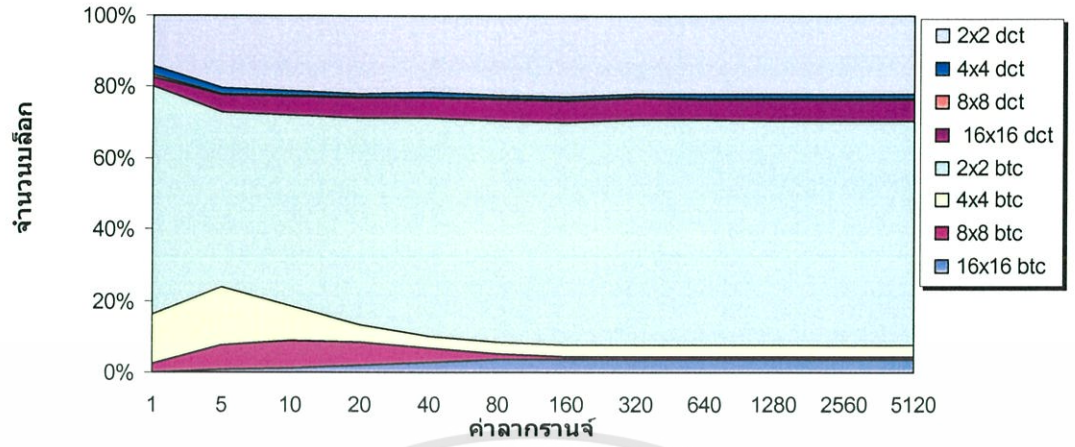
รูปที่ 5.5 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Lena ขนาด512x512 โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8



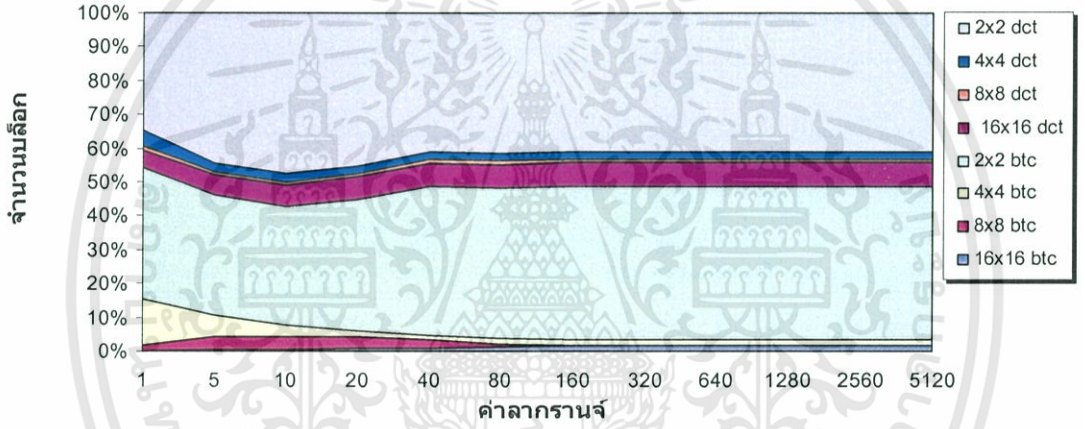
รูปที่ 5.6 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Jet ขนาด512x512 โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8



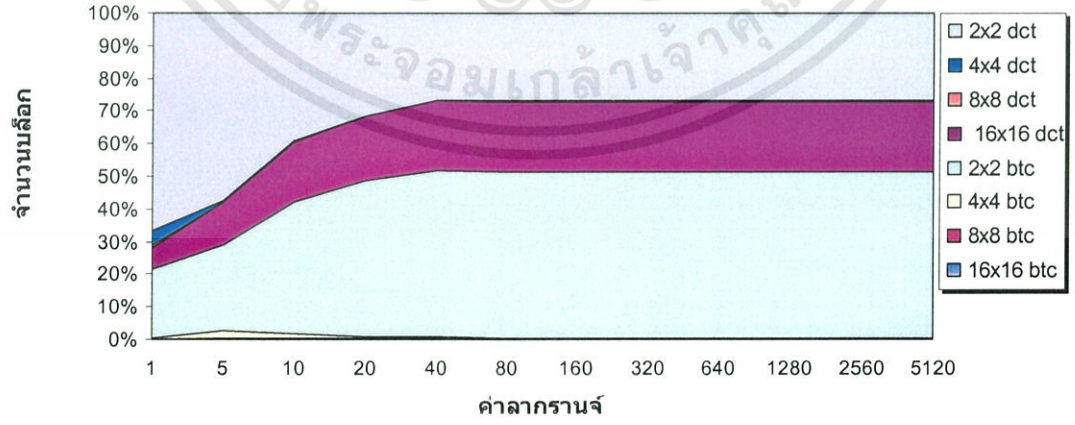
รูปที่5.7 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Boat ขนาด512x512 โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8



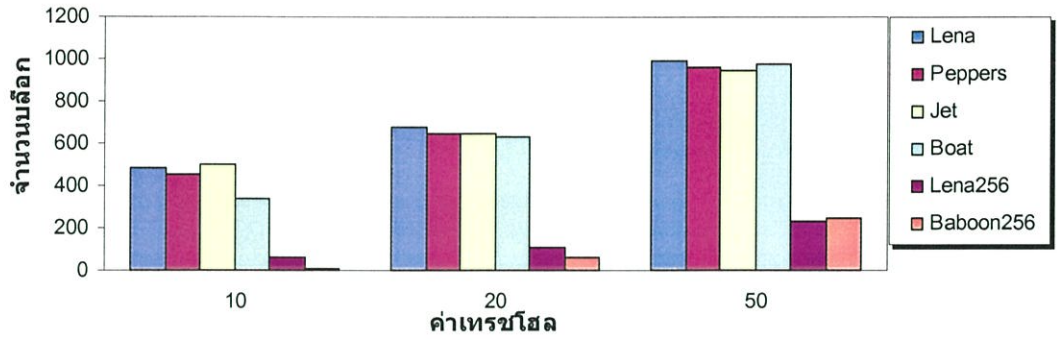
รูปที่ 5.8 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Peppers ขนาด512x512 โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8



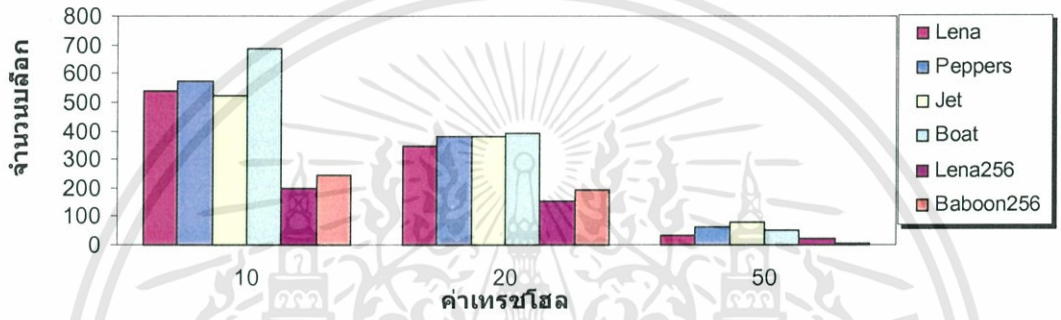
รูปที่ 5.9 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Lena ขนาด256x256 โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8



รูปที่ 5.10 แสดงการเปรียบเทียบจำนวนบล็อกของภาพ Baboon ขนาด256x256 โดยเทรชโฮลเท่ากับ 10 และระดับควอนไทซ์เท่ากับ 8



รูปที่ 5.11 แสดงการเปรียบเทียบจำนวนบล็อกของภาพที่เข้าสู่ขบวนการเข้ารหัสแบบ AMBTC

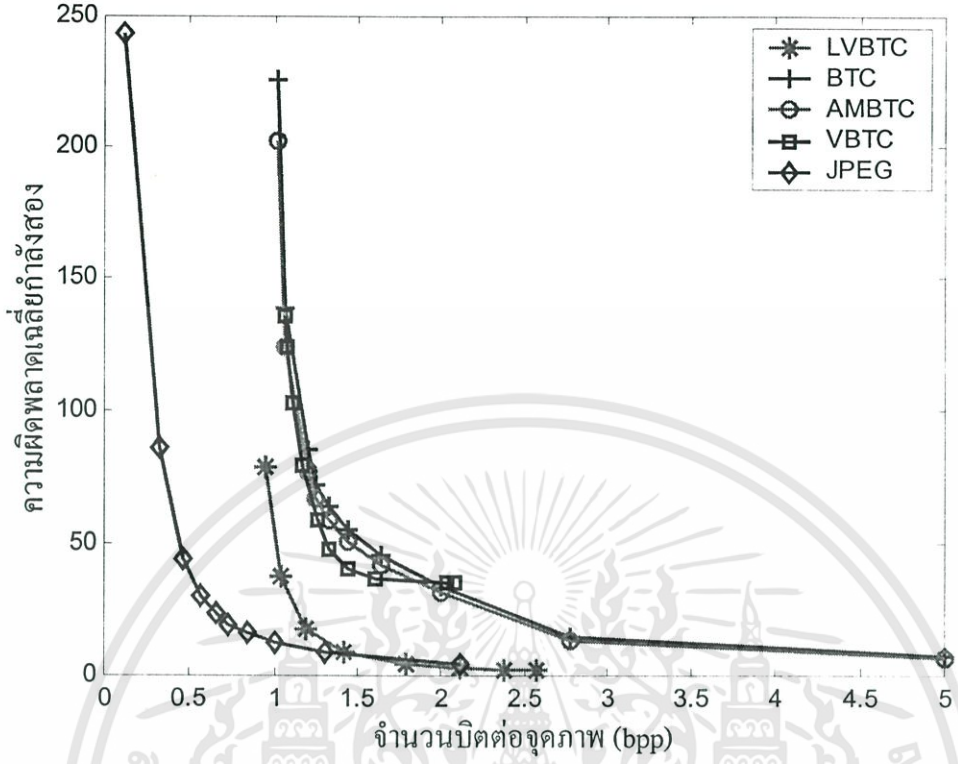


รูปที่ 5.12 แสดงการเปรียบเทียบจำนวนบล็อกของภาพที่เข้าสู่ขบวนการเข้ารหัสแบบ DCT

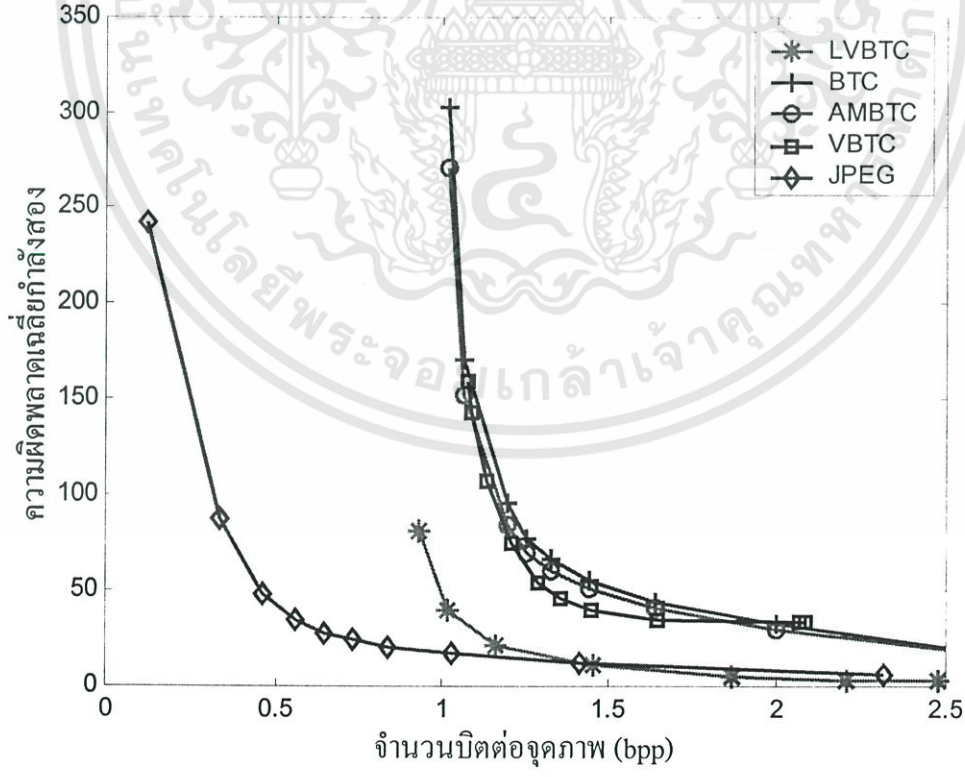
ซึ่งในแต่ละภาพจะมีจำนวนบล็อกเมื่อแบ่งแล้วแตกต่างกันไป ดังที่ในงานวิจัยนี้กำหนดให้บล็อกที่มีขนาดใหญ่ที่สุดในการแบ่งมีขนาด 16×16 และบล็อกที่เล็กที่สุดในการแบ่งกำหนดให้มีขนาดเป็น 2×2 ทั้งนี้การแบ่งจะขึ้นอยู่กับว่าภาพดังกล่าวมีรายละเอียดมากเพียงใด และยังขึ้นอยู่กับค่าลากรานจ์มัลติพลายเออร์ อีกด้วย โดยถ้าภาพมีรายละเอียดสูงก็จะมีการแบ่งจำนวนบล็อกออกเป็นบล็อกขนาดเล็กจำนวนมากดังเช่นรูปที่ 5.10 ในช่วงแรกลากรานจ์มัลติพลายเออร์มีค่าน้อยและภาพดังกล่าวมีรายละเอียดมากดังนั้นจึงมีจำนวนบล็อกขนาด 2×2 มากและเมื่อค่าลากรานจ์มัลติพลายเออร์มากขึ้นจำนวนบล็อกจึงลดลง

5.4 การเปรียบเทียบค่าความผิดพลาดกำลังสองของวิธีการต่างๆ

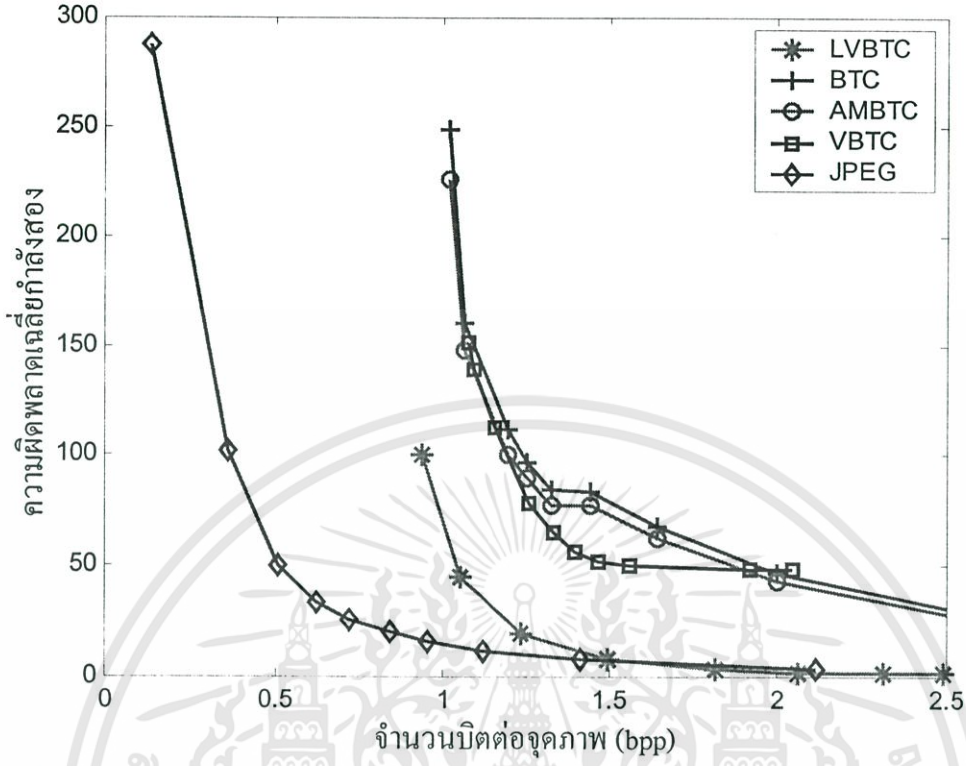
ในงานวิจัยได้ทำการเปรียบเทียบผลการวิจัยโดยวิธีการที่ได้นำเสนอคือ (LVBTC) และนำมาเปรียบเทียบกับวิธีการอื่นได้แก่ การเข้ารหัสแบบบล็อกคงที่(BTC) การเข้ารหัสแบบบล็อกคงที่โดยใช้ค่าสมบูรณ์โมเมนต์ (AMBTC) การเข้ารหัสแบบบล็อกหลายขนาดโดยใช้ค่าเทรชโฮลในการแบ่งบล็อกย่อย (VBTC) ที่ได้กล่าวไว้ในบทที่ 3 และ การบีบอัดมาตรฐาน JPEG ซึ่งได้กล่าวไว้ในบทที่ 2



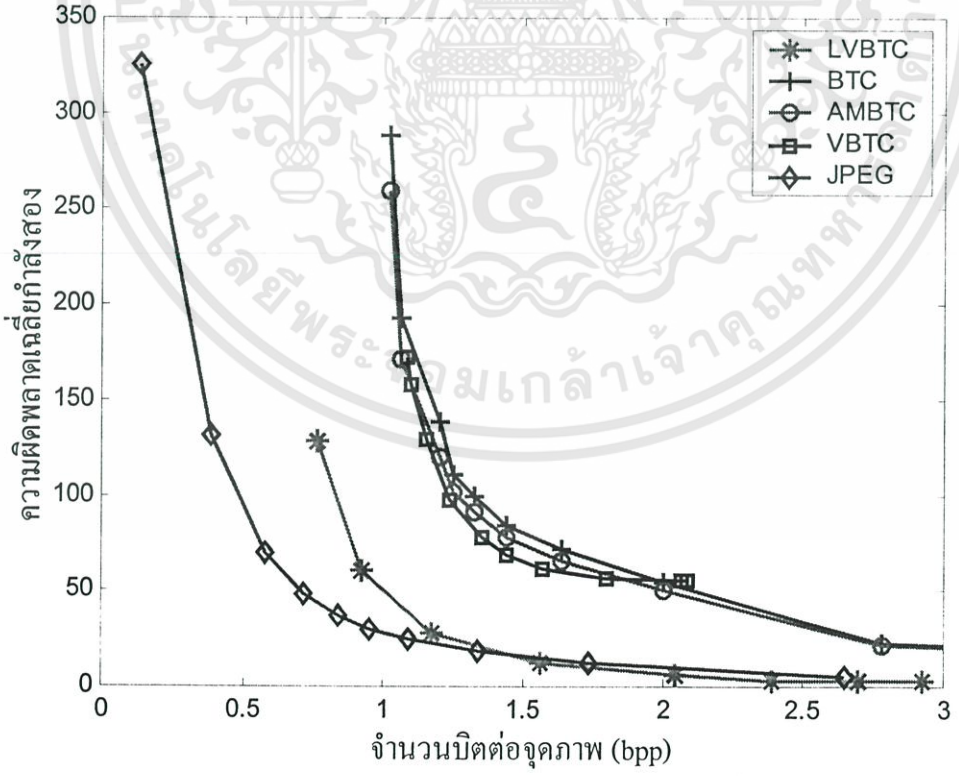
รูปที่ 5.13 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Lena ขนาด 512x512



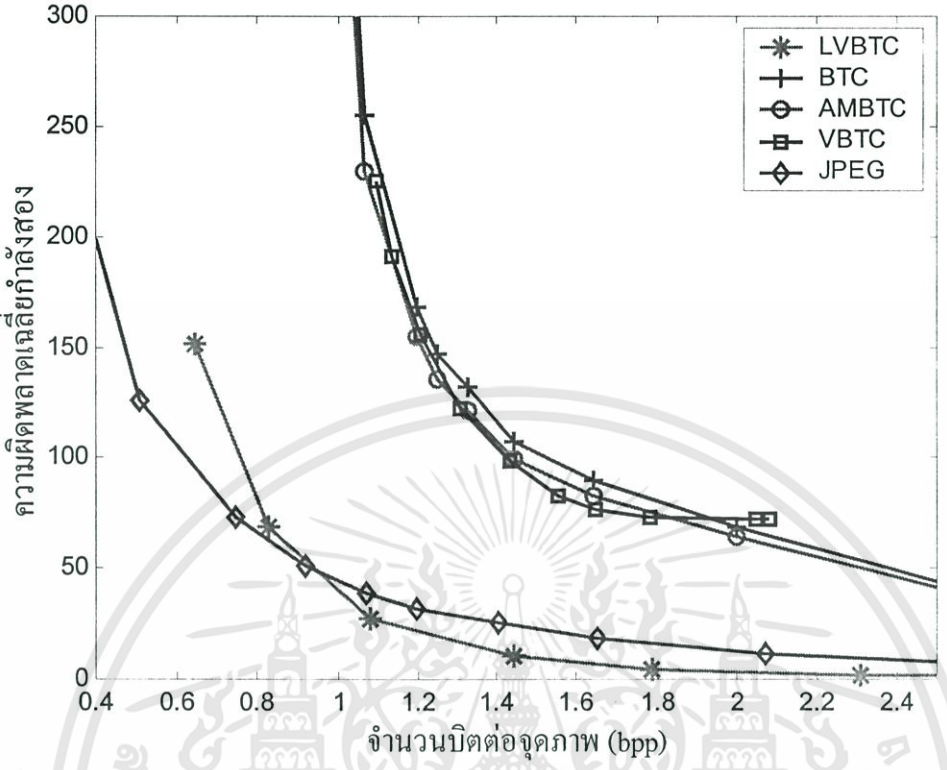
รูปที่ 5.14 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Peppers ขนาด 512x512



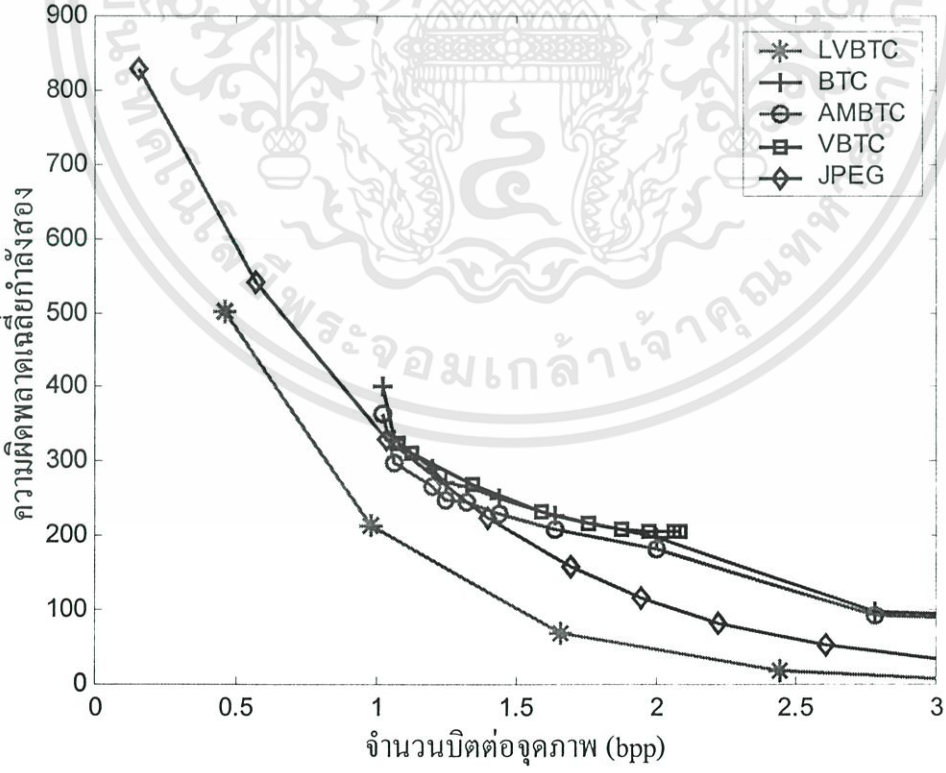
รูปที่ 5.15 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Jet ขนาด 512x512



รูปที่ 5.16 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Boat ขนาด 512x512



รูปที่ 5.17 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Lena ขนาด 256x256

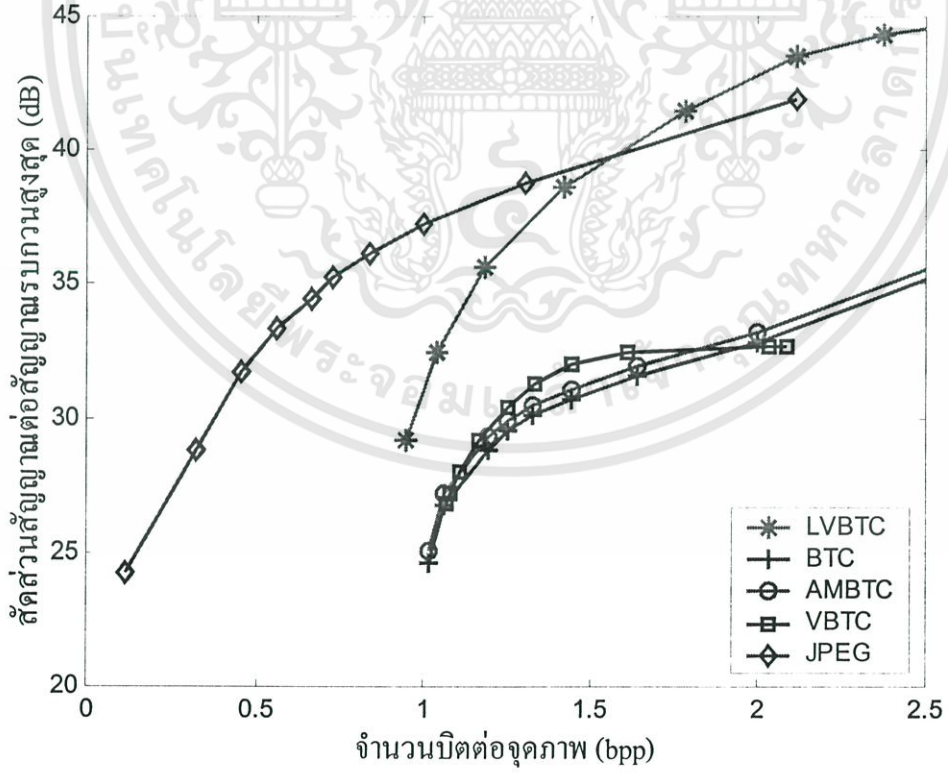


รูปที่ 5.18 แสดงการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพ Baboon ขนาด 256x256

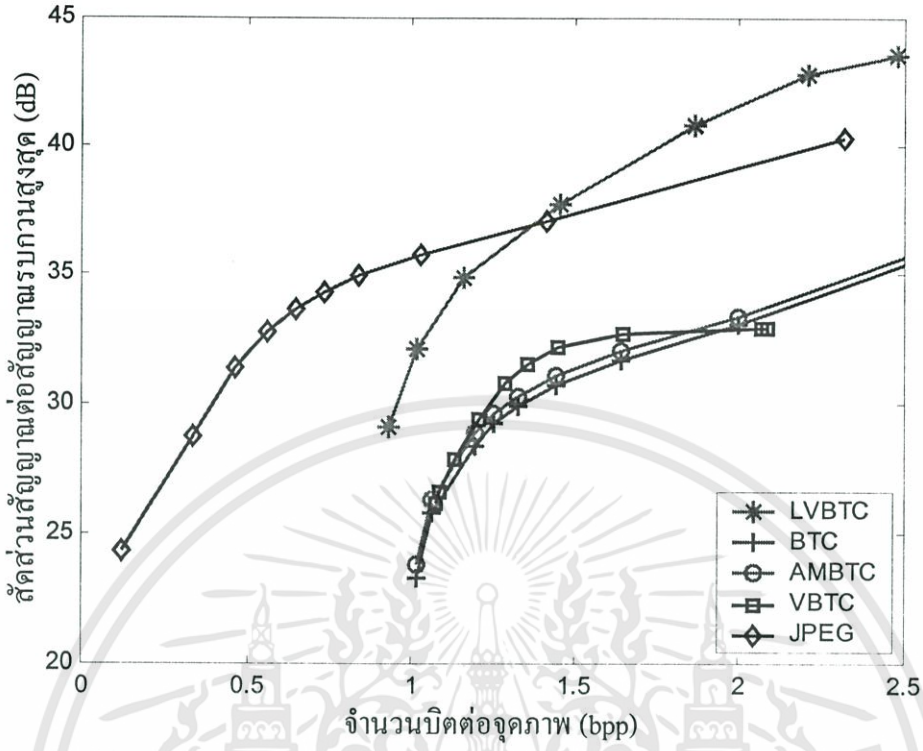
รูปที่ 5.13 ถึงรูปที่ 5.18 เป็นการเปรียบค่าความผิดพลาดเฉลี่ยกำลังสองของภาพต่างๆที่ใช้ในงานวิจัย ซึ่งผลที่ได้ในบางภาพเช่นภาพ baboon ขนาด 256x256 นั้นมีความผิดพลาดเฉลี่ยกำลังสองน้อยกว่าวิธีอื่นๆ แต่ในรูปอื่นนั้นค่าความผิดพลาดเฉลี่ยกำลังสองจะมีค่ามากกว่าวิธีการของการบีบอัดภาพมาตรฐาน JPEG ในช่วงที่อัตราบิตต่ำแต่จะมีความผิดพลาดกำลังสองน้อยว่าวิธีการของการบีบอัดภาพแบบ JPEG ในช่วงที่อัตราบิตสูง

5.5 การเปรียบเทียบค่าค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของวิธีการต่างๆ

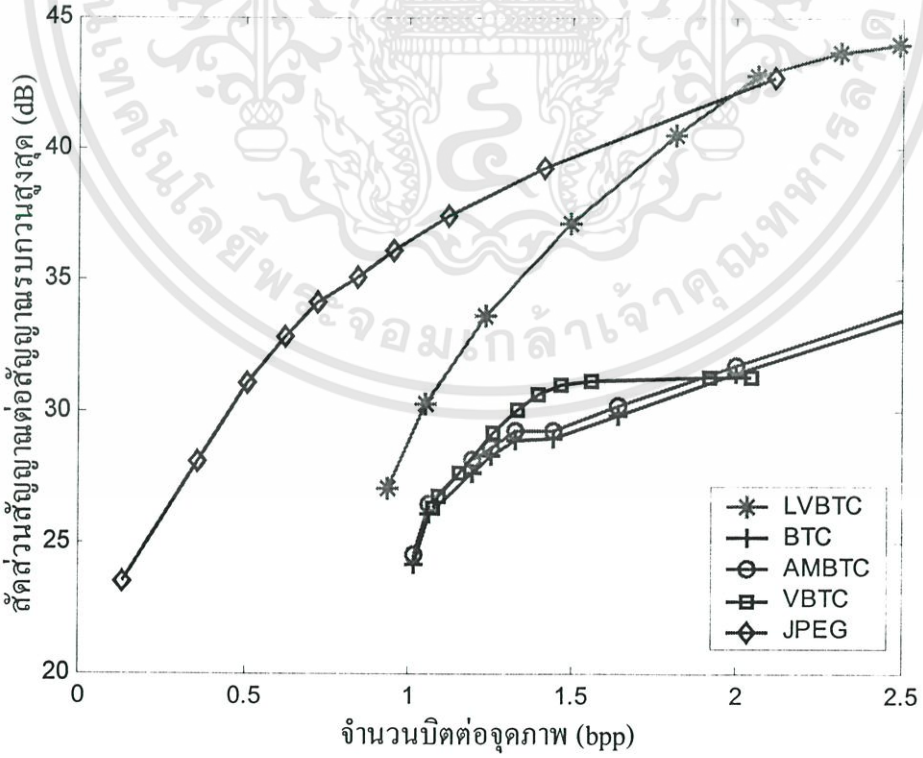
การเปรียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของวิธีการต่างๆ เป็นดังรูปที่ 5.19 ถึงรูปที่ 5.24 โดยวิธีการที่ได้นำมาเปรียบเทียบเหมือนกันกับที่กล่าวไว้ในหัวข้อที่ 5.4 พบว่าในช่วงที่อัตราบิตต่ำค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพต่างๆ ยกเว้นภาพ baboon ที่มีค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดมีค่าน้อยกว่าวิธีการบีบอัดภาพแบบ JPEG (ซึ่งค่าค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดค่ามากจะมีประสิทธิภาพดีกว่าค่าน้อย) ส่วนในช่วงที่มีอัตราบิตสูงจะมีค่ามาก กว่าวิธีการของ JPEG แต่อย่างไรก็ตามวิธีการที่ได้นำเสนอมีค่าค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดมากกว่าวิธีการของ BTC, AMBTC และ VBTC



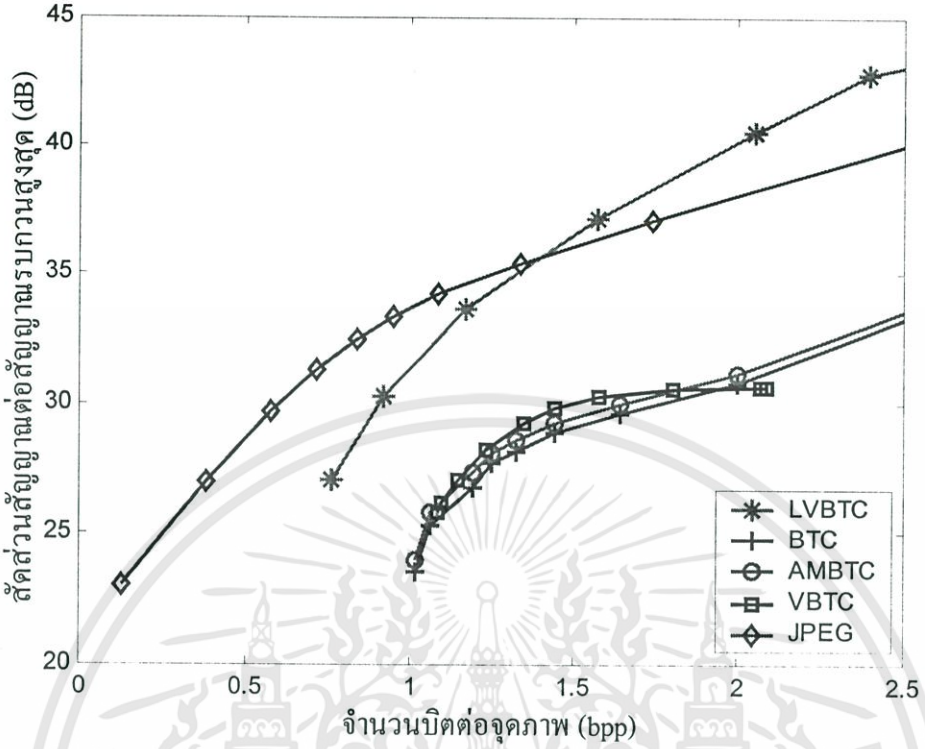
รูปที่ 5.19 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Lena ขนาด 512x512



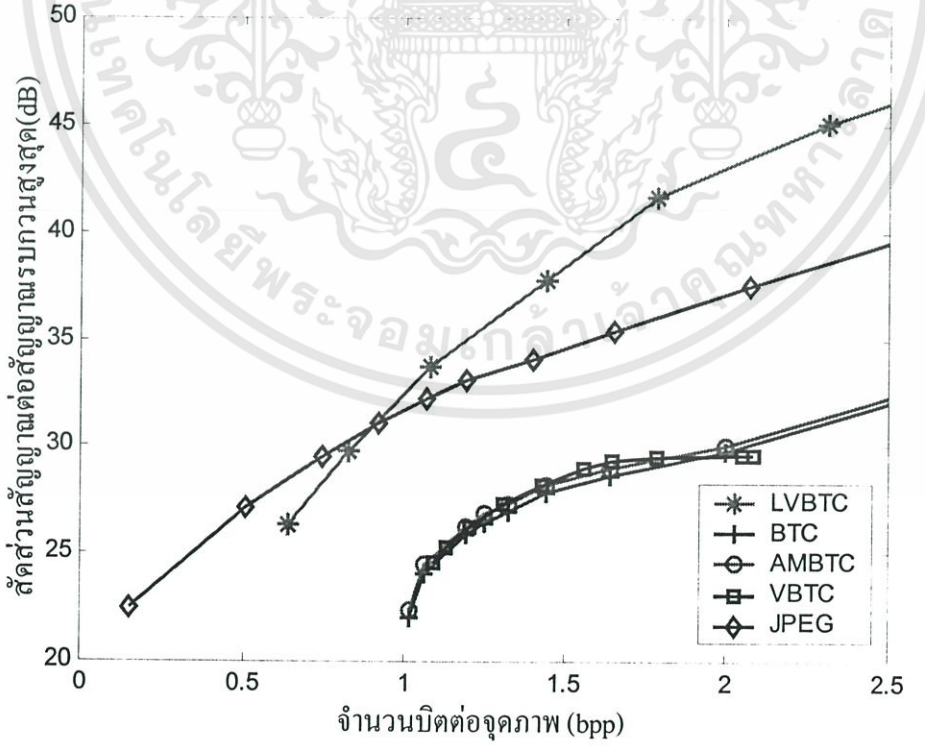
รูปที่ 5.20 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Peppers ขนาด 512x512



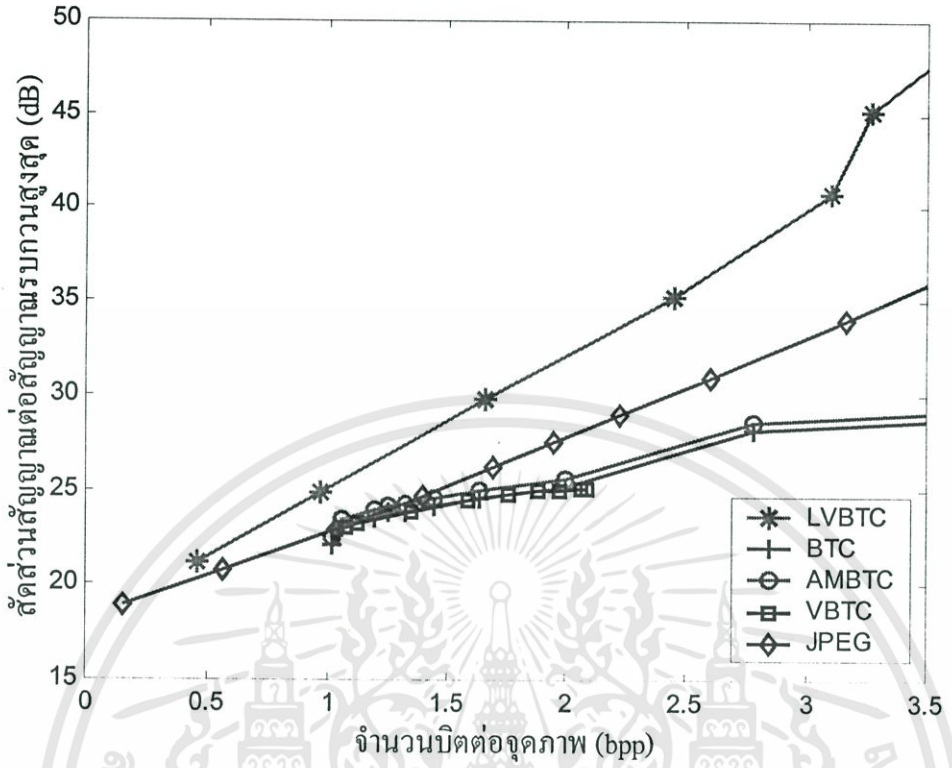
รูปที่ 5.21 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Jet ขนาด 512x512



รูปที่ 5.22 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Boat ขนาด 512x512



รูปที่ 5.23 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Lena ขนาด 256x256



รูปที่ 5.24 แสดงการเปรียบเทียบค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุดของภาพ Baboon ขนาด 256x256

ในส่วนสุดท้ายเป็นการเปรียบเทียบภาพที่ได้จากการสร้างกลับ ซึ่งได้เปรียบเทียบผลของการสร้างกลับภาพ Lena ขนาด 256x256 โดยผลที่ได้เป็นดังต่อไปนี้

ภาพ Lena ต้นแบบขนาด 512x512

(ก)

LVBTC , rate = 1.1174 bpp

MSE = 10.665, PSNR = 37.851

(ข)

รูปที่ 5.25 แสดงภาพที่ได้จากการสร้างกลับของวิธีการต่างๆ

VBTC bit rate = 1.0667 bpp

MSE = 136.3611, PSNR =26.7839

(ค)

BTC rate = 1.0156 bpp

MSE = 226.0661, PSNR = 24.5871

(ง)

AMBTC rate = 1.0156 bpp

MSE = 202.0255, PSNR= 25.0767

(จ)

JPEG rate = 1.0214 bpp

MSE = 10.7818, PSNR = 37.8039

(ฉ)

รูปที่ 5.25 (ต่อ)

ตารางที่ 5.7 การเปรียบเทียบวิธีการที่ได้นำเสนอกับวิธีการอื่น

วิธีการ	อัตราบิต	ค่าความผิดพลาดเฉลี่ยกำลังสอง	ค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด
HBTC [14]	1.425	47.32	31.38
VBTC [7]	1.425	36.06	32.56
L. W. chang [12]	1.425	30.06	33.35
วิธีการที่นำเสนอ	1.425	11.406	37.56

ตารางที่ 5.7 (ต่อ)

วิธีการ	อัตราบิต	ค่าความผิดพลาดเฉลี่ยกำลังสอง	ค่าสัดส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด
BTC [2]	1.625	47.32	31.38
VBTC [7]	1.625	23.99	34.33
L. W. chang [12]	1.625	21.78	34.75
วิธีการที่นำเสนอ	1.625	10.61	37.87

ตารางที่ 5.8 การเปรียบเทียบจำนวนบล็อกของวิธีการที่นำเสนอกับวิธีการอื่น

วิธีการ	อัตราบิต	บล็อกที่ผ่านการเข้ารหัส AMBTC					
		32x32	16x16	8x8	4x4	2x2	รวมทั้งหมด
HBTC [14]	1.425	20	123	716	10272	0	11131
VBTC [7]	1.425	0	0	3263	1827	6020	11110
L. W. chang [12]	1.425	14	278	1400	4113	5368	11173
วิธีการที่นำเสนอ	1.425	0	342	821	908	4608	6679
		บล็อกที่ผ่านการแปลง DCT					
		0	333	19	29	540	921

ตารางที่ 5.8 (ต่อ)

วิธีการ	อัตราบิต	บล็อกที่ผ่านการเข้ารหัส AMBTC					
		32x32	16x16	8x8	4x4	2x2	รวมทั้งหมด
BTC [2]	1.625	0	0	0	16384	0	16384
VBTC [7]	1.625	0	0	2782	2473	11132	16387
L. W. chang [12]	1.625	5	190	1469	4633	10060	16357
วิธีการที่นำเสนอ	1.625	0	128	1039	2774	7352	11293
		บล็อกที่ผ่านการแปลง DCT					
		0	330	16	48	704	1098

บทที่ 6
สรุปผลและแนวทางการพัฒนา

6.1 สรุปผลการวิจัย

ในงานวิจัยได้ทำการทดลองผลและทำการเปรียบเทียบกับวิธีการบีบอัดภาพแบบต่างๆซึ่งได้แก่ การเข้ารหัสแบบบล็อกคงที่ (BTC) การเข้ารหัสแบบบล็อกโดยใช้ค่าสมบูรณ์โมเมนต์ (AMBTC) การเข้ารหัสแบบบล็อกหลายขนาด (VBTC) และ การบีบอัดโดยวิธีการของ JPEG จากผลของที่ได้พบว่าวิธีการที่ได้นำเสนอ (LVBTC) มีประสิทธิภาพในการสร้างภาพกลับได้ดีกว่าวิธีการแบบอื่นๆ ที่นำมาเปรียบเทียบ ซึ่งสามารถสรุปผลการวิจัยได้ดังต่อไปนี้

1. การบีบอัดวิธีที่ได้นำเสนอเป็นวิธีการบีบอัดโดยแบ่งข้อมูลภาพออกเป็นบล็อกหลายขนาด ดังนั้นข้อมูลที่เป็นส่วนที่เพิ่มเข้ามาและมีผลต่ออัตราบิตคือส่วนที่เป็นต้นไม้รหัส (code tree) ถ้าข้อมูลภาพในแต่ละบล็อกมีรายละเอียดของภาพสูง ดังนั้นจำเป็นที่จะต้องเก็บต้นไม้รหัสไว้สำหรับถอดรหัสเป็นจำนวนบิตที่มากด้วยเช่นกัน ซึ่งหมายความว่าเมื่อคิดอัตราบิตเฉลี่ยแล้วทำให้มีอัตราบิตรวมทั้งหมดสูง ข้อดีคือถ้าเก็บรายละเอียดของภาพได้มากในการสร้างภาพกลับคืนย่อมมีประสิทธิภาพมากด้วย

2. ระดับการควอนไทซ์ภาพ (ในกรณีบล็อกภาพผ่านขบวนการแปลงโคซายน์เต็มหน่วย) นั้นมีผลต่อการสร้างกลับภาพและอัตราบิตเฉลี่ยของภาพ ในกรณีที่ให้ระดับการควอนไทซ์ (step size) มีระดับต่ำจากผลการวิจัยพบว่าทำให้การสร้างภาพกลับมีประสิทธิภาพสูงและมีความผิดเพี้ยนน้อยแต่อัตราบิตที่ใช้จะสูงด้วยเช่นกันทั้งนี้เพราะว่าเมื่อใช้ระดับการควอนไทซ์ต่ำ สัมประสิทธิ์ที่ใช้ในการสร้างกลับภาพมีจำนวนมากกว่าการที่ใช้ระดับการควอนไทซ์สูง

3. ในช่วงที่ใช้เทรชโฮลค่าต่ำๆ (เทรชโฮลของการแยกประเภทบล็อกที่มีรายละเอียดสูงและต่ำ) บล็อกส่วนมากจะผ่านขบวนการของการแปลงโคซายน์เต็มหน่วย (DCT) ดังนั้นประสิทธิภาพการสร้างภาพกลับจะดีกว่าการที่ใช้ค่าเทรชโฮลสูงเพราะว่าที่ค่าเทรชโฮลสูงๆ บล็อกส่วนมากจะผ่านขบวนการเข้ารหัสโดยใช้ค่าสมบูรณ์โมเมนต์ซึ่งใช้ระดับการควอนไทซ์เพียงสองระดับ

4. อัตราการบีบอัดขึ้นอยู่กับหลายปัจจัยด้วยกันคือ ค่าเทรชโฮล ระดับการควอนไทซ์ รวมไปถึงค่าลากรานจ์ที่ใช้

5. การเข้ารหัสแบบบล็อกมีความรวดเร็วและง่ายในการเข้ารหัส แต่ข้อเสียคือจำนวนบิตต่อพิกเซลมีค่าสูงซึ่งในงานวิจัยพบว่าการเข้ารหัสที่บล็อกขนาดต่ำกว่า 16×16 โดยที่ใช้บิตในการ

เข้ารหัสค่าเฉลี่ยเท่ากับ 8 บิตและบิตที่ใช้ในการเข้ารหัสค่าสมบูรณ์โมเมนต์เท่ากับ 8 บิต ตามลำดับ เมื่อคิดจำนวนบิตเฉลี่ยที่ใช้ต่อจุดภาพนั้นจะมีค่ามากกว่า 1 เสมอ

6. จากรูปกราฟการเปรียบเทียบค่าความผิดพลาดเฉลี่ยกำลังสอง พบว่าจำนวนบิตที่ใช้ต่อจุดภาพสูงและค่าความผิดพลาดเฉลี่ยยังมากใช้ช่วงอัตราบิตต่ำๆ ที่เป็นเช่นนี้เพราะว่าในการเข้ารหัสแบบบล็อกโดยใช้ค่าสมบูรณ์โมเมนต์ (AMBTC) นั้นใช้จำนวนบิตต่อจุดภาพสูงดังที่กล่าวมาแล้วในหัวข้อที่ 5 ดังนั้นถ้าค่าเทรชโฮลในการแยกประเภทบล็อกมีค่ามากทำให้บล็อกส่วนใหญ่ผ่านขบวนการของการเข้ารหัสแบบบล็อกโดยใช้ค่าสมบูรณ์โมเมนต์ซึ่งทำให้จำนวนบิตต่อจุดภาพสูงนั่นเอง

7. จากผลการวิจัยพบว่าการเข้ารหัสโดยแบ่งบล็อกหลายขนาดมีประสิทธิภาพดีกว่าการเข้ารหัสด้วยวิธีการ JPEG ในกรณีที่ภาพมีรายละเอียดสูงเช่นภาพ Baboon256 ซึ่งในการวิจัยพบว่ามีค่าความผิดพลาดเฉลี่ยกำลังสองน้อยกว่าการเข้ารหัสด้วยวิธีการ JPEG และในที่นี้จะเห็นได้ว่าวิธีการแบ่งบล็อกมีผลต่อข้อมูลที่จะนำมาเข้ารหัส

6.2 แนวทางการพัฒนา

1. วิธีการที่ได้นำเสนอนั้นจำนวนบิตต่อจุดภาพยังค่อนข้างมากในช่วงอัตราบิตต่ำ ถึงแม้ว่าจะมีคุณภาพในการสร้างกลับภาพได้ดี ดังนั้นแนวทางการพัฒนาคือการลดจำนวนบิตต่อจุดภาพโดยการใช้การเข้ารหัสโดยที่ไม่มีการสูญเสียข้อมูลเช่น การเข้ารหัสฮัฟแมน การเข้ารหัสเลขคณิต หรือการเข้ารหัสเอ็นโทรปีมาใช้เพื่อลดจำนวนบิตต่อจุดภาพ

2. ในงานวิจัยนี้ผู้ทำวิจัยใช้โปรแกรมการคำนวณทางคณิตศาสตร์ MATLAB เวอร์ชั่น 5.3.1 ซึ่งข้อดีคือการใช้งานง่ายและสะดวกในการเขียนโปรแกรม แต่ข้อเสียคือในการประมวลผลค่อนข้างช้าและเกิดปัญหาหน่วยความจำไม่เพียงพอในกรณีใช้กับภาพที่มีขนาดใหญ่หรือโปรแกรมที่มีการคำนวณ และเก็บตัวแปรมากๆ ดังนั้นแนวทางการพัฒนาคือการนำคอมพลายเลอร์ (Complier) ที่มีประสิทธิภาพมาใช้เช่น C /C++ หรือ คอมพลายเลอร์อื่นๆ

3. ในวิธีการที่ได้นำเสนอเป็นวิธีการที่เรียกว่าการเติบโตต้นไม้แบบบนลงล่าง (Top-down) แต่มีวิธีการที่ตรงกันข้ามที่เรียกว่าการเติบโตต้นไม้จากล่างขึ้นบน (Bottom-up) ซึ่งสามารถนำมาใช้กับวิธีการที่ได้นำเสนอได้ และนอกจากนั้นยังมีวิธีการตัดส่วนต้นไม้ออกจากโครงสร้างต้นไม้ (Pruning) [9] ซึ่งเป็นวิธีการที่มีประสิทธิภาพในการตัดส่วนของโครงสร้างต้นไม้ที่ไม่จำเป็นออกไป ซึ่งสามารถลดจำนวนบล็อกที่จะนำมาเข้ารหัสได้

4. ในส่วนของภาพที่ได้มีการแปลงข้อมูล ซึ่งในงานวิจัยใช้วิธีการแปลงโคซายน์เต็มหน่วย ดังนั้นแนวทางการพัฒนาสามารถนำวิธีการแปลงที่มีประสิทธิภาพในการสร้างกลับภาพได้ดีมาใช้เพื่อช่วยให้ประสิทธิภาพของการสร้างกลับภาพทำได้ดี

เอกสารอ้างอิง

[1] Alsaka Y. A., and Lee D. A. " Three levels block truncation coding" Proceeding IEEE 1990.

[2] Delp E. J., and Mitchell O. R. "Image Compression Using Block Truncation Coding," IEEE Transactions on Communications, vol. 27, Sep. 1979. pp. 1335-1342.

[3] Franti P., Nevalainen O., and Kaukoranta T. "Compression of Digital Image by Block Truncation Coding: A Survey" The Computer Journal, vol. 37, no. 4, 1994. pp. 308-332.

[4] Halverson D. R., Griswold N. C., and Wise G. L. "A generalized Block Truncation Coding Algorithm for Image Compression," IEEE Transaction on Acoustics, Speech, and Signal Processing, vol.ASSP-32, no.3, June. 1984. pp. 664-668.

[5] Lema M. D., and Mitchell O. R. "Absolute moment block truncation coding and its application to color images," IEEE Transaction on Communications, vol. 32, Oct. 1984. pp. 1148-1157.

[6] Yiyan W., and David C. C. "Multilevel Block Truncation Coding Using a Minimax Error Criterion for High-Fidelity Compression of digital Images," IEEE Transactions on Communications, vol. 41, no. 8, Aug. 1993. pp. 1179-1191.

[7] Kamel M., Sun C. T., and Guan L. "Image Compression by Variable Block Truncation Coding with Optimal Threshold," IEEE Transaction on Signal Processing. vol. 39, no.1, 1991. pp. 208-212.

[8] Samet H. "The quadtree and related hierarchical data structure," ACM Computing Surveys vol. 16, no. 2, June 1984. pp. 187-260.

[9] Chou P. A., Lookabaugh T., and Gray R. M. "Optimal pruning with applications to tree-structured source coding and modeling," IEEE Trans. Inform. Theory, vol. 35, no.2, Mar. 1989. pp. 299-315.

[10] Gersho A., and Gray R. M. **Vector Quantization and Signal Compression**. Kluwer Academic Publishes, Boston, 1992.

[11] Guido M. S., and Aggelos K. K. **Rate-Distortion Based Video Compression Optimal Video Frame compression and Object Boundary Encoding**. Kluwer Academic Publishes,1992.

[12] Chang L. W., and Wang C. Y. "Image Compression Using Optimal Variable Block Truncation Coding," IEEE 3rd Workshop on Multimedia Signal Processing, 1999. pp. 413-418.

[13] Jacques V., and Allen G. "Image Compression with Variable Block size Segmentation," IEEE Transactions on Signal Processing, vol. 40, no. 8, Aug. 1992. pp. 2040-2059.

[14] Jennifer U. R., and Nasser M. N. "Hierarchical Block Truncation Coding," Optical Engineering, vol. 30 no. 5, May. 1991. pp. 551-556.

[15] Khawne A., Kawabata T., and Noppanakeepong S. " An Image Compression Based on a Quadtree and Variable Block Truncation Code," Technical report of IEICE, vol.101 no. 304. IT 2001. pp. 35~42.

[16] Khawne A., and Noppanakeepong S. "Synthetic Aperture Radar (SAR) Image Compression with Optimum Quadtree Growing and Variable Absolute Moment Block Truncation Code," International Symposium on Communications and Information Technology, Chiang Mai, Thailand. 2001.

[17] Khawne A., and Kawabata T. "SAR Image Coding with Optimally Grown Quadtree and Block Truncation Code," Proceeding of JUSST program. UEC, Tokyo, JAPAN. 2001.

[18] Riskin E. A., and Gray R. M. "A Greedy Tree Growing Algorithm for the Design of Variable Rate Vector Quantizers," IEEE Transactions on Signal Processing, vol. 39, no.11, Nov. 1991. pp 2500-2507.

[19] Proakis J. G. **Digital Communications.** McGraw-Hill, Fourth Edition, 2001.

[20] Rafael C. G., and Richard E. W. **Digital Image Processing.** Addison-Wesley Publishing Company, 1992.

[21] Lloyd S. P. "Least Squares Quantization in PCM," IEEE Transaction on Information Theory, vol. IT-28, pp. 129-137, Mar. 1982.

[22] ศิริพงษ์ วงษ์คาร. "การลดข้อมูลภาพถ่ายดาวเทียมด้วยการควอนไทซ์แบบเวกเตอร์" วิทยานิพนธ์ วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมไฟฟ้า บัณฑิตวิทยาลัย,สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2540.

[23] บัณฑิต โรจน์อารยานนท์. **หลักการไฟฟ้าสื่อสาร** พิมพ์ครั้งที่สี่ สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย, 2537

[24] Thomas M. C., and Joy A. T. **Elements of Information Theory.** John-Wiley&sons, INC. 1991.

[25] Anil K. J. **Fundamentals of Digital Image Processing.** Prentice-Hall, Inc. 1989.

[26] Gregory K. W. " The JPEG still picture compression standard," IEEE Transactions on Consumer Electronics, pp. 2500-2507,Dec. 1991.

[27] Darrel H., Greg A. H., and Peter D. J. Jr. **Introduction to Information Theory and Data Compression.** CRC press, 1998.

[28] Pitas I. **Digital Image Processing Algorithms and Applications.** John-Wiley&sons, INC. 2000.

[29] นิสาชล โตอดิเทพ. **โครงสร้างข้อมูล** พิมพ์ครั้งที่สอง สำนักพิมพ์โอเดียนสโตร์. 2541.

[30] Chun-Hung K., Chang-Fuu C., and Wen-Nan H., " A compression Algorithm Based on Classified Interpolative Block Truncation Coding and Vector Quantization," Journal of Information Science and Engineering, Feb. 1999.

[31] The USC-SIPI image database *http://sipi.usc.edu/services/database/Database.html*

ภาคผนวก

ภาคผนวก ก
วิธีการของตัวคูณลากรานจ์

วิธีการของตัวคูณลากรานจ์เป็นวิธีที่รู้จักกันวิธีหนึ่งที่ใช้เพื่อหาคำตอบของปัญหาที่มีเงื่อนไขหรือข้อจำกัดในการทำให้มีค่าต่ำที่สุดในรูปแบบที่ต่อเนื่อง

ทฤษฎีหลัก

ทฤษฎีที่ 1 ให้ S_B เป็นเซตจำกัดและ $B \in S_B$ เป็นจำนวนของเซตนั้นๆ ให้ $R(B)$ และ $D(B)$ เป็นฟังก์ชันจำนวนจริงที่ถูกกำหนดใน S_B แล้วสำหรับค่าของ $\lambda \geq 0$ คำตอบที่ดีที่สุด $B^*(\lambda)$ ในปัญหาที่ไม่มีข้อจำกัดแล้วนั้น

$$\min_{B \in S_B} \left(D(B) + \lambda R(B) \right) \tag{ก.1}$$

เป็นคำตอบที่ดีที่สุดของปัญหาที่มีเงื่อนไข

$$\min_{B \in S_B} D(B), \text{ภายใต้เงื่อนไข } R(B) \leq R\left(B^*(\lambda)\right)$$

<u>**พิสูจน์**</u> สมมติว่าทฤษฎีที่กล่าวมานั้นผิด การสมมติดังกล่าวนั้นยังคงเป็น $B \in S_B$ ดังนั้น

$$D(B) < D\left(B^*(\lambda)\right) \text{และ } R(B) < R\left(B^*(\lambda)\right)$$

ดังนั้น $D(B) + \lambda R(B) < D\left(B^*(\lambda)\right) + \lambda R\left(B^*(\lambda)\right)$ ซึ่งมีข้อขัดแย้งกันเพราะว่า $B^*(\lambda)$ คือคำตอบที่ดีที่สุดกับปัญหาที่มีเงื่อนไข

ในทางทฤษฎีไม่ได้การันตีคำตอบในกรณีปัญหาที่มีเงื่อนไข แต่ยังมีปัญหาเช่นเดียวกันนี้ซึ่งคำตอบเหมือนกันกับปัญหาที่ไม่มีเงื่อนไขแต่อย่างไรก็ตามปัญหาสำคัญก็คือถ้า $R\left(B^*(\lambda)\right)$ เกิดขึ้นแล้วมีค่าเท่ากับขอบเขตบน $R\max$, $B^*(\lambda)$ คือคำตอบที่ถูกพิจารณากับปัญหาที่มีเงื่อนไข

$$\min_{B \in S_B} D(B), \text{ภายใต้เงื่อนไข } R(B) \leq R\max$$

ปัญหาซึ่งยังคงหาคำตอบที่ว่าทำอย่างไรที่จะหา λ ซึ่ง $R\left(B^*(\lambda)\right) = R\max$

ทฤษฎีของ λ

ทฤษฎีที่ 2 ถ้า $R\left(B^*(\lambda_1)\right) > R\left(B^*(\lambda_2)\right)$ แล้ว

$$\lambda_2 \geq -\frac{D\left(B^*(\lambda_1)\right)-D\left(B^*(\lambda_2)\right)}{R\left(B^*(\lambda_1)\right)-R\left(B^*(\lambda_2)\right)} \geq \lambda_1 \qquad \text{(ก.2)}$$

พิสูจน์ โดยค่าสูงสุดของ $B^*(\lambda_1)$ ดังนั้น

$$D\left(B^*(\lambda_1)\right)+\lambda_1 R\left(B^*(\lambda_1)\right) \leq D\left(B^*(\lambda_2)\right)+\lambda_1 R\left(B^*(\lambda_2)\right) \qquad \text{(ก.3)}$$

หาค่าสำหรับ λ_1 ซึ่ง $R\left(B^*(\lambda_1)\right) > R\left(B^*(\lambda_2)\right)$ ผลก็คือ

$$-\frac{D\left(B^*(\lambda_1)\right)-D\left(B^*(\lambda_2)\right)}{R\left(B^*(\lambda_1)\right)-R\left(B^*(\lambda_2)\right)} \geq \lambda_1 \qquad \text{(ก.4)}$$

โดยการพิสูจน์สมการทางขวามือของทฤษฎีและสมการทางซ้ายมือก็สามารถพิสูจน์ได้เช่นเดียวกัน โดยที่ค่าสูงสุดของ $B^*(\lambda_2)$ เป็นดังนี้

$$D\left(B^*(\lambda_2)\right)+\lambda_2 R\left(B^*(\lambda_2)\right) \leq D\left(B^*(\lambda_1)\right)+\lambda_2 R\left(B^*(\lambda_1)\right) \qquad \text{(ก.5)}$$

แก้สมการสำหรับ λ_2 และโดยที่ $R\left(B^*(\lambda_1)\right) > R\left(B^*(\lambda_2)\right)$ ผลที่ได้ก็คือ

$$\lambda_2 \geq -\frac{D\left(B^*(\lambda_1)\right)-D\left(B^*(\lambda_2)\right)}{R\left(B^*(\lambda_1)\right)-R\left(B^*(\lambda_2)\right)} \qquad \text{(ก.6)}$$

ทฤษฎี λ ชี้ให้เห็นว่าจะให้ผลลัพธ์ออกมาเป็น 2 คำตอบ อัตราส่วนของการเปลี่ยนแปลงในความผิดเพี้ยนที่มากที่สุดกับการเปลี่ยนแปลงของบิตที่ต้องการเป็นขอบเขตระหว่างตัวคูณสองตัวโดยเฉพาะถ้าเซตของคำตอบถูกกำหนดโดยผลคูณลากรานจ์ ผลที่ได้ในความผิดเพี้ยนซึ่งเป็นฟังค์ชั่นอนุพันธ์ของอัตราบิตที่จุดเดียวกัน แล้วมันเป็นไปตามทฤษฎีที่ได้กล่าวมาแล้วข้างต้นที่ว่า λ ที่จุดดังกล่าวนั้นคืออนุพันธ์ของความผิดเพี้ยนโดยสัมพันธ์กับอัตราบิตดังต่อไปนี้

$$-\frac{dD}{dR} = \lambda \qquad \text{(ก.7)}$$

โดยการประยุกต์ผลดังกล่าวกับ ORDF (Optimal Rate Distortion Function) ซึ่งจะได้ว่า

$$\frac{dD}{dR} = -\frac{1}{\lambda} \qquad \text{(ก.8)}$$

ดังนั้นทฤษฎี λ เป็นพื้นฐานของการหาจุด convex อย่างเร็ว

คำตอบใน convex hull

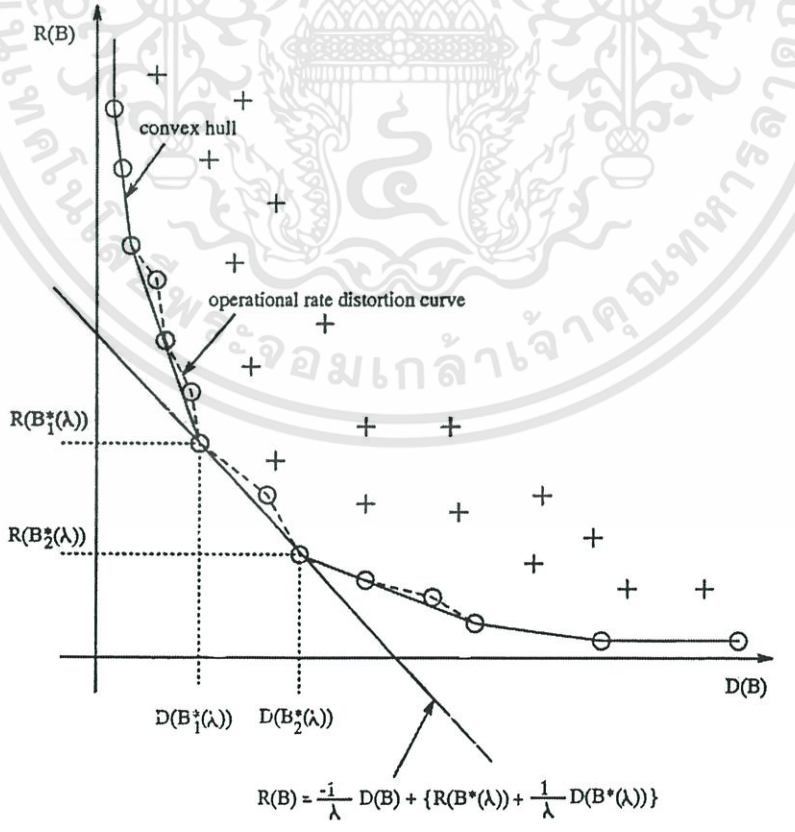
จากทฤษฎีอัตราบิต-ความผิดเพี้ยนเราทราบว่าฟังก์ชันอัตราบิต-ความผิดเพี้ยนคือฟังก์ชัน convex ที่ไม่มีการเพิ่มค่า ในส่วนนี้จะแสดงว่าตัวดูณวิธีลากรานจ์สามารถใช้ได้เพียงแค่หาจุดของ ORDF ซึ่งเป็นส่วนของ convex hull ของ ORDF

สมมติว่า $B^*(\lambda)$ คือคำตอบที่ดีที่สุดโดยที่ $\min_{B\in S_B}(D(B)+\lambda R(B))$ นั่นหมายถึง

$$D(B^*(\lambda))+\lambda R(B^*(\lambda))\le D(B)+\lambda R(B) \qquad (ก.9)$$

เมื่อ $B\in S_B$ สามารถเป็นเวกเตอร์ใดๆที่มีค่ายอมรับได้ จัดเทอมสมการใหม่จะได้

$$R(B)\ge -\frac{1}{\lambda}D(B)+\left\{R(B^*(\lambda))+\frac{1}{\lambda}D(B^*(\lambda))\right\} \qquad (ก.10)$$



รูปที่ ก.1 ความสัมพันธ์ของอัตราบิตและความผิดเพี้ยน

พิจารณากรณีเมื่อเป็นเครื่องหมายเท่ากับในสมการ (ก.10) แล้ว (ก.10) กำหนดเป็นเส้นตรงในระนาบอัตราบิต-ความผิดเพี้ยน จากสมการ (ก.10) พบว่าทุกจุด $R(B), D(B)$ ของ QF มักจะอยู่ทางด้านขวามือเกินครึ่งของระนาบอัตราบิต-ความผิดเพี้ยน ด้วยเหตุนี้เซตของคำตอบที่ดีที่สุด $D(B^*(\lambda))+\lambda R(B^*(\lambda))$ อยู่บนเส้นตรงดังกล่าว ดังที่กล่าวมาแล้วนั้นว่าความชันของเส้นตรงเท่ากับ $-\frac{1}{\lambda}$ อัตราบิต $R(B^*(\lambda))$ คือฟังก์ชันที่ไม่เพิ่มค่าขึ้นของ λ และความผิดเพี้ยน $D(B^*(\lambda))$ คือฟังก์ชันที่ไม่ลดลงของ λ ดังนั้นถ้าเราเปลี่ยนค่าของ λ จากค่าศูนย์ไปจนถึงค่าอนันต์และเชื่อมต่อจุด $D(B^*(\lambda)), R(B^*(\lambda))$ด้วยเส้นตรงแล้วนั้น จะได้convex hull ของ ORDF และในทำนองเดียวกันถ้าจุดของ ORDF ซึ่งไม่อยู่บน convex hull นั้นก็ไม่สามารถหาได้โดยใช้วิธีการประมาณของลากรานจ์เพราะว่าการประมาณดังกล่าวไม่สามารถหาเส้นตรงที่จะมาเชื่อมต่อกันได้ ซึ่งการแบ่งระนาบเช่นนี้นั้น ทุกๆคำตอบจะเป็นของส่วนที่อยู่ทางด้านขวามือเหนือหรือเกินครึ่งของระนาบทั้งสิ้น

ภาคผนวก ข

โปรแกรม MATLAB ที่ใช้ในการทดลอง

ฟังก์ชันหาค่าลากรานจ์ของบล็อกภาพ การเข้ารหัสแบบการเข้ารหัสโดยเก็บรักษาค่าสัมบูรณ์โมเมนต์และการเข้ารหัสโดยการแปลงโคซายน์เต็มหน่วย

```
clc;
clear;

I =imread('image file');
 [I1 I2 I3] = size(I);
   if I3 == 1
      II = I;
   else
      II = rgb2gray(I);
   end
Th = 10;
step = 16;
Count_blkbtc = 0;
Count_blkdct = 0;
[M1 N1]= size(II);
img = double(II);
dim = size(img,1);
M = [16, 8, 4, 2];          %% Size of block
B_cnt1 = dim.^2./(M(1)*M(1));%% # of blocks per image
B_cnt  = dim./M;
B_opt = size(M,2);     %% # of block sizes available
l = [6, 10, 14, 18];
img_recon = zeros(dim,dim);
bin_recon = zeros(dim,dim);
img_final = zeros(dim,dim);

lambda = [1, 5, 10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5210];
lambda_sz = size(lambda, 2);
J = zeros(B_cnt(1),B_cnt(1));   %% Block Cost Function
D = zeros(B_cnt(1),B_cnt(1));   %% Block Distortion
R = zeros(B_cnt(1),B_cnt(1));   %% Block Rate

% QUADTREE CODE-
%   | 0 | 0 0 0 0 | - use 16x16 block (0 bits per 16x16 pixels)
%   | 1 | 0 0 0 0 | - use 4 8x8 blocks (1 bit per 16x16 pixels)
%   | 1 | 1 0 0 0 | - use 3 8x8 blocks & 4 4x4 blocks (5 bits per 16x16 pixels)
%   | 1 | 1 0 1 0 | - use 3 8x8 blocks & 4 4x4 blocks (5 bits per 16x16 pixels)
%   | 1 | 1 1 1 1 | 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  - use 4 8x8 blocks & 4 2x2
blocks
QTcode = zeros(B_cnt1,22,lambda_sz);
```

```
codebook = zeros(85,2,B_cnt1,lambda_sz);
bp_rest = zeros(dim,dim,lambda_sz);

% per pixel QTrate
QTrate = [0,1,5,21]/(M(1)*M(1));
% Defined codebook
codebook16 = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define Block image
B16 = zeros(M(1),M(1));       % 16x16 block
B8  = zeros(M(2), M(2),4);    % 8x8 block
B4  = zeros(M(3), M(3),4);    % 4x4 block
B2  = zeros(M(4), M(4),4);    % 2x2 block

for w=1:lambda_sz

  %% Loop through each 16x16 block and make the decision whether or
  %%  not to break down the block into 8x8 and/or 4x4 blocks
  k = 1;
    for i=1:B_cnt(1)
    for j=1:B_cnt(1)
     row = M(1)*(i-1);
     col = M(1)*(j-1);
     s = 1;
     t = 1;
     u = 1;
     codebook2 = zeros(64,2);
     codebook4 = zeros(16,2);
     codebook8 = zeros(4,2);

      %Create  16x16 block
      B16(:,:) = img(row+1:row+M(1),col+1:col+M(1));

      %Subdivide the 16x16 block into 4 8x8 blocks
     B8(:,:,1) = img(row+1:row+M(2), col+1:col+M(2));
    B8(:,:,2) = img(row+1:row+M(2), col+1+M(2):col+2*M(2));
    B8(:,:,3) = img(row+1+M(2):row+2*M(2), col+1:col+M(2));
    B8(:,:,4) = img(row+1+M(2):row+2*M(2), col+1+M(2):col+2*M(2));

      %Calculate the varience use as a threshold
      zigma= round(sqrt((mean(mean((B16).^2)))-((mean(mean(B16))).^2)));

   %Compare BTC cost and DCT cost
  if zigma <= Th
    %% For 16x16 block, find the AMBTC, distortion, bitrate, IAMBTC
```

```
        QTcode(k,1,w) = 1;

        %% For each 8x8 block, find the AMBTC, distortion, bitrate, IAMBTC
        %%   and cost function
        [D16, R16, B_recon16, Bitplane16, H16, L16] = blockdis(B16);
        J16 = D16 + lambda(w).*R16;

for p=1:4
        [D8(p), R8(p), B_recon8(:,:,p), Bitplane8(:,:,p), H8(:,:,p), L8(:,:,p)] =
        blockdis(B8(:,:,p));
        J8(p)  = D8(p) + lambda(w).*R8(p);
end

        % Find the average distortion,bitrate, & cost for the 4 8x8 blocks
        % Note that for the cost, you must add the additional bits for
        %   the quadtree code
        J8_QT = sum(J8)/4 + lambda(w)*QTrate(2);

        % Compare costs: if the cost of 4 8x8 blocks is less than the
        %  the cost of a single 16x16 block, break up the block
        if J16 > J8_QT

          % Update the quadtree code
         QTcode(k,2,w) = 1;
         codebook16 = [0 0];

          % Now, compare each 8x8 block with its corresponding 4x4 blocks
          for q = 1:4

            %Subdivide the 8x8 block into 4 4x4 blocks
           B4(:,:,1) = B8(1:4,1:4,q);
          B4(:,:,2) = B8(1:4,5:8,q);
          B4(:,:,3) = B8(5:8,1:4,q);
          B4(:,:,4) = B8(5:8,5:8,q);

            %% each 4x4 block, find the AMBTC, distortion, bitrate, IAMBTC
            %%   and cost function
            for p = 1:4
           [D4(p,q), R4(p,q), B_recon4(:,:,p), Bitplane4(:,:,p), H4(:,:,p) L4(:,:,p)] =
            blockdis(B4(:,:,p));
            J4(p,q) = D4(p,q) + lambda(w).*R4(p,q);
            end

             J4_QT =  sum(J4(:,q))/4 + lambda(w)*QTrate(3);

            % Compare costs; if the cost of 4 4x4 blocks is less than the
            %  the cost of a single 8x8 block, break up the block
```

```
if J8(q) > J4_QT

    % Update the quadtree code
    QTcode(k,2+q,w) = 1;
    % Keep codebook
    codebook8(s,:) = [0 0];
for r = 1:4

    %Subdivide the 4x4 block into 4 2x2 blocks
    B2(:,:,1) = B4(1:2,1:2,r);
    B2(:,:,2) = B4(1:2,3:4,r);
    B2(:,:,3) = B4(3:4,1:2,r);
    B2(:,:,4) = B4(3:4,3:4,r);

   %% For each 2x2 block, find the AMBTC, distortion, bitrate, invert
AMBTC
   %% and cost function
     for p=1:4
          [D2(p,r), R2(p,r), B_recon2(:,:,p), Bitplane2(:,:,p), H2(:,:,p),
       L2(:,:,p)] = blockdis(B2(:,:,p));
       J2(p,r) = D2(p,r) + lambda(w).*R2(p,r);
     end

      J2_QT =  sum(J2(:,r))/4 + lambda(w)*QTrate(4);

    if J4(r,q)> J2_QT

     QTcode(k,l(q)+r,w) = 1;
       % Keep codebook
       codebook4(t,:)=[0 0];
       % Create a "new" 4x4 idct block made up of 2x2 blocks
       %keep codebook 2x2
    codebook2(u:(u+3),:) = [H2(:) L2(:)];
    u = u + 4;
    t = t + 1;
    B4_new(1:2,1:2,r) = B_recon2(:,:,1);
    B4_new(1:2,3:4,r) = B_recon2(:,:,2);
    B4_new(3:4,1:2,r) = B_recon2(:,:,3);
    B4_new(3:4,3:4,r) = B_recon2(:,:,4);
        %keeping the bit plane
        bin4_new(1:2,1:2,r)= Bitplane2(:,:,1);
    bin4_new(1:2,3:4,r)= Bitplane2(:,:,2);
    bin4_new(3:4,1:2,r)= Bitplane2(:,:,3);
    bin4_new(3:4,3:4,r)= Bitplane2(:,:,4);

    J4_new(r) = J2_QT;
    D4_new(r) = sum(sum(D2(:,r)))/4;
```

```
        R4_new(r) = sum(sum(R2(:,r)))/4 + QTrate(4);
      else
          codebook4(t,:) = [H4(:,:,r) L4(:,:,r)];
        u = u + 4;
        t = t + 1;
        B4_new(:,:,r) = B_recon4(:,:,r);
        bin4_new(:,:,r) = Bitplane4(:,:,r);
        J4_new(r) = J4(r);
        D4_new(r) = D4(r);
        R4_new(r) = R4(r) + QTrate(3);
      end

    end %for r
     % Create a "new" 8x8 block made up of 4x4 blocks

          B8_new(1:4,1:4,q) = B4_new(:,:,1);
        B8_new(1:4,5:8,q) = B4_new(:,:,2);
        B8_new(5:8,1:4,q) = B4_new(:,:,3);
        B8_new(5:8,5:8,q) = B4_new(:,:,4);
        bin8_new(1:4,1:4,q)= bin4_new(:,:,1);
        bin8_new(1:4,5:8,q)= bin4_new(:,:,2);
        bin8_new(5:8,1:4,q)= bin4_new(:,:,3);
        bin8_new(5:8,5:8,q)= bin4_new(:,:,4);
        J8_new(q) = J4_QT;
        D8_new(q) = sum(sum(D4(:,q)))/4;
        R8_new(q) = sum(sum(R4(:,q)))/4 + QTrate(3);

      % If it costs less to use this 8x8 block, don't split it
      else
        codebook8(s,:)=[H8(:,:,q) L8(:,:,q)];
        t = t + 4;
        B8_new(:,:,q) = B_recon8(:,:,q);
        bin8_new(:,:,q) = Bitplane8(:,:,q);
        J8_new(q) = J8(q);
        D8_new(q) = D8(q);
        R8_new(q) = R8(q) + QTrate(2);
      end
      s = s+1;
end % for q


% Create a "new" 16x16 invert AMBTC block made up of 8x8 blocks
        B16_new(1:8,1:8) = B8_new(:,:,1);
        B16_new(1:8,9:16) = B8_new(:,:,2);
        B16_new(9:16,1:8) = B8_new(:,:,3);
        B16_new(9:16,9:16) = B8_new(:,:,4);
        bin16_new(1:8,1:8) = bin8_new(:,:,1);
        bin16_new(1:8,9:16) = bin8_new(:,:,2);
```

```
                bin16_new(9:16,1:8) = bin8_new(:,:,3);
                bin16_new(9:16,9:16) = bin8_new(:,:,4);


    % Store the cost, distortion and rate for the new image
                J(i,j) = sum(J8_new)/4;
                D(i,j) = sum(D8_new)/4;
                R(i,j) = sum(R8_new)/4;

    % Add the created 16x16 block to the reconstructed image
                img_recon(row+1:row+M(1),col+1:col+M(1)) = B16_new;
                bin_recon(row+1:row+M(1),col+1:col+M(1)) = bin16_new;

    % If it costs less to use the 16x16 block, don't split it
else

 % Add the 16x16 block to the reconstructed image
                codebook16 = [H16, L16];
                img_recon(row+1:row+M(1),col+1:col+M(1)) = B_recon16;
                bin_recon(row+1:row+M(1),col+1:col+M(1)) = Bitplane16;

                J(i,j) = J16;
                D(i,j) = D16;
                R(i,j) = R16;
end

 if sum(codebook16)>0
     codebook(1,:,k,w)= codebook16;
     else
         codebook(1,:,k,w)= [0 0];
     end
     if sum(codebook8)>0
         codebook(2:5,:,k,w) = codebook8;
     else
         codebook(2:5,:,k,w) = 0;
     end
     if sum(codebook4)>0
         codebook(6:21,:,k,w) = codebook4;
     else
         codebook(6:21,:,k,w) = 0;
     end
     if sum(codebook2)>0
             codebook(22:85,:,k,w) = codebook2;
     else
         codebook(22:85,:,k,w) = 0;
     end
                k = k+1;
                s = 0;
```

```
                t = 0;
                u = 0;
                codebook16 = [];
                codebook8  = [];
                codebook4  = [];
                codebook2  = [];
                clear  bp_rest ;
                bp_rest(:,:,w) = bin_recon;

else % J16> J_dct16

                QTcode(k,1,w) = 0;
                [D_dct16, R_dct16, B_idct16 ] = blk_calc1(B16, step);
                J_dct16 = D_dct16 + lambda(w).*R_dct16;

            for p=1:4
             [D_dct8(p), R_dct8(p), B_idct8(:,:,p)] = blk_calc1(B8(:,:,p), step);
            J_dct8(p)  = D_dct8(p) + lambda(w).*R_dct8(p);
            end

    % Find the average distortion.bitrate. & cost for the 4 8x8 blocks
    % Note that for the cost, you must add the additional bits for
    %   the quadtree code
                J_dct8_QT = sum(J_dct8)/4 + lambda(w)*QTrate(2);

    % Compare costs: if the cost of 4 8x8 blocks is less than the
    %  the cost of a single 16x16 block, break up the block
            if J_dct16 > J_dct8_QT

        % Update the quadtree code
                QTcode(k,2,w) = 1;

        % Now, compare each 8x8 block with its corresponding 4x4 blocks
       for q=1:4

        %Subdivide the 8x8 block into 4 4x4 blocks
                B4(:,:,1) = B8(1:4,1:4,q);
                B4(:,:,2) = B8(1:4,5:8,q);
                B4(:,:,3) = B8(5:8,1:4,q);
                B4(:,:,4) = B8(5:8,5:8,q);

        %% For each 4x4 block, find the DCT, distortion, bitrate, IDCT
        %%   and cost function
        for p=1:4
          [D_dct4(p,q), R_dct4(p,q), B_idct4(:,:,p)] = blk_calc1(B4(:,:,p), step);
           J_dct4(p,q) = D_dct4(p,q) + lambda(w).*R_dct4(p,q);
        end
```

```
J_dct4_QT =  sum(J_dct4(:,q))/4 + lambda(w)*QTrate(3);

% Compare costs: if the cost of 4 4x4 blocks is less than the
%  the cost of a single 8x8 block, break up the block
if J_dct8(q) > J_dct4_QT

 % Update the quadtree code
      QTcode(k,2+q,w) = 1;

   for r = 1:4

   %Subdivide the 4x4 block into 4 2x2 blocks
         B2(:,:,1) = B4(1:2,1:2,r);
         B2(:,:,2) = B4(1:2,3:4,r);
         B2(:,:,3) = B4(3:4,1:2,r);
         B2(:,:,4) = B4(3:4,3:4,r);

    %% For each 2x2 block, find the DCT, distortion, bitrate, IDCT
  %% and cost function
    for p=1:4

      [D_dct2(p,r), R_dct2(p,r), B_idct2(:,:,p)] = blk_calc1(B2(:,:,p),
       step);
       J_dct2(p,r) = D_dct2(p,r) + lambda(w).*R_dct2(p,r);
      end

       J_dct2_QT =  sum(J_dct2(:,r))/4 + lambda(w)*QTrate(4);

    if J_dct4(r,q)> J_dct2_QT

          QTcode(k,l(q)+r,w) = 1;


       % Create a "new" 2x2 idct block made up of 2x2 blocks

          B4_new(1:2,1:2,r) = B_idct2(:,:,1);
          B4_new(1:2,3:4,r) = B_idct2(:,:,2);
          B4_new(3:4,1:2,r) = B_idct2(:,:,3);
          B4_new(3:4,3:4,r) = B_idct2(:,:,4);
              J_dct4_new(r) = J_dct2_QT;
          D_dct4_new(r) = sum(sum(D_dct2(:,r)))/4;
          R_dct4_new(r) = sum(sum(R_dct2(:,r)))/4 + QTrate(4);
     else

              B4_new(:,:,r) = B_idct4(:,:,r);
              J_dct4_new(r) = J_dct4(r);
          D_dct4_new(r) = D_dct4(r);
          R_dct4_new(r) = R_dct4(r) + QTrate(3);
```

```
            end

        end %for r

        % Create a "new" 8x8 idct block made up of 4x4 blocks
            B_dct8_new(1:4,1:4,q) = B_idct4(:,:,1);
            B_dct8_new(1:4,5:8,q) = B_idct4(:,:,2);
            B_dct8_new(5:8,1:4,q) = B_idct4(:,:,3);
            B_dct8_new(5:8,5:8,q) = B_idct4(:,:,4);
            J_dct8_new(q) = J_dct4_QT;
            D_dct8_new(q) = sum(sum(D_dct4(:,q)))/4;
            R_dct8_new(q) = sum(sum(R_dct4(:,q)))/4 + QTrate(3);

    % If it costs less to use this 8x8 block, don't split it
    else
            B_dct8_new(:,:,q) = B_idct8(:,:,q);
            J_dct8_new(q) = J_dct8(q);
            D_dct8_new(q) = D_dct8(q);
            R_dct8_new(q) = R_dct8(q) + QTrate(2);
    end
end % for q


 % Create a "new" 16x16 idct block made up of 8x8 blocks
B_dct16_new(1:8,1:8) = B_dct8_new(:,:,1);
B_dct16_new(1:8,9:16) = B_dct8_new(:,:,2);
B_dct16_new(9:16,1:8) = B_dct8_new(:,:,3);
B_dct16_new(9:16,9:16) = B_dct8_new(:,:,4);

  % Store the cost, distortion and rate for the new image
 J_dct(i,j) = sum(J_dct8_new)/4;
D_dct(i,j) = sum(D_dct8_new)/4;
R_dct(i,j) = sum(R_dct8_new)/4;

  % Add the created 16x16 block to the reconstructed image
img_recon(row+1:row+M(1),col+1:col+M(1)) = B_dct16_new;

% If it costs less to use the 16x16 block, don't split it
else

  % Add the 16x16 block to the reconstructed image
img_recon(row+1:row+M(1),col+1:col+M(1)) = B_idct16;

J(i,j) = J_dct16;
D(i,j) = D_dct16;
R(i,j) = R_dct16;
kk = k;
k = kk + 1;
```

```
        end
      end  % for j
    end  % for i

end

   % find the total cost function, rate and distortion for the entire image
   %   (2x2, 4x4, 8x8, 16x16, best)
   JT(w) = sum(sum(J(:,:)))/(B_cnt(1)*B_cnt(1));
  DT(w) = sum(sum(D(:,:)))/(B_cnt(1)*B_cnt(1));
  RT(w) = sum(sum(R(:,:)))/(B_cnt(1)*B_cnt(1));


fprintf('J = \n');
fprintf('%8.4f\n',JT(w));
fprintf('D = \n');
fprintf('%8.4f\n',DT(w));
fprintf('R = \n');
fprintf('%8.4f\n',RT(w));


 %Calculate a number of blocks

% Block of BTC
 Blkbtc(:,:,w) = sum(QTcode(:,1,w)==1);
 Blkdct(:,:,w) = sum(QTcode(:,1,w)==0);
 Allblk(:,:,w) = Blkbtc(:,:,w)+Blkdct(:,:,w);
 fbtc = find(QTcode(:,1,w)==1);
 Blk16_btc(:,:,w) = sum(QTcode(fbtc,2,w)==0);
 Blk8_btc(:,:,w) = sum(sum(QTcode(fbtc,3:6,w)<1))- (Blk16_btc(:,:,w)*4);
 Blk4_btc(:,:,w) = sum(sum(QTcode(fbtc,7:22,w)<1))-(Blk8_btc(:,:,w)*4)-
(Blk16_btc(:,:,w)*16);
 Blk2_btc(:,:,w) = sum(sum(QTcode(fbtc,7:22,w)==1))*4;

 % Block of DCT
 fdct = find(QTcode(:,1,w)==0);
 Blk16_dct(:,:,w) = sum(QTcode(fdct,2,w)==0);
 Blk16_dct(:,:,w) = sum(QTcode(fdct,2,w)==0);
 Blk8_dct(:,:,w) = sum(sum(QTcode(fdct,3:6,w)<1))- (Blk16_dct(:,:,w)*4);
 Blk4_dct(:,:,w) = sum(sum(QTcode(fdct,7:22,w)<1))-(Blk8_dct(:,:,w)*4)-
(Blk16_dct(:,:,w)*16);
 Blk2_dct(:,:,w) = sum(sum(QTcode(fdct,7:22,w)==1))*4;
  fprintf('BTC and DCT blocks\n');
 fprintf('%8.4f\n',Blkbtc(:));
 fprintf('%8.4f\n',Blkdct(:));
 fprintf('block size 16x16 BTC, DCT\n');
 fprintf('%8.4f\n',Blk16_btc(:));
 fprintf('%8.4f\n',Blk16_dct(:));
```

```
fprintf('block size 8x8,BTC, DCT\n');
fprintf('%8.4f\n',Blk8_btc(:));
fprintf('%8.4f\n',Blk8_dct(:));
fprintf('block size 4x4 ,BTC, DCT\n');
fprintf('%8.4f\n',Blk4_btc(:));
fprintf('%8.4f\n',Blk4_dct(:));
fprintf('block size 2x2, BTC, DCT\n');
fprintf('%8.4f\n',Blk2_btc(:));
fprintf('%8.4f\n',Blk2_dct(:));
```

Calculate PSNR

```
PSNR = 10*log10(255*255./DT);
end % for w
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Function Calculate Distortion for AMBTC

```
function [D, R, B_recon, Bp, H, L] = blockdis(B)
        y = B;
        [m1 n1]=size(y);
        mu = mean(mean(y));
        H = mu;
        alpha = absdev1(y);
        L = alpha;
        aa = mu ;
        Bit = 16;
        bb = alpha;
        b = y > mu ; %the binary block
        Bp = b;
        K = sum ( sum ( b ) ) ;
          if sum(b)==0
             B = y;
          else
          if (K ~= m1*n1 ) & ( K ~= 0)
            aaa = aa - ((m1*n1*bb)./(2*(m1*n1-K)));
           bbb = aa + ((m1*n1*bb)./(2*K));
          B = b*bbb + (1- b )*aaa;
           end
       end
      Er=(y-B).^2;Er=sum(Er(:));
      D = Er./(m1*n1);
       B_recon = B;

       bitplane = length(b(:)');

        R = (Bit + bitplane)./(m1*n1);
```

%%%

```
%function to create blocks from code tree
%ฟังก์ชันการถอดรหัสของ Lagran variable block truncation encode โดยการถอดรหัสจากต้นไม้รหัส
bin_recon = bp_rest(:,:,1);
dim = size(bin_recon,1);
B_cnt  = dim./16;
r = 1;
p = 1;
q = [5, 9, 13, 17];;
bp_recon = zeros(dim,dim);
for i = 1:B_cnt(1)
   for j = 1:B_cnt(1)
     row = M(1)*(i-1);
        col = M(1)*(j-1);
         %Create  16x16 block
         B16(:,:) = bin_recon(row+1:row+M(1),col+1:col+M(1));
         %Subdivide the 16x16 block into 4 8x8 blocks
         B8(:,:,1) = bin_recon(row+1:row+M(2), col+1:col+M(2));
         B8(:,:,2) = bin_recon(row+1:row+M(2), col+1+M(2):col+2*M(2));
         B8(:,:,3) = bin_recon(row+1+M(2):row+2*M(2), col+1:col+M(2));
         B8(:,:,4) = bin_recon(row+1+M(2):row+2*M(2),
         col+1+M(2):col+2*M(2));
          CT = QTcode(r,:,1);
          if sum(CT)==0
            bp_recon = bin_recon;
           else
             for k=1:4
               if (CT(k+p)) > 0
                      B4(:,:,1) = B8(1:4,1:4,k);
                      B4(:,:,2) = B8(1:4,5:8,k);
                      B4(:,:,3) = B8(5:8,1:4,k);
                      B4(:,:,4) = B8(5:8,5:8,k);
                 for l=1:4
                      B2(:,:,1) = B4(1:2,1:2,l);
                      B2(:,:,2) = B4(1:2,3:4,l);
                      B2(:,:,3) = B4(3:4,1:2,l);
                      B2(:,:,4) = B4(3:4,3:4,l);
                   if (CT(l+q(k)))>0
                      B4_new(1:2,1:2,l) = B2(:,:,1);
                      B4_new(1:2,3:4,l) = B2(:,:,2);
                      B4_new(3:4,1:2,l) = B2(:,:,3);
                      B4_new(3:4,3:4,l) = B2(:,:,4);
                    else
                      B4_new(:,:,l) = B4(:,:,l);
                   end
```

```
end % for l
B8_new(1:4,1:4,k) = B4_new(:,:,1);
B8_new(1:4,5:8,k) = B4_new(:,:,2);
B8_new(5:8,1:4,k) = B4_new(:,:,3);
B8_new(5:8,5:8,k) = B4_new(:,:,4);
else
B8_new(1:8,1:8,k) = B8(:,:,k);
end
end % for k
%reconstruct from 8x8 to 16x16
B16_new(1:8,1:8) = B8_new(:,:,1);
B16_new(1:8,9:16) = B8_new(:,:,2);
B16_new(9:16,1:8) = B8_new(:,:,3);
B16_new(9:16,9:16) = B8_new(:,:,4);
%final reconstructed
bp_recon(row+1:row+M(1),col+1:col+M(1)) = B16_new;
end
end % for B_cnt
r=r+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This is m-file for BTC encoding
function [codeword,codebook] = stbtcencode(I,blksize)
%I is an image for encode
%blksize is  size for divided image to block
%codeout is a codeword (binary)
%codebook is a mean of data that higher and lower then data
[N M]=size(I);
oriimg=imgtoblock(I,[blksize blksize]);
[xx yy zz]=size(oriimg);
codedata = [];
codebook1 = [];
blockdata = [];
for jj =1:zz
img =imgtovector(oriimg(:,:,jj));
mu = mean(img);
zigma = std(img);
f1 = find(img >= mu);
f2 = find(img < mu);

max1 = max(img(f1));
max2 = max(img(f2));
min1 = min(img(f1));
min2 = min(img(f2));
max22=isempty(max2);
if ((max22~=1)|(min2>0)|(min2<0))
```

```
            codebook1(:,:,jj)=[mu zigma  ];
        else
            codebook1(:,:,jj)=[mu  0];
    end
    %conversion each picxel to '0' or '1'
    img(f1)= 1;
    img(f2)= 0;
    codedata(:,:,jj) = img;

end
codeword(:,:)=codedata(:,:);
codebook=codebook1;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This m-file for decode of BTC decode
function image = stbtcdecode(codeword,codebook,imsize,blksize)
%codeword is  a transform of image to binary code
%codebook is a code for keep original image
%imsize is a size of reconstruct image ( using [n m])
%blksize is  size for divided image to block
m=blksize*blksize;
code = vec2mat(codeword,m);
[n m] = size(code);
row = imsize(1);
col = imsize(2);
%n is a row of code
%m is a column of code
for v=1:n
   datablk(:,:,v)=vec2mat(code(v,:),blksize);
end
      k=1;
      for j =1:n
       a = codebook(:,:,j);
       data = imgtovector(datablk(:,:,j));
          for i = 1:blksize
          for l=1:blksize

             x = datablk(i,l,j);
                f1 = find(data==1);
             f2 = length(f1);
              if (x==1)
                        aa =a(1)+a(2)*sqrt((m-f2)./f2);
                        datablk(i,l,j)=aa ;
                 else
                        bb =a(1)-a(2)*sqrt(f2./(m-f2));
                        datablk(i,l,j)=bb;
                 end
```

```
            aa=0;
            bb=0;
              end
        end
         k=k+1;
      end
I = blockToimg(datablk,[row col]);
image = (I);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This is m-file for AMBTC encoding
function [codeword,codebook] = ambtcencode(I,blksize)
%I is an image for encode
%blksize is  size for divided image to block
%codeout is a codeword (binary)
%codebook is a mean of data that higher and lower then data
[N M]=size(I);
oriimg=imgtoblock(I,[blksize blksize]);
[xx yy zz]=size(oriimg);
codedata = [];
codebook1 = [];
blockdata = [];
for jj =1:zz
    img =imgtovector(oriimg(:,:,jj));
    mu = mean(img);
    alpha = absdev(img);
    f1 = find(img >= mu);
    f2 = find(img < mu);
    max1 = max(img(f1));
    max2 = max(img(f2));
    min1 = min(img(f1));
    min2 = min(img(f2));
        max22=isempty(max2);
      if ((max22~=1)|(min2>0)|(min2<0))
            codebook1(:,:,jj)=[mu  alpha ];
        else
            codebook1(:,:,jj)=[mu  0];
     end
     %conversion each picxel to '0' or '1'
        img(f1)= 1;
        img(f2)= 0;
        codedata(:,:,jj) = img;
end
codeword(:,:)=codedata(:,:);
codebook=codebook1;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%This m-file for decode of AMBTC
function image = ambtcdecode(codeword,codebook,imsize,blksize)
%codeword is  a transform of image to binary code
%codebook is a code for keep original image
%imsize is a size of reconstruct image ( using [n m])
%blksize is  size for divided image to block
m=blksize*blksize;
code = vec2mat(codeword,m);
[n m] = size(code);
row = imsize(1);
col = imsize(2);
%n is a row of code
%m is a column of code
for v=1:n
   datablk(:,:,v)=vec2mat(code(v,:),blksize);
end
      k=1;
      for j =1:n
       a = codebook(:,:,j);
       data = imgtovector(datablk(:,:,j));
           for i = 1:blksize
           for l=1:blksize
                  x = datablk(i,l,j);
                   f1 = find(data==1);
                   q = length(f1);
                  if (x==1)
                      bb = a(1)+((m*a(2))./(2*q));
                      datablk(i,l,j)=bb ;
                   else
                      aa = a(1)-((m*a(2))./(2*(m-q)));
                      datablk(i,l,j)=aa;
                end
               aa=0;
              bb=0;
              end
         end
         k=k+1;

        end
I = blockToimg(datablk,[row col]);
image = (I);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This is m-file for encode BTC variable coding
function [codeword,codebook,index,S,numblk] =
var_btcencode(I,th,minsize,maxsize)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%I is an image for division ,but I should be double type
%th is a threshold.Note that the threshold value is specified
%as a value between 0 and 1 regardless of the class of I.
%If I is uint8,the threshold value you supply is
%multiplied by 255 to determine the actual threshold to use;If I is
%uint16,the threshold value you supply is multiplied by 65535
%minsize is a mininum divided block size ,that at least equal to 2
%maxsize is a maximun divided block size ,that at most equal to 4
[numblk,row_ind,col_ind]=var_btcencode1(I,th,minsize,maxsize);
S = qtdecomp(I,th,[minsize maxsize]);
%S = qtdecomp(I,th,[minsize maxsize]);
[a1 b1]=size(numblk);
blksize = numblk;
k=0;
p=1;
[s1 s2]=size(S);
codebook=[];
init_output=I;
while(k<a1)
for l =1:a1
  a2 = numblk(l,:);
  [vals,r,c] = qtgetblk(I,S,a2(1));
    a3 = length(r);
    image = zeros(a2(1),a2(1));
     for n =1 :a3
     vector = vals(:,:,n);
     val = imgtovector(vector);
     mu = round(mean(val));
     zigma = round(std(val));
     f1 = find(val >= mu);
     f2 = find(val < mu);

     max1 = max(val(f1));
     max2 = max(val(f2));
     min1 = min(val(f1));
     min2 = min(val(f2));
         max22=isempty(max2);
      if ((max22~=1)|(min2>0)|(min2<0))
           codebook1(:,:,p)=[(mu) (zigma)];
        else
           codebook1(:,:,p)=[(mu)  (0)];
      end
   val(f1)= 1;
   val(f2)= 0;
   row = r(n);
   col = c(n);
```

```
    pic = vec2mat(val(:,:),a2(1));
    image(:,:,n) = pic;
    p=p+1;
   end
   out = qtsetblk(init_output,S,a2(1),image);
  init_output=out;
  k=k+1;
 end
end
codeword=imgtovector(out);

%find the index of row and col and blocksize
a4=find(row_ind>0);
b4=find(col_ind>0);
index1=[row_ind(a4),col_ind(b4)];
k=1;
for i=1:a1
  num = numblk(i,2);
   for j=1:num
     ind(:,:,k)=[index1(k,:) numblk(i,1)];
     k=k+1;
   end
end
index=ind;
sizeind = length(index);
for r=1:sizeind
  codebook(:,:,r) = [codebook1(:,:,r)];
end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This is m-file for decode BTC variable coding
function image = var_btcdecode(codeword,codebook,c,r,imgsize)
%imgsize is a size of image ex. [256 256]
%transform codeword to image (black&white)
a = imgsize;
code = vec2mat(codeword,a(1));
%check index
image1  = zeros(a(1),a(2));
%determine posititon
%build Sparse matrix
image1 = runleng_decode(c,r);
S=vec2mat(image1,[imsize]);
%prepare matrix of image for instread sparse matrix

cbleng = length(codebook);
cwleng = length(codeword);
datablk=[];
```

```
blk = [1 2 4 8 16 32];
blkleng =length(blk);
l=1;
picture=image1;
while (l< cbleng)
   for j=1:blkleng
        [vals,r,c]=qtgetblk(code,S,blk(j));
        lengvals = length(r);

    if (r>0)
     for k=1:lengvals
        [svals1 svals2 svals3]= size(vals);
        datablk = zeros(svals1,svals2);
      for m=1:svals1
        for n=1:svals2
        a = codebook(:,:,l);
        x = vals(m,n,k);
                if (x==1)
                vals(m,n,k) = a(1);
                else
                vals(m,n,k) = a(2);
               end
       end
       end
       newvals(:,:,k) = vals(:,:,k);
      l=l+1;

     end
      out = qtsetblk(picture,S,blk(j),newvals);
     picture = out;
     newvals = [];
 else
  end
 end
end
image = (out);
```

ภาคผนวก ค
ผลงานวิจัยที่ได้รับการตีพิมพ์

[1] **Amnach K.**, Somchai O., Suthichai N., and Krit W., "Subband SAR Image Coding by Using Quadtree Decomposition on Variable Block Truncation Code",The 21st Asian Conference on Remote Sensing,Taipei, Taiwan, vol 1, Dec. 2000. pp. 510-515.

[2] **Amnach K.**, and Kawabata T., "SAR Image Coding with Optimally Grown Quadtree and Block Truncation Code", Proceeding of JUSST program. UEC, Tokyo, JAPAN. 2001.

[3] **Amnach K.**, Kawabata T., and Suthichai N., " An Image Compression Based on a Quadtree and Variable Block Truncation Code", Technical report of IEICE, vol.101, no. 304, IT 2001. pp.35~42.

[4] **Amnach K.**, and Suthichai N., "Synthetic Aperture Radar (SAR) Image Compression with Optimum Quadtree Growing and Variable Absolute Moment Block Truncation Code," International Symposium on Communications and Information Technology, Chiang Mai, Thailand. Sep. 2001. pp. 267-270.

ประวัติผู้เขียน

ชื่อ นายอำนาจ ขาวเน

เกิดวันที่ 7 สิงหาคม 2517 จังหวัดสุพรรณบุรี

การศึกษา ปีการศึกษา 2538-2540
ระดับปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมโทรคมนาคม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ประสบการณ์การทำงาน ปี พ.ศ. 2539-2542
ช่างโทรคมนาคมระดับ 3 แผนกควบคุมข่ายการติดต่อ
กองวิศวกรรม การสื่อสารแห่งประเทศไทย