การบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต

Image compressions by using Wavelet transform

นายจักรพงษ์ เสือมี รหัส 44370096

นายสิทธิชาติ เทิดสิทธิกุล รหัส 44370492

ห้องสมุดคณะวิศวกรรมศาสตร์
วันที่รับ 25 พ.ค. 2553
เลขทะเบียน 15000658
เลขเรียกหนังสือ ปร. ว223ก. 2547
มหาวิทยาลัยนเรศวร

ปริญญานิพนธ์เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2547

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	การบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต		
ผู้เสนอโครงงาน	นายจักรพงษ์	เสือมี	รหัส 44370096
	นายสิทธิชาติ	เทิดสิทธิกุล	รหัส 44370492
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.สุชาติ	แย้มเม่น	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2547		

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

..... ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น)

..... กรรมการ

(ดร.พนมขวัญ ริยะมงคล)

..... กรรมการ

(นายแสงชัย มังกรทอง)

หัวข้อโครงงาน	**การบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต**
ผู้ดำเนินโครงการ	**นายจักรพงษ์ เสือมี รหัส 44370096**
	นายสิทธิชาติ เทิดสิทธิกุล รหัส 44370492
อาจารย์ที่ปรึกษา	**ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มแม่น**
สาขาวิชา	**วิศวกรรมคอมพิวเตอร์**
ภาควิชา	**วิศวกรรมไฟฟ้าและคอมพิวเตอร์**
ปีการศึกษา	2547

.....

บทคัดย่อ

โครงงานนี้เป็นการศึกษาและพัฒนาด้วยโปรแกรมแมทแล็ปสำหรับการบีบอัดข้อมูล 2 มิติ โดยใช้วิธีการแปลงเวฟเล็ต ผลที่ได้จากโครงงานนี้ จะได้ค่าอัตราการบีบภาพอัดสูง คือ 2.33 นอกจากนี้ค่า SNR มีค่าเท่ากับ 316 [dB] และ PSNR มีค่าเท่ากับ 284 [dB] ซึ่งทำให้คุณภาพของภาพที่ได้จากการคลายการบีบอัดข้อมูลเหมือนกับภาพต้นแบบเมื่อมองด้วยตา

Project Title	Image compressions by using Wavelet transform
Name	Mr.Chakkrapong Suami ID 44370096
	Mr.Sittichat Therdsittikul ID 44370492
Project Advisor	Assistant Professor Suchart Yammen , Ph. D.
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic Year	2004

ABSTRACT

This project is to study and develop a Matlab program for compressing two-dimensional data by using the Wavelet transform. Form the project results, the maximum compression ratio is 2.33. Furthermore, the SNR value is 316 [dB] and the PSNR value is 284 [dB]; that is the decompressed image is the same as the original based on a visual criteria

กิตติกรรมประกาศ

โครงงานวิศวกรรมคอมพิวเตอร์ สำเร็จได้ด้วยดีก็เนื่องด้วยความอนุเคราะห์จากอาจารย์ที่ปรึกษาโครงงานคือ ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มแม่น ที่คอยให้ความช่วยเหลือในทุก ๆ ด้าน ในโอกาสนี้ทางคณะผู้จัดทำโครงงานจึงขอขอบคุณทุก ๆ ท่านที่มีส่วนช่วยทำให้โครงงานนี้ประสบความสำเร็จได้ด้วยดี

นายจักรพงษ์ เสือมี

นายสิทธิชาติ เทิดสิทธิกุล

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ก
บทคัดย่อภาษาอังกฤษ ..... ข
กิตติกรรมประกาศ ..... ค
สารบัญ ..... ง
สารบัญตาราง ..... ช
สารบัญรูป ..... ซ

บทที่ 1 บทนำ
1.1 หลักการและเหตุผล 1
1.2 วัตถุประสงค์ 1
1.3 ขอบข่ายของโครงงาน 1
1.4 ขั้นตอนการดำเนินงาน 2
1.5 ผลที่คาดว่าจะได้รับ 2
1.6 งบประมาณ 2

บทที่ 2 หลักการและทฤษฎี
2.1 การบีบอัดข้อมูล 3
2.2 ทฤษฎีเวฟเล็ต 3
2.2.1 พื้นฐานของการแปลงเวฟเล็ต 3
2.2.2 การวิเคราะห์สัญญาณแบบหลายระดับความละเอียด 6
2.2.3 การแปลงเวฟเล็ต 10
2.2.4 การแปลงกลับเวฟเล็ต 12
2.2.5 การประยุกต์ใช้กับข้อมูลภาพ 13
2.2.6 การประยุกต์ใช้เวฟเล็ตในการลดขนาดข้อมูล 14
2.3 การหาค่าความผิดพลาด (SNR และPSNR)
และการหาค่าอัตราส่วนการบีบอัดข้อมูล 15

สารบัญ (ต่อ)

หน้า

2.4 ทฤษฎีของฮัฟฟ์แมน 15

2.4.1 การเข้ารหัสแบบฮัฟฟ์แมน 16

2.4.2 การถอดรหัสของแบบฮัฟฟ์แมน 20

2.5 หลักการหาขอบเขตที่สนใจ 21

บทที่ 3 วิธีการดำเนินโครงงาน

3.1 วิธีการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต 22

3.2 กำหนดวิธีการหาขอบเขตที่สนใจ 23

3.3 การบีบอัดข้อมูลภาพ 24

3.3.1 การเข้ารหัส 25

3.3.2 การถอดรหัส 26

3.4 การออกแบบโปรแกรม 27

3.4.1 ส่วนของโปรแกรม 27

3.4.2 ส่วนของฟังก์ชั่นที่เรียกใช้ 30

บทที่ 4 ผลการทดลองและผลการวิเคราะห์

4.1 ผลการทดลอง 36

4.4.1 การทดลองการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียว 36

4.4.2 การทดลองการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมน 39

4.4.3 โปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต 43

4.2 ผลการวิเคราะห์ 46

บทที่ 5 บทสรุป

5.1 สรุปผลการทดลอง 47

5.2 ปัญหาในการทดลองและแนวทางแก้ไข 48

5.3 แนวทางในการพัฒนาโปรแกรมต่อไป 48

สารบัญ (ต่อ)

หน้า

เอกสารอ้างอิง 49

ภาคผนวก 50

ประวัติผู้เขียนโครงงาน 76

สารบัญตาราง

ตารางที่ หน้า

2.1 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 1 17

2.2 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 2 17

2.3 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 3 17

2.4 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 4 18

2.5 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 5 18

2.6 ขั้นตอนการเข้ารหัสแบบฮัฟฟ์แมนขั้นตอนที่ 6 19

4.1 ผลที่ได้จากการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียว 37

4.2 ผลที่ได้จากการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมน 40

5.1 ตารางเปรียบเทียบค่า SNR และ PSNR ของ Huffman และ Wavelet & Huffman 47

สารบัญรูป

รูปที่		หน้า
2.1	ลักษณะของเวฟเล็ตแม่ชนิด Daubechies20	4
2.2	แสดงลักษณะของเวฟเล็ตแม่ที่ถูกสเกลและเลื่อนตำแหน่งไปที่ค่า a,b ต่างๆกัน	5
2.3	ลักษณะของการแตกกระจายสัญญาณและการรวมกลับสัญญาณของเวฟเล็ต	5
2.4	ลักษณะการกระจาย (Decomposition) ของสเปซของเวกเตอร์	9
2.5	ลักษณะการกระจาย (Decomposition) สัญญาณไปยังระดับความละเอียดต่างๆ	10
2.6	Two-channel filter banks	11
2.7	Reconstruction Two-channel filter banks	11
2.8	ลักษณะของ DWT โดยใช้ Dyadic tree structure (Octave filter banks)	12
2.9	ลักษณะของ Reconstruction Two-channel filter banks	12
2.10	ลักษณะของ IDWT โดยการใช้ dyadic tree structure	12
2.11	แสดงการแปลงเวฟเล็ตของข้อมูลภาพออกเป็น 4 ส่วน คือ AGG1 aGH1	14
2.12	รูปแสดงขั้นตอนการลดขนาดข้อมูล	14
2.13	แสดงการกำหนดจุดพิกเซลในภาพ	21
3.1	แผงผังวิธีการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต	22
3.2	ภาพที่แสดงการกำหนดจุดพิกเซล 2 จุด	23
3.3	ภาพแสดงการกำหนดขอบเขตที่สนใจ	24
3.4	แสดงภาพที่ได้จากการแปลงเวฟเล็ตของเส้นภาพตามแนวนอน	25
3.5	แสดงภาพที่ได้จากการแปลงเวฟเล็ตของเส้นภาพตามแนวนอนและแนวตั้ง	26
3.6	Flowchart โปรแกรมการบีบอัดข้อมูลภาพ โดยวิธีการแปลงเวฟเล็ตและการแบ่งส่วนหลัง	27
3.7	Flowchart โปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ตและฮัฟฟ์แมน	28
3.8	Flowchart โปรแกรมการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียว	29
3.9	Flowchart ฟังก์ชั่นการกำหนดขอบเขตที่สนใจ	30
3.10	Flowchart ฟังก์ชั่นการหาค่าความถี่ของข้อมูล	31
3.11	Flowchart ฟังก์ชั่นสร้างตารางรหัสฮัฟฟ์แมน	32
3.12	Flowchart ฟังก์ชั่นการเข้ารหัสแบบฮัฟฟ์แมน	33
3.13	Flowchart ฟังก์ชั่นการถอดรหัสแบบฮัฟฟ์แมน	34

สารบัญรูป (ต่อ)

รูปที่		หน้า
3.14	Flowchart ฟังก์ชั่นการถอดรหัสแบบฮัฟฟ์แมน (ต่อ)	35
4.1	แสดงภาพต้นฉบับ nemo6464.bmp	36
4.2	แสดงภาพต้นฉบับ eye6464.bmp	36
4.3	แสดงภาพต้นฉบับ lena6464.bmp	36
4.4	แสดงภาพ nemo (ขนาด 64x64 พิกเซล) ของผลที่ได้จากการรันโปรแกรม hufftest.m	37
4.5	แสดงภาพ eye (ขนาด 64x64 พิกเซล) ของผลที่ได้จากการรันโปรแกรม hufftest.m	38
4.6	แสดงภาพ lena (ขนาด 64x64 พิกเซล) ของผลที่ได้จากการรันโปรแกรม hufftest.m	39
4.7	แสดงภาพ nemo (ขนาด 64x64 พิกเซล) ของผลที่ได้จากการรันโปรแกรม test3.m	40
4.8	แสดงภาพ eye (ขนาด 64x64 พิกเซล) ของผลที่ได้จากการรันโปรแกรม test3.m	41
4.9	แสดงภาพ lena (ขนาด 64x64 พิกเซล) ของผลที่ได้จากการรันโปรแกรม test3.m	42
4.10	แสดง ภาพที่รอการกำหนดขอบเขตที่สนใจ	43
4.11	แสดงภาพการกำหนดขอบเขตที่สนใจ	43
4.12	แสดงภาพที่ถูกกำหนดขอบเขตที่สนใจและภาพที่ถูกบีบอัดข้อมูลแล้ว	44
4.13	แสดงผลของขนาดและ SNR , PSNR	45

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ในปัจจุบันการบีบอัดข้อมูลภาพ จะมีอัตราการบีบอัดข้อมูลสูง (High-compression rate) หากคุณภาพที่สูญเสียไปเป็นที่ยอมรับได้ ไม่ให้มีความบกพร่องต่างๆในขอบเขตที่สำคัญในการวิเคราะห์ ความต้องการคุณภาพที่เคร่งครัด ซึ่งจุดนี้เป็นข้อจำกัดของการใช้การสื่อสาร การบีบข้อมูลเพื่อลดขนาดข้อมูลให้มีขนาดเล็กลงเพื่อประหยัดเนื้อที่ นอกจากนี้ขนาดข้อมูลที่เล็กลงยังเป็นการช่วยประหยัดเวลาในการสื่อสารข้อมูลได้วิธีหนึ่ง และหากมีการกำหนดขอบเขตที่สนใจ (Region of interest : ROI) ก่อนที่จะทำการบีบอัดข้อมูลก็สามารถลดขนาดได้อีกและจะได้ในสิ่งที่ต้องการอย่างเดียวโดยที่ขอบเขตที่สนใจถูกถือว่าเหมือนกันและถูกแบ่งเป็นส่วนๆก่อนทำการถอดรหัส (Encode) โดยการถอดรหัสจะใช้ทฤษฎีเวฟเล็ต (Wavelet tranform) หลังจากที่เข้าทฤษฎีเวฟเล็ตแล้ว ก็จะนำผลที่ได้มาถอดรหัสโดยใช้การถอดรหัสแบบฮัฟฟ์แมน (Encode huffman) แล้วจะนำมาเก็บในไฟล์ที่อยู่ในรูปของเลขไบนารี่ (Binary files) ไฟล์ที่ได้จะมีขนาดเล็กกว่าไฟล์ต้นฉบับ ทำให้มีอัตราส่วนการบีบอัดข้อมูล (Compression ratio) สูง ซึ่งการบีบข้อมูลภาพที่ดีจะต้องให้อัตราส่วนการบีบอัดข้อมูลสูงและคุณภาพของภาพต้องคมชัดและสมบูรณ์ ซึ่งเป็นหนึ่งในมาตรฐานการบีบอัดภาพที่ใช้กันอย่างแพร่ หลายในปัจจุบันคือ เทคนิคการบีบอัดภาพแบบเจเพ็ก (JPEG หรือ Joint Photographic Expert Group) ซึ่งมีความสามารถบีบอัดภาพแบบสูญเสียข้อมูลบางส่วนได้ในอัตราส่วนการบีบ 20:1 ถึง 30:1 หลังจากที่เก็บเป็นไฟล์ไบนารี่แล้วจะทำการเข้ารหัสแบบฮัฟฟ์แมน (Decode huffman) แล้วจึงนำผลที่เข้ารหัสแล้วมาเข้าทฤษฎีแปลงผกผันของเวฟเล็ต (Inverse Wavelet transform) แล้วจะได้ภาพที่มีคุณภาพใกล้เคียงกับภาพต้นฉบับ

ปัจจุบันมีการนำหลักการของการบีบอัดข้อมูลภาพ โดยวิธีการแปลงเวฟเล็ตมาพัฒนาขึ้นเป็นโปรแกรมในภาษาต่างๆ และยังสามารถนำมาใช้ประโยชน์ได้ในวงการแพทย์ และวงการอื่นๆ

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการแปลงรูปเวฟเล็ต
2. เพื่อศึกษาการเข้ารหัสด้วยวิธีฮัฟฟ์แมน
3. เพื่อศึกษาการบีบอัดข้อมูลภาพด้วยวิธีเวฟเล็ตและการเข้ารหัสแบบฮัฟฟ์แมน

1.3 ขอบข่ายของโครงงาน

1. ศึกษาวิธีการบีบอัดข้อมูลภาพโดยใช้วิธีการแปลงเวฟเล็ต
2. ศึกษาการเขียนโปรแกรมที่เกี่ยวข้องกับการบีบอัดข้อมูลภาพโดยใช้ MATLAB

1.4 ขั้นตอนการดำเนินงาน

	2546		2547									
	พย	ธค	มค	กพ	มีค	เมย	พค	มิย	กค	สค	กย	ตค
ศึกษาค้นคว้าในเรื่องการบีบอัดข้อมูล	←				→							
ศึกษาค้นคว้าการใช้โปรแกรม Matlab				←		→						
ออกแบบการเขียนโปรแกรม						←			→			
พัฒนาทดสอบและแก้ไข								←		→		
สรุปและจัดทำรายงาน											←	→

1.5 ผลที่คาดว่าจะได้รับ

1. ลดจำนวนบิตในการจัดเก็บของข้อมูลภาพโดยคุณภาพของข้อมูลภาพของขอบเขตที่สนใจยังคงเหมือนภาพต้นแบบ
2. นำไปประยุกต์ใช้ในการส่งข้อมูลภาพได้เร็วขึ้น

1.6 งบประมาณ

1. ค่าวัสดุอุปกรณ์	1,500	บาท
2. ค่าถ่ายเอกสาร	1,000	บาท
รวมทั้งสิ้น	**2,000**	**บาท (สองพันบาทถ้วน)**

(หมายเหตุ) ถัวเฉลี่ยทุกรายการ

บทที่ 2
หลักการและทฤษฎี

2.1 การบีบอัดข้อมูลภาพ (Image Compression)

หลักในการบีบอัดข้อมูลภาพคือพยายามลดหรือกำจัดส่วนของข้อมูลที่เกินความจำเป็นหรือซ้ำซ้อนกัน (Data redundancy) โดยยังคงข่าวสารไว้เหมือนเดิม ซึ่งจะทำให้ข้อมูลภาพมีขนาดลดลงจากเดิมและสามารถนำภาพกลับมาแสดงภายหลังโดยผ่านกระบวนการคลาย (Decompression)

ประเภทของการบีบอัดข้อมูลภาพแบ่งได้เป็น 2 แบบคือ

2.1.1 การบีบอัดข้อมูลแบบไม่มีการสูญเสีย (Lossless compression หรือ Bit-preserving หรือ Versible compression)

2.1.2 การบีบอัดข้อมูลแบบมีการส่วนเสียบางส่วน (Lossy Compression หรือ Irreversible compression)

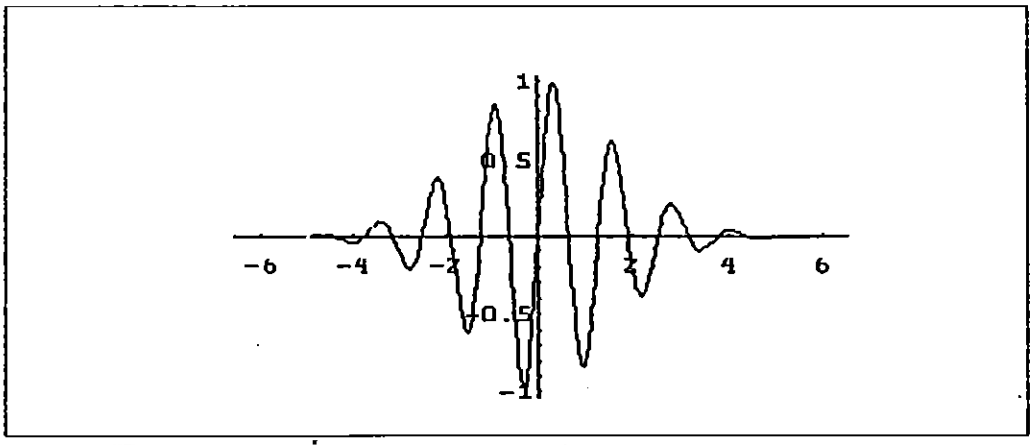
2.2 ทฤษฎีเวฟเล็ต (Wavelet transform)

การแปลงเวฟเล็ต เป็นคณิตศาสตร์ที่ใช้ในการวิเคราะห์และสังเคราะห์ลักษณะของสัญญาณซึ่งมีประโยชน์มากในงานทางด้านการประมวลผลสัญญาณ (Signal processing) ทฤษฎีเวฟเล็ตสามารถนำมาประยุกต์เพื่ออธิบายลักษณะของสิ่งต่างๆหรือระบบใดๆได้ เช่น อธิบายการแก้ปัญหาสมการดิฟเฟอเรลเชียลที่จำลองระบบใดระบบหนึ่ง การเต้นของหัวใจ การไหลเวียนของเลือดผ่านร่างกาย การหาขอบภาพ การลดขนาดข้อมูล (ภาพ สัญญาณ) และใช้ในการแก้ปัญหาวิศวกรรมทางการแพทย์

2.2.1 พื้นฐานของการแปลงเวฟเล็ต

เมื่อใช้การแปลงฟูเรียร์ในการวิเคราะห์สัญญาณเฉพาะบางช่วงเวลาและความถี่เท่านั้นจะเสียเวลาในการคำนวณมากเพราะต้องคำนวณใหม่ตลอดย่าน ดังนั้นจึงได้มีการพัฒนาการแปลงที่สามารถวิเคราะห์สัญญาณได้เฉพาะช่วงเวลาและช่วงความถี่ที่สนใจเท่านั้น ทำให้สามารถพิจารณาผลกระทบของการเปลี่ยนแปลงของสัญญาณเฉพาะช่วงได้อย่างสะดวกและรวดเร็ว การแปลงที่พัฒนาขึ้นนี้เรียกว่า "การแปลงเวฟเล็ต"

การแปลงเวฟเล็ตจะใช้ในการอธิบายโครงสร้างของระบบสัญญาณที่ประกอบด้วยกลุ่มของสัญญาณเฉพาะมารวมกันเป็นสัญญาณหรือระบบนั้นๆ โดยสัญญาณเฉพาะนี้จะเป็นคลื่นเล็กๆ ที่เรียกว่าเวฟเล็ต ลักษณะของเวฟเล็ตจะเป็นคลื่นที่มีการเปลี่ยนแปลงอย่างต่อเนื่อง (Oscillatary) และขนาดของคลื่นจะลดลงสู่ศูนย์อย่างรวดเร็วทั้งสองด้าน ดังรูปที่ 2.1 ซึ่งเป็นเวฟเล็ตชนิดหนึ่งที่เรียกว่า เวฟเล็ตแบบ Daubechies20

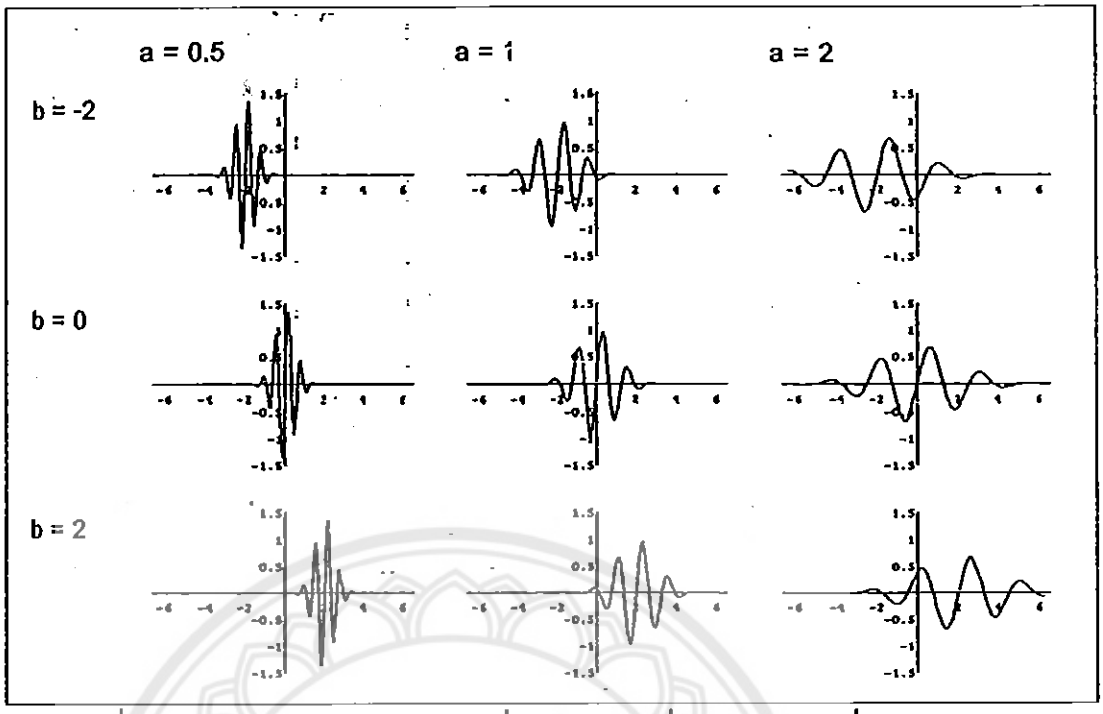


รูปที่ 2.1 ลักษณะของเวฟเล็ตแม่ชนิด Daubechies20

การนำเวฟเล็ตหลายๆอันมารวมกัน เป็นกลุ่มเพื่อใช้ในการอธิบาย**โครงสร้างของสัญญาณ**ใดๆ โดยที่คลื่นเวฟเล็ตแต่ละตัวจะมีโครงสร้างมาจากฟังก์ชั่นเดียวกันซึ่ง**ฟังก์ชั่นนี้จะเป็นเวฟเล็ตต้นกำเนิดที่เรียกว่า** เวฟเล็ตแม่ (Mother Wavelets) คลื่นเวฟเล็ตแต่ละอันจะ**อยู่ภายในเซตของเวฟเล็ตนี้**โดยแต่ละคลื่นจะเกิดจากการสเกล (Scaling:"a") และการเลื่อน**ตำแหน่ง** (Translation:"b") **ดังนั้น**ถ้าให้ W(t) เป็นฟังก์ชั่นเวฟเล็ตแม่ สามารถเขียนเป็นสมการทั่วไป**ของเวฟเล็ตที่ตำแหน่ง** "a","b" ใดๆ ที่สัมพันธ์กันได้ดังนี้

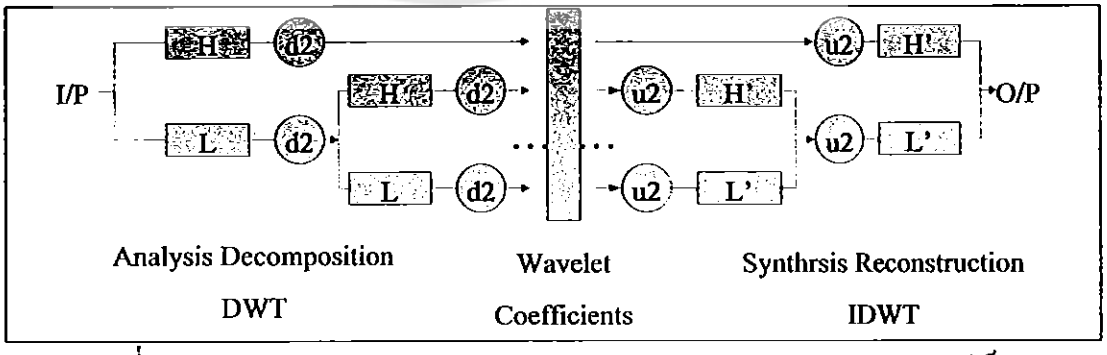
$$W_{b,a}(t) = \frac{1}{\sqrt{a}} W(\frac{t-b}{a}) \qquad \textbf{2.1}$$

W(t) จะเป็นฟังก์ชั่นเวฟเล็ตแม่ที่ถูกเลื่อนตำแหน่งและถูกสเกลโดย **ทารามิเตอร์** "a" และ "b" ตามลำดับ โดยที่ช่วงเวลาและความถี่ในการแปลงจะสัมพันธ์กันและ**เพื่อให้เวฟเล็ตที่ถูกสเกลไปแล้วมีพลังงานเท่ากับเวฟเล็ตแม่จึงต้องทำการนอมอร์ไลซ์ด้วย** $1/\sqrt{a}$ เสมอ



รูปที่ 2.2 แสดงลักษณะของเวฟเล็ตแม่ที่ถูกสเกลและเลื่อนตำแหน่งไปที่ค่า a,b ต่างๆกัน

ทฤษฎีเวฟเล็ตจะใช้ในการอธิบายสิ่งๆสิ่งหนึ่งเสมือนการแตกสิ่งนั้นออกเป็นส่วนประกอบเล็กๆ สัมพันธ์กันโดยที่ชิ้นส่วนเหล่านี้จะอยู่ในรูปเวฟเล็ตที่ถูกสเกลและเลื่อนตำแหน่ง ดังนั้นจึงเปรียบเสมือนว่าสัญญาณใดๆ สามารถสร้างขึ้นมาได้โดยมีฟังก์ชั่นพื้นฐาน (Basis function) การแตกกระจายเวฟเล็ต(Wavelet Decomposition) ก็คือการทำการแปลงเวฟเล็ต (Wavelet Transform : WT)นั่นเองในทำนองเดียวกันการรวมกลับเวฟเล็ต(Wavelet Reconstruction) จะเป็นการแปลงกลับของเวฟเล็ต (Inverse Wavelet Transform : IWT) ซึ่งเป็นการนำส่วนประกอบย่อยๆ เหล่านี้มารวมกันเพื่อประกอบเป็นสัญญาณเดิมดังในรูปที่ 2.3



รูปที่2.3 ลักษณะของการแตกกระจายสัญญาณและการรวมกลับสัญญาณของเวฟเล็ต

2.2.2 การวิเคราะห์สัญญาณแบบหลายระดับความละเอียด

การวิเคราะห์สัญญาณแบบหลายระดับความละเอียด (Multiresolution Analysis : MRA) จะเป็นการวิเคราะห์สัญญาณที่สามารถเลือกระดับความละเอียดได้โดยการนำสัญญาณเล็กๆ ที่ระดับความละเอียด a ซึ่งมี b หลายๆตำแหน่งมารวมกันเกิดเป็นสัญญาณที่ระดับความละเอียดที่เลือกไว้ และเมื่อสัญญาณที่ทุกระดับความละเอียดมารวมกันจะเกิดเป็นสัญญาณอินพุทจริง

ก่อนที่จะศึกษาถึงการวิเคราะห์สัญญาณแบบMRAจะขออธิบายถึงทฤษฎีพื้นฐานของสเปซเวกเตอร์ (Vector space) ก่อนเนื่องจากมีความเกี่ยวข้องกับการวิเคราะห์แบบ MRA

สเปซของเวกเตอร์ในการวิเคราะห์สัญญาณ คือ สเปซหรือปริภูมิของสัญญาณใดๆ ที่เกิดจากการรวมกันของสัญญาณพื้นฐานย่อยๆ ที่เรียกว่า Basis function ถ้ากำหนดให้ a,b เป็นจำนวนจริงใดๆ และ $\hat{i}, \hat{j}$ เป็นเวกเตอร์หนึ่งหน่วยในสเปซเวกเตอร์ V เสมอ ดังนั้นถ้าเวกเตอร์ใดประกอบขึ้นจากหลายเวกเตอร์หนึ่งหน่วยจะได้ว่าเวกเตอร์นั้นยังคงอยู่ในสเปซเวกเตอร์นี้ ดังนั้น $a\hat{i} + b\hat{j} + c\hat{k} + \ldots \in V$ โดยที่เวกเตอร์หนึ่งหน่วย $\hat{i}, \hat{j}$ จะเป็นลักษณะเชิงตั้งฉาก (Orthogonal) ซึ่งกันและกัน ถ้าพิจารณาในลักษณะของสัญญาณอาจมองได้ว่า $\hat{i}, \hat{j}$ คือ Basis function ที่เป็นสัญญาณเล็กๆ ที่นำมาประกอบกันเป็นสัญญาณใดๆ

สมมุติให้ V^j เป็นสเปซเวกเตอร์ที่มี j แสดงถึงระดับความละเอียดและจำนวนของ Basis function ที่ประกอบขึ้นเป็นฟังก์ชั่นนั้น ถ้า j มีค่าสูงขึ้นก็แสดงว่าที่ระดับความละเอียดสูงขึ้นจะมีจำนวน Basis function มากขึ้นทำให้สัญญาณที่เกิดจากการประกอบกันจาก Basis function มีความละเอียดมากขึ้นด้วย ดังนั้นอาจกล่าวได้ว่า j เป็นค่าแสดงถึงระดับความละเอียดของสัญญาณนั่นเอง จากข้อกำหนดเหล่านี้สามารถสรุปเป็นลักษณะสมบัติของการวิเคราะห์สัญญาณที่ระดับความละเอียดต่างๆ ได้ดังนี้

1) $V^{-\infty} \ldots \subset V^{-1} \subset V^{1} \ldots \subset V^{\infty}$

2) Close $_{L^2}(\bigcup_{j \in ZZ} V^j) = L^2(IR)$:IR = เซตของจำนวนจริง

3) $(\bigcap_{j \in Z} V^j) = \{0\}$

4) $V^j + W^j = V^{j+1}$: $j \in ZZ$: ZZ = เซตของจำนวนเต็ม

5) $f(x) \in V^j \longleftrightarrow f(2x) \in V^{j+1}$: $j \in ZZ$

จากการที่ Basis function ประกอบกันเป็นสัญญาณการประมาณที่ระดับความละเอียด j ภายในสเปซ V^j จะเรียก Basis function เหล่านี้ว่า ฟังก์ชั่นสเกลลิ่ง (Scaling function : $\phi(t)$) สัญญาณเหล่านี้จะเกิดที่ตำแหน่งเวลาต่างๆกันของสเปซและมีความถี่เท่ากันภายในสเปซเดียวกัน

ฟังก์ชั่นสเกลลิ่งที่ระดับสเปซสูง (ระดับความละเอียดสูง) จะมีความถี่สูงและที่ระดับต่ำกว่าจะมีความต่ำกว่า ดังนั้นความสัมพันธ์กันระหว่างฟังก์ชั่นสเกลลิ่งของแต่ละสเปซจะเป็นดังนี้

$$f(x) \in V^j \quad \longleftrightarrow \quad f(2x) \in V^{j+1} \qquad : j \in ZZ : ZZ = \text{เซตของจำนวนเต็ม} \qquad 2.2$$

จากสมการที่ 2.2 ทำให้สามารถเขียนเป็นความสัมพันธ์ระหว่างฟังก์ชั่นสเกลลิ่งภายในสเปซใดๆ ได้ดังนี้

$$\phi_{j,k}(t) = 2^{j/2}\phi(2^j t - k) \qquad : j, k \in ZZ : ZZ = \text{เซตของจำนวนเต็ม} \qquad 2.3$$

จากสมการที่ 2.3 จะพบว่าระดับความละเอียดต่ำลงมาหนึ่งระดับ Basis function จะมีความถี่ลดลงมาครั้งละสองเท่า อาศัยลักษณะคุณสมบัติ MRA จะทำให้สามารถทำการประมาณสัญญาณ $f(t) \in L^2(IR)$ ไปอยู่ในสเปซที่ระดับความละเอียด j ใดๆ ก็ได้ดังนี้

$$f_j(t) = \sum_k c_k^j \phi_{j,k}(t) \qquad 2.4$$

โดยที่ C_k^j เป็นสัมประสิทธิ์หรือน้ำหนักที่คูณกับฟังก์ชั่นสเกลลิ่งที่ตำแหน่ง k ใดๆ แล้วประกอบขึ้นเป็น f(t) ที่ระดับความละเอียด j นั้นๆ

จากการวิเคราะห์สัญญาณที่ระดับความละเอียดต่ำลงมาจะทำให้พลังงานหรือสัญญาณบางส่วนหายไปอยู่ในสเปซอีกอันหนึ่งซึ่งจะเรียกว่า สเปซของเวกเตอร์เวฟเล็ต (Wavelet vector space : W^j) สเปซชนิดนี้จะคล้ายกับสเปซของเวกเตอร์ ดังนั้นสัญญาณภายใน W^j จะประกอบด้วย Basis function เช่นเดียวกันจะเรียกว่า ฟังก์ชั่นเวฟเล็ต (Wavelet function : $\psi(t)$) ดังนั้นสามารถเขียนสมการฟังก์ชั่นเวฟเล็ตที่ระดับความละเอียดใดๆ ได้ดังนี้

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \qquad : j, k \in ZZ : ZZ = \text{เซตของจำนวนเต็ม} \qquad 2.5$$

จากสมการที่ 2.5 ซึ่งมีลักษณะเป็นสัญญาณที่มีการเลื่อนตำแหน่งและเปลี่ยนความถี่อย่างต่อเนื่องจะเปลี่ยนเป็นฟังก์ชั่นเวฟเล็ต ที่มีการเลื่อนตำแหน่งและเปลี่ยนความถี่แบบเต็มหน่วยโดยที่ $a = 2^{-j}, b = 2^{-j}k$

ถ้ากำหนดให้ $g_j(t)$ เป็นสัญญาณที่เกิดจาก Basis function และ $\psi_{j,k}(t)$ ภายในสเปซเดียวกันมารวมกันเป็นสัญญาณใดๆ จะได้ว่า

$$g_j(t) = \sum_k d_k^j \psi_{j,k}(t) \qquad 2.6$$

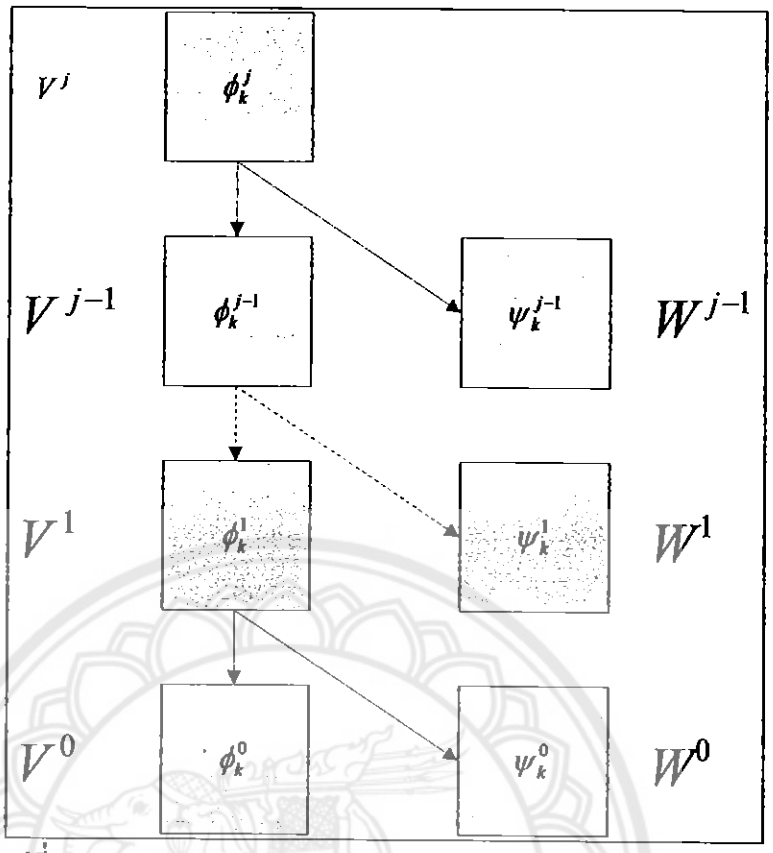
โดยที่ d_k^j เป็นสัมประสิทธิ์หรือค่าน้ำหนักที่คูณเกับฟังก์ชั่นเวฟเล็ตที่ตำแหน่งนั้นๆ เพื่อเกิดเป็นสัญญาณ $g_j(t)$ ดังนั้นจากความสัมพันธ์ $V^j + W^j = V^{j+1}$ และจากสมการที่ 2.4 และ 2.6 จะได้ว่า

$$f_{j+1} = f_j + g_j \qquad 2.7$$

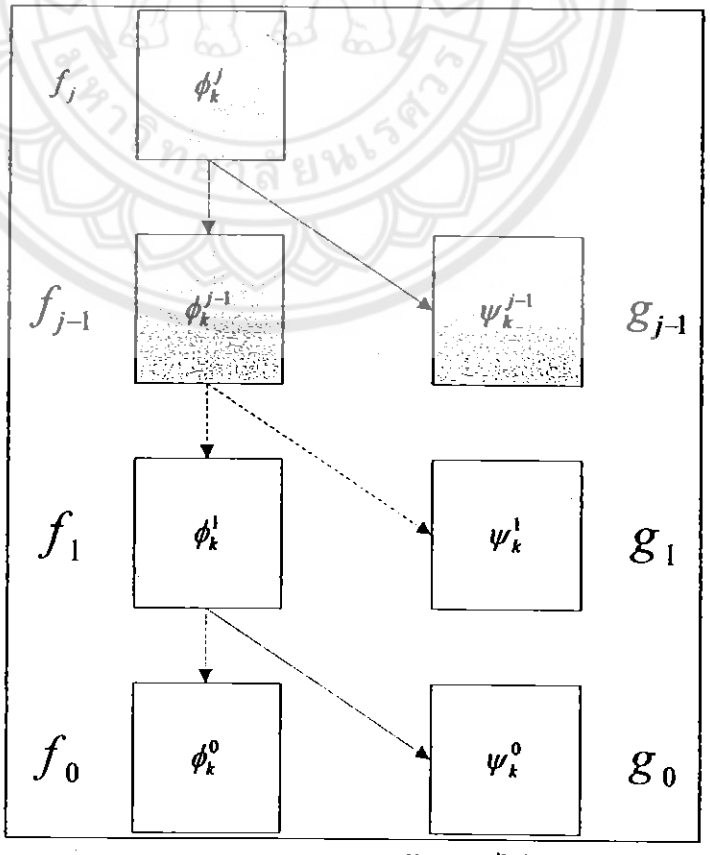
สมมุติให้ $f(t) \in V^{j+1}$ จะสามารถแตกกระจายให้ $f(t)$ ให้มีความละเอียดน้อยลงได้จากสมการ $V^j + W^j = V^{j+1}$ ซึ่งในขณะเดียวกัน V^j สามารถแตกต่อไปได้เรื่อยจนกระทั่ง $j = 0$ ดังนั้นจะได้เป็นความสัมพันธ์ว่า

$$V^{j+1} = V^0 + W^0 + W^1 + \ldots + W^j \qquad 2.8$$

ในทำนองเดียวกัน f_{j+1} ก็สามารถแตกกระจายเป็น f_j และ g_j ซึ่งสามารถแสดงเป็นภาพการแตกกระจายสเปซและสัญญาณได้ดังรูปที่ 2.4 และ 2.5



รูปที่ 2.4 ลักษณะการกระจาย (Decomposition) ของสเปซของเวกเตอร์



รูปที่ 2.5 ลักษณะการกระจาย (Decomposition) สัญญาณไปยังระดับความละเอียดต่างๆ

f และ g ที่ลดระดับความละเอียดลงมาจะมีความถี่ของ Basis function ลดลงครั้งละสองเท่าเสมอและเราสามารถแจกแจงสัญญาณ f(t) ที่อยู่ในรูปแบบของฟังก์ชั่นสเกลลิ่งและฟังก์ชั่นเวฟเล็ตได้ดังนี้

$$f(t) = f_j + g_j + g_{j+1} + g_{j+2} + \ldots + g_\infty \qquad 2.9$$

$$f(t) = \sum_m c_j(m)\phi_{j,m}(t) + \sum_{i=j}^{\infty}\sum_m d_i(m)\psi_{i,m}(t) \qquad 2.10$$

ดังนั้นสัมประสิทธิ์เวฟเล็ตสามารถหาได้จากสมการการโปรเจคชั่น ดังสมการที่ 2.11 และ 2.12

$$c_j(m) = \left\langle f(t), \phi_{j,m}(t) \right\rangle \qquad 2.11$$

$$d_j(m) = \left\langle f(t), \psi_{j,m}(t) \right\rangle \qquad 2.12$$

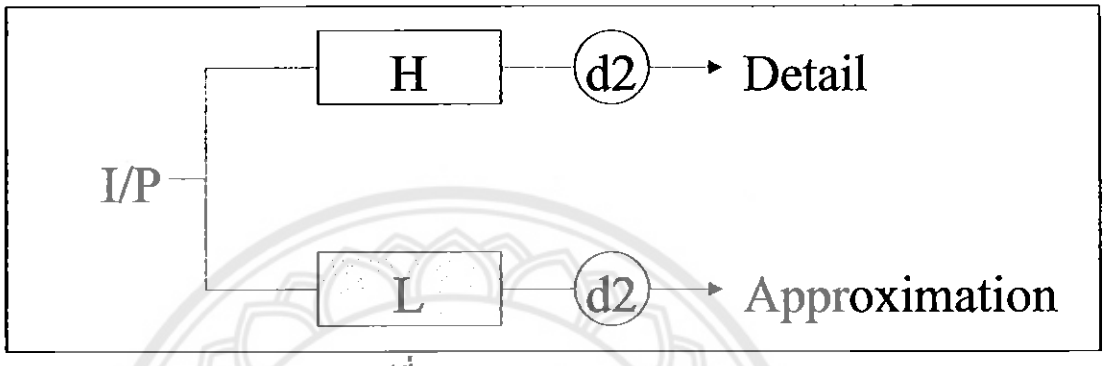
การแตกกระจายสัญญาณ f(t) ในสเปซ V^0 ไปจนถึงระดับความละเอียดที่ j ดังนั้นสัมประสิทธิ์ $c_0(m)$ จะถูกแตกเป็นสัมประสิทธิ์ในเซตของ $c_j(m)$ ในสเปซ V^j และกลุ่มของเซต $d_{-1}(m)$, $d_{-2}(m)$,..., $d_j(m)$ ซึ่งอยู่ในเวฟเล็ตสเปซที่ระดับความละเอียดต่างๆกัน จากขบวนการต่างๆที่กล่าวมาจะเป็นลักษณะของวิธีการแปลงเวฟเล็ตแบบเต็มหน่วย (Discreate Wavelet Transform : DWT)

2.2.3 การแปลงเวฟเล็ต

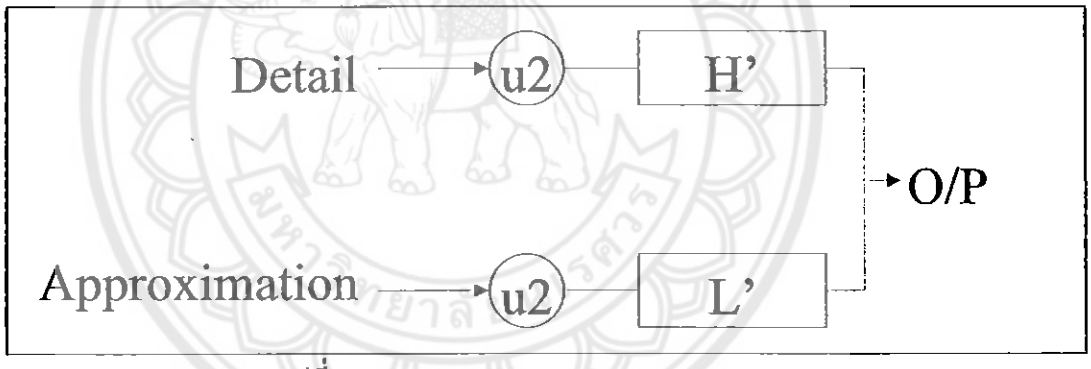
ขบวนการแปลงเวฟเล็ตจะมีลักษณะคล้ายกับขบวนการออคเทฟฟิลเตอร์แบงค์ (Octave filter banks) เนื่องจากการพิจารณาสัญญาณที่สามารถเปลี่ยนแปลงความถี่ในอัตราครั้งละสองเท่า ซึ่งเปรียบเสมือนกับการนำสัญญาณอินพุทผ่านวงจรกรองความถี่ที่มีแบนวิธที่มีอัตราการลดลงสองเท่าเหมือนกับแบนวิธของฟังก์ชั่นหน้าต่าง (Window function) ในขณะนั้นนั่นเอง ดังนั้นจะสามารถนำเอาหลักการของฟิลเตอร์แบงค์ (Filter banks) มาใช้ในการแปลงเวฟเล็ตในทางปฏิบัติได้ ก่อนที่จะอธิบายการสร้างการแปลงเวฟเล็ตในลักษณะฟิลเตอร์แบงค์ จะขออธิบายหลักการพื้นฐานของฟิลเตอร์แบงค์ก่อน

ฟิลเตอร์แบงค์แบบสองช่องสัญญาณ (Two-channel filter banks) เป็นการแยกสัญญาณอินพุทออกเป็นสองส่วน โดยแบ่งเป็นส่วนของความถี่ต่ำและส่วนของความถี่สูง ดังนั้นฟิลเตอร์แบงค์แบบสองช่องสัญญาณ จึงประกอบด้วยส่วนที่เป็น Low pass filter :L และ Complementary

highpass filter : H ดังรูปที่ 2.6 ซึ่งมีลักษณะของการวิเคราะห์ฟิลเตอร์แบงค์แบบสองช่องสัญญาณ และเป็นโครงสร้างที่กลับกัน กับการสังเคราะห์การสร้างกลับฟิลเตอร์แบงค์แบบสองช่องสัญญาณ (Reconstruction Two-channel filter banks) ความถี่ต่ำและความถี่สูง คือ L และ H กับตัวกรองการสร้างกลับความถี่ต่ำและความถี่สูงคือ L' , H' มีความสัมพันธ์กันในลักษณะที่เรียกว่า Quadrature mirror filters

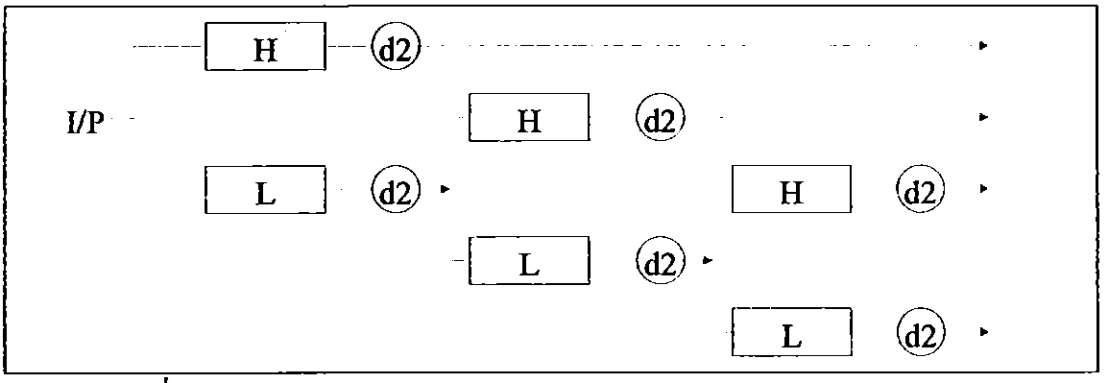


รูปที่ 2.6 Two-channel filter banks



รูปที่ 2.7 Reconstruction Two-channel filter banks

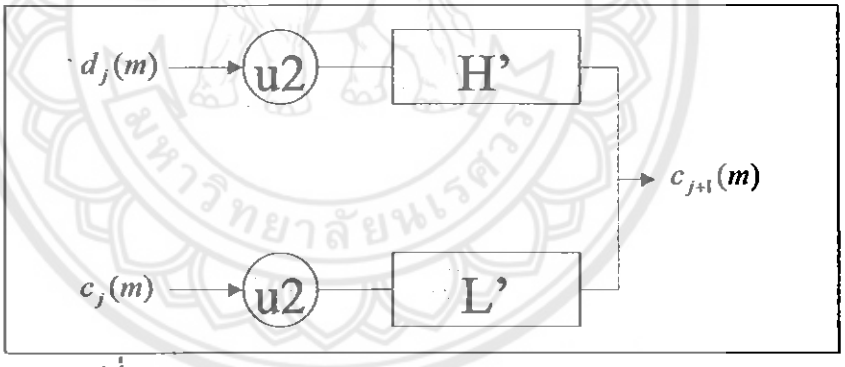
ลักษณะการวิเคราะห์ออคเทฟฟิลเตอร์แบงค์ จะเป็นโครงสร้างแบบต้นไม้ (Tree Structure) ซึ่งเป็นการสร้างเอาฟิลเตอร์แบงค์แบบสองช่องสัญญาณ มาเรียงต่อกันโดยใช้สัญญาณเอาท์พุทในส่วนที่เป็นความถี่ต่ำมาทำการแยกความถี่ออกอีกครั้งหนึ่ง ในกรณีที่ทำการแปลงเวฟเล็ตซ้ำในแนวของ Lowpass จะเป็นลักษณะของ Dyadic tree structure ดังรูปที่ 2.8 ซึ่งโครงสร้างในรูปนี้จะเป็นการแปลงแบบเต็มหน่วย (Discrete Wavelets Transform : DWT)



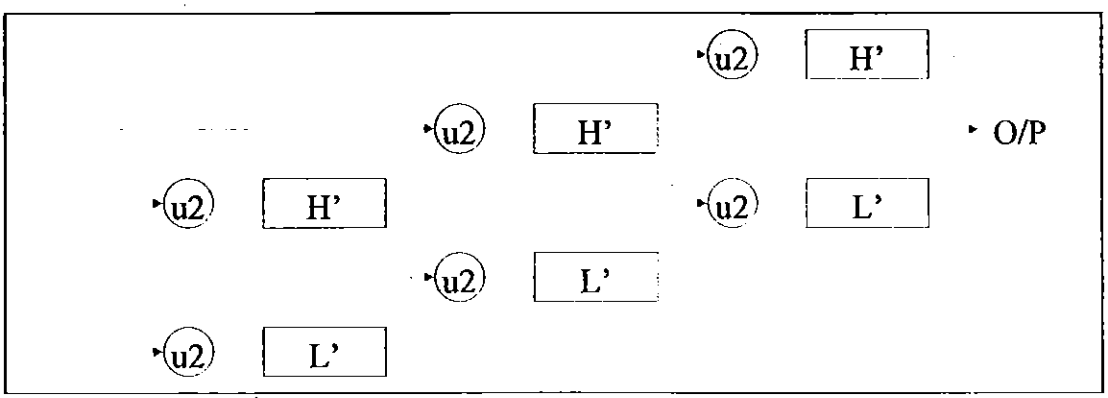
รูปที่ 2.8 ลักษณะของ DWT โดยใช้ Dyadic tree structure (Octave filter banks)

2.2.4 การแปลงกลับเวฟเล็ต

การแปลงแบบเต็มหน่วย (DWT) เป็นการแตกกระจาย (Decomposition) สัญญาณหรือการโปรเจคสัญญาณลงไปในสเปซของ V^j และ W^j กลับไปเป็นสเปซ V^{j+1} ก็จะเป็นการเพิ่มรายละเอียดของสัญญาณ f(t) จากระดับ j ไปเป็นระดับ j+1 วิธีการดังกล่าวนี้จะเหมือนกับขบวนการฟิลเตอร์แบงค์แบบสองช่องสัญญาณ ซึ่งเป็นการหาค่าของ $c_{j+1}(n)$ จากค่า $c_j(m)$ และ $d_j(m)$ ซึ่งสามารถอธิบายได้ดังรูปที่ 2.9



รูปที่ 2.9 ลักษณะของ Reconstruction Two-channel filter banks



รูปที่ 2.10 ลักษณะของ IDWT โดยการใช้ dyadic tree structure

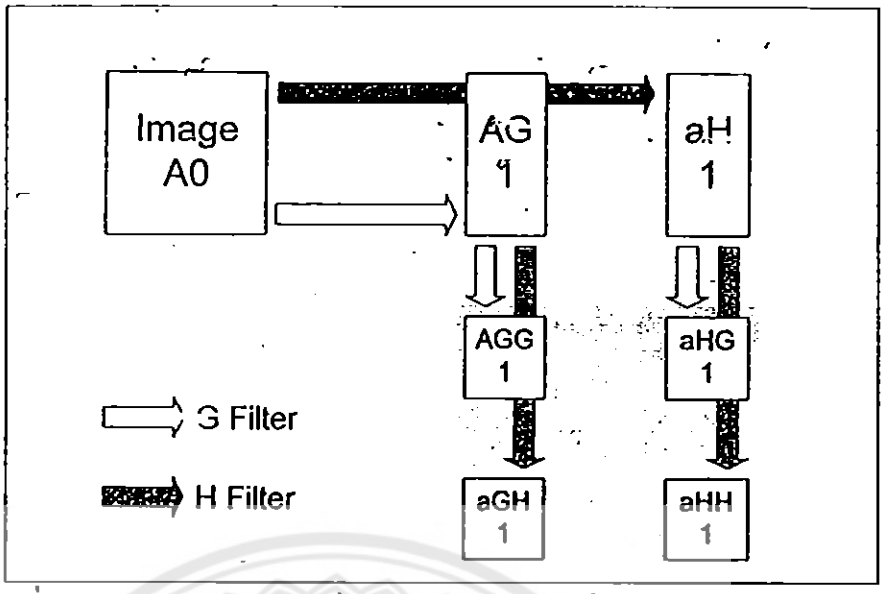
ลักษณะของ Tree structure filter banks เพื่อทำการรวบรวมสัมประสิทธิ์ $c_j(m)$ และ $d_j(m)$ กลับมาเป็น $c_{j+1}(n)$ อีกครั้ง ขบวนการนี้เรียกว่า การแปลงกลับเวฟเล็ต (Inverse Wavelet Transform : IWT) จากกระบวนการแปลงเวฟเล็ตและการแปลงกลับเวฟเล็ตจะสังเกตได้ว่าสัญญาณเอาท์พุทที่ได้จากการแปลงกลับจะมีค่าประมาณเท่ากับสัญญาณอินพุทของการแปลงเวฟเล็ตโดยที่รูปแบบของฟังก์ชั่นถ่ายโอน (Transfer function) จะมีในลักษณะการสร้างกลับอย่างสมบูรณ์ของฟิลเตอร์แบงค์ (Perfectly reconstructing filter banks) จากที่กล่าวมานี้จะใช้เฉพาะในกรณีของ Orthonormal wavelets หรือ Orhtonormal filter banks เท่านั้น

2.2.5 การประยุกต์ใช้กับข้อมูลภาพ

จากคุณสมบัติของเวฟเล็ต ที่เหมาะกับการนำมาประยุกต์ใช้กับการลดขนาดข้อมูลภาพ ในหัวข้อนี้จึงกล่าวถึงวิธีนำการแปลงเวฟเล็ตมาใช้งาน ขบวนการแปลงเวฟเล็ตที่นำมาใช้งานนี้จะเป็นวิธีการของ Multi resolution wavelet transform

จากข้อมูลภาพที่เป็นสัญญาณในสองมิตินี้ ขบวนการแปลงเวฟเล็ตที่ใช้จะต้องเป็นสองมิติด้วย แต่เพื่อความง่ายในการใช้งานข้อมูลภาพจะถูกแปลงสัญญาณเป็นหนึ่งมิติแทน เพื่อให้สามารถใช้การแปลงเวฟเล็ตที่กล่าวมาได้ โดยการจัดเรียงของแต่ละจุดภาพใหม่ แยกตามแนวแกนตั้ง และแกนนอน ข้อมูลที่ได้จากเส้นภาพในแนวแกนนอนแต่ละเส้นจะเป็นสัญญาณหนึ่งมิติ ซึ่งจะถูกกระทำด้วยการแปลงเวฟเล็ตจนครบทุกเส้นภาพ ผลที่ได้ออกมาจะเป็นค่าสัมประสิทธิ์ของเวฟเล็ตของสัญญาณในแต่ละเส้นภาพที่ไม่เกี่ยวข้องกันแล้วข้อมูลนี้จะถูกนำไปผ่านขบวนการแปลงเวฟเล็ตอีกครั้งตามแนวแกนตั้ง

จากการแปลงเวฟเล็ตแบบ Multiresolution ที่มีอัตราส่วนในการ Scaling เท่ากับสอง ผลลัพธ์ ที่ได้ จะแบ่งออกเป็นสองส่วนคือ High pass filter และ Low pass filter ดังนั้น เมื่อข้อมูลภาพถูกนำมาทำการแปลง ค่าสัมประสิทธิ์ที่ได้จึงแบ่งออกเป็นสี่ส่วน แสดงดังรูปที่ 2.11

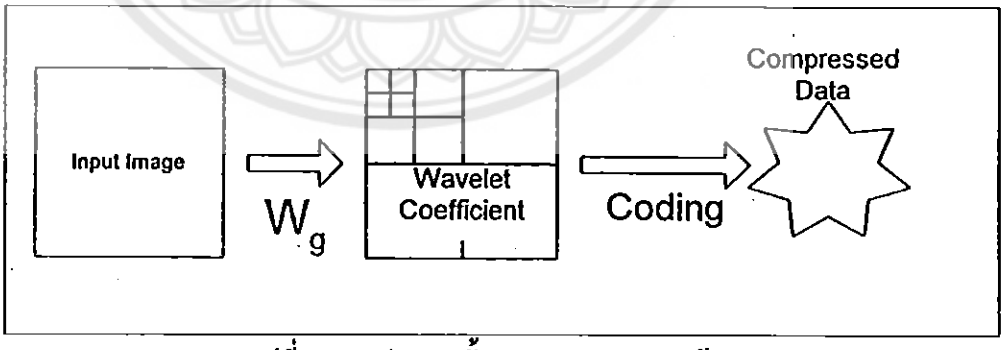


รูปที่ 2.11 แสดงการแปลงเวฟเล็ตของข้อมูลภาพออกเป็น 4 ส่วน คือ AGG1 aGH1 aHG1 aHH1 (ส่วนที่แรเงา) ซึ่งองค์ประกอบทั้งสี่นี้บังคงมีจำนวนจุดรวมกันทั้งหมดเท่ากับภาพต้นแบบ (AO)

จากนั้นส่วนของข้อมูลความถี่ต่ำที่เหลืออยู่ (AGG1) จะสามารถนำไปแตกกระจายต่อไปได้เช่นเดียวกันกับการแปลงเวฟเล็ตของสัญญาณหนึ่งมิติ

2.2.6 การประยุกต์ใช้เวฟเล็ตในการลดขนาดข้อมูล

ขบวนการลดขนาดข้อมูลจะกระทำกับค่าสัมประสิทธิ์เวฟเล็ตที่ได้ จากขบวนการวิเคราะห์ดังรูปที่ 2.12



รูปที่ 2.12 รูปแสดงขั้นตอนการลดขนาดข้อมูล

ค่าสัมประสิทธิ์ที่ได้ ซึ่งเป็นส่วนประกอบย่อยของสัญญาณจะถูกนำมาผ่านขบวนการเข้ารหัส ในขั้นตอนของการเข้ารหัสนี้ อาจมีข้อมูลบางส่วนที่หายไปหรือเกินออกมา จากนั้นในขั้นตอนของการวิเคราะห์จะสามารถสร้างสัญญาณเอาท์พุท ที่มีลักษณะใกล้เคียงกับสัญญาณอินพุท ได้ทั้งนี้ก็ขึ้นอยู่กับว่าในขั้นตอนของการลดขนาดข้อมูลมีความผิดพลาดเกิดขึ้นเพียงใด

2.3 การหาค่าความผิดพลาด (SNR และPSNR) และการหาค่าอัตราส่วนการบีบอัดข้อมูล

อัตราส่วนของการบีบอัดข้อมูล (Compression Ratio : CR) คือ การเปรียบเทียบค่าระหว่างขนาดของข้อมูลต้นฉบับ (n1) กับข้อมูลที่ถูกบีบอัดข้อมูลแล้ว (n2) ดังสมการที่ 2.13

$$CR = \frac{n1}{n2} \qquad 2.13$$

การหาความผิดพลาดของการบีบอัดข้อมูลนั้น เราสามารถใช้หลักการของ Signal to Noise Ratio (SNR) และ Peak Signal to Noise Ratio (PSNR) โดยจะเป็นตัววัดคุณภาพของการบีบอัดข้อมูลว่า เมื่อบีบอัดข้อมูลแล้วคุณภาพของภาพที่ได้ใหม่จะมีการสูญเสียมากน้อยเพียงไหน ดังสมการที่ 2.14 และสมการที่ 2.15

$$SNR_{(dB)} = 10\log_{10}\left\{\frac{\left[\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left(s(m,n)\right)^2\right]}{\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left(s(m,n)-\hat{s}(m,n)\right)^2}\right\} \qquad 2.14$$

$$PSNR_{(dB)} = 10\log_{10}\left\{\frac{\max|s(m,n)|}{\frac{1}{MN}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left(s(m,n)-\hat{s}(m,n)\right)^2}\right\} \qquad 2.15$$

กำหนดให้ : ค่า m และ n เป็นค่าของแกนนอนและแกนตั้ง

ค่า $s(m,n)$ เป็นข้อมูลของภาพต้นฉบับ

ค่า $\hat{s}(m,n)$ เป็นข้อมูลของภาพที่ถูกบีบอัดแล้ว

ค่า max เป็นค่าที่มากที่สุด

2.4 ทฤษฎีของฮัฟฟ์แมน (Huffman code)

กระบวนการที่กำหนดให้การจัดลำดับของเลขฐานสอง ไปเป็นสัญลักษณ์ เรียกว่า การเข้ารหัส (coding) เซตของการจัดลำดับของเลขฐานสอง เรียกว่า รหัส และเลขที่เป็นเลขเฉพาะของเซต เรียก รหัสคำ(codeword)

การเข้ารหัสที่เราสนใจ คือ การเข้ารหัสเอนโทรปี(Entropy coding) คือการลดความฟุ่มเฟือยด้วยวิธีการบีบอัดข้อมูลโดยไม่สนใจถึงความหมายของข้อมูล เพียงแต่มองข้อมูลเป็นบิต 0 หรือ 1 ซึ่งในการบีบอัดภาพที่นิยมอยู่ คือ การเข้ารหัสแบบฮัฟฟ์แมน (Huffman coding)

กระบวนการนี้ถูกพัฒนาขึ้นโดย เดวิท ฮัฟฟ์แมน(David Huffman) ซึ่งเป็นสมาชิกในทฤษฎีข่าวสารแรกของ โรเบิร์ต ฟาโน(Robert Fano) ที่ MIT การเกิดรหัสโดยการใช้กระบวนการนี้ จึงเรียกว่า รหัสของฮัฟฟ์แมน เป็นรหัสแบบ prefix code และเป็นสถานการณ์ที่ให้แบบตัวอย่าง(เซตของความน่าจะเป็น)

2.4.1การเข้ารหัสแบบฮัฟฟ์แมน (Encode Huffman)

กระบวนการของฮัฟฟ์แมนมีข้อสังเกตุพื้นฐาน 2 ข้อ คือ

2.4.1.1 ในรหัสข้อมูลปัจจุบันสัญลักษณ์ที่มีอัตราการเกิดบ่อยครั้ง (มีความน่าจะเป็นในการเกิดสูง) จะมีจำนวนบิตของรหัสข้อมูลสั้นลง

2.4.1.2 ในรหัสข้อมูลปัจจุบันสัญลักษณ์สองตัวที่มีการปรากฏน้อยที่สุดจะมีความยาวเท่ากัน

วิธีการสังเกต คือ ถ้าสัญลักษณ์มีอัตราการเกิดขึ้นสูง จะมีรหัสคำยาวกว่า จำนวนเฉลี่ยบิตต่อสัญลักษณ์มากกว่า ในเงื่อนไขการเปลี่ยนตำแหน่ง ดังนั้นจึงมีการกำหนดให้ รหัสคำที่ยาวกว่าเป็นสัญลักษณ์ที่มีความถี่ในของการเกิดสูงกว่า ไม่สามารถเป็น รหัสปัจจุบัน

การสังเกตุอีกวิธีหนึ่ง พิจารณาตามตำแหน่ง โดยการคาดคะเน จากรหัสที่เข้ามาที่มีรหัสคำเหมือนกันมีความถี่ในการเกิดต่ำ สัญลักษณ์จะไม่มีความยาวเท่ากัน สมมุติให้ความยาวของรหัสคำคือ k บิต ยาวกว่ารหัสคำที่สั้นรหัสคำที่สั้นกว่าจะไม่เป็น prefix ของรหัสคำที่ยาวกว่า หมายความว่าถ้าเราตกอยู่ที่ k บิตตัวสุดท้ายของรหัสคำที่ยาวกว่ารหัสคำอีกตัวหนึ่งจะมีความชัดเจนขึ้น

เนื่องจาก รหัสคำที่เหมือนกันที่มีอัตราการเกิดต่ำที่สุด ในสัญลักษณ์ที่เป็นกลุ่มตัวอักขระไม่มีรหัสคำตัวใดยาวกว่าตัวนี้ได้ ดังนั้นจึงไม่มีผลกระทบอันใดในการทำให้รหัสคำสั้นลงโดยจะเริ่มที่การเข้ารหัสพรีฟิก ของตัวอื่น เมื่อทำการตกที่ k บิต เราจะได้รหัสใหม่ที่มีความยาวเฉลี่ยสั้นกว่าเดิม

จะมีวิธีการโดยรวมค่าอัตราการเกิดของสัญลักษณ์2ตัวที่มีค่าอัตราการเกิดต่ำที่สุดจะมีความแตกต่างที่บิตสุดท้ายเท่านั้น นั่นคือ ถ้า γ และ δ คือสัญลักษณ์ 2ตัวสุดท้ายที่มีอัตราการเกิดต่ำที่สุดในสัญลักษณ์ของตัวอักขระ และถ้ารหัสคำของ γ คือ m*0 รหัสคำของδ จะเป็น m*1 โดยที่ m คือค่าของ 1s และ 0s และ * จะเห็นได้จากตัวอย่าง

ตัวอย่างที่ 1 การทำงานการเข้ารหัสแบบฮัฟฟ์แมน โดยการใช้ Table

ตารางที่ 2.1

Letter	Probability	Codeword
a1	0.2	c(a1)
a2	0.4	c(a2)
a3	0.2	c(a3)
a4	0.1	c(a4)
a5	0.1	c(a5)

ขั้นตอนที่1 ทำการเรียงสัญลักษณ์ใหม่โดยเรียงจากอัตราการการเกิด จากมาก ไปน้อย

ตารางที่ 2.2

Letter	Probability	Codeword
a2	0.4	c(a2)
a1	0.2	c(a1)
a3	0.2	c(a3)
a4	0.1	c(a4)
a5	0.1	c(a5)

ขั้นตอนที่2 สัญลักษณ์ที่มีอัตราการเกิดต่ำที่สุด2ตัวสุดท้าย ในที่นี้คือ a4 และ a5 นำค่าอัตราการเกิดรวมกันจะได้ค่า ของอัตราการเกิดใหม่ และกำหนดให้ รหัสคำใหม่ คือ α ดังตาราง ที่2.2

ตารางที่ 2.3

Letter	Probability	Codeword
a2	0.4	c(a2)
a1	0.2	c(a1)
a3	0.2	c(a3)
a4	0.2	α1

ทำขั้นตอนที่1และ2ไปเรื่อยๆจนกว่าจะได้ค่าที่เหลือสุดท้าย 2ค่า ดังตารางที่2.4 และตารางที่2.5

ตารางที่ 2.4

Letter	Probability	Codeword
a2	0.4	c(a2)
a3	0.4	$\alpha 2$
a1	0.2	c(a1)

ตารางที่ 2.5

Letter	Probability	Codeword
a3	0.6	$\alpha 3$
a2	0.4	c(a2)

ขั้นตอนที่3 กำหนดค่าโคทเวิร์ดให้กับสองค่าสุดท้ายโดยให้ค่าที่มีอัตราการเกิดสูงสุดมีค่าเท่ากับ **0** และค่าที่มีอัตราการเกิดต่ำกว่ามีค่าเท่ากับ 1

$$\alpha 3 = 0$$

$$c(a2) = 1$$

จาก$\alpha 3$ คือ ค่าของ $\alpha 2$ + c(a1) ดังนั้นจะได้

$$\alpha 2 = (\alpha 3 * 0) = 00$$

$$c(a1) = (\alpha 3 * 1) = 01$$

จาก$\alpha 2$ คือ ค่าของ c(a3) +$\alpha 1$ ดังนั้นจะได้

$$c(a3) = (\alpha 2 * 0) = 000$$

$$\alpha 1 = (\alpha 2 * 1) = 001$$

จาก$\alpha 1$ คือ ค่าของ c(a4) +c(a5) ดังนั้นจะได้

$$c(a4) = (\alpha 1 * 0) = 0010$$

$$c(a5) = (\alpha 1 * 1) = 0011$$

เราจะได้ผลของการถอดรหัสแบบฮัฟฟ์แมนได้ดังตารางที่4.6

ตารางที่ 2.6

Letter	Probability	Codeword
a2	0.4	1
a1	0.2	01
a3	0.2	000
a4	0.1	0010
a5	0.1	0011

ตัวอย่างที่ 2 การทำงานการเข้ารหัสแบบฮัฟฟ์แมน โดยใช้แผนภูมิต้นไม้

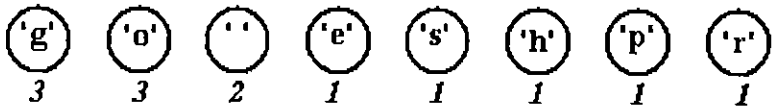
ยกตัวอย่าง ประโยค "GO GO GOPHERS " ซึ่งมีทั้งหมด 13 อักขระ (รวมช่องว่างภายใน) สามารถจัดเก็บได้ในเครื่องขนาด 8 บิท โดยใช้หน่วยความจำทั้งสิ้น 104 บิท (1 อักขระ = 8 บิท) แต่ถ้าเก็บแบบ 3 บิท จะใช้หน่วยความจำทั้งสิ้นเพียง 39 บิท ซึ่งถือเป็นค่า ต่ำสุด ที่เราต้องใช้สำหรับการเก็บข้อมูลชุดนี้ตามปกติ

อย่างไรก็ตาม ประโยคนี้เมื่อผ่านการเข้ารหัสแบบฮัฟฟ์แมน จะทำให้มีความจุเหลือเพียง 37 บิท เท่านั้น ดังมีรายละเอียดดังที่เห็น

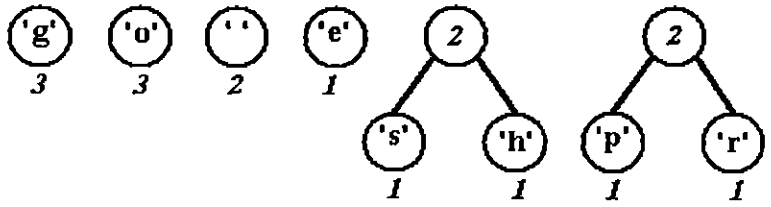
ตัวอย่างการเข้ารหัสแบบฮัฟฟ์แมน (Huffman Encoding)

เราสามารถจัดเก็บข้อมูลในประโยค "GO GO GOPHERS " ซึ่งมีทั้งหมด 13 อักขระ (รวมช่องว่างภายใน) โดยการเข้ารหัสแบบฮัฟฟ์แมน ได้ดังนี้

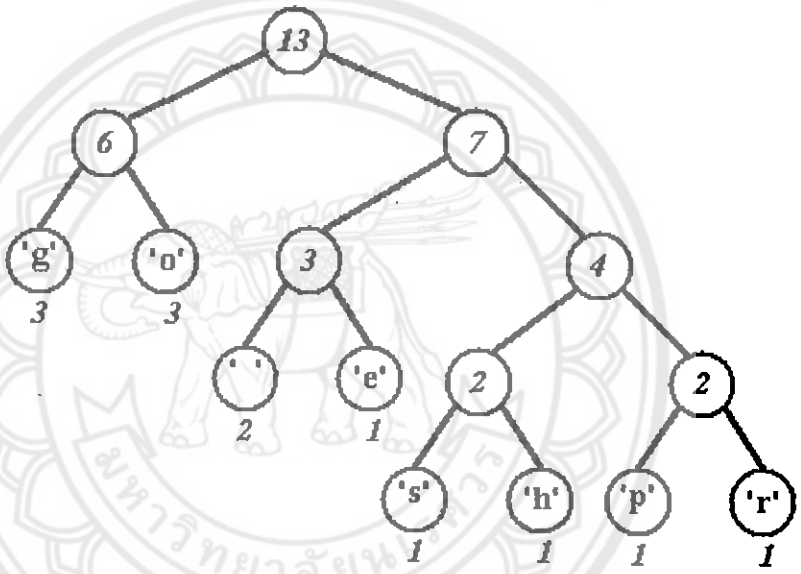
1. คำนวณ ค่าความถี่ ในการปรากฏของแต่ละอักขระในประโยค ได้ผลดังนี้



2. สร้าง แผนภูมิต้นไม้ โดยนำเอาค่าความถี่ น้อยที่สุด ในชุด มาบวกกันครั้งละ 2 ตัว ได้เป็นค่าความถี่ใหม่ขึ้นมา



3. จากนั้นดำเนินการแบบเดิม (ตามขั้นที่ 2) ไปเรื่อย ๆ จนกระทั่ง ได้ค่าความถี่รวมสูงสุดค่าเดียว อยู่บนสุด ซึ่งจะมีค่าเท่ากับจำนวนอักขระในชุดข้อมูลดังกล่าวพอดี (คือ 13 ในที่นี้) สุดท้ายเราจะได้ แผนภูมิต้นไม้ ดังแสดงในรูป



จากนั้น จึงมาสร้างเส้นทางของอักขระแต่ละตัว นับจากส่วนยอดลงมา โดยกำหนดให้การเคลื่อนที่ ไปทางซ้ายแทนด้วย 0 และ ทางขวาด้วย 1 ได้ว่า G = 00, O = 01, _= 100, E = 101, S = 1100, H = 1101, P = 1110,และ R = 1111

ด้วยเหตุนี้เราจึงเขียนประโยค "GO GO GOPHERS" ใหม่โดยใช้การเข้ารหัสแบบฮัฟฟ์แมน ข้างต้นได้เป็น 00 01 100 00 01 100 00 01 1110 1101 101 1111 1100 ซึ่งใช้หน่วยความจำเพียง 37 บิต เท่านั้น

2.4.2 การถอดรหัสของแบบฮัฟฟ์แมน (Decode Huffman)

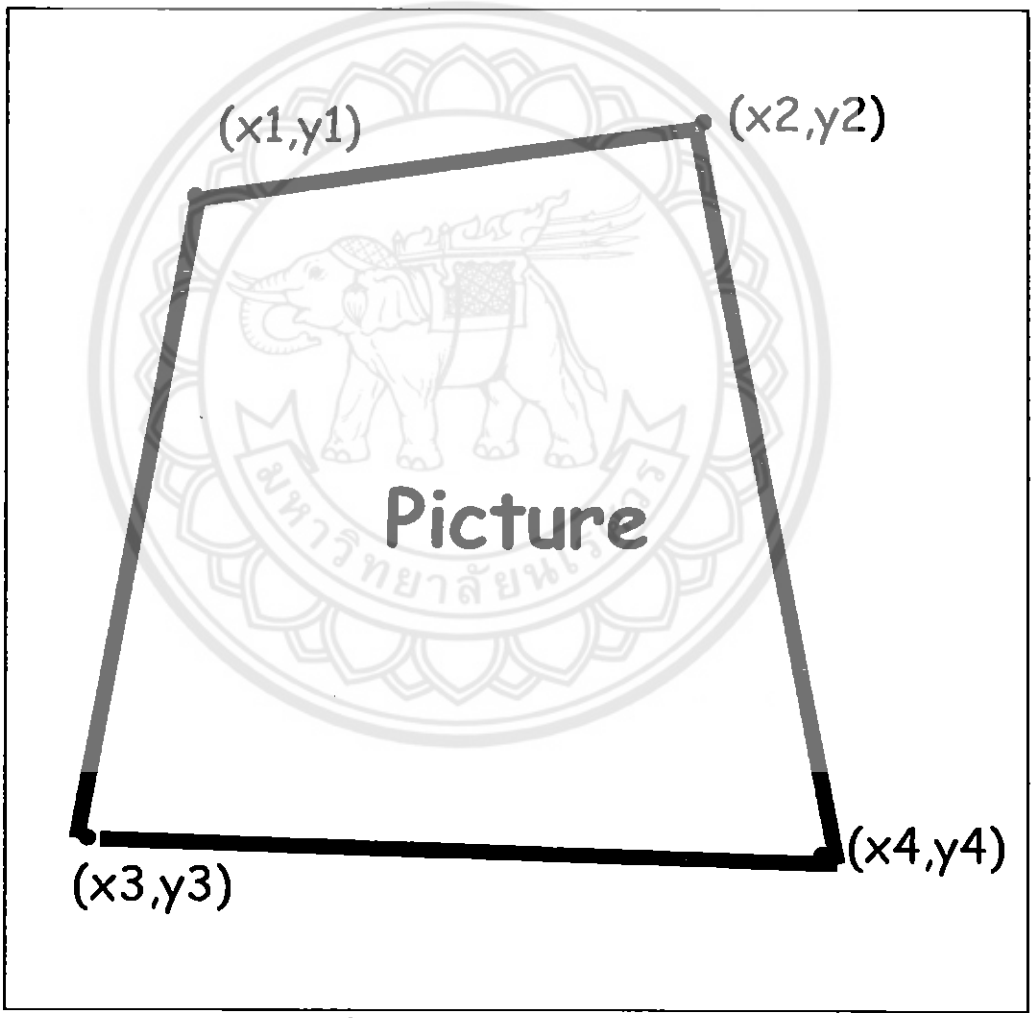
การถอดรหัสจะทำกลับการกระบวนการเข้ารหัส กล่าวคือนำค่าที่เก็บมาใช้ในการถอดรหัส ดังตัวอย่างที่แล้วจะเก็บค่าของ G=00,O=01, _=100,E =101, S=1100, H=1101, P = 1110,และ R=1111 ไว้ เมื่อนำมาเทียบกันค่าที่เข้ารหัสไปแล้วคือ 00 01 100 00 01 100 00 01 1110 1101 101 1111 1100 ก็จะได้ประโยคนี้ "GO GO GOPHERS"

2.5 หลักการหาขอบเขตที่สนใจ (Region of interest : ROI)

หลักการในการหาค่าของขอบเขตที่สนใจ คือ การกำหนดขอบเขตที่สนใจโดยที่จะใช้การกำหนดพิกัดในพิกเซลของภาพเข้ามาช่วยในการกำหนดขอบเขตขึ้นมา

คอมพิวเตอร์สามารถคำนวณจุดพิกเซลจำนวนหนึ่ง หรือค่าจุดพิกเซลของขอบเขตที่สนใจข้างในภาพนั้น

กล่าวคือถ้ามีภาพหนึ่งภาพแล้วเราจะกำหนดขอบเขตที่สนใจเราสามารถกำหนดจากจุดพิกเซลบนภาพนั้นๆได้เลย ดังรูปที่ 2.13

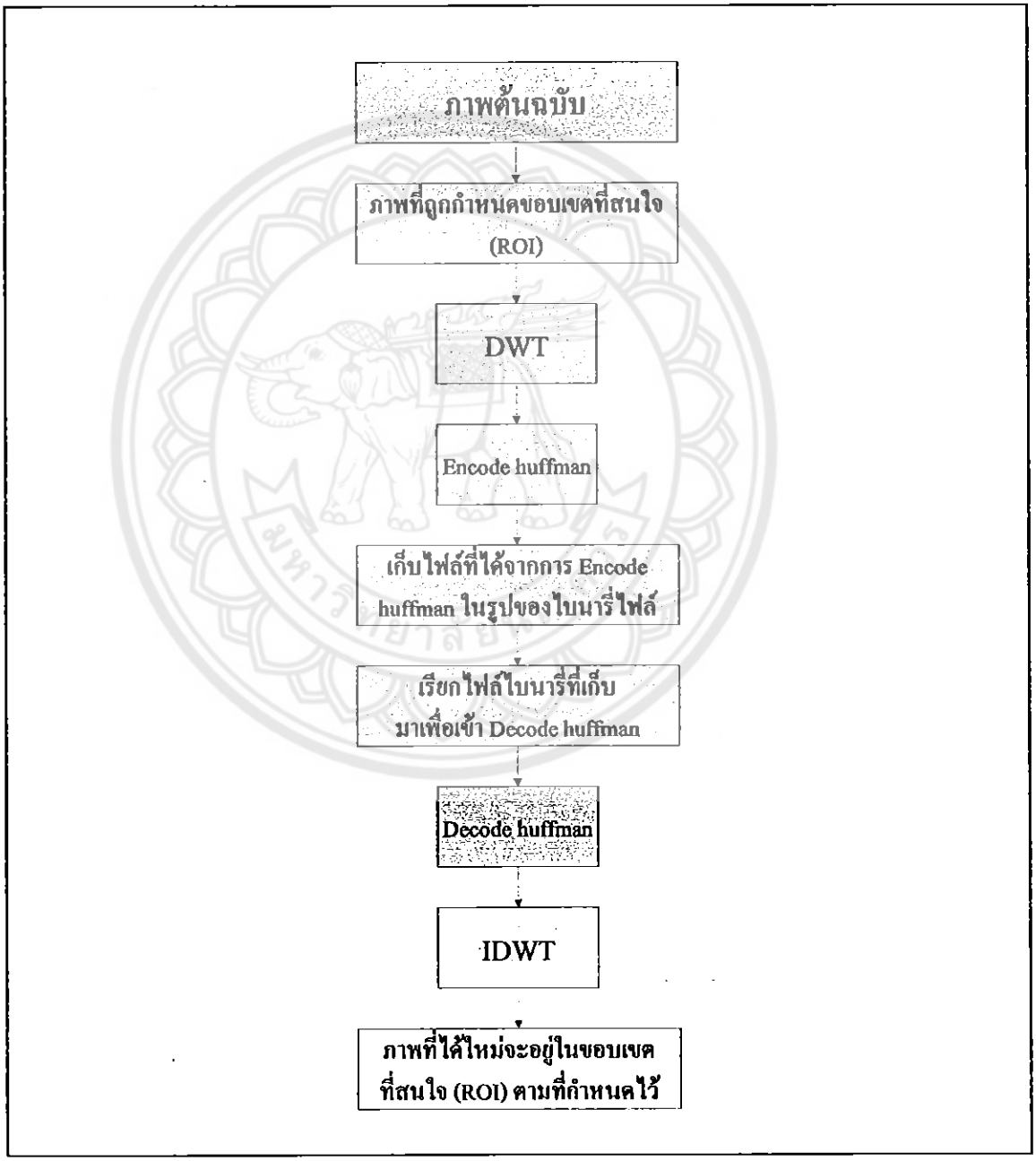


รูปที่ 2.13 แสดงการกำหนดจุดพิกเซลในภาพ

บทที่3
วิธีการดำเนินโครงงาน

3.1 วิธีการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต

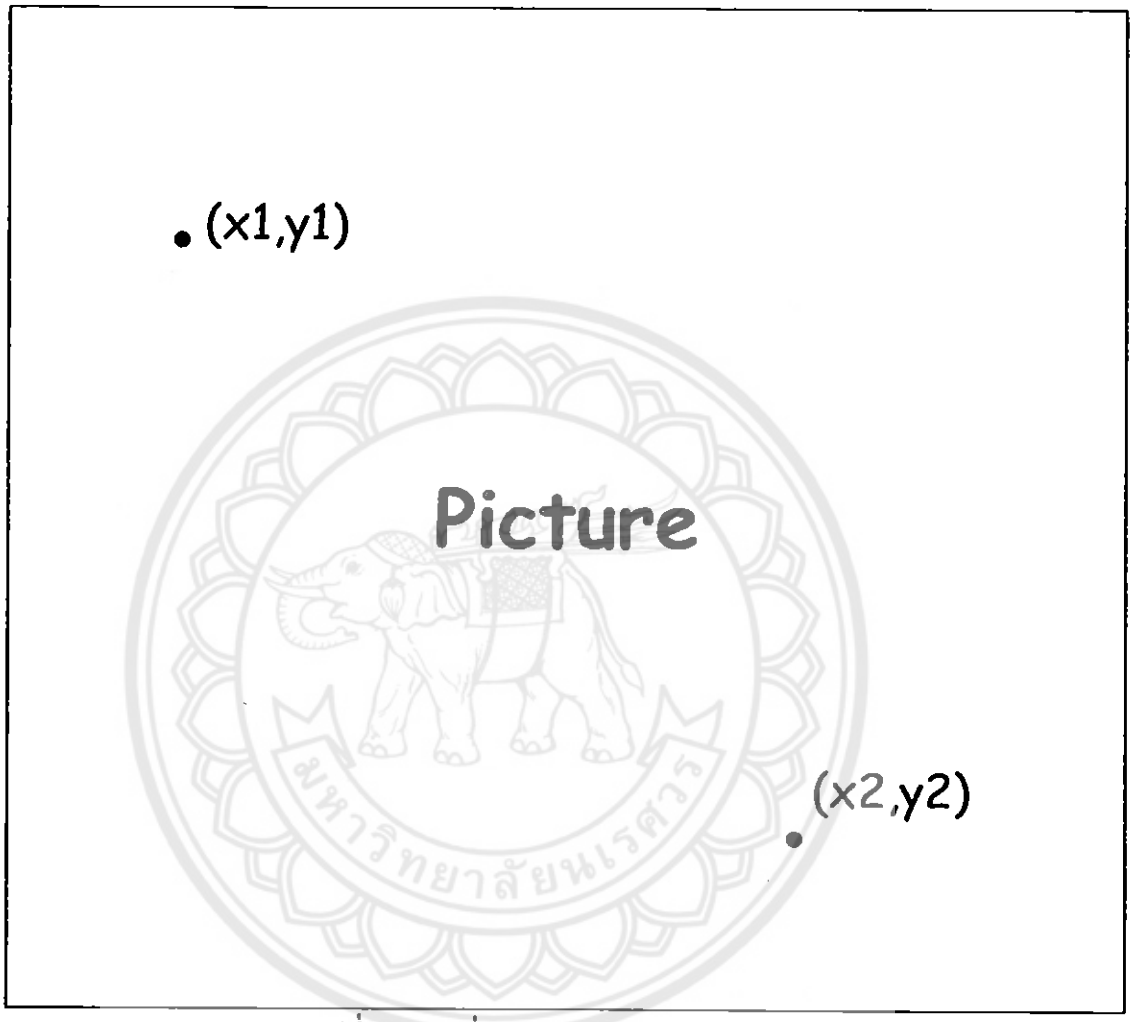
ลักษณะของการบีบอัดข้อมูลภาพโดยวิธีการการแปลงเวฟเล็ต คือ



รูปที่ 3.1 แผงผังวิธีการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต

3.2 วิธีการหาขอบเขตที่สนใจ (Region of interest : ROI)

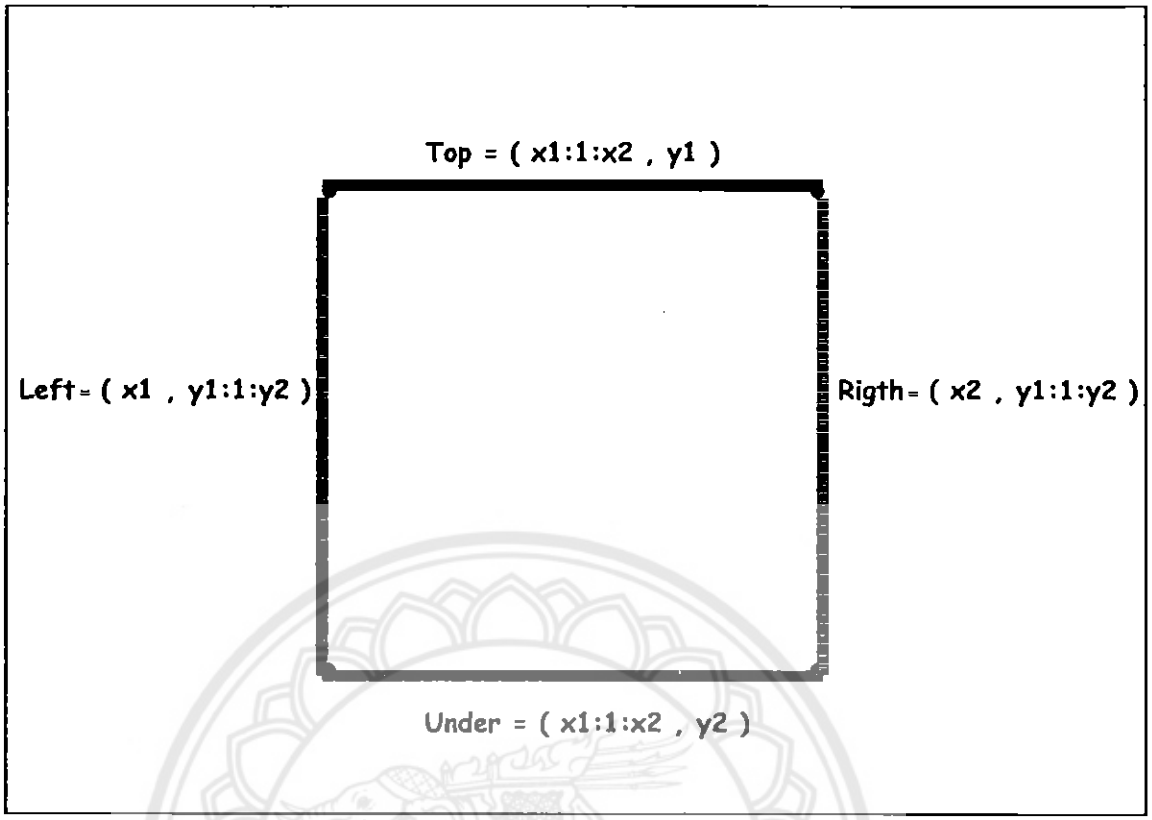
กำหนดขอบเขตที่สนใจด้วยวิธีการกำหนดจุดพิกเซลในภาพต้นฉบับจำนวน 2 จุด โดยจุดพิกเซลที่กำหนดจะเป็นจุดทแยงกันและกัน ดังรูปที่ 3.2



รูปที่ 3.2 ภาพที่แสดงการกำหนดจุดพิกเซล 2 จุด

จากรูปที่ 3.2 นั้นจุดพิกเซล 2 จุดที่กำหนดจะมีค่าเป็น (x1 , y1) และ (x2 , y2) บนแกน x และ y ตามลำดับ

ในการกำหนดขอบเขตที่สนใจ ทำได้โดยการกำหนดจุดสองจุดในขอบเขตที่เราสนใจ แล้วจึงนำค่าพิกัดที่กำหนดจุดมาคิดหาระยะ หรือ กรอบของภาพ ที่เป็นขอบเขตที่เราสนใจ และนำส่วนดังกล่าวมาเข้าขบวนการปรับอัดข้อมูลภาพ ดังรูปที่ 3.3



รูปที่ 3.3 ภาพแสดงการกำหนดขอบเขตที่สนใจ

3.3 การบีบอัดข้อมูลภาพ

การบีบอัดข้อมูลภาพ สามารถแบ่งเป็นสองส่วนคือ เข้ารหัส (Encode) และ ถอดรหัส (Decode)

1. การเข้ารหัสของการบีบอัดข้อมูลภาพ โดยเริ่มจากการนำภาพที่อยู่ในขอบเขตที่เราสนใจ มาเข้าขบวนการแปลงเวฟเล็ต แล้วจึงเข้ารหัสแบบฮัฟฟ์แมน เพื่อลดจำนวนบิตของข้อมูลลง ผลที่ได้ก็คือจำนวนบิตของข้อมูลลดลงจากข้อมูลภาพที่ที่ผ่านขบวนการดังกล่าว (ภาพที่อยู่ในขอบเขตที่สนใจ) และค่าที่จะเป็นเลขไบนารี่

2. การถอดรหัสของการบีบอัดข้อมูลภาพ คือ การนำเอาผลที่ได้จากการเข้ารหัสเรียบร้อยแล้วมาถอดรหัสแบบฮัฟฟ์แมน แล้วจึงเข้าขบวนการแปลงกลับเวฟเล็ต เพื่อให้ได้ภาพใหม่ที่มีคุณภาพเท่ากับภาพต้นแบบ (ภาพที่อยู่ในขอบเขตที่สนใจ) โดยพิจารณาคุณภาพได้จากค่าของ Compression ratio(CR), SNR และ PSNR

จากวิธีการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ตและการแบ่งส่วนหลังมาแล้ว จะนำหลักการที่ได้ศึกษามาในบทที่ 2 มาประยุกต์ใช้ เนื้อหาในหัวข้อนี้จะแบ่งเป็น 2 ส่วนใหญ่ๆคือการเข้ารหัสและการถอดรหัส

ขั้นตอนในการดำเนินโครงงานนี้พัฒนาโดยใช้โปรแกรมMatlab เวอร์ชั่น 7 ในบทนี้จะอธิบายถึงวิธีการเข้ารหัสและถอดรหัส ส่วนผลที่ได้จะได้อธิบายที่บทต่อไป และโปรแกรมที่ที่ได้พัฒนาแล้วจะนำไว้ในภาคผนวก

ว000658

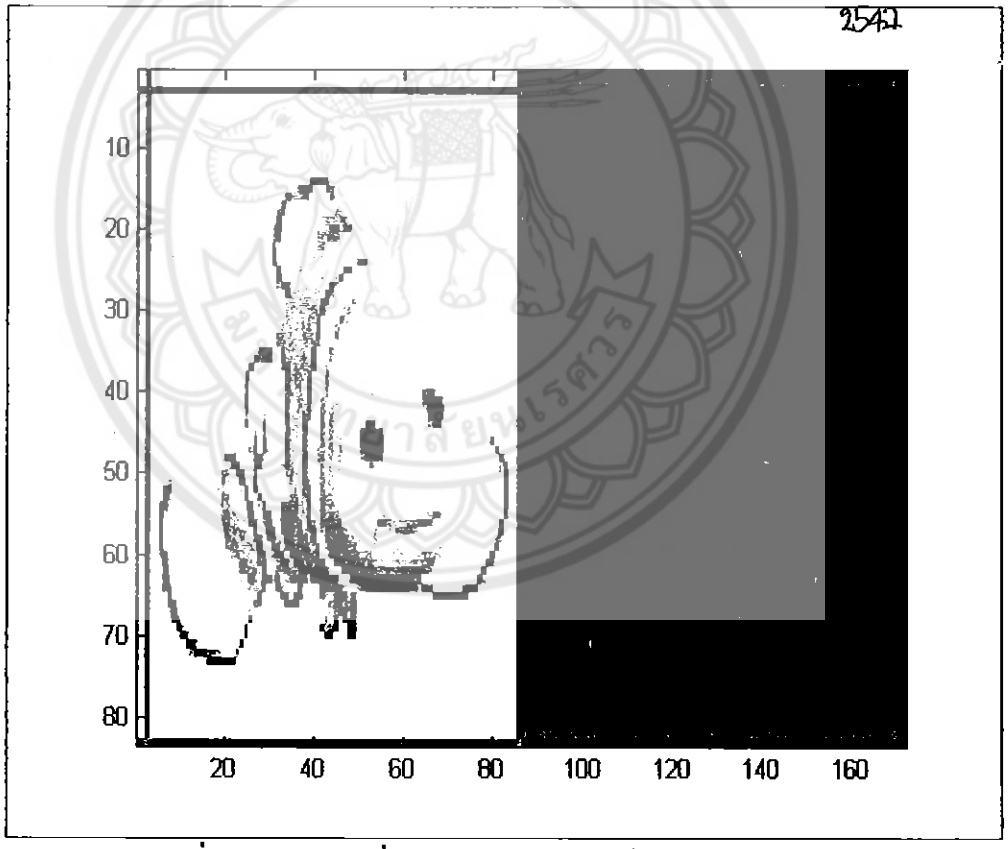
3.3.1 การเข้ารหัส

การแปลงเวฟเล็ตที่ใช้คือ การแปลงแบบเต็มหน่วย (Discrete Wavelets Transform : DWT) โดยมีวิธีการคือ เมื่อมีสัญญาณเข้ามา จะนำมาแยกสัญญาณเป็น 2 ส่วน แล้วจะได้ความถี่ต่ำ(Low pass filter) และความที่สูง(High pass filter)

ข้อมูลภาพเป็นข้อมูลของสัญญาณที่อยู่ในสองมิติ คือ ในแกนแนวนอน(X) และในแกนแนวตั้ง (Y) ในโปรแกรม Matlab เวอร์ชั่น 7 จะเก็บอยู่ในรูปของ Matrix สองมิติ

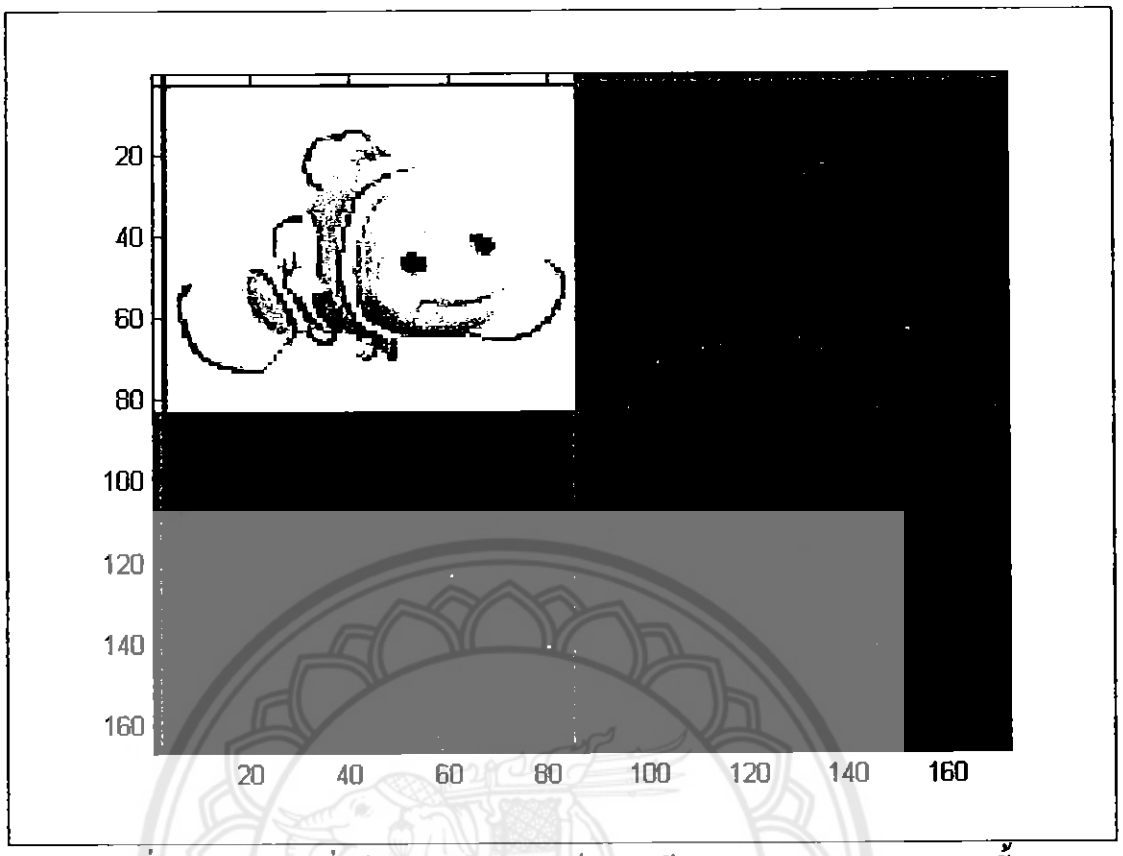
การแปลงเวฟเล็ตของข้อมูลภาพจะทำโดยการเลือกข้อมูลของแต่ละแถว ตามแนวนอน มาผ่านการแปลงเวฟเล็ตแบบหนึ่งมิติ จนครบข้อมูลของทุกแถว ดังรูปที่ 3.4

ปร.
ว223ก
2543



รูปที่ 3.4 แสดงภาพที่ได้จากการแปลงเวฟเล็ตของเส้นภาพตามแนวนอน

จากนั้น ข้อมูลที่ได้จะถูกแปลงเวฟเล็ตอีกครั้งตามแนวตั้ง ดังรูปที่ 3.5



รูปที่ 3.5 แสดงภาพที่ได้จากการแปลงเวฟเล็ตของเส้นภาพตามแนวนอนและแนวตั้ง

เมื่อได้ข้อมูลที่แปลงเวฟเล็ตแล้วจะนำข้อมูลดังกล่าวมาเข้ารหัสแบบฮัฟฟ์แมน เพื่อที่จะทำการลดขนาดให้เล็กลงอีก แล้วค่าที่ได้จะเป็นค่าของเลขไบนารี่ โดยในการเข้ารหัสแบบฮัฟฟ์แมนนั้นเราทำการเก็บค่าตารางของการเข้ารหัสไว้เพื่อที่จะใช้ในการถอดรหัสต่อไป

เลขไบนารี่ที่ได้จะนำมาเก็บค่าไว้ที่ไฟล์ "medical.kj"

3.3.2 การถอดรหัส

การถอดรหัสจะทำการเรียกไฟล์ "medical.kj" มาทำการถอดรหัสแบบฮัฟฟ์แมนเพื่อที่จะได้ค่าของข้อมูลที่ไม่ใช่เลขไบนารี่ ในการถอดรหัสแบบฮัฟฟ์แมนจะนำค่าตารางที่เก็บไว้ตอนที่เข้ารหัสมาใช้โดยการเปรียบเทียบค่ากับตารางนั้น

หลักจากที่ทำการถอดรหัสแบบฮัฟฟ์แมนแล้ว นำข้อมูลที่ได้มาทำการแปลงกลับเวฟเล็ตของข้อมูลภาพโดยจะทำกลับจากการทำการแปลงเวฟเล็ต

โดยที่การแปลงรหัสจะทำที่แนวนอนก่อนแล้วจึงทำที่แนวตั้ง แต่การแปลงกลับจะทำที่แนวตั้งก่อนแล้วจึงทำที่แนวนอน

จะได้ภาพใหม่ที่ได้ผ่านการบีบอัดข้อมูลแล้ว นำข้อมูลภาพใหม่ไปหาความผิดพลาด (SNR , PSNR) รวมถึงอัตราการบีบอัดข้อมูล (Compression Ratio : CR) ตามสมการในบทที่ 2

3.4 การออกแบบโปรแกรม

จากบทที่ 2 และหัวข้อที่ 3.1 , หัวข้อที่ 3.2 และหัวข้อที่ 3.3 นั้นสามารถนำทฤษฎีและวิธีการที่ได้ศึกษามา ออกแบบโปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ตและการแบ่งส่วนหลัง ดังจะแสดงต่อไปนี้

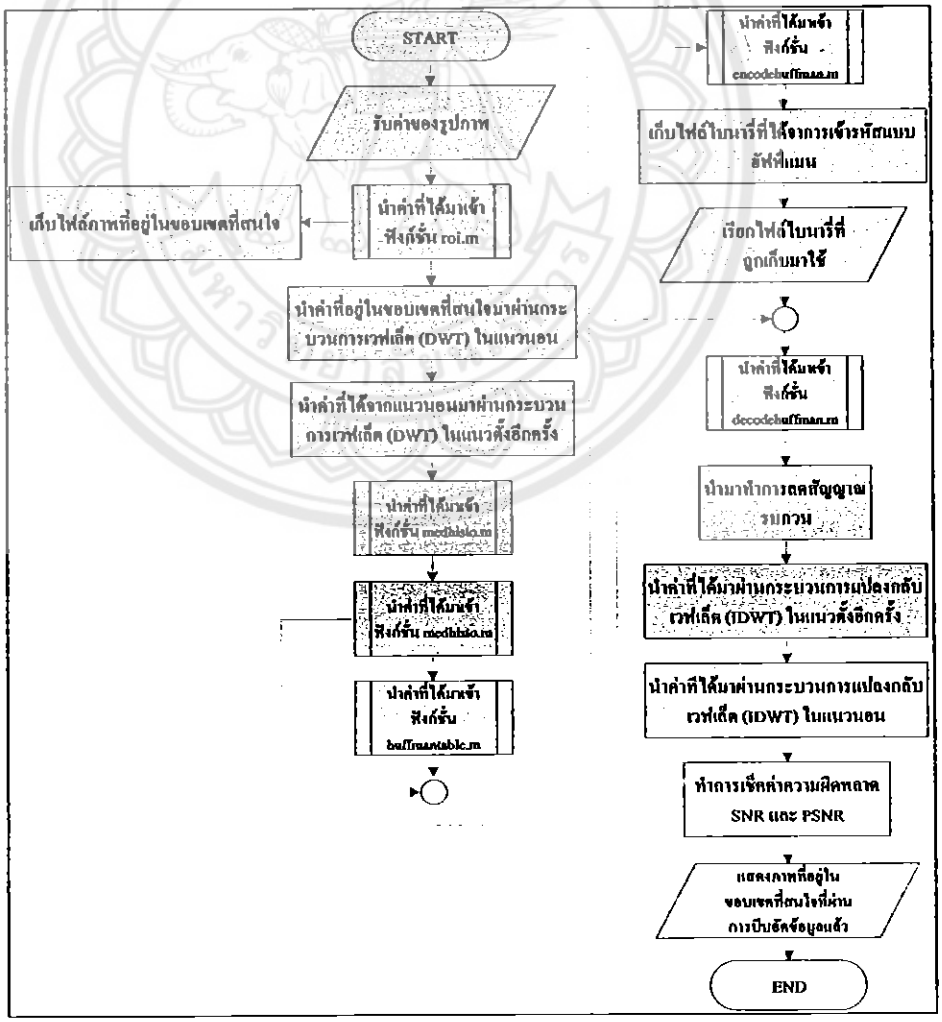
โปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ตและการแบ่งส่วนหลัง แบ่งออกเป็น 2 ส่วนใหญ่ คือ

1. ส่วนของโปรแกรม ได้แก่ testmed.m , test3.m และ hufftest.m

2. ส่วนของฟังก์ชั่นที่เรียกใช้ ได้แก่ roi.m , medhisto.m , huffmantable , encodehuffman และ decodehuffman.m

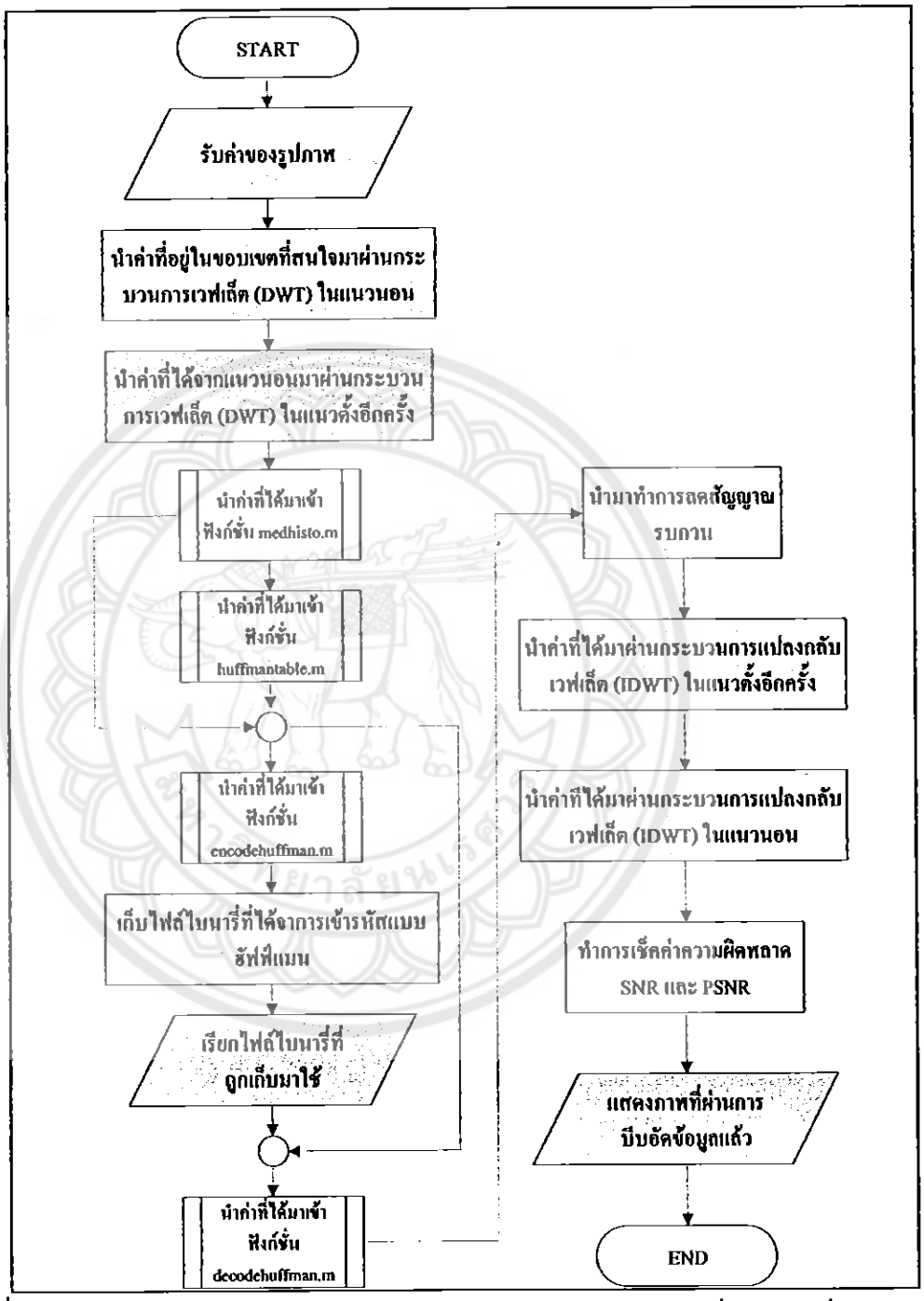
3.4.1 ส่วนของโปรแกรม

ไฟล์ testmed.m เป็นโปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ตและการแบ่งส่วนหลัง มีขั้นตอนการทำงานดังรูปที่ 3.6 (สามารถดูโปรแกรมได้จากภาคผนวก)



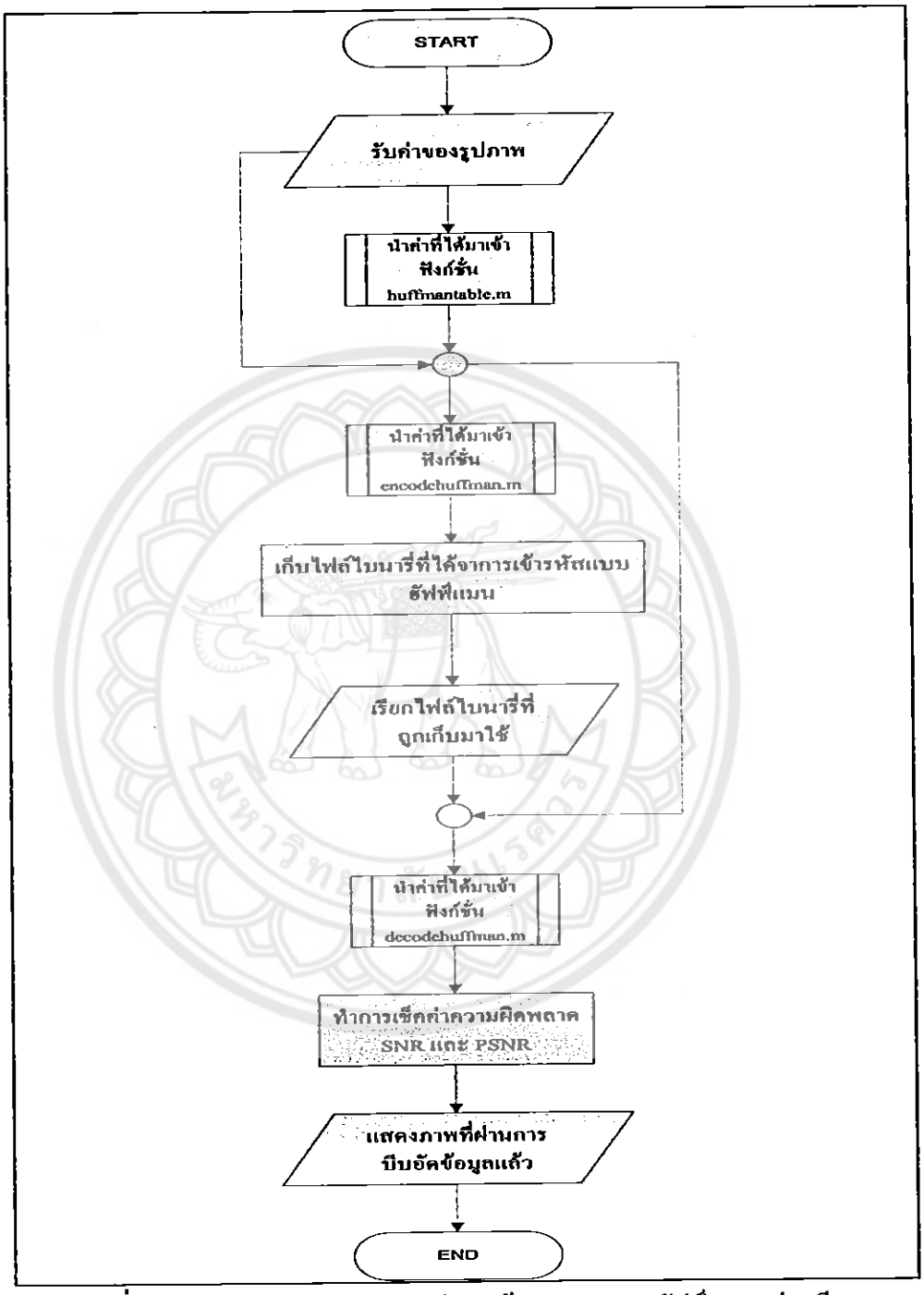
รูปที่ 3.6 Flowchart โปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ตและการแบ่งส่วนหลัง

ไฟล์ test3.m เป็นโปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ตและฮัฟฟ์แมน มีขั้นตอนการทำงานดังรูปที่ 3.7 (สามารถดูโปรแกรมได้จากภาคผนวก)



รูปที่ 3.7 Flowchart โปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ตและฮัฟฟ์แมน

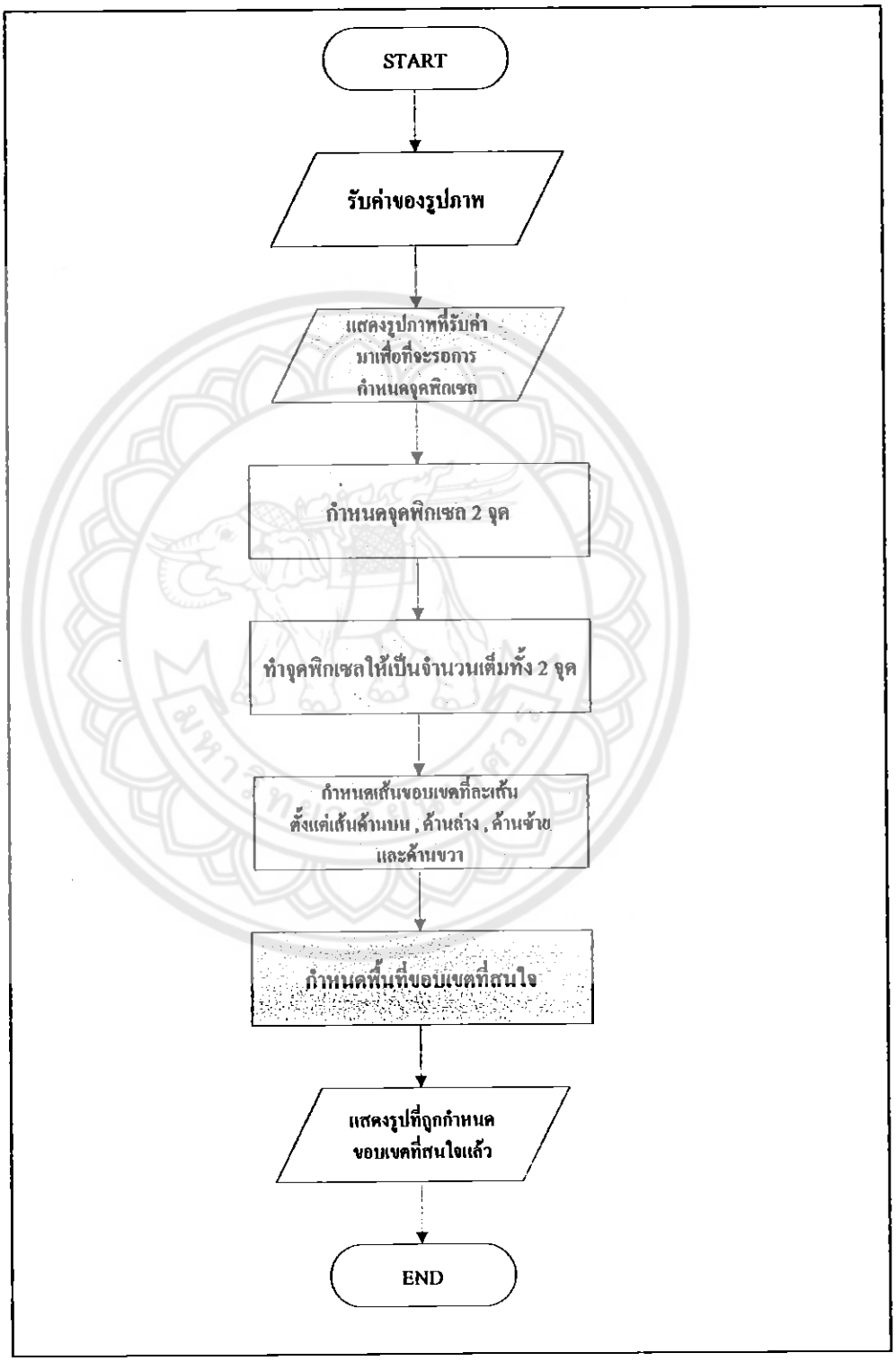
ไฟล์ hufftest.m เป็นโปรแกรมการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียว มีขั้นตอนการทำงานดังรูปที่ 3.8 (สามารถดูโปรแกรมได้จากภาคผนวก)



รูปที่ 3.7 Flowchart โปรแกรมการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียว

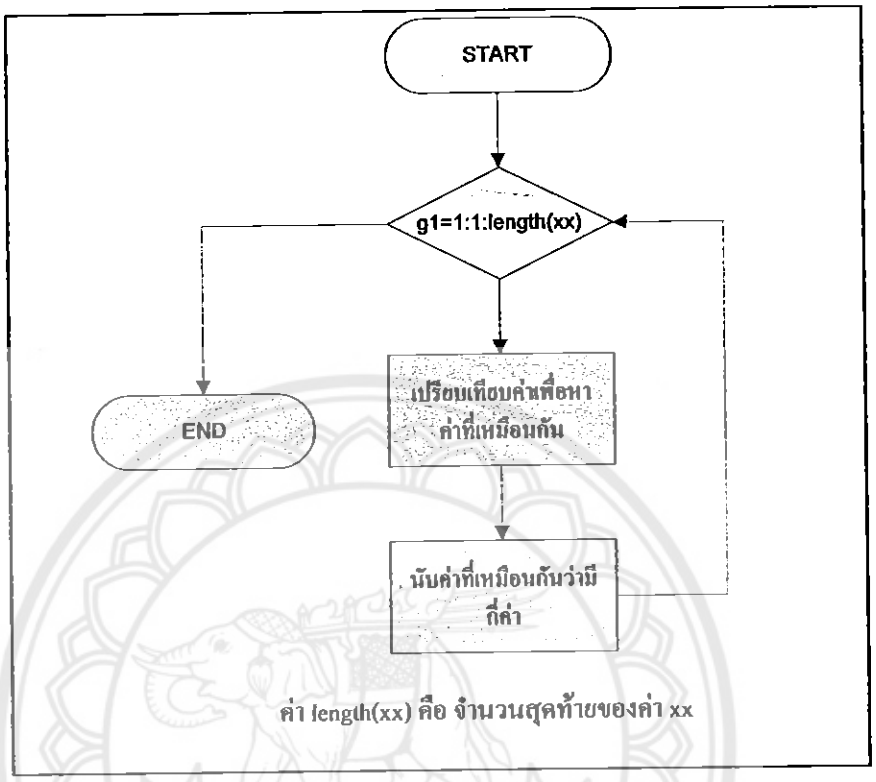
3.4.2 ส่วนของฟังก์ชั่นที่เรียกใช้

ฟังก์ชั่น roi.m เป็นฟังก์ชั่นที่ทำหน้าที่ในการกำหนดขอบเขตที่สนใจ มีขั้นตอนการทำงานดังรูปที่ 3.8 (สามารถดูฟังก์ชั่นได้จากภาคผนวก)



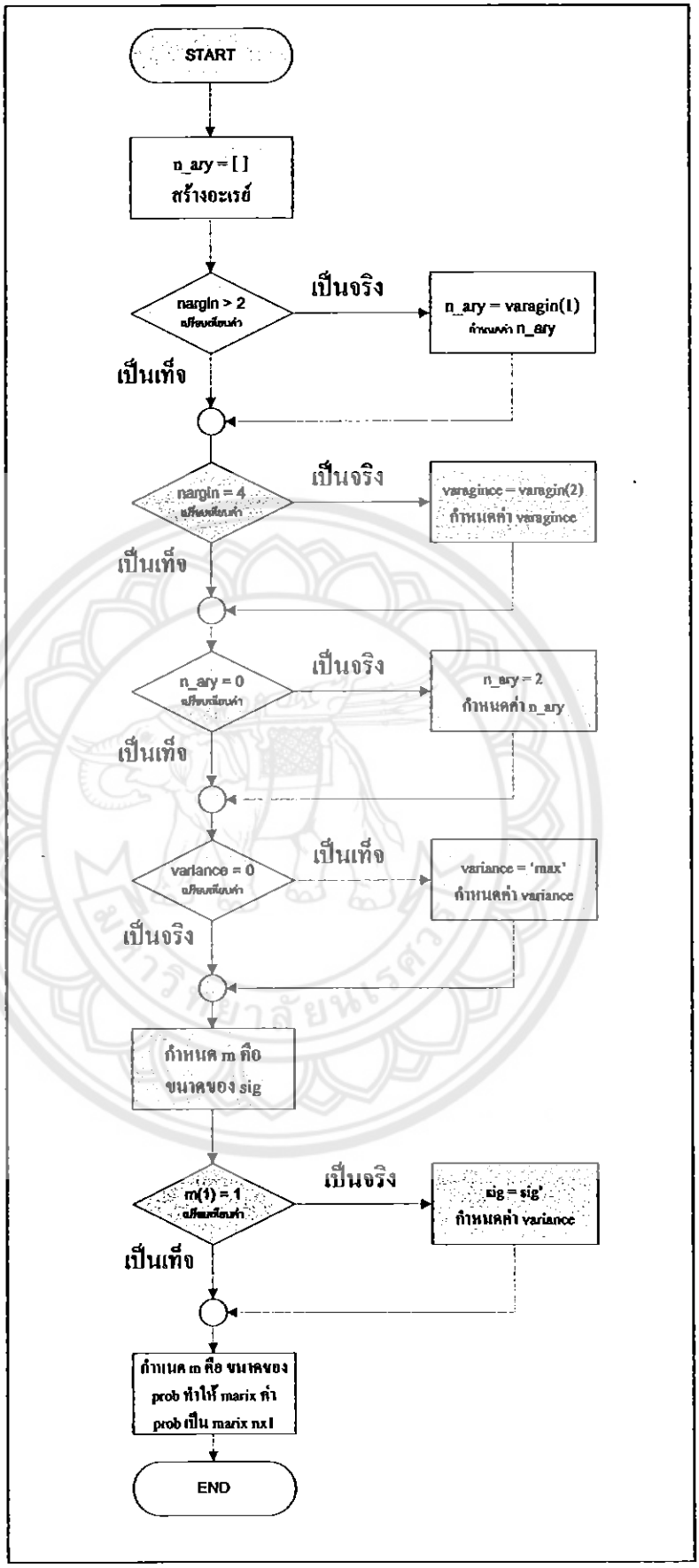
รูปที่ 3.8 Flowchart ฟังก์ชั่นการกำหนดขอบเขตที่สนใจ

ฟังก์ชั่น medhisto.m เป็นฟังก์ชั่นที่ทำหน้าที่ในการหาค่าความถี่ของข้อมูล มีขั้นตอนการทำงานดังรูปที่ 3.9 (สามารถดูฟังก์ชั่นได้จากภาคผนวก)



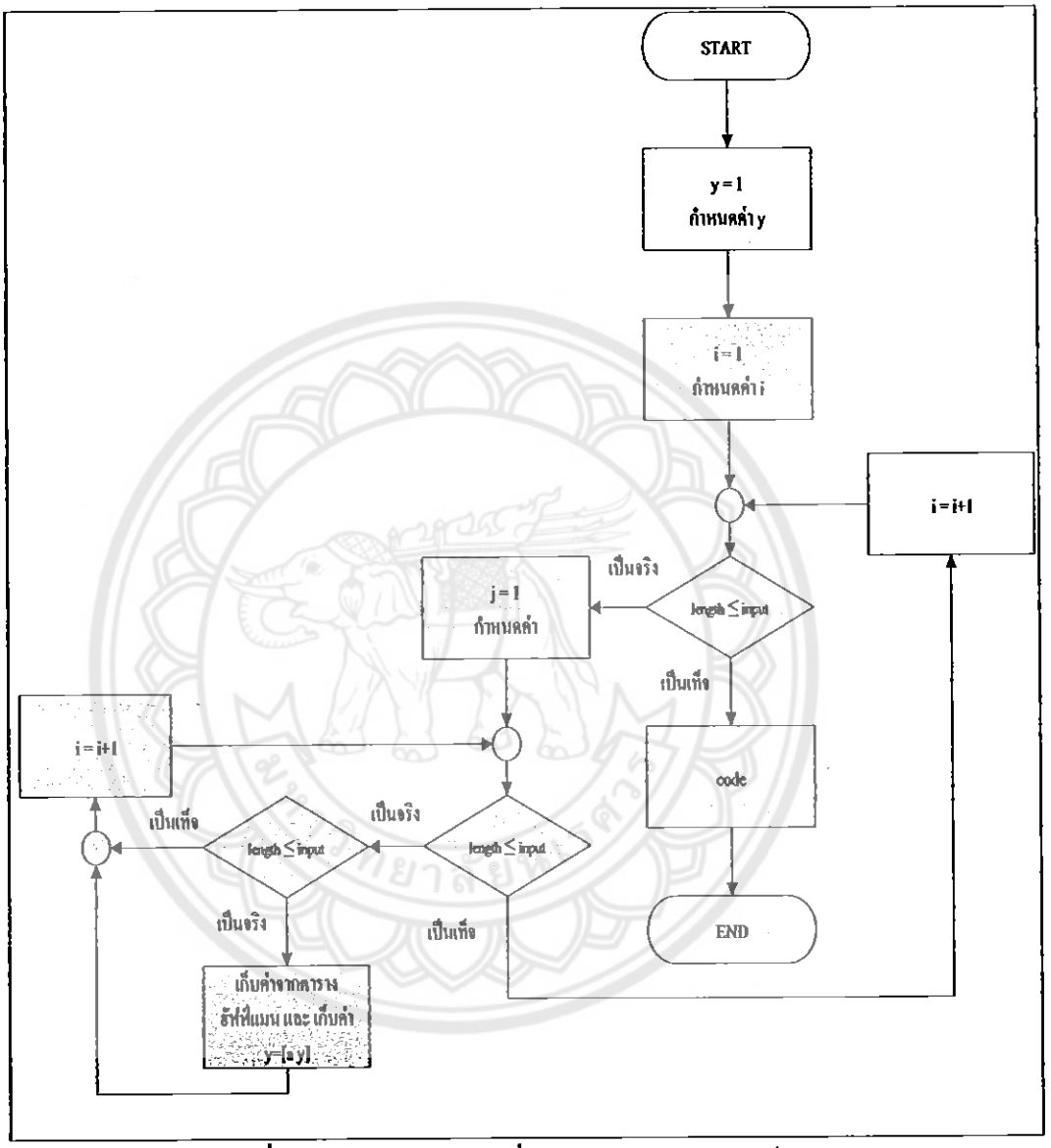
รูปที่ 3.9 Flowchart ฟังก์ชั่นการหาค่าความถี่ของข้อมูล

ฟังก์ชั่น huffmantable.m เป็นฟังก์ชั่นที่ทำหน้าที่ในการสร้างตารางรหัส ตามวิธีของการเข้ารหัสแบบฮัฟฟ์แมน โดยจะเริ่มจากการสร้างแผนภูมิฮัฟฟ์แมนก่อน ทำให้ได้โครงสร้างของแผนภูมิฮัฟฟ์แมน โดยเก็บไว้ในตัวแปรอาเรย์ จากนั้นจะทำการแปลงแผนภูมิฮัฟฟ์แมนให้เป็นรหัสฮัฟฟ์แมนโดยจะทำการเก็บไว้ในตัวแปรอาเรย์ เพื่อใช้ในการเข้ารหัสแบบฮัฟฟ์แมนและถอดรหัสแบบฮัฟฟ์แมนต่อไป มีขั้นตอนการทำงานดังรูปที่ 3.10 (สามารถดูฟังก์ชั่นได้จากภาคผนวก)



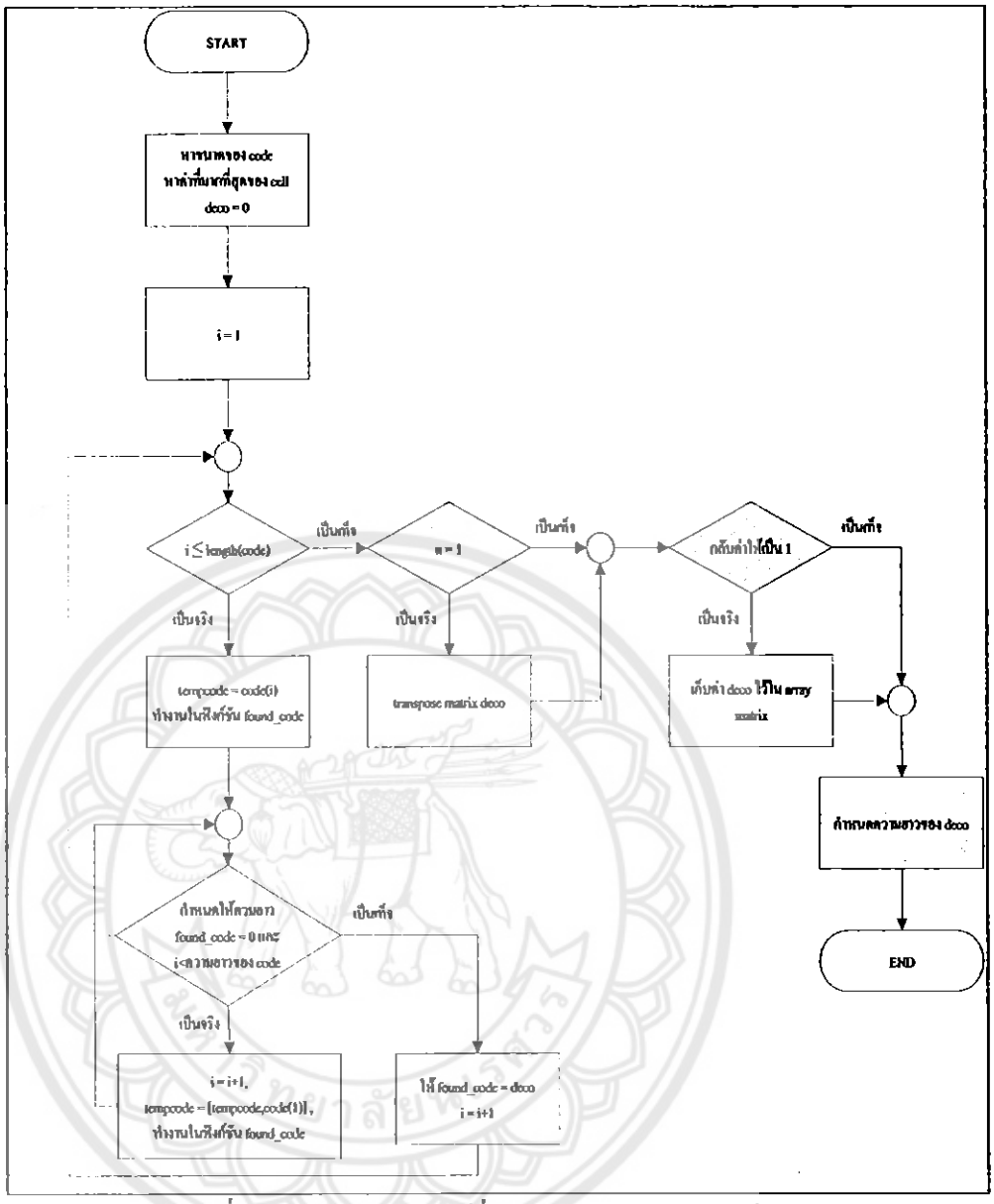
รูปที่ 3.10 Flowchart ฟังก์ชั่นสร้างตารางรหัสฮัฟฟ์แมน

ฟังก์ชั่น encodehuffman.m เป็นทำหน้าที่ในการเขียนข้อมูลที่ละบิตโดยจะรวมให้ครบ 8 บิต ก่อนแล้วจึงทำการเขียน เนื่องจากรหัสฮัฟฟ์แมนมีความยาวของบิตที่แปรแปลี่ยนได้หรือมีความยาวของจำนวนบิตที่ไม่เท่ากัน มีขั้นตอนการทำงานดังรูปที่ 3.11 (สามารถดูฟังก์ชั่นได้จากภาคผนวก)

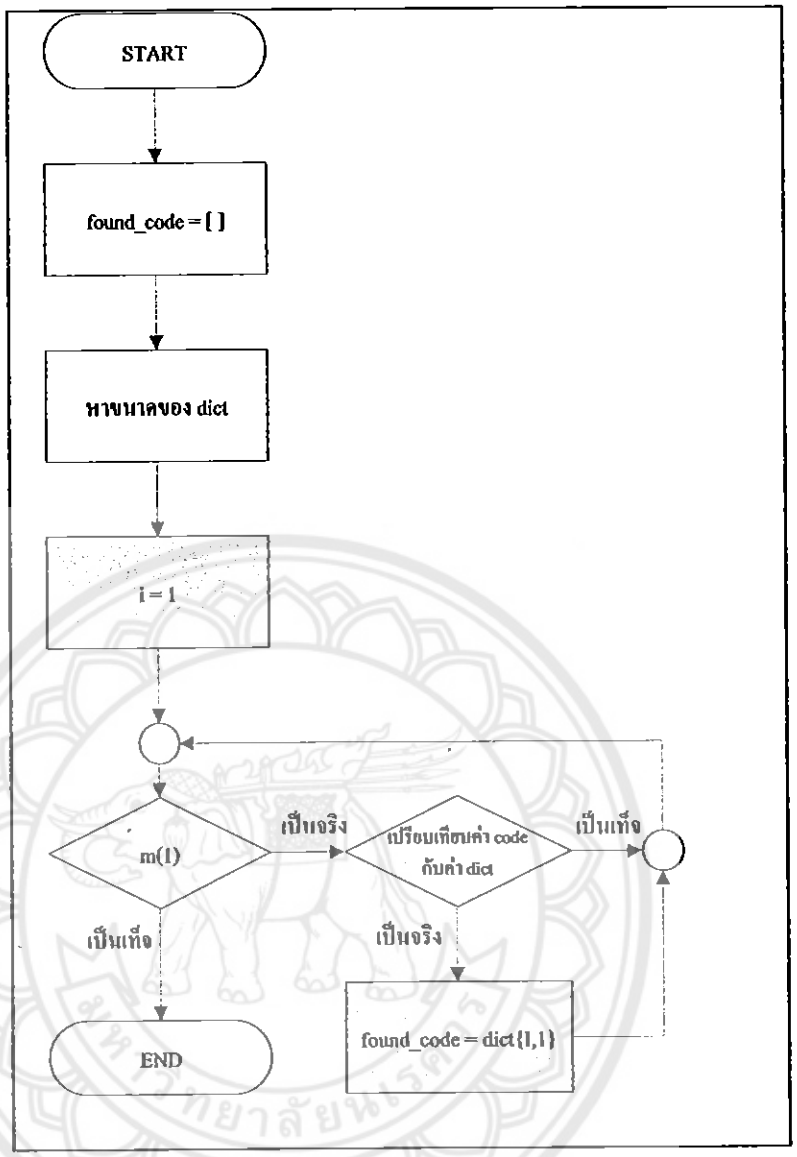


รูปที่ 3.11 Flowchart ฟังก์ชั่นการเข้ารหัสแบบฮัฟฟ์แมน

ฟังก์ชั่น decodehuffman.m เป็นทำหน้าที่ในการแปลงรหัสที่เข้ารหัสไว้แล้ว เทียบกับตารางฮัฟฟ์แมนและจะได้ค่าที่ถอดรหัสออกมา มีขั้นตอนการทำงานดังรูปที่ 3.12 และรูปที่ 3.13 (สามารถดูฟังก์ชั่นได้จากภาคผนวก)



รูปที่ 3.12 Flowchart ฟังก์ชั่นการถอดรหัสแบบฮัฟฟ์แมน

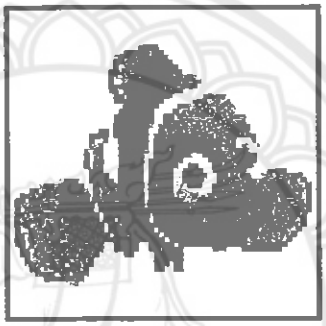


รูปที่ 3.13 Flowchart ฟังก์ชั่นการถอดรหัสแบบฮัฟฟ์แมน (ต่อ)

บทที่ 4
ผลการทดลองและผลการวิเคราะห์

4.1 ผลการทดลอง

4.4.1 การทดลองการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียวโดยที่ไม่มีการกำหนดขอบเขตที่สนใจ การทดลองจะทำการทดลองกับภาพต้นฉบับ 3 ภาพ โดยภาพที่นำมาทดลองจะมีขนาด 64x64 พิกเซล ทั้ง 3 ภาพ คือ ภาพ nemo ขนาด 5,176 ไบต์ดังแสดงไว้ในรูปที่ 4.1, ภาพ eye ขนาด 5,172 ไบต์ดังแสดงไว้ในรูปที่ 4.2 และภาพ lena ขนาด 5,176 ไบต์ดังแสดงไว้ในรูปที่ 4.3



รูปที่ 4.1 แสดงภาพต้นฉบับ nemo ขนาด 64x64 พิกเซล

รูปที่ 4.2 แสดงภาพต้นฉบับ eye ขนาด 64x64 พิกเซล

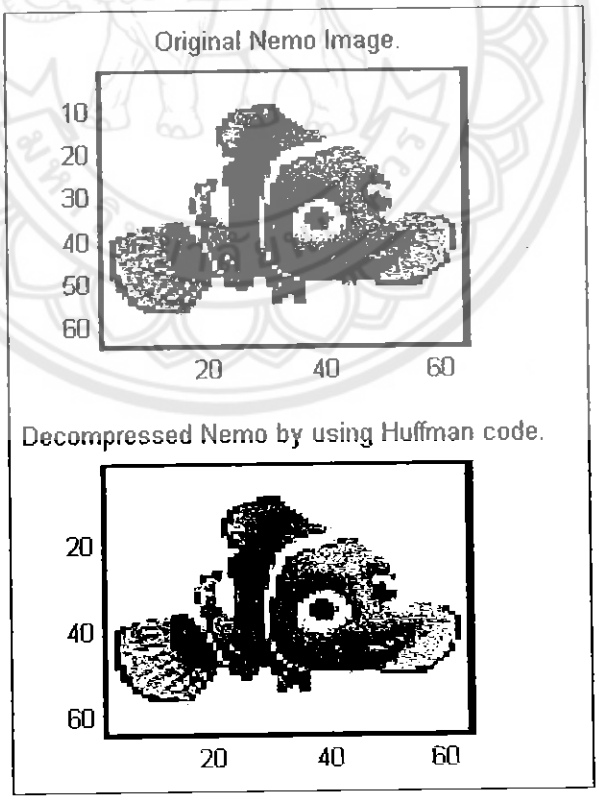
รูปที่ 4.3 แสดงภาพต้นฉบับ lena ขนาด 64x64 พิกเซล

จากผลที่ได้ของการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียว สามารถเขียนเป็นตารางที่ 4.1 และดังรูปที่ 4.4 ,รูปที่ 4.5 และรูปที่ 4.6 ดังนี้

ตารางที่ 4.1 ผลที่ได้จากการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียว

ภาพที่ทดสอบ	CR	SNR (dB)	PSNR (dB)
ภาพ nemo ขนาด 64x64 พิกเซล	1.3228	∞	∞
ภาพ eye ขนาด 64x64 พิกเซล	1.1068	∞	∞
ภาพ lena ขนาด 64x64 พิกเซล	1.1750	∞	∞

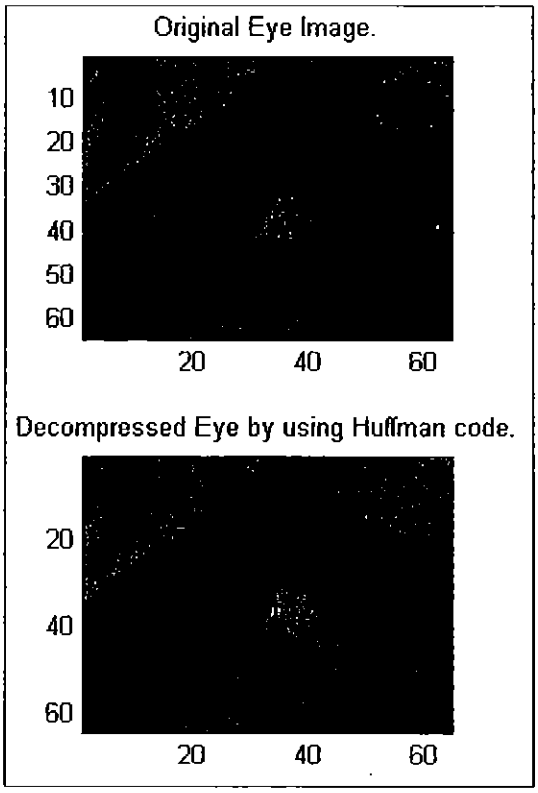
รูปที่ 4.4 (Original Nemo Image) แสดงภาพ nemo ก่อนทำการบีบอัดข้อมูลภาพซึ่งมีขนาดไฟล์ที่จัดเก็บ 5,176 ไบต์ และรูปที่ 4.4 (Decompressed Nemo by using Huffman code) แสดงภาพ nemo หลังทำการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมน ซึ่งมีขนาดไฟล์ 3,913 ไบต์ และได้ค่าอัตราส่วนการบีบอัดข้อมูลเท่ากับ 1.3228 และได้ค่า SNR และ PSNR เข้าสู่เลขอนันต์ แสดงให้เห็นว่าภาพที่ถูกบีบอัดแล้วมีคุณภาพใกล้เคียงกับภาพที่ยังไม่ได้ทำการบีบอัด



รูปที่ 4.4 แสดงภาพ nemo ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน

(CR = 1.3228 , SNR = ∞ และ PSNR = ∞)

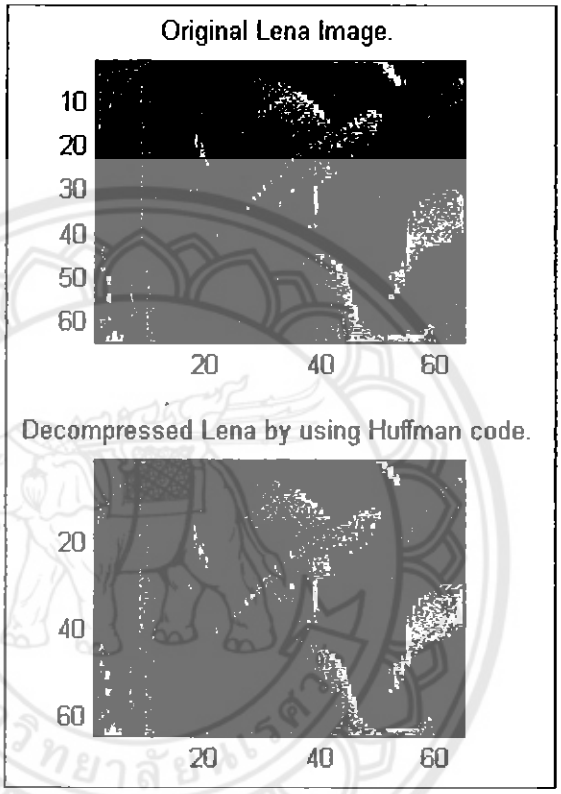
รูปที่ 4.5 (Original Eye Image) แสดงภาพ eye ก่อนทำการบีบอัดข้อมูลภาพซึ่งมีขนาดไฟล์ที่จัดเก็บ 5,172 ไบต์ และรูปที่ 4.4 (Decompressed Eye by using Huffman code) แสดงภาพ eyeหลังทำการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมน ซึ่งมีขนาดไฟล์ 4,673 ไบต์ และได้ค่าอัตราส่วนการบีบอัดข้อมูลเท่ากับ 1.068 และได้ค่า SNR และ PSNR เข้าสู่เลขอนันต์ แสดงให้เห็นว่าภาพที่ถูกบีบอัดแล้วมีคุณภาพใกล้เคียงกับภาพที่ยังไม่ได้ทำการบีบอัด



รูปที่ 4.5 แสดงภาพ eye ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน

(ค่า CR = 1.1068 , SNR = ∞ และ PSNR = ∞)

รูปที่ 4.6 (Original Lena Image) แสดงภาพ eye ก่อนทำการบีบอัดข้อมูลภาพซึ่งมีขนาดไฟล์ที่จัดเก็บ 5,176 ไบต์ และรูปที่ 4.4 (Decompressed Lena by using Huffman code) แสดงภาพ eye หลังทำการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมน ซึ่งมีขนาดไฟล์ 4,405 ไบต์ และได้ค่าอัตราส่วนการบีบอัดข้อมูลเท่ากับ 1.1750 และได้ค่า SNR และ PSNR เข้าสู่เลขอนันต์ แสดงให้เห็นว่าภาพที่ถูกบีบอัดแล้วมีคุณภาพใกล้เคียงกับภาพที่ยังไม่ได้ทำการบีบอัด



รูปที่ 4.6 แสดงภาพ lena ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยฮัฟฟ์แมน
(ค่า CR = 1.1750 , SNR = ∞ และ PSNR = ∞)

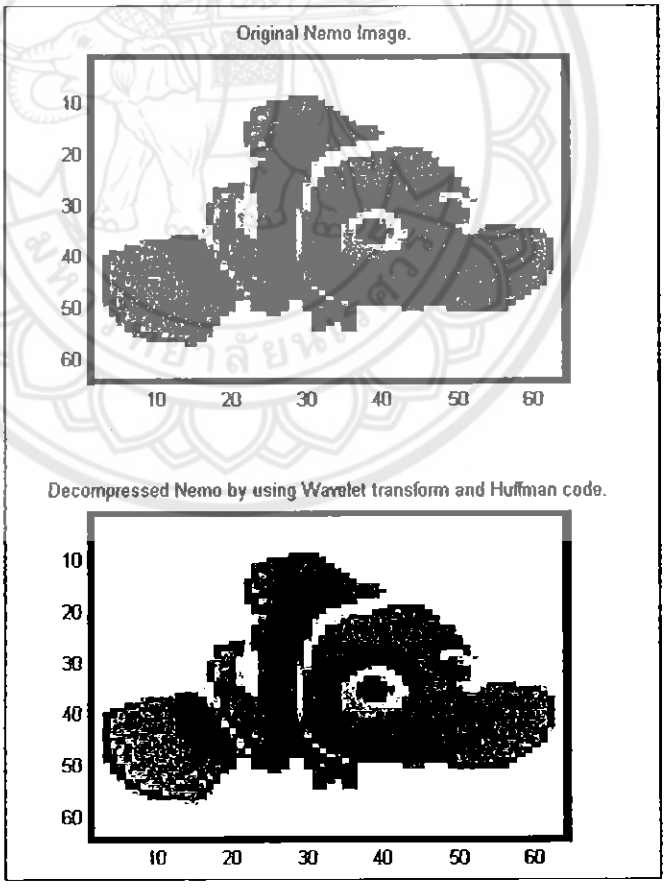
4.4.2 การทดลองการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมนโดยที่ไม่มีการกำหนดขอบเขตที่สนใจ การทดลองจะทำการทดลองกับภาพต้นฉบับ 3 ภาพ โดยภาพที่นำมาทดลองจะมีขนาด 64x64 พิกเซล ทั้ง 3 ภาพ คือ ภาพ nemo ขนาด 5,176 ไบต์ดังแสดงไว้ในรูปที่ 4.1, ภาพ eye ขนาด 5,172 ไบต์ดังแสดงไว้ในรูปที่ 4.2 และภาพ lena ขนาด 5,176 ไบต์ดังแสดงไว้ในรูปที่ 4.3

จากผลที่ได้ของการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมน สามารถเขียนเป็นตารางที่ 4.2 และดังรูปที่ 4.7 ,รูปที่ 4.8 และรูปที่ 4.9 ดังนี้

ตารางที่ 4.2 ผลที่ได้จากการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมน

ภาพทดสอบ	CR	SNR (dB)	PSNR (dB)
ภาพ nemo ขนาด 64x64 พิกเซล	1.9517	68.1660	34.2702
ภาพ eye ขนาด 64x64 พิกเซล	2.3276	316.4255	282.4491
ภาพ lena ขนาด 64x64 พิกเซล	1.4531	57.5791	24.1670

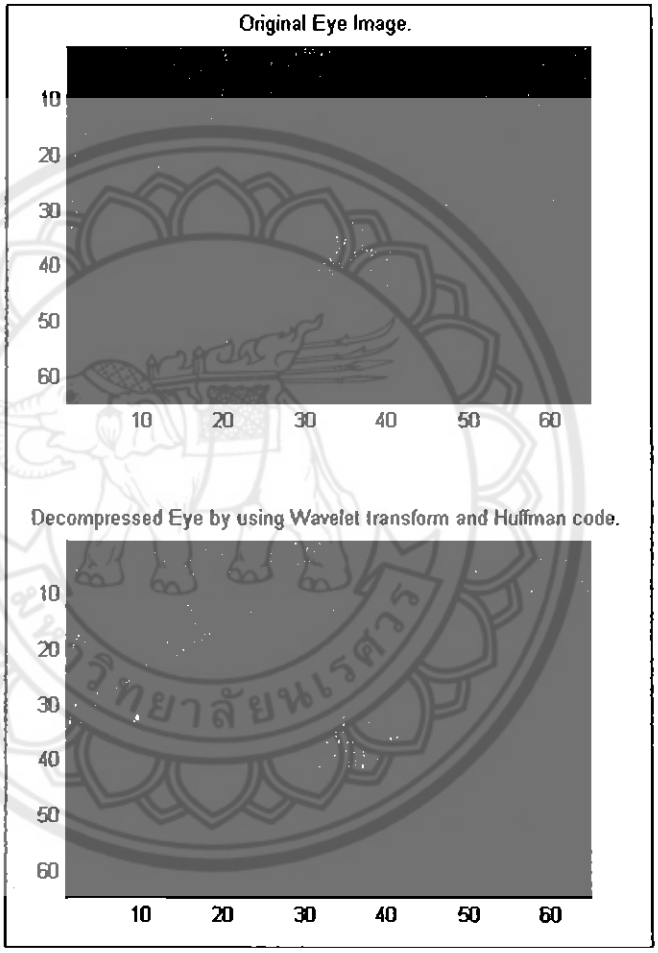
รูปที่ 4.6 (Original Nemo Image) แสดงภาพ eye ก่อนทำการบีบอัดข้อมูลภาพซึ่งมีขนาดไฟล์ที่จัดเก็บ 5,176 ไบต์ และรูปที่ 4.4 (Decompressed Nemo by using Wavelet transform and Huffman code) แสดงภาพ eye หลังทำการบีบอัดข้อมูลภาพโดยใช้การแปลงเวฟเล็ตและฮัฟฟ์แมน ซึ่งมีขนาดไฟล์ 2,652 ไบต์ และได้ค่าอัตราส่วนการบีบอัดข้อมูลเท่ากับ 1.9517 และได้ค่า SNR และ PSNR เข้าสู่เลขอนันต์ แสดงให้เห็นว่าภาพที่ถูกบีบอัดแล้วมีคุณภาพใกล้เคียงกับภาพที่ยังไม่ได้ทำการบีบอัด



รูปที่ 4.7 แสดงภาพ nemo ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยเวฟเล็ตและฮัฟฟ์แมน

(ค่า CR = 1.9517 , SNR = 68.1660 และ PSNR = 34.2702)

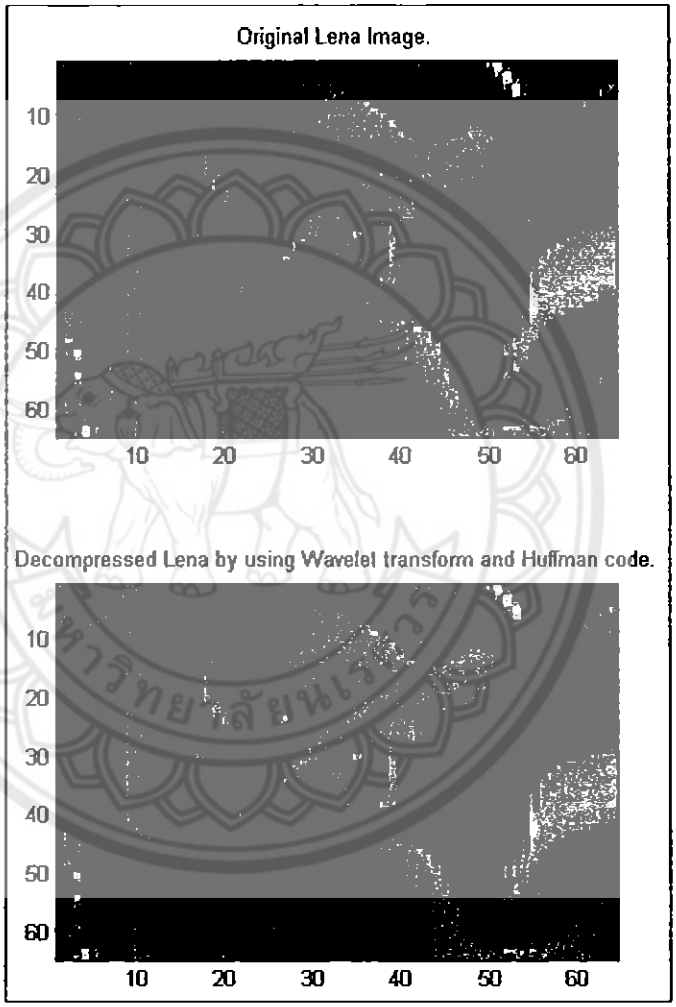
รูปที่ 4.6 (Original Eye Image) แสดงภาพ eye ก่อนทำการบีบอัดข้อมูลภาพซึ่งมีขนาดไฟล์ที่จัดเก็บ 5,172 ไบต์ และรูปที่ 4.4 (Decompressed Eye by using Wavelet transform and Huffman code) แสดงภาพ eye หลังทำการบีบอัดข้อมูลภาพโดยใช้การแปลงเวฟเล็ตและฮัฟฟ์แมน ซึ่งมีขนาดไฟล์ 2,222 ไบต์ และได้ค่าอัตราส่วนการบีบอัดข้อมูลเท่ากับ 2.3276 และได้ค่า SNR และ PSNR เข้าสู่เลขอนันต์ แสดงให้เห็นว่าภาพที่ถูกบีบอัดแล้วมีคุณภาพใกล้เคียงกับภาพที่ยังไม่ได้ทำการบีบอัด



รูปที่ 4.8 แสดงภาพ eye ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยเวฟเล็ตและฮัฟฟ์แมน

(ค่า CR = 2.3276 , SNR = 316.4255 และ PSNR = 282.4492)

รูปที่ 4.6 (Original Lena Image) แสดงภาพ eye ก่อนทำการบีบอัดข้อมูลภาพซึ่งมีขนาดไฟล์ที่จัดเก็บ 5,176 ไบต์ และรูปที่ 4.4 (Decompressed Lena by using Wavelet transform and Huffman code) แสดงภาพ lena หลังทำการบีบอัดข้อมูลภาพโดยใช้การแปลงเวฟเล็ตและฮัฟฟ์แมน ซึ่งมีขนาดไฟล์ 3,562 ไบต์ และได้ค่าอัตราส่วนการบีบอัดข้อมูลเท่ากับ 1.4531 และได้ค่า SNR และ PSNR เข้าสู่เลขอนันต์ แสดงให้เห็นว่าภาพที่ถูกบีบอัดแล้วมีคุณภาพใกล้เคียงกับภาพที่ยังไม่ได้ทำการบีบอัด

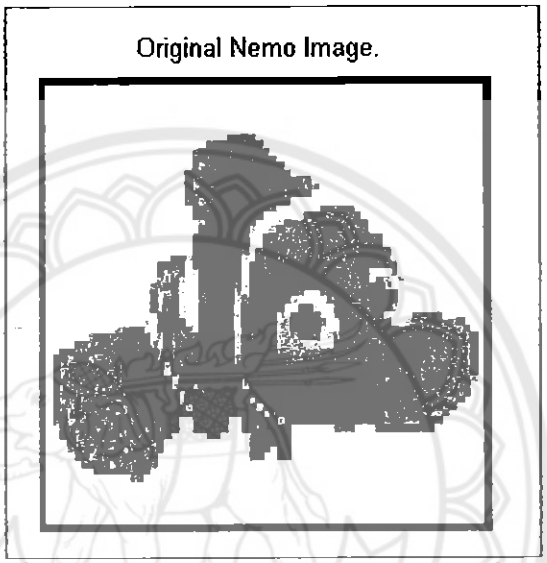


รูปที่ 4.9 แสดงภาพ lena ก่อนการบีบอัดและหลังการบีบอัดข้อมูลภาพด้วยเวฟเล็ตและฮัฟฟ์แมน

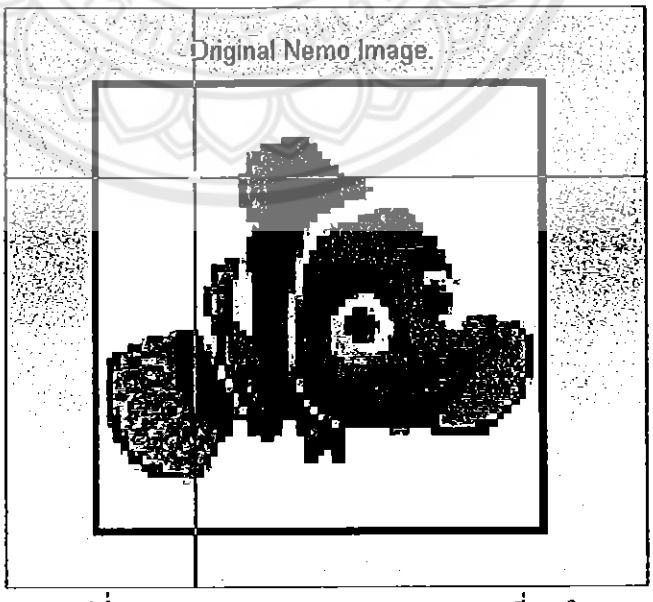
(ค่า CR = 1.4531 , SNR = 57.5791 และ PSNR = 24.1670)

4.4.3 โปรแกรมการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต

การทดลองได้นำภาพ nemo ขนาด 64x64 พิกเซล (รูปที่ 4.1) มาทำการทดลองโดยมีการกำหนดขอบเขตที่สนใจ (ROI) ก่อนที่จะมีการบีบอัดข้อมูลภาพ และการบีบอัดข้อมูลภาพจะเป็นไปตามวิธีการของการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ตและการแบ่งส่วนหลัง ดังรูปที่ 4.10 , รูปที่ 4.11 , รูปที่ 4.12 และรูปที่ 4.13 ตามลำดับวิธีการของการบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต

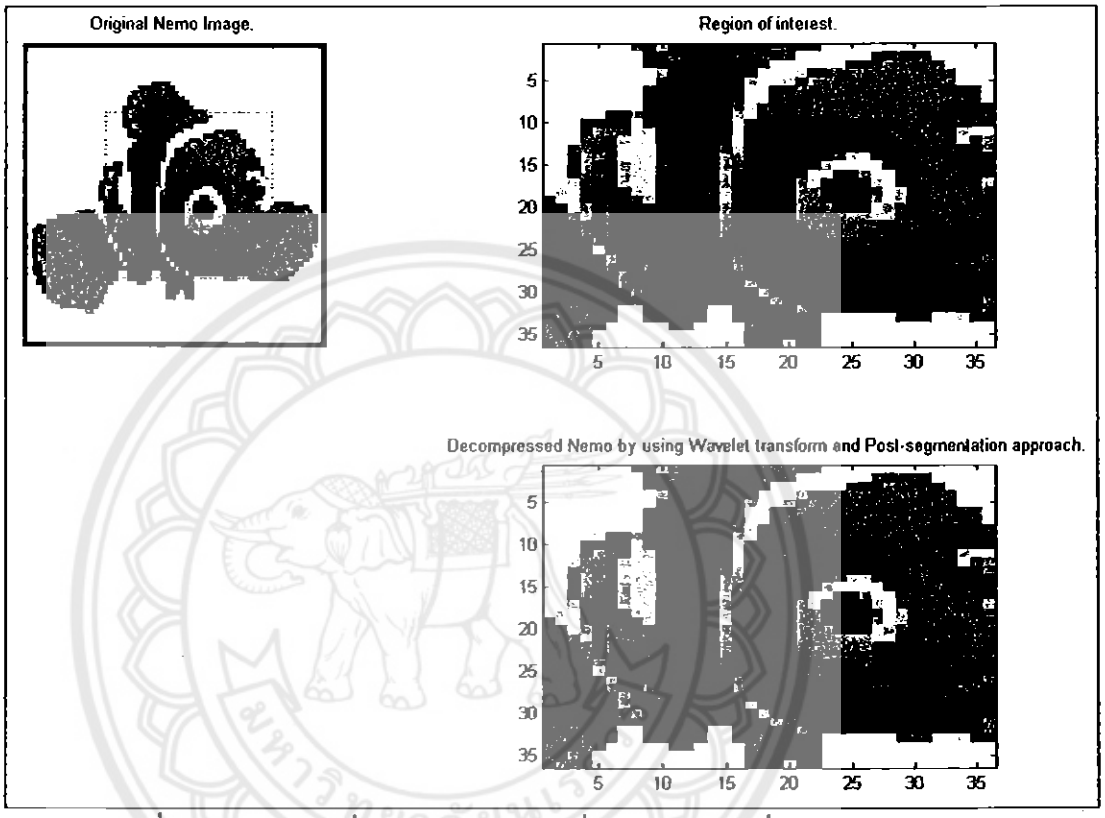


รูปที่ 4.10 แสดง ภาพที่รอการกำหนดขอบเขตที่สนใจ



รูปที่ 4.11 แสดงภาพการกำหนดขอบเขตที่สนใจ

รูปที่ 4.12 (Original Nemo Image) แสดงภาพต้นฉบับที่มีเส้นกำหนดขอบเขตที่สนใจ รูปที่ 4.12(Region of interest) แสดงภาพที่อยู่ภายในขอบเขตที่สนใจ และรูปที่4.12(Decompressed by using Wavelet transform and Post-segmentation approach) แสดงภาพที่อยู่ภายในขอบเขตที่สนใจถูกบีบอัดข้อมูลภาพแล้ว



รูปที่ 4.12 แสดงภาพที่ถูกกำหนดขอบเขตที่สนใจและภาพที่ถูกบีบอัดข้อมูลแล้ว

รูปที่ 4.13 จะแสดงให้เห็นถึงผลการที่ได้จากการรันโปรแกรมโดยจะได้ค่าตามรูปที่ 4.13 ค่า OP คือ ค่าของขนาดภาพต้นฉบับที่ยังไม่ถูกกำหนดขอบเขตที่สนใจ (รูปที่ 4.10) ค่า AD คือ ค่าของขนาดภาพที่ถูกกำหนดขอบเขตที่สนใจ (Region of interestในรูปที่ 4.12) ค่า BD คือ ค่าของขนาดไฟล์ที่ถูกบีบอัดข้อมูลภาพแล้วจะเก็บในไฟล์ medical.kj ค่า SNR เป็นค่าความผิดพลาดของการบีบอัดข้อมูล (ตามสมการที่ 2.14 ในบทที่ 2) และค่า PSNR เป็นค่าความผิดพลาดของการบีบอัดข้อมูล (ตามสมการที่ 2.15 ในบทที่ 2)

หากนำค่าของ $\frac{AD}{BD}$ จะได้ค่าของอัตราส่วนของการบีบอัดข้อมูลภาพ (ตามสมการที่ 2.13 ในบทที่ 2)

```
>> testmed

OP =

      name: 'nemo6464.bmp'
      date: '24-Mar-2005 10:14:54'
     bytes: 5176
     isdir: 0

AD =

      name: 'roi_img.bin'
      date: '13-May-2005 00:34:43'
     bytes: 1296
     isdir: 0

BD =

      name: 'medical.kj'
      date: '13-May-2005 00:34:50'
     bytes: 1188
     isdir: 0


SNR =

   61.0195


PSNR =

   32.8943
```

รูปที่ 4.13 แสดงผลของขนาดและ SNR , PSNR

4.2 ผลการวิเคราะห์

จากผลการทดลองเราจะเห็นว่าค่าในตารางที่ 4.1 นั้น ค่าของ SNR และ PSNR เป็น ∞ ทั้ง 3 ภาพ แสดงว่าการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนนั้น มีการผิดพลาดน้อยมากและเมื่อดูที่รูปที่ 4.5, รูปที่ 4.7 และรูปที่ 4.9 จะไม่เห็นความแตกต่างระหว่างภาพต้นฉบับและภาพที่ถูกบีบอัดข้อมูลภาพ

ส่วนการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมนนั้นค่าที่ได้ในตารางที่ 4.2 คือ ค่า SNR และ PSNR เป็นค่าที่มากแต่ไม่ถึง ∞ ทั้งนี้เพราะการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตนั้นจะมีการสูญเสียคุณภาพของภาพไปบ้าง เมื่อเทียบกับการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียว และเมื่อไปดูที่รูปที่ 4.11 , รูปที่4.13 และรูปที่ 4.15 นั้นเราจะดูไม่ออกว่ามีการสูญเสียคุณภาพไปเลยหรือกล่าวคือค่าของ SNR และ PSNR อยู่ในช่วงที่เรารับได้ในเรื่องของคุณภาพของภาพที่ถูกบีบอัดข้อมูลภาพแล้ว

จากตารางที่ 4.2 นั้นจะเห็นว่าค่า SNR และ PSNR ในภาพ eye มีค่าที่สูงกว่าอีก 2 ภาพมาก ทั้งนี้เนื่องมาจากจำนวนความถี่ของข้อมูลภาพมีมากกว่า ทำให้ในการบีบอัดข้อมูลจะได้ขนาดที่เล็กกว่า และยังมีผลทำให้มีการสูญเสียคุณภาพที่น้อยกว่าด้วย

การบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนนั้น ค่าอัตราส่วนการบีบอัดข้อมูลภาพ (Compression Ratio) ในตารางที่ 4.1 ของภาพทั้งสาม จะมีอัตราส่วนที่ไม่ห่างกันและมีค่าอยู่ประมาณ 1-1.3 เท่า และเมื่อเทียบกับค่าอัตราส่วนของการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมน ในตารางที่ 4.2 นั้นจะเห็นว่าค่าในตารางที่ 4.2 จะอยู่ประมาณ 1.4-2 เท่า ซึ่งมีค่ามากกว่า แสดงว่าการบีบอัดข้อมูล ภาพแบบฮัฟฟ์แมนอย่างเดียว มีการบีบอัดข้อมูลภาพที่น้อยกว่าการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมนที่มีการบีบอัดข้อมูลภาพได้มากกว่า

ดังนั้นการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมน จะมีการบีบอัดข้อมูลที่มากและคุณภาพของภาพก็อยู่ในช่วงที่รับได้หรือกล่าวคือมีการสูญเสียที่น้อย ทำให้มองด้วยตาไม่ออกว่ามีการสูญเสียคุณภาพ

การบีบอัดข้อมูลภาพโดยวิธีการแปลงเวฟเล็ต จึงนำวิธีการของการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมนมาใช้ แต่ก่อนทำการบีบอัดนั้นจะต้องมีการกำหนดขอบเขตที่สนใจก่อน ซึ่งผลที่ได้จะเป็นไปตามรูปที่ 4.10 , รูปที่ 4.11 ,รูปที่ 4.12 และรูปที่ 4.13 ตามลำดับขั้นตอนการทำงานของโปรแกรม

ค่าของอัตราส่วนการบีบอัดข้อมูลภาพนั้นมีค่า 1.114 (รูปที่ 4.13) ค่าของ SNR , PSNR ก็อยู่ในช่วงที่รับได้ เมื่อดูที่รูปที่ 4.12

บทที่ 5
บทสรุป

5.1 สรุปผลการทดลอง

จากบทที่ 3 และบทที่ 4 ทำให้ทราบว่าการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียวจะมีการบีบอัดข้อมูลภาพน้อยกว่าการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมน โดยดูได้จากค่าอัตราส่วนการบีบอัดข้อมูล (Compression Ratio)

นอกจากนี้การบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมนนั้น ค่า SNR และ PSNR ที่ได้อยู่ในช่วงที่รับได้ เมื่อดูจากภาพที่ถูกบีบอัดแล้วจะไม่เห็นถึงความแตกต่างด้วยสายตา

การบีบอัดข้อมูลภาพโดยวิธีการการแปลงเวฟเล็ต จึงเลือกวิธีการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมน เพราะมีอัตราส่วนการบีบอัดข้อมูลที่มากกว่าการบีบอัดข้อมูลภาพแบบฮัฟฟ์แมนอย่างเดียว และมีค่า SNR และ PSNR ที่อยู่ในช่วงที่รับได้

ในการกำหนดขอบเขตที่สนใจ (Region of interest : ROI) เราจะใช้วิธีการจุดพิกเซลในภาพต้นฉบับจำนวน 2 จุด โดยจุดพิกเซลที่กำหนดจะเป็นจุดทแยงกันและกันแล้ว จึงนำมาคำนวณหาค่าขอบเขตที่สนใจ

เมื่อได้ขอบเขตที่สนใจแล้ว จึงนำข้อมูลภาพที่อยู่ในขอบเขตที่สนใจ มาเข้าขบวนการบีบอัดข้อมูลภาพโดยการแปลงเวฟเล็ตและฮัฟฟ์แมนตามลำดับ

ตารางที่ 5.1 ตารางเปรียบเทียบค่า SNR และ PSNR ของ Huffman และ Wavelet & Huffman

ภาพที่ทดสอบ	CR		SNR (dB)		PSNR (dB)	
	Huffman	Wavelet & Huffman	Huffman	Wavelet & Huffman	Huffman	Wavelet & Huffman
ภาพ nemo ขนาด 64x64 พิกเซล	1.3228	1.9517	∞	68.166	∞	34.2702
ภาพ eye ขนาด 64x64 พิกเซล	1.1068	2.3276	∞	316.4255	∞	282.4491
ภาพ lena ขนาด 64x64 พิกเซล	1.1750	1.4531	∞	57.5791	∞	24.167

5.2 ปัญหาในการทดลองและแนวทางแก้ไข

ปัญหา	แนวทางการแก้ไขปัญหา
1. การกำหนดขอบเขตที่สนใจ	กำหนดจุดพิเซล 2 จุด โดยกำหนดจุดพิเซลให้ทแยงกัน และนำค่ามาคำนวณหาระยะ เพื่อที่จะกำหนดขอบเขตที่สนใจ
2. การเข้ารหัสแบบฮัฟฟ์แมน (Encode huffman)	เก็บค่าตารางที่ได้จากกำหนดตัวเลขฐานสิบ มาเป็นเลขไบนารี่ โดยใช้วิธีการของฮัฟฟ์แมนแล้วจึงนำมาเปรียบเทียบกับข้อมูลภาพที่ละตัว เพื่อที่จะแปลงจากเลขฐานสิบเป็นเลขไบนารี่
3. การถอดรหัสแบบฮัฟฟ์แมน (Decode huffman)	นำค่าตารางมาเปรียบเทียบกับเลขไบนารี่ ที่ได้จากการเข้ารหัสแบบฮัฟฟ์แมน แล้วจึงแปลงจากเลขไบนารี่มาเป็นเลขฐานสิบ
4. เวลาในการรันโปรแกรม	เนื่องจากการเข้ารหัส และการถอดแบบฮัฟฟ์แมนนั้นจะทำทีละตัว ทำให้ใช้เวลานานและถ้าเป็นภาพที่มีขนาดใหญ่จะยิ่งใช้เวลานาน ดังนั้นจึงมีการกำหนดขนาดภาพให้เล็กลง โดยกำหนดที่ขนาด 64x64 เพื่อที่จะทำให้เวลารันโปรแกรมน้อยลง

5.3 แนวทางในการพัฒนาโปรแกรมต่อไป

1. ควรเขียนโปรแกรมที่ใช้เวลาน้อยลง โดยที่ขนาดภาพมีขนาดใหญ่
2. ควรเขียนโปรแกรมที่มีอัตราส่วนการบีบอัดข้อมูลภาพที่มากกว่านี้

เอกสารอ้างอิง

[1] รศ.ดร.มนัส สังวรศิลป์ . **คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์** .กรุงเทพมหานคร : อินโฟเพรส, 2543

[2] A. S. Lewis and G. Knowies, **Image Compression Using the 2-D Wavelet Transform,** IEEE Trans Image Processing, Vol. 1, no. 2, 1992

[3] M. A. Sid-Ahmed, **Image Processing Theory, Algorithms, & Architectures.** McGraw-hill, Inc., 1995

[4] Randy K. Young, **Wavelet Theory and its Applications**, Kluwer Academic Publisherd, 1993

[5] Eric J. Stollnitz, Tony D. DeRose and David H. Selesin, **Wavelets for Computer Graphics: A Primer, Part1**, IEEE Computer Graphics and Applications, May 1995

ภาคผนวก
โปรแกรมการบีบอัดภาพโดยวิธีการแปลงเวฟเล็ต
ฟังก์ชั่นที่ใช้ในโปรแกรม

โปรแกรมการบีบอัดภาพโดยวิธีการแปลงเวฟเล็ต

```
%=========================================================================
%
%  การบีบอัดภาพโดยวิธีการแปลงเวฟเล็ต
%  Image Compresstion Using by Wavelet transform
%
%  File name : testmed.m
%  By : Mr.Chakkrapong Suami          ID 44370096
%       Mr.Sittichat Therdsittikul    ID 44370492
%
%=========================================================================

clear all
close all

% Load files(Picture) & Find ROI(Region of Interest)
img  = imread('nemo6464.bmp','bmp');          การรับค่าจากไฟล์ภาพ
roi_img = roi(img);                           การค่าจากไฟล์ภาพไปเข้าฟังก์ชั่น roi
figure(1)

% Save Files
fid = fopen('roi_img.bin','wb');              การเก็บไฟล์ที่อยู่ภายในขอบเขตที่สนใจ
    fwrite(fid,roi_img,'integer*1');
    fclose(fid);

%=========================================================================
% Find Wavelet Transform
[nrow ncol] = size(roi_img);                  การหาขนาดของแนวนอนและแนวตั้ง
CDl = [];
```

```
% Thing on Row Wevelet Transform of Image
  for ii=1:1:nrow                                    การแปลงเวฟเล็ตในแนวนอน
    xn = roi_img(ii,:);
    [cn,dn] = dwt(xn,'db1');
    C = [cn dn];
    CDI = [CDI; C];
  end

  % Thing on Column Wevelet Transform of Image
  CD = [];                                           การแปลงเวฟเล็ตในแนวตั้ง
  for jj=1:1:ncol
    xn = CDI(:,jj);
    [cn,dn] = dwt(xn,'db1');
    C = [cn;dn];
    CD = [CD C];
  end
% ================================================================
% Find Huffman code
% Table of Huffman code
[M N] = size(roi_img);
input = CD;
new = reshape(input,1,M*N);
sig = round(new)+512;
histo = medhisto(sig);                    การหาค่าความถี่โดยการเรียกใช้ฟังก์ชั่น medhisto
[I J V] = find(histo);
p = V/sum(V);                             การหาค่าความน่าจะเป็น
symbols = J;
actualsig = sig;
dict = huffmantable(symbols,p);           การหาค่าตารางของฮัฟฟ์แมน
```

```
% Encode of Huffman code
input = actualsig;
code = encodehuffman(input,dict);    การเข้ารหัสแบบฮัฟฟ์แมน

% ------------------------------------------------------------------
% Save Files
fid = fopen('medical.kj','wb');      การเปิดไฟล์ที่ชื่อ medical.kj เป็นไฟล์ไบนารี่
fwrite(fid,code,'ubit1');            นำค่า code มาเก็บลงในไฟล์
fclose(fid);                         ปิดไฟล์
[a b] = size(code);

% Read in the same file
fid  = fopen('medical.kj','rb');     การเปิดไฟล์ที่ชื่อ medical.kj เป็นไฟล์ไบนารี่
data = fread(fid,[a b],'ubit1');     นำค่าที่เก็บอยู่ในไฟล์ medical.kj มาเป็นค่า data
fclose(fid);                         ปิดไฟล์

% Show size files
OP=dir('nemo6464.bmp')               ขนาดของภาพต้นฉบับ
AD=dir('roi_img.bin')                ขนาดของภาพที่อยู่ในขอบเขตที่สนใจ
BD=dir('medical.kj')                 ขนาดของไฟล์ไบนารี่ที่ผ่านการบีบอัดข้อมูลภาพแล้ว

% ------------------------------------------------------------------
% Decode of Huffman code
deco = decodehuffman(data, dict);    การถอดรหัสแบบฮัฟฟ์แมน
desig = reshape(deco,M,N);
newsig = desig-512;

% ==================================================================
```

```
LL=newsig(1:nrow,1:ncol);                    การลดสัญญาณรบกวน
CDTH = zeros(size(CD));
CDTH(find(abs(CD) <= 10000000000000000000)) = 0;
CDTH(1:nrow,1:ncol)=LL;

% ═══════════════════════════════════════════════════════════════

  % Thing on Column Invert Wevelet Transform of the image
  X = [];                                    การแปลงกลับของเวฟเล็ตในแนวตั้ง
  for jj=1:1:ncol
     cd = CDTH(:,jj);
     cn = cd(1:nrow/2);
     dn = cd((nrow/2)+1:end);
     xn = idwt(cn,dn,'db1');
     X = [X xn];
  end

  % Thing on Row Invert Wevelet Transform of the image
  S = [];                                    การแปลงกลับของเวฟเล็ตในแนวนอน
  for ii=1:1:nrow
     cd = X(ii,:);
     cn = cd(1:ncol/2);
     dn = cd((ncol/2)+1:end);
     xn = idwt(cn,dn,'db1');
     S = [S; xn];
  end

% ---------------------------------------------------------------
```

```
% Check Error
nsig = roi_img;                          กำหนดให้ nsig เป็นภาพที่อยู่ในขอบเขตที่สนใจ
desig = S;                               กำหนดให้ desig เป็นภาพที่ผ่านการบีบอัดข้อมูลภาพ
ER = (nsig-desig);
ERR = ER^2;
nesig = nsig^2;
Error = (sum(ERR(:)))/M*N;
SNR = 10*log10((sum(nesig(:))/M*N)/Error)      หาค่า SNR
PSNR = 10*log10(max(nesig(:))/Error)           หาค่า PSNR

% ------------------------------------------------------------------
% Display(Show)
subplot(2,2,4),image(S),colormap(gray(256)),title('Image Compression using by Wavelet
transform and Post-segmentation approach.')
```

```
%=========================================================
% Wavelet & Huffman Code
%
%  File name : test3.m
%  By : Mr.Chakkrapong Suami      ID 44370096
%       Mr.Sittichat Therdsittikul  ID 44370492
%
%=========================================================
clear all
close all

% Load files(Picture)
img  = imread('nemo6464.bmp','bmp');              การรับค่าจากไฟล์ภาพ
roi_img = double(img);
[M N] = size(roi_img);

% show picture
figure(1)
subplot(2,2,1),image(roi_img),colormap(gray(256)),title('Image Original.')

% Find Wavelet Transform
[nrow ncol] = size(roi_img);                      การหาขนาดของแนวนอนและแนวตั้ง
CDl = [];

  % Thing on Row Wevelet Transform of Image
  for ii=1:1:nrow                                 การแปลงเวฟเล็ตในแนวนอน
    xn = roi_img(ii,:);
    [cn,dn] = dwt(xn,'db1');
    C = [cn dn];
    CDl = [CDl; C];
  end
```

```
% Thing on Column Wevelet Transform of Image
CD = [];                                    การแปลงเวฟเล็ตในแนวตั้ง
for jj=1:1:ncol
   xn = CDl(:,jj);
   [cn,dn] = dwt(xn,'db1');
   C = [cn;dn];
   CD = [CD C];
end


% Find Huffman code
% Table of Huffman code
input = CD;
[M N] = size(input);
new = reshape(input,1,M*N);
sig = round(new)+512;
histo = medhisto(sig);                      การหาค่าความถี่โดยการเรียกใช้ฟังก์ชั่น medhisto
[I J V] = find(histo);
p = V/sum(V);                               การหาค่าความน่าจะเป็น
symbols = J;
actualsig = sig;
dict = huffmantable(symbols,p);             การหาค่าตารางของฮัฟฟ์แมน

% Encode of Huffman code
input = actualsig;
code = encodehuffman(input,dict);           การเข้ารหัสแบบฮัฟฟ์แมน

% Save Files
fid = fopen('medical.kj','wb');             การเปิดไฟล์ที่ชื่อ medical.kj เป็นไฟล์ไบนารี่
fwrite(fid,code,'ubit1');                   นำค่า code มาเก็บลงในไฟล์
fclose(fid);                                ปิดไฟล์
[a b] = size(code);
```

```
%read in the same file
fid  = fopen('medical.kj','rb');          การเปิดไฟล์ที่ชื่อ medical.kj เป็นไฟล์ไบนารี่
data = fread(fid,[a b],'ubit1');          นำค่าที่เก็บอยู่ในไฟล์ medical.kj มาเป็นค่า data
fclose(fid);                              ปิดไฟล์

% Show size files
AD=dir('nemo6464.bmp')                    ขนาดของภาพต้นฉบับ
BD=dir('medical.kj')                      ขนาดของไฟล์ไบนารี่ที่ผ่านการบีบอัดข้อมูลภาพแล้ว

% Decode of Huffman code
deco = decodehuffman(data, dict);         การถอดรหัสแบบฮัฟฟ์แมน
dsig = reshape(deco,M,N);
newsig = dsig-512;

LL=newsig(1:nrow,1:ncol);                 การลดสัญญาณรบกวน
CDTH = zeros(size(CD));
CDTH(find(abs(CD) <= 10^20)) = 0;
CDTH(1:nrow,1:ncol)=LL;

  % Thing on Column Invert Wevelet Transform of the image
  X = [];                                 การแปลงกลับของเวฟเล็ตในแนวตั้ง
  for jj=1:1:ncol
    cd = CDTH(:,jj);
    cn = cd(1:nrow/2);
    dn = cd((nrow/2)+1:end);
    xn = idwt(cn,dn,'db1');
    X = [X xn];
  end
```

```
% Thing on Row Invert Wevelet Transform of the image
S = [];                                    การแปลงกลับของเวฟเล็ตในแนวนอน
for ii=1:1:nrow
   cd = X(ii,:);
   cn = cd(1:ncol/2);
   dn = cd((ncol/2)+1:end);
   xn = idwt(cn,dn,'db1');
   S = [S; xn];
end

%  ═══════════════════════════════════════════════════════════════
% Check Error
nsig = roi_img;                            กำหนดให้ nsig เป็นภาพที่อยู่ในขอบเขตที่สนใจ
desig = S;                                 กำหนดให้ desig เป็นภาพที่ผ่านการบีบอัดข้อมูลภาพ
ER = (nsig-desig);
ERR = ER^2;
nesig = nsig^2;
Error = (sum(ERR(:)))/M*N;
SNR = 10*log10((sum(nesig(:))/M*N)/Error)      หาค่า SNR
PSNR = 10*log10(max(nesig(:))/Error)           หาค่า PSNR


%  ═══════════════════════════════════════════════════════════════
% Display(Show)
subplot(2,2,3),image(S),colormap(gray(256)),title('Image Compression using by Wavelet
transform and Huffman code.')
```

```
% ═══════════════════════════════════════════
% Test Huffman code
%
%  File name : hufftest.m
%  By : Mr.Chakkrapong Suami      ID 44370096
%       Mr.Sittichat Therdsittikul   ID 44370492
%
% ═══════════════════════════════════════════

clear all
close all

nsig = double(imread('nemo6464.bmp','bmp'));     การรับค่าจากไฟล์ภาพ
nesig = nsig(:);

[M N] = size(nsig);
input = nesig;
new = reshape(input,1,M*N);
p = (0:1:255)/sum(0:1:255);                      การหาความน่าจะเป็น
symbols = 0:1:255;
actualsig = new;
dict = huffmantable(symbols,p);                  การหาค่าตารางของฮัฟฟ์แมน

% Encode of Huffman code
input = actualsig;
code = encodehuffman(input,dict);                การเข้ารหัสแบบฮัฟฟ์แมน

% Save Files
fid = fopen('medical.kj','wb');                  การเปิดไฟล์ที่ชื่อ medical.kj เป็นไฟล์ไบนารี่
fwrite(fid,code,'ubit1');                        นำค่า code มาเก็บลงในไฟล์
fclose(fid);                                     ปิดไฟล์
[a b] = size(code);
```

```
%read in the same file
fid  = fopen('medical.kj','rb');          การเปิดไฟล์ที่ชื่อ medical.kj เป็นไฟล์ไบนารี่
data = fread(fid,[a b],'ubit1');          นำค่าที่เก็บอยู่ในไฟล์ medical.kj มาเป็นค่า data
fclose(fid);                              ปิดไฟล์

% Show size files
AD=dir('nemo6464.bmp')                    ขนาดของภาพต้นฉบับ
BD=dir('medical.kj')                      ขนาดของไฟล์ไบนารี่ที่ผ่านการบีบอัดข้อมูลภาพแล้ว

% Decode of Huffman code
deco = decodehuffman(data, dict);         การถอดรหัสแบบฮัฟฟ์แมน
desig = reshape(deco,M,N);

% ------------------------------------------------------------
% Check Error
ER = (nsig-desig);                        กำหนดให้ภาพที่ต้นฉบับ - ภาพที่ผ่านการบีบอัดข้อมูลภาพ
ERR = ER^2;
nesig = nsig^2;
Error = (sum(ERR(:)))/M*N;
SNR = 10*log10((sum(nesig(:))/M*N)/Error)     หาค่า SNR
PSNR = 10*log10(max(nesig(:))/Error)          หาค่า PSNR

% ------------------------------------------------------------
% Display(Show)
figure(1)
subplot(2,2,1),image(nsig),colormap(gray(256)),title('Image Original.')
subplot(2,2,3),image(desig),colormap(gray(256)),title('Image Compression using by Huffman
code.')
```

ฟังก์ชั่นที่ใช้ในโปรแกรม

```
function roi_img = roi(img);

% ═══════════════════════════════════════════════════════════
% FUNCTION : roi_img = roi(img);
% This function find ROI(Region of Interest)
% img คือ ภาพต้นฉบับ(Input Picture)
% roi_img คือ ภาพที่ถูกกำหนดอยู่ในขอบเขตของความสนใจ(Region of Interest : ROI)
%
%   File name : roi.m
%   By : Mr.Chakkrapong Suami     ID 44370096
%        Mr.Sittichat Therdsittikul  ID 44370492
%
% ═══════════════════════════════════════════════════════════

% Load files(Picture)
imgk = double(img);                                  รับค่าอินพุท

% Display Show Original Picture
subplot(2,2,1),imshow(img),colormap(gray(256)),title('Image Original.')

% Find ROI
[x1,y1] = ginput(1);   % Find Point 1.              การกำหนดจุดพิกเซลจุดที่ 1
[x2,y2] = ginput(1);   % Find Point 2.              การกำหนดจุดพิกเซลจุดที่ 2

x1 = round(x1);                                      การกำหนดให้เป็นค่าจำนวนเต็ม
x2 = round(x2);
y1 = round(y1);
y2 = round(y2);
```

```
% Find Line
if mod(x1,2)~=0 & mod(x2,2)==0,
   x1=x1+1; x2=x2;
elseif mod(x1,2)~=0 & mod(x2,2)~=0,
   x1=x1; x2=x2;
elseif mod(x1,2)==0 & mod(x2,2)~=0,
   x1=x1; x2=x2+1;
elseif mod(x1,2)==0 & mod(x2,2)==0,
   x1=x1; x2=x2;
end

if mod(y1,2)~=0 & mod(y2,2)==0,
   y1=y1+1; y2=y2;
elseif mod(y1,2)~=0 & mod(y2,2)~=0,
   y1=y1; y2=y2;
elseif mod(y1,2)==0 & mod(y2,2)~=0,
   y1=y1; y2=y2+1;
elseif mod(y1,2)==0 & mod(y2,2)==0,
   y1=y1; y2=y2;
end

if (x1-x2) <= 0,
   line_x = sqrt((x1-x2)^2);
elseif (x1-x2) > 0,
   line_x = x1-x2;
end
```

การกำหนดจุดก่อนทำการคำนวณหาค่าขอบเขตที่สนใจ

```
if line_x <= 30,
   line_x = 32;
elseif line_x <= 64,
   line_x = line_x;
elseif line_x > 64,
   line_x = 64;
end

line_y = line_x;
top   = line(x1+1:1:x1+line_x,y1+line_y);          การกำหนดขอบเขตด้านบน
under = line(x1+1:1:x1+line_x,y1+1);               การกำหนดขอบเขตด้านล่าง
right = line(x1+1,y1+1:1:y1+line_y);               การกำหนดขอบเขตด้านขวา
left  = line(x1+line_x,y1+1:1:y1+line_y);          การกำหนดขอบเขตด้านซ้าย

% Show Picture of ROI
roi_img = imgk(x1+1:1:x1+line_x,y1+1:1:y1+line_y);      การกำหนดขอบเขตที่สนใจ
subplot(2,2,2),image(roi_img),colormap(gray(256)),title('Region of interest.')
```

```
function h = medhisto(xx);

%=============================================================
% HISTO Display histogram of image data.
%  h = medhisto(xx) displays a histogram with
%  xx is input
%
%  File name : medhisto.m
%  By : Mr.Chakkrapong Suami    ID 44370096
%       Mr.Sittichat Therdsittikul  ID 44370492
%
%=============================================================

h = [];

for gl = 1: 1: length(xx)                  การกำหนดค่า gl
  [indx,indy,val] =  find(xx==gl);          การนำค่า xx มาเปรียบเทียบกับค่า gl
  h          =  [h sum(val)];               การนับค่าที่เหมือนกันว่ามีกี่ค่า
end
```

```
function dict = huffmantable(sig,prob);

% ================================================================
% FUNCTION dict = huffmantable(sig,prob);
% This function is Table of huffman code
% sig is input
% prob is Probability distribution
% dict is Table of huffman code
%
%  File name : huffmantable.m
%  By : Mr.Chakkrapong Suami      ID 44370096
%       Mr.Sittichat Therdsittikul  ID 44370492
%
% ================================================================

n_ary = [];
variance = '';
msg=nargchk(2,4, nargin);

if nargin > 2
        n_ary = varargin{1};
end
if nargin == 4
        variance = varargin{2};
end
```

```
if isempty(n_ary)
   n_ary = 2; % default value is binary encryption
end
if ( variance )
        % if variance contains a non-null string do nothing
else
   variance = 'max'; % default is maximum variance Huffman code
end

% Make sure that internally all vectors are represented as column vectors
m = size(sig);
if( m(1) == 1 )
   sig = sig';
end
m = size(prob);
prob = prob(:);

% Make sure that the input symbols are in a cell array format
if ~iscell(sig)
        [m,n] = size(sig);
        sig = mat2cell(sig, ones(1,m), ones(1,n)) ;
end

% Check if all the input symbols are either alphabets or numbers or a
% combination of the two
for i=1:length(sig)
        isalphanumeric(i) = ischar(sig{i}) || isnumeric(sig{i});
end
```

```
% Check if the each symbol in the first input is unique
for i = 1:length(sig)-1
        pilotpoint = sig{i};
        for j = i+1:length(sig)
                if length(pilotpoint) == length(sig(j)) && min(pilotpoint == sig{j})
                        error('comm:huffmandict:RepeatedSymbols', 'Source symbols repeat')
                end
        end
end

% Create tree nodes with the signals and the corresponding probabilities
huff_tree = struct('signal', [], 'probability', [],...
   'child', [], 'code', [], 'origOrder', -1);
for i=1:length( sig )
   huff_tree(i).signal = sig{i};
   huff_tree(i).probability = prob(i);
        huff_tree(i).origOrder = i;
end

% Sort the signal and probability vectors based on ascending order of
% probability
[s, i] = sort(prob);
huff_tree = huff_tree(i);
huff_tree = create_tree(huff_tree, n_ary, variance); % create a Huffman tree
[huff_tree,dict,avglen] = create_dict(huff_tree,{},0, n_ary); % create the codebook
```

```
% The next few lines of code are to sort the dictionary.
% If sorting based on original order then use dict{:,4}.
% If sorting based on the lenght of code, then use  dict{:,3}.
[dictsort,dictsortorder] = sort([dict{:,4}]);
finaldict = {};
for i=1:length(dictsortorder)
   finaldict{i,1} = dict{dictsortorder(i), 1};
   finaldict{i,2} = dict{dictsortorder(i), 2};
end
dict = finaldict;

%-------------------------------------------------------------------
%% Function: huff_tree
% Input: An array of structures to be arranged into a Huffman tree
% Utility: This is a recursive algorithm to create the Huffman Code
%          tree. This is a recursive function
function huff_tree = create_tree(huff_tree, n_ary, variance)

% if the length of huff_tree is 1, it implies there is no more than one
% node in the array of nodes. This is the termination condition for the
% recursive loop
if( length(huff_tree) <= 1)
   return;
end
% Combine the first n_ary (lowest probability) number of nodes under one
% parent node, remove these n_ary nodes from the list of nodes and add
% the new parent node that was just created
temp = struct('signal', [], 'probability', 0, ...
   'child', [], 'code', []);
```

```
for i=1:n_ary
   if( length(huff_tree) == 0), break; end
   temp.probability = temp.probability + huff_tree(1).probability; % for ascending order
   temp.child{i} = huff_tree(1);
         temp.origOrder = -1;
   huff_tree(1) = [];
end
if( strcmpi(variance, 'min') == 1 )
   huff_tree = insertMinVar(huff_tree, temp);
else
   huff_tree = insertMaxVar(huff_tree, temp);
end
% create a Huffman tree from the reduced number of free nodes
huff_tree = create_tree(huff_tree, n_ary, variance);
return;

%-----------------------------------------------------------------
%% This function will insert a node in the sorted list such that the
% resulting list will be sorted (ascending). If there exists node with the
% same probability as the new node, the new node  is placed after these
% same value nodes.
function huff_tree = insertMaxVar(huff_tree, newNode)
sortedOn = [huff_tree.probability];
i = 1;
while i <= length(huff_tree) && ...
      newNode.probability > huff_tree(i).probability
   i = i+1;
end
huff_tree = [huff_tree(1:i-1) newNode huff_tree(i:end)];
```

```
%-------------------------------------------------------------------
%% This function will insert a node in the sorted list such that the
% resulting list will be sorted (ascending). If there exist nodes with the
% same probability as the new node, the new node  is placed before these
% same value nodes.
function huff_tree = insertMinVar(huff_tree, newNode)
sortedOn = [huff_tree.probability];
i = 1;
while i <= length(huff_tree) && ...
      newNode.probability >= huff_tree(i).probability
   i = i+1;
end
huff_tree = [huff_tree(1:i-1) newNode huff_tree(i:end)];


%-------------------------------------------------------------------
%% This function does a pre-order traversal of the tree to create the codes
% for each leaf node. This is a recursive function
function [huff_tree,dict,total_wted_len] = create_dict(huff_tree,dict,total_wted_len, n_ary)
% Check if the current node is a leaf node If it is, then add the signal on
% this node and its corresponding code to the dictionary global n_ary
if( length(huff_tree.child) == 0 )
   dict{end+1,1} = huff_tree.signal;
   dict{end, 2} = huff_tree.code;
   dict{end, 3} = length(huff_tree.code);
         dict{end, 4} = huff_tree.origOrder;
   total_wted_len = total_wted_len + length(huff_tree.code)*huff_tree.probability;
   return;
end
num_childrens = length(huff_tree.child);
```

```
for i = 1:num_childrens
   huff_tree.child{i}.code = [huff_tree(end).code, (num_childrens-i)];
   [huff_tree.child{i}, dict, total_wted_len] = ...
      create_dict(huff_tree.child{i}, dict, total_wted_len, n_ary);
end
```

```
function code = encodehuffman(input,dict);

%=============================================================
% FUNCTION : code = encodehuffman(input,dict);
% This function is encodehuffman
% input is input signal
% dict is table's huffman code
% enco is output's binary
%
%  File name : encodehuffman.m
%  By : Mr.Chakkrapong Suami      ID 44370096
%       Mr.Sittichat Therdsittikul  ID 44370492
%
%=============================================================

y = 0 ;
for i = 1:1:length(input),
   for j = 1:1:length(dict),
      if input(i) == dict{j,1},
         a=dict{j,2};
         dict{j,1};
         y = [a y];
      end
   end
end
code = y(1:1:end-1) ;
```

```
function deco = decodehuffman(comp, dict)

%=========================================================================
% FUNCTION : deco = decodehuffman(comp,dict);
% This function is decodehuffman
% comp is input's binary
% dict is Table's huffman code
% deco is output
%
%  File name : decodehuffman.m
%  By : Mr.Chakkrapong Suami    ID 44370096
%       Mr.Sittichat Therdsittikul  ID 44370492
%
%=========================================================================

msg=nargchk(2,2, nargin);
[m,n] = size(comp);
isSigNonNumeric = max(cellfun('isclass', {dict{:,1}}, 'char') );
deco = {};
i = 1;
while(i <= length(comp))
   tempcode = comp(i);
   found_code = is_a_valid_code(tempcode, dict);
   while(length(found_code) == 0 && i < length(comp))
      i = i+1;
      tempcode = [tempcode, comp(i)];
      found_code = is_a_valid_code(tempcode, dict);
   end
         deco{end+1} = found_code;
   i=i+1;
end
```

```
if( n == 1 )        % if input was a column vector
   deco = deco';    % the decoded output should also be a column vector
end
if ( ~isSigNonNumeric )
          deco = cell2mat(deco);
end
deco = deco(length(deco):-1:1);

%--------------------------------------------------------------------------
function found_code = is_a_valid_code(code, dict)
found_code = [];
m = size(dict);
for i=1:m(1)
   if ( isequal(code, dict{i,2}) )
      found_code = dict{i,1};
      return;
   end
end
```

ประวัติผู้เขียนโครงงาน

ชื่อ นามสกุล จักรพงษ์ เสือมี
เกิดวันที่ 20 มีนาคม 2526
ภูมิลำเนา 234 ม.1 ต.ยางซ้าย
อำเภอเมือง จังหวัดสุโขทัย

ประวัติการศึกษา
- จบการศึกษาชั้นประถมศึกษาจากโรงเรียนอนุบาลสุโขทัย
อำเภอเมือง จังหวัดสุโขทัย
- จบการศึกษาชั้นมัธยมศึกษาจากโรงเรียนสุโขทัยวิทยาคม
อำเภอเมือง จังหวัดสุโขทัย
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรี ชั้นปีที่ 4
สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยนเรศวร

E-Mail : mojojojo_ch@hotmail.com

ชื่อ นามสกุล สิทธิชาติ เทิดสิทธิกุล
เกิดวันที่ 6 กรกฎาคม 2525
ภูมิลำเนา 900/41 ถนนบรมไตรโลกนารถ2
ตำบลในเมือง อำเภอเมือง
จังหวัดพิษณุโลก

ประวัติการศึกษา
- จบการศึกษาชั้นประถมศึกษาจากโรงเรียนจ่าการบุญ
อำเภอเมือง จังหวัดพิษณุโลก
- จบการศึกษาชั้นมัธยมศึกษาจากโรงเรียนพิษณุโลก
พิทยาคม อำเภอเมือง จังหวัดพิษณุโลก
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรี ชั้นปีที่ 4
สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยนเรศวร

E-Mail : t_sittichat@hotmail.com