การลดขนาดข้อมูลภาพโดยใช้เวฟเล็ตทรานฟอร์ม

IMAGE DATA COMPRESSION USING WAVELET TRANSFORM

ชินภัทร นันทจิวากรชัย
CHINNAPAT NANTAJIWAKORNCHAI

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2542

ISBN 974-622-362-3

การลดขนาดข้อมูลภาพโดยใช้ เวฟเล็ตทรานฟอร์ม

IMAGE DATA COMPRESSION USING WAVELET TRANSFORM

ชินภัทร นันทจิวากรชัย

CHINNAPAT NANTAJIWAKORNCHAI

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2542

ISBN 974-622-362-3

IMAGE DATA COMPRESSION USING WAVELET TRANSFORM

CHINNAPAT NANTAJIWAKORNCHAI

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
1999

ISBN 974-622-362-3

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การลดขนาดข้อมูลภาพโดยใช้เวฟเล็ตทรานฟอร์ม

IMAGE DATA COMPRESSION USING WAVELET TRANSFORM

ชื่อนักศึกษา นายชินภัทร นันทจิวากรชัย **รหัสประจำตัว** 36061208

หลักสูตร วิศวกรรมศาสตรมหาบัณฑิต **สาขาวิชา** วิศวกรรมไฟฟ้า

อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ดร.มนัส สังวรศิลป์

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
รศ.ดร.มนัส	สังวรศิลป์	
ดร.กิติพล	ชิตสกุล	
รศ.ดร.กิตติ	ไพฑูรย์วัฒนกิจ	
ดร.สุริภณ	สมควรพาณิชย์	

ค่าระดับคะแนนที่ผ่านเป็นเอกฉันท์จากคณะกรรมการสอบ GOOD

วัน/เดือน/ปี ที่สอบ 30 ตุลาคม 2541 เวลา 10.00 - 12.00 น.

สถานที่สอบ ห้องสอบวิทยานิพนธ์ คณะวิศวกรรมศาสตร์ ตึก 12 ชั้น ชั้น 4 ห้อง (E12-404)

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.มนัส สังวรศิลป์)

คณบดีบัณฑิตวิทยาลัย

วันที่ 11 **เดือน** พฤศจิกายน **พ.ศ.** 2541

หมายเหตุ การวัดผลวิทยานิพนธ์ให้ใช้ค่าระดับคะแนนดังนี้

ค่าระดับคะแนน	ผลการศึกษา
O	Outstanding (ดีเยี่ยม)
G	Good (ดี)
P	Pass (ผ่าน)
F	Fail (ไม่ผ่าน)

หัวข้อวิทยานิพนธ์	การลดข้อมูลภาพโดยใช้เวฟเล็ตทรานฟอร์ม
นักศึกษา	นาย ชินภัทร นันทจิวากรชัย
รหัสประจำตัว	36061208
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2542
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร. มนัส สังวรศิลป์

บทคัดย่อ

ในปัจจุบันนี้ข้อมูลภาพได้เข้ามามีบทบาทเพิ่มขึ้น ในด้านการสื่อสาร เช่นการส่งสำเนาเอกสาร การส่งภาพในระยะไกล และยังใช้ประโยชน์ในการจัดเก็บเป็นฐานข้อมูล เช่นการบันทึกภาพถ่ายทางการแพทย์ ภาพลายนิ้วมือ เป็นต้น เนื่องจากการใช้ภาพจะสามารถสื่อความหมายได้ดี แต่การใช้ข้อมูลภาพนั้นจะมีอุปสรรคที่สำคัญคือ ปริมาณของข้อมูลที่มีจำนวนมากซึ่งจะทำให้การสื่อสารเป็นไปได้ช้า หรือในการจัดเก็บก็จะสิ้นเปลืองเนื้อที่มาก ดังนั้นการลดขนาดของข้อมูลภาพจึงเป็นสิ่งที่จำเป็น ในการลดขนาดของข้อมูลให้ได้มากนี้จะต้องมีการตัดรายละเอียดบางส่วนของภาพที่ไม่สำคัญออกไป ซึ่งจะทำให้ภาพที่ได้มีความผิดเพี้ยนจากภาพต้นแบบ ในวิทยานิพนธ์นี้จะได้นำเสนอถึงวิธีการลดข้อมูลภาพ โดยใช้การแปลงเวฟเล็ตแบบหลายระดับความละเอียด ในการแยกองค์ประกอบต่างๆ ของภาพออกมา แล้วทำการตัดข้อมูลในส่วนที่ไม่จำเป็นออกไปด้วยการตั้งระดับการรับรู้ และด้วยขบวนการเข้ารหัสแบบ Huffman ร่วมกับ การเข้ารหัสแบบ Run-length จะทำให้สามารถลดขนาดข้อมูลลงได้มาก

ผลการวิจัยที่ได้กับข้อมูลภาพทดสอบ ซึ่งเป็นข้อมูลภาพขาวดำ 256 ระดับ (Gray 8 Bit/Pixel) ขนาด 256x256 จุด อัตราการลดขนาดข้อมูลที่ได้ประมาณ 15-30 เท่า โดยมีค่าความผิดพลาด (e_{rms}) ระหว่าง 5 – 15 ขึ้นอยู่กับลักษณะของข้อมูลภาพ และเมื่อเปรียบเทียบกับการลดขนาดข้อมูลแบบ JPEG แล้วค่าความผิดพลาดที่ได้จะไกลเคียงกันที่อัตราการลดขนาดข้อมูลที่เท่ากัน แต่ภาพที่ได้จะไม่มีปัญหาของ blocking-effect

Thesis Title	Image Data Compression Using Wavelet Transform
Student	Mr. Chinnapat Nantajiwakornchai
Student ID.	36061208
Degree	Master of Engineering
Programme	Electrical Engineering
Year	1999
Thesis advisor	Assoc.Prof.Dr. Manus Sangworasilp

ABSTRACT

Up to now, image data play an important role in most communication activities, for instance, tele-communication system of an images and many business documents. Consequently, image data are stored and manipulated as an image database in some application tasks, such as, medical image database. Due to amount of image data, a high capacity of mass storage device or more capability of communication channel is required. An image data compression technique is a solution of this problem. To achieve more of compression ratio, unnecessary parts in the image must be discarded.

In this thesis, image data compression technique based on wavelet transform is presented. In the first phase, some image components, which extracted by using the wavelet transform is discarded by thresholding. In the second phase, Huffman and Run-length coding are applied to reduce some redundancy parts in the transformed data. Finally, a reduced image data size is obtained, however, the image quality must be accepted for using in most applications.

By using this technique with the 8 bit/pixel gray scale image, 256x256 pixel in size, the compression rate of 15 to 30 times can be retrieved while the error (e_{rms}) is between 5 - 15 depend on the details of an image. When compare to JPEG compression, this technique give the same error at equal compression ratio but no blocking-effect presented in an image.

กิตติกรรมประกาศ

ขอขอบคุณ รศ.ดร. มนัส สังวรศิลป์ ที่ได้ให้ความดูแล เอาใจใส่ และให้คำปรึกษา ในแนวทางการทำวิทยานิพนธ์ฉบับนี้ และให้ความรู้ ความสะดวก ในการทำงานเป็นอย่างดี ขอขอบพระคุณ ท่านคณะกรรมการสอบทุกท่านที่ได้สละเวลาอันมีค่า ขอขอบพระคุณเพื่อนๆ ทุกคนที่ได้ให้กำลังใจในการทำวิจัยตลอดมา

ชินภัทร นันทจิวากรชัย

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

สารบัญภาพ

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

ภาพที่ หน้า

บทที่ 1
บทนำ

ทุกวันนี้คอมพิวเตอร์ได้เข้ามามีบทบาทมากขึ้น โดยมีการประยุกต์ใช้งานในด้านต่างๆ รวมทั้งทางด้านของการสื่อสารข้อมูล และใช้ในการบันทึกข้อมูลต่างๆ เป็นต้น และข้อมูลส่วนหนึ่งที่ถูกนำมาใช้กับคอมพิวเตอร์นี้ จะเป็นข้อมูลของรูปภาพ โดยมีแนวโน้มที่จะมี การใช้งานเพิ่มขึ้น เนื่องจากข้อมูลภาพนี้จะสามารถสื่อความหมายได้ดีกว่า แต่การใช้งานข้อมูลภาพนี้จะมีข้อเสียคือ ปริมาณของข้อมูลที่มีจำนวนมาก จะเป็นอุปสรรคให้การสื่อสารข้อมูลกระทำได้ช้า และสิ้นเปลืองเนื้อที่ในการจัดเก็บมาก

ดังนั้นขบวนการลดขนาดของข้อมูลภาพจึงเป็นส่วนสำคัญที่จะเพิ่มความสามารถในการทำงานของอุปกรณ์ที่มีอยู่ ซึ่งจะทำให้ประหยัดค่าใช้จ่ายในการเพิ่มความสามารถของอุปกรณ์ได้เป็นอันมาก ยกตัวอย่างเช่น ในการส่งข้อมูลภาพทางไกลถ้าเราสามารถลดขนาดของข้อมูลภาพลงได้ 20 เท่าค่าใช้จ่ายที่ต้องเสียไปจะลดลง 20 เท่าด้วย และในกรณีของการเก็บบันทึกข้อมูลภาพก็เช่นเดียวกัน

ตัวอย่างเช่น ข้อมูลภาพ ขาวดำ 256 ระดับ ขนาด 256 X 256 จุดภาพ ซึ่งจะเป็นจำนวนของข้อมูลทั้งสิ้น 65536 ไบต์ ถ้าส่งข้อมูลนี้ ด้วยอัตรา 9600 บิตต่อวินาที จะต้องใช้เวลาเท่ากับ (65536*8)/9600 = 54.613 วินาที ซึ่งค่าใช้จ่ายในการส่งข้อมูลจะขึ้นอยู่กับเวลาที่ใช้ ดังนั้นถ้าปริมาณข้อมูล ที่เป็นตัวแทนของภาพนี้ มีขนาดที่ลดลง เวลาที่ใช้ในการส่งข้อมูล จะลดลงด้วย หรือในทางกลับกัน ในเวลาที่เท่าเดิม จะสามารถส่งข้อมูลภาพ ไปได้มากขึ้น ค่าใช้จ่ายในการส่งข้อมูลภาพ จึงลดลง

ด้วยเหตุนี้จึงได้มีแนวคิดที่จะพัฒนาขบวนการลดขนาดของข้อมูลภาพ ให้มีความสามารถมากขึ้น เพื่อนำไปประยุกต์ใช้งานอื่นๆ ได้ต่อไป

1.1 วัตถุประสงค์ของการวิจัย

การวิจัยที่ได้จัดทำขึ้นนี้ มีวัตถุประสงค์ดังต่อไปนี้

1. เพื่อศึกษาวิธีการของขบวนการการลดขนาดข้อมูลภาพ ซึ่งประกอบไปด้วยขั้นตอนของการประมวลผลต่างๆ ให้เข้าใจ และสามารถพัฒนาขบวนการลดขนาดข้อมูลภาพได้เอง

2. เพื่อที่จะนำขบวนการที่ได้พัฒนาขึ้นนี้ ไปประยุกต์ใช้ใน ระบบฐานข้อมูลภาพได้ต่อไป ทั้งนี้เพื่อเป็นการประหยัดวัสดุอุปกรณ์บันทึกข้อมูล
3. เพื่อเป็นแนวทางในการพัฒนาขบวนการลดขนาดของข้อมูลภาพ ให้กับผู้ที่สนใจได้ศึกษาต่อไป

1.2 แนวทางของการวิจัย

จากลักษณะของข้อมูลภาพที่ประกอบเข้าด้วยกันจากจุดที่มีระดับความสว่างต่างๆ จำนวนมากนั้น การที่จะลดขนาดข้อมูลที่จะต้องจัดเก็บเหล่านี้ โดยอาศัยแต่ลักษณะทางสถิติของข้อมูลที่เกิดขึ้นซ้ำๆ กันเพียงอย่างเดียว เช่นวิธีการของ DPCM (Differential Pulse Code Modulation) หรือการใช้ Entropy coding ของ Sub band Image ด้วยการนำเอาลักษณะที่ซ้ำๆ กันออกไปนั้น จะไม่สามารถลด ปริมาณของข้อมูลลงได้มากนัก ในกรณีที่ต้องการลดปริมาณของข้อมูลลงให้ได้มากขึ้นอีกนั้น จำเป็นจะต้องมีการตัดเอาข้อมูลบางส่วนออกไป ซึ่งการพิจารณาข้อมูลที่ต้องตัดออกไปโดยใช้เพียงแต่ลักษณะทางสถิตินี้ จะส่งผลให้ภาพที่ได้มีความผิดพลาดที่สามารถสังเกตได้ชัดเจน แม้ว่าค่าความผิดพลาดที่คำนวณเปรียบเทียบกับภาพต้นแบบจะมีค่าน้อยก็ตาม แต่ด้วยการแยกองค์ประกอบของภาพออกมาเป็นส่วนย่อยตามลักษณะการมองเห็นของมนุษย์ จะทำให้เราสามารถเลือกเอาข้อมูลที่ไม่มีความสำคัญ ออกไปได้ตามต้องการ

การมองเห็นของมนุษย์ซึ่งจะรับรู้โดยการที่แสงตกกระทบเซลประสาทรับแสงที่อยู่ภายในดวงตาแล้วส่งเป็นสัญญาณไปสู่สมองเพื่อทำการประมวลผลต่อไป ภาพของวัตถุที่มองเห็นจะถูกตีความหมายเป็น โครงสร้างของพื้นผิวที่มีสีหรือลักษณะเดียวกัน แต่เนื่องจากสีหรือลักษณะของพื้นผิวนี้จะขึ้นอยู่กับแสงและเงา [1] ซึ่งจะแตกต่างกันไป มนุษย์จึงอาศัยขอบของภาพหรือรูปร่างในการแยกแยะวัตถุต่างๆ ดังนั้นวิธีการที่จะใช้ในการลดขนาดข้อมูลภาพที่ดี จะต้องพยายามไม่ให้เกิดความผิดพลาดที่ขอบของรูปหรือลายเส้นต่างๆ

ในงานการวิจัยนี้จะเป็นการลดขนาดข้อมูลภาพขาวดำตามหลักการข้างต้น โดยจะใช้การแปลงเวฟเล็ตแบบ Multiresolution Analysis (MRA) [2] ในการแยกส่วนประกอบต่างๆ ของภาพออกมา จากนั้นจะเลือกตัดข้อมูลส่วนที่ไม่มีความสำคัญออก ข้อมูลที่เหลือนี้ก็จะถูกเข้ารหัสที่เหมาะสมต่อไป โดยเนื้อหาในรายงานการวิจัยฉบับนี้จะแบ่งเป็นหัวข้อดังนี้ บทที่ 1 นี้จะกล่าวถึงที่มาและหลักการที่ใช้ในการลดข้อมูลภาพ ในบทที่ 2 จะเป็นทฤษฎีของการแปลงเวฟเล็ตแบบ discrete time wavelet series (DTWS) ในบทที่ 3 เป็นทฤษฎีของการลดข้อมูลและการเข้ารหัสข้อมูลที่ใช้ ในบทที่ 4 จะเป็นวิธีการลดขนาดข้อมูลภาพที่ได้วิจัยขึ้นประกอบกับผลการทดลองที่

ได้ ในกรณีต่างๆ กัน และในบทที่ 5 จะเป็นการสรุปผลงานวิจัยที่ได้ และแนวทางในการพัฒนาต่อไป

บทที่ 2
การแปลงเวฟเล็ต (Wavelet transform)

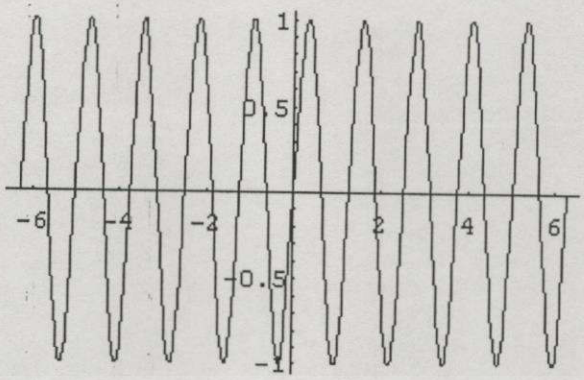
2.1 บทนำ

ทฤษฎี เวฟเล็ต (Wavelet theory) เป็นขบวนการทางคณิตศาสตร์ ที่ใช้อธิบายถึง โมเดล ของสัญญาณ ระบบ หรือขบวนการ โดยใช้เซตของสัญญาณ ที่มีลักษณะเฉพาะ เป็นองค์ประกอบพื้นฐาน โดยจะเรียกว่า เซตของเวฟเล็ต (Wavelet set) สมาชิกในเซตของ เวฟเล็ตนี้ จะมีคุณสมบัติสำคัญคือ เป็นสัญญาณที่เกิดจากต้นแบบอันเดียวกัน ต้นแบบของสัญญาณใน เซตของเวฟเล็ตนี้ จะเรียกว่า เวฟเล็ตแม่ (Mother wavelet)

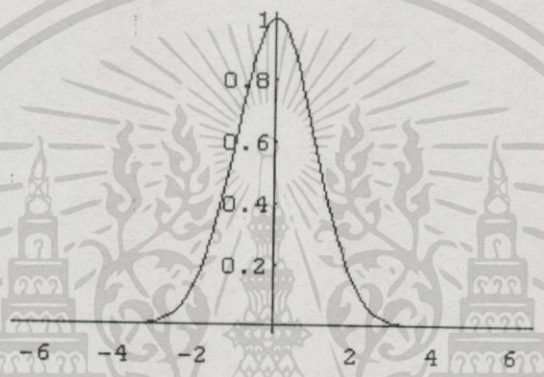
เวฟเล็ต จะมีคุณสมบัติเป็นสัญญาณที่เกิดขึ้นในช่วงเวลาสั้นๆ หรือกล่าวคือ เวฟเล็ตเป็นสัญญาณที่เกิดขึ้นอย่างต่อเนื่องและ มีขนาดที่ลดลงสู่ ศูนย์ ในทั้งสองด้าน ดังแสดงในรูปที่ 2.1

รูปที่ 2.1 Morlet mother wavelet

สัญญาณในรูปที่ 2.1 จะประกอบด้วยลักษณะทั้งสองคือ การเกิดขึ้นอย่างต่อเนื่อง ซึ่งได้จากสัญญาณ Sine ในรูปที่ 2.2 และการลู่เข้าสู่ศูนย์ของสัญญาณในทั้งสองด้านจาก Window function ดังรูปที่ 2.3 เมื่อนำ Function ทั้งสองนี้มาคูณเข้าด้วยกัน จะได้เป็นลักษณะสัญญาณของ เวฟเล็ต ดังรูปที่ 2.1



รูปที่ 2.2 รูปสัญญาณของ y = Sin(5 x)



รูปที่ 2.3 รูปสัญญาณของ y = Exp(-.5x^2)

จากเวฟเล็ตแม่ ที่ได้ จะถูกนำไปสร้างเป็นสมาชิกตัวอื่นๆ ใน เซตของเวฟเล็ต ที่ต้องการจะนำไปอธิบายถึงสัญญาณ ขบวนการ หรือระบบต่างๆ โดยสมาชิกตัวอื่นๆ นี้จะได้มาจาก เวฟเล็ตแม่ที่ถูก "สเกล" (Scaling : a) และ "เลื่อนตำแหน่ง" (Translation : b) ไปตามแกนของเวลา ถ้าให้ $g(t)$ เป็น function ของ Mother wavelet แล้ว การ "สเกล" ด้วยพารามิเตอร์ของ "a" และ "การเลื่อนตำแหน่ง" ด้วยพารามิเตอร์ของ "b" จะสามารถหาได้จากสมการ

$$g_{b,a}(t) = g\left(\frac{t-b}{a}\right) \quad \text{.....(2.1)}$$

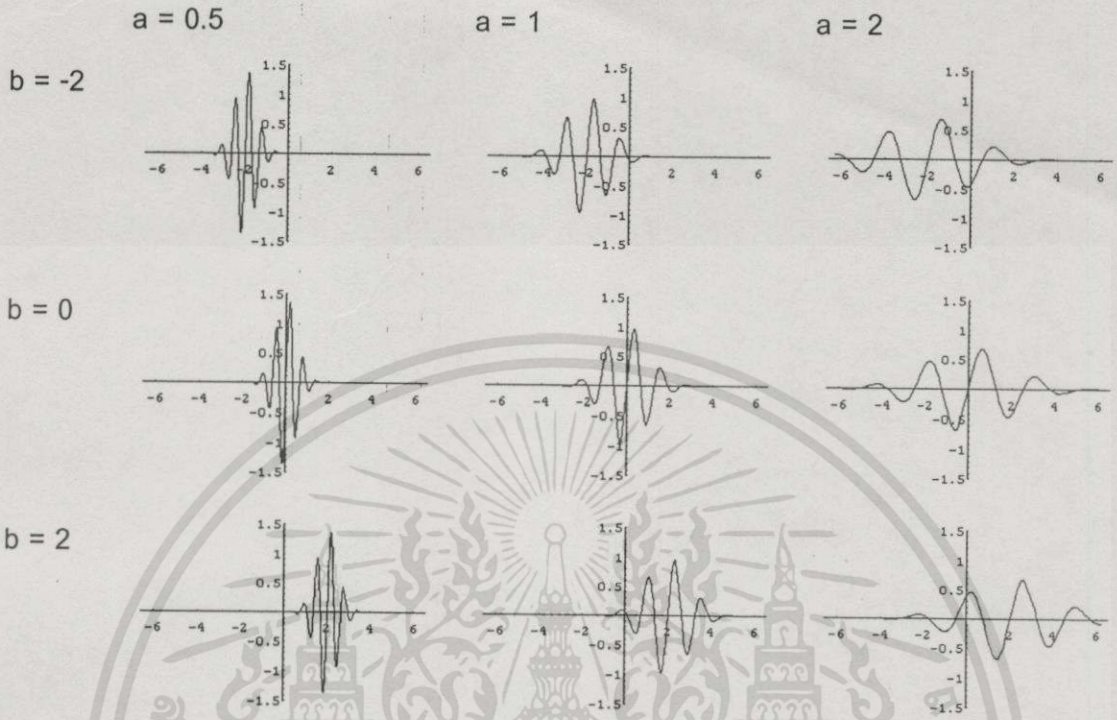
เมื่อ a ค่าอัตราส่วนของการ Scaling

b การเลื่อนตำแหน่ง

ซึ่งการทำ Scaling และ Translation ตามแกนของเวลานี้ จะรวมเรียกว่า Affine Operation (การทำ Linear Mapping ร่วมกับการทำ Translation) และเพื่อให้สัญญาณที่ถูก "สเกล" แล้วมีค่าพลังงานของสัญญาณ ที่เท่าเดิม จึงจะต้องทำการ Normalization สัญญาณที่ได้นี้ด้วย $1/\sqrt{a}$ เสมอ จะได้

$$g_{b,a}(t) = \frac{1}{\sqrt{a}} g\left(\frac{t-b}{a}\right) \quad \text{.....(2.2)}$$

จากรูปที่ 2.4 จะแสดงถึงตัวอย่างของสมาชิกใน เซตของเวฟเล็ต ที่มีเวฟเล็ตแม่ แบบ Morlet ตามตัวอย่างที่ได้กล่าวมาแล้ว



รูปที่ 2.4 Scaling and translation of mother wavelet

จากตัวอย่าง จะสามารถสังเกตได้ว่า ลักษณะและจำนวนลูกคลื่นของสัญญาณนั้นจะเหมือนกันทั้งหมด สมาชิกแต่ละตัวจะแตกต่างกันที่ ขนาดและ ตำแหน่ง ทางแกนเวลาเท่านั้น

2.2 พื้นฐานของ ทฤษฎีเวฟเล็ต

ทฤษฎีของเวฟเล็ต ก็เป็นวิธีการอย่างหนึ่ง เช่นเดียวกับระบบตัวเลข ที่ใช้แทนจำนวนของสิ่งต่างๆ ทฤษฎีของเวฟเล็ตจะสามารถใช้อธิบายถึงสิ่งต่างๆ ได้ ไม่ว่าจะเป็น ราคาหุ้น ผลเฉลยของสมการดิฟเฟอเรนเชียล การเต้นของหัวใจ การไหลเวียนของโลหิต สภาพภูมิอากาศ เป็นต้น [8] และนอกเหนือจากนี้ทฤษฎีของเวฟเล็ตยังสามารถใช้อธิบายถึงการทำงานของระบบต่างๆ ได้

ทฤษฎีของเวฟเล็ตซึ่งเป็นเครื่องมือคณิตศาสตร์อย่างหนึ่งที่สามารถนำไปประยุกต์ใช้งานได้กับสิ่งต่างๆ โดยมีวัตถุประสงค์ที่จะใช้อธิบาย ระบบ หรือสัญญาณเหล่านั้นได้ดีกว่าวิธีการอื่นๆ การที่จะนำทฤษฎีของเวฟเล็ตไปใช้งาน จึงอยู่บนพื้นฐานที่ว่า การนำมาใช้งานนั้นมีประสิทธิภาพเพียงใด

ทฤษฎีของเวฟเล็ตจะอธิบายสิ่งใดสิ่งหนึ่ง โดยการแยกสิ่งเหล่านั้น ออกเป็นส่วนประกอบย่อยๆ ที่มีความสัมพันธ์กัน โดยที่ส่วนย่อยๆ เหล่านี้ก็คือ เวฟเล็ตแม่ที่ถูก สเกลและเลื่อนตำแหน่งไป และจะมีค่าสัมประสิทธิ์ (Coefficient) คูณอยู่กับเวฟเล็ตแต่ละตัว

ขบวนการในการแยกสัญญาณออกเป็นส่วนย่อยๆ หรือการหาค่า ของสัมประสิทธิ์นี้ จะเรียกว่า "การแตกกระจายเวฟเล็ต" (Wavelet decomposition) หรือ "การแปลงเวฟเล็ต" (Wavelet transform : WT) และในทางกลับกัน การนำค่าสัมประสิทธิ์แยกออกเป็นส่วนย่อยๆ มารวมกลับเพื่อได้สัญญาณเดิมที่ต้องการ จะเรียกว่า "การรวมกลับเวฟเล็ต" (Wavelet reconstruction) หรือ "การแปลงผกผันแบบเวฟเล็ต" (Inverse wavelet transform : IWT)

การแปลงเวฟเล็ตแบบต่อเนื่อง ของฟังก์ชัน $f(x)$ จะสามารถแสดงได้โดยสมการทางคณิตศาสตร์คือ

$$W_g[f(x)](a,b) = |a|^{-\frac{1}{2}} \int f(x) g^*(\frac{x-b}{a}) dx \quad \text{.....(2.3)}$$

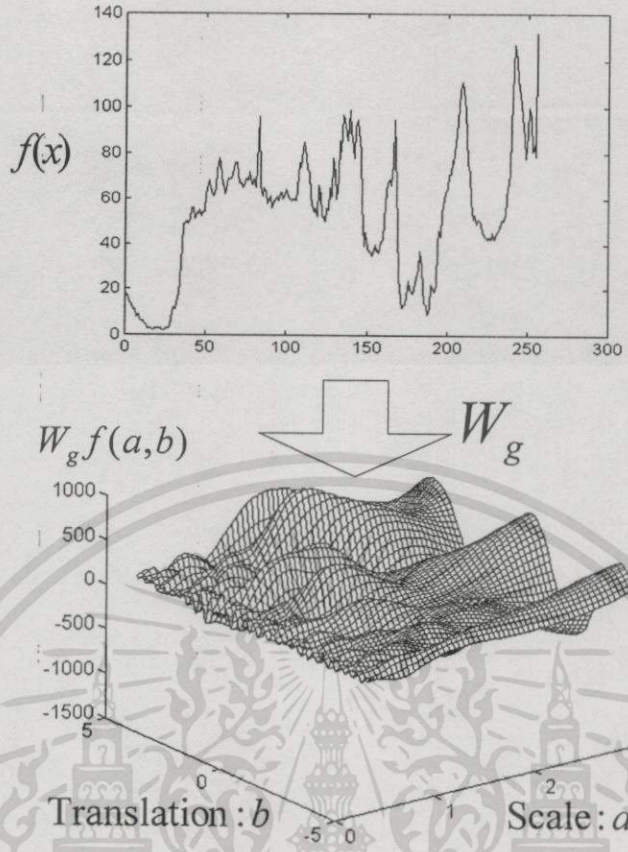
เมื่อ $g(x)$ ฟังก์ชันของ เวฟเล็ตแม่ และ $g^*(x)$ คือ Complex conjugate ของ $g(x)$

a Scaling

b Translation

และจะใช้ สัญลักษณ์ $W_g f(a,b)$ แทนค่าของสัมประสิทธิ์ที่ได้ เพื่อความสะดวก

จากข้างต้นจะเห็นได้ว่าการแปลงเวฟเล็ต จะเป็นการส่งผ่านฟังก์ชัน ที่มีตัวแปรอิสระ x ไปยังปฐิมภูมิ (Space) ของฟังก์ชันที่เป็นสองมิติ ของตัวแปรอิสระคือ a และ b ตามลำดับ ค่าสัมประสิทธิ์ของเวฟเล็ต ที่ a,b นี้ จะแสดงถึง สหสัมพันธ์ (Degree of Correlation) ระหว่าง เวฟเล็ตแม่ ที่สเกล a และตำแหน่ง b กับสัญญาณอินพุท f ที่ตำแหน่งเดียวกัน โดยถ้าสัญญาณทั้งสองนี้มีความเหมือนกันมาก ค่าสัมประสิทธิ์ที่ได้ก็จะมีค่ามากตามไปด้วย และเซตของ ค่าสัมประสิทธิ์ทั้งหมดที่ได้ ($W_g f(a,b)$ ที่ a,b ใดๆ) จะเรียกว่า โดเมน ของเวฟเล็ต (Wavelet domain) ซึ่งจะเป็นการอธิบายถึงฟังก์ชัน f โดยใช้เวฟเล็ตแม่ g ดังรูปที่ 2.1 จะแสดงถึงการแปลงเวฟเล็ต ของสัญญาณ $f(x)$ ให้อยู่ใน โดเมนของเวฟเล็ต g



รูปที่ 2.5 แสดงการแปลงเวฟเล็ต ของสัญญาณ $f(x)$ ให้อยู่ใน โดเมนของเวฟเล็ต g

จะสังเกตได้ว่า สำหรับการแปลงเวฟเล็ตนั้น ส่วนของ เวฟเล็ตแม่ที่ สเกลและ ตำแหน่งต่างๆ (จากตัวอย่างในรูปที่ 2.4) จะมีจำนวนของลูกคลื่นสัญญาณที่เท่ากัน เปรียบเสมือนกับ Windows Function ที่มีขนาดเปลี่ยนไปตามความถี่ ถ้าค่าของ สเกล (a) น้อยกว่าหนึ่ง ขนาดของ window นี้ก็จะเล็กลง จำนวนลูกคลื่นต่อหนึ่งหน่วยเวลาก็จะมาก จำนวนลูกคลื่นสัญญาณที่มากก็จะหมายถึงความถี่สูงนั้นเอง แต่ถ้าจำนวนลูกคลื่นต่อหนึ่งหน่วยเวลามีค่าน้อย ก็จะหมายถึงความถี่ต่ำ จะเห็นได้ว่าค่าของช่วงเวลา กับความถี่ จะมีความสัมพันธ์ กัน การแปลงเวฟเล็ตจะกำหนด ช่วงเวลาที่สั้น สำหรับการวิเคราะห์ ความถี่สูง และช่วงเวลาที่ยาวขึ้น สำหรับความถี่ที่ต่ำลงมา

2.3 ตัวแปลงเวฟเล็ต และคำจำกัดความ

ก่อนที่จะอธิบายถึงขบวนการแปลงเวฟเล็ตนั้น จะต้องกำหนดถึงฟังก์ชันที่จะสามารถนำมาใช้เป็น เวฟเล็ตแม่ได้ จากที่ได้กล่าวมาแล้ว ฟังก์ชันที่จะเป็นเวฟเล็ตได้นั้น จะต้องมีคุณสมบัติของการแกว่งไปมา และการลู่เข้าสู่ศูนย์ ซึ่งฟังก์ชันที่จะสามารถใช้เป็นเวฟเล็ตแม่ จะต้องมีคุณสมบัติทั้งสองข้อนี้ ทั่วไปแล้วสัญญาณต่างๆ นั้นจะมีคุณสมบัติทั้งสองข้อนี้อยู่แล้ว หรือจะกล่าวได้ว่า

ฟังก์ชันที่สามารถหาค่าของพลังงานได้ (Finite energy function : $L^2(\mathbf{R})$)) g จะใช้เป็นเวฟเล็ตแม่ ได้เมื่อ

$$c_g = \int_{-\infty}^{\infty} \frac{|G(\omega)|^2}{|\omega|} d\omega < \infty \quad \text{.....(2.4)}$$

เมื่อ $G(\omega)$ Fourier transform ของ g

c_g ค่าพลังงานของ g

ตัวแปลงเวฟเล็ต W_g จะเป็นการ Mapping สัญญาณที่เป็น Finite energy ของ $L^2(\mathbf{R}) \to L^2(\mathbf{R}\setminus\{0\}\times\mathbf{R})$ หรือจะอธิบายได้คือ สัญญาณที่เป็น Finite energy ใน Time หรือ Space Domain ถูก Mapping ไปเป็นสัญญาณ สองมิติ ของ Scale และ Translation หรือ Wavelet Domain การแปลงเวฟเล็ตของฟังก์ชัน f ด้วยฟังก์ชันของเวฟเล็ตแม่ g จะกำหนดโดย

$$\begin{aligned} \text{Wavelet Coefficient} &= W_g f(a,b) \\ &= |a|^{-\frac{1}{2}} \int f(x) g^*\left(\frac{x-b}{a}\right) dx \\ &= \left\langle f, \frac{1}{\sqrt{|a|}} g\left(\frac{x-b}{a}\right) \right\rangle \\ &= \langle f, g_{a,b} \rangle = \langle f, U(a,b) g \rangle \end{aligned} \quad \text{.....(2.5)}$$

เมื่อ * แสดงถึง Complex conjugate

$\langle .,. \rangle$ Inner product

$$U(a,b) : g(x) \to \frac{1}{\sqrt{|a|}} g\left(\frac{x-b}{a}\right)$$

จากรูปที่ 2.5 ซึ่งเป็นตัวอย่างของการแปลงเวฟเล็ต ในตัวอย่างนี้ ฟังก์ชัน $g(x)$ จะต้องมีคุณสมบัติของเวฟเล็ต เซตของเวฟเล็ตจะสร้างได้จาก $g_{a,b}$ ที่กำหนดโดย Unitary affine mapping $U(a,b): g(x) \to \frac{1}{\sqrt{|a|}} g\left(\frac{x-b}{a}\right)$ หรือจะสามารถอธิบายได้ง่ายๆ คือ $g_{a,b}$ เกิดจากกฟังก์ชันของ เวฟเล็ตแม่ ที่ถูกสเกล ด้วยอัตราส่วน a และเลื่อนไปด้วยค่าของ b ค่าอัตราส่วนของการ สเกลนี้ จึงอยู่ในเซตของเลขจำนวนจริง ที่ไม่เท่ากับ 0 (R \ { 0 }) ส่วนคำว่า "Unitary" นั้นจะหมายถึง Energy normalization ซึ่งได้จาก $\frac{1}{\sqrt{|a|}}$ การทำ Energy normalization นี้จะทำให้ เวฟเล็ตแม่ที่ถูกสเกลแล้วนั้นยังคงมีค่าขนาดของพลังงานที่เท่าเดิม หรืออาจใช้เวฟเล็ตแม่ ที่มีค่าของพลังงานเท่ากับ หนึ่งหน่วย ($c_g = 1$) ก็ได้

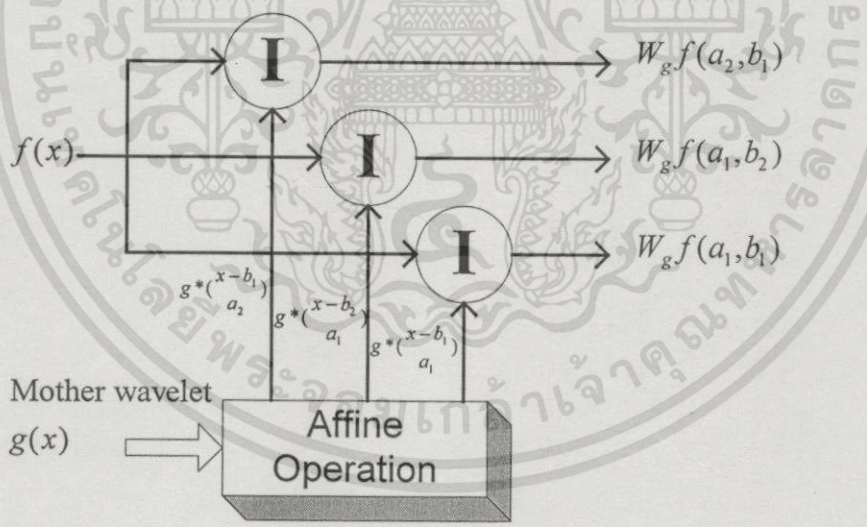
การที่ค่าพลังงานของเวฟเล็ตแม่มีค่าเป็นหนึ่งหน่วย จะทำให้เกิดคุณสมบัติที่สำคัญ คือค่าพลังงานของสัมประสิทธิ์ที่ได้ในเวฟเล็ตโดเมน จะเท่ากับค่าพลังงานของสัญญาณที่อยู่ใน โดเมน

ของเวลา หรือในปฐิมภูมิ และยังทำให้ ค่าสัมประสิทธิ์เวฟเล็ต ที่ (a,b) ใดๆ สามารถที่จะนำมาเปรียบเทียบกันได้

สำหรับการการแปลงเวฟเล็ตแบบต่อเนื่อง (Continuous wavelet transform) นั้น คุณสมบัติที่ต้องการของฟังก์ชันที่จะนำมาเป็น เวฟเล็ตแม่ นั้น จะมีเพียงดังที่กล่าวข้างต้น ซึ่งการเลือกใช้ฟังก์ชันของเวฟเล็ตแม่ ก็จะขึ้นอยู่กับ การประยุกต์ใช้งานที่ต้องการ

เซตของค่าสัมประสิทธิ์เวฟเล็ตนี้ จะสามารถใช้เป็นตัวแทนของสัญญาณ และจากการแปลงผกผันของค่าสัมประสิทธิ์ที่ได้ โดยใช้เซตของเวฟเล็ตที่ได้ ด้วยเวฟเล็ตแม่ตัวเดิม ก็จะสามารถนำมาใช้ในการแปลงผกผันได้ (Inverse transform)

ค่าสัมประสิทธิ์เวฟเล็ต แต่ละตัว จะเปรียบเสมือนเป็น ส่วนประกอบย่อยๆ ของสัญญาณ (ฟังก์ชัน) และเมื่อรวมส่วนประกอบย่อยๆ เหล่านี้เข้าด้วยกัน จะได้สัญญาณกลับออกมา ในการที่จะหาค่าของสัมประสิทธิ์เวฟเล็ตที่จะเป็นตัวแทนของสัญญาณ สัญญาณนั้นจะถูกฉาย (Projection) ลงไปยังส่วนประกอบย่อยในเวฟเล็ตเซต ซึ่งผลลัพท์จะได้เป็นค่าจำนวนจริง หรือจำนวนเชิงซ้อน ที่เรียกว่า ค่าสัมประสิทธิ์เวฟเล็ต (Wavelet coefficient) ของสัญญาณ หรือฟังก์ชัน f ที่สัมพันธ์กับ เวฟเล็ตแม่ g หรือแสดงด้วย $W_g f(a,b)$ ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 ขบวนการแปลงเวฟเล็ต

นอกเหนือจากที่อธิบายได้ด้วยสมการที่ 2.5 แล้ว ขบวนการฉาย จะสามารถเรียกได้คือ ขบวนการ Correlation ซึ่งเป็นการเปรียบเทียบว่าสัญญาณ ทั้งสองมีความเหมือนกันมากเพียงใด สัญญาณ f จะถูกนำมาเปรียบเทียบกับ เวฟเล็ตแม่ g ที่ถูกสเกล และเลื่อนออกไป จากในรูป

ที่ 2.6 ขบวนการของฉายหรือ Correlation นี้ จะแสดงด้วยสัญลักษณ์ I อยู่ภายในวงกลม ข้อมูลเอาท์พุทที่ได้ ก็จะเป็น ค่าสัมประสิทธิ์เวฟเล็ต $W_g f(a,b)$ ที่ต้องการ

ในขบวนการแปลง เวฟเล็ตที่แสดงในรูปที่ 2.6 จะสามารถเรียกได้อีกอย่างหนึ่งว่า เป็น Analysis filter โดยมันจะทำการแตกกระจายสัญญาณ ออกเป็นส่วนย่อย ซึ่งจะถูกนำไปใช้ในการวิเคราะห์ต่อไป ขบวนการ แปลงผกผันเวฟเล็ต หรือ Synthesis filter จะเป็นการนำส่วนประกอบย่อยๆ เหล่านี้ รวมกลับเข้าด้วยกัน อีกครั้ง ในการแตกกระจายสัญญาณออกเป็นส่วนย่อยๆ นี้ก็เพื่อประสิทธิภาพ ในการประมวลผลสัญญาณนั้นๆ ทั้งนี้เนื่องจากการวิเคราะห์ส่วนประกอบย่อยนี้จะทำได้ดีกว่า ยกตัวอย่างเช่น เมื่อพิจารณาถึงเสียงของวงดนตรีหนึ่งๆ ที่ประกอบไปด้วยเสียงจากเครื่องดนตรี หลายๆชนิด อย่างไรก็ดี สำหรับผู้ฟังที่มีประสพการ จะสามารถแยกแยะเสียงที่มาจากเครื่องดนตรีแต่ละชิ้นได้ และยิ่งถ้าเป็นคนที่เล่นดนตรีได้อยู่แล้ว ก็จะสามารถ ย่อ/แทน เพลงๆ นี้ด้วยโน้ตของเครื่องดนตรีต่างๆ ได้ ซึ่งต่อมาภายหลังถ้าต้องการที่จะฟังเพลงนี้อีกครั้ง ก็จะสามารถทำได้โดยการเล่นตามโน้ตที่ได้บันทึกไว้ จะเห็นได้ว่าโน้ตเพลงเป็นตัวแทนที่มีประสิทธิภาพของเพลงๆ หนึ่ง อย่างไรก็ดี ในการแทนเพลงๆ หนึ่งด้วย โน้ตดนตรีนั้น เพลงที่เล่นกลับออกมาได้จะไม่เหมือนต้นฉบับที่เดียวนัก อาจมีที่ผิดเพี้ยนไปบ้าง แต่ก็ยังคงสื่อความหมาย ถึงเพลงเดียวกันได้อยู่ดี

2.4 การแปลงผกผัน เวฟเล็ต

การแปลงผกผันเวฟเล็ต ที่ได้จากขบวนการแปลงเวฟเล็ต ตามสมการที่ 2.5 นั้น จะสามารถทำได้ดังขบวนการแสดงดังรูปที่ 2.7 ซึ่งในการแปลงผกผันเวฟเล็ตนี้จะได้ฟังก์ชันเดิมกลับออกมา

ถ้า $f(x)$ และ $g(x)$ เป็นฟังก์ชันที่สามารถหาค่าของพลังงานได้แล้ว (อยู่ใน $L^2(\mathbf{R})$) และ $g(x)$ เป็นฟังก์ชันที่มีคุณสมบัติเป็นเวฟเล็ตแม่ได้ แล้ว $g_{b,a}(x)=\frac{1}{\sqrt{a}}g\left(\frac{x-b}{a}\right)$ $(a,b) \in (\mathbf{R} \setminus \{0\} \times \mathbf{R})$ ขบวนการแปลงผกผันเวฟเล็ต W_g^{-1} จะเป็นการ Mapping พื้นผิวของค่าสัมประสิทธิ์เวฟเล็ต มาเป็นสัญญาณ หนึ่งมิติ ของเวลา คือ $W_g^{-1} : L^2(\mathbf{R} \setminus \{0\} \times \mathbf{R}) \rightarrow L^2(\mathbf{R})$ แสดงได้ดังสมการ

$$W_g^{-1} : W_g f(a,b) \rightarrow f(x)$$

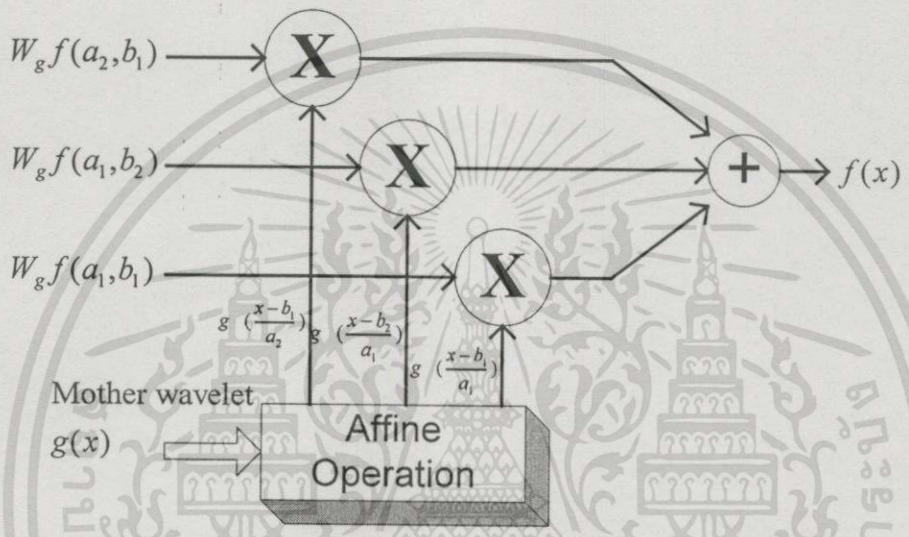
$$f(x)=\frac{1}{c_g}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} W_g f(a,b)\frac{1}{\sqrt{|a|}}g\left(\frac{x-b}{a}\right)\frac{dbda}{a^2} \quad \text{.....(2.6)}$$

การแปลงผกผันเวฟเล็ตจะสร้างฟังก์ชัน/สัญญาณ ต้นฉบับขึ้นมาจากการรวมกันของ เวฟเล็ตแม่ (Mother wavelet : g) ที่ Scale - Translation ต่างๆ ที่คูณอยู่กับค่าถ่วงน้ำหนัก (ค่าสัมประสิทธิ์เวฟเล็ต : $W_g f(a,b)$) และสำหรับในกรณีที่ การแปลงผกผันเวฟเล็ตนี้ ใช้เวฟเล็ตแม่

ที่แตกต่างไปจาก เวฟเล็ตแม่ที่ใช้ในขบวนการแปลงเวฟเล็ต ผลลัพท์ที่ได้ก็จะเป็นสัญญาณ หรือฟังก์ชันที่แตกต่างไปจากฟังก์ชัน/สัญญาณ ต้นฉบับ คือ

$$\frac{1}{c_g}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} W_g f(a,b)\frac{1}{\sqrt{|a|}} g_2\left(\frac{x-b}{a}\right)\frac{dbda}{a^2} \neq f(x) \quad \text{.....(2.7)}$$

ถ้า $g_2(x) \neq g(x)$



รูปที่ 2.7 ขบวนการแปลงผกผันเวฟเล็ต

2.5 การกระจายของพลังงาน ในเวฟเล็ตโดเมน

จากที่ สัมประสิทธิ์เวฟเล็ต เป็นตัวแทนของ ฟังก์ชัน/สัญญาณ ได้รูปแบบหนึ่ง มันก็จะมีคุณสมบัติเฉพาะตัว เช่นเดียวกันสัญญาณนั้นๆ คุณสมบัติที่สำคัญของสัญญาณก็คือ ค่าพลังงานของสัญญาณนั้นๆ เอง ค่าพลังงานของสัญญาณ (E_f) จะสามารถคำนวณได้จากสมการ

$$E_f = \int_{-\infty}^{\infty} |f(t)|^2 dt \quad \text{.....(2.8)}$$

เมื่อ E_f ค่าพลังงานของสัญญาณ

$f(t)$ สัญญาณ

สำหรับตัวแทนของสัญญาณนี้ ที่อยู่ในรูปแบบอื่น เช่น ใน Fourier transform domain ค่าของพลังงานใน Fourier transform domain จะมีความสัมพันธ์กับความถี่คือ $\Delta E_f = |F(\omega)|^2 d\omega$

และเมื่อรวมค่าของพลังงานทั้งหมดจะได้

$$E_f = \frac{1}{2\pi}\int_{-\infty}^{\infty}|F(\omega)|^2 d\omega \quad \text{.....(2.9)}$$

และก็เช่นเดียวกับกรณีของ Fourier Transform ค่าพลังงานของสัญญาณ ที่อยู่ใน Wavelet transform domain ก็จะมีความสัมพันธ์กับความถี่ด้วย ใน Wavelet transform domain ค่าความถี่จะหมายถึงการ Scaling ค่าพลังงานจะเป็นไปตามสมการ

$$\Delta E_f = \left|W_g f(a,b)\right|^2 \frac{dadb}{a^2} \quad \text{.....(2.10)}$$

และค่าพลังงานรวมจะได้เป็น

$$E_f = \frac{1}{c_g}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left|W_g f(a,b)\right|^2 \frac{dadb}{a^2} \quad \text{.....(2.11)}$$

การ แสดงกราฟของค่าพลังงาน ของสัมประสิทธิ์เวฟเล็ต จึงจะเรียกว่า Scalogram จากสมการที่ 2.11 ค่าของ c_g จะหมายถึงค่าพลังงานของ เวฟเล็ตแม่ ซึ่งในกรณีที่เวฟเล็ตแม่มีค่าของพลังงานเป็นหนึ่งหน่วย แล้วค่าของ c_g จะถูกละไว้ ทั้งนี้ค่าการกระจายของพลังงานใน Wavelet transform domain จะขึ้นอยู่กับ ตัวเวฟเล็ตแม่ที่ใช้ด้วย ซึ่งการใช้เวฟเล็ตแม่ ที่ไม่เหมือนกันนั้น ก็ได้ Scalogram ที่แตกต่างกันออกไป

2.6 อนุกรมเวฟเล็ตเวลาต่อเนื่อง (Continuous Time Wavelet Series)

เช่นเดียวกับ ทฤษฎีของ Fourier transform การแปลงแบบต่อเนื่อง (continuous transform) จะใช้เฉพาะในการพิสูจน์คุณสมบัติต่างๆ เท่านั้น ส่วนในการประยุกต์ใช้งานจริง โดยใช้ตัวประมวลผลสัญญาณแบบ ดิจิตอลนั้น จะกระทำโดยใช้ การแปลงแบบเต็มหน่วย (Discrete transform) อนุกรมเวฟเล็ตเวลาต่อเนื่อง ในที่นี้ จะหมายถึงเฉพาะ Transform Domain parameter เท่านั้น (เช่น ค่าของความถี่ สำหรับการแปลง Fourier หรือ ค่าของ Scale และ Translation สำหรับการแปลง wavelet) ที่เป็นแบบ เต็มหน่วย โดยจะไม่รวมถึงตัวแปรอิสระของฟังก์ชันที่นำมา transform (เช่น ค่าของเวลา เป็นต้น) หรือจะกล่าวได้ว่าการแปลงเวฟเล็ตนี้ จะมีค่าของ Scale (a) และ Translation (b) ที่มีค่าแบบเต็มหน่วย ส่วนค่าเวลาของสัญญาณยังคงต่อเนื่อง ขบวนการแปลง เวฟเล็ต ที่อธิบายข้างต้น จะเรียกว่า Continuous time wavelet series (CTWS)

ในอนุกรมเวฟเล็ตเวลาต่อเนื่อง (CTWS) นี้ จำนวนของค่าสัมประสิทธิ์ ที่ได้ใน Transform domain จะมีจำนวนจำกัด (ในกรณีของ การแปลงเวฟเล็ตแบบต่อเนื่อง จะมีจำนวนของค่า

สัมประสิทธิ์ที่ได้ เป็นอนันต์) ซึ่งค่าสัมประสิทธิ์ ก็จะแบ่งออกเป็นส่วนๆ ตามค่าของ Scale ที่กำหนดโดยเลขจำนวนเต็ม m และค่าของ Translation ที่กำหนด โดยเลขจำนวนเต็ม n

ดังนั้นค่าของ Scale (a) และค่าของ Translation (b) จะเป็น

$$a = a_0^m \quad \text{.....(2.12)}$$

$$b = nb_0a_0^m \quad \text{.....(2.13)}$$

เมื่อ a_0 ช่วงของ scale

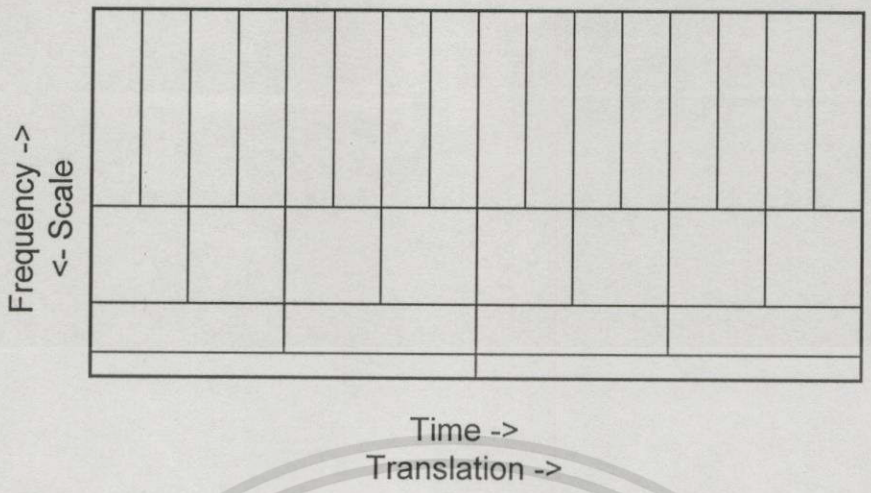
b_0 ช่วงของ translation

ซึ่งเมื่อนำไปแทนในสมการ 2.3 แล้ว สมการของ อนุกรมเวฟเล็ตเวลาต่อเนื่อง หรือ CTWS จะได้เป็น

$$\begin{aligned} W_g f(m,n) &= \frac{1}{\sqrt{a_0^m}} \int_{-\infty}^{\infty} f(x) g\left(\frac{x - nb_0a_0^m}{a_0^m}\right) dx \\ &= a_0^{-m/2} \int_{-\infty}^{\infty} f(x) g(a_0^{-m}x - nb_0) dx \end{aligned} \quad \text{.....(2.14)}$$

ซึ่งสมการข้างต้นนี้ จะยังคงใช้การ Integrate เนื่องจาก สัญญาณ $f(x)$ และ Mother wavelet $g(x)$ นั้นยังเป็นแบบเวลาต่อเนื่อง และเช่นเดียวกันกับการแปลงเวฟเล็ตแบบต่อเนื่อง ค่าของ สัมประสิทธิ์ที่ได้ก็จะเป็นตัวแทนของสัญญาณนี้ ใน Wavelet Domain แต่ใน อนุกรมเวฟเล็ตเวลาต่อเนื่องนี้จะใช้เฉพาะค่าของ scale ที่อยู่ในช่วงบวกเท่านั้น ($a_0 > 0$) อย่างไรก็ดีข้อกำหนดนี้ก็ไม่ได้สำคัญมากนัก เนื่องจากสามารถที่จะกำหนด Mother wavelet ที่เป็นลบ ($-g(x)$) แทนได้ ซึ่งจะให้ผลลัพท์เช่นเดียวกัน

จากค่าของการเลื่อนตำแหน่ง ที่เป็นไปตามสมการ $b = nb_0a_0^m$ ซึ่งจะเห็นได้ว่าค่าของการเลื่อนตำแหน่งนี้ จะสัมพันธ์กันกับค่าของ Scaling ดังแสดงในรูปที่ 2.8



รูปที่ 2.8 แสดง time-scale resolution ของอนุกรมเวฟเล็ตเวลาต่อเนื่อง (CTWS)

จากตัวอย่างดังรูปที่ 2.4 ที่ผ่านมาแล้ว ที่ค่าของการ scale มีค่ามาก เวฟเล็ต ที่ได้จะเป็น mother wavelet ที่ถูกยืดออกตามแกนของเวลา ซึ่งคาบเวลาที่นานขึ้น นี้จะทำให้ความละเอียดตามแกนของเวลาลดลง ดังช่องในตอนล่างของรูปที่ 2.8 และสำหรับที่ค่าของการ scale มีค่าน้อย เวฟเล็ต ที่ได้จะเป็น mother wavelet ที่ลดขนาดลงตามแกนของเวลา คาบเวลาที่ลดลงนี้จะทำให้มีความละเอียดตามแกนของเวลามากขึ้น แต่ความละเอียดทางความถี่จะลดลง ดังช่องในส่วนบนของรูป ช่องสี่เหลี่ยมที่แสดงในภาพนี้ จะถูกกำหนดโดยขนาดของ mother wavelet ที่ค่าของ scale และ translate ต่างๆ นั้นเอง จะสังเกตได้ว่าช่องสี่เหลี่ยมนี้จะมีขนาดไม่เท่ากัน ที่ค่าของ scale ต่างๆ แต่ที่ระดับ scale เดียวกันแล้วนั้นขนาดของช่องจะเท่ากัน ซึ่งจะเปรียบเทียบได้กับตัวกรองสัญญาณแบบ แบนพาส (Band pass filter) ที่มีค่า ของ Q คงที่นั้นเอง (Q มีค่าเท่ากับ Center frequency / Bandwidth) ดังนั้นในส่วนบนของภาพซึ่งจะหมายถึงความถี่สูง เพื่อให้ค่าของ Q คงที่ ค่าของ bandwidth จึงต้องมากด้วย และเมื่อพิจารณารูปที่ 2.8 นี้ในแกนของความถี่จะเห็นได้ว่าที่ค่าของ center frequency ที่ต่ำลงมา ค่าของ bandwidth จะลดลงครึ่งหนึ่ง

และนอกจากนี้รูปที่ 2.8 ยังจะแสดงถึง การเกิดและ ลำดับของ wavelet coefficient ด้วย จากที่ได้กล่าวถึงก่อนหน้านี้ ช่องสี่เหลี่ยมแต่ละช่อง จะเป็นตัวแทนของ mother wavelet ที่ scale และ translation ต่างๆ กัน ซึ่งจะนำไปสู่ค่าของ wavelet coefficient นั้นๆ (ในปฐิมภูมิของเวฟเล็ต $W_g(m,n)$ ที่มีค่า ของ scale และ translation แบบ เต็มหน่วย) ค่าของ wavelet coefficient ที่ระดับของ scale ค่ามากๆ ก็จะใช้เวลาประมวลผลที่นาน จาก mother wavelet ที่ยืดออกนั้นเอง ดังนั้นค่าของสัมประสิทธิ์ที่สัมพันธ์กับระดับ scale ที่มากนี้ จะได้จำนวนของสัมประสิทธิ์ต่อหน่วยเวลา ที่น้อยกว่า ค่าสัมประสิทธิ์ ที่ระดับ scale น้อยๆ เนื่องจาก ขนาดของ mother wavelet ที่หดสั้นเข้า จึงใช้เวลาประมวลผลที่น้อย และเมื่อพิจารณาไปตามแกนของเวลา จะพบว่าค่าของ

สัมประสิทธิ์ ที่ระดับ scale ที่มาก จะมีค่าซ้ำเดิมในขณะที่ ค่าสัมประสิทธิ์ ที่ระดับ scale น้อยๆ นั้นจะเปลี่ยนไปเรื่อยๆ ดังนั้นเพื่อเป็นลดการซ้ำกันของค่าสัมประสิทธิ์ จึงต้องตัดส่วนที่มีค่าซ้ำกันนี้ออกไป ซึ่งเรียกว่า sub sampling นั้นเอง

2.7 อนุกรมเวฟเล็ตเต็มหน่วยเวลา (Discrete Time Wavelet Series)

นอกเหนือจากการแปลง เวฟเล็ต ที่มีค่าของ scale และ translation แบบเต็มหน่วยแล้ว ค่าของแกนเวลาก็สามารถเป็นแบบเต็มหน่วยได้เช่น ลำดับของตัวเลข สามารถนำมาใช้กับ wavelet transform ได้ ซึ่งจะเรียกว่าเป็น อนุกรมเวฟเล็ตเต็มหน่วยเวลา (Discrete Time Wavelet Series : DTWS) การแปลงแบบ DTWS นี้ก็จะเปรียบเทียบได้กับ การแปลง Fourier แบบเต็มหน่วยเวลานั้นเอง สมการของการแปลงแบบ DTWS นี้ก็จะได้จากสมการของ CTWS โดยเปลี่ยนจากการ Integration มาเป็น Summation

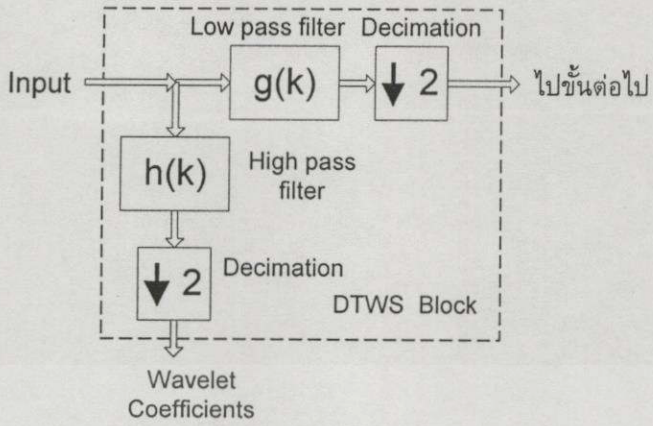
ใน Discrete Time Wavelet Series นี้ก็จะนิยาม Mother wavelet ให้เป็นแบบเต็มหน่วยเวลาด้วย (Discrete Mother wavelet : $h(k)$) สมการของ DTWS จึงเป็น

$$W_h f(m,n) = \frac{1}{\sqrt{a_0^m}} \sum_{k=-\infty}^{\infty} f(k) h\left(\frac{k - nb_0 a_0^m}{a_0^m}\right)$$
$$= a_0^{-m/2} \sum_{k=-\infty}^{\infty} f(k) h(a_0^{-m} k - nb_0) \quad \text{.....(2.15)}$$

จากสมการข้างต้น จะใช้สัญลักษณ์ $h(k)$ แทนตัว Mother wavelet เพื่อให้สอดคล้องกับทฤษฎีของการกรองสัญญาณ และจากการที่ $h(k)$ เป็นแบบเต็มหน่วยเวลา ดังนั้นที่ค่าของ k ที่ไม่เป็นจำนวนเต็ม ค่าของ $h(k)$ ก็จะได้การประมาณค่า

2.8 รูปแบบของการแปลงเวฟเล็ตแบบเต็มหน่วยเวลา

การแปลงเวฟเล็ตแบบ เต็มหน่วยเวลานี้ ทั้งค่าใน time domain และ scale-translation domain (wavelet domain) จะเป็นเลขจำนวนเต็มหน่วย ที่ค่าอัตราส่วนการย่อของแกนเวลา (scale) เท่ากับสอง แล้วการทำ Scaling ด้วยค่าอัตราส่วนของสองนี้สามารถจะกระทำได้ง่าย และมีประสิทธิภาพ ได้โดยการตัด sampling ของข้อมูลนั้น ออกไปครึ่งหนึ่ง (ในตัวประมวลผลสัญญาณเชิงเลข จะทำโดยการใช้ตัวชี้ตำแหน่งข้อมูลที่มี บิตสุดท้ายเป็น '0') วิธีการนี้จะเรียกว่า decimation หรือ sub sampling ด้วยค่าของ 2 รูปแบบทั่วไปของการแปลง เวฟเล็ตที่มีค่าของการย่อขนาดเป็นค่า กำลังของ 2 ก็จะได้คือ

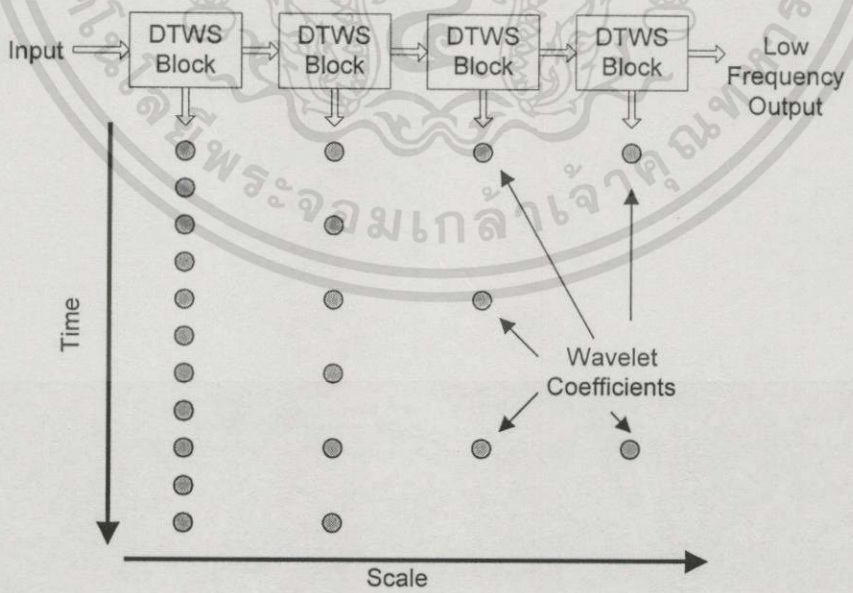


รูปที่ 2.9 รูปแบบของขบวนการ DWTS ที่มีค่าของการ scaling เท่ากับ 2

ขบวนการดังรูปที่ 2.9 นี้ สามารถจะนำไปใช้สำหรับการแปลงเวฟเล็ต แบบ Multiresolution, Orthogonal, Biorthogonal และ Perfect Reconstruction – Quadrature Mirror Filter (PR-QMF) ได้ ซึ่งในแต่ละแบบนี้จะแตกต่างกันที่ คุณสมบัติของ filter coefficient ที่ใช้

ตัวกรอง จะถูกแบ่งออกเป็น low pass filter และ high pass filter ที่กำหนดด้วย ค่าสัมประสิทธิ์ $g(k)$ และ $h(k)$ ตามลำดับ ซึ่งค่าสัมประสิทธิ์ ของตัวกรองนี้ ก็จะสัมพันธ์ กับเวฟเล็ตแม่ ของการแปลงเวฟเล็ต นั่นเอง โดยเมื่อ high pass filter $h(k)$ คือเวฟเล็ตแม่ ดังนั้นเอาท์พุทที่ได้จาก high pass filter นี้ ก็จะคือ wavelet coefficient

ขบวนการของ "การแตกกระจายเวฟเล็ต" (Wavelet decomposition) จะเป็นดังรูปที่ 2.10

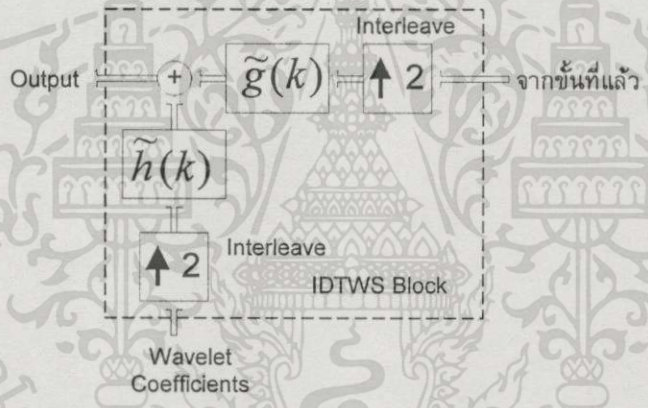


รูปที่ 2.10 Wavelet decomposition ด้วย DWTS Block

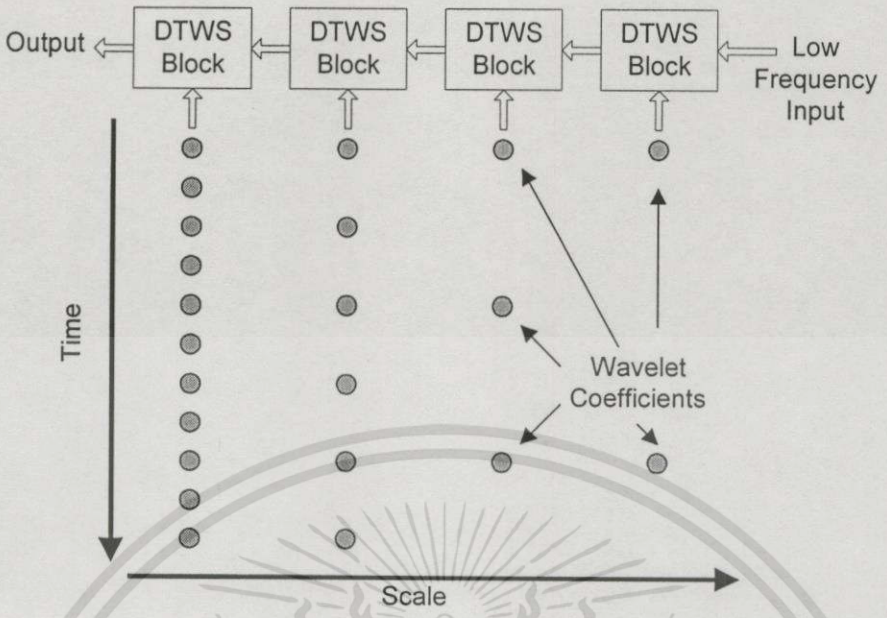
ขบวนการทั้งหมดก็จะเกิดจากการนำเอา block ย่อยๆ ของ DTWS ที่มีค่าของการ scaling เท่ากับ 2 มาต่อเข้าด้วยกัน สัญญาณเอาท์พุทที่ได้จาก block แรก ในส่วนของ low pass filter และผ่านการ decimation แล้ว จะไปเป็นสัญญาณอินพุทให้ส่วนต่อไป อาจกล่าวได้ว่า low pass filter นี้ก็คือ scaling function นั้นเอง

จากรูปที่ 2.10 เห็นได้ว่าสัญญาณ อินพุท จะถูกป้อนเข้าทางด้านซ้ายมือ ค่าของ wavelet coefficient จะได้ออกมาทางด้านล่าง ซึ่งจะมีจำนวนที่ลดเรื่อยๆ และ สุดท้าย ส่วนของสัญญาณความถี่ต่ำที่เหลือจะได้ออกมาทางขวามือ สัญญาณและ ค่าสัมประสิทธิ์ ทั้งหมดที่ได้ออกมานี้ ก็จะเป็นตัวแทนของสัญญาณใน time domain นั้นเอง

การแปลงผกผันเวฟเล็ต หรือ "การรวมกลับเวฟเล็ต" (Wavelet reconstruction) แบบเต็มหน่วยเวลา (inverse discrete time wavelet series : IDTWS) ก็สามารถกระทำได้โดยวิธีการดังรูปที่ 2.11 และ 2.12



รูปที่ 2.11 รูปแบบของขบวนการ IDWTS ที่มีค่าของการ scaling เท่ากับ 2



รูปที่ 2.12 การรวมกลับเวฟเล็ตด้วย IDTWS block

จากรูป ซึ่งคล้ายกับ การแปลงเวฟเล็ต ดังที่กล่าวมาแล้ว แต่จะมีการทำงานที่กลับกัน block ของ IDTWS นี้จะรับสัญญาณความถี่ต่ำ และค่าของ wavelet coefficient เพื่อนำมาสังเคราะห์ หรือสร้างสัญญาณ ใน time domain ออกมา สำหรับการ decimation หรือ sub sampling ด้วยค่าอัตราส่วนของ 2 นี้ จะถูกแทนด้วยขบวนการที่กลับกัน คือ ขบวนการ interleave หรือ up sampling ที่จะแทรกข้อมูล 0 เข้าไปแต่ละ sampling คุณสมบัติของ low pass filter และ high pass filter ที่ใช้นี้ จะได้กล่าวถึงต่อไป

2.9 Multi resolution wavelet transform

จากการทำ wavelet transform แบบที่มีเงื่อนไขเฉพาะต่างๆ การทำ wavelet transform แบบ Multi resolution wavelet transform ดูจะเป็นกรณีที่มีการใช้กันมากที่สุด ในการทำ Multiresolution wavelet transform นั้นฟังก์ชัน ที่จะนำมาใช้เป็น mother wavelet อาจมีคุณสมบัติเป็น Non orthogonal หรือ อื่นๆ ก็ได้

ข้อกำหนดเบื้องต้นของตัว Mother wavelet หรือ high pass filter ก็คือมันจะต้องสัมพันธ์กันกับ ฟังก์ชันที่เป็น scaling function หรือ low pass filter ด้วย การทำ Multi resolution transform นี้ จะมีคุณสมบัติ ของจำนวนของ ค่าสัมประสิทธิ์ ที่ได้ออกมาเป็นแบบ pyramidal structure อยู่ในตัว ซึ่งคุณสมบัติข้อนี้ไม่จำเป็นสำหรับการทำ wavelet transform โดยทั่วไป แต่ด้วยคุณสมบัติของ pyramidal structure นี้ทำให้ในการแปลง เวฟเล็ต ที่ระดับ scale ต่างๆ

สามารถที่จะทำได้โดยใช้ scaling function (low pass filter) และ wavelet function (high pass filter) เพียงตัวเดียวได้ ซึ่งจะสามารถทำได้โดยง่ายในทางปฏิบัติด้วยการทำงานซ้ำ

โดยทั่วไปในการใช้งานของ Multiresolution wavelet transform จะเสมือนกับเป็นการกรองสัญญาณอินพุทที่เข้ามาให้แยกออกเป็นสองส่วนคือ ส่วนที่มีความถี่สูง (high frequency band) และ ส่วนที่มีความถี่ต่ำ (low frequency band) การกรองนี้ก็จะเริ่มจาก bandwidth ทั้งหมดของสัญญาณก่อน ซึ่งในการกรองนี้ก็จะได้สัญญาณส่วนที่มีความถี่สูง (มีค่า scaling ต่ำ) ออกมา จากนั้นสัญญาณอินพุท จะถูกลด bandwidth ลง ในแต่ละขั้นของการแปลง (ที่ค่าของ bandwidth น้อยๆ ก็จะสัมพันธ์กับค่าของการ scaling ที่มาก) จากรูปที่ 2.10 สัญญาณเอาท์พุท ที่ได้จาก high frequency band ก็จะคือ wavelet coefficient ที่ ความละเอียดสูง (fine scaling) และสำหรับสัญญาณเอาท์พุท ที่ low frequency band จะนำไปผ่านการ decimation ด้วยค่าอัตราของ 2 จากนั้น สัญญาณส่วนที่เหลือนี้ จะถูกแบ่งออกเป็น high และ low frequency band อีกครั้งหนึ่ง สำหรับสัญญาณ เอาท์พุทที่ได้จาก การทำ high pass filter นี้ แต่ละจุดของ Sampling ก็จะเป็น wavelet coefficient ที่ Translation ต่างๆ ที่ค่าระดับ Scale เดียวกันนั้นเอง

การกรองสัญญาณแยกออกเป็นสองส่วนนี้สามารถกระทำต่อไปได้เรื่อยๆ จนกระทั่งได้สัญญาณเอาท์พุท จาก low pass filter สุดท้าย ซึ่งคือองค์ประกอบทาง DC ของสัญญาณนั้นๆ เอง

2.9.1 Orthogonal Wavelet Transform

เช่นเดียวกันกับการทำ Fourier analysis หรือการแปลงสัญญาณในแบบอื่นๆ คุณสมบัติที่สำคัญจะเกิดขึ้นได้ เมื่อองค์ประกอบที่เป็นพื้นฐานของระบบนั้นๆ มีคุณสมบัติเป็น Orthogonal (Orthogonal basis) โดยการฉายภาพของสัญญาณอินพุท ลงบน Orthogonal basis แล้วค่าสัมประสิทธิ์ที่ได้ ก็จะมีคุณสมบัติของความเป็น Orthogonal ด้วย

ในกรณีของการแปลงเวฟเล็ตนั้นก็เช่นเดียวกัน ค่าสัมประสิทธิ์เวฟเล็ตที่ได้แต่ละตัว ซึ่งเป็นตัวแทนของสัญญาณอินพุท ในส่วนที่สัมพันธ์กับ เวฟเล็ตนั้นๆ ซึ่งถ้าเวฟเล็ตแม่ มีคุณสมบัติของการเป็น Orthogonal แล้ว เซตของค่าสัมประสิทธิ์ที่ได้ ก็จะไม่มีความซ้ำซ้อนกัน (No redundancy) ทำให้เซตของค่าสัมประสิทธิ์ที่ได้นี้ เป็นตัวแทนของสัญญาณที่มีประสิทธิภาพ คุณสมบัติของการเป็น Orthogonal หรือ Biorthogonal ของการแปลงเวฟเล็ตนี้ จึงถูกนำมาใช้กันมากในการทำ Multiresolution wavelet decomposition และโดยเฉพาะการนำมาใช้ ในการลดขนาดข้อมูลภาพ หรือในการวิเคราะห์ข้อมูล

การแปลงเวฟเล็ตที่มีคุณสมบัติของ Orthogonal หรือ Biorthogonal นี้จะเกิดขึ้นได้จาก การใช้ เวฟเล็ตแม่ที่มีคุณสมบัติของ Orthogonal หรือ Biorthogonal นั้นเอง ในการวิเคราะห์

สัญญาณ ด้วยวิธีการของ Multiresolution analysis ความเป็น Orthogonal จะกำหนดโดยฟังก์ชันของ low pass filter หรือ Scaling function จะต้องมีคุณสมบัติของ Orthogonal ที่ตำแหน่งของ Scale-Translation (m,n) ซึ่งขึ้นกับ a_0 และ b_0 นั้นเอง

ฟังก์ชันของเวฟเล็ตแม่ที่มีคุณสมบัติของ Orthogonal นั้น ค่าผลคูณภายใน (Inner product) ของเวฟเล็ตแม่ที่ Scale-Translation ต่างๆ จะได้เป็น Impulse ในปฐิมภูมิของ Scale และ Translation

$$\int_{-\infty}^{\infty} g_{m,n}(t) g^{*}_{m,n} dt = \delta(m-m')\delta(n-n')$$
$$= \begin{cases} 1 \text{ if } m = m' \,\&\, n = n' \\ 0 \text{ otherwise} \end{cases} \quad \text{.....(2.16)}$$

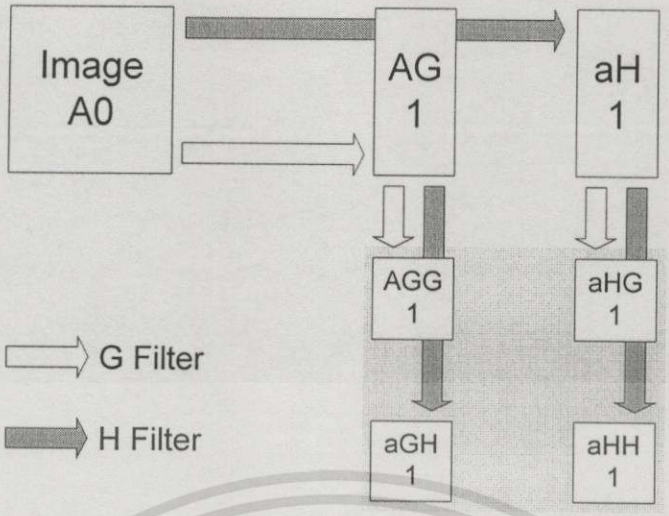
คุณสมบัติเฉพาะของการแปลงแบบ Orthogonal คือตัวกรอง (high pass filter และ low pass filter) ที่ใช้ในขบวนการ วิเคราะห์สัญญาณ (รูปที่ 2.9) กับตัวกรอง ที่ใช้ในการ สังเคราะห์สัญญาณ (รูปที่ 2.11) จะเป็นตัวเดียวกัน

2.10 การประยุกต์ใช้กับข้อมูลภาพ

จากคุณสมบัติของ Orthogonal wavelet transform ที่เหมาะในการนำมาประยุกต์ใช้กับการลดขนาดข้อมูลภาพ ในหัวข้อนี้จึงจะกล่าวถึงวิธีนำ การแปลงเวฟเล็ตมาใช้งาน ขบวนการแปลงเวฟเล็ตที่นำมาใช้งานนี้จะเป็นวิธีการของ Multi resolution wavelet transform ที่ได้อธิบายในหัวข้อ 2.9

จากที่ข้อมูลภาพ ที่เป็นสัญญาณ ในสองมิตินี้ ขบวนการแปลงเวฟเล็ตที่ใช้จะต้องเป็นสองมิติด้วย แต่เพื่อความง่ายในการใช้งาน ข้อมูลภาพจะถูกแปลงเป็นสัญญาณข้อมูลใน หนึ่งมิติแทน เพื่อให้สามารถใช้ขบวนการแปลงเวฟเล็ต ที่กล่าวมาแล้วได้ โดยการจัดเรียงของแต่ละจุดภาพใหม่ แยกตามแนวแกนตั้ง และแกนนอน ข้อมูลที่ได้จากเส้นภาพในแนวแกนนอนแต่ละเส้นจะเป็นสัญญาณหนึ่งมิติ ซึ่งจะถูกกระทำด้วยการแปลงเวฟเล็ตจนครบทุกเส้นภาพ ผลที่ได้ออกมาจะเป็นค่าสัมประสิทธิ์ของเวฟเล็ต ของสัญญาณ ในแต่ละเส้นภาพที่ไม่เกี่ยวข้องกัน แล้วข้อมูลนี้จะถูกนำไปผ่านขบวนการแปลงเวฟเล็ตอีกครั้ง ตามแนวตั้ง [3]

จากที่ในการแปลงเวฟเล็ตแบบ Multiresolution ที่มีค่าอัตราส่วนในการ Scaling เท่ากับสองนี้ ผลลัพท์ที่ได้จะแบ่งออกเป็นสองส่วนคือ ส่วนของ high pass filter และ low pass filter ดังนั้น เมื่อข้อมูลภาพถูกนำมาทำการแปลงเวฟเล็ต ค่าสัมประสิทธิ์ที่ได้จึงแบ่งออกเป็น 4 ส่วน แสดงดังรูปที่ 2.13

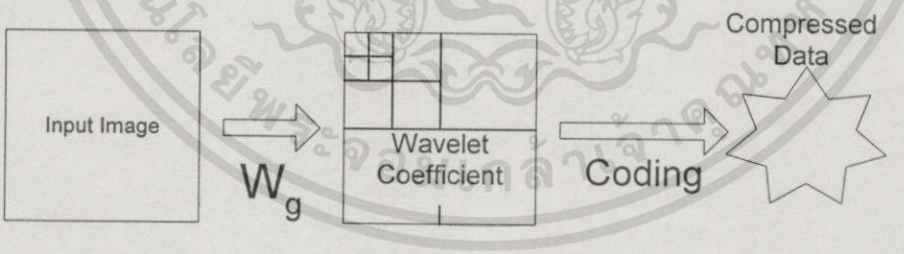


รูปที่ 2.13 แสดงการแปลงเวฟเล็ตของข้อมูลภาพออกเป็น 4 Sub band คือ AGG1 aGH1 aHG1 aHH1 (ส่วนที่แรเงา) ซึ่งองค์ประกอบทั้ง 4 นี้ยังคงมีจำนวนจุดข้อมูลรวมกันทั้งหมดเท่ากับภาพต้นแบบ (A0)

จากนั้น ส่วนของข้อมูลความถี่ต่ำที่เหลืออยู่ (AGG1) จะสามารถนำไปแตกกระจายต่อไปได้เช่น เดียวกันกับการแปลงเวฟเล็ตของสัญญาณ หนึ่งมิติ

2.11 การประยุกต์ใช้ Wavelet transform ในการลดขนาดข้อมูล

ขบวนการลดขนาดข้อมูลจะกระทำกับค่าสัมประสิทธิ์เวฟเล็ตที่ได้ จากขบวนการ Analysis ดังรูปที่ 2.14



รูปที่ 2.14 รูปแสดงขั้นตอนการลดขนาดข้อมูล

ค่าสัมประสิทธิ์ที่ได้ ซึ่งเป็นส่วนประกอบย่อยของสัญญาณ จะถูกนำมาผ่านขบวนการเข้ารหัส (Coding) ในขั้นตอนของการเข้ารหัสนี้ อาจข้อมูลบางส่วนที่สูญเสียไป หรือเกินออกมา จากนั้นในขั้นตอนของการ Synthesis จะสามารถสร้างสัญญาณเอาท์พุท ที่มีลักษณะที่ใกล้เคียงกับสัญญาณอินพุทได้ ทั้งนี้ก็ขึ้นอยู่กับว่า ในขั้นตอนของการ ลดขนาดข้อมูล มีความผิดพลาดเกิดขึ้นเพียงใด สำหรับรายละเอียดของขบวนการลดขนาดข้อมูลนี้ จะได้อธิบายในบทต่อไป

บทที่ 3
การลดขนาดข้อมูลภาพ

ในช่วงหลายปีที่ผ่านมานี้ ความต้องการของการลดขนาดข้อมูลภาพมีเพิ่มขึ้นอย่างต่อเนื่อง ตัวอย่างเช่น การลดขนาด ข้อมูลภาพที่เป็นส่วนสำคัญของการใช้งานคอมพิวเตอร์มัลติมีเดีย (Computer Multimedia) การประชุมด้วยภาพระยะไกล (Televideo Conferencing) รีโหมดเซนซิ่ง (Remote Sensing) ภาพถ่ายทางการแพทย์ (Medical Image) การส่งโทรสาร (Facsimile) และอื่นๆ อีกมากที่ต้องใช้ความสามารถในการจัดการ จัดเก็บ และการส่งข้อมูลภาพ ขบวนการลดขนาดข้อมูลจะเป็นสิ่งที่จำเป็นทั้งสิ้น

จากการที่ข้อมูลภาพซึ่งเป็นสองมิติ (2-D) ที่ได้มาจากการสุ่ม (Sampling) และ ควอนไตซ์ (Quantization) ของวัตถุที่แสงตกกระทบ เข้ามาเป็นข้อมูลดิจิตอลในคอมพิวเตอร์นั้น ข้อมูลของภาพที่ได้จะมีปริมาณมาก ซึ่งจะเป็นอุปสรรคในการจัดเก็บ การประมวลผล และในการสื่อสารข้อมูลภาพนี้

ขบวนการลดขนาดข้อมูลภาพ (Image Compression) จะเป็นการลดจำนวนของข้อมูลที่จะใช้แทนภาพนั้นๆ ลง โดยมีหลักการคือ การตัดข้อมูลส่วนที่เกินความจำเป็นออกไป จึงทำให้ข้อมูลที่เหลือนั้นลดลงได้ โดยจะกระทำก่อนที่ข้อมูลภาพจะถูกจัดเก็บในอุปกรณ์บันทึกข้อมูล หรือก่อนที่จะใช้ในการสื่อสาร และในเวลาต่อมาเมื่อต้องการใช้งาน ข้อมูลจะถูกแปลงผกผันออกมา (Decompression) เป็นภาพฯ เดิม หรือเป็นภาพที่ใกล้เคียงกับภาพเดิม เพื่อนำไปใช้ต่อไป

3.1 หลักการพื้นฐาน

ขบวนการลดขนาดข้อมูล (Data compression) จะหมายถึง ขบวนการที่ใช้ในการทำให้ข้อมูล (Data) ที่ต้องใช้แทนข่าวสารหนึ่งๆ นั้นลดลง ซึ่งสามารถเปรียบเทียบข้อมูลได้กับตัวหนังสือที่จะสื่อความหมายถึงเนื้อหาสาระภายในหนังสือเล่มหนึ่งๆ นั้นเอง ในกรณีของหนังสือสองเล่มที่แต่งโดยคนละคนกันแต่มีเนื้อหาที่เหมือนกันนั้น จำนวนตัวอักษร (ข้อมูล) ที่ใช้ในการบอกเล่าจะไม่เท่ากันก็ได้ นั้นก็แสดงว่าหนังสือที่ผู้แต่งใช้จำนวนของตัวอักษรที่มากกว่าจะต้องมีคำหรือประโยคบางประโยค ที่ส่วนเกินความจำเป็น เช่น อาจเป็นประโยคที่บอกเล่าสิ่งที่ได้กล่าวถึงมาแล้วก่อนหน้านี้ หรือเป็นคำ/ประโยคที่ไม่ได้สื่อความหมายใดๆ

การลดขนาดข้อมูลภาพก็เช่นเดียวกัน ข้อมูลของระดับความสว่างจุดภาพแต่ละจุดร่วมกันเพื่อ สื่อถึงความหมายของภาพ ก็จะมีส่วนที่เกินความจำเป็นที่สามารถตัดออกไปได้

ถ้ากำหนดให้ n_1 และ n_2 เป็นจำนวนของข้อมูลที่ใช้ในการสื่อความหมายของภาพฯ หนึ่ง อัตราส่วนการลดขนาดของข้อมูล (Data compression ratio) จะคำนวณได้คือ

$$\text{Data compression ratio} = \frac{n_1}{n_2} \quad \text{.....(3.1)}$$

และอีกวิธีการหนึ่งในการวัดค่าการลดขนาดของข้อมูลภาพที่นิยมใช้กันคือ การวัดจำนวนบิต (Bit) ของข้อมูลที่ต้องใช้แทนระดับความสว่างของจุดภาพใดๆ ของภาพฯ นั้น (Bit Per Pixel : bpp) โดยทั่วไปแล้วภาพระดับสีเทา 256 ระดับจะต้องใช้จำนวนบิตข้อมูลต่อหนึ่งจุดภาพเท่ากับ 8 บิต (8 bpp) เมื่อนำมาผ่านขบวนการลดขนาดข้อมูลแล้วจำนวนบิตที่ต้องใช้แทนระดับความสว่างนี้จะลดลง อาจเหลือเพียง 1.1 bpp ได้โดยที่ระดับความสว่างยังคงมีได้ 256 ระดับ เท่าเดิม

วิธีการลดขนาดข้อมูลภาพ จะสามารถแบ่งออกได้ตามหลักการที่ใช้ในการตัดข้อมูลในส่วนที่ไม่จำเป็นในภาพออกไป ได้สามวิธีการใหญ่คือ

1. ความซ้ำซ้อนของรหัสข้อมูล (Coding Redundancy)
2. ความซ้ำซ้อนของข้อมูลระหว่างจุดภาพ (Inter Pixel Redundancy)
3. ข้อมูลที่เกินจำเป็นในการรับรู้ของมนุษย์ (Psychovisual Redundancy)

จะสามารถอธิบายรายละเอียดในแต่ละหัวข้อได้ดังต่อไปนี้ [7]

3.1.1 ความซ้ำซ้อนของรหัสข้อมูล (Coding Redundancy)

จากข้อมูลภาพซึ่งประกอบขึ้นจากจุดภาพที่มีระดับความสว่างต่างๆ กัน ข้อมูลของแต่ละจุดภาพนี้จะถูกแทนด้วยรหัสที่แตกต่างกันไปตามระดับควาสว่างเช่น ถ้าเป็นข้อมูลของภาพระดับสีเทา 256 ระดับแล้วนั้น โดยทั่วไป แล้วจะใช้รหัสเลขฐานสอง (Binary number) ขนาด 8 บิต ในการแทนข้อมูลของหนึ่งจุดภาพตามระดับความสว่างเป็นต้น

ในภาพหนึ่งๆ จำนวนข้อมูลของระดับความสว่างแต่ละระดับ จะมีเกิดขึ้นมากน้อยไม่เท่ากัน โดยจะสังเกตได้จาก การทำ ฮิสโตแกรม (Histogram) ของภาพ ซึ่งจะเห็นได้ว่าจำนวนของข้อมูลหรือ รหัส ในแต่ละระดับความสว่างจะมีไม่เท่ากัน

จาก Histogram จะสามารถหาค่าความน่าจะเป็น (P_r) ในการเกิดขึ้นของรหัสต่างๆ (r_k) ได้ตามสมการ

$$P_r(r_k) = \frac{n_k}{n} \; ; \; k = 0,1,\ldots,L\text{-}1 \quad \text{.....(3.2)}$$

เมื่อ L จำนวนของระดับความสว่างทั้งหมด

n_k จำนวนที่พบรหัสที่ระดับความสว่าง k

n จำนวนของข้อมูลทั้งหมด

ถ้าให้ขนาดความยาวของรหัส (จำนวนบิต) ที่ใช้ในการแทนระดับความสว่าง (r_k) แต่ละระดับเป็น $l(r_k)$ ดังนั้นค่า เฉลี่ยความยาวของรหัส (L_{avg}) ที่ใช้จะคำนวณได้จาก

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k).P_r(r_k) \quad \text{.....(3.3)}$$

เมื่อ L_{avg} ค่าความยาวเฉลี่ยของรหัสต่อหนึ่งจุดภาพ (bpp)

ดังนั้นจำนวนบิตของข้อมูลทั้งหมดที่ต้องใช้ในการเข้ารหัสภาพ ซึ่งมีขนาด n จุดภาพ จะเท่ากับ $L_{avg} * n$

ในการเข้ารหัสระดับความสว่าง โดยใช้เลขฐานสองขนาด m บิต จะทำให้ค่าของ L_{avg} เท่ากับ m ด้วยเพราะ การใช้รหัสเลขฐานสองขนาด m บิต ค่าของ $l(r_k)$ จะเท่ากับ m คงที่ เมื่อนำมาแทนค่าในสมการที่ 3.3 ค่าคงที่ m จะสามารถยกออกมาได้ และผลรวมของค่าความน่าจะเป็นทั้งหมดจะเท่ากับหนึ่ง ดังแสดงได้ในตารางที่ 3.1

ตารางที่ 3.1 แสดงตัวอย่างของรหัสความยาวไม่คงที่

r_k	$P_r(r_k)$	แบบที่ 1	$l_1(r_k)$	แบบที่ 2	$l_2(r_k)$
r_0	0.19	000	3	11	2
r_1	0.25	001	3	01	2
r_2	0.21	010	3	10	2
r_3	0.16	011	3	001	3
r_4	0.08	100	3	0001	4
r_5	0.06	101	3	00001	5
r_6	0.03	110	3	000001	6
r_7	0.02	111	3	000000	6

จากข้อมูลภาพที่มี 8 ระดับความสว่าง ซึ่งในแต่ละระดับมีค่าความน่าจะเป็นที่จะเกิดขึ้นคือ P_r ในตารางที่ 3.1 ถ้าเลือกใช้รหัสเลขฐานสอง ขนาด 3 บิต เป็นตัวแทนของระดับความสว่างแต่ละระดับ (แบบที่ 1) จะได้ ค่าความยาวเฉลี่ยของรหัสต่อหนึ่งจุดภาพ (bpp) คือ L_{avg} = 3 บิต จากค่าของ $l_1(r_k)$ = 3 บิตคงที่ แต่ถ้าเลือกใช้รหัสที่มีขนาดความยาวของรหัสที่ไม่คงที่ (Variable-Length Coding, แบบที่ 2) ค่าความยาวเฉลี่ยของรหัสต่อหนึ่งจุดภาพ L_{avg} จะเป็น

$$\begin{aligned} L_{avg} &= \sum_{k=0}^{7} l_2(r_k).P_r(r_k) \\ &= 2(0.19) + 2(0.25) + 2(0.12) + 3(0.16) + \\ &\quad 4(0.08) + 5(0.06) + 6(0.03) + 6(0.02) \\ &= 2.7 \text{ Bit} \end{aligned}$$

จากสมการที่ 3.1 จะได้ว่าค่าอัตราการลดขนาดข้อมูล (Data compression ratio) จะเท่ากับ 3 / 2.7 หรือ 1.11 เท่า หรือลดลงประมาณ 10 เปอร์เซ็นต์จากการเข้ารหัสในแบบที่ 1

จากตัวอย่างข้างต้น การเข้ารหัสในแบบที่ 2 นี้ จะเป็นการเข้ารหัสโดยใช้จำนวนบิตที่น้อย กับข้อมูลระดับความสว่างที่มีระดับความน่าจะเป็นสูง และจำนวนบิตที่มากขึ้นกับข้อมูลที่มีค่าความน่าจะเป็นต่ำลงมา การเข้ารหัสแบบนี้จะเรียกว่า Variable-Length Coding

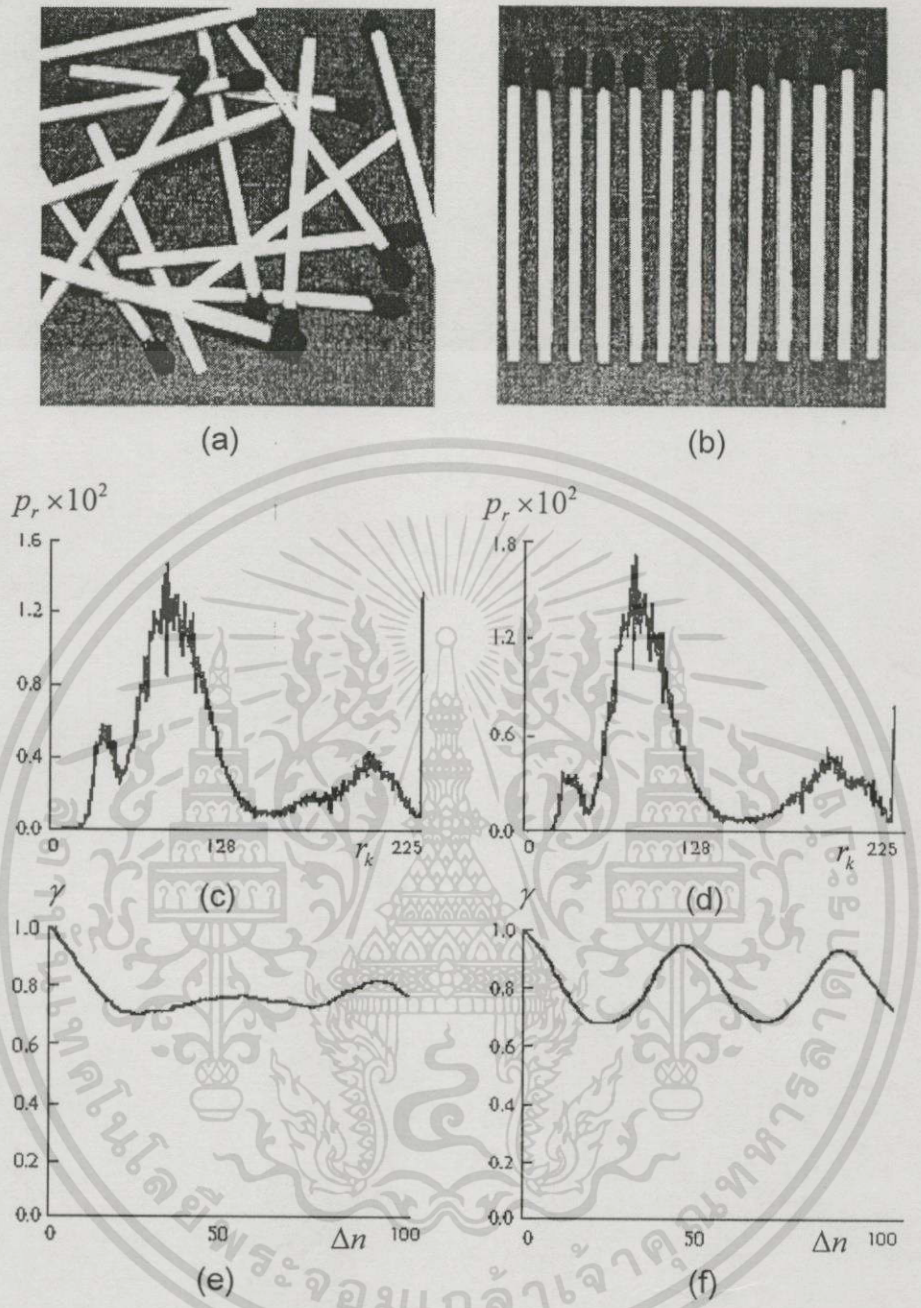
ดังนั้น จึงอาจสรุปได้ว่าในการเข้ารหัสระดับความสว่าง (Gray-Level) ของภาพที่ใช้จำนวนรหัสมากกว่าที่จำเป็นแล้ว จะเรียกว่ารหัสนี้มีความซ้ำซ้อนกัน (Coding Redundancy)

โดยทั่วไปแล้วการซ้ำซ้อนกันของรหัสที่ใช้ จะเกิดขึ้นเมื่อมีการใช้รหัส (Code) แทนเซตของเหตุการณ์ต่างๆ เช่นค่าระดับความสว่างของจุดภาพ โดยไม่ได้คำนึงถึงค่าความน่าจะเป็นในการเกิดขึ้นของเหตุการณ์นั้นๆ ซึ่งส่วนใหญ่แล้วภาพระดับสีเทาจะมีลักษณะการกระจาย (Histogram) ของระดับความสว่างต่างๆ ที่ไม่เท่ากัน เนื่องจากภาพโดยมากจะเป็นภาพเพียงบางส่วนหรือส่วนของวัตถุที่มีระดับความสว่างใกล้เคียงกัน

ดังนั้นการใช้รหัสเลขฐานสอง (Binary Number) ในการแทนระดับความสว่างก็จะไม่เป็นวิธีการที่จะให้ได้จำนวนของข้อมูลที่น้อยที่สุด

3.1.2 ความซ้ำซ้อนของข้อมูลระหว่างจุดภาพ (Interpixel Redundancy)

Interpixel Redundancy จะสามารถอธิบายได้จากตัวอย่างในรูปที่ 3.1 (a) และ (b) ทั้งสองรูปนี้จะมีค่าของ Histogram ที่เหมือนกันดังแสดงในรูปที่ 3.1 (c) และ (d) ตามลำดับ เมื่อสังเกตลักษณะของการแจกแจง (Histogram) ของระดับความสว่างนี้ จะเห็นได้ว่าภาพจะประกอบขึ้นจากจุดภาพที่มีระดับความสว่างอยู่สามช่วงใหญ่ๆ ด้วยกัน ซึ่งจากการที่ค่าของความน่าจะเป็นของระดับความสว่างที่ไม่เท่ากันนี้ สามารถที่จะนำหลักการของ Variable-Length Coding มาใช้เพื่อลดขนาดของรหัสที่ต้องใช้ลงได้



รูปที่ 3.1 (a)(b) แสดงข้อมูลภาพ (c)(d) แสดงการแจกแจงของระดับความสว่างที่พบในภาพ (e) (f) แสดงเปรียบเทียบค่าสัมประสิทธิ์ Auto Correlation ระหว่างภาพสองภาพ

แต่อย่างไรก็ตามขบวนการเข้ารหัสระดับความสว่างนี้ ไม่ได้เปลี่ยนแปลงข้อมูลของจุดภาพเลย ซึ่งที่จริงแล้วข้อมูลของจุดภาพแต่ละจุดนั้น จะมีความสัมพันธ์ซึ่งกันและกันอยู่ ความสัมพันธ์กันของของจุดภาพนี้จะเกิดขึ้นได้จาก โครงสร้างหรือ ลักษณะทางเลขาคณิตของวัตถุที่อยู่ในภาพนั้นๆ

จากการคำนวณหาสัมประสิทธิ์ Auto Correlation ที่ผ่านการ Normalize แล้ว ของเส้นภาพหนึ่งเส้นในภาพ (a) และ (b) โดยใช้สมการของ Normalized Auto Correlation คือ

$$\gamma(\Delta n) = \frac{A(\Delta n)}{A(0)} \quad \text{.....(3.4)}$$

เมื่อ

$$A(\Delta n) = \frac{1}{N - \Delta n} \sum_{y=0}^{N-1-\Delta n} f(x,y).f(x,y+\Delta n) \quad \text{.....(3.5)}$$

x ตำแหน่งของเส้นภาพที่นำมาคำนวณ

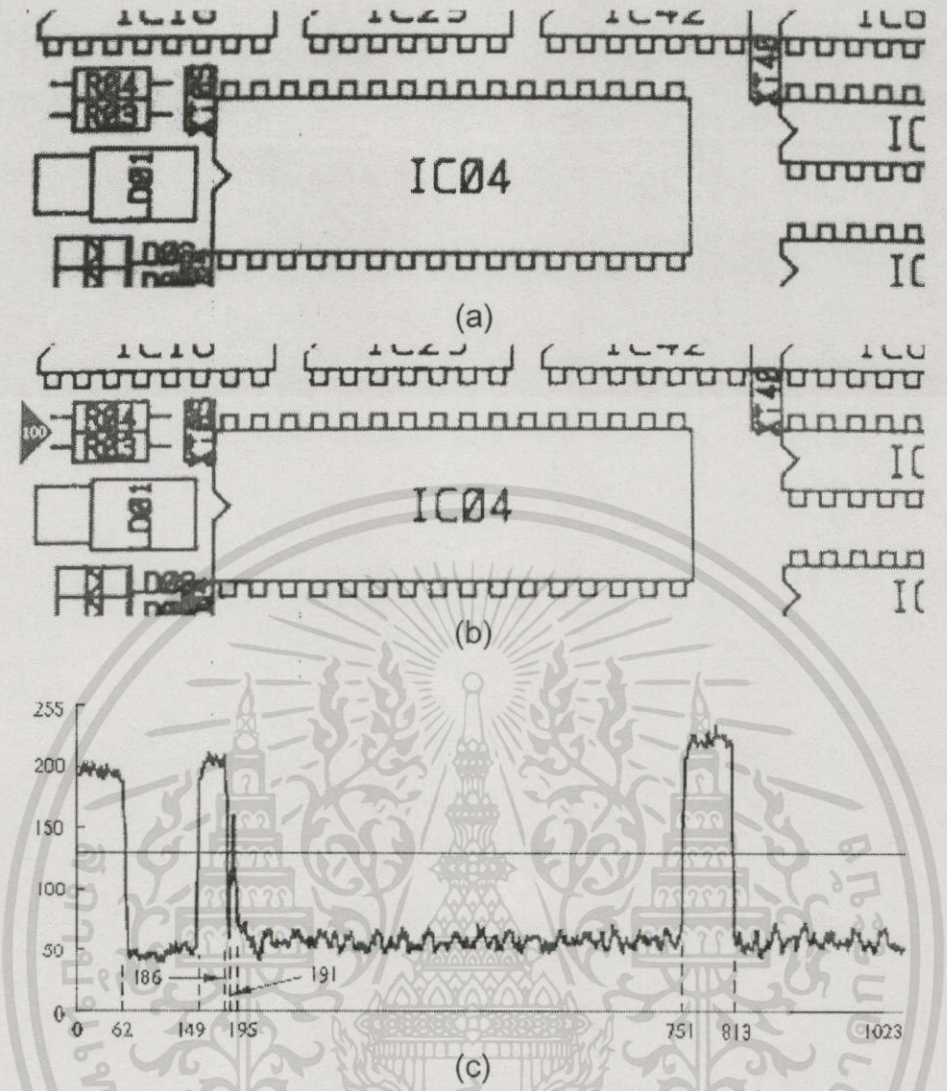
Δn ระยะต่างของจุดภาพที่นำมาเปรียบเทียบ

ผลที่ได้ของสัมประสิทธิ์ Auto Correlation ที่ได้โดยการเปลี่ยนค่าของ Δn จะได้ดังแสดงในรูปที่ 3.1 (e) และ (f) ตามลำดับ ซึ่งแสดงให้เห็น ถึงความแตกต่างกันคือ ที่ค่าของ Δn เท่ากับ 45 และ 90 จุดภาพ ข้อมูลจะมีความสัมพันธ์กันมาก (highly correlated) ซึ่งที่ค่าของ Δn เท่ากับ 45 และ 90 จุดภาพนี้ เป็นระยะเดียวกันกับความห่างของก้านไม้ขีดไฟในภาพพอดี และนอกจากนี้จะเห็นได้ว่าค่าของสัมประสิทธิ์ γ ของเส้นภาพทั้งสองนี้ ที่ค่าของ Δn เท่ากับหนึ่ง จะเป็น 0.9922 และ 0.9928

จากตัวอย่างข้างต้นจะพบว่า การซ้ำกันของข้อมูลภาพ จะเกิดขึ้นได้จากความสัมพันธ์กันระหว่างจุดภาพที่อยู่ติดๆ กัน ซึ่งจะมีค่าที่ใกล้เคียงกัน ดังนั้นค่าของข้อมูลที่จุดภาพใดๆ จะสามารถตัดทิ้งออกไปได้ และจะสามารถสร้างกลับมาใหม่โดยการประมาณค่าจากจุดภาพที่อยู่รอบๆ นั้นเอง และจากการที่ข้อมูลในแต่ละจุดภาพสามารถที่จะคำนวณกลับออกมาใหม่ได้นั้น แสดงว่าข้อมูลในจุดภาพหนึ่งๆ นั้นจะมีความสำคัญน้อยมากต่อความหมายทั้งหมดของภาพ ด้วยเหตุนี้ จึงอาจกล่าวได้ว่าข้อมูลในแต่ละจุดภาพนั้นมีความซ้ำซ้อนกันเอง (Interpixel Redundancy)

การที่จะลด Interpixel Redundancy ที่เกิดขึ้นในภาพ ข้อมูลภาพซึ่งเป็น อาเรย์สองมิติของระดับความสว่าง จะต้องถูกแปลงให้อยู่ในรูปแบบที่เหมาะสมกว่า (ซึ่งข้อมูลที่ถูกแปลงแล้วจะไม่สามารถมองออกเป็นภาพได้) ตัวอย่างเช่น การแทนข้อมูลภาพด้วยผลต่างระหว่างจุดภาพสองจุดที่อยู่ติดกันก็จะแทนข้อมูลภาพทั้งหมดได้ ซึ่งจากที่ข้อมูลระดับความสว่างนั้นมีค่าใกล้กัน จะทำให้จำนวนแบบของรหัสที่ต้องใช้แทนข้อมูลที่ถูกแปลงแล้วลดลง

ขบวนการแปลงข้อมูลเพื่อใช้ในการลด Interpixel Redundancy นี้จะเรียกว่าส่งถ่าย (Mapping) และจะเรียกขบวนการส่งถ่าย นี้ว่าสามารถย้อนกลับได้ (Reversible) ถ้าข้อมูลของภาพต้นแบบ สามารถสร้างกลับออกมาได้ จากข้อมูลที่ผ่านขบวนการส่งถ่ายแล้ว

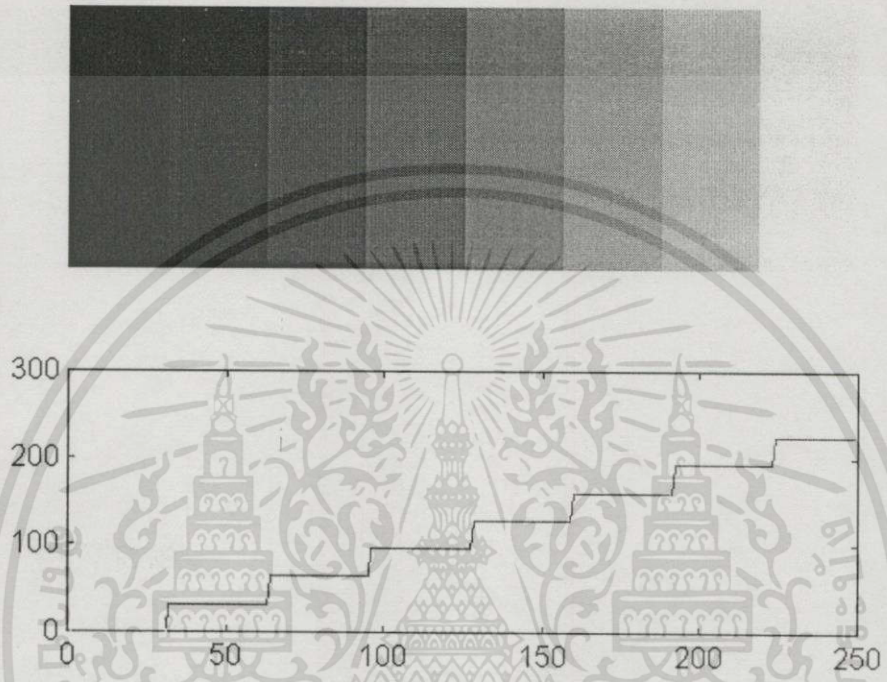


รูปที่ 3.2 แสดงตัวอย่างของ Run-Length Coding

การทำ Mapping สามารถแสดงได้ดังตัวอย่างจากรูปที่ 3.2 (a) เป็นภาพของส่วนประกอบของ วงจรอิเล็กทรอนิกส์ ขนาด 1 X 3 นิ้ว ที่ความละเอียด 330 dpi (Dot Per Inch : จุดต่อนิ้ว) และรูปที่ 3.2 (b) เป็นภาพที่มีสองระดับความสว่าง (Binary Image) การ Mapping ข้อมูลของภาพให้เป็น Binary Image จะทำโดยใช้วิธีการตั้งค่า Threshold ซึ่งแสดงในรูปที่ 3.2 (c) ซึ่งเป็นค่าระดับความสว่างของเส้นภาพหนึ่งเส้น จะเห็นได้ว่า Binary Image ที่ได้จะมีลักษณะที่สังเกตได้คือจะมีจุดภาพที่มีระดับความสว่างเดียวกันอยู่ติดกันเป็นจำนวนมาก ด้วยเหตุนี้จะมีวิธีการที่จะเก็บข้อมูลลักษณะนี้ให้น้อยลงได้โดย จากข้อมูลระดับความสว่างของจุดภาพในเส้นภาพ คือ f(x, 0) f(x, 1) f(x, 2) . . . f(x, N-1) ให้เป็นคู่ลำดับของข้อมูล (g_1, r_1) (g_2, r_2) (g_3, r_3) . . . เมื่อ g_i หมายถึงค่าของระดับความสว่าง และ r_i หมายถึงจำนวนที่พบติดต่อกัน วิธีการดังกล่าวนี้เรียกว่า Run-Length Coding ข้อมูลเมื่อผ่านขบวนการส่งถ่ายนี้แล้วจะมีความซ้ำซ้อนกันน้อยลง และไม่สามารถมองออกเป็นภาพได้

3.1.3 **ข้อมูลที่เกินจำเป็นในการรับรู้ของมนุษย์** (Psychovisual Redundancy)

จากการรับรู้ภาพวัตถุของตามนุษย์ที่ไม่ได้เป็นสัดส่วนโดยตรงกับระดับความสว่าง หรือแสงที่สะท้อนมาจากวัตถุเท่านั้น แต่จะขึ้นอยู่กับระดับความสว่างของสิ่งแวดล้อมรอบๆ ด้วย ดังตัวอย่างจากการทดลองของ Ernest Mach [7] ดังรูปที่ 3.3



รูปที่ 3.3 แสดง ตัวอย่างการทดลองของ Ernest Mach

เส้นกราฟด้านล่างจะแสดงถึงระดับความสว่างของภาพที่อยู่ด้านบน แต่จะพบว่าในการรับรู้ความสว่างของตามนุษย์นั้น เมื่อสังเกตภาพข้างต้นจะเห็นเป็นแถบมีความมืดลงที่ตำแหน่ง ด้านขวาของแถบ และสว่างขึ้นทางด้านซ้าย จากการทดลองนี้จะแสดงให้เห็นว่า แถบที่มีระดับความสว่างคงที่ จะมีการเปลี่ยนแปลง ไปในการรับรู้ของตามนุษย์

ปรากฏการณ์ลักษณะนี้เกิดขึ้นจากความจริงที่ว่า ตาของมนุษย์นั้น ตอบสนองต่อระดับความสว่างในส่วนต่างๆ ของภาพได้ไม่เท่ากัน ระดับความสว่างของภาพบางส่วนไม่ได้มีความหมายใดๆ ต่อการรับรู้ข้อมูลภาพนั้นๆ ซึ่งลายละเอียดของภาพในส่วนที่ไม่มีความสำคัญนี้ จะเรียกว่าเป็น Psychovisual Redundant ที่จะสามารถตัดออกไปจากข้อมูลภาพได้ โดยไม่ทำให้ความหมายของภาพในการรับรู้เปลี่ยนไป เนื่องจากในการรับรู้ความหมายของภาพ โดยมนุษย์นั้นไม่ได้เป็นการพิจารณาถึงระดับความสว่างของแต่ละจุดภาพ โดยทั่วไปตาของมนุษย์จะสังเกตถึงคุณลักษณะที่ จะแบ่งแยกวัตถุ หรือสิ่งต่างๆ ออกจากกัน เช่นขอบของรูป หรือบริเวณของพื้นผิว เป็นต้น รวมเข้าด้วยกันเป็นวัตถุ หรือรูปร่างต่างๆ และสมองจะทำการแยกแยะวัตถุ หรือรูปร่างนั้นๆ ออกมาโดยเปรียบเทียบเข้ากับสิ่งที่เคยรู้จักมาแล้ว

Psychovisual Redundancy จะมีหลักการพื้นฐาน ที่ไม่เหมือนกับ การลดความซ้ำซ้อนกันทั้งสองแบบที่ได้อธิบายไปแล้ว Psychovisual Redundancy นี้จะเกี่ยวข้องโดยตรงกับการมองเห็นของตามนุษย์ ข้อมูลส่วนที่ซ้ำซ้อนกันนั้นก็เนื่องมาจาก มันไม่ได้มีความหมายในการรับรู้นั้นเอง ซึ่งการตัดข้อมูลในส่วนนี้ทิ้งไปนั้น จะทำให้ ข้อมูลของภาพในเชิงปริมาณเสียไป

ขบวนการนี้เช่นขบวนการของ Quantization ซึ่งจะหมายเป็นขบวนการ Mapping ข้อมูลอินพุทที่มีจำนวนหลายๆ แบบ ไปเป็นข้อมูลที่มีจำนวนแบบที่ลดลง และขบวนการ Quantization นี้จะเป็นขบวนการที่ไม่สามารถกระทำย้อนกลับได้ (Irreversible) เนื่องจากข้อมูลส่วนที่ถูกตัดออกไปแล้ว นั้นจะไม่สามารถนำกลับมาได้

รูปที่ 3.4 แสดงตัวอย่างผลของขบวนการ Quantization

จากรูปที่ 3.4 (a) เป็นภาพระดับสีเทา 256 ระดับ (รหัส 8 Bit) และรูปที่ 3.4 (b) เป็นภาพระดับสีเทาที่มีจำนวนระดับที่ลดลง ด้วยการทำ Uniform Quantization ให้เหลือเพียง 8 ระดับ (รหัส 3 Bit) ซึ่งจะได้ค่าอัตราการลดขนาดข้อมูล เป็น 2.66 เท่า และจากการสังเกตของตามนุษย์นั้นจะพบว่า ในรูปที่ 3.4 (b) นั้นความต่อเนื่องกันของระดับสีเทาจะเห็นแยกออกมาเป็นขั้นๆ กันอย่างชัดเจน ซึ่ง เป็นผลมาจากการที่ระดับความสว่างแต่ละระดับห่างกันมากเกินไป

ในรูปที่ 3.4 (c) จะแสดงถึงภาพที่ผ่านขบวนการปรับปรุงแล้ว โดยอาศัยลักษณะที่พิเศษของตามนุษย์ จากรูปที่ 3.4 (b) ซึ่งไม่มีความต่อเนื่องกันของระดับ สีเทานั้นสามารถจะลดลง ได้โดยการเพิ่มรายละเอียดเล็กๆ ที่ไม่สามารถสังเกตเห็น ซึ่งวิธีการที่ใช้นี้จะมีชื่อเรียกว่า Improved Gray-Scale (IGS) Quantization จะมีหลักการคือ จากการที่ตาของมนุษย์นั้นมีความไวต่อขอบเขตของวัตถุในภาพมาก ดังนั้นวิธีการของ IGS จะทำการลดความแตกต่างของระดับความสว่างลง โดยการนำค่าของตัวเลขสุ่ม มารวมกับข้อมูลภาพก่อนการ Quantizing โดยที่ ค่าของตัวเลขสุ่มนี้จะได้มาจาก บิตทางด้านต่ำของข้อมูลจุดภาพที่อยู่รอบๆ นั้นเอง เพราะว่าบิตทางต่ำของข้อมูลนี้จะมีลักษณะของการเกิดขึ้นแบบสุ่มอยู่แล้วนั้นเอง [7]

3.2 การวัดความเหมือนกันของภาพ

จากหัวข้อที่ผ่านมา ในการที่จะลดความซ้ำซ้อนกันของข้อมูลภาพ จะสามารถทำได้สองลักษณะใหญ่ๆ คือ การลดการซ้ำซ้อนกันที่ ตัวของข้อมูลภาพเอง และการลดการซ้ำซ้อนกัน โดยความหมายของภาพที่สามารถรับรู้ได้ ในกรณีหลังนี้ข้อมูลในเชิงปริมาณ บางส่วนของภาพจะสูญเสียไป ซึ่ง อาจมีความจำเป็นในการนำไปประมวลผลต่อไป

ในการวัดคุณภาพหรือความเหมือนกันของภาพ จะสามารถกระทำได้สองวิธีการใหญ่ๆ คือ

1. การวัดคุณภาพหรือความเหมือนกันในเชิงปริมาณ
2. การวัดคุณภาพหรือความเหมือนกัน โดยความหมายของภาพ

ค่าของ ปริมาณของข้อมูลที่สูญเสียหรือผิดพลาดไป จะสามารถอธิบายได้ในรูปของ Function ของภาพต้นแบบ กับ ภาพที่ได้ หลังจากผ่านขบวนการลดขนาดแล้ว ซึ่งวิธีการวัดแบบนี้จะเป็นการวัดคุณภาพหรือความเหมือนกันในเชิงปริมาณ ตัวอย่างเช่น การวัดโดยใช้ค่าของ Root-mean Square (RMS) Error ระหว่างภาพต้นแบบ กับ ภาพผลลัพท์ที่ได้ ถ้าให้ $f(x,y)$ เป็นข้อมูลของภาพต้นแบบ และ $\hat{f}(x,y)$ เป็นข้อมูลของภาพผลลัพท์ที่ได้ หลังจากผ่านขบวนการลดขนาดแล้วที่ทุกๆ ตำแหน่งของจุดภาพ (x,y) ใดๆ ค่าของความผิดพลาดที่เกิดขึ้น $e(x,y)$ จะสามารถคำนวณได้จากสมการที่ 3.6

$$e(x,y) = \hat{f}(x,y) - f(x,y) \quad \text{.....(3.6)}$$

ดังนั้นค่าความผิดพลาดทั้งหมดระหว่างภาพทั้งสองจะเป็น

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right] \quad \text{.....(3.7)}$$

เมื่อ M คือค่าของจำนวนเส้นภาพ และ

N คือค่าของจำนวนจุดภาพตามแนวนอน

ค่าของ Root-mean Square Error (e_{rms}) จะหมายถึง รากที่สองของค่าความผิดพลาดกำลังสองเฉลี่ยของทั้งรูป หรือเขียนเป็นสมการได้คือ

$$e_{rms} = \left[\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2\right]^{\frac{1}{2}} \quad \text{.....(3.8)}$$

หรืออีกวิธีการหนึ่งที่ใช้ในการวัดคุณภาพหรือความเหมือนกันในเชิงปริมาณ จะทำโดยการหาค่าของ Mean-Square Signal-to-Noise Ratio (SNR_{ms}) ของข้อมูลภาพที่ถูกลดขนาดและแปลงผกผันออกมาแล้ว คือ

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} \hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2} \quad \text{.....(3.9)}$$

และ ถ้าวัดในหน่วยของค่า RMS จะได้เป็น SNR_{rms} ซึ่ง หาได้จากการถอดรากที่สองของสมการที่ 3.9

ถึงแม้ว่าการใช้วิธีการวัดคุณภาพหรือความเหมือนกันในเชิงปริมาณ จะเป็นวิธีที่ง่ายและ มีความสะดวก ในการที่จะวัดจำนวนของข้อมูลที่สูญเสียไป แต่ท้ายที่สุดแล้ว การใช้งานของข้อมูลภาพนี้ก็ต้องถูกนำมาใช้กับมนุษย์ในการรับรู้ ดังนั้นการวัดคุณภาพหรือความเหมือนกัน ของภาพโดยอาศัยการประเมินจากผู้สังเกต จึงจะให้ผลที่ถูกต้องกว่า ถึงคุณภาพของภาพที่ได้

ในการประเมินนี้ ผู้สังเกตจะให้เป็นระดับคะแนน ดังตัวอย่าง ตารางที่ 3.2

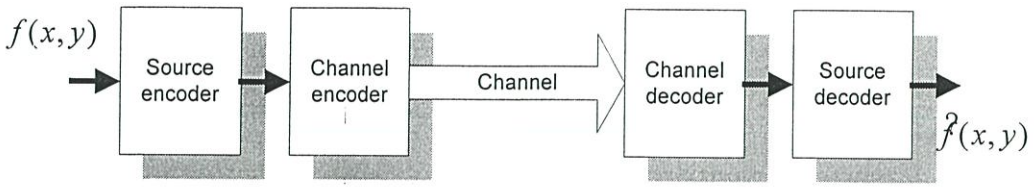
ตารางที่ 3.2 การประเมินคุณภาพของภาพ

คะแนน	ความหมาย	อธิบาย
1	ดีมาก	ภาพมีความคมชัดดีมาก, ไม่มีที่ติ
2	ดี	ภาพชัดเจนดี, ไม่มีการรบกวน
3	ผ่าน	ภาพสามารถยอมรับได้, ไม่มีการรบกวน
4	แย่	ภาพไม่ชัดเจนนัก, ดูแล้วน่าจะดีกว่านี้สักเล็กน้อย, สามารถสังเกตพบสิ่งรบกวนได้
5	แย่มาก	ภาพแย่มาก, เห็นสิ่งรบกวนชัดเจน
6	ใช้ไม่ได้	ภาพไม่สามารถดูได้

หรือจะเป็นการประเมินในลักษณะของการเปรียบเทียบ กันระหว่างภาพต้นแบบ กับภาพที่ได้หลังจากผ่านขบวนการลดขนาดข้อมูลแล้ว ก็จะสามารถกระทำได้โดยให้คะแนนเป็น { -3, -2, -1, 0, 1, 2, 3 } แทนความหมายคือ { ด้อยกว่ามาก, ด้อยกว่า, ด้อยกว่าเล็กน้อย, เหมือนกัน, ดีกว่าเล็กน้อย, ดีกว่า ดีกว่ามาก } ตามลำดับ

3.3 แบบอย่างของขบวนการลดขนาดข้อมูลภาพ

จากหัวข้อที่ผ่านมา ได้กล่าวถึงวิธีการในการลดความซ้ำซ้อนกันที่มีในภาพ ซึ่งมีหลักการอยู่สามแบบด้วยกัน ในทางปฏิบัติแล้ว การลดขนาดของข้อมูลภาพ จะเป็นการนำ เอาหลักการการทั้งสามนี้มาประยุกต์ใช้งานร่วมกัน เป็นขบวนการลดขนาดข้อมูลภาพที่ต้องการ



รูปที่ 3.5 แสดงขบวนการทั้งหมดในการลดข้อมูลภาพ

จากรูปที่ 3.5 จะเป็นขบวนการ ทั้งหมดของการลดขนาดข้อมูลภาพ โดยจะแยกออกเป็นสองส่วนคือขั้นตอนของขบวนการเข้ารหัส (Encoder) และ ขั้นตอนของการถอดรหัส (Decoder) ข้อมูลภาพต้นแบบ (Input Image : $f(x,y)$) จะผ่านขบวนการของ Encoder ซึ่งจะได้รหัสแทนของข้อมูล นี้ ที่มีจำนวนลดลง หลังจากรหัสแทนข้อมูลนี้ผ่านตัวกลาง (Channel) มายังภาค Decoder ที่จะทำการสร้าง ภาพผลลัพท์ (Output Image : $\hat{f}(x,y)$) กลับออกมาเพื่อนำไปใช้งานต่อไป

โดยทั่วไปแล้ว $\hat{f}(x,y)$ ที่ได้กลับออกมานี้ อาจเหมือนหรือไม่เหมือนกับ $f(x,y)$ ก็ได้ ขึ้นอยู่กับขบวนการที่ใช้ว่าเป็นแบบที่มีความผิดพลาดหรือไม่ ถ้าเป็นขบวนการแบบที่มีความผิดพลาดแล้วนั้น $\hat{f}(x,y)$ ที่ได้ก็จะมีความผิดพลาดเกิดขึ้น

จากรูปที่ 3.5 ในส่วนของ Encoder และ Decoder จะประกอบขึ้นมาจากส่วนย่อยๆ สองส่วนที่มีหน้าที่ๆ แตกต่างกันคือ สำหรับในสวนของ Encoder จะประกอบไปด้วย Source Encoder ซึ่งทำหน้าที่ในการลดการซ้ำซ้อนกันของข้อมูลภาพ และ Channel Encoder ซึ่งทำหน้าที่ในการเพิ่มความต้านทานของข้อมูล ต่อสัญญาณรบกวนที่จะเกิดขึ้นในการส่งข้อมูลผ่านตัวกลาง และเช่นเดียวกันสำหรับในส่วนของ Decoder ซึ่งประกอบไปด้วย Channel Decoder และตามด้วย Source Decoder ตามลำดับ

ในกรณีที่ตัวกลางในการส่งข้อมูล เป็นแบบที่ไม่มีความผิดพลาด หรือสัญญาณรบกวนเกิดขึ้นแล้ว เช่นในการเก็บข้อมูลในรูปแบบของ ดิจิตอล ในอุปกรณ์บันทึกข้อมูล ส่วนของ Channel Encoder และ Channel Decoder จะไม่มีความจำเป็น จึงจะเหลือเฉพาะแต่ Source Encoder และ Source Decoder ตามลำดับ

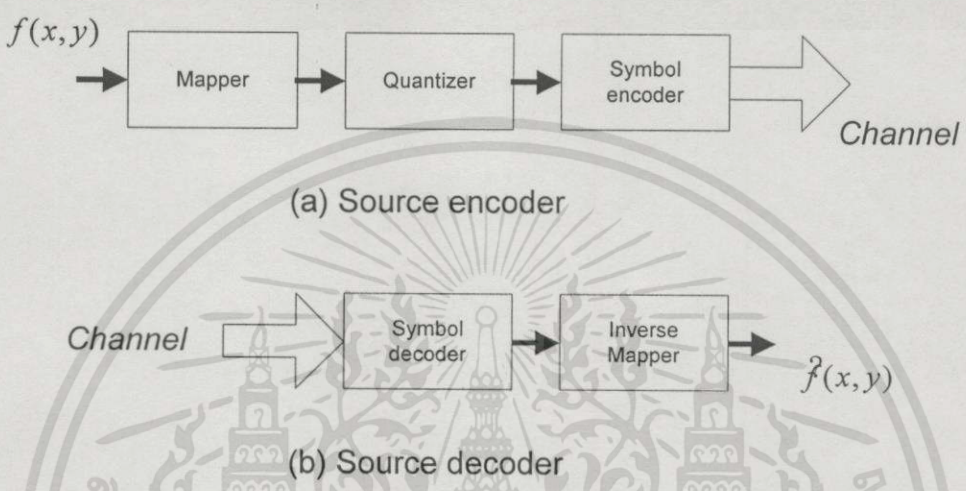
3.3.1 Source Encoder และ Decoder

ในส่วนของ Source Encoder นี้จะมีหน้าที่สำหรับ การลดความซ้ำซ้อนกันของข้อมูลภาพทั้งในส่วนของ Coding Redundancy Interpixel Redundancy และ Psychovisual Redundancy

ในการออกแบบขบวนการของ Source Encoder นั้นลักษณะของการนำไปประยุกต์ใช้งานและคุณภาพของภาพ ที่ต้องการจะเป็นตัวกำหนดถึงวิธีการที่เหมาะสมจะนำมาใช้ เช่น ข้อมูลภาพที่ต้องการนำไปประมวลผลต่อไป ควรต้องใช้ขบวนการลดขนาดข้อมูลแบบที่ไม่มีความผิด

พลาด และสำหรับในการใช้งานทั่วไปแล้วนั้น ภาพอาจมีความผิดพลาดได้บ้าง ตามแต่คุณภาพของภาพที่ต้องการ เป็นต้น

โดยส่วนใหญ่แล้วสำหรับการใช้งานทั่วไป ส่วนของ Source Encoder จะประกอบไปด้วยสามส่วนใหญ่ ดังแสดงในรูปที่ 3.6 (a) และสำหรับส่วนของ Source Decoder จะแสดงในรูปที่ 3.6 (b)



รูปที่ 3.6 (a) แสดงขบวนการของ Source Encoder และ (b) ขบวนการของ Source Decoder

จากองค์ประกอบของ Source Encoder นี้ ในส่วนแรก ซึ่งเรียกว่า Mapper จะทำหน้าที่ในการ แปลงข้อมูลภาพที่เข้ามาให้อยู่ในรูปแบบที่เหมาะสมในการลด Interpixel Redundancy

โดยที่การทำ Mapping นี้ อาจเป็นการลด Interpixel Redundancy โดยตรง หรือไม่ก็จะเป็นการทำให้ ส่วนที่เป็น Interpixel Redundancy นี้สามารถที่จะเข้าถึงหรือ ลดลงได้ โดยขบวนการขั้นต่อๆ ไป โดยทั่วไปแล้วขบวนการส่งถ่ายนี้จะเป็นขบวนการที่สามารถกระทำย้อนกลับได้ (Reversible)

ข้อมูลที่ได้หลังจากที่ผ่านขบวนการส่งถ่ายแล้ว อาจมีขนาดที่ลดลงหรือไม่ก็ได้ สำหรับตัวอย่างของขบวนการส่งถ่ายเช่น ขบวนการของ Run-length Coding ซึ่งเป็นวิธีการส่งถ่ายวิธีหนึ่ง ที่เป็นการลดจำนวนของข้อมูลลงโดยตรง หรือขบวนการของการแปลงข้อมูลด้วย DCT (Discrete Cosine Transform) และ DWT (Discrete Wavelet Transform) ซึ่งข้อมูลที่ผ่านการแปลงแล้ว จะมีจำนวนที่เท่าเดิม แต่รูปแบบการเรียงของข้อมูลจะเปลี่ยนไปให้ง่ายต่อการจัดการลด Interpixel Redundancy ตามที่ต้องการ

สำหรับในส่วนที่สองของ Block Diagram ในรูปที่ 3.6 (a) หรือ Quantizer ซึ่ง จะเป็นการลดความละเอียดของข้อมูลที่ได้จากการส่งถ่ายแล้ว การลดความละเอียดของข้อมูลนี้จะเป็นผลให้ข้อมูลมีความผิดพลาดเกิดขึ้น ขบวนการ Quantization จะเป็นการลด Psychovisual

Redundancy ดังในหัวข้อที่ผ่านมาแล้ว ซึ่งเป็นขบวนการที่ไม่สามารถกระทำย้อยกลับได้ (Irreversible) ดังนั้น ถ้าต้องการลดขนาดข้อมูลภาพแบบที่ไม่มีความผิดพลาด ส่วนของการทำ Quantization นี้จะไม่มี

และสำหรับส่วนสุดท้าย หรือ Symbol Coder ในส่วนนี้จะเป็นการแทนข้อมูลที่ได้จากการ Quantize ด้วยรหัสที่เหมาะสม ซึ่งขบวนการนี้จะเป็นการลด Coding Redundancy นั้นเอง โดยทั่วไปแล้วการใช้ Variable-length Coding จะเป็นวิธีการที่มีประสิทธิภาพ เนื่องจากมันจะใช้รหัสที่สั้นที่สุด แทนข้อมูลที่พบเป็นจำนวนมาก ซึ่งจะทำให้จำนวนของรหัสที่ต้องใช้แทนข้อมูลทั้งหมดนี้ลดลง

หลังจากที่ข้อมูลภาพ ผ่านขบวนการทั้งสามนี้แล้ว ความซ้ำซ้อนกัน ที่มีในข้อมูลภาพจะลดลง และจำนวนของรหัสข้อมูลทั้งหมดจะลดลงด้วย

เมื่อต้องการนำข้อมูลภาพนี้มาใช้งาน จะกระทำได้โดยใช้ขบวนการของ Source Decoder ในการสร้างข้อมูลภาพกลับออกมา ซึ่งจะมีขบวนการดังแสดงในรูปที่ 3.6 (b) โดยจะประกอบด้วยส่วนย่อยๆ เพียงสองส่วนคือ

ในส่วนแรก หรือ Symbol Decoder จะทำหน้าที่ในการแปลงรหัส กลับออกมาเป็นข้อมูล จากนั้นข้อมูลนี้จะผ่านไปยังส่วนที่สอง หรือ Inverse Mapper ซึ่งจะทำการแปลงข้อมูลนี้ กลับออกมาเป็นข้อมูลของภาพที่ต้องการ จะเห็นได้ว่าในขบวนการของ Source Decoder นี้จะเป็นขบวนการที่กระทำย้อนกลับของ Source Encoder แต่จะไม่มีส่วนของการทำ Inverse Quantization เพราะว่าขบวนการ Quantization นี้เป็นขบวนการที่ไม่สามารถกระทำย้อนกลับได้

3.4 การลดขนาดข้อมูลแบบไม่มีความผิดพลาด

ในการประยุกต์ใช้ ขบวนการลดขนาดข้อมูลภาพ กับข้อมูลบางชนิด จะมีความจำเป็นที่จะต้องใช้ ขบวนการลดขนาดข้อมูลภาพ แบบที่ไม่มีความผิดพลาด (Loss-less Compression) เช่น การเก็บเอกสารทางการแพทย์หรือ เอกสารหลักฐานทางธุรกิจ ซึ่งเป็นข้อกำหนดทางกฎหมายที่ต้องใช้ขบวนการลดขนาดข้อมูลแบบที่ไม่มีความผิดพลาด หรืออีกตัวอย่างหนึ่งเช่น ในการประมวลผลข้อมูลภาพถ่ายดาวเทียมที่ได้จาก LANDSAT ซึ่งเป็นข้อมูลที่ได้มาด้วยค่าใช้จ่ายที่มาก จึงไม่ควรให้การเก็บข้อมูล นี้มีความผิดพลาดเกิดขึ้น หรือในการนำไปใช้ในการเก็บข้อมูลของภาพถ่าย X-Ray ซึ่งความผิดพลาดที่เกิดขึ้น จะส่งผลให้ความแม่นยำในการวินิจฉัยโรค ลดลง จากตัวอย่างข้างต้นเหล่านี้ จำเป็นต้องใช้ขบวนการลดขนาดข้อมูล แบบที่ไม่มีความผิดพลาด

สำหรับขบวนการพื้นฐานของ Loss-less Compression ที่ใช้กันอยู่ในปัจจุบัน จะให้ค่าอัตราการลดขนาดข้อมูล ประมาณ 2 ถึง 10 เท่า ซึ่งสามารถนำไปใช้กับข้อมูลภาพ ขาวดำ หรือข้อมูลภาพ ระดับความสว่าง (Binary Image) ได้

จากในหัวข้อที่ผ่านมา ขบวนการของ Loss-less Compression จะประกอบขึ้นจากขั้นตอนสองขั้นตอนคือ (1) ขบวนการแปลงข้อมูลภาพใหม่ เพื่อเป็นการลด Inter pixel Redundancy และ (2) ขบวนการเข้ารหัสข้อมูล เพื่อที่จะลด Coding Redundancy หรือจากรูปของขบวนการ Source Coding รูปที่ 3.6 (a) ขบวนการทั้งสองข้างต้นก็จะคือ ส่วนของ Mapping และ Symbol Coding ตามลำดับ นั้นเอง

3.4.1 การเข้ารหัสแบบ ฮัฟฟ์แมน (Huffman Coding)

วิธีการที่ง่ายที่สุดในการทำ Loss-less Compression คือ จะเป็นการการลด เพียงแต่ส่วนของ Coding Redundancy เท่านั้น ซึ่งเป็นความซ้ำซ้อนกันที่เกิดขึ้น เมื่อมีการใช้รหัสเลขฐานสอง ในการแทนความหมาย ระดับความสว่างของจุดภาพต่างๆ

จากในหัวข้อที่ผ่านมาของ Coding Redundancy การเข้ารหัสข้อมูลของระดับความสว่าง ที่จะทำให้ สมการที่ 3.3 มีค่าน้อยที่สุด นั้น จะสามารถทำได้หลายวิธีด้วยกัน ซึ่งวิธีการหนึ่งที่มีประสิทธิภาพนี้คือ Huffman Coding [5] เป็นวิธีที่นิยมใช้ในการลดข้อมูลในส่วนของ Coding Redundancy เนื่องจากมันเป็นวิธีการที่จะให้รหัสที่เป็นตัวแทนของข้อมูลที่ต้องการได้สั้นที่สุด สำหรับวิธีการของ Huffman Coding จะสามารถทำได้โดยมีขั้นตอนดังนี้

ในขั้นแรกจะเป็นขบวนการที่เรียกว่า Source Reduction โดยการนำค่าของ ความน่าจะเป็นในการที่จะพบ ค่าระดับความสว่างต่างๆ (P_r) ในข้อมูลภาพ นำค่าของความน่าจะเป็นนี้ มาจัดเรียงกันตามลำดับจากมากไปหาน้อย จากนั้นก็จะทำการรวมค่าความน่าจะเป็นของระดับความสว่าง ที่น้อยที่สุดสองลำดับเข้าด้วยกัน เป็นค่าเดียว จากนั้นก็จะทำการเรียง ค่าของความน่าจะเป็นนี้ใหม่ และกระทำซ้ำเช่นนี้ต่อไปจนหมด

ข้อมูลเริ่มต้น		การจัดกลุ่มข้อมูล			
ความสว่าง	ความน่าจะเป็น	1	2	3	4
a2	0.4	0.4	0.4	0.4	0.6
a6	0.3	0.3	0.3	0.3	0.4
a1	0.1	0.1	0.2	0.3	
a4	0.1	0.1	0.1		
a3	0.06	0.1			
a5	0.04				

รูปที่ 3.7 แสดงขั้นตอนการจัดกลุ่มข้อมูล

ดังรูปที่ 3.7 แสดงถึงวิธีการของ Source Reduction ทางด้านซ้ายมือจะเป็นค่าของระดับความสว่างต่างๆ เรียงลงมาตามค่าของความน่าจะเป็น จากมากไปหาน้อย ในการทำ Source Reduction ครั้งที่ 1 ค่าที่น้อยที่สุดของความน่าจะเป็นสองค่า คือ 0.06 และ 0.04 จะถูกรวมเข้าด้วยกันเป็น 0.1 เพื่อแสดงถึงความน่าจะเป็นที่จะพบระดับความสว่างทั้งสองนี้ แล้วนำมา จัดเรียงใหม่ในช่องของ Source Reduction ครั้งที่ 1 จากนั้นจะกระทำเช่นนี้ต่อไปเรื่อยๆ จนกระทั่ง เหลือรหัสเพียงสองกลุ่มเท่านั้น (ทางด้านขวาสุด)

ขั้นที่สองของการทำ Huffman Coding คือการกำหนด รหัสที่จะใช้แทนข้อมูลระดับความสว่างนี้ใหม่ โดยเริ่มจากข้อมูลที่ ถูกจัดกลุ่มแล้ว (ทางด้านขวา) ย้อนกลับไปยังข้อมูลเริ่มต้น ดังแสดงในรูปที่ 3.8

ข้อมูลเริ่มต้น			การจัดกลุ่มข้อมูล			
ข้อมูล	ความน่าจะเป็น	รหัส	1	2	3	4
a2	0.4	1	0.4 1	0.4 1	0.4 1	0.6 0
a6	0.3	00	0.3 00	0.3 00	0.3 00	0.4 1
a1	0.1	011	0.1 011	0.2 010	0.3 01	
a4	0.1	0100	0.1 0100	0.1 011		
a3	0.06	01010	0.1 0101			
a5	0.04	01011				

รูปที่ 3.8 แสดงขั้นตอนการกำหนดรหัสแทนข้อมูล

โดยเริ่มแรก จากระดับความสว่างที่ถูกจัดให้เหลือเพียงสองกลุ่ม จะถูกแทนด้วยรหัสเลขฐานสองคือ "0" และ "1"

ซึ่ง จะให้รหัส "0" แทนกลุ่มของข้อมูลที่ค่าความน่าจะเป็น 0.6 และ "1" แทนกลุ่มของข้อมูลที่ค่าความน่าจะเป็น 0.4 (สำหรับการแทนรหัส ของกลุ่มของข้อมูลระดับความสว่างนี้ จะสามารถ

สลับกันได้ ระหว่าง "0" และ "1") จากกลุ่มของระดับความสว่างที่มีค่าความน่าจะเป็น เท่ากับ 0.6 ซึ่งได้มาจากการรวมกันของกลุ่มข้อมูลระดับความสว่าง ดังนั้นเพื่อที่จะแยกความแตกต่างของ ระดับความสว่างทั้งสองนี้ จึงจะใช้เลขฐานสอง "0" และ "1" โดยการต่อเพิ่มเข้าไปกับรหัส ที่ได้กำหนดให้ก่อนหน้านี้ และกระทำต่อไปจนกระทั่ง ย้อนกลับไปถึงระดับความสว่างเริ่มต้น ทางด้านซ้ายมือ

จากรหัสที่หาได้ตามขบวนการของ Huffman Coding ตามตัวอย่างข้างต้นนี้ จะได้ค่าของความยาวรหัสเฉลี่ย (L_{avg}) คือ

$$L_{avg} = 1(0.4) + 2(0.3) + 3(0.1) + 4(0.1) + 5(0.06) + 5(0.04)$$
$$= 2.2 \text{ Bit}$$

วิธีการของ Huffman Coding นี้ จะเป็นวิธีการที่มีประสิทธิภาพมากในการสร้างรหัสแทนข้อมูลที่ต้องการ หลังจากที่หารหัสของ Huffman ได้แล้ว ขบวนการเข้ารหัสข้อมูล หรือถอดรหัสข้อมูลสามารถทำได้ โดยการเปิด ตารางเทียบข้อมูลที่ต้องการกับรหัสที่ได้

ข้อมูลที่เข้ารหัสแบบ Huffman นี้จะมีลักษณะที่พิเศษคือ ในการถอดรหัสข้อมูลจะสามารถทำได้โดยที่ไม่ต้องมีข้อมูลอ้างอิง ขั้นระหว่างรหัสแต่ละตัว รหัสของ Huffman นี้ จะสามารถแยกออกมาได้โดยไม่ซ้ำกัน และจะแยกรหัสนี้ได้เพียงวิธีเดียวเท่านั้น จากการถอดรหัสจากซ้ายไปขวา

<u>ตัวอย่าง</u> จากรหัส Huffman ที่ได้ในรูปที่ 3.8 ถ้าข้อมูลที่ผ่านการเข้ารหัสแล้วมีดังนี้ "010100111100" จะพบว่า ถ้าอ่านจากซ้ายมาขวาแล้ว รหัสตัวแรกที่สามารถอ่านได้ (มี) คือ "0101" ซึ่งแทนข้อมูลของระดับความสว่างที่ a_3 และรหัสตัวต่อไปที่มีคือ "011" ซึ่งเป็นรหัสของข้อมูล a_1 ทำต่อไปจนหมดจะได้ว่าข้อมูลของรหัสนี้คือ $a_3\, a_1\, a_2\, a_2\, a_6$ ซึ่งจากตัวอย่างนี้ จะเห็นได้ว่าในการถอดรหัสของ Huffman จะสามารถทำได้แบบเดียวเท่านั้น

วิธีการเข้ารหัสแบบ Huffman นี้ เป็นวิธีการที่มีประสิทธิภาพมากที่สุด จึงนิยมใช้กันอย่างทั่วไป

3.4.2 การเข้ารหัสแบบ รัน-เรนจ์ (Run-length Coding)

ในการนำขบวนการเข้ารหัสแบบ Run-length Coding มาใช้กับข้อมูลภาพนี้ จะเป็นการลดความซ้ำซ้อนกันของข้อมูล (Interpixel Redundancy) สำหรับการนำมาใช้งานกับข้อมูลที่เป็นหนึ่งมิติ (1-D) จะสามารถอธิบายได้คือ ในกรณีที่ระดับความสว่างของภาพมีค่าคงที่ ติดต่อกัน ค่ารหัสที่ใช้แทนข้อมูลระดับความสว่างนี้ ก็จะมีค่าซ้ำๆ กันไป ในการเก็บรหัสเหล่านี้ลงไปทั้งหมดก็จะใช้เนื้อที่มาก วิธีการของ Run-length Coding จะเก็บข้อมูลนี้ในลักษณะของ รหัสระดับความสว่างกับจำนวนที่อยู่ติดต่อกัน เป็นคู่ลำดับของ (รหัสข้อมูล , จำนวน)

อย่างไรก็ตาม รหัสที่ได้หลังจากผ่านขบวนการทำ Run-length Coding แล้ว อาจมีขนาดหรือจำนวนที่มากขึ้นกว่าเดิมก็ได้ ถ้าภาพนั้น มีรายละเอียดมาก ซึ่งข้อมูลภาพเหล่านี้ค่าระดับความสว่างจะไม่คงที่ หรือคงที่ติดต่อกันไม่มากนัก

ดังนั้นในการใช้ Run-length Coding โดยทั่วไปจะใช้กับข้อมูลระดับความสว่างของภาพ ซึ่งได้แปลงให้อยู่ในรูปแบบที่เหมาะสมแล้ว จึงจะให้ค่าอัตราส่วนการลดขนาดข้อมูลที่น่าพอใจ

3.5 การลดขนาดข้อมูลแบบที่มีความผิดพลาด

วิธีการของ Lossy Compression นี้จะไม่เหมือนกับ ขบวนการดังที่ผ่านมาแล้ว โดยมันจะยอมลดความถูกต้องของภาพผลลัพท์ที่ได้ เพื่อแลกกับอัตราการลดขนาดข้อมูลที่เพิ่มมากขึ้น

ทั้งนี้ก็ขึ้นอยู่กับว่า ในการใช้งานของ ข้อมูลภาพ จะสามารถยอมรับความผิดพลาดได้มากเพียงใด โดยทั่วไปแล้ว การใช้ขบวนการของ Lossy Compression จะสามารถใช้ลดขนาดข้อมูลภาพ ระดับสีเทา (Gray level image) ได้ถึง 30 เท่า โดยที่ภาพที่ได้ยังคงใช้สื่อความหมายได้ และที่ค่าอัตราส่วนการลดขนาดข้อมูลที่ 10-20 เท่า ภาพที่ได้จะเหมือนกับภาพต้นแบบ เมื่อเปรียบเทียบกับการลดขนาดข้อมูลด้วยขบวนการของ Lose-less Compression ซึ่งจะสามารถลดขนาดข้อมูลได้เพียง 2-3 เท่า

จากในหัวข้อที่ผ่านมา สิ่งที่แตกต่างกันระหว่างวิธีการทั้งสองนี้คือ ส่วนของขบวนการ Quantization นั้นเอง

3.5.1 Transform Coding

จากขบวนการลดขนาดข้อมูลภาพที่ได้กล่าวมาแล้ว จะเป็นการกระทำโดยตรงกับข้อมูลภาพ ซึ่ง เรียกว่าเป็นวิธีการใน Spatial domain

สำหรับในหัวข้อนี้จะได้กล่าวถึงขบวนการลดขนาดข้อมูลภาพที่ถูกแปลงให้อยู่ในรูปแบบอื่นแล้ว สำหรับขบวนการของ การแปลง ที่ใช้นี้จะเป็นแบบที่สามารถกระทำย้อนกลับได้ เช่น Fourier transform Discrete cosine transform หรือ Discrete wavelet transform ก็ได้

ข้อมูลภาพเมื่อผ่านขบวนการแปลงแล้วจะได้ออกมาเป็นค่าสัมประสิทธิ์ (Transform coefficient) เพื่อนำไปผ่านขั้นตอนของ Quantization และ การเข้ารหัสข้อมูลต่อไป

โดยส่วนใหญ่แล้ว ค่าสัมประสิทธิ์ที่ได้จากขบวนการแปลงข้อมูลภาพที่มีค่าน้อยจะสามารถปัดรวมเข้าด้วยกันหรือ ตัดทิ้งออกไปได้ โดยไม่ทำให้ภาพผลลัพท์ที่ได้จากการแปลงผกผัน (Inverse transform) ผิดเพี้ยนไปมากนัก

ในขั้นตอนของ Forward transform จะมีหน้าที่ในการหาความสัมพันธ์ ระหว่างจุดภาพต่างๆ รวมเข้าด้วยกัน โดยใช้จำนวนของ ค่าสัมประสิทธิ์ ให้น้อยที่สุด จากนั้นขบวนการของ Quantization จะทำการแยกส่วนของค่าสัมประสิทธิ์ ที่ ไม่มีความสำคัญออกไป หรือเป็นการลดความละเอียดของค่าสัมประสิทธิ์ นี้ และส่วนสุดท้ายคือการเข้ารหัส ซึ่งจะเป็นการเลือกรหัสที่เหมาะสม

สำหรับข้อมูลของค่าสัมประสิทธิ์ ที่ได้จากขบวนการ Quantization โดยมากแล้ว การใช้วิธีการเข้ารหัสแบบ Variable-length coding จะเป็นวิธีการที่มีประสิทธิภาพมากที่สุด

3.5.2 การเลือก Transform ที่เหมาะสม

ขบวนการแปลงข้อมูลจากโดเมน หนึ่งๆ ไปยังโดเมนใดๆ นั้น จะมีอยู่หลายวิธีที่นิยมใช้กันคือ Discrete Fourier Transform (DFT) Discrete Cosine Transform (DCT) Discrete Wavelet Transform (DWT) และอื่นๆ ในการที่จะเลือกใช้ การแปลง ที่เหมาะสมสำหรับใช้ในขบวนการลดขนาดข้อมูลภาพนี้ จะขึ้นอยู่กับความสามารถในการลดขนาดของข้อมูลโดยขบวนการแปลงนั้นๆ (Information packing) และความสามารถของการประมวลผลที่มี ซึ่งโดยมากแล้วขบวนการ Transform ที่มีความสามารถสูง จะต้องใช้การคำนวณมาก ในขั้นตอนของ Image Transform นี้ ขนาด (ปริมาณ) ของข้อมูลจะไม่ลดลง ข้อมูลของค่าสัมประสิทธิ์ ที่เป็นตัวแทนของภาพ จะถูกลดจำนวน (ปริมาณ) ลง โดยขบวนการของ Quantization และ Coding ต่อไป

สำหรับความสามารถในการหาความสัมพันธ์ (Information packing) ของขบวนการ แปลงแบบต่างๆ นี้สามารถกระทำได้โดยการทดลองดังต่อไปนี้

ทำการแปลงข้อมูลภาพที่ใช้เป็นต้นแบบ ด้วย การแปลง ต่างๆ เช่น DFT DCT DWT สำหรับในและ การแปลง ก็ จะได้ค่าของสัมประสิทธิ์ ออกมา จากค่าของสัมประสิทธิ์ ที่ได้นี้ จะทำการเลือกออกมาเป็นจำนวนที่เท่าๆ กันเช่น 10% 20% 30% . . . เป็นต้น ของการ การแปลง แต่ละแบบ สำหรับในการเลือกค่าของสัมประสิทธิ์ นี้จะกระทำโดยพิจารณาจากค่าขนาดของสัมประสิทธิ์ นั้นๆ เรียงจากมากไปหาน้อย

จากข้อมูลของค่าสัมประสิทธิ์ ที่เลือกออกมานี้ จะนำมากระทำ Inverse transform เพื่อให้ได้ออกมาเป็นข้อมูลภาพ ข้อมูลภาพที่ได้นี้ จะถูกนำมาเปรียบเทียบกับ ข้อมูลของภาพที่เป็นต้นแบบ โดยใช้วิธีการของค่าความผิดพลาด (e_{rms})

ขบวนการ แปลง ที่มีประสิทธิภาพ จะให้ค่าของ ความผิดพลาดน้อย หรือจะกล่าวได้ว่าขบวนการ แปลง ที่เหมาะสมนี้ สามารถหาความสัมพันธ์กันของข้อมูลภาพ ออกมาได้ในจำนวนของค่าสัมประสิทธิ์ ที่น้อยที่สุดนั้นเอง สำหรับผลการทดลองจะได้แสดงในหัวข้อของการทดลอง บทที่ 4

บทที่ 4
การประยุกต์ใช้การแปลงเวฟเล็ตในการลดขนาดข้อมูลภาพ

4.1 บทนำ

ในบทนี้จะได้กล่าวถึงการนำหลักการที่ได้ศึกษามาแล้วในบทที่ 2 และ 3 มาประยุกต์ใช้ในการลดขนาดของข้อมูลภาพ ขาวดำ เนื้อหาในบทนี้จะแบ่งออกเป็น 2 ส่วนใหญ่ๆ ในส่วนแรกจะแสดงถึงการนำ ขบวนการแปลงเวฟเล็ตมาใช้กับข้อมูลภาพ ประกอบด้วยหัวข้อย่อย ที่แสดงถึงขบวนการแปลงเวฟเล็ต การแปลงเวฟเล็ตของภาพและผลที่ได้ และการเลือกเวฟเล็ตแม่ สำหรับส่วนที่ สองจะได้แสดงถึง ขบวนการลดขนาดข้อมูลค่าสัมประสิทธิ์เวฟเล็ตที่ได้ แบ่งเป็นขั้นตอนย่อยคือ การลดขนาดข้อมูลที่มีการสูญเสีย ซึ่งจะมีผลต่อคุณภาพของภาพผลลัพท์ที่ได้ และการลดขนาดข้อมูลที่ไม่มีการสูญเสีย ที่จะเพิ่มอัตราการลดขนาดข้อมูล

ขั้นตอนในการทดลองต่างๆ ที่ได้พัฒนาขึ้นนี้ จะทำโดยใช้โปรแกรม Matlab® V4.2 และโปรแกรมภาษา C และท้ายที่สุดได้พัฒนาโปรแกรมใช้งานขึ้นโดยใช้ Delphi 3 ในบทนี้ จะแสดงเฉพาะสมการสมการคณิตศาสตร์ ขบวนการที่ใช้ในการทดลอง และผลการทดลองที่ได้ ส่วนรายละเอียดของ โปรแกรมจะได้อธิบายไว้ในภาคผนวก

4.2 การกระจายและรวมข้อมูลภาพด้วยเวฟเล็ต

ในหัวข้อนี้ จะเป็นการแสดงถึงการการกระจายและ รวมข้อมูลด้วยเวฟเล็ต การทดลองในส่วนนี้จะทำโดยใช้โปรแกรม Matlab® V4.2 เป็นเครื่องมือ การทดลองจะประกอบด้วย

1. การแปลงเวฟเล็ตของข้อมูลหนึ่งมิติ
   การทดลองในส่วนนี้ จะแสดงให้เข้าใจถึง การแปลง และ แปลงผกผันเวฟเล็ตของสัญญาณ ตามวิธีการของ Multi resolution wavelet decomposition และวิธีการที่มีประสิทธิภาพ ในการใช้งานจริง
2. การแปลงเวฟเล็ตของข้อมูลภาพ
   การทดลองจะแสดงให้เห็นถึง การแปลงเวฟเล็ตของข้อมูลภาพ และคุณสมบัติที่สำคัญของ ค่าสัมประสิทธิ์ที่ได้

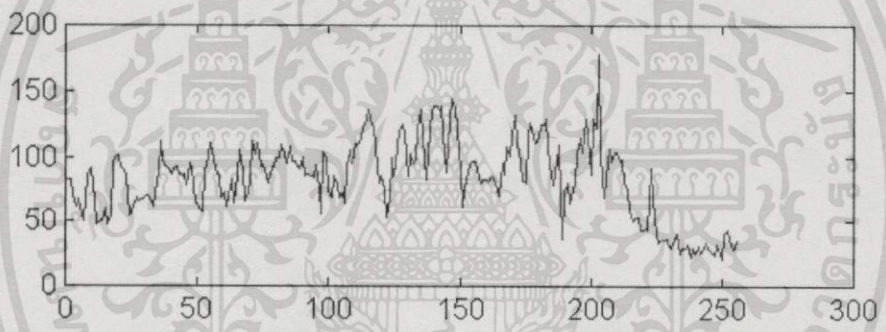
3. วิธีการเลือกใช้ฟังก์ชันของเวฟเล็ตแม่ที่เหมาะสม

เป็นการเปรียบเทียบการเลือก ใช้เวฟเล็ตแม่ ต่างๆ และเปรียบเทียบกับขบวนการแปลงแบบ DCT ที่ใช้ในปัจจุบัน

4.2.1 การแปลงเวฟเล็ต

การแปลงเวฟเล็ต หรือการแตกกระจายข้อมูล ที่จะใช้นั้นจะเป็นการแปลงเวฟเล็ตในแบบของ Discrete time wavelet series เนื่องมาจากการแปลงแบบนี้ ข้อมูลที่ได้ จะไม่มีความซ้ำซ้อนกัน จึงเหมาะที่จะนำมาใช้ เพื่อการลดขนาดข้อมูล ซึ่งการแปลงเวฟเล็ตนี้จะสามารถกระทำได้ด้วยวิธีการของ Multi resolution signal analysis ซึ่งจะประกอบไปด้วยส่วนต่างๆ คือ

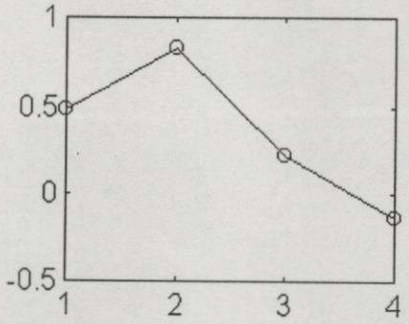
สัญญาณอินพุท ที่ใช้ในการประมวลผลสัญญาณเชิงเลขนี้ จะเป็นสัญญาณ แบบเต็มหน่วยเวลา ที่ได้จากขบวนการ Sampling ดังรูปที่ 4.1 จะเป็นตัวอย่างของสัญญาณ ที่ใช้ในการทดสอบ



รูปที่ 4.1 แสดงสัญญาณ อินพุท ที่ใช้ทดสอบ ซึ่งเป็นสัญญาณขนาด 8 บิต จำนวน 256 จุดข้อมูล

ส่วนของตัวกรอง สามารถที่สร้างได้จากการ Convolution ของสัญญาณ กับ Transfer function ซึ่งในการใช้งานนี้ จะต้องใช้ตัวกรองทั้งหมด 2 ตัว คือ G, H

ตัวกรอง $g(k)$ ที่จะใช้ก็คือ ฟังก์ชันของเวฟเล็ตแม่ ในการทดลองจะเลือกใช้ ฟังชันที่เป็นเวฟเล็ตแม่ คือ Dabuchies 4 [6]



รูปที่ 4.2 แสดงลักษณะของ Daubechies 4 ซึ่งประกอบด้วย 4 จุดข้อมูลคือ [0.4830, 0.8365, 0.2241, -0.1294]

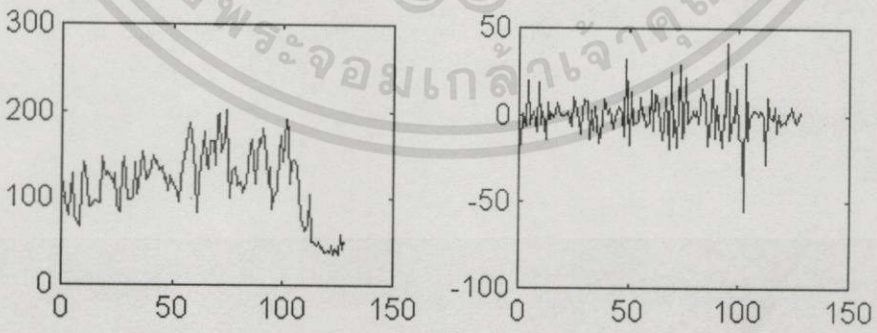
จากที่ได้อธิบายในบทที่ 2 ตัวกรองแต่ละตัวจะสามารถหาได้จาก การคำนวณคือ

$$G = g(k) = [0.4830 \quad 0.8365 \quad 0.2241 \quad -0.1294]$$

$$H = h(k) = (-1)^n g(N-k) = [-0.1294 \quad -0.2241 \quad 0.8365 \quad -0.4830]$$

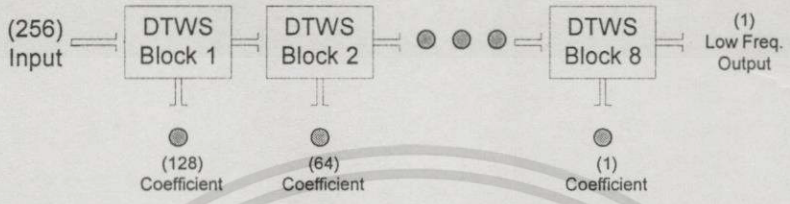
การทำ Down sampling หรือ Decimation เป็นการลดอัตราการสุ่มของสัญญาณสัญญาณลง การลดอัตราการสุ่มนี้ จะกระทำกับสัญญาณที่ผ่านตัวกรองมาแล้ว เพื่อเป็นการลดความซ้ำซ้อนกันของข้อมูลที่ได้จากตัวกรองทั้งสอง ซึ่งจะทำให้สัญญาณ ที่ได้จากตัวกรองนี้ มีจำนวนจุดข้อมูลลดลงครึ่งหนึ่ง

สัมประสิทธิ์เวฟเล็ต สัมประสิทธิ์เวฟเล็ตที่ได้นี้ จะแบ่งออกเป็นสองส่วน ตามตัวกรองที่ใช้คือ G, H ดังแสดงในรูปที่ 4.3



รูปที่ 4.3 แสดงข้อมูลที่ได้จากการแปลงเวฟเล็ต โดยรูป (ซ้าย) เป็นสัญญาณส่วนของความถี่ต่ำ และ (ขวา) เป็นสัญญาณส่วนความถี่สูง หรือสัมประสิทธิ์เวฟเล็ต

จากการทดลองข้างต้นนี้ จะเป็นการแปลง หรือการแตกกระจายเวฟเล็ต หนึ่งระดับ ของสัญญาณ ซึ่งการแตกกระจายของสัญญาณจะสามารถกระทำต่อไปในระดับความละเอียดที่ต่ำลงมาได้ โดยทำการแตกกระจาย ค่าสัมประสิทธิ์ที่ได้ จากส่วนของตัวกรองความถี่ต่ำ (G) ก็จะทำให้ได้ การแตกกระจายสัญญาณในทุกๆ ระดับความละเอียด ดังแสดงในรูปที่ 4.4



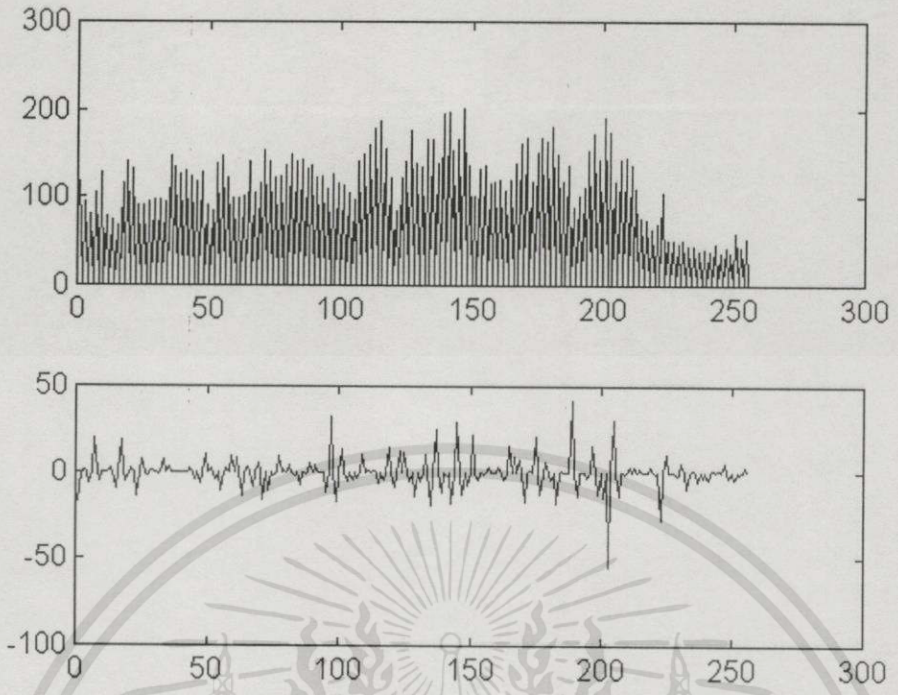
รูปที่ 4.4 แสดงการแตกกระจายเวฟเล็ตของสัญญาณอินพุท โดยตัวเลขในวงเล็บ จะแสดงถึงจำนวนจุดของข้อมูลที่ได้

ทั้งนี้ การแตกกระจายเวฟเล็ตจะกระทำต่อไปได้ เฉพาะในกรณีที่ขนาดของสัญญาณ มีจำนวนจุดของข้อมูลเป็นจำนวนคู่เท่านั้น จากตัวอย่างข้างต้นที่ข้อมูลของสัญญาณอินพุท มีจำนวนจุดข้อมูลทั้งหมด 256 จุดข้อมูล จึงสามารถที่จะทำการแตกกระจายเวฟเล็ตได้ 8 ครั้ง

4.2.2 การแปลงผกผันเวฟเล็ต

การแปลงผกผันเวฟเล็ตจะสามารถทำได้จาก ค่าสัมประสิทธิ์เวฟเล็ต ซึ่งถ้าเวฟเล็ตแม่ที่ใช้เป็นตัวเดียวกัน แล้ว สัญญาณเอาท์พุท ที่ได้จะเหมือนกับสัญญาณอินพุท ทุกประการ โดยมีขั้นตอนดังนี้

<u>การทำ Up sampling</u> โดยการแทรกข้อมูล ศูนย์ เพิ่มลงไปใน ข้อมูลของค่าสัมประสิทธิ์ ที่ได้จากการแปลงเวฟเล็ต เพื่อเป็นการเพิ่มความถี่ของสัญญาณ ให้สูงขึ้น สัญญาณหลังผ่านการทำ Up Sampling แล้วจะได้ดังรูปที่ 4.5



รูปที่ 4.5 แสดงสัญญาณจากรูปที่ 4.3 เมื่อนำมาผ่านขบวนการ Up sampling โดยในรูป(บน) เป็นของสัญญาณในส่วนของความถี่ต่ำ และ (ล่าง) เป็นของสัมประสิทธิ์เวฟเล็ต

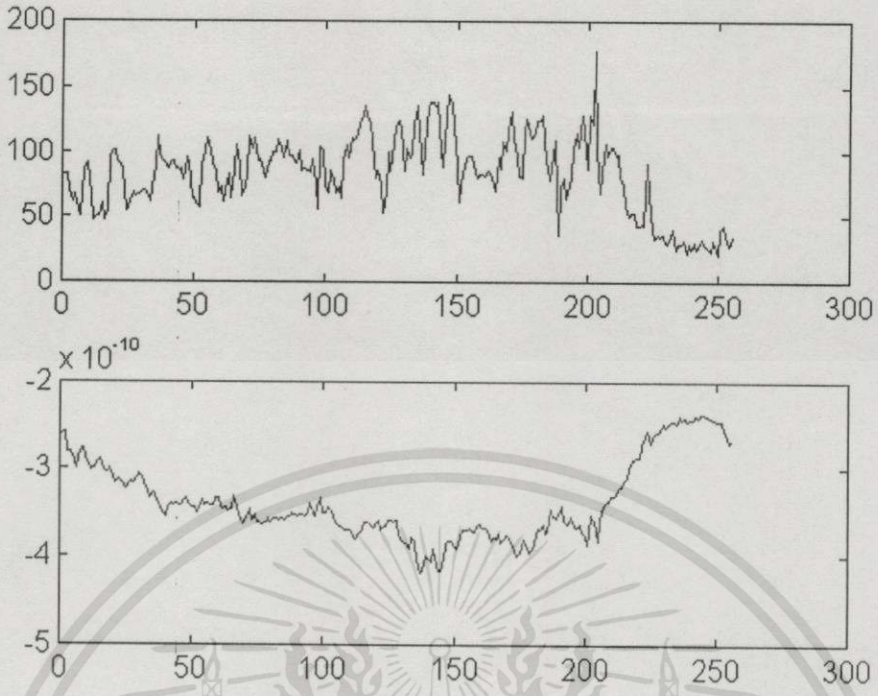
<u>ส่วนของตัวกรอง</u> สามารถที่สร้างได้จากการ Convolution ของสัญญาณ กับ Transfer function ซึ่งในการใช้งานนี้ จะต้องใช้ตัวกรอง $\widetilde{G}, \widetilde{H}$

ตัวกรอง $g(k)$ ที่จะใช้ก็คือ Function ของเวฟเล็ตแม่ ในการทดลองจะเลือกใช้ ฟังชันที่เป็นเวฟเล็ตแม่ คือ Dabuchies 4

ตัวกรองแต่ละตัวจะสามารถหาได้จาก การคำนวณคือ

$$\widetilde{G} = \widetilde{g}(k) = g(N-k) = [-0.1294 \quad 0.2241 \quad 0.8365 \quad 0.4830]$$
$$\widetilde{H} = \widetilde{h}(k) = h(N-k) = [-0.4830 \quad 0.8365 \quad -0.2241 \quad -0.1294]$$

<u>สัญญาณเอาท์พุท</u> สัญญาณเอาท์พุทจะสร้างกลับมาได้จาก การผลรวมของ สัญญาณ ที่ได้จากตัวกรองทั้งสอง ซึ่งสัญญาณเอาท์พุทนี้ จะเหมือนกับสัญญาณใช้เป็นอินพุทในขบวนการแปลงเวฟเล็ต ซึ่งความผิดพลาดที่เกิดขึ้นนี้จะแสดงได้ดังรูปที่ 4.6



รูปที่ 4.6 (บน) แสดงสัญญาณเอาท์พุท ที่ได้จากการรวมกลับเวฟเล็ต (ล่าง) ค่าความผิดพลาดเมื่อเปรียบเทียบกับสัญญาณอินพุท

และด้วยการวัดหาค่าความผิดพลาดของสัญญาณ ทั้งสอง โดยวิธีของการวัดค่า Mean Square Error ซึ่งได้เท่ากับ 1.1608×10^{-019}

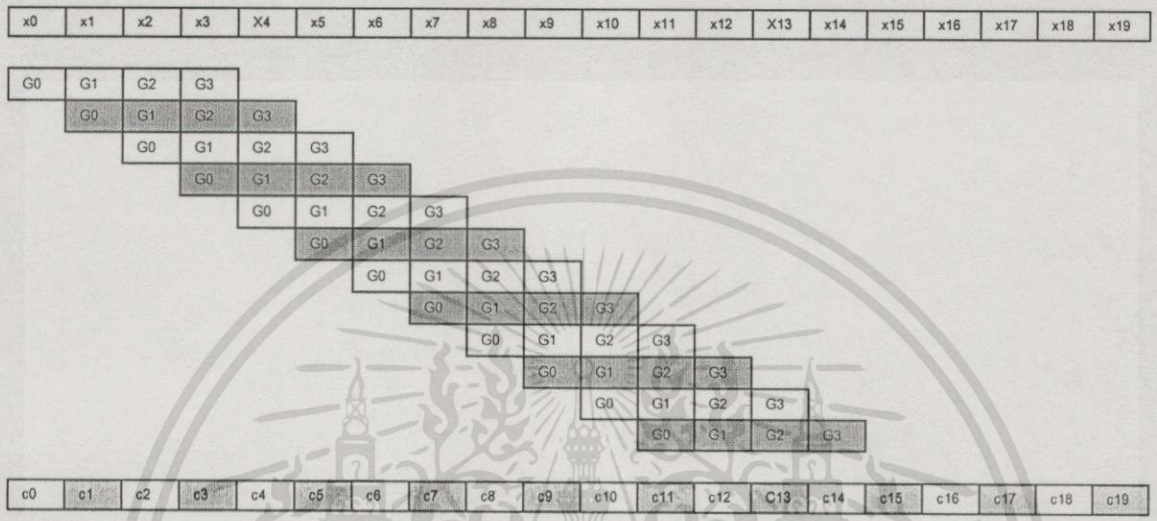
จากผลการทดลองที่ได้จะเห็นได้ว่า ค่าความผิดพลาด ระหว่างสัญญาณ ทั้งสองนี้จะมีอยู่น้อยมาก ซึ่งสาเหตุที่ทำให้ สัญญาณเอาท์พุท มีความแตกต่างจาก อินพุท นั้นก็เนื่องมาจากความผิดพลาดในการปัดเศษของการคำนวณ (Round off Error)

4.2.3 วิธีการที่มีประสิทธิภาพในการแปลงเวฟเล็ต

ขบวนการแปลงเวฟเล็ตแบบ Multi resolution wavelet transform ที่ได้แสดงข้างต้นนี้ เป็นขบวนการที่อธิบาย ไว้ให้ผู้อ่านสามารถเข้าใจถึงหลักการได้โดยง่าย แต่จะเป็นขบวนการที่ไม่มีประสิทธิ์ภาพ เนื่องจาก จะมีการคำนวณที่มากเกินความจำเป็น คือ ในส่วนของการ Convolution ของสัญญาณกับฟังชันของเวฟเล็ต ซึ่งผลลัพท์ที่ได้จะถูกนำมาใช้งานเพียงครึ่งหนึ่งเท่านั้น จากการทำ Decimation by 2 ดังนั้นในการใช้งานจริงของ ขบวนการแปลงเวฟเล็ต จึงต้องถูกปรับปรุงโดยแทนที่จะต้องทำการ Convolution ของสัญญาณทุกๆ จุด ก็จะกระทำเพียงเฉพาะจุดที่ต้องการนำไปใช้เท่านั้น ซึ่งก็จะทำให้ เวลาที่ต้องใช้ในการคำนวณนี้ ลดลงครึ่งหนึ่ง

<u>การแปลงเวฟเล็ต</u> ขบวนการของ Multi resolution wavelet transform นี้จะประกอบด้วยการกระทำ พื้นฐานอยู่สองแบบ คือ การทำ Convolution และการทำ Down sampling สัญญาณ อินพุท A0 จะถูก Convolution กับฟังก์ชันของ Mother wavelet (G) และส่วนกลับ

ของ Mother wavelet (H) จากสัญญาณที่ ผ่านการกรอง หรือ Convolution นี้แล้ว จะถูกนำมาผ่านการ Down sampling เพื่อเป็นการลดความซ้ำซ้อนกันของข้อมูล ขบวนการ Down sampling นี้ จะเป็นการตัดเอาข้อมูลบางส่วนของสัญญาณ ทิ้งไป ขบวนการที่เกิดขึ้นนี้สามารถแสดงได้ดังรูปที่ 4.7



รูปที่ 4.7 แสดงค่าสัมประสิทธิ์ ที่ได้จากการ Convolution ของสัญญาณอินพุท กับเวฟเล็ตแม่ G

จากรูป ถ้าสัญญาณ เอาท์พุท C ถูกทำ Down sampling โดนการ ตัดเอาสัญญาณ ที่ตำแหน่งของ c2 c4 c6 c8 ... c2n ทิ้งไป จะเห็นได้ว่าข้อมูลที่คำนวณได้จากการทำ Convolution บางส่วน ไม่ได้ถูกนำมาใช้งาน ดังนั้น ในการใช้ขบวนการ Convolution ตามปกติ จะเป็นการทำงานที่ไม่มีประสิทธิภาพ แนวทางการปรับปรุงจึงจะสามารถกระทำได้คือ ให้ทำการ Convolution สัญญาณ โดยมีการเลื่อนตำแหน่งที่ละ 2 แทนขบวนการเดิม ก็จะได้ผลลัพท์เช่นเดียวกับวิธีการตามปกติ แต่จะใช้เวลาในการคำนวณลดลงครึ่งหนึ่ง

แต่การทำตามวิธีการที่กล่าวมานี้ เมื่อนำมาประยุกต์ใช้ใน ตัวประมวลผลสัญญาณเชิงเลข (DSP Processor) แล้ว อาจทำได้ไม่สะดวกนัก เนื่องจากชุดคำสั่งที่มีในตัวประมวลผลสัญญาณเชิงเลข ส่วนใหญ่จะถูกออกแบบไว้สำหรับการทำ Convolution ตามปกติ เมื่อพิจารณาตามรูปแสดง การ Convolution แล้ว Down sampling แล้ว จะพบว่าค่าสัมประสิทธิ์ ที่จะนำมาใช้งาน

$$c0 = x0\ G0 + x1\ G1 + x2\ G2 + x3\ G3$$

$$c2 = x2\ G0 + x3\ G1 + x4\ G2 + x5\ G3$$

$$c4 = x4\ G0 + x5\ G1 + x6\ G2 + x7\ G3$$

.

ถ้าทำการสลับที่กันใหม่จะได้

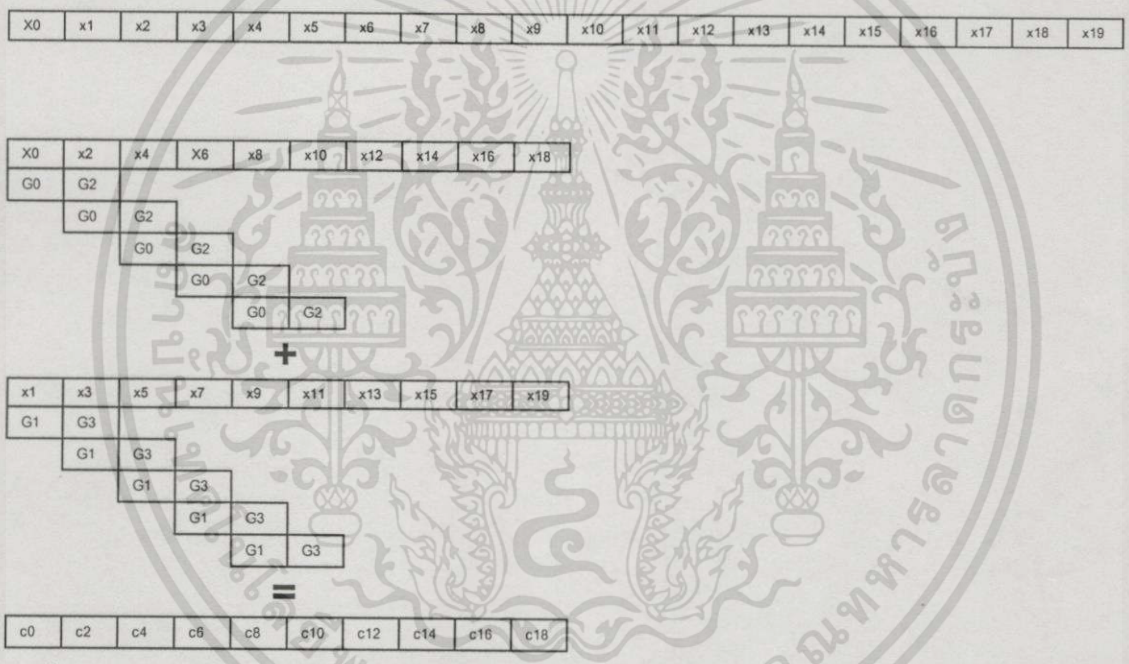
$$c0 = (x0\ G0 + x2\ G2) + (x1\ G1 + x3\ G3)$$

$$c2 = (x2\ G0 + x4\ G2) + (x3\ G1 + x5\ G3)$$

$$c4 = (x4\ G0 + x6\ G2) + (x5\ G1 + x7\ G3)$$

.

จากข้างต้นนี้ จะสังเกตได้ว่า ในเทอมแรก ก็จะคือ การ Convolution ของ สัญญาณ ตำแหน่งคู่ (2n) กับ wavelet function ตำแหน่งคู่เช่นเดียวกัน และ ในเทอมที่สองนั้น ก็จะเป็น การ Convolution ขอสัญญาณ ในตำแหน่งคี่ (2n +1) กับ wavelet function ตำแหน่งคี่ แล้ว จึงทำการบวก สองส่วนนี้ เข้าด้วยกันภายหลัง ดังนั้น การทำงานที่เกิดขึ้นจึงจะแสดงได้ดังรูปที่ 4.8



รูปที่ 4.8 แสดงการแบ่งสัญญาณ ออกเป็นตำแหน่งคู่ และตำแหน่งคี่ แล้วทำการ Convolution

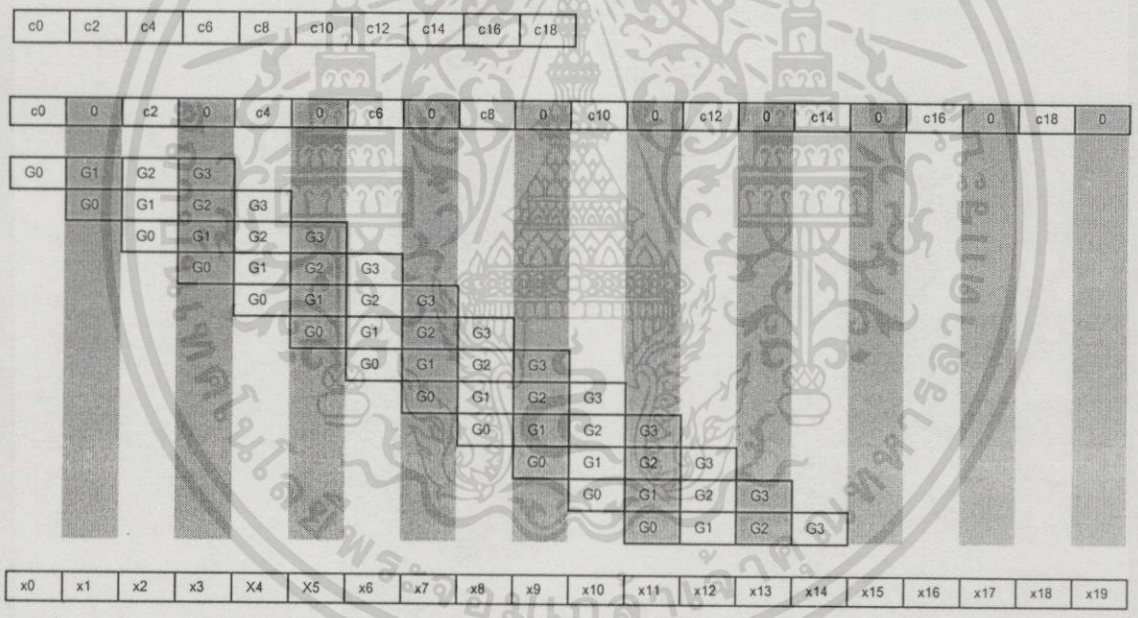
ซึ่งจะสามารถสรุปเป็นขั้นตอนได้ดังนี้

1. ทำการแยก สัญญาณอินพุท และ wavelet function ออกเป็น ข้อมูล ตำแหน่งคู่ กับตำแหน่งคี่
2. ทำการ Convolution ของสัญญาณ ที่แยกออกเป็นสองส่วนนี้
3. รวมผลลัพท์ที่ได้จากการ Convolution

ขบวนการดังที่กล่าวข้างต้นนี้ จะมีข้อดีคือ

1. การคูณเลข ที่น้อยกกว่า ขบวนการตามปกติ ครึ่งหนึ่ง ตามที่ได้อธิบายข้างต้น
2. การแยกสัญญาณออกเป็น ตำแหน่งคู่ และ คี่ ซึ่งจะเป็นวิธีการเดียวกันกับการทำ Down sampling ดังนั้นจึงเปรียบเทียบได้กับวิธีการนี้ จะมีการทำ Down sampling ของสัญญาณเพียงครั้งเดียว
3. การที่สัญญาณถูกแยก และการ Convolution ที่แบ่งออกเป็นสองส่วน จะทำให้การพัฒนาให้ขบวนการนี้เป็นการทำงาน ในแบบของ Parallel processing ได้โดยง่าย ในกรณีที่มีหน่วยประมวลผลหลายตัว

<u>การแปลงผกผันเวฟเล็ต</u> การแปลงผกผันเวฟเล็ต ก็เช่นเดียวกันกับการแปลงเวฟเล็ต ซึ่งจะมีการกระทำพื้นฐาน คือ การทำ Convolution และการทำ Up sampling จาก Wavelet coefficient (c) ที่เข้ามา จะถูกทำ Up sampling โดยการ เติมข้อมูล "0" แทรกเข้าไป จากนั้นจึงทำการ Convolution กับ ส่วนกลับของ wavelet function ซึ่งขบวนการนี้สามารถแสดงได้ดังรูปที่ 4.9



รูปที่ 4.9 แสดงการแปลงผกผัน เวฟเล็ต ของค่าสัมประสิทธิ์ c

จากรูปแสดงการทำงาน ข้างต้น จะเห็นได้ว่า

$$x0 = c0\ G0 + 0\ G1 + c2\ G2 + 0\ G3$$

$$x1 = 0\ G0 + c2\ G1 + 0\ G2 + c4\ G3$$

$$x2 = c2\ G0 + 0\ G1 + c4\ G2 + 0\ G3$$

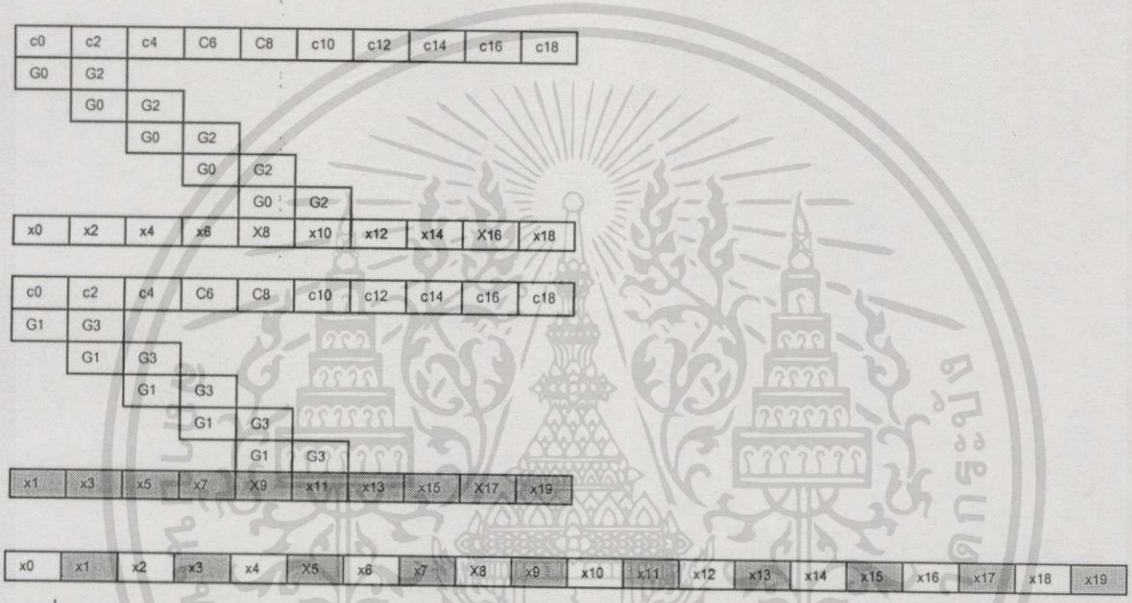
ซึ่งจะมีการ คูณ ค่าของ wavelet function กับ ข้อมูล "0" ซึ่งเป็นสิ่งที่ไม่จำเป็น ดังนั้น ถ้าตัดออกไปจะได้เป็น

$$x0 = c0\ G0 + c2\ G2$$

$$x1 = c2\ G1 + c4\ G3$$

$$x2 = c2\ G0 + c4\ G2$$

ซึ่งก็จะคือ การ Convolution ของ wavelet coefficient กับ wavelet function ตำแหน่งคู่ (G0 G2) สำหรับตำแหน่งของ X เป็นคู่ (2n) และที่ตำแหน่งของ X ที่เป็นคี่ ก็จะเป็นการ Convolution ของ wavelet coefficient กับ wavelet function ที่ตำแหน่ง คี่ (G1 G3) แล้วจึงนำข้อมูลทั้งสองส่วนนี้ มาจัดเรียงเข้าด้วยกัน



รูปที่ 4.10 แสดงการแปลงผกผันเวฟเล็ตของค่าสัมประสิทธิ์ที่ ถูกแยกออกเป็นสองส่วน

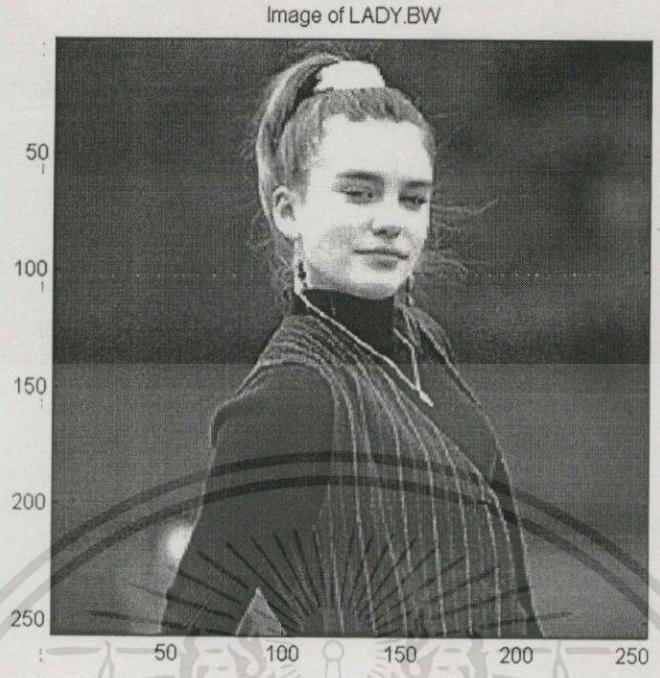
ซึ่งจะสรุปเป็นขั้นตอนได้ดังนี้

1. ทำการแยก wavelet function ออกเป็น ข้อมูล ตำแหน่งคู่ กับตำแหน่งคี่
2. ทำการ Convolution ของ wavelet coefficient กับ wavelet function ที่แยกออกเป็นสองส่วนนี้
3. ทำการเรียงสลับ ของสัญญาณ ที่ได้จากการ Convolution ทั้งสอง

การทำ Invert wavelet transform ด้วยวิธีการที่กล่าวข้างต้น ก็จะมีข้อดี เช่นเดียวกันกับการแปลงเวฟเล็ต ดังที่ได้อธิบายไปแล้ว

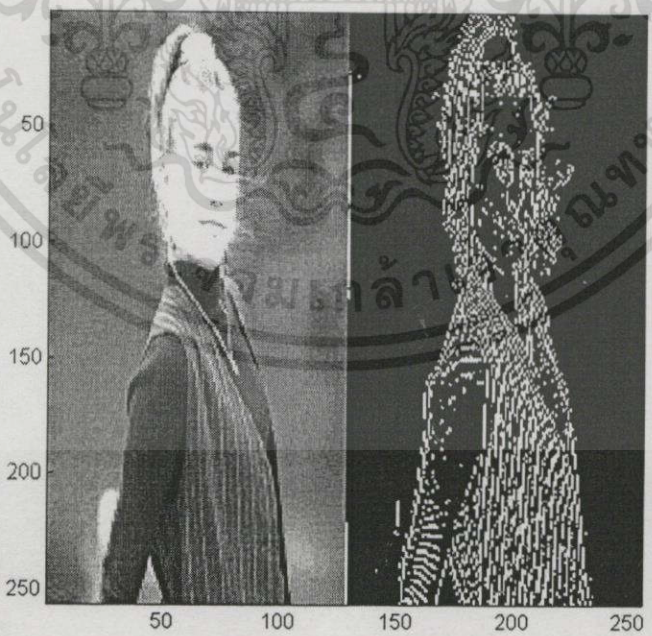
4.2.4 การแปลงเวฟเล็ตของข้อมูลภาพ

ข้อมูลภาพ เป็นข้อมูลของสัญญาณ ที่อยู่ในสองมิติ คือ ในแกนของระยะในแนวนอน (X) และ ระยะในแนวแกนตั้ง (Y) สำหรับข้อมูลภาพ ในโปรแกรม Matlab® V4.2 จะถูกเก็บอยู่ในรูปของ Matrix สองมิติ จากภาพต้นแบบ ขนาด 256x256 จุดภาพ ดังรูปที่ 4.11



รูปที่ 4.11 ภาพต้นแบบที่ใช้ในการแปลงเวฟเล็ต

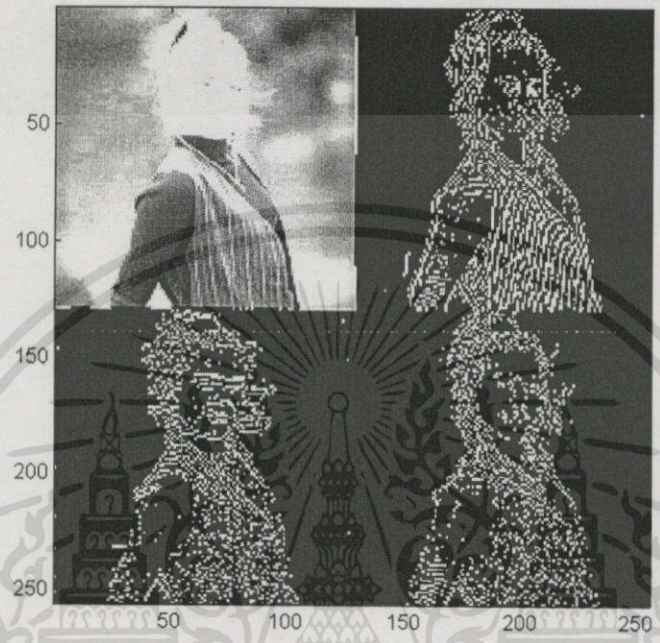
การแปลง เวฟเล็ต ของข้อมูลภาพจะทำ โดยเลือกข้อมูลของแต่ละแถว ตามแนวนอน มาผ่านการแปลงเวฟเล็ตแบบ หนึ่งมิติ จนครบข้อมูลของทุกแถว ดังแสดงในรูปที่ 4.12



รูปที่ 4.12 แสดงภาพที่ได้จากการแปลงเวฟเล็ตของเส้นภาพตามแนวนอน

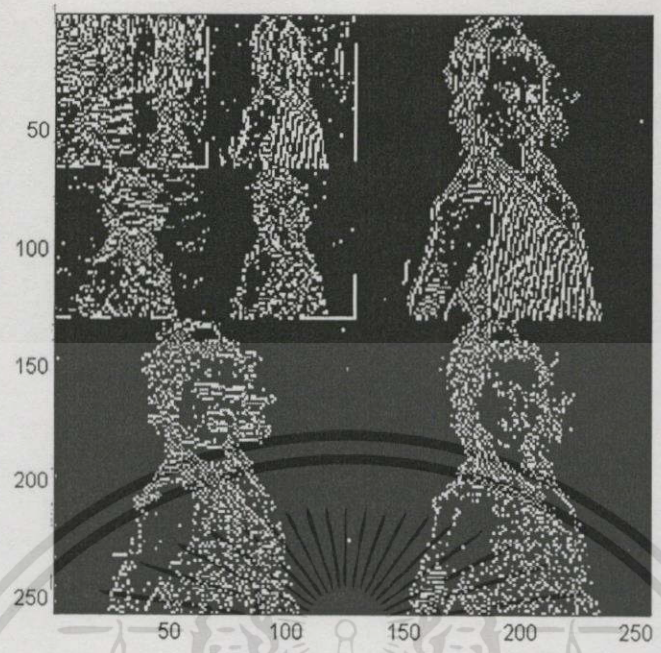
หมายเหตุ ภาพของค่าสัมประสิทธิ์ที่ได้นี้ ได้ถูกปรับแต่งเพื่อการ นำมาแสดง

จากนั้น ข้อมูลที่ได้ จะถูกแปลงเวฟเล็ตอีกครั้ง ตามแนวตั้ง แต่ในทางปฏิบัติแล้ว จะใช้วิธีการ Transpose ของ Matrix ข้อมูลภาพแทน แล้วใช้ขบวนการแปลงเวฟเล็ตตามแนวนอนอีกครั้ง จึงได้ผลลัพท์ดังรูปที่ 4.13



รูปที่ 4.13 แสดงภาพที่ได้จากการแปลงเวฟเล็ตของเส้นภาพตามแนวนอน และแนวตั้ง
หมายเหตุ ภาพของค่าสัมประสิทธิ์ที่ได้นี้ ได้ถูกปรับแต่งเพื่อการ นำมาแสดง

ผลที่ได้นี้ จะเป็นการแปลงเวฟเล็ต แบบ Multi resolution ของข้อมูลภาพในหนึ่งระดับ ซึ่งการแปลงเวฟเล็ตในระดับต่อๆ ไป จะทำได้โดยการนำเอาข้อมูลในส่วนของความถี่ต่ำ มาทำการแปลงเวฟเล็ตต่อไป ด้วยวิธีการเช่นเดิม ซึ่งจะได้ผลลัพท์ดังแสดงในรูปที่ 4.14



รูปที่ 4.14 แสดงภาพที่ได้จากการแปลงเวฟเล็ตของเส้นภาพตามแนวนอน และแนวตั้ง จำนวน 8 ครั้ง

หมายเหตุ ภาพของค่าสัมประสิทธิ์ที่ได้นี้ ได้ถูกปรับแต่งเพื่อการ นำมาแสดง

4.2.5 การเลือก Transform ที่เหมาะสม

การทดลอง ในหัวข้อของ การเลือก การแปลง ที่เหมาะสมนี้ จะแสดงเปรียบเทียบ ประสิทธิภาพในการใช้งานของ การแปลง แบบต่างๆ คือ Discrete Cosine Transform (DCT) Discrete Wavelet Transform (DWT) ของเวฟเล็ตแม่ต่างๆ ในการที่จะเลือกใช้ การแปลง ที่เหมาะสมสำหรับใช้ในขบวนการลดขนาดข้อมูลภาพนี้ จะขึ้นอยู่กับความสามารถในการลดขนาดของข้อมูลโดยขบวน การแปลง นั้นๆ

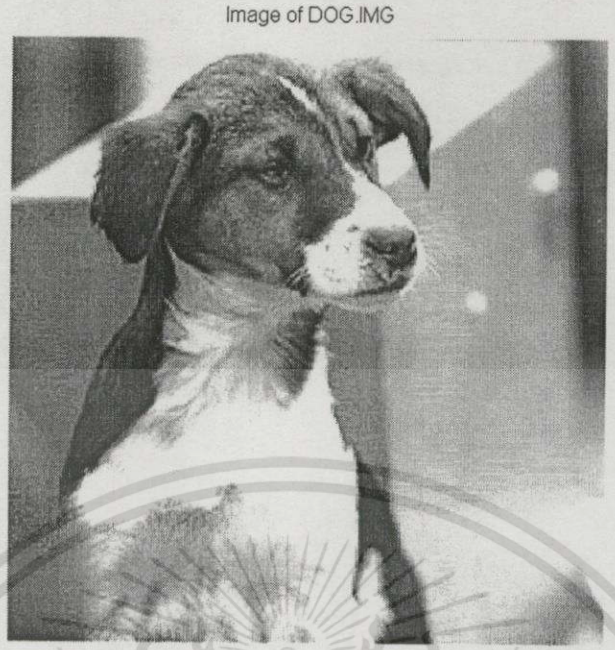
สำหรับความสามารถของขบวน การแปลง แบบต่างๆ นี้สามาถกระทำได้โดยการทดลองดังขั้นตอนต่อไปนี้

1. ทำการแปลงข้อมูลภาพที่ใช้เป็นต้นแบบด้วย การแปลง ต่างๆ คือ DCT DWT สำหรับใน และ การแปลง ก็ จะได้ค่าของสัมประสิทธิ์ ออกมา
2. จากค่าของสัมประสิทธิ์ ที่ได้ จะทำการเลือกออกมาเป็นจำนวนที่เท่าๆ กันเช่น 10% 20% 30% . . . เป็นต้น ของการ แปลง แต่ละแบบ สำหรับในการเลือกค่าของสัมประสิทธิ์ นี้จะกระทำโดยพิจารณาจากค่าขนาดของสัมประสิทธิ์ นั้นๆ เรียงจากมากไปหาน้อย

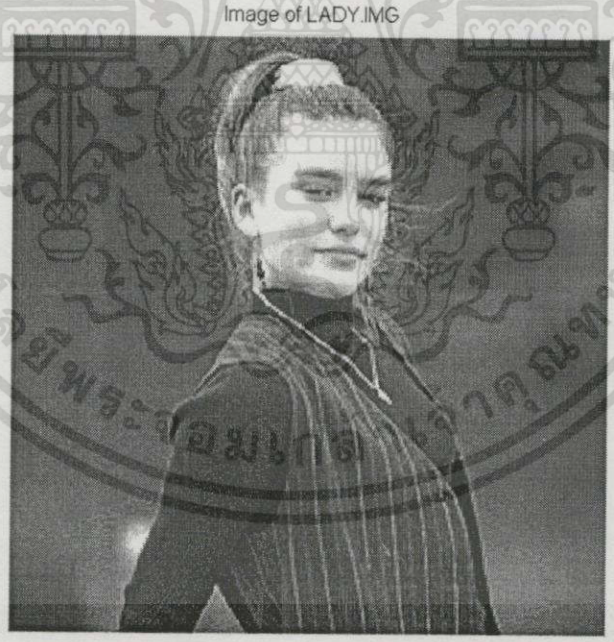
3. จากข้อมูลของค่าสัมประสิทธิ์ ที่เลือกออกมานี้ จะนำมากระทำ Inverse transform เพื่อให้ได้ออกมาเป็นข้อมูลภาพ ข้อมูลภาพที่ได้นี้ จะถูกนำมาเปรียบเทียบกับ ข้อมูลของภาพที่เป็นต้นแบบ โดยใช้วิธีการของค่าความผิดพลาด (e_{rms})

ขบวนการ แปลง ที่มีประสิทธิภาพ จะให้ค่าของ ความผิดพลาดน้อย หรือจะกล่าวได้ว่าขบวนการ แปลง ที่เหมาะสมนี้ สามารถหาความสัมพันธ์กันของข้อมูลภาพ ออกมาได้ในจำนวนของค่าสัมประสิทธิ์ ที่น้อยที่สุดนั้นเอง สำหรับผลการที่แสดงต่อไปนี้ จะใช้ภาพต้นแบบคือ LENA.IMG, DOG.IMG, LADY.IMG และ PLANT.IMG ดังแสดงในรูปที่ 4.15, 4.16, 4.17 และ 4.18 ตามลำดับ

รูปที่ 4.15 แสดงภาพต้นแบบที่ใช้ในการทดลอง ภาพที่1 (LENA.IMG)

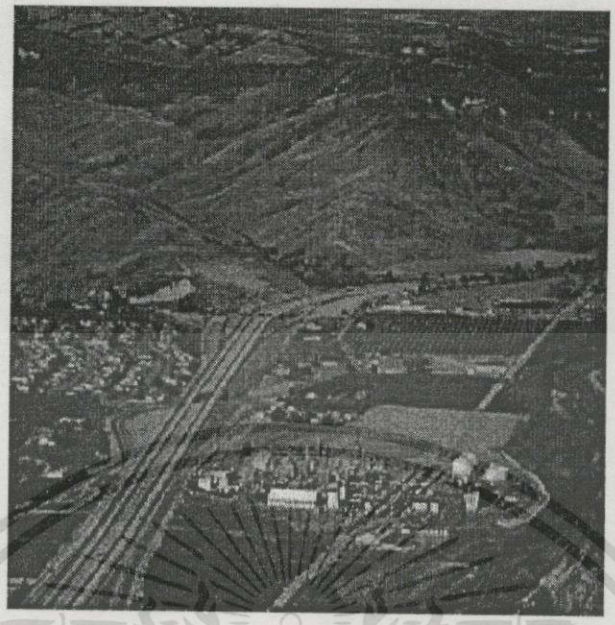


รูปที่ 4.16 แสดงภาพต้นแบบที่ใช้ในการทดลอง ภาพที่ 2 (DOG.IMG)



รูปที่ 4.17 แสดงภาพต้นแบบที่ใช้ในการทดลอง ภาพที่ 3 (LADY.IMG)

Image of PLANT.IMG



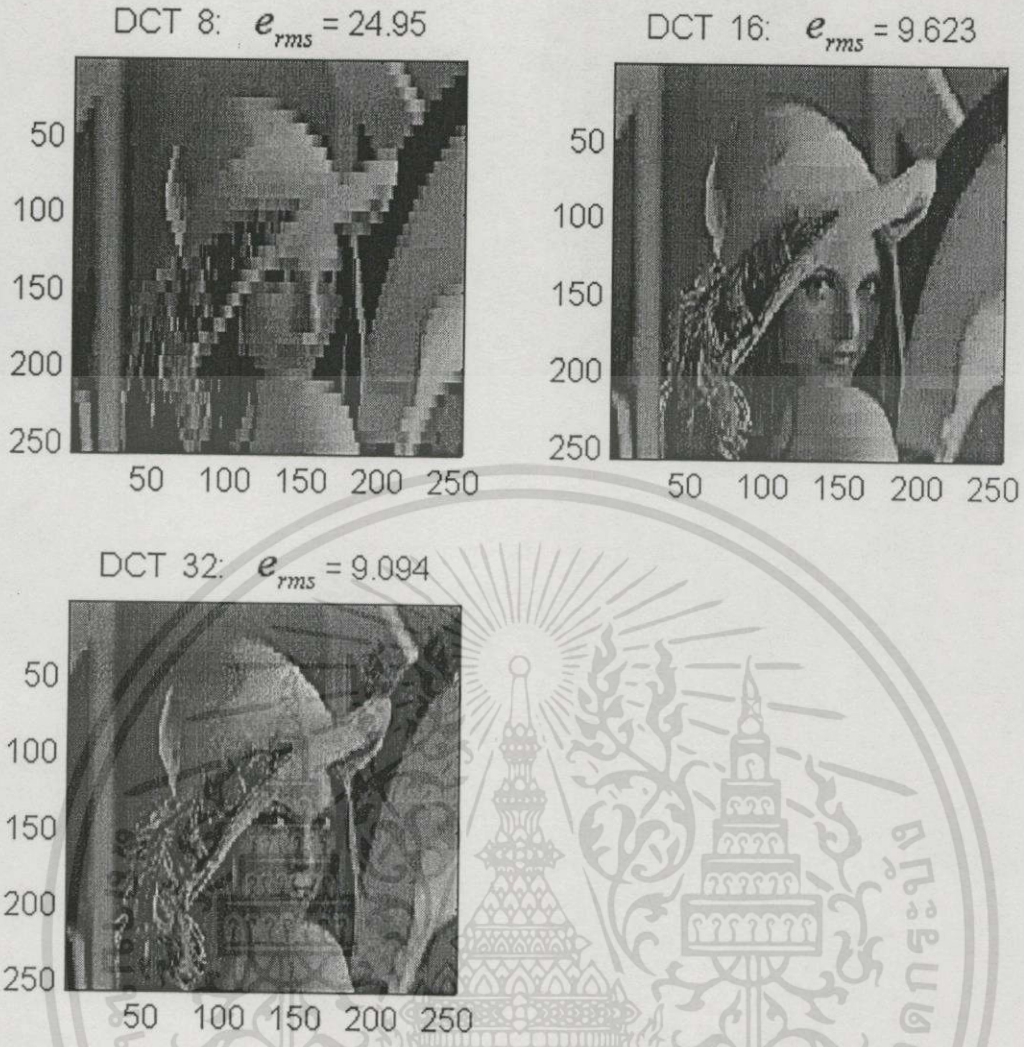
รูปที่ 4.18 แสดงภาพต้นแบบที่ใช้ในการทดลอง ภาพที่ 4 (PLANT.IMG)

เมื่อใช้ Discrete Cosine Transform การแปลงโดยใช้ DCT นี้ ทั่วไปแล้วจะทำการแบ่งข้อมูล เป็น Block ย่อยๆ ขนาด 8 16 และ 32 จุดข้อมูล ซึ่งได้ผลการทดลองดังตารางที่ 4.1

ตารางที่ 4.1 แสดงผลการทดลองที่ได้จากการทำ DCT ของภาพต้นแบบที่ 1

วิธีการ	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
DCT 8	60.8694	24.9482	8.4667	5.2490	3.8330	3.0103	2.4509	2.0319	1.6943
DCT 16	29.4087	9.6229	6.2990	4.6591	3.6449	2.9526	2.4466	2.0463	1.7159
DCT32	15.3777	9.0936	6.4930	4.9668	3.9371	3.1955	2.6378	2.0319	1.6943

และเมื่อนำข้อมูลที่เลือกมาสร้างกลับเป็นภาพใหม่ แล้วจะได้ภาพดังแสดงในรูปที่ 4.19 ในทำนองเดียวกัน เมื่อทำการทดลองโดยใช้ภาพทดสอบ DOG, LADY และ PLANT ก็จะได้ข้อมูลของค่าความผิดพลาด ตามตารางที่ 4.2, 4.3 และ 4.4 ตามลำดับ สำหรับตารางที่ 4.5 จะเป็นการผลการทดลองที่ได้จากภาพทั้งหมดมาเฉลี่ย



รูปที่ 4.19 แสดงภาพผลการทดลองที่ได้ของภาพต้นแบบที่1 ที่จำนวนค่าสัมประสิทธิ์ 10% ภาพที่ 1 (DCT8) ภาพที่ 2 (DCT16) ภาพที่ 3 (DCT32)

ตารางที่ 4.2 แสดงผลการทดลองที่ได้จากการทำ DCT ของภาพต้นแบบที่ 2

วิธีการ	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
DCT 8	111.152	43.4382	7.5819	4.5130	3.0456	2.1443	1.5431	1.1299	0.8487
DCT 16	47.1937	8.6253	5.4197	3.7706	2.7065	1.9747	1.4596	1.0906	0.8295
DCT32	14.3835	7.6895	5.2779	3.8387	2.8476	2.1283	1.5958	1.2006	0.9149

ตารางที่ 4.3 แสดงผลการทดลองที่ได้จากการทำ DCT ของภาพต้นแบบที่ 3

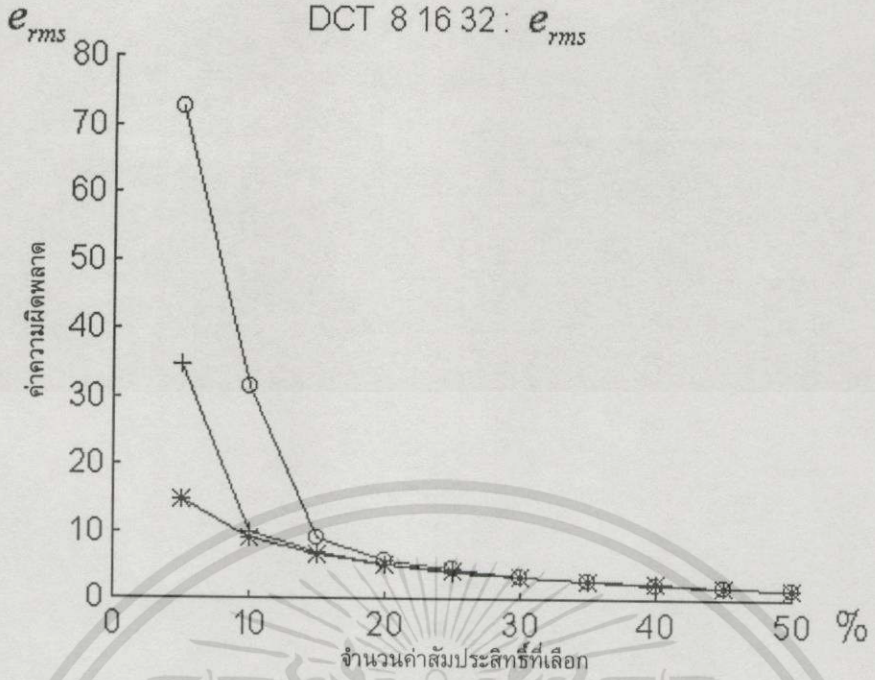
วิธีการ	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
DCT 8	64.2840	29.0020	6.1541	3.3157	2.1857	1.5486	1.1408	0.8716	0.6867
DCT 16	31.5109	7.0185	4.0418	2.6680	1.8742	1.3758	1.0404	0.8133	0.6501
DCT32	11.4776	6.1573	3.9202	2.6758	1.9071	1.4095	1.0693	0.8317	0.6646

ตารางที่ 4.4 แสดงผลการทดลองที่ได้จากการทำ DCT ของภาพต้นแบบที่ 4

วิธีการ	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
DCT 8	54.2250	27.5531	13.4094	10.1836	8.2955	6.9007	5.8033	4.9002	4.1244
DCT 16	30.7253	14.2039	11.1829	9.2767	7.8286	6.6649	5.6858	4.8470	4.1165
DCT32	17.7022	13.2993	10.8984	9.1927	7.8554	6.7481	5.7979	4.9702	4.2376

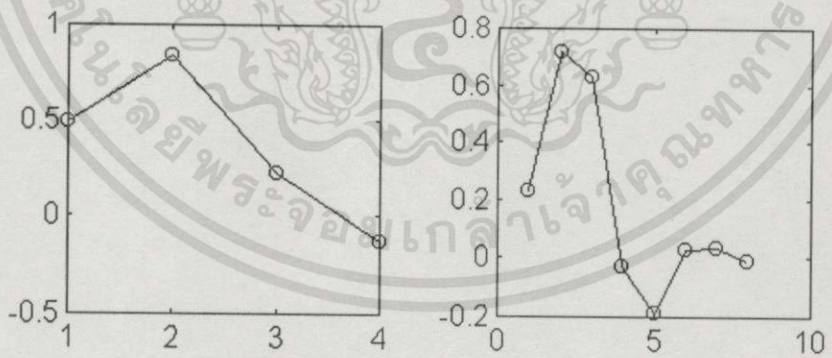
ตารางที่ 4.5 แสดงผลการทดลองที่ได้เฉลี่ยจากการทำ DCT

วิธีการ	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
DCT 8	72.6325	31.2354	8.9030	5.8153	4.3399	3.4010	2.7345	2.2334	1.8386
DCT 16	34.7097	9.8676	6.7358	5.0936	4.0136	3.2420	2.6581	2.1993	1.8280
DCT32	14.7353	9.0599	6.6474	5.1685	4.1368	3.3704	2.7752	2.2992	1.9121



รูปที่ 4.20 แสดงกราฟผลการทดลองที่ได้ DCT8 (o) DCT16 (+) DCT32(*)

<u>เมื่อใช้เวฟเล็ตแม่ Daubechies [6]</u> การแปลงเวฟเล็ต โดยใช้ฟังก์ชันเวฟเล็ตแม่ ของ Daubechies ซึ่งฟังก์ชันของ Daubechies นี้สามารถที่จะแบ่งเป็น Daubechies 4 Dauebechies 8 และอื่นๆ ตามจำนวนของจุดข้อมูล ของเวฟเล็ต ในการทดลองนี้ ได้เลือกใช้เพียง Daubechies 4 และ 8 เท่านั้น เนื่องจากการใช้เวฟเล็ตแม่ที่มีจำนวนจุดข้อมูลมากๆ แล้วจะทำการคำนวณได้ช้า ลักษณะของเวฟเล็ตแม่ที่ใช้เป็นดังรูปที่ 4.21



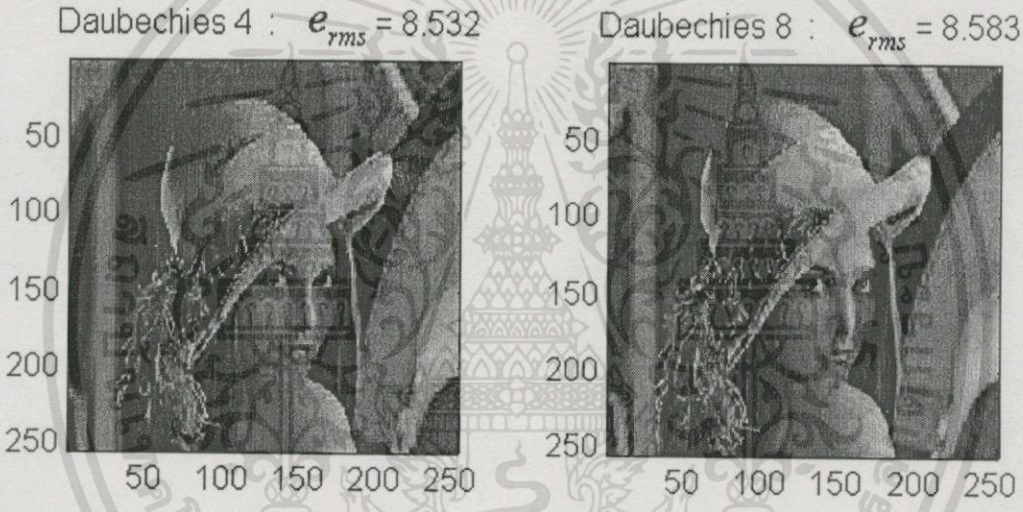
รูปที่ 4.21 แสดงลักษณะของ Daubechies 4 (ซ้าย) ซึ่งประกอบด้วย 4 จุดข้อมูลคือ [0.4830, 0.8365, 0.2241, -0.1294] และ Daubechies 8 (ขวา) ซึ่งประกอบด้วย 8 จุดข้อมูลคือ [0.2304, 0.7148, 0.6309, -0.0280, -0.1870, 0.0308, 0.0329, -0.0106]

ผลการทดลองที่ได้จากการใช้ภาพต้นแบบ LENA เป็นดังตารางที่ 4.6 และภาพที่สร้างกลับมาได้จะแสดงดังรูปที่ 4.22 และเมื่อทำการทดลองกับภาพต้นแบบ DOG, LADY, PLANT จะได้ผลการทดลอง ดังตารางที่ 4.7, 4.8 และ 4.9 ตามลำดับ และในตารางที่ 4.10 จะเป็นผลการ

ทดลองที่ได้เฉลี่ย และเมื่อนำข้อมูลที่เลือกมาสร้างกลับเป็นภาพใหม่ แล้วจะได้ภาพดังแสดงในรูปที่ 4.23, 4.24 และ 4.25 ตามลำดับ

ตารางที่ 4.6 แสดงผลการทดลองที่ได้จากการทำ WT ด้วยเวฟเล็ตแม่ Daubechies ของภาพต้นแบบ LENA

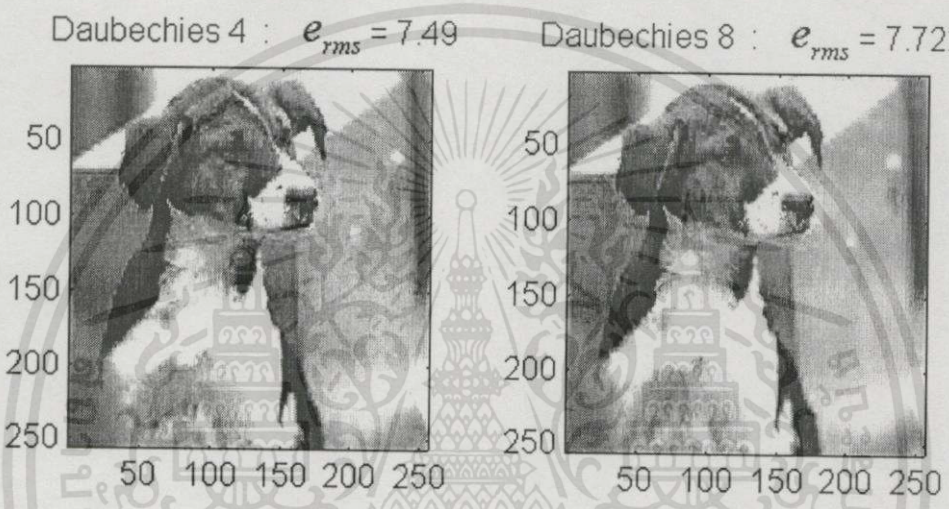
Wavelet	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Daubehies 4	13.9808	8.5318	5.8922	4.3784	3.4255	2.7759	2.3009	1.9281	1.6183
Daubehies 8	13.6513	8.5831	6.0309	4.5024	3.5234	2.8464	2.3501	1.9632	1.6432



รูปที่ 4.22 แสดงภาพผลการทดลองที่ได้ของภาพต้นแบบ LENA ที่จำนวนค่าสัมประสิทธิ์ 10% ภาพที่1 (Daubechies 4) ภาพที่ 2 (Daubechies 8)

ตารางที่ 4.7 แสดงผลการทดลองที่ได้จากการทำ WT ด้วยเวฟเล็ตแม่ Daubechies ของภาพต้นแบบ DOG

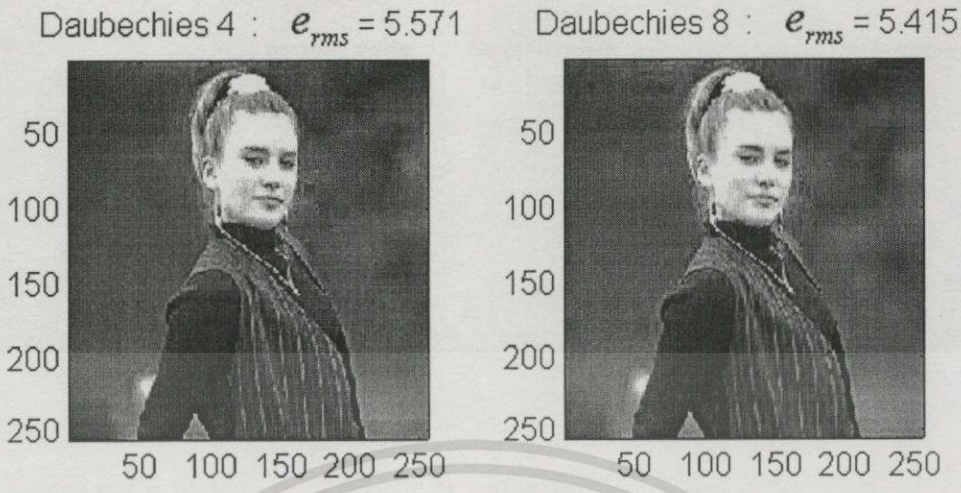
Wavelet	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Daubehies 4	12.9822	7.4899	4.9758	3.4991	2.5386	1.8633	1.3760	1.0256	0.7809
Daubehies 8	13.2931	7.7217	5.1570	3.6488	2.6365	1.9329	1.4245	1.0622	0.8056



รูปที่ 4.23 แสดงภาพผลการทดลองที่ได้ของภาพต้นแบบ DOG ที่จำนวนค่าสัมประสิทธิ์ 10% ภาพที่1 (Daubechies 4) ภาพที่ 2 (Daubechies 8)

ตารางที่ 4.8 แสดงผลการทดลองที่ได้จากการทำ WT ด้วยเวฟเล็ตแม่ Daubechies ของภาพต้นแบบ LADY

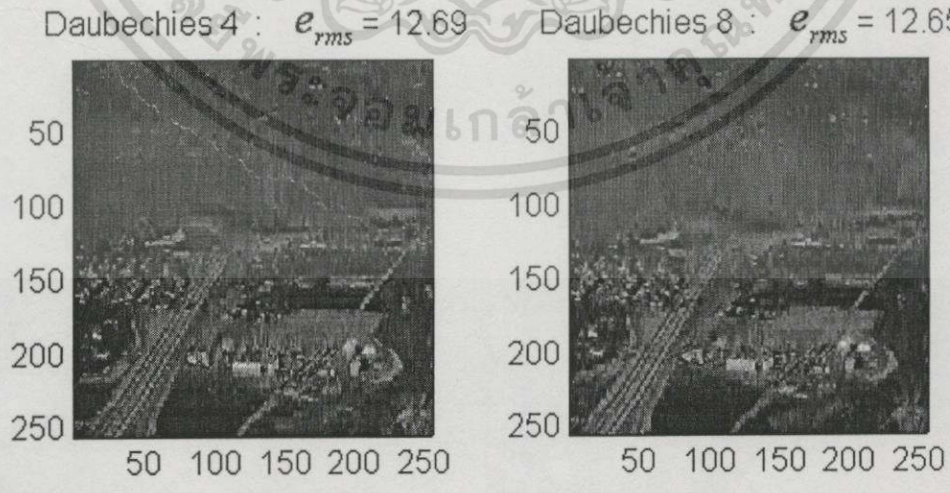
Wavelet	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Daubehies 4	10.3349	5.5713	3.4752	2.3510	1.6883	1.2648	0.9765	0.7767	0.6300
Daubehies 8	9.6797	5.4146	3.4240	2.3277	1.6692	1.2396	0.9511	0.7530	0.6101



รูปที่ 4.24 แสดงภาพผลการทดลองที่ได้ของภาพต้นแบบ LADY ที่จำนวนค่าสัมประสิทธิ์ 10% ภาพที่1 (Daubechies 4) ภาพที่ 2 (Daubechies 8)

ตารางที่ 4.9 แสดงผลการทดลองที่ได้จากการทำ WT ด้วยเวฟเล็ตแม่ Daubechies ของภาพต้นแบบ PLANT

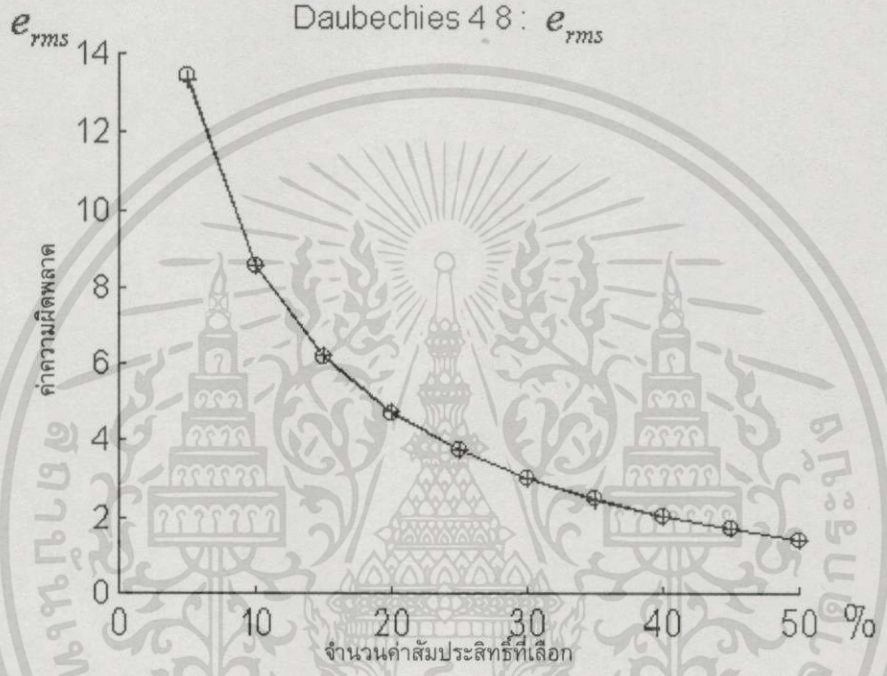
Wavelet	จำนวนค่าสัมประสิทธิ์ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Daubehies 4	16.4229	12.6881	10.3505	8.6441	7.3055	6.2136	5.2940	4.5016	3.8111
Daubehies 8	16.5937	12.6465	10.2362	8.5251	7.1972	6.1103	5.1954	4.4177	3.7394



รูปที่ 4.25 แสดงภาพผลการทดลองที่ได้ของภาพต้นแบบ PLANT ที่จำนวนค่าสัมประสิทธิ์ 10% ภาพที่1 (Daubechies 4) ภาพที่ 2 (Daubechies 8)

ตารางที่ 4.10 แสดงผลการทดลองที่ได้เฉลี่ยจากการทำ WT ด้วยเวฟเล็ตแม่ Daubechies

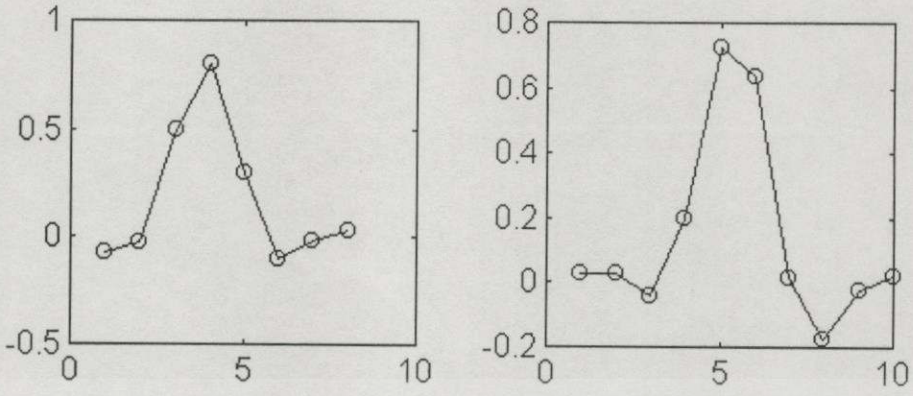
Wavelet	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Daubehies 4	13.4302	8.5703	6.1734	4.7182	3.7395	3.0294	2.4868	2.0580	1.7101
Daubehies 8	13.3045	8.5915	6.2120	4.7510	3.7566	3.0323	2.4803	2.0491	1.6996



รูปที่ 4.26 แสดงกราฟผลการทดลองที่ได้เฉลี่ย Daubechies 4 (o) และ Daubechies 8 (+)

<u>เมื่อใช้เวฟเล็ตแม่ Symmlet</u> การแปลงเวฟเล็ต โดยใช้ฟังก์ชันเวฟเล็ตแม่ ของ Symmlet ซึ่งฟังก์ชันของ Symmlet นี้สามารถที่จะแบ่งเป็น Symmlet 4 Symmlet 8 และอื่นๆ ตามจำนวนของจุดข้อมูล ของเวฟเล็ต ในการทดลองนี้ ได้เลือกใช้เพียง Symmlet 4 และ 8 เท่านั้น เนื่องจากการใช้เวฟเล็ตแม่ที่มีจำนวนจุดข้อมูลมากๆ แล้ว จะทำการคำนวณได้ช้า ลักษณะของเวฟเล็ตแม่ที่ใช้เป็นดังรูปที่ 4.27

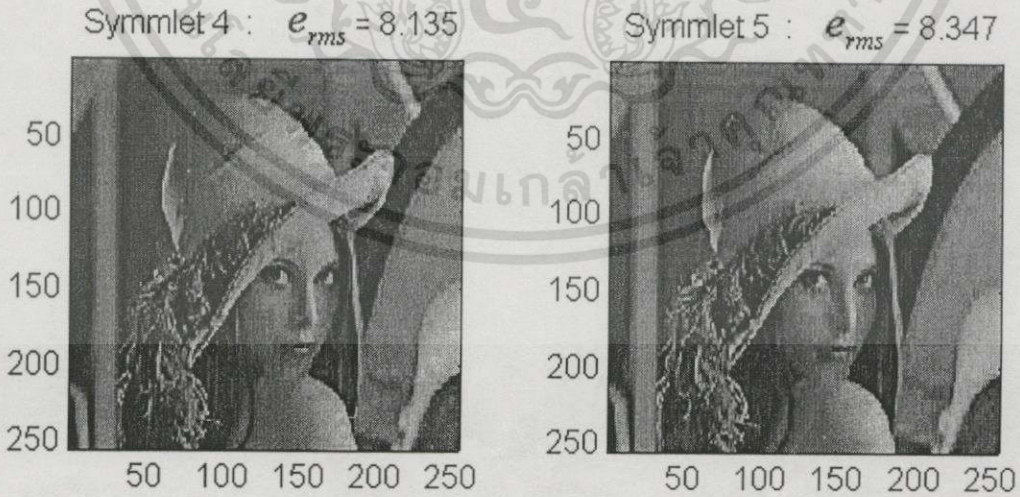
ผลการทดลองที่ได้จากการใช้ภาพต้นแบบ LENA เป็นดังตารางที่ 4.11 และภาพที่สร้างกลับมาได้จะแสดงดังรูปที่ 4.28 และเมื่อทำการทดลองกับภาพต้นแบบ DOG, LADY, PLANT จะได้ผลการทดลอง ดังตารางที่ 4.12, 4.13 และ 4.14 ตามลำดับ และในตารางที่ 4.15 จะเป็นผลการทดลองที่ได้เฉลี่ย



รูปที่ 4.27 แสดงลักษณะของ Symmlet 4 (ซ้าย) ซึ่งประกอบด้วย 8 จุดข้อมูลคือ [-0.0758, -0.0296, 0.4976, 0.8037, 0.2979,, -0.0992, -0.0126, 0.0322] และ Symmlet 5 (ขวา) ซึ่งประกอบด้วย 10 จุดข้อมูลคือ [0.0273, 0.0295, -0.0391, 0.1994, 0.7234, 0.6340, 0.0166, -0.1753, -0.0211, 0.0195]

ตารางที่ 4.11 แสดงผลการทดลองที่ได้จากการทำ WT ด้วยเวฟเล็ตแม่ Symmlets ของภาพต้นแบบ LENA

Wavelet	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Symmlet 4	13.5059	8.1349	5.6812	4.2562	3.3497	2.7257	2.2678	1.9019	1.5962
Symmlet 5	13.5873	8.3466	5.8279	4.3465	3.4095	2.7677	2.2930	1.9186	1.6086



รูปที่ 4.28 แสดงภาพผลการทดลองที่ได้ของภาพต้นแบบ LENA ที่จำนวนค่าสัมประสิทธิ์ 10% ภาพที่1 (Symmlet 4) ภาพที่ 2 (Symmlet 5)

ตารางที่ 4.12 แสดงผลการทดลองที่ได้จากการทำ WT ด้วยเวฟเล็ตแม่ Symmlets ของภาพต้นแบบ DOG

Wavelet	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Symmlet 4	12.2937	7.0998	4.7724	3.3823	2.4699	1.8220	1.3575	1.0206	0.7820
Symmlet 6	12.9204	7.3419	4.8793	3.4494	2.4936	1.8200	1.3450	1.0093	0.7748

ตารางที่ 4.13 แสดงผลการทดลองที่ได้จากการทำ WT ด้วยเวฟเล็ตแม่ Symmlets ของภาพต้นแบบ LADY

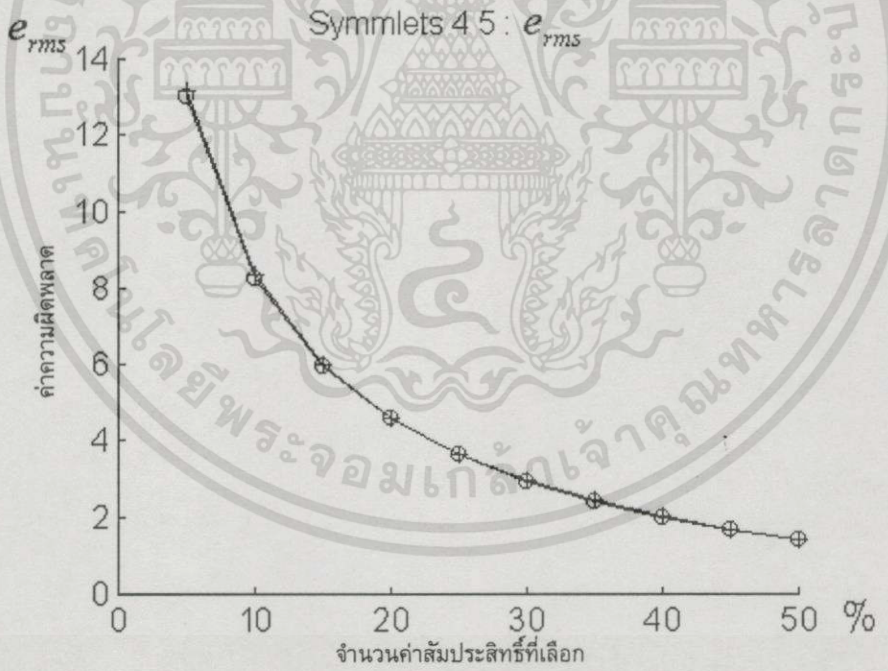
Wavelet	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Symmlet 4	9.9778	5.2569	3.2509	2.2132	1.5966	1.1932	0.9261	0.7406	0.6034
Symmlet 6	9.7379	5.2179	3.2515	2.2035	1.5855	1.1885	0.9225	0.7379	0.5998

ตารางที่ 4.14 แสดงผลการทดลองที่ได้จากการทำ WT ด้วยเวฟเล็ตแม่ Symmlets ของภาพต้นแบบ PLANT

Wavelet	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Symmlet 4	16.3465	12.5533	10.1860	8.4879	7.1539	6.0611	5.1492	4.3707	3.6935
Symmlet 6	16.3112	12.5223	10.1654	8.4702	7.1464	6.0708	5.1619	4.3842	3.7080

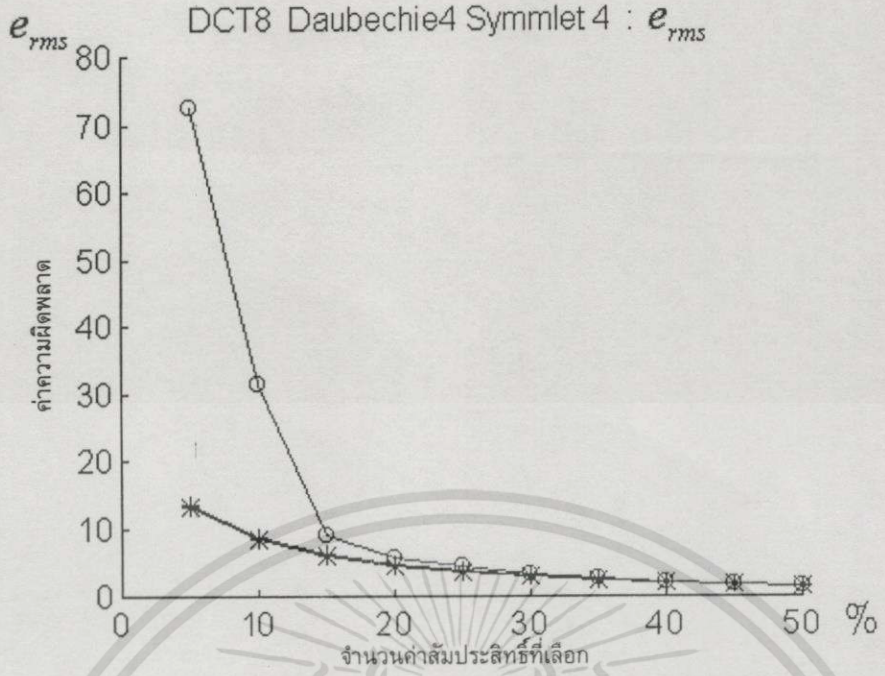
ตารางที่ 4.15 แสดงผลการทดลองที่ได้เฉลี่ยจากการทำ WT ด้วยเวฟเล็ตแม่ Symmlets

Wavelet	จำนวนค่าสัมประสิทธิ์ ที่เลือก และ ค่า Error ที่เกิดขึ้น (e_{rms})								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Symmlet 4	13.0310	8.2612	5.9726	4.5849	3.6425	2.9505	2.4251	2.0084	1.6688
Symmlet 6	13.1392	8.3572	6.0310	4.6174	3.6587	2.9617	2.4306	2.0125	1.6728



รูปที่ 4.29 แสดงกราฟผลการทดลองเฉลี่ยที่ได้ Symmlet 4 (o) และ Symmlet 5 (+)

เมื่อนำผลการทดลองมาเปรียบเทียบกัน เพื่อหาวิธีการที่เหมาะสม สามารถสรุปได้ว่า ขบวนการแปลงเวฟเล็ต จะมีประสิทธิภาพที่ดีกว่า ขบวนการแปลงด้วย DCT 8 ที่ใช้ในขบวนการลดขนาดข้อมูลแบบ JPEG ดังแสดงได้จากกราฟ จะเห็นได้ว่าที่จำนวนของค่าสัมประสิทธิ์ ที่น้อยลงนั้น ขบวนการแปลงแบบ DCT 8 จะมีค่าความผิดพลาดที่เพิ่มขึ้นอย่างรวดเร็ว



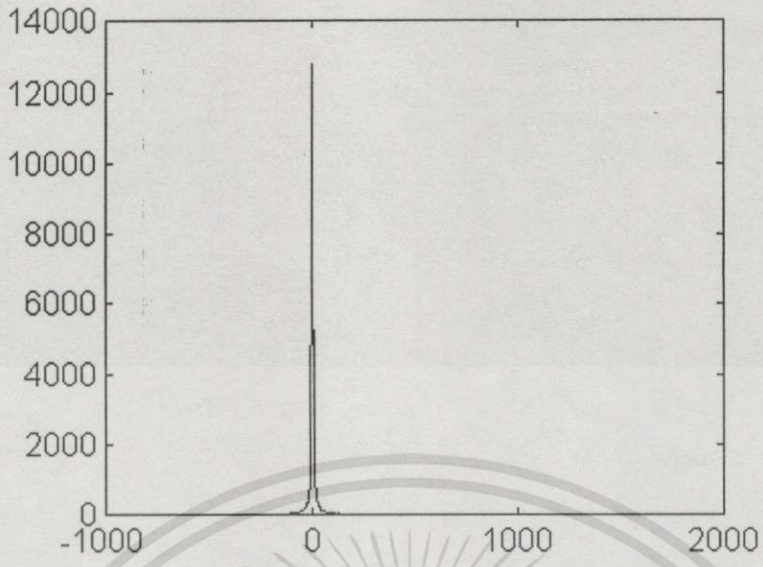
รูปที่ 4.30 แสดงผลการทดลองเปรียบเทียบกันระหว่าง ขบวนการแปลงด้วย DCT 8 (o) Daubechie 4 (x) และ Symmlet 4 (*)

จากผลการทดลองที่ได้นี้ จะสรุปได้ว่า การลดขนาดข้อมูล โดยใช้การแปลงเวฟเล็ต จะให้ประสิทธิ์ภาพที่ดีกว่าการแปลงแบบ DCT ที่จำนวนของค่าสัมประสิทธิ์ ที่ใช้น้อยกว่า 30% และ ผลการทดลองที่ได้ไม่มีความแตกต่างกันมาก ระหว่างการใช้เวฟเล็ตแม่ที่ต่างกัน ดังนั้น ในการใช้งานจึงจะเลือกใช้ เวฟเล็ตแม่แบบ Daubechie 4 ที่จะมีการคำนวณน้อยกว่า เนื่องจากจำนวนจุดของข้อมูลที่น้อยกว่านั้นเอง

4.2.6 คุณสมบัติของค่าสัมประสิทธิ์เวฟเล็ตของข้อมูลภาพ

ข้อมูลของค่าสัมประสิทธิ์ ที่ได้จากขบวนการ แปลงเวฟเล็ตนี้ จะมีจำนวนที่มากกว่า ข้อมูลของภาพต้นแบบ เนื่องจากในขบวนการแปลงเวฟเล็ต ข้อมูลสัมประสิทธิ์ที่ได้ ซึ่งจะมีจำนวนของจุดข้อมูลเท่ากับภาพต้นแบบ แต่จะเป็นเลขจำนวนจริงแบบมีทศนิยม (Floating-Point) ซึ่งใช้จำนวนของหน่วยความจำที่มากกว่า ข้อมูลของภาพต้นแบบที่มีระดับความสว่างเป็นเลขจำนวนเต็ม 256 ระดับ

คุณสมบัติที่สำคัญอีกประการหนึ่งคือ ลักษณะการกระจายของค่าสัมประสิทธิ์ (Histogram) ซึ่งจากข้อมูลของค่าสัมประสิทธิ์ จะมีลักษณะการกระจายดังแสดงในรูปที่ 4.31

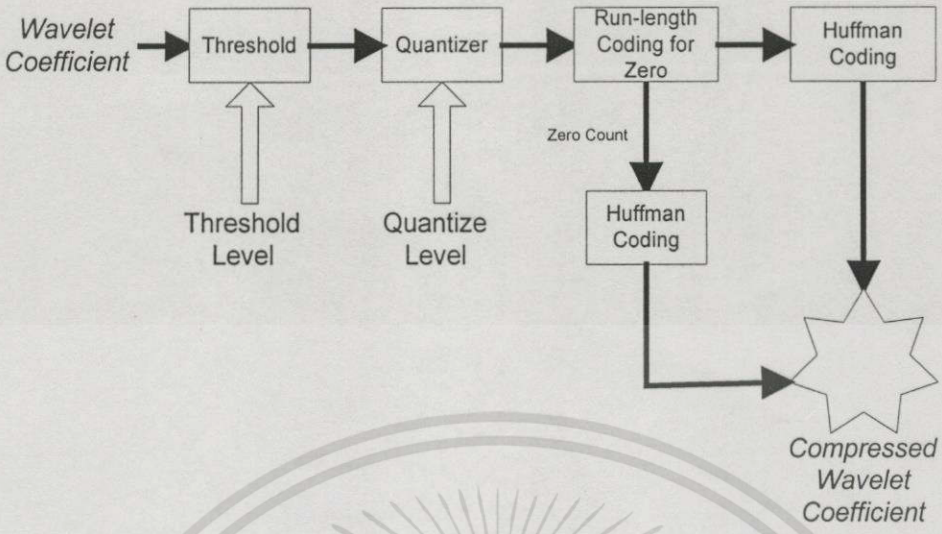


รูปที่ 4.31 แสดง Histogram ของค่าสัมประสิทธิ์เวฟเล็ต ที่ได้จากการใช้เวฟเล็ตแม่ Daubechie 4 ของภาพต้นแบบที่ 1 ที่มี 256x256 จุดภาพ

จากรูปข้างต้นจะเห็นได้ว่าค่าสัมประสิทธิ์ที่ได้ มีค่าอยู่ใกล้ศูนย์เป็นจำนวนมาก ซึ่งจากลักษณะดังกล่าวนี้ จะทำให้ได้ขบวนการลดขนาดข้อมูลดัง ที่จะทดลองในหัวข้อต่อไป

4.3 การลดขนาดข้อมูลสัมประสิทธิ์เวฟเล็ต

จากคุณสมบัติของค่าสัมประสิทธิ์ที่ได้ การที่จะลดจำนวนของข้อมูลลงได้นั้น จะสามารถทำได้โดย ลดจำนวนระดับที่ต้องใช้ของค่าสัมประสิทธิ์ลง จากนั้นจึงใช้ ขบวนการเข้ารหัสข้อมูล ในการลดจำนวนของข้อมูล ขบวนการที่ใช้สำหรับการลดจำนวนระดับของข้อมูลลงนั้น จะประกอบด้วยการทำ Threshold เพื่อตัดรายละเอียดที่ไม่จำเป็น และการ Quantization ข้อมูลที่ได้หลังจากการ Quantize แล้ว จะพบว่าส่วนใหญ่แล้วจะมีค่าเท่ากับ '0' ดังนั้น จึงจะใช้ ขบวนการ Run-length Coding ในการลดจำนวนของข้อมูล '0' ลง และใช้การเข้ารหัสข้อมูลทางสถิติของ Huffman Coding เพื่อสร้างรหัสที่เหมาะสมและสั้นที่สุด จากคุณสมบัติการกระจายของข้อมูลสัมประสิทธิ์ ขบวนการทั้งหมดจะสามารถแสดงเป็นแผนภาพได้ดังรูปที่ 4.32



รูปที่ 4.32 แสดงขบวนการลดขนาดของค่าสัมประสิทธิ์เวฟเล็ต

จากรูปที่ 4.32 จะสามารถอธิบายถึงส่วนประกอบต่างๆ ได้ดังต่อไปนี้

ส่วน Threshold จะทำการตัดข้อมูลที่มีขนาดต่ำกว่าค่าที่ผู้ใช้ตั้ง (Threshold Level) ทั้งนี้เพื่อเป็นการลดรายละเอียดของภาพลง และจะทำให้ข้อมูลของ ค่าสัมประสิทธิ์ที่ได้ มีจำนวนของข้อมูลที่เป็น ศูนย์ เพิ่มมากขึ้น ทั้งนี้จากที่การแปลง นี้ใช้เวฟเล็ตแม่ ที่มีค่าพลังงานเท่ากับหนึ่ง ค่าผลรวมของสัมประสิทธิ์ที่ถูดตัดออกไปจะเท่ากับค่าความผิดพลาดที่จะเกิดขึ้นกับข้อมูลภาพ

ส่วน Quantizer จะทำหน้าที่ ลดจำนวนระดับของข้อมูลลง โดยจะทำการแบ่ง ค่าของข้อมูลออกเป็นช่วงๆ ตามที่ผู้ใช้กำหนด (Quantize Level) เพื่อให้รหัสข้อมูลที่ต้องใช้น้อยลง ในส่วนของ Quantizer นี้ จะได้ออกแบบไว้สองวิธีด้วยกัน คือ เป็นแบบ Uniform และ Non Uniform

ส่วน Run-Length Coding จะทำการแยกข้อมูลของค่า ศูนย์ ที่มีอยู่ติดกันออกไป โดยที่ข้อมูลของค่า ศูนย์ ที่มีอยู่ติดต่อกันนี้ จะถูกแทนด้วยข้อมูลของ ศูนย์ เพียงตัวเดียว จากนั้นข้อมูลที่ถูกลดจำนวน ศูนย์ แล้ว จะถูกนำไปเข้ารหัสต่อไป ส่วนค่า จำนวนของข้อมูล ศูนย์ ที่ติดกัน (Zero Count) ก็ต้องถูกนำมาเข้ารหัสด้วยเช่นกัน

ส่วน Huffman Coding จะมีหน้าที่ในการหารหัสที่สั้นที่สุด

การทดลองจะแบ่งเป็นหัวข้อการทดลองต่างๆ ดังนี้

1. การทำ Threshold ของค่าสัมประสิทธิ์ ซึ่งสามารถทำได้โดยการตัดข้อมูลที่มีขนาดต่ำกว่าระดับที่ตั้งไว้ออกไป

2. การ Quantization ค่าของสัมประสิทธิ์ โดยการแบ่งข้อมูลของค่าสัมประสิทธิ์ อกเป็นช่วงๆ ละเท่าๆ กัน แล้วใช้ค่ากลางของช่วงนั้นเป็นตัวแทนของข้อมูล
3. การ Quantization ตามค่าของสัมประสิทธิ์ ที่ปรับปรุง โดยการแบ่งข้อมูลของค่าสัมประสิทธิ์ อกเป็นช่วงๆ ละเท่าๆ กัน แล้วใช้ค่าเฉลี่ยของข้อมูลที่มีในช่วงนั้นเป็นตัวแทนของข้อมูล

โดยในการทดลองในแต่ละหัวข้อ จะเป็นการหาความสัมพันธ์ ระหว่าง อัตราการลดขนาดข้อมูลที่ได้ กับ ค่าความผิดพลาดที่เกิดขึ้น ผลการทดลองที่ได้ต่อไปนี้ จะทำโดยใช้โปรแกรม ภาษา Delphi® ที่ได้พัฒนาขึ้น เนื่องจาก เป็นขั้นตอนที่ใช้การคำนวณมาก การทำการทดลอง ด้วย MatLab® จะใช้เวลาที่นาน

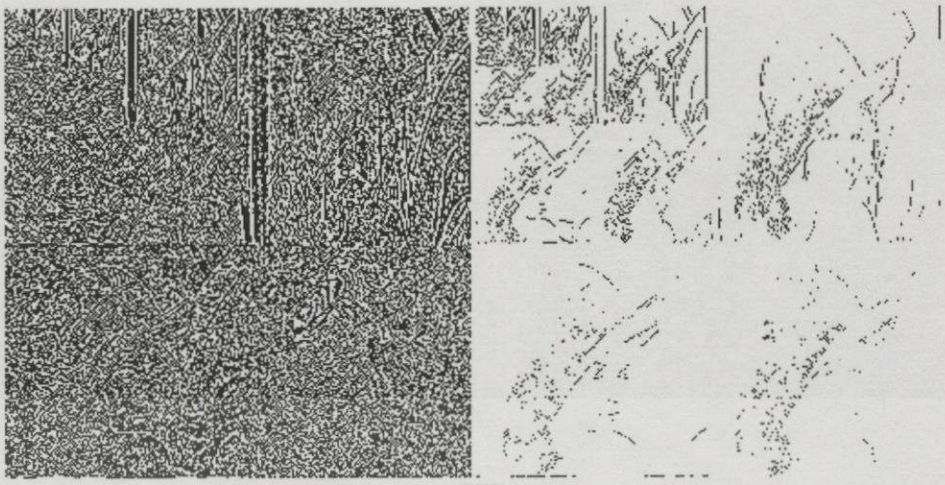
สำหรับวิธีที่ใช้ในการทดลองจะเป็นดังต่อไปนี้

1. ข้อมูลภาพต้นแบบที่ใช้ทั้งหมด จะเป็นภาพ ระดับสีเทา 256 ระดับ (8 bpp) ขนาด 256 x 256 จุดภาพ
2. นำข้อมูลภาพต้นแบบ ผ่านขบวนการลดขนาดข้อมูล โดยกำหนด ระดับการ Threshold และ Quantize ต่างๆ กัน
3. วัดอัตราการลดขนาดข้อมูล โดยพิจารณาจำนวนข้อมูลที่ได้ ต่อจำนวนจุดภาพทั้งหมด ได้ออกมาในหน่วยของ เท่า
4. นำข้อมูลที่ถูกลดขนาดแล้วมาทำการขยาย เพื่อให้ได้กลับมาเป็นข้อมูลภาพ แล้ววัดค่าความผิดพลาดของข้อมูลภาพ ที่ได้นี้ เปรียบเทียบกับภาพต้นแบบ

จากที่การวัดผลการทดลองในส่วนนี้ ค่าความผิดพลาดจะได้จากการเปรียบเทียบ จากข้อมูลภาพที่ได้ กับภาพต้นแบบ และสำหรับค่าอัตราการลดขนาดของข้อมูล จะได้จากการทำ Run-length Coding และ Huffman Coding จากขบวนการดังต่อไปนี้

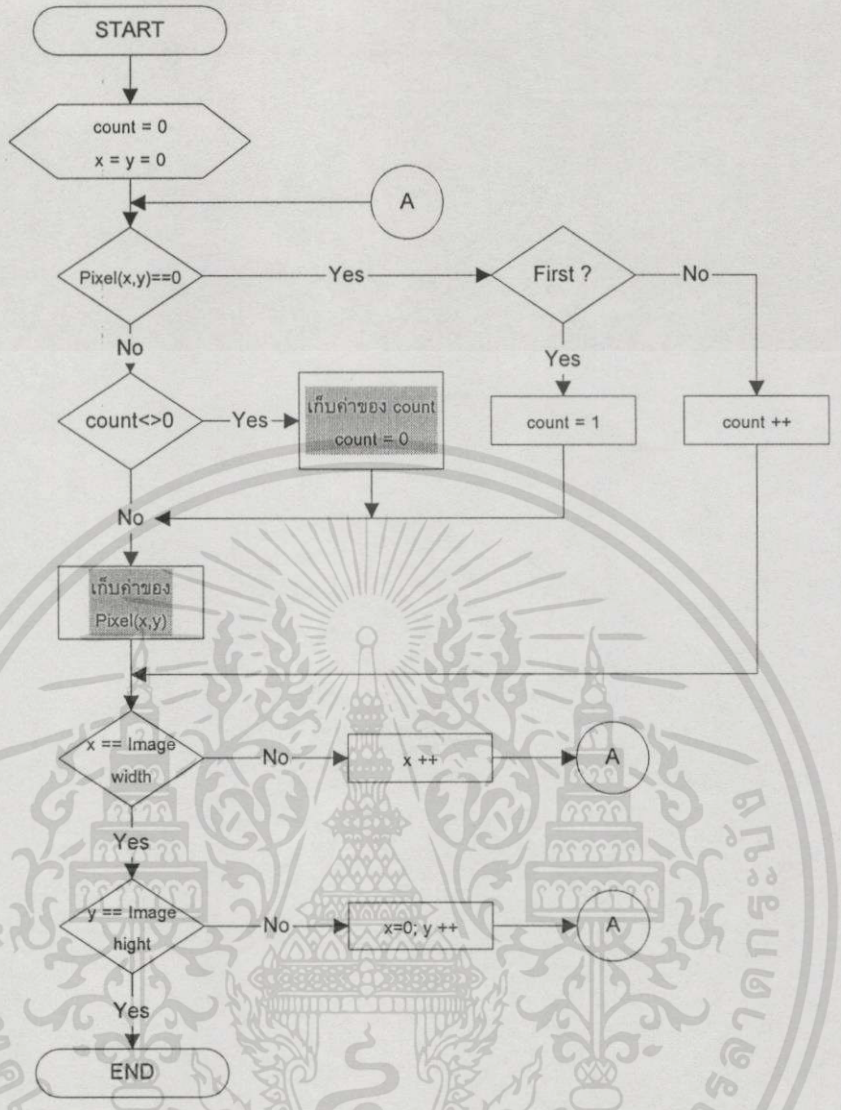
4.3.1 การทำ Run-Length Coding และHuffman Coding

จากค่าสัมประสิทธิ์ที่ได้จากการทดลอง ซึ่งประกอบด้วยข้อมูลของ '0' เป็นจำนวนมากนี้ จะทำให้ โอกาสที่ข้อมูล '0' จะอยู่ติดกันเป็นจำนวนมากมีอยู่สูง จึงต้องใช้วิธีการ ของ Run-Length Coding ในการลดจำนวนของข้อมูล '0' ที่ซ้ำๆ กันลง รูปที่ 4.33 จะแสดงตัวอย่างของ ค่าสัมประสิทธิ์เวฟเล็ตที่ได้จาก ภาพต้นแบบที่ 1 โดยจะแสดงตำแหน่งที่ค่าของสัมประสิทธิ์ ไม่เป็นศูนย์ ก่อน และหลังจากการทำ Threshold = 10 แล้ว



รูปที่ 4.33 แสดงตำแหน่งที่ค่าสัมประสิทธิ์ ไม่เท่ากับ ศูนย์ (ซ้าย) ก่อนการ Threshold (ขวา) หลังจากทำ Threshold = 10 แล้ว ของภาพต้นแบบที่ 1

จากหลักการของขบวนการ Run-Length Coding ซึ่งจะเก็บข้อมูล ที่ซ้ำๆ กันนี้ ในรูปแบบของข้อมูล และจำนวนที่มีซ้ำกัน จากตัวอย่างข้างต้น จะเลือกทำ Run-Length Coding เฉพาะข้อมูลของ '0' เท่านั้น โดยโปรแกรมที่ใช้มีการทำงานดังแผนภาพในรูปที่ 4.34



รูปที่ 4.34 แสดงแผนภาพการทำงานของ ขบวนการ Run-Length Coding ที่ใช้

จากขบวนการข้างต้นผลลัพท์ที่ได้จากขบวนการจะประกอบ ด้วยกันสองส่วนคือ ส่วนที่เป็นข้อมูลที่ ลดจำนวนของ '0' แล้ว และจำนวนของข้อมูล '0' ที่มีติดต่อกันในแต่ละช่วง ขบวนการนี้จะแทนข้อมูล ของ '0' ที่อยู่ติดต่อกันด้วย '0' เพียงตัวเดียวเพื่อให้ทราบว่า ณ ตำแหน่งนี้มีข้อมูลของ '0' อยู่ ข้อมูลที่ได้ ทั้งสองส่วนนี้ จะถูกนำไปเข้าขบวนการ Huffman Coding ต่อไป ขบวนการของ Huffman Coding จะให้เอาท์พุทออกมาสองส่วนคือ ข้อมูลที่แทนด้วยรหัส Huffman และส่วนของ Header หรือข้อมูลที่จะใช้สำหรับการถอดรหัส ส่วนของ Header จะต้องประกอบด้วยข้อมูลดังตารางที่ 4.16

ตารางที่ 4.16 แสดง Header ที่จำเป็นในการถอดรหัส Huffman และการเก็บข้อมูล

ข้อมูล	จำนวน บิท ที่ ใช้
จำนวนแบบของ Code	12 บิท
ข้อมูลที่ ถูกแทนด้วย Code	16 บิท x จำนวนแบบของ Code ที่มี
ความยาวของ Code	4 บิท x จำนวนแบบของ Code ที่มี
Huffman Code	1-16 บิท x จำนวน Code ที่มี (ขึ้นอยู่กับ Huffman Code)

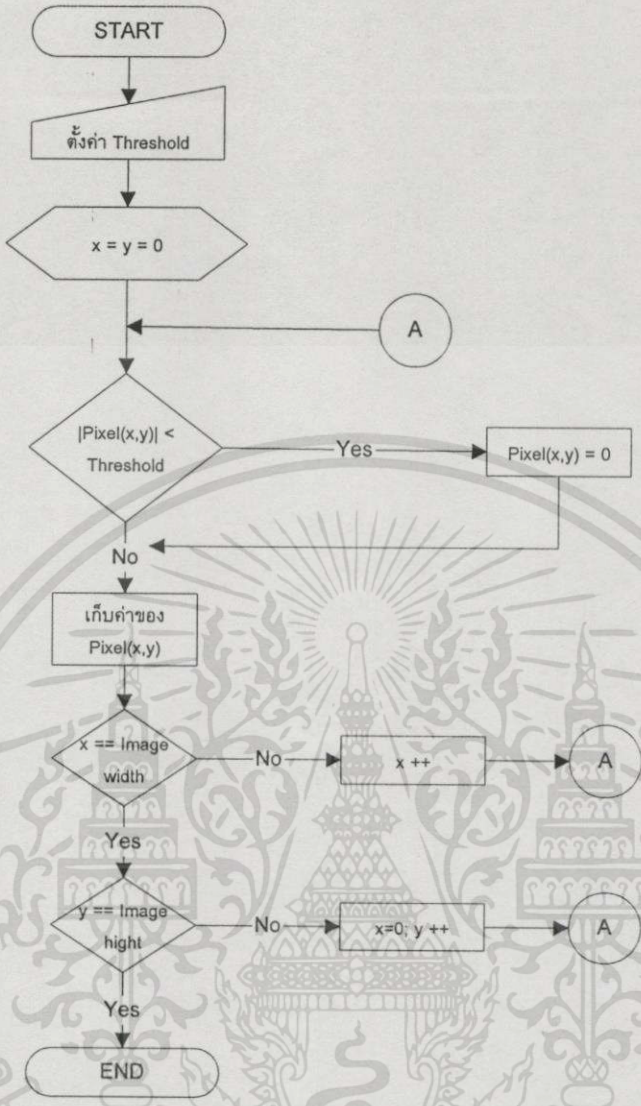
และการเก็บข้อมูลทั้งหมดของค่าสัมประสิทธิ์เวฟเล็ต จึงจะเป็นดังนี้

ตารางที่ 4.17 แสดงการเก็บข้อมูลทั้งหมดของสัมประสิทธิ์เวฟเล็ต

ข้อมูล	จำนวน บิท ที่ ใช้
ขนาดความกว้างของภาพ	12 บิท
ขนาดความสูงของภาพ	12 บิท
Header ของข้อมูล จำนวนของ '0'	ขึ้นอยู่กับข้อมูลที่ได้
ข้อมูล จำนวนของ '0' ที่แทนด้วย Huffman Code แล้ว	ขึ้นอยู่กับข้อมูลที่ได้
Header ของข้อมูล ค่าสัมประสิทธิ์ที่ลดจำนวนของ '0'	ขึ้นอยู่กับข้อมูลที่ได้
ข้อมูล ค่าสัมประสิทธิ์ที่ลดจำนวนของ '0' ที่แทนด้วย Huffman Code แล้ว	ขึ้นอยู่กับข้อมูลที่ได้

4.3.2 การทำ Threshold ของค่าสัมประสิทธิ์

การทำ Threshold ของค่าสัมประสิทธิ์ จะทำโดยใช้ขบวนการตามแผนภาพในรูปที่ 4.35



รูปที่ 4.35 แสดงแผนภาพขบวนการทำ Threshold ของค่าสัมประสิทธิ์

การทดลองจะกระทำกับภาพต้นแบบทั้ง 4 ภาพ ที่ระดับของการ Threshold ค่าต่างๆ (Level) เพื่อหาความสัมพันธ์ ระหว่างค่าความผิดพลาดที่เกิดขึ้น (e_{rms}) กับ การลดขนาดข้อมูลที่ได้ (Compression Ratio) ผลการทดลองที่ได้เป็นไปดังตารางที่ 4.18, 4.19, 4.20 และ 4.21 ตามลำดับ ส่วนในรูปที่ 4.36 จะแสดงกราฟผลการทดลองที่ได้

ตารางที่ 4.18 ผลการทดลองจากการทำ Threshold ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 1 (LENA)

Level	*5*	*10*	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*
e_{rms}	1.75	3.34	4.49	5.51	6.47	7.36	8.17	8.9	9.61	10.27
SNR_{rms}	4008.64	1103.36	608.06	404.38	293.17	225.9	183.3	154.22	132.2	115.59
Bpp	3.83043	2.53323	1.99826	1.67523	1.44757	1.27237	1.13495	1.02748	0.93516	0.86346
Ratio	2.08854	3.15802	4.00348	4.77546	5.5265	6.28748	7.04877	7.78603	8.55464	9.26500

ตารางที่ 4.19 ผลการทดลองจากการทำ Threshold ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 2 (DOG)

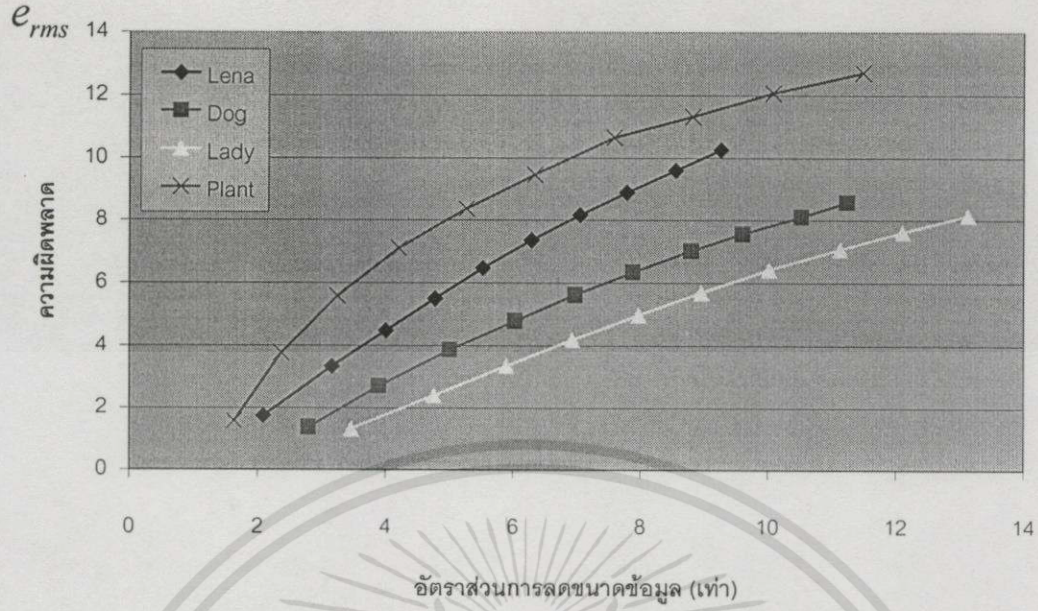
Level	*5*	*10*	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*
e_{rms}	1.37	2.71	3.87	4.8	5.63	6.34	7.02	7.56	8.11	8.59
SNR_{rms}	17193.6	4389.68	2152.57	1395.97	1016.54	801.15	652.68	563.56	488.91	436.09
Bpp	2.86606	2.05466	1.59825	1.32640	1.14688	1.01436	0.90886	0.83252	0.76003	0.71173
Ratio	2.79129	3.89359	5.00547	6.03136	6.97544	7.88676	8.80224	9.60938	10.5260	11.2402

ตารางที่ 4.20 ผลการทดลองจากการทำ Threshold ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 3 (LADY)

Level	*5*	*10*	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*
e_{rms}	1.33	2.41	3.35	4.19	4.98	5.7	6.42	7.07	7.61	8.15
SNR_{rms}	7066.61	2141.49	1112.27	708.39	501.02	382.25	301.42	248.67	214.43	186.6
Bpp	2.30989	1.68044	1.35628	1.15485	1.00357	0.89273	0.79837	0.71875	0.66092	0.60951
Ratio	3.46337	4.76067	5.89850	6.92733	7.97154	8.96127	10.0204	11.1304	12.1044	13.1253

ตารางที่ 4.21 ผลการทดลองจากการทำ Threshold ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 4 (PLANT)

Level	*5*	*10*	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*
e_{rms}	1.58	3.77	5.59	7.1	8.38	9.48	10.66	11.32	12.07	12.72
SNR_{rms}	2905.97	506.22	229.58	142.39	101.69	79.36	62.56	55.29	48.58	43.59
Bpp	4.89821	3.38426	2.46918	1.90212	1.51894	1.26227	1.05446	0.90689	0.79347	0.69653
Ratio	1.63325	2.36388	3.23995	4.20585	5.26684	6.33780	7.58683	8.82135	10.0823	11.4855

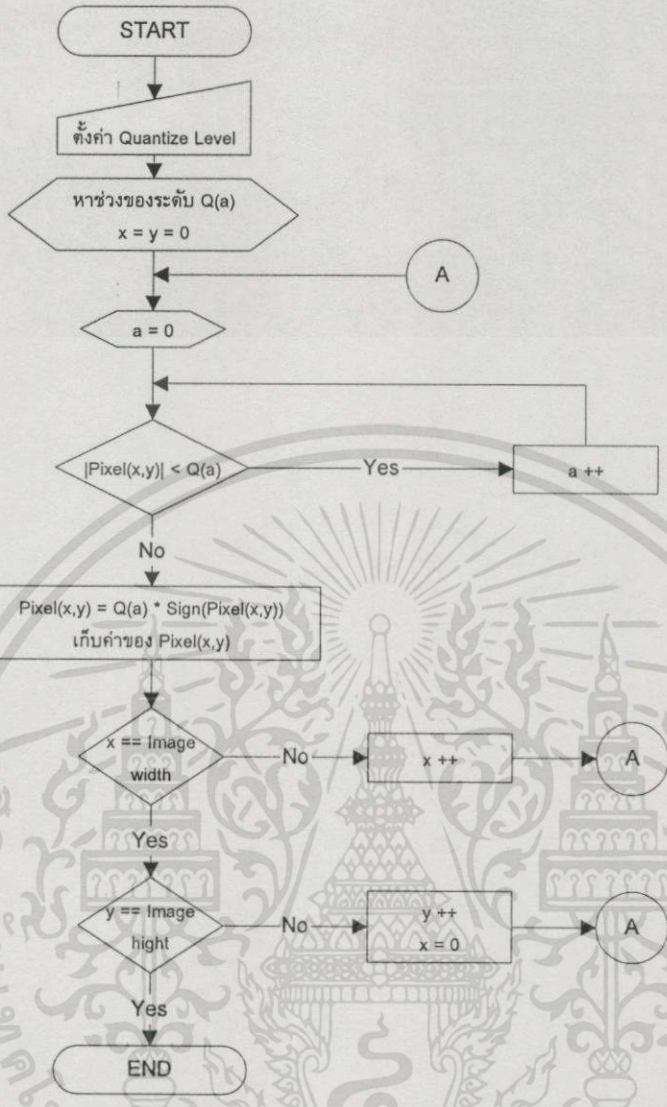


รูปที่ 4.36 แสดงความสัมพันธ์ ของอัตราส่วนการลดขนาดข้อมูลที่ได้ (Ratio) กับความผิดพลาด (e_{rms}) ที่ได้จากการทำ Threshold

จากผลการทดลองที่ได้นี้ จะเห็นได้ว่าการเพิ่มค่าของ Threshold Level จะทำให้ได้อัตราการลดขนาดข้อมูลที่มากขึ้น ในขณะเดียวกันค่าความผิดพลาดจะเพิ่มขึ้นด้วย จากภาพต้นแบบทั้ง 4 ภาพ ซึ่งจะได้อัตราการลดขนาดข้อมูลที่ไม่เท่ากัน ขึ้นอยู่กับรายละเอียดที่มีในภาพต้นแบบ

4.3.3 การ Quantization ค่าของสัมประสิทธิ์

จากข้อมูลของค่าสัมประสิทธิ์ที่ได้ ซึ่งจะมีจำนวนของระดับข้อมูลที่มาก ซึ่งจะต้องใช้รหัสข้อมูลมาก การ Quantization จะทำให้จำนวนระดับของข้อมูลลดลง โดยการแบ่งระดับข้อมูลออกเป็นช่วงๆ ละเท่าๆ กัน และใช้ค่ากลางของช่วงนั้น เป็นตัวแทนของระดับข้อมูล ซึ่งจะสามารถแสดงเป็นขั้นตอนตามแผนภาพดังรูปที่ 4.37



รูปที่ 4.37 แสดงแผนภาพขบวนการ Quantization ของค่าสัมประสิทธิ์

ในการทดลอง จะเป็นการหาความสัมพันธ์ระหว่าง อัตราการลดขนาดข้อมูลที่ได้ต่อ ค่าความผิดพลาดที่เกิดขึ้น เมื่อทำการทดลองกับภาพต้นแบบทั้ง 4 ภาพแล้ว ได้ผลการทดลองดังตารางที่ 4.22, 4.23, 4.24 และ 4.25 ตามลำดับ ส่วนในรูปที่ 4.38 จะแสดงกราฟผลการทดลองที่ได้

ตารางที่ 4.22 ผลการทดลองจากการ Quantization ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 1 (LENA)

Quantize	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*
e_{rms}	4.01	6.43	8.36	9.97	11.38	12.5	13.56	14.49
SNR_{rms}	758.64	293.66	172.62	120.7	92.34	76.3	64.44	56.35
bpp	1.59293	0.93614	0.66513	0.50763	0.40718	0.34079	0.28769	0.25102
Ratio	5.02220	8.54571	12.0277	15.7595	19.6473	23.4749	27.8078	31.8697

ตารางที่ 4.23 ผลการทดลองจากการ Quantization ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 2 (DOG)

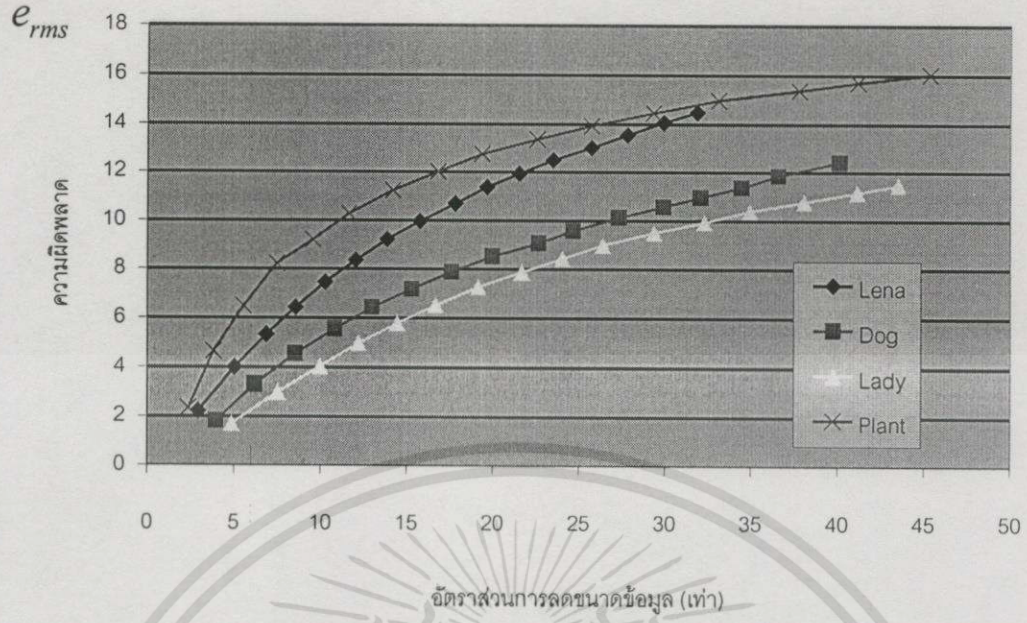
Quantize	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*
e_{rms}	3.32	5.56	7.18	8.53	9.58	10.58	11.38	12.43
SNR_{rms}	2923.63	1039.37	621.05	439.39	348.25	284.48	246.13	205.87
bpp	1.29432	0.74020	0.52353	0.40065	0.32465	0.26732	0.23210	0.19936
Ratio	6.18082	10.8078	15.2809	19.9676	24.6422	29.9268	34.4677	40.1292

ตารางที่ 4.24 ผลการทดลองจากการ Quantization ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 3 (LADY)

Quantize	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*
e_{rms}	2.99	4.97	6.52	7.88	9	9.94	10.79	11.48
SNR_{rms}	1391.34	500.66	290.05	197.88	150.97	123.8	104.59	92.11
bpp	1.06734	0.65677	0.48010	0.36928	0.30333	0.24762	0.21005	0.18388
Ratio	7.49529	12.1809	16.6631	21.6639	26.3740	32.3076	38.0857	43.5058

ตารางที่ 4.25 ผลการทดลองจากการ Quantization ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 4 (PLANT)

Quantize	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*
e_{rms}	4.66	8.22	10.28	12.02	13.38	14.46	15.38	16.04
SNR_{rms}	328.08	104.29	66.3	48.1	38.59	32.89	29.03	26.48
Bpp	2.10474	1.07561	0.68846	0.47600	0.35477	0.27284	0.21182	0.17650
Ratio	3.80095	7.43766	11.6201	16.8068	22.5500	29.3210	37.7675	45.3262



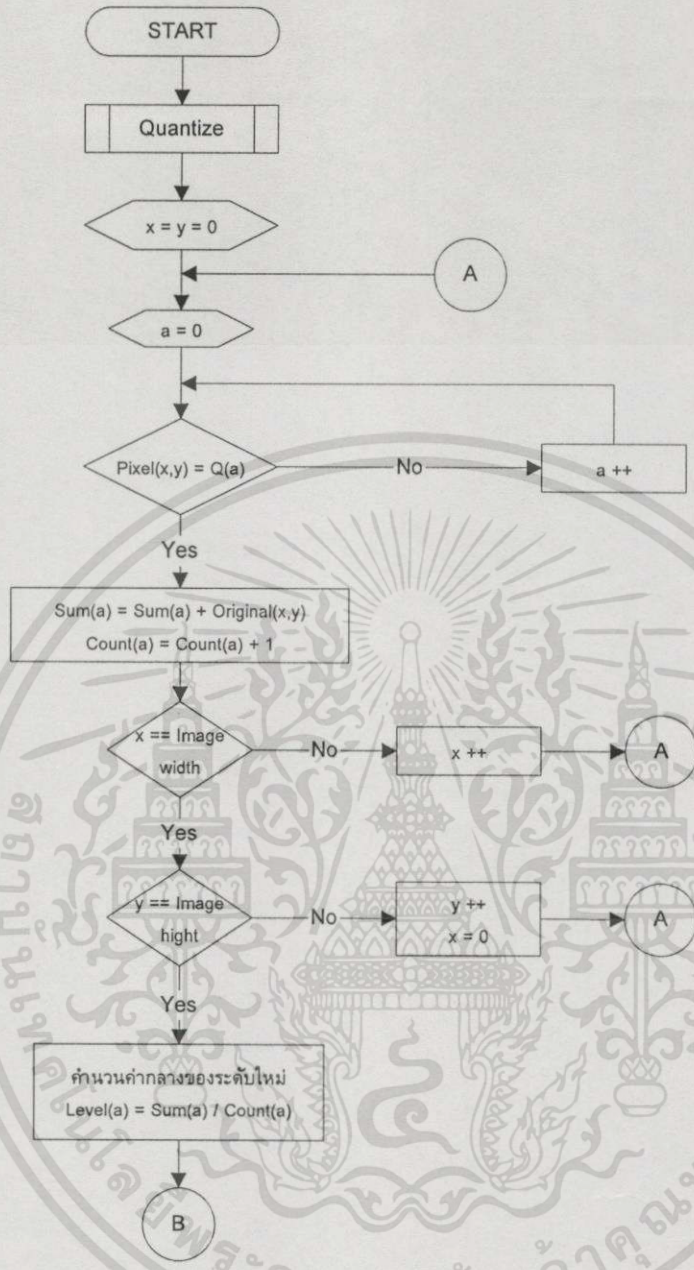
รูปที่ 4.38 แสดงความสัมพันธ์ ของอัตราส่วนการลดขนาดข้อมูลที่ได้ (Ratio) กับความผิดพลาด (e_{rms}) ที่ได้จากการ Quantization

ในการทำ Quantization ค่าสัมประสิทธิ์ที่มีขนาดน้อยกว่า ระดับการ Quantize ระดับแรกจะถูกปัดให้เป็น '0' เช่นเดียวกับการทำ Threshold และจากผลการทดลองที่ได้ ค่าอัตราการลดขนาดข้อมูลจะขึ้นอยู่กับลักษณะของข้อมูลภาพเช่นเดียวกับการทำ Threshold ในการทดลองที่แล้ว และเมื่อเปรียบเทียบ ผลการทดลองที่ได้จากการทำ Quantization ค่าสัมประสิทธิ์ กับผลที่ได้จากขบวนการทำ Threshold ที่ค่าความผิดพลาดที่เท่ากัน ค่าอัตราการลดขนาดข้อมูลที่ได้จะมากกว่า

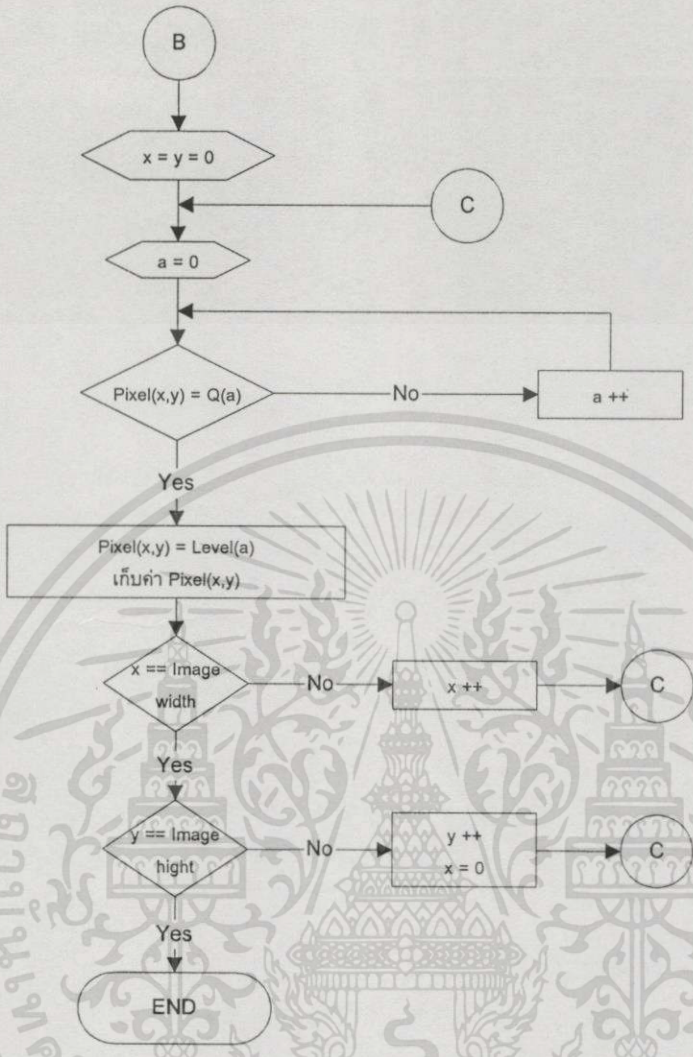
4.3.4 การ Quantization ค่าของสัมประสิทธิ์ ที่ปรับปรุง

จากการ Quantization ตามขบวนการในการทดลองที่ผ่านมานั้น ที่ระดับของการ Quantize ที่ห่างกันมากๆ นั้น การใช้ค่ากลางของระดับ เพื่อเป็นตัวแทนของข้อมูล ในช่วงระดับนั้น อาจไม่เป็นค่าที่เหมาะสม เนื่องจาก ในบางครั้งอาจมีข้อมูลเพียงจุดเดียวที่อยู่ในช่วงนี้ ดังนั้นการใช้ค่าเฉลี่ยของข้อมูลที่อยู่ในช่วง จะเป็นวิธีการที่จะให้ค่าความผิดพลาดที่น้อยกว่า ดังนั้นจึงได้เสนอขบวนการ Quantization แบบใหม่ โดยมีขั้นตอนการทำงานแสดงได้ตามแผนภาพในรูปที่ 4.39

และได้ทำการทดลอง การ Quantization แบบใหม่นี้ กับภาพทดสอบทั้ง 4 ซึ่งได้ผลการทดลอง ตามตารางที่ 4.26 – 4.29 ตามลำดับ และในรูปที่ 4.40 เป็นกราฟผลการทดลองที่ได้



รูปที่ 4.39 แสดงแผนภาพวิธีการปรับปรุงขบวนการ Quantization



รูปที่ 4.39 (ต่อ) แสดงแผนภาพวิธีการปรับปรุงขบวนการ Quantization

ตารางที่ 4.26 ผลการทดลองจากการทำ Quantization ที่ปรับปรุง ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 1 (LENA)

Quantize	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*
e_{rms}	3.6	5.83	7.71	9.27	10.65	11.75	12.81	13.75
SNR_{rms}	947.88	360.64	205.8	142.07	107.48	88.07	73.94	64.04
bpp	1.592926	0.936142	0.665131	0.507629	0.407181	0.34079	0.287689	0.251022
Ratio	5.022204	8.545712	12.02771	15.75953	19.64729	23.47488	27.80779	31.86967

ตารางที่ 4.27 ผลการทดลองจากการทำ Quantization ที่ปรับปรุง ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 2 (DOG)

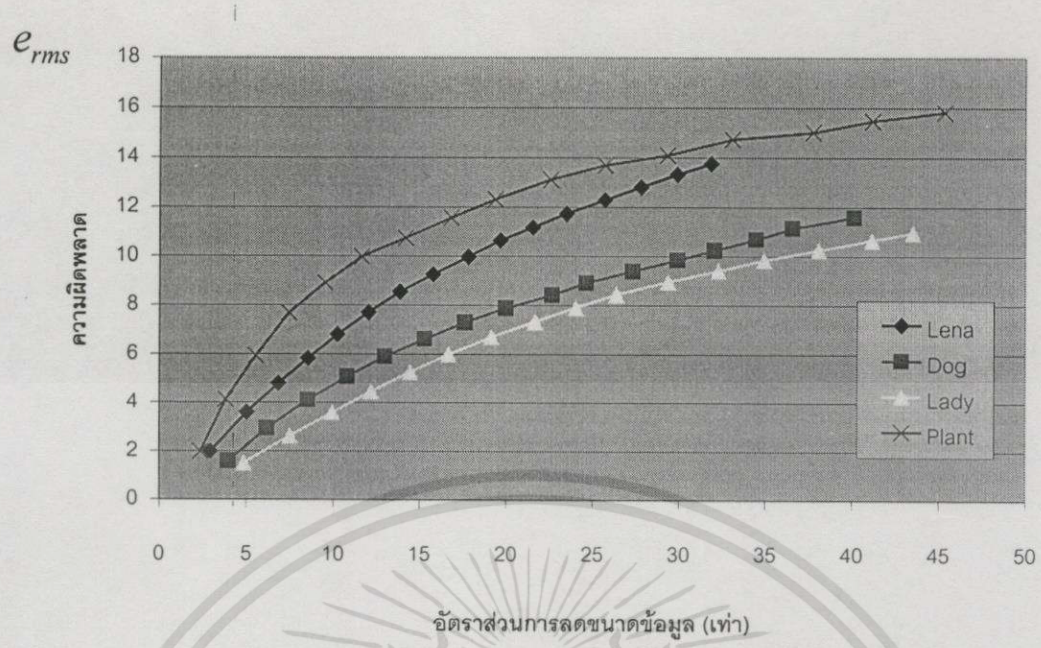
Quantize	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*
e_{rms}	2.95	5.07	6.63	7.88	8.92	9.85	10.69	11.61
SNR_{rms}	3709.58	1255.12	732.25	518.08	404.05	330.95	281.19	238.27
bpp	1.294327	0.740204	0.523529	0.40065	0.324646	0.267319	0.232101	0.199356
Ratio	6.180819	10.80783	15.28091	19.96755	24.64223	29.92682	34.46769	40.1292

ตารางที่ 4.28 ผลการทดลองจากการทำ Quantization ที่ปรับปรุง ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 3 (LADY)

Quantize	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*
e_{rms}	2.63	4.46	5.99	7.34	8.44	9.42	10.25	10.97
SNR_{rms}	1794.33	624.99	346.23	230.33	173.8	139.58	117.56	102.5
bpp	1.067337	0.656769	0.480103	0.369278	0.303329	0.24762	0.210052	0.183884
Ratio	7.495289	12.18085	16.66311	21.6639	26.37396	32.30762	38.08572	43.50577

ตารางที่ 4.29 ผลการทดลองจากการทำ Quantization ที่ปรับปรุง ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 4 (PLANT)

Quantize	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*
e_{rms}	4.13	7.69	9.98	11.58	13.11	14.09	15.02	15.83
SNR_{rms}	420.3	121.01	71.37	52.76	41	35.34	30.97	27.82
Bpp	2.104736	1.075607	0.688461	0.475998	0.354767	0.272842	0.211823	0.176498
Ratio	3.800951	7.437659	11.62012	16.8068	22.55002	29.32096	37.76747	45.32619



รูปที่ 4.40 แสดงความสัมพันธ์ ของอัตราส่วนการลดขนาดข้อมูลที่ได้ (Ratio) กับความผิดพลาด (e_{rms}) ที่ได้จากการ Quantization ที่ปรับปรุง

ผลการทดลองที่ได้จากการทำ Quantization ที่ปรับปรุง ค่าอัตราการลดขนาดข้อมูลจะขึ้นอยู่กับลักษณะของข้อมูลภาพเช่นเดียวกัน และจากกราฟที่แสดงถึงค่าความผิดพลาด จะมีพบว่ามีบางจุดข้อมูลที่มีค่าความผิดพลาดต่ำลง ทั้งนี้เนื่องจากการใช้ค่าเฉลี่ยของข้อมูลในช่วงนั้นเอง

จากผลการทดลองที่ได้นี้ เมื่อเปรียบเทียบวิธีการ Quantization ทั้งสองแล้วจะให้การลดขนาดข้อมูลที่ใกล้เคียงกัน แต่ในวิธีการ Quantization ที่ปรับปรุงนี้ จะให้ค่าความผิดพลาดของข้อมูลน้อยกว่า 5.86% เฉลี่ยจากผลการทดลองที่ได้ จากข้อมูลภาพทั้ง 4

จากผลการทดลองที่ได้นี้ เมื่อนำไปเปรียบเทียบกับขบวนการลดขนาดข้อมูลที่ใช้ในปัจจุบัน (JPEG สำหรับภาพ ขาว-ดำ) ซึ่งเป็นการลดขนาดข้อมูลภาพ โดยใช้การแปลงข้อมูลแบบ DCT8 ซึ่งผลการทดลองจะได้จากการใช้โปรแกรม Microsoft Photo Editor ® ทำการลดขนาดข้อมูลภาพด้วยวิธีการของ JPEG ที่ค่า Quality factor ต่างๆ แล้วทำการวัดค่าความผิดพลาด ที่เกิดขึ้นในภาพที่ได้ และค่าอัตราส่วนการลดขนาดข้อมูลจะสามารถวัดได้จากขนาดของ ไฟล์ภาพ (Size) ผลการทดลองที่ได้ เมื่อใช้ภาพต้นแบบทั้ง 4 เป็นดังแสดงในตารางที่ 4.30, 4.31, 4.32 และ 4.33 ตามลำดับ และในรูปที่ 4.41 แสดงกราฟผลการทดลองที่ได้

ตารางที่ 4.30 ผลการทดลองจากวิธีการ JPEG โดยใช้ภาพต้นแบบที่ 1 (LENA)

Quality	*90*	*80*	*70*	*60*	*50*	*40*	*30*	*20*	*10*	*5*
e_{rms}	2.87	4.22	5.12	5.8	6.34	6.93	7.65	8.78	11.04	14.39
SNR_{rms}	1494.36	692.09	469.92	366.02	305.75	256.39	210.05	159.42	100.86	58.83
Size	19913	13169	10367	8641	7547	6548	5509	4294	2864	1965
Bpp	2.43079	1.60754	1.26550	1.05481	0.92126	0.79932	0.67249	0.52417	0.34961	0.23987
Ratio	3.29112	4.97654	6.32160	7.58431	8.68372	10.00855	11.89617	15.26223	22.88268	33.35165

ตารางที่ 4.31 ผลการทดลองจากวิธีการ JPEG โดยใช้ภาพต้นแบบที่ 2 (DOG)

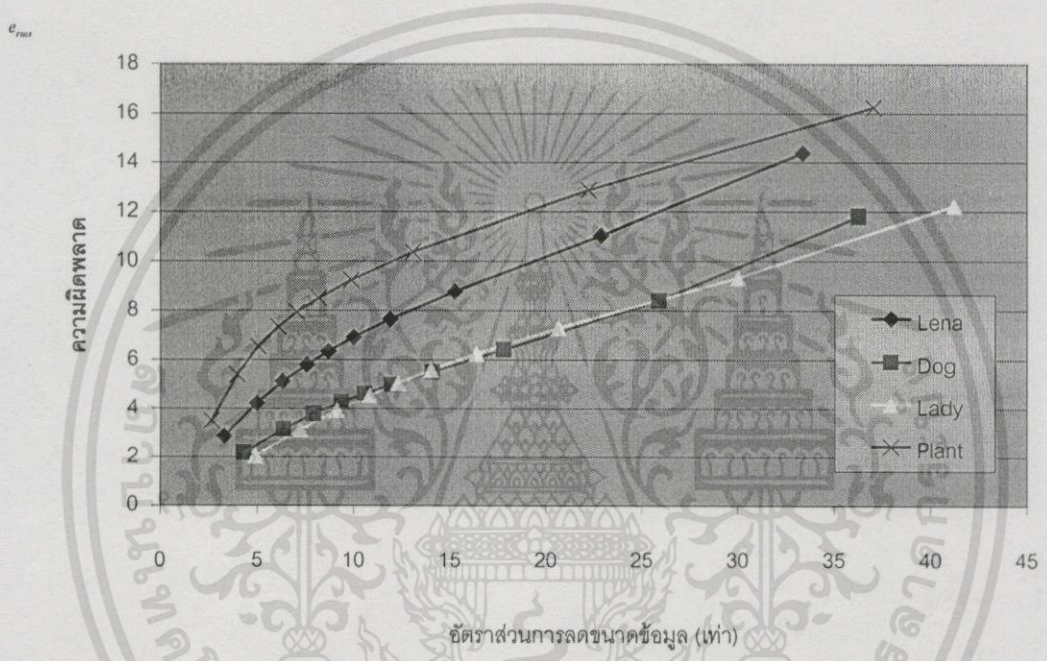
Quality	90	80	70	60	50	40	30	20	10	5
e_{rms}	2.17	3.16	3.79	4.26	4.61	4.98	5.52	6.42	8.41	11.8
SNR_{rms}	6856.98	3226.6	2245.52	1773.02	1514.66	1296.69	1056.11	779.65	454.55	229.52
Size	15176	10275	8275	6994	6210	5469	4651	3678	2531	1808
Bpp	1.85254	1.25427	1.01013	0.85376	0.75806	0.66760	0.56775	0.44897	0.30896	0.22070
Ratio	4.31840	6.37820	7.91976	9.37032	10.55330	11.98318	14.09073	17.81838	25.89332	36.24779

ตารางที่ 4.32 ผลการทดลองจากวิธีการ JPEG โดยใช้ภาพต้นแบบที่ 3 (LADY)

Quality	90	80	70	60	50	40	30	20	10	5
e_{rms}	2.05	3.13	3.92	4.53	5.05	5.59	6.25	7.29	9.33	12.23
SNR_{rms}	2968.21	1272.7	812.6	607.19	488.72	399.08	318.44	233.78	142.81	83.23
Size	13374	9051	7179	6038	5320	4672	3999	3173	2184	1591
Bpp	1.63257	1.10486	0.87634	0.73706	0.64941	0.57031	0.48816	0.38733	0.26660	0.19421
Ratio	4.90025	7.24075	9.12885	10.85393	12.31880	14.02740	16.38810	20.65427	30.00733	41.19170

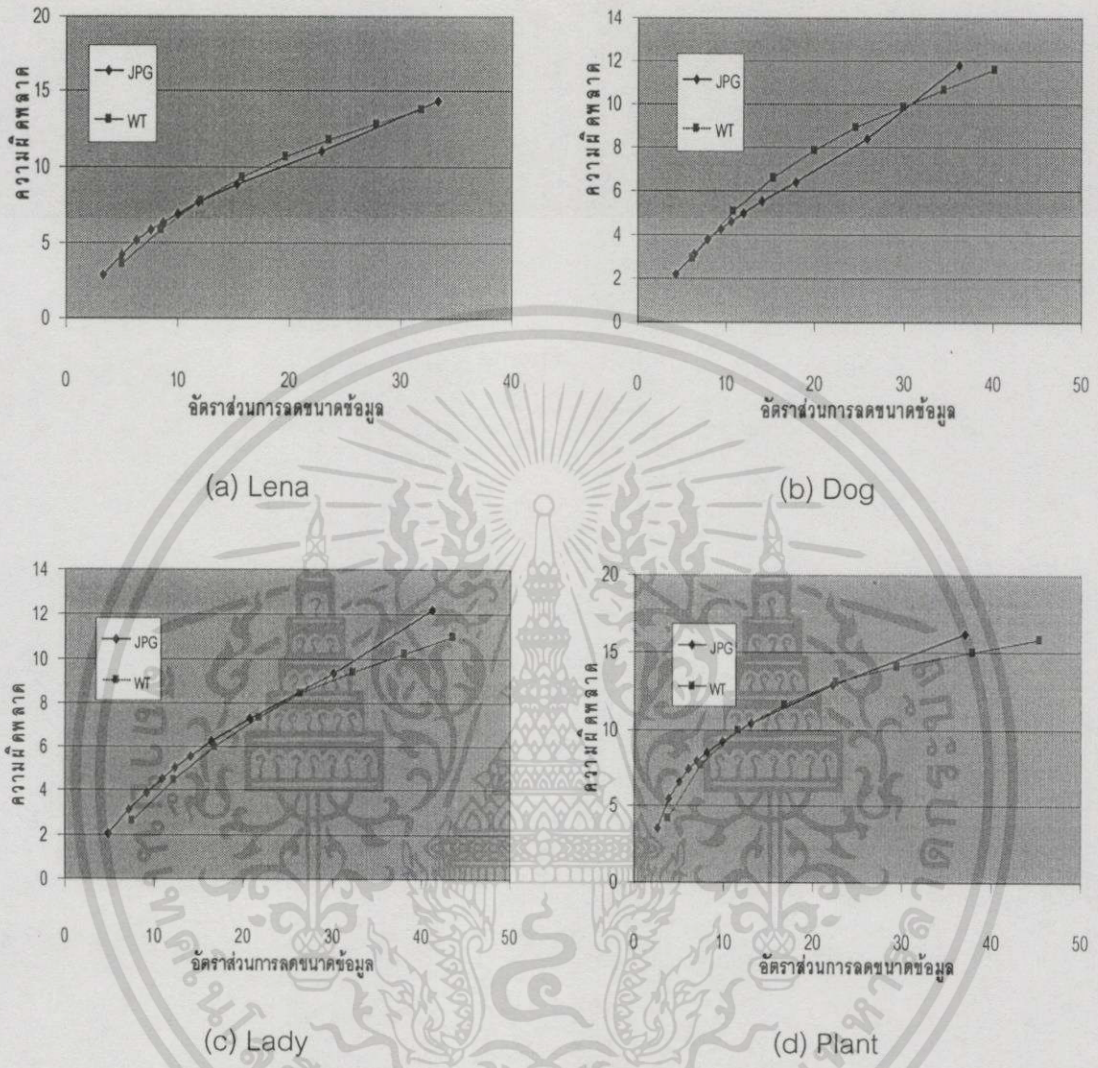
ตารางที่ 4.33 ผลการทดลองจากวิธีการ JPEG โดยใช้ภาพต้นแบบที่ 4 (PLANT)

Quality	90	80	70	60	50	40	30	20	10	5
e_{rms}	3.49	5.4	6.54	7.34	7.94	8.51	9.24	10.38	12.89	16.25
SNR_{rms}	593.6	247.47	168.4	133.76	114.27	99.5	84.23	66.51	43.16	27.1
Size	24914	16534	12987	10747	9292	8014	6641	5006	2950	1771
Bpp	3.04126	2.01831	1.58533	1.31189	1.13428	0.97827	0.81067	0.61108	0.36011	0.21619
Ratio	2.63049	3.96371	5.04628	6.09807	7.05295	8.17769	9.86839	13.09149	22.21559	37.00508



รูปที่ 4.41 แสดงความสัมพันธ์ ของอัตราส่วนการลดขนาดข้อมูลที่ได้ (Ratio) กับความผิดพลาด (e_{rms}) ที่ได้จากการขบวนการ JPEG

เมื่อนำผลการทดลองที่ได้จากขบวนการ Quantize ที่ปรับปรุงแล้ว มาแสดงเปรียบเทียบกันดังในรูปที่ 4.42



รูปที่ 4.42 แสดงผลการทดลองเปรียบเทียบ ระหว่างการลดข้อมูลด้วย Wavelet (WT) กับการใช้ขบวนการ JPEG (JPG) ของภาพต้นแบบต่างๆ

จากการเปรียบเทียบ ข้างต้น จะเห็นได้ว่า ขบวนการลดขนาดข้อมูลด้วย Wavelet transform นี้ จะ ให้ค่าความผิดพลาดที่ใกล้เคียงกับ การลดขนาดข้อมูลด้วย JPEG ที่อัตราการลดขนาดข้อมูล ต่ำกว่า 30 เท่า และ จะมีประสิทธิภาพดีกว่า ที่อัตราการลดขนาดข้อมูล มากๆ โดยจะสังเกตได้จากเส้นกราฟที่แยกออกจากกัน และเมื่อเปรียบเทียบคุณภาพของ ภาพผลลัพท์ที่ได้ โดยการประเมินด้วยสายตา ของภาพตัวอย่าง LENA กับ LADY แล้วจะได้ดังแสดงในรูปที่ 4.43 และ 4.44 ตามลำดับ

การลดขนาดด้วย Wavelet transform

(a) e_{rms} = 7.71 ; Ratio = 11.91 เท่า

การลดขนาดด้วย JPEG

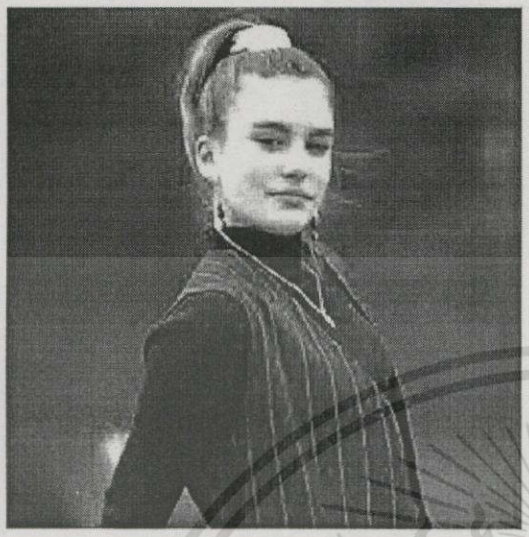
(b) e_{rms} = 7.65 ; Ratio = 11.89 เท่า

(c) e_{rms} = 14.24 ; Ratio = 33.14 เท่า

(d) e_{rms} = 14.39 ; Ratio = 33.35 เท่า

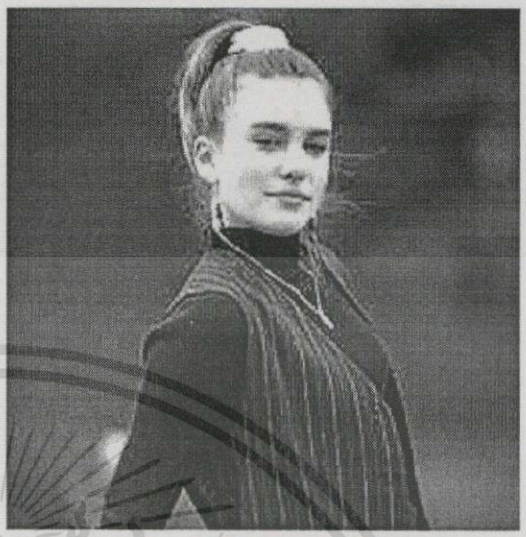
รูปที่ 4.43 แสดงเปรียบเทียบภาพผลลัพท์ที่ได้ จากขบวนการลดขนาดข้อมูลด้วย Wavelet transform กับ วิธีการของ JPEG ของภาพต้นแบบ LENA

การลดขนาดด้วย Wavelet transform



(a) e_{rms} = 6.05 ; Ratio = 16.52 เท่า

การลดขนาดด้วย JPEG



(b) e_{rms} = 6.25 ; Ratio = 16.39 เท่า

(c) e_{rms} = 10.67 ; Ratio = 41.15 เท่า

(d) e_{rms} = 12.23 ; Ratio = 41.19 เท่า

รูปที่ 4.44 แสดงเปรียบเทียบภาพผลลัพท์ที่ได้ จากขบวนการลดขนาดข้อมูลด้วย Wavelet transform กับ วิธีการของ JPEG ของภาพต้นแบบ LADY

จากการเปรียบเทียบคุณภาพของภาพ ข้างต้น ด้วยสายตา จะสามารถสรุปได้ว่า ที่ระดับการลดขนาดของข้อมูลที่ต่ำ รูปที่ 4.43, 4.44 (a) และ (b) คุณภาพของภาพที่ได้ ไม่มีความแตกต่างกัน แต่ที่ระดับการลดขนาดข้อมูลที่มากแล้ว รูปที่ 4.43, 4.44 (c) และ (d) คุณภาพของภาพที่ได้จากขบวนการของ การลดขนาดด้วย Wavelet transform จะดีกว่า ภาพจากขบวนการ JPEG ที่มีลักษณะของการเกิด Blocking effect ที่ชัดเจน ถ้าจะต้องนำข้อมูลนี้ไปใช้ในการประมวลผลต่อไป ก็จะเกิดความผิดพลาดมาก

บทที่ 5
สรุป ปัญหาและข้อเสนอแนะ

จากวัตถุประสงค์ของงานวิจัย และผลการวิจัยที่กล่าวมาทั้งหมด จะสามารถสรุปได้ดังต่อไปนี้

5.1 สรุปทฤษฎีเวฟเล็ต

การแปลงเวฟเล็ต เป็นการแตกกระจายของสัญญาณออกเป็นส่วนประกอบย่อยๆ ของสมาชิกในเซตของเวฟเล็ต ที่มีขนาดต่างๆ กัน จากการคูณกับค่าสัมประสิทธิ์ สมาชิกในเวฟเล็ตเซตนี้จะได้มาจาก เวฟเล็ตแม่ ที่มีค่าสเกล (Scale: a) และการเลื่อนตำแหน่ง (Translation: b) ต่างๆ กัน เป็นไปดังสมการ $g_{b,a}(t) = \frac{1}{\sqrt{a}} g\left(\frac{t-b}{a}\right)$ ค่าสัมประสิทธิ์เวฟเล็ตที่ได้นี้ จะสามารถใช้เป็นตัวแทนของสัญญาณได้ สำหรับการแปลงผกผันเวฟเล็ตนั้น จะสามารถทำได้โดยการนำค่าสัมประสิทธิ์นี้ มาคูณกับ เวฟเล็ตเซตที่สัมพันธ์กัน ก็จะได้สัญญาณ ที่ต้องการกลับออกมา

ซึ่งการแปลงเวฟเล็ตที่นิยมใช้กัน และที่ใช้ในงานวิจัยนี้ จะเป็นการแปลงเวฟเล็ตแบบ Discrete Wavelet Series ที่มีค่าของ a b และ t เป็นแบบเต็มหน่วย โดยจะใช้ค่าของ $a = a_0^m$ และ $b = nb_0a_0^m$ ที่ค่าของ $a_0 = 2$ และ $b_0 = 1$ เนื่องจากขบวนการแปลงเวฟเล็ตจะสามารถทำได้ง่าย

จากข้างต้นนี้ ถ้าสมาชิกในเวฟเล็ตเซต แต่ละตัวไม่มีความซ้ำซ้อนกัน หรือมีคุณสมบัติของการเป็น Orthogonal แล้ว ค่าสัมประสิทธิ์เวฟเล็ตที่ได้ จะไม่มีความซ้ำซ้อนกัน และคุณสมบัติของค่าสัมประสิทธิ์เวฟเล็ตที่สำคัญอีกประการหนึ่งคือ การแปลงเวฟเล็ตจะให้ผลลัพท์ออกมาทั้งในแกนของความถี่ และตำแหน่ง จึงสามารถที่จะเลือกตัดข้อมูล ณ ตำแหน่ง หรือความถี่ใดๆ ได้อิสระ ซึ่งเหมาะที่จะนำมาใช้ในขบวนการลดขนาดข้อมูลต่อไป

การที่จะแปลงเวฟเล็ตของข้อมูลภาพ จะทำได้โดยการพิจารณาข้อมูลภาพที่เป็นสองมิติออกเป็นกลุ่มของข้อมูลเพียงมิติเดียว ก่อน โดยการพิจารณา ข้อมูลภาพ ออกเป็นเส้นภาพ ตามแนวตั้งและแนวนอน แล้วจึงนำขบวนการแปลงเวฟเล็ตของของข้อมูล หนึ่งมิติมาใช้งาน

5.2 สรุปงานวิจัย

จากค่าสัมประสิทธิ์ที่ได้ จากการแปลงเวฟเล็ตของข้อมูลภาพนั้น จะนำมาผ่านขบวนการลดขนาดข้อมูล เพื่อให้ได้ข้อมูลที่ลดลงนั้น ขั้นตอนที่ใช้จะสามารถแบ่งได้สองขั้นคือ ขบวนการที่ใช้ลดรายละเอียดของข้อมูลภาพ และขบวนการหารหัสที่เหมาะสมสำหรับข้อมูลค่าสัมประสิทธิ์

ขบวนการที่จะลดรายละเอียดของข้อมูลภาพ จะได้จากการทำ Threshold ของค่าสัมประสิทธิ์ เนื่องจากค่าสัมประสิทธิ์นี้ จะแสดงถึงขนาด ของเวฟเล็ตที่ตำแหน่งและความถี่ใดๆ ดังนั้นส่วนประกอบที่มีขนาดเล็ก หรือไม่สำคัญ จึงสามารถตัดทิ้งได้ และค่าสัมประสิทธิ์ที่ได้ จะเป็นเลขจำนวนจริง จึงต้องทำการ Quantize ของข้อมูลด้วย เพื่อเป็นการลดขนาดของข้อมูลให้ได้มากยิ่งขึ้น ขบวนการที่กล่าวมานี้ จะทำให้เกิด ความผิดพลาด กับข้อมูลภาพที่จะได้

ขบวนการที่ใช้หารหัสที่เหมาะสม จะทำโดยใช้วิธีการของ Run-Length Coding ของข้อมูล '0' ทั้งนี้ เมื่อพิจารณาข้อมูลที่ผ่านขบวนการ Threshold และ Quantize แล้วจะพบว่าข้อมูลส่วนใหญ่ที่เหลืออยู่ จะเป็นข้อมูลของ '0' แล้วจึงใช้ขบวนการ Huffman Coding ในการหารหัสที่เหมาะสมต่อไป

จากหลักการข้างต้นได้ถูกนำมาเขียนเป็นโปรแกรม เพื่อใช้ในการทดลอง ซึ่งจะให้ผลการทดลองสรุปได้ดังต่อไปนี้

1. สำหรับข้อมูลภาพ แต่ละภาพแล้ว อัตราส่วนการลดขนาดข้อมูลที่ได้ จะมีค่าไม่เท่ากันที่ระดับความผิดพลาดหนึ่งๆ ทั้งนี้ขึ้นอยู่กับว่า ภาพที่นำมาลดขนาดนั้น มีรายละเอียดมากเพียงใด และลักษณะของภาพ
2. อัตราการลดขนาดข้อมูล ต่อค่าความผิดพลาด จะเพิ่มมากขึ้นที่ การลดขนาดข้อมูลที่เพิ่มขึ้น เนื่องจากจะมีข้อมูลของ '0' อยู่ติดต่อกันมากขึ้น
3. อัตราการลดขนาดข้อมูลที่ได้ เป็นไปตามวัตถุประสงค์ที่ตั้งไว้ คือสามารถใช้ลดขนาดข้อมูลภาพได้ถึง15 เท่า โดยภาพยังคงสามารถสื่อความหมายได้
4. เมื่อเปรียบเทียบกับวิธีการลดขนาดข้อมูลแบบ JPEG แล้วพบว่าอัตราการลดขนาดข้อมูล ต่อค่าความผิดพลาด ที่ได้จากขบวนการที่วิจัยขึ้นมานี้ จะให้มีประสิทธิภาพดีกว่า ที่อัตราการลดขนาดข้อมูลมาก และจะมีประสิทธิภาพใกล้เคียงกันที่อัตราการลดขนาดข้อมูลน้อย และคุณภาพของภาพที่ผ่านขบวนการลดขนาดข้อมูลด้วยเวฟเล็ตนี้ จะไม่มีปัญหาของ Blocking effect ที่อัตราการลดขนาดข้อมูลมาก

จากผลการทดลองที่ได้ทั้งหมดนี้ จะได้เป็นแนวทางที่ผู้สนใจจะสามารถนำมาศึกษาค้นคว้าเพื่อพัฒนาขบวนการลดขนาดข้อมูลของสัญญาณต่างๆ ได้ต่อไป ไม่เฉพาะแต่การลดขนาดข้อมูล

ภาพเท่านั้น และจากข้อดีต่างๆ ที่ได้จากการแปลงเวฟเล็ตของข้อมูลภาพ ค่าสัมประสิทธิ์ที่ได้นี้ยังสามารถที่จะนำไปประยุกต์ใช้ในการ วิเคราะห์ข้อมูลภาพได้ต่อไป

5.3 ปัญหาและข้อเสนอแนะ

ปัญหาที่พบในการลดขนาดข้อมูลภาพด้วยการแปลงเวฟเล็ต ที่สำคัญคือการออกแบบขบวนการ Quantize และ Coding ที่เหมาะสม โดยเฉพาะกับข้อมูลของ '0' ที่มีอยู่มาก การใช้ขบวนการ Run-Length Coding เป็นวิธีการที่ดี แต่ถ้าข้อมูลของ '0' มีอยู่กระจายกันไปแล้ว จะทำให้ไม่สามารถลดขนาดข้อมูลได้เท่าที่ควร แนวทางการแก้ไขสามารถจะทำได้โดยการจัดเรียงข้อมูล ใหม่ให้มีข้อมูลของ '0' ที่อยู่ติดต่อกันมากยิ่งขึ้น และนอกจากนี้ปัญหาที่พบอื่นๆ สามารถที่จะแยกย่อยได้ดังต่อไปนี้

ในส่วนของการแปลงเวฟเล็ต เนื่องจากขบวนการแปลงเวฟเล็ตที่ใช้ เป็นแบบ Discrete Time Wavelet Series ที่ใช้ค่าของ $a_0 = 2$ นั้นจะเห็นได้ว่าการเพิ่มของ ค่าการ Scaling $a = 2^m$ จะเป็นผลให้ ขนาดของข้อมูลจำเป็นที่จะต้องมีขนาดเป็น 2^m (m : 1 2 3 ...) ด้วย แต่ในทางปฏิบัติแล้ว ขนาดของข้อมูลภาพ จะมีแตกต่างกันออกไป ในกรณีที่ขนาดของข้อมูล ไม่เท่ากับ 2^m (m : 1 2 3 ...) แล้ว การแปลง หรือการแตกกระจายเวฟเล็ตนี้ จะไม่สามารถทำได้ครบทุกระดับ ความละเอียด เป็นผลให้การลดขนาดข้อมูลทำได้ไม่ดีเท่าที่ควร แนวทางการแก้ไขปัญหานี้ คือ จะต้องออกแบบขบวนการ Quantize และ Coding แยกเป็นสองส่วน ระหว่างส่วนของค่าสัมประสิทธิ์เวฟเล็ต กับส่วนของข้อมูลภาพที่ไม่สามารถทำการแปลงต่อไปได้

การทำ Threshold ที่ใช้ในงานวิจัยนี้ จะมีค่าคงที่ในทุกส่วนของข้อมูลภาพ การปรับปรุงขบวนการ Threshold ให้สามารถตัดข้อมูลได้มากขึ้น ซึ่งทำได้โดยการกำหนด ค่าของ Threshold ให้เพิ่มขึ้นในบริเวณ ส่วนของภาพที่มีความสว่างมาก

บรรณานุกรม

[1] A. S. Lewis and G. Knowles, "Image Compression Using the 2-D Wavelet Transform," *IEEE Trans. Image Processing*, vol. 1, no. 2, pp. 244-250, 1992.

[2] Stephane G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, pp. 674-693, 1989.

[3] Eric J. Stollnitz, Tony D. DeRose and David H. Selesin, "Wavelets for Computer Graphics: A Primer, Part 1," *IEEE Computer Graphics and Applications*, pp. 76-84, May 1995.

[4] P. P. Vaidyanathan, "Quadrature Mirror Filter Banks, M-Band Extensions and Perfect-Reconstruction Techniques," *IEEE Assp Magazine*, pp. 4-20, July 1987

[5] M. A. Sid-Ahmed, "Image Processing Theory, Algorithms, & Architectures." *McGraw-Hill, Inc.*, pp. 377-400, 1995.

[6] I. Daubechies, "Orthonomal bases of compactly supported wavelet," *IEEE Trans. Inform. Theory*, vol. 36, pp. 961-1005, Sept 1990.

[7] Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing", *Addison Wesley, Inc.*, 1992.

[8] Randy K. Young, "Wavelet Theory and its Applications", Kluwer Academic Publishers, 1993.

ภาคผนวก ก

ผลงานวิจัยที่รับการตีพิมพ์

1. ชินภัทร นันทจิวากรชัย, มนัส สังวรศิลป์, สุรพันธุ์ เอื้อไพบูลย์, "การลดขนาดข้อมูลภาพโดยใช้การแปลงเวฟเล็ต", วิศวสาร ลาดกระบัง, ปีที่ 13 ฉบับที่ 2, หน้าที่ 28-35 , 2540

ภาคผนวก ข

โปรแกรมที่ใช้ในงานวิจัย

สำหรับโปรแกรมที่ใช้ในงานวิจัยนี้ จะแบ่งออกได้เป็นสองส่วนคือ ในส่วนแรกจะเป็นโปรแกรมย่อยๆ สำหรับการทดลองและศึกษา ทฤษฎีพื้นฐานต่างๆ ของการแปลงเวฟเล็ต และคุณสมบัติของค่าสัมประสิทธิ์ สำหรับการลดขนาดข้อมูลภาพ ซึ่งโปรแกรมในส่วนแรกนี้ เพื่อความสะดวก จึงได้เขียนขึ้นด้วย ภาษา/คำสั่งของ Matlab® และเมื่อเข้าใจ ถึงขบวนการต่างๆ ทั้งหมดแล้ว จึงได้รวบรวมส่วนต่างๆ เขียนขึ้นเป็นโปรแกรมสำหรับการลดขนาดข้อมูลภาพ จากข้อจำกัดของโปรแกรมที่เขียนด้วย Matlab® คือ การทำงานที่ช้า ดังนั้นโปรแกรมสำหรับการลดขนาดข้อมูลภาพ จึงจะเขียนโดยใช้ภาษา Delphi®

โปรแกรมที่เขียนด้วย Matlab®

โปรแกรมในส่วนของ Matlab® นี้จะใช้ในการทดลอง การแปลงเวฟเล็ตของสัญญาณ และการทดลอง การเลือก การแปลง ที่เหมาะสม ในการแปลงเวฟเล็ตของสัญญาณ จะทำโดยการเขียน เป็นฟังก์ชัน การแปลงเวฟเล็ต และการแปลงผกผันเวฟเล็ต ตามหลักการที่ได้อธิบายในหัวข้อ วิธีการที่มีประสิทธิภาพในการแปลงเวฟเล็ต ได้เป็นโปรแกรมต่อไปนี้

ฟังก์ชันการแปลงเวฟเล็ต [fwt2.m]

จะทำการแปลงเวฟเล็ตของสัญญาณอินพุท (X) ด้วยเวฟเล็ตแม่ (QMF) และส่งกลับ ค่าสัมประสิทธิ์เวฟเล็ต กับสัญญาณส่วนที่เหลือ

```
function   wcoef = fwt2( X,  QMF)
% FWT2  Fast Wavelet Transform.
% wcoef = fwt2( X, QMF)
% Where
%          wcoef     Wavelet coefficients []
%          X         Input Siganl[]
%          QMF       Wavelet

G = reverse( QMF);
H = reverse(mirrorfilt( G));


[ Xm, Xn] = size( X);
x1 =  X( :, 1:2:Xn);
x2 =  X( :, 2:2:Xn);

[ Gm, Gn] = size( G);
g2 =  G( :, 1:2:Gn);
g1 =  G( :, 2:2:Gn);

h2 =  H( :, 1:2:Gn);
h1 =  H( :, 2:2:Gn);

n = Gn/2;
```

```
x1g1 = conv2( [ x1 x1(1:(n-1))], g1, 'valid');
x2g2 = conv2( [ x2 x2(1:(n-1))], g2, 'valid');
x1h1 = conv2( [ x1 x1(1:(n-1))], h1, 'valid');
x2h2 = conv2( [ x2 x2(1:(n-1))], h2, 'valid');

wcoef = [ (x1g1+x2g2) (x1h1+x2h2)];
```

ฟังก์ชัน แปลงผกผันเวฟเล็ต [iwt2.m]

ฟังก์ชันการแปลงผกผันเวฟเล็ตนี้ จะรับค่าของสัมประสิทธิ์เวฟเล็ต (W) และ เวฟเล็ตแม่ที่ใช้ (QMF) โดยจะส่งกลับ เป็นสัญญาณที่แปลงผกผันแล้ว

```
function   data = iwt2( W, QMF)
% IWT2   Fast Invert Wavelet Transform.
% data = iwt2( W, QMF)
% Where
%         data       Data []
%         W          Waveley coefficient []
%         QMF        Wavelet
G = QMF;
H = mirrorfilt( reverse(G));

[ Wm, Wn] = size( W);
w1 =  W( 1:Wn/2);
w2 =  W( Wn/2+1:Wn);

[ Gm, Gn] = size( G);
g1 =  G( :, 1:2:Gn);
g2 =  G( :, 2:2:Gn);
h1 =  H( :, 1:2:Gn);
h2 =  H( :, 2:2:Gn);

n = Gn/2;
w1g1 = conv2( [ w1 w1(1:(n-1))], g1, 'valid');
w1g2 = conv2( [ w1 w1(1:(n-1))], g2, 'valid');
w2h1 = conv2( [ w2 w2(1:(n-1))], h1, 'valid');
w2h2 = conv2( [ w2 w2(1:(n-1))], h2, 'valid');

d1 = [ w1g1; w1g2];
d1 = d1(:)';
d2 = [ w2h1; w2h2];
d2 = d2(:)';
data = d1+d2;
```

การทดลอง ในหัวข้อการเลือก Transform ที่เหมาะสม จะทำโดยการเขียนโปรแกรมให้คอมพิวเตอร์ทำการคำนวณ ค่าความผิดพลาดที่เกิดขึ้น เมื่อเลือกใช้ สัมประสิทธิ์เวฟเล็ตจำนวนต่างๆ กัน

โปรแกรมที่ใช้ในส่วนนี้ จะประกอบด้วยโปรแกรมย่อยๆ ต่อไปนี้

โปรแกรม อ่านข้อมูลภาพ [showpic.m]

ทำหน้าที่อ่านข้อมูลภาพต้นแบบ มาแสดง และเก็บเป็นตัวแปรเริ่มต้น (AX)

```
[fN, pN] = uigetfile( '*.img *.bw', 'Input image');
fid = fopen( [pN fN], 'r');
MM = input( 'Enter image M size : ');
NN = input( 'Enter image N size : ');
fig = input( 'Enter figure number to display : ');
AX = fread(fid,[MM, NN]);
fclose(fid);
AX = AX';

figure( fig);
```

```
image( AX);
axis( 'off');
title( ['Image of ',fN]);
colormap( gray( 256));
```

ฟังก์ชัน สำหรับการเลือกค่าสัมประสิทธิ์ [select.m และ dist.m]

ฟังก์ชัน สำหรับการเลือกข้อมูล (Y) ตามจำนวนที่ต้องการ (perc) โดยในฟังก์ชันนี้ จะทำการแจกแจงข้อมูล [dist.m] ตามลำดับจากขนาดมาก ไปน้อย แล้วทำการเลือกข้อมูลตามเปอร์เซ็นต์ที่กำหนด

```
function    X = select( Y, perc)

% SELECT          Select 1st half of coefficient by maximum magnitude.
%  X = select( Y, perc)
% Where
%         X       Percent of original coefficient by maximum magnitude
%         Y       Original coefficient
%         perc    Percent of coefficient needed

[ data, count] = dist( abs(Y(:)));
[ m, n] = size( Y);

s = 0;
i = 0;
while ( s <= ( (m*n) *(1- perc) ) )
        i = i + 1;
        s = s + count( i);
        end;
BY = abs( Y) >= data( i);
X = Y .* BY;
```

ฟังก์ชัน สำหรับการแจกแจงค่าสัมประสิทธิ์ [dist.m] จากข้อมูลอินพุท (Z) จะทำการหาการแจกแจงของข้อมูล โดยจะส่งค่ากลับเป็น ค่า (bin) และจำนวนที่พบ (count)

```
function   [ bin, count] = dist( Z)
% DIST              Distribution of Z ( use only for data type > 3 )
% [ bin, count] = dist( Z);
% Where
%         bin       Vector of value in Z
%         count     Number of occurrence in each bin
%         Z         Input data

s = sort( Z(:));
mark = conv( s, [ 1 -1]);
[ a, b] = size( mark);
mark = mark( 2:a);
bin = s( mark ~= 0);
mark_pos = find( mark ~= 0);
count = conv( mark_pos, [ 1 -1]);
[ a, b] = size( count);
count = count( 1:a-1);
```

โปรแกรม หาค่าความผิดพลาดเมื่อใช้เวฟเล็ต [rdaub4.m rdaub8.m rsymm4.m rsymm5.m]

โปรแกรมจะทำการแปลงเวฟเล็ตของข้อมูลภาพเริ่มต้น (AX) แล้วเรียกใช้ฟังก์ชันการเลือกค่าสัมประสิทธิ์ จำนวนต่างๆ กัน และนำข้อมูลที่เหลือมาทำการแปลงผกผันเวฟเล็ต เพื่อหาค่าความผิดพลาดที่เกิดขึ้น (e)

ที่แสดงต่อไปนี้เป็นโปรแกรม หาค่าความผิดพลาดเมื่อใช้ ค่าสัมประสิทธิ์เวฟเล็ต จำนวนต่างๆ ของเวฟเล็ตแม่ Daubechies4 [rdaub4.m] ส่วนโปรแกรมสำหรับเวฟเล็ตแม่อื่นๆ นั้นจะแตกต่างที่ตัวเวฟเล็ตที่ใช้เท่านั้น

```
%       compute error when using Daubechies 4

qmf = makeonfilter('Daubechies',4);

Y = fwt_po( AX, 1, qmf);
e = zeros( 10,1);
for j = 1:10
        X = select( Y, j/20);
        OUT = iwt_po( X, 1, qmf);
        e( j) = sqrt( sum(( AX(:) - OUT(:)).^2) / 65536);
        end;
figure
plot(e)
title( ['Daubechies 4 : E rms ']);
```

โปรแกรม หาค่าความผิดพลาดเมื่อใช้ DCT [rdct8.m rdct16.m rdct32.m]

โปรแกรมจะทำการ จัดเรียงข้อมูล ใหม่ให้ได้เป็น Block ตามที่ต้องการ (8 16 32) ด้วยคำสั่ง reshape แล้วทำการแปลง DCT ของข้อมูลภาพ และเรียกใช้ฟังก์ชันการเลือกค่าสัมประสิทธิ์นำข้อมูลที่เหลือมาทำการแปลงผกผัน DCT แล้วเรียงข้อมูลกลับเช่นเดิม เพื่อนำมาหาค่าความผิดพลาดต่อไป

ตัวอย่างโปรแกรม หาค่าความผิดพลาดเมื่อใช้ ค่าสัมประสิทธิ์ DCT8 จำนวนต่างๆ [rdct8.m]

```
%       compute erroe when using DCT Block @8
Y = zeros( 256, 256);
for i = 1:256
        ORG = AX( :, i);
        temp = dct( reshape( ORG, 8, 32));
        Y( :, i) = temp( :);
        end;
X = select( Y,.1);

OUT = zeros( 256, 256);
for i = 1:256
        ORG = X( :, i);
        temp = idct( reshape( ORG, 8, 32));
        OUT( :, i) = temp( :);
        end;
%OUT=OUT';
figure
colormap( gray( 256));
image( OUT);

e = sqrt( sum(( AX(:) - OUT(:)).^2) / 65536);
title( ['DCT  8: E rms = ',num2str(e)]);
```

โปรแกรมที่เขียนด้วย Delphi®

โปรแกรมในส่วนที่สอง ซึ่งเขียนขึ้นด้วยภาษา Delphi® นี้ จะใช้ในการทดลองลดขนาดข้อมูลภาพ โดยจะสามารถเลือกและแสดงรูปภาพต้นแบบได้ สามารถแปลงเวฟเล็ตและแสดงค่าของสัมประสิทธิ์เวฟเล็ตได้ สามารถทำ Quantize ของค่าสัมประสิทธิ์ สามารถแสดงอัตราการลด

ขนาดข้อมูล และสามารถเปรียบเทียบความแตกต่างของข้อมูลสองชุดได้ โปรแกรมการทำงานทั้งหมดนี้ จะถูกแบ่งออกเป็น Unit ต่างๆ ดังต่อไปนี้

MdiApp.dpr

เป็น Delphi project

โปรแกรม Mdiapp.dpr

```
program Mdiapp;

uses
  Forms,
  Main in 'MAIN.PAS' {MainForm},
  Childwin in 'CHILDWIN.PAS' {MDIChild},
  About in 'about.pas' {AboutBox},
  Quantize in 'Quantize.pas' {QuantizeDlg},
  compare in 'compare.pas' {CompareDlg},
  Coding in 'Coding.pas' {CodingDlg};

{$R *.RES}

begin
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TAboutBox, AboutBox);
  Application.CreateForm(TQuantizeDlg, QuantizeDlg);
  Application.CreateForm(TCompareDlg, CompareDlg);
  Application.CreateForm(TCodingDlg, CodingDlg);
  Application.Run;
end.
```

Main.pas , Main.dfm

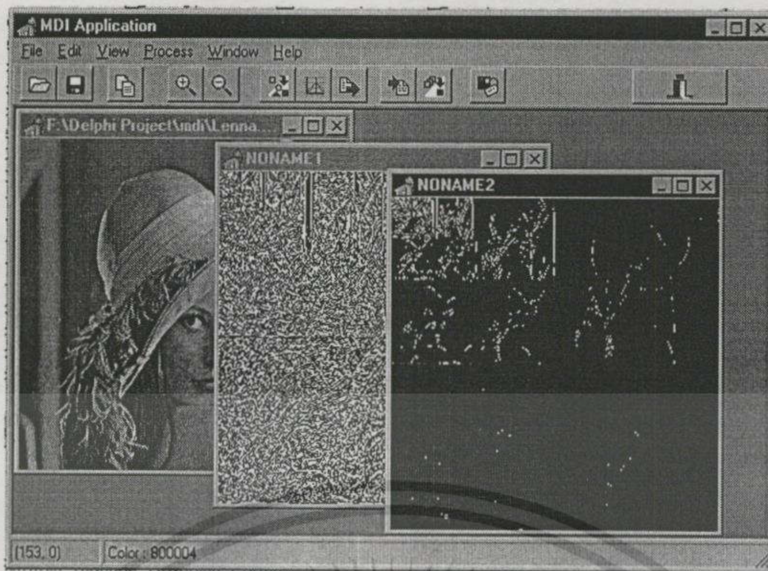
เป็นโปรแกรมหลักที่จะเรียกใช้การทำงานของ Procedure หรือ Unit อื่นๆ และยังมีหน้าที่ในการแปลงเวฟเล็ตในตัวโปรแกรมจะประกอบด้วย Procedure ที่สำคัญ ดังนี้

TMainForm.DTWS1Click(Sender: TObject); เมื่อผู้ใช้ต้องการแปลงเวฟเล็ตของข้อมูลภาพ ข้อมูลภาพจะถูกแปลงให้เป็น array ของข้อมูล (data) แล้วส่งให้ Procedure DTWS2 ทำการแปลงเวฟเล็ต หลังจากที่แปลงเวฟเล็ตแล้ว ค่าสัมประสิทธิ์ที่ได้ จะถูกส่งกลับมาที่ตัวแปล (data) เช่นเดิม โปรแกรมจะทำการ แปลงค่าสัมประสิทธิ์ให้เป็นรูปภาพ และแสดงผลใน Child window

DTWS2(var Input: array of Single; const xx, yy, xz, yz: Integer); โปรแกรมจะรับข้อมูลอินพุท (input) และขนาดของ อินพุท array (xx, yy) และกำหนดขนาดของข้อมูลที่ต้องการแปลงเวฟเล็ต (xz, yz) โดยมีค่าเริ่มต้นเท่ากับ (xx, yy) โปรแกรมจะทำการแปลงเวฟเล็ตของข้อมูลถ้า (xz) และ (yz) หารสองแล้วลงตัว และจะทำการเรียกตัวเอง โดยจะลดค่าของ (xz) และ (yz) ลงครึ่งหนึ่ง เพื่อเป็นการแปลงเวฟเล็ตที่ระดับความละเอียดต่ำลง

TMainForm.IDTWS1Click(Sender: TObject); โปรแกรมจะแปลงภาพของค่าสัมประสิทธิ์ให้ได้เป็นค่าสัมประสิทธิ์ ในตัวแปล (data) แล้วส่งให้ Procedure IDTWS2 ทำการแปลงผกผันเวฟเล็ต หลังจากข้อมูลที่แปลงผกผันแล้วส่งกลับมาที่ตัวแปล (data) โปรแกรมจะแสดงผลรูปภาพที่ได้ใน Child window

IDTWS2(var Input: array of Single; const xx, yy, xz, yz: Integer); เช่นเดียวกับ DTWS2 แต่การทำงานจะเริ่มจาก การเรียกตัวเองก่อน ถ้า (xz) และ (yz) หารสองแล้วลงตัว จากนั้นจึงทำการแปลงผกผันเวฟเล็ต การทำงานก็จะเป็นการแปลงผกผันเวฟเล็ตที่ระดับความละเอียดที่ต่ำที่สุดก่อน แล้วจึงทำระดับที่สูงขึ้นมา



รูปแสดงหน้าต่างของโปรแกรม Mdiapp.exe ที่ประกอบด้วยหน้าต่างย่อยของ ภาพต้นแบบ LENA ค่าสัมประสิทธิ์เวฟเล็ต ก่อนและ หลังการ Quantize

โปรแกรม Main.dfm

```
object MainForm: TMainForm
 Left = 132
 Top = 131
 Width = 568
 Height = 391
 Caption = 'MDI Application'
 Color = clAppWorkSpace
 Font.Charset = DEFAULT_CHARSET
 Font.Color = clBlack
 Font.Height = -11
 Font.Name = 'MS Sans Serif'
 Font.Style = []
 FormStyle = fsMDIForm
 Menu = MainMenu1
 Position = poDefault
 WindowMenu = Window1
 OnCreate = FormCreate
 OnDestroy = FormDestroy
 PixelsPerInch = 96
 TextHeight = 14
 object SpeedPanel: TPanel
  Left = 0
  Top = 0
  Width = 560
  Height = 32
  Align = alTop
  ParentShowHint = False
  ShowHint = True
  TabOrder = 0
  object OpenBtn: TSpeedButton
   Left = 6
   Top = 2
   Width = 28
   Height = 27
   Hint = 'Open|'
   Glyph.Data = {
06020000424D0602000000000000760000002800000
28000000140000000100040000000000090010000000000
00000000000100000000000000000000000000000800000
80000000808000800000008000800080800000C0C0C0
0080808000000000FF0000FF000000FFFF00FF000000F
F00FF00FFFF0000FFFFFF0033333333333333333333
333333333333333333333333333333333333333333333333
333333333333333333333333333333333333333333333333
3333333333333333333333333333333333333FFFFFFFFFF
FFFF3333380000000000000333333888888888888883
F3333007B7B7B7B7B7B033333883F33333333338F33
330F07B7B7B7B7B70333338F8F33333333383F3330
B0B7B7B7B7B7B7033338F83F33333333338F3330FB0
B7B7B7B7B7B033338F38F333333333383F330BF07B7
B7B7B7B7B03338F383FFFFF3333338F330FBF00000
7B7B7B703338F33888883FFFFFF83330BFBFBFBF00
0000033338F333333888888833330FBFBFBFBFBFB0
3333338F33333333338F333330BFBFBFBFBFBF0333
3338F33333FFFFFF83333330FBFBF0000000333333
87FFFF888888833333333000003333333333333888
8833333333333333333333333333333333333333333333
333333333333333333333333333333333333333333333333
333333333333333333333333333333333333333333333333
3333333333333333333333333333333333333333}
   NumGlyphs = 2
   OnClick = FileOpenItemClick
  end
  object SaveBtn: TSpeedButton
   Left = 34
   Top = 2
   Width = 28
   Height = 27
   Hint = 'Save|'
   Enabled = False
   Glyph.Data = {
06020000424D0602000000000000760000002800000
28000000140000000100040000000000090010000000000
00000000000100000000000000000000000000000800000
80000000808000800000008000800080800000C0C0C0
0080808000000000FF0000FF000000FFFF00FF000000F
F00FF00FFFF0000FFFFFF0033333333333333333333
333333333333333333333333333333333333333333333333
3333333333333333333333333333333333333333333FFFFF
FFFFFFFFF333338000000000000008333333888F8FF88
8F888F333330CC08CCF770CC0333333888F8FF888F
888F333330CC08CCF770CC0333333888F888888F88
8F333330CC07887770CC03333338888FFFFFF8888F3
33330CC60000006CC03333333888888888888888F3333
30CCCCCCCCCCCC03333338888888888888888F33333
0C6000000006C03333333888888888888888F333330C0
FFFFFFFF0C0333333888FFFFFFFF888F333330C0FF
FFFFFF0C0333333888FFFFFFFF888F333330C0FFFF
FFFF0C0333333888FFFFFFFF888F333330C0FFFFFF
FF0C0333333888FFFFFFFF888F33333000FFFFFFFF
000333333888FFFFFFFF888F333330C0FFFFFFFF0C
```

```
0333333888FFFFFFFF888F333338000000000008333
33388888888888883333333333333333333333333333
33333333333333333333333333333333333333333333
33333333333333333333333333333333333333333333
33333}
    NumGlyphs = 2
    OnClick = FileSaveItemClick
  end
  object CopyBtn: TSpeedButton
    Left = 73
    Top = 2
    Width = 28
    Height = 27
    Hint = 'Copy|'
    Enabled = False
    Glyph.Data = {
06020000424D060200000000000076000000028000000
28000000140000000100040000000000960100000000000
0000000000010000000000000000000000000000800000
80000000808000800000008000800080800000C0C0C0
00808080000000FF0000FF000000FFFF00FF000000F
F00FF00FFFF0000FFFFFF0033333333333333333333
33333333FFFFFFFFFF333333333000000000003333333
3338888888888F3333333330FFFFFFFF033333333338
F33333338F3333333330BFBFBFBF033333333338F3F
FFFFF8F3333333330F888888F033333333338F888888
38F3333333330BFBFBFBF0333333FFFF8F3FFFFFF8
F3333000000F888888F033333888888F88888838F333
30FFFF0BFBFBFBF0333338F3338F3FFFFFF8F33330
FFFF0F888888F0333338F3338F88888838F33330F777
0BFBFBFBF0333338F3338F333333F8F33330FFFF0F
FFFFF000333338F3338F33333888333330F7770BFBF
BF003333338F3338FFFFFF883333330FFFF00000000
33333338F33388888888333333F777777F03333333
3338F33333338F3333333330FFFFFFFF033333333338
F333333F8F3333333330FFFFFF00033333333338F333
3388833333333330FFFFFF00333333333338FFFFFF8
83333333333000000003333333333338888888833333
33333333333333333333333333333333333333333333
33333333333333333333333333333333333333333333}
    NumGlyphs = 2
  end
  object ExitBtn: TSpeedButton
    Left = 483
    Top = 2
    Width = 74
    Height = 27
    Hint = 'Exit|'
    Glyph.Data = {
06020000424D060200000000000076000000028000000
28000000140000000100040000000000960100000000000
0000000000010000000000000000000000000000800000
80000000808000800000008000800080800000C0C0C0
00808080000000FF0000FF000000FFFF00FF000000F
F00FF00FFFF0000FFFFFF0037777777777777777777
3FFFFFFFFFFFF333333F888888888888F7F7F788888
88888888833333888888888888877F7F78888888888
8888F333FF88844444400888FFF44444488888888888
3338888833333334D5007FFF433333333338F888F3338
F33333333345D50FFFF4333333333388788F3338F333
3333334D5D0FFFF433333333338F878F3338F333333
33345D50FEFE4333333333388788F3338F333333333
4D5D0FFFF433333333338F878F3338F33333333345D
50FEFE4333333333388788F3338F3333333334D5D0F
FFF433333333338F878F3338F33333333345D50FEFE
4333333333388788F3338F3333333334D5D0EFEF433
333333338F878F3338F33333333345D50FEFE433333
3333388788F3338F3333333334D5D0EFEF433333333
338F8F8FFFF8F33333333444444444433333333388
88888883333333333333333333333333333333333FFF
FFF333333333333300000033333333333333888888F3
3333333333330AAAA0333333333333338FFFF8F3333
33333333000000333333333333338888883333333}
    NumGlyphs = 2
    OnClick = FileExitItemClick
  end
  object DTWSBtn: TSpeedButton
    Left = 192
    Top = 2
    Width = 28
    Height = 27
    Hint = 'DTWS'
    Enabled = False
    Glyph.Data = {
76010000424D760100000000000076000000028000000
20000000100000000100400000000000001000012 0B0
000120B000010000000000000000000000000080000
08000000080800080000000800080008080000007F7F7
F00BFBFBF000000FF0000FF000000FFFF00FF000000
FF00FF00FFFF0000FFFFFF003FF0000000F0000033F
77777773777773FFF0CCC0FF09990333F73F37337F3
3733FFF0C0FFF099903333F7373337F337333FFF0FF
FF0999033333F73FFF7FFF73333FFF000F000003333
33F77737777733333F07B70FFFFF3333337F337F333
33333330BBB0FFFFF3333337F337F333333333307B7
0FFFFF33333373FF733F333333333000FF0FFF33333
33777337FF3333333333FF000FF33FFFFF3333777FF
300000333300000F377777F33377777F30EEE0333000
000037F337F33777777730EEE033333000FF37F337F
3333777F330EEE033333000FF37FFF7F3333777F330
000033333000FF3777773333F7773333333333300003
3F3333333337777333}
    NumGlyphs = 2
    OnClick = DTWS1Click
  end
  object DecodingBtn: TSpeedButton
    Left = 286
    Top = 2
    Width = 28
    Height = 27
    Hint = 'Decoding'
    Enabled = False
    Glyph.Data = {
76010000424D760100000000000076000000028000000
20000000100000000100400000000000001000013 0B
0000130B000010000000000000000000000000080000
08000000080800080000000800080008080000007F7F
7F00BFBFBF000000FF0000FF000000FFFF00FF000000
0FF00FF00FFFF0000FFFFFF0033333333333333333333
33333FFFFFFFFF3333330000000000333333777777777
773333330FFFFFFFF03333337F3333333733333330FF
FFFFFF03333337F3FF3FFF73333330F00F000F03333
F37F7737773733330330FFFFFFFF03337FF7F3F3FF3F
73339030F0800F0F033377F7F7377373733399900FFFF
FFFF03FF7777F3FF3FFF70999990F00F00000777777
7F77377777709999990FFF0FF0377777777FF37F37309
99999908F0F033777777777337F73309999990FFF003
3377777777FFF773330999990000003333777777777
773333339903333333333337773333333333333339033
333333333337733333333333333033333333333333337
333333333333}
    NumGlyphs = 2
  end
  object PropertyBtn: TSpeedButton
    Left = 356
    Top = 2
    Width = 28
    Height = 27
    Hint = 'Property'
    Enabled = False
    Glyph.Data = {
76010000424D760100000000000076000000028000000
20000000100000000100400000000000001000012 0B
0000120B000010000000000000000000000000080000
08000000080800080000000800080008080000007F7F
7F00BFBFBF000000FF0000FF000000FFFF00FF00000
0FF00FF00FFFF0000FFFFFF0033333330033333333
33333773FF333333333330F0033333333333373773FF
333333330FFFF003333333337F333773FF3333330FFF
FFF0033333333733FF33773333333 0FF00FFFF8033333
```

```
7F3773F3337333330FFFF0FFFF03FFFF7FFF3733F37
00000000FFFF0FF0777777773FF373370000000000FF
FFF07FFFFFF377FFF3370CCCCC000000FF03777777
3337773F7300CCC000003300307F77733337F377370
00C0000003333307F373333F7F333370000007B70333
3307FFFF337F7F33337099900BBB03333307777F377
77FF33370999007B700333037777F3373773FF730999
0000003000337777FFFFF7F777330000000000333333
7777777777333333}
    NumGlyphs = 2
    OnClick = PropertyBtnClick
  end
  object IDTWSBtn: TSpeedButton
    Left = 314
    Top = 2
    Width = 28
    Height = 27
    Hint = 'IDTWS'
    Enabled = False
    Glyph.Data = {
76010000424D760100000000000076000000280000000
2000000010000000010004000000000000010000120B
0000120B000010000000000000000000000008000
00800000008080008000000080008000808000007F7F
7F00BFBFBF000000FF0000FF000000FFFF00FF00000
0FF00FF00FFFF0000FFFFFF003FFFFFFFFFFFFFF3
3333333333FFFFF3FFFFFFFFF00000F333333333377
777F33FFFFFFFF09990F33333333337F337F333FFFF
FFF09990F33333333337F337F3333FFFFFF09990F33
333333337FFF7F33333FFFFF00000F33333333337777
73333333FFFFFFFFFF3FFFFF3333333F330000033FF
FFF0FFF77777F3333337FF30EEE0333FFF000FF7F3
37FFF333777FF0EEE00033F00000F7F33777F377777
7F0EEE0E03300000007FFF7F7FF777777700000E000
33000FF777773777F3777F3330EEE0E033000FF337F
FF7F7F3777F33300000E033000FF337777737F377733
33330EEE030003FF33337FFF7777733333330000033
3333F3333777773333333}
    NumGlyphs = 2
    OnClick = IDTWSBtnClick
  end
  object QuantizeBtn: TSpeedButton
    Left = 220
    Top = 2
    Width = 28
    Height = 27
    Hint = 'Quantizing'
    Enabled = False
    Glyph.Data = {
76010000424D760100000000000076000000280000000
2000000010000000010004000000000000010000120B
0000120B000010000000000000000000000008000
00800000008080008000000080008000808000007F7F
7F00BFBFBF000000FF0000FF000000FFFF00FF00000
0FF00FF00FFFF0000FFFFFF003373337337337333
7F3F7F3F7F3F7F337373737373733F7F7F7F7F7F7
F7F77000000000000077777777777777777330333333
C333333337FFF3337F3333F370993333C3333993777
73F337F33377330339333C3339333F7FF7FFF7FFF7F
F770777977C779777777777777777777330333933C3
39333337F3373F7F37333370333393C393333377F333
737F733333033333999333333F7FFFFF777FFFFF77
0777777C77777777777777777777330333333C3333
33337F333337F33333370333333C333333377F333337
F33333330333333C3333333F7FFFFFF7FFFFFF77077
77777777777777777777777773333333333333333333
33333333333333}
    NumGlyphs = 2
    OnClick = QuantizeBtnClick
  end
  object CodingBtn: TSpeedButton
    Left = 248
    Top = 2
    Width = 28
    Height = 27
    Hint = 'Coding'
    Enabled = False
    Glyph.Data = {
76010000424D760100000000000076000000280000000
2000000010000000010004000000000000010000130B
0000130B000010000000000000000000000008000
00800000008080008000000080008000808000007F7F
7F00BFBFBF000000FF0000FF000000FFFF00FF00000
0FF00FF00FFFF0000FFFFFF003333333333303333333
333333337FF333333333333903333333333333377FF
33333333339903333FFFFFFFFF777FF3000000999
99990337777777777777FF0FFFF099999999037F333
7777777777F0FFFF099999999907F3FF777777777770
F00F099999999037F773777777777730FFFF09999999
0337F3FF777777777330F00FFFFF09903337F7733333
77773330FFFFFFFF09033337F3FF3FFF77733330F00
F0000000333337F773777777333330FFFF0FF03333333
7F3FF7F3733333330F08F0F0333333337F7737F73333
33330FFFF003333333337FFFF7733333333300000033
3333333777777333333333333333333333333333333333
33333333333}
    NumGlyphs = 2
    OnClick = CodingBtnClick
  end
  object ZoomInBtn: TSpeedButton
    Left = 120
    Top = 2
    Width = 28
    Height = 27
    Hint = 'Zoom In'
    Enabled = False
    Glyph.Data = {
76010000424D760100000000000076000000280000000
2000000010000000010004000000000000010000130B
0000130B000010000000000000000000000008000
00800000008080008000000080008000808000007F7F
7F00BFBFBF000000FF0000FF000000FFFF00FF00000
0FF00FF00FFFF0000FFFFFF003333333333303333333
33333333F7F333333333333300033333333333F7
773333333333300033333333333F77733333333333
3000333333333333F77733333333333300033333333F
FF3F77733333700073B703333333F7773F777333333
07777700B33333377333777733333307F8F8F703333
3337F333F337F3333377F8F9F8F7733333373337F337
3F3333078F898F870333337F33F7FFF37F333307F999
99F70333337F377777337F3333078F898F8703333373
F337F33373333778F9F8F773333337F3373337F333
33307F8F8F70333333373FF333F73333333077777703
33333333773FF77333333333370007333333333333377
7333333333}
    NumGlyphs = 2
    OnClick = ZoomIn1Click
  end
  object ZoomOutBtn: TSpeedButton
    Left = 148
    Top = 2
    Width = 28
    Height = 27
    Hint = 'Zoom Out'
    Enabled = False
    Glyph.Data = {
76010000424D760100000000000076000000280000000
2000000010000000010004000000000000010000130B
0000130B000010000000000000000000000008000
00800000008080008000000080008000808000007F7F
7F00BFBFBF000000FF0000FF000000FFFF00FF00000
0FF00FF00FFFF0000FFFFFF003333333333303333333
33333333F7F333333333333300033333333333F7
773333333333300033333333333F77733333333333
3000333333333333F77733333333333300033333333F
FF3F77733333700073B703333333F7773F777333333
07777700B33333377333777733333307F8F8F703333
3337F333F337F3333377F8F9F8F7733333373337F337
3F3333078F898F870333337F33F7FFF37F333307F999
99F70333337F377777337F3333078F898F8703333373
F337F33373333778F9F8F773333337F3373337F333
33307F8F8F70333333373FF333F73333333077777703
33333333773FF77333333333370007333333333333377
7333333333}
```

```
3F3333333733333778F8F8F8773333337F3333337F33
3333078F8F870333333373FF333F7333333330777770
333333333773FF77333333333370007333333333337
77333333333}
    NumGlyphs = 2
    OnClick = ZoomOut1Click
  end
 end
 object StatusBar: TStatusBar
  Left = 0
  Top = 324
  Width = 560
  Height = 21
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clBlack
  Font.Height = -12
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  Panels = <
   item
    Width = 70
   end
   item
    Width = 50
   end>
  ParentFont = False
  SimplePanel = False
 end
 object MainMenu1: TMainMenu
  Left = 51
  Top = 276
  object File1: TMenuItem
   Caption = '&File'
   Hint = 'File related commands'
   object FileOpenItem: TMenuItem
    Caption = '&Open'
    Hint = 'Open an existing file'
    OnClick = FileOpenItemClick
   end
   object FileCloseItem: TMenuItem
    Caption = '&Close'
    Hint = 'Close current file'
    OnClick = FileCloseItemClick
   end
   object FileSaveItem: TMenuItem
    Caption = '&Save'
    Enabled = False
    Hint = 'Save current file'
    OnClick = FileSaveItemClick
   end
   object FileSaveAsItem: TMenuItem
    Caption = 'Save &As...'
    Enabled = False
    Hint = 'Save current file as...'
    OnClick = FileSaveAsItemClick
   end
   object N1: TMenuItem
    Caption = '-'
   end
   object FileExitItem: TMenuItem
    Caption = 'E&xit'
    Hint = 'Exit the application'
    OnClick = FileExitItemClick
   end
  end
  object Edit1: TMenuItem
   Caption = '&Edit'
   object CopyItem: TMenuItem
    Caption = '&Copy'
    Enabled = False
    Hint = 'Copy to clipboard'
    OnClick = CopyItemClick
   end
  end
  object View1: TMenuItem
   Caption = '&View'
   object Zoom111: TMenuItem
    Caption = 'Original size'
    Enabled = False
    OnClick = Zoom111Click
   end
   object ZoomIn1: TMenuItem
    Caption = 'Zoom &In'
    Enabled = False
    OnClick = ZoomIn1Click
   end
   object ZoomOut1: TMenuItem
    Caption = 'Zoom &Out'
    Enabled = False
    OnClick = ZoomOut1Click
   end
  end
  object Process1: TMenuItem
   Caption = '&Process'
   object DTWS1: TMenuItem
    Caption = 'DTWS'
    Enabled = False
    OnClick = DTWS1Click
   end
   object Quantize1: TMenuItem
    Caption = 'Quantize'
    Enabled = False
   end
   object Coding1: TMenuItem
    Caption = 'Coding'
    Enabled = False
   end
   object N2: TMenuItem
    Caption = '-'
   end
   object Decoding1: TMenuItem
    Caption = 'Decoding'
    Enabled = False
   end
   object IDTWS1: TMenuItem
    Caption = 'IDTWS'
    Enabled = False
    OnClick = IDTWSBtnClick
   end
  end
  object Window1: TMenuItem
   Caption = '&Window'
   Hint = 'Window related commands such as Tile and
Cascade'
   object WindowCascadeItem: TMenuItem
    Caption = '&Cascade'
    Hint = 'Arrange windows to overlap'
    OnClick = WindowCascadeItemClick
   end
   object WindowTileItem: TMenuItem
    Caption = '&Tile'
    Hint = 'Arrange windows without overlap'
    OnClick = WindowTileItemClick
   end
   object WindowArrangeItem: TMenuItem
    Caption = '&Arrange Icons'
    Hint = 'Arrange window icons at bottom of main
window'
    OnClick = WindowArrangeItemClick
   end
   object WindowMinimizeItem: TMenuItem
    Caption = '&Minimize All'
    Hint = 'Minimize all windows'
    OnClick = WindowMinimizeItemClick
   end
  end
  object Help1: TMenuItem
   Caption = '&Help'
   Hint = 'Help topics'
   object HelpAboutItem: TMenuItem
```

```
      Caption = '&About'
      OnClick = HelpAboutItemClick
    end
  end
 end
 object OpenDialog: TOpenPictureDialog
  DefaultExt = '.bmp'
  Filter = 'Bitmaps (*.bmp)|*.bmp'
  Left = 92
  Top = 276
 end
 object SaveDialog: TSavePictureDialog
  DefaultExt = '.bmp'
  Filter = 'Bitmaps (*.bmp)|*.bmp'
  Options = [ofOverwritePrompt, ofHideReadOnly]
  Left = 132
  Top = 276
 end
end
```

โปรแกรม Main.pas

```
unit Main;

interface

uses Windows, SysUtils, Classes, Graphics, Forms,
Controls, Menus,
 StdCtrls, Dialogs, Buttons, Messages, ExtCtrls,
ComCtrls, ExtDlgs;


type
 TData = array [0..9000000] of Single;
 TMainForm = class(TForm)
  MainMenu1: TMainMenu;
  File1: TMenuItem;
  FileOpenItem: TMenuItem;
  FileCloseItem: TMenuItem;
  Window1: TMenuItem;
  Help1: TMenuItem;
  N1: TMenuItem;
  FileExitItem: TMenuItem;
  WindowCascadeItem: TMenuItem;
  WindowTileItem: TMenuItem;
  WindowArrangeItem: TMenuItem;
  HelpAboutItem: TMenuItem;
  FileSaveItem: TMenuItem;
  FileSaveAsItem: TMenuItem;
  Edit1: TMenuItem;
  CopyItem: TMenuItem;
  WindowMinimizeItem: TMenuItem;
  SpeedPanel: TPanel;
  OpenBtn: TSpeedButton;
  SaveBtn: TSpeedButton;
  CopyBtn: TSpeedButton;
  ExitBtn: TSpeedButton;
  StatusBar: TStatusBar;
  View1: TMenuItem;
  Process1: TMenuItem;
  Zoom111: TMenuItem;
  DTWS1: TMenuItem;
  IDTWS1: TMenuItem;
  Quantize1: TMenuItem;
  Coding1: TMenuItem;
  N2: TMenuItem;
  DTWSBtn: TSpeedButton;
  IDTWSBtn: TSpeedButton;
  CodingBtn: TSpeedButton;
  DecodingBtn: TSpeedButton;
  PropertyBtn: TSpeedButton;
  QuantizeBtn: TSpeedButton;
  Decoding1: TMenuItem;
  ZoomIn1: TMenuItem;
  ZoomOut1: TMenuItem;
  OpenDialog: TOpenPictureDialog;
  SaveDialog: TSavePictureDialog;
  ZoomInBtn: TSpeedButton;
  ZoomOutBtn: TSpeedButton;
  procedure FormCreate(Sender: TObject);
  procedure NewMDIChild(Sender: TObject);
  procedure WindowCascadeItemClick(Sender:
TObject);
  procedure UpdateMenuItems(Sender: TObject);
  procedure WindowTileItemClick(Sender: TObject);
  procedure WindowArrangeItemClick(Sender:
TObject);
  procedure FileCloseItemClick(Sender: TObject);
  procedure FileOpenItemClick(Sender: TObject);
  procedure FileExitItemClick(Sender: TObject);
  procedure FileSaveItemClick(Sender: TObject);
  procedure FileSaveAsItemClick(Sender: TObject);
  procedure CopyItemClick(Sender: TObject);
  procedure WindowMinimizeItemClick(Sender:
TObject);
  procedure FormDestroy(Sender: TObject);
  procedure HelpAboutItemClick(Sender: TObject);
  procedure Zoom111Click(Sender: TObject);
  procedure DTWS1Click(Sender: TObject);
  procedure IDTWSBtnClick(Sender: TObject);
  procedure ZoomIn1Click(Sender: TObject);
  procedure ZoomOut1Click(Sender: TObject);
  procedure QuantizeBtnClick(Sender: TObject);
  procedure CodingBtnClick(Sender: TObject);
  procedure PropertyBtnClick(Sender: TObject);
 private
  { Private declarations }
  procedure CreateMDIChild(const Name: string);
  procedure ShowHint(Sender: TObject);
  procedure DTWS2(var Input: array of Single; const
xx, yy, xz, yz: Integer);
  procedure IDTWS2(var Input: array of Single; const
xx, yy, xz, yz: Integer);
 public
  res : TBitmap;
  { Public declarations }
 end;

var
 MainForm: TMainForm;
 ChildCount : Integer;

implementation

{$R *.DFM}

uses ChildWin, About, Quantize, compare, Coding;

procedure TMainForm.FormCreate(Sender: TObject);
begin
 Application.OnHint := ShowHint;
 Screen.OnActiveFormChange := UpdateMenuItems;
 ChildCount := 1;
end;

procedure TMainForm.ShowHint(Sender: TObject);
begin
 StatusBar.SimpleText := Application.Hint;
end;

procedure TMainForm.CreateMDIChild(const Name:
string);
var
 Child: TMDIChild;
begin
 { create a new MDI child window }
 Child := TMDIChild.Create(Application);
 Child.Caption := Name;
end;
```

```
procedure TMainForm.NewMDIChild(Sender: TObject);
begin
  CreateMDIChild('NONAME' + IntToStr(ChildCount));
  Inc(ChildCount);
end;

procedure TMainForm.FileOpenItemClick(Sender:
TObject);
begin
  if OpenDialog.Execute then
    begin
      CreateMDIChild(OpenDialog.FileName);
      with ActiveMDIChild as TMDIChild do
      begin
        LoadData(OpenDialog.FileName);
        end;
      end;
end;

procedure TMainForm.FileCloseItemClick(Sender:
TObject);
begin
  if ActiveMDIChild <> nil then
    ActiveMDIChild.Close;
end;

procedure TMainForm.FileSaveItemClick(Sender:
TObject);
begin
  { save current file (ActiveMDIChild points to the
window) }
  if ActiveMDIChild <> nil then
    if SaveDialog.Execute then
      with ActiveMDIChild as TMDIChild do
        SaveData(SaveDialog.FileName);
end;

procedure TMainForm.FileSaveAsItemClick(Sender:
TObject);
begin
  { save current file under new name }
end;

procedure TMainForm.FileExitItemClick(Sender:
TObject);
begin
  Close;
end;

procedure TMainForm.CopyItemClick(Sender: TObject);
begin
  {copy selection to clipboard}
end;

procedure TMainForm.WindowCascadeItemClick
(Sender: TObject);
begin
  Cascade;
end;

procedure TMainForm.WindowTileItemClick(Sender:
TObject);
begin
  Tile;
end;

procedure TMainForm.WindowArrangeItemClick
(Sender: TObject);
begin
  ArrangeIcons;
end;

procedure TMainForm.WindowMinimizeItemClick
(Sender: TObject);
var
  I : Integer;
begin
  { Must be done backwards through the MDIChildren
array }
  for I := MDIChildCount - 1 downto 0 do
    MDIChildren[I].WindowState := wsMinimized;
end;

procedure TMainForm.UpdateMenuItems(Sender:
TObject);
begin
  FileCloseItem.Enabled := MDIChildCount > 0;
  FileSaveItem.Enabled := MDIChildCount > 0;
  FileSaveAsItem.Enabled := MDIChildCount > 0;
  CopyItem.Enabled := MDIChildCount > 0;
  SaveBtn.Enabled := MDIChildCount > 0;
  CopyBtn.Enabled := MDIChildCount > 0;
  WindowCascadeItem.Enabled := MDIChildCount > 0;
  WindowTileItem.Enabled := MDIChildCount > 0;
  WindowArrangeItem.Enabled := MDIChildCount > 0;
  WindowMinimizeItem.Enabled := MDIChildCount > 0;
  Zoom111.Enabled := MDIChildCount > 0;
  ZoomIn1.Enabled := MDIChildCount > 0;
  ZoomOut1.Enabled := MDIChildCount > 0;
  ZoomInBtn.Enabled := MDIChildCount > 0;
  ZoomOutBtn.Enabled := MDIChildCount > 0;
  DTWSBtn.Enabled := MDIChildCount > 0;
  DTWS1.Enabled := MDIChildCount > 0;
  QuantizeBtn.Enabled := MDIChildCount > 0;
  Quantize1.Enabled := MDIChildCount > 0;
  CodingBtn.Enabled := MDIChildCount > 0;
  Coding1.Enabled := MDIChildCount > 0;
  DecodingBtn.Enabled := MDIChildCount > 0;
  Decoding1.Enabled := MDIChildCount > 0;
  IDTWSBtn.Enabled := MDIChildCount > 0;
  IDTWS1.Enabled := MDIChildCount > 0;
  PropertyBtn.Enabled := MDIChildCount > 1;
end;

procedure TMainForm.FormDestroy(Sender: TObject);
begin
  Screen.OnActiveFormChange := nil;
end;

procedure TMainForm.HelpAboutItemClick(Sender:
TObject);
begin
  AboutBox.ShowModal;
end;

procedure TMainForm.Zoom111Click(Sender: TObject);
begin
  if ActiveMDIChild <> nil then
    with ActiveMDIChild as TMDIChild do
      begin
        if WindowState = wsNormal then
          begin
            ClientHeight := Image1.Picture.Bitmap.Height;
            ClientWidth := Image1.Picture.Bitmap.Width;
            end;
        Image1.Height := Image1.Picture.Bitmap.Height;
        Image1.Width := Image1.Picture.Bitmap.Width;
        end;
end;

procedure TMainForm.DTWS1Click(Sender: TObject);
var
  data : ^TData;
  m, n : Integer;
  name : String[100];
begin
  if ActiveMDIChild <> nil then
    begin
      data := nil;
      try
```

```
      with ActiveMDIChild as TMDIChild do
       begin
        res := TBitmap.Create;
        res.Height := Image1.Picture.Bitmap.Height;
        res.Width := Image1.Picture.Bitmap.Width;
        res.PixelFormat := pf24bit;

        GetMem(data,
sizeof(Single)*res.Width*res.Height);
        for m := 0 to res.Height-1 do
         for n := 0 to res.Width-1 do
          data^[(m*res.Width)+n] :=
Image1.Picture.Bitmap.Canvas.Pixels[n,m] and $FF;
        end;

      DTWS2(data^, res.Width, res.Height, res.Width,
res.Height);

      for m := 0 to res.Height-1 do
       for n := 0 to res.Width-1 do
        res.canvas.Pixels[n,m] := abs(round(data^
[(m*res.Width)+n])+$800000);

      NewMDIChild(Sender);
      if ActiveMDIChild <> nil then
       with ActiveMDIChild as TMDIChild do
        begin
         Image1.Picture.Bitmap.Height := res.Height;
         Image1.Picture.Bitmap.Width := res.Width;
         Zoom111Click(Sender);
         Image1.Picture.Bitmap.PixelFormat := pf24bit;
         Image1.Picture.Bitmap.IgnorePalette := True;
         Image1.Picture.Bitmap.Canvas.Draw(0,0,res);
         end;
    finally
     FreeMem(data);
     res.Free;
     end;
    end;
end;

procedure TMainForm.DTWS2(var Input: array of
Single; const xx, yy, xz, yz : Integer);
var
 m, n, j, k, a : Integer;
 dat : ^TData;
const
 G : array [0..3] of Single = ( -0.1294, 0.2241, 0.8365,
0.4830);
 H : array [0..3] of Single = ( -0.4830, 0.8365,-0.2241,-
0.1294);
begin
 dat := nil;
 if (xz mod 2)+(yz mod 2) = 0 then
  begin
  try
   GetMem(dat, sizeof(Single)*(xz+1)*(yz+1));

   j := 0;
   for n := 0 to yz-1 do
    for m := 0 to xz-1 do
     begin
      dat^[j] := Input[(n*xx)+m];
      inc(j);
     end;

   a := xz div 2;
   for m := 0 to yz-1 do
    for n := 0 to a-1 do
     begin
      Input[(m*xx)+a+n] := 0;
      Input[(m*xx)+n] := 0;
       for j := 0 to 1 do
        begin
         k := 2*(n+j);
         if k >= xz then
          k := k - xz;
         Input[(m*xx)+a+n] := Input[(m*xx)+a+n]+
(dat^[(m*xz)+k]*H[2*j])+(dat^[(m*xz)+k+1]*H[(2*j)+1]);
         Input[(m*xx)+n] := Input[(m*xx)+n]+(dat^
[(m*xz)+k]*G[2*j])+(dat^[(m*xz)+k+1]*G[(2*j)+1]);
        end;

     end;

   j := 0;
   for n := 0 to yz-1 do
    for m := 0 to xz-1 do
     begin
      dat^[j] := Input[(n*xx)+m];
      inc(j);
     end;

   a := yz div 2;
   for m := 0 to xz-1 do
    for n := 0 to a-1 do
     begin
      Input[((n+a)*xx)+m] := 0;
      Input[(n*xx)+m] := 0;
       for j := 0 to 1 do
        begin
         k := 2*(n+j);
         if k >= yz then
          k := k - yz;
         Input[((n+a)*xx)+m] := Input[((n+a)*xx)+m]+
(dat^[(k*xz)+m]*H[2*j])+(dat^[((k+1)*xz)+m]*H[(2*j)+1]);
         Input[(n*xx)+m] := Input[(n*xx)+m]+(dat^
[(k*xz)+m]*G[2*j])+(dat^[((k+1)*xz)+m]*G[(2*j)+1]);
        end;
     end;

    finally
     FreeMem(dat);
    end;
   DTWS2(Input,xx,yy,xz div 2, yz div 2);
   end;
end;


procedure TMainForm.IDTWSBtnClick(Sender:
TObject);
var
 data : ^TData;
 m, n : Integer;
 name : String[100];

 pal: PLogPalette;
 hpal: HPALETTE;
 i: Integer;
begin
 if ActiveMDIChild <> nil then
  begin
   data := nil;
   try
    with ActiveMDIChild as TMDIChild do
     begin
      res := TBitmap.Create;
      res.Height := Image1.Picture.Bitmap.Height;
      res.Width := Image1.Picture.Bitmap.Width;
      res.PixelFormat := pf24bit;

      GetMem(data,
sizeof(Single)*res.Width*res.Height);
      for m := 0 to res.Height-1 do
       for n := 0 to res.Width-1 do
        data^[(m*res.Width)+n] :=
Image1.Picture.Bitmap.Canvas.Pixels[n,m]-$800000;
      end;
```

```
      IDTWS2(data^, res.Width, res.Height, res.Width,
res.Height);
      for m := 0 to res.Height-1 do
       for n := 0 to res.Width-1 do
        begin
         i := round(data^[(m*res.Width)+n]);
         if i > 255 then
          i := 255
          else
           if i < 0 then
            i := 0;
         i := i or (i shl 8) or (i shl 16);

         res.canvas.Pixels[n,m] := i;
         end;

      NewMDIChild(Sender);
      if ActiveMDIChild <> nil then
       with ActiveMDIChild as TMDIChild do
        begin
         Image1.Picture.Bitmap.Height := res.Height;
         Image1.Picture.Bitmap.Width := res.Width;
         Image1.Picture.Bitmap.PixelFormat := pf24bit;
         Zoom111Click(Sender);
         end;

     finally
      FreeMem(data);
      res.Free;
      end;
     end;
end;

procedure TMainForm.IDTWS2(var Input: array of
Single; const xx, yy, xz, yz: Integer);
var
 m, n, j, k, a, b : Integer;
 dat : ^TData;
 tmp : Single;
const
 G : array [0..3] of Single = ( 0.4830, 0.8365, 0.2241,-
0.1294);
 H : array [0..3] of Single = ( -0.1294,-0.2241, 0.8365,-
0.4830);
begin
 dat := nil;
 if (xz mod 2)+(yz mod 2) = 0 then
  begin
   IDTWS2(Input,xx,yy,xz div 2,yz div 2);
   try
    GetMem(dat, sizeof(Single)*(xz+1)*(yz+1));

    j := 0;
    for n := 0 to yz-1 do
     for m := 0 to xz-1 do
      begin
       dat^[j] := Input[(n*xx)+m];
       inc(j);
       end;

    a := yz div 2;
    for m := 0 to xz-1 do
     for n := 0 to a-1 do
      begin
       b := (2*n)+2;
       if b >= yz then
        b := b - yz;
       Input[(b*xx)+m] := 0;
       Input[((b+1)*xx)+m] := 0;
        for j := 0 to 1 do
         begin
          k := n+j;
          if k >= a then
          k := k - a;
          Input[((b+1)*xx)+m] := Input[((b+1)*xx)+m]+
(dat^[(k*xz)+m]*G[2*j])+(dat^[((k+a)*xz)+m]*H[2*j]);
          Input[(b*xx)+m] := Input[(b*xx)+m]+(dat^
[(k*xz)+m]*G[(2*j)+1])+(dat^[((k+a)*xz)+m]*H[(2*j)+1]);
          end;

        end;

    j := 0;
    for n := 0 to yz-1 do
     for m := 0 to xz-1 do
      begin
       dat^[j] := Input[(n*xx)+m];
       inc(j);
       end;

    a := xz div 2;
    for m := 0 to yz-1 do
     for n := 0 to a-1 do
      begin
       b := (2*n)+2;
       if b >= xz then
        b := b - xz;
       Input[(m*xx)+b] := 0;
       Input[(m*xx)+b+1] := 0;
        for j := 0 to 1 do
         begin
          k := n+j;
          if k >= a then
           k := k - a;
          Input[(m*xx)+b+1] := Input[(m*xx)+b+1]+
(dat^[(m*xz)+k]*G[(2*j)])+(dat^[(m*xz)+k+a]*H[(2*j)]);
          Input[(m*xx)+b] := Input[(m*xx)+b]+(dat^
[(m*xz)+k]*G[(2*j)+1])+(dat^[(m*xz)+k+a]*H[(2*j)+1]);
          end;
        end;
     finally
      FreeMem(dat);
     end;
    end;
end;

procedure TMainForm.ZoomIn1Click(Sender: TObject);
begin
 if ActiveMDIChild <> nil then
  with ActiveMDIChild as TMDIChild do
   begin
    Image1.Height := 2*Image1.Height;
    Image1.Width := 2*Image1.Width;
    end;
end;

procedure TMainForm.ZoomOut1Click(Sender:
TObject);
begin
 if ActiveMDIChild <> nil then
  with ActiveMDIChild as TMDIChild do
   begin
    Image1.Height := Image1.Height div 2;
    Image1.Width := Image1.Width div 2;
    end;
end;

procedure TMainForm.QuantizeBtnClick(Sender:
TObject);
var
 m, n : Integer;
 name : String[100];
 a, b : Longint;
begin
 with ActiveMDIChild as TMDIChild do
  begin
   res := TBitmap.Create;
   res.Height := Image1.Picture.Bitmap.Height;
   res.Width := Image1.Picture.Bitmap.Width;
```

```
    res.PixelFormat := pf24bit;
    end;

  if QuantizeDlg.Showmodal = mrOk then
    begin
     NewMDIChild(Sender);
     if ActiveMDIChild <> nil then
       with ActiveMDIChild as TMDIChild do
         begin
          Image1.Picture.Bitmap.Height := res.Height;
          Image1.Picture.Bitmap.Width := res.Width;
          Image1.Picture.Bitmap.PixelFormat := pf24bit;
          Zoom111Click(Sender);
          Image1.Picture.Bitmap.Canvas.Draw(0,0,res);
          end;
     end;
    res.Free;
end;

procedure TMainForm.PropertyBtnClick(Sender:
TObject);
begin
  CompareDlg.ShowModal;
end;

procedure TMainForm.CodingBtnClick(Sender:
TObject);
begin
  CodingDlg.ShowModal;
end;

end.
```

Childwin.pas , Childwin.dfm

เป็นหน้าต่างลูก (Child window) สำหรับการแสดงภาพ หรือค่าสัมประสิทธิ์เวฟเล็ต และยังควบคุมการอ่านและเขียน ไฟล์ รูปภาพ

รูปภาพที่ใช้กับโปรแกรมนี้ จะอยู่ในรูปแบบของไฟล์ BMP ในกรณีที่ใช้เป็นภาพต้นแบบ ค่าระดับความสว่างของ สีแดง จะถูกนำมาใช้งาน แต่ในกรณีที่ใช้เป็นค่าสัมประสิทธิ์เวฟเล็ตนั้นรูปภาพจะถูกมองว่าเป็นที่เก็บข้อมูลขนาด 24 bit ข้อมูลทั้งหมด จะถูกนำมาใช้งาน

โปรแกรม Childwin.dfm

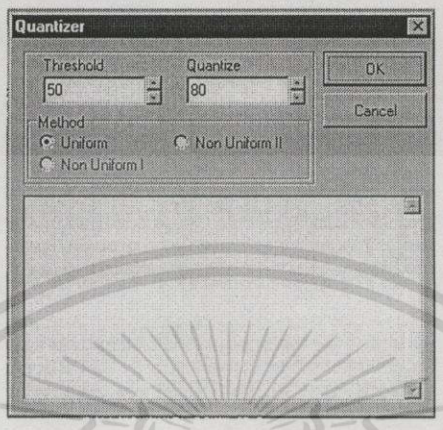
```
object MDIChild: TMDIChild
  Left = 358
  Top = 144
  Width = 264
  Height = 283
  HorzScrollBar.Tracking = True
  VertScrollBar.Tracking = True
  Caption = 'MDI Child'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -13
  Font.Name = 'System'
  Font.Style = []
  FormStyle = fsMDIChild
  Position = poDefault
  Visible = True
  OnClose = FormClose
  PixelsPerInch = 96
  TextHeight = 16
  object Image1: TImage
    Left = 0
    Top = 0
    Width = 256
    Height = 256
    AutoSize = True
    Center = True
    Stretch = True
    OnMouseMove = Image1MouseMove
  end
end
```

โปรแกรม Childwin.pas

```
unit Childwin;

interface

uses Windows, Classes, Graphics, Forms, Controls,
ExtCtrls, StdCtrls,
  SysUtils, ComCtrls;

type
  TMDIChild = class(TForm)
    Image1: TImage;
    procedure FormClose(Sender: TObject; var Action:
TCloseAction);
    procedure Image1MouseMove(Sender: TObject;
Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure LoadData(const FileName: String); virtual;
    procedure SaveData(const FileName: String); virtual;
  end;

implementation

uses Main;

{$R *.DFM}

procedure TMDIChild.FormClose(Sender: TObject; var
Action: TCloseAction);
begin
  Action := caFree;
end;

procedure TMDIChild.LoadData(const FileName:
String);
begin
  Image1.Picture.LoadFromFile(FileName);
  Image1.Height := Image1.Picture.Bitmap.Height;
  Image1.Width := Image1.Picture.Bitmap.Width;
  ClientHeight := Image1.Picture.Bitmap.Height;
  ClientWidth := Image1.Picture.Bitmap.Width;

end;

procedure TMDIChild.SaveData(const FileName:
String);
begin
  Image1.Picture.SaveToFile(FileName);
end;

procedure TMDIChild.Image1MouseMove(Sender:
TObject; Shift: TShiftState; X, Y: Integer);
var
  xx, yy : Integer;
begin
  if (Image1.Height = Image1.Picture.Bitmap.Height) and
    (Image1.Width = Image1.Picture.Bitmap.Width) then
    begin
      xx := X;
      yy := Y;
    end
    else
      begin
        yy := (Y * Image1.Picture.Bitmap.Height) div
Image1.Height;
        xx := (X * Image1.Picture.Bitmap.Width) div
Image1.Width;
      end;

  MainForm.StatusBar.Panels[0].Text := format('(%d,
%d)',[xx,yy]);
  MainForm.StatusBar.Panels[1].Text := format('Color :
%x',[Image1.Picture.Bitmap.Canvas.Pixels[xx,yy]]);
end;

end.
```

Quantize.pas , Quantize.dfm

เป็นส่วนของการทำ Quantize ของค่าสัมประสิทธิ์ เวฟเล็ต โดยจะสามารถตั้งระดับ Threshold และระดับของการ Quantize ได้



รูปแสดงหน้าต่างของการ Quantize

โปรแกรม Quantize.dfm

```
object QuantizeDlg: TQuantizeDlg
  Left = 214
  Top = 234
  BorderStyle = bsDialog
  Caption = 'Quantizer'
  ClientHeight = 290
  ClientWidth = 327
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  Position = poScreenCenter
  PixelsPerInch = 96
  TextHeight = 14
  object Bevel1: TBevel
    Left = 9
    Top = 9
    Width = 228
    Height = 104
    Shape = bsFrame
  end
  object Label1: TLabel
    Left = 24
    Top = 12
    Width = 47
    Height = 14
    Caption = 'Threshold'
  end
  object Label2: TLabel
    Left = 136
    Top = 12
    Width = 42
    Height = 14
    Caption = 'Quantize'
  end
  object OKBtn: TButton
    Left = 243
    Top = 9
    Width = 81
    Height = 27
    Caption = 'OK'
    Default = True
    ModalResult = 1
    TabOrder = 0
    OnClick = OKBtnClick
  end
  object CancelBtn: TButton
    Left = 243
    Top = 41
    Width = 81
    Height = 27
    Cancel = True
    Caption = 'Cancel'
    ModalResult = 2
    TabOrder = 1
  end
  object UpDown1: TUpDown
    Left = 105
    Top = 28
    Width = 12
    Height = 22
    Associate = TholdEdit
    Min = 1
    Position = 10
    TabOrder = 2
    Wrap = False
  end
  object UpDown2: TUpDown
    Left = 217
    Top = 28
    Width = 12
    Height = 22
    Associate = QuanEdit
    Min = 5
    Position = 5
    TabOrder = 3
    Wrap = False
  end
  object TholdEdit: TMaskEdit
    Left = 24
    Top = 28
    Width = 81
    Height = 22
    TabOrder = 4
    Text = '10'
  end
```

```
 object QuanEdit: TMaskEdit
  Left = 136
  Top = 28
  Width = 81
  Height = 22
  TabOrder = 5
  Text = '5'
 end
 object MethodGrp: TRadioGroup
  Left = 12
  Top = 56
  Width = 221
  Height = 53
  Caption = 'Method'
  Columns = 2
  ItemIndex = 0
  Items.Strings = (
   'Uniform'
   'Non Uniform I'
   'Non Uniform II')
  TabOrder = 6
 end
 object Memo1: TMemo
  Left = 8
  Top = 120
  Width = 313
  Height = 161
  ScrollBars = ssVertical
  TabOrder = 7
 end
end
```

โปรแกรม Quantize.pas

```
unit Quantize;

interface

uses Windows, SysUtils, Classes, Graphics, Forms,
Controls, StdCtrls,
 Buttons, ExtCtrls, Mask, ComCtrls;

type
 TQuantizeDlg = class(TForm)
  OKBtn: TButton;
  CancelBtn: TButton;
  Bevel1: TBevel;
  UpDown1: TUpDown;
  Label1: TLabel;
  UpDown2: TUpDown;
  Label2: TLabel;
  TholdEdit: TMaskEdit;
  QuanEdit: TMaskEdit;
  MethodGrp: TRadioGroup;
  Memo1: TMemo;
  procedure OKBtnClick(Sender: TObject);
 private
  procedure UniformQuan(Sender: TObject);
  procedure NonUniformQuan(Sender: TObject);
  procedure NonUniform2Quan(Sender: TObject);
  { Private declarations }
 public
  { Public declarations }
 end;

var
 QuantizeDlg: TQuantizeDlg;

implementation

uses  Main, ChildWin;
{$R *.DFM}

type
 TArray = array[0..3000000] of Longint;
 PArray = ^Tarray;

procedure TQuantizeDlg.OKBtnClick(Sender:
TObject);
begin
 OKBtn.Enabled := False;
 if MethodGrp.ItemIndex = 0 then
  UniformQuan(Sender)
  else
   if MethodGrp.ItemIndex = 1 then
    NonUniformQuan(Sender)
    else
     NonUniform2Quan(Sender);

 OKBtn.Enabled := True;
end;

procedure TQuantizeDlg.UniformQuan(Sender:
TObject);
var
 m, n : Integer;
 a, b : Longint;
begin
 with MainForm.ActiveMDIChild as TMDIChild do
  begin
   a := $800000 - StrToInt(TholdEdit.Text);
   b := $800000 + StrToInt(TholdEdit.Text);
   for m := 0 to MainForm.res.Height-1 do
    for n := 0 to MainForm.res.Width-1 do
     if (Image1.Picture.Bitmap.Canvas.Pixels[n,m] >
a) and  (Image1.Picture.Bitmap.Canvas.Pixels[n,m] <
b) then
      Mainform.res.Canvas.Pixels[n,m] := $800000
      else
       MainForm.res.Canvas.Pixels[n,m] :=
(StrToInt(QuanEdit.Text)*
Round((Image1.Picture.Bitmap.Canvas.Pixels[n,m]-
$800000) / StrToInt(QuanEdit.Text)))+$800000;
  end;
end;

procedure TQuantizeDlg.NonUniformQuan(Sender:
TObject);

 procedure QSort(var A : PArray; sz : Longint);

  procedure QuickSort(var A: PArray; iLo, iHi:
Longint);
  var
   Lo, Hi, Mid, T :  Longint;
  begin
   Lo := iLo;
   Hi := iHi;
   Mid := A^[(Lo + Hi) div 2];
   repeat
    while A^[Lo] < Mid do Inc(Lo);
    while A^[Hi] > Mid do Dec(Hi);
     if Lo <= Hi then
      begin
       T := A^[Lo];
       A^[Lo] := A^[Hi];
       A^[Hi] := T;
       Inc(Lo);
       Dec(Hi);
      end;
   until Lo > Hi;

   if Hi > iLo then QuickSort(A, iLo, Hi);
   if Lo < iHi then QuickSort(A, Lo, iHi);
  end;

 begin
  QuickSort(A, 0, sz);
 end;
```

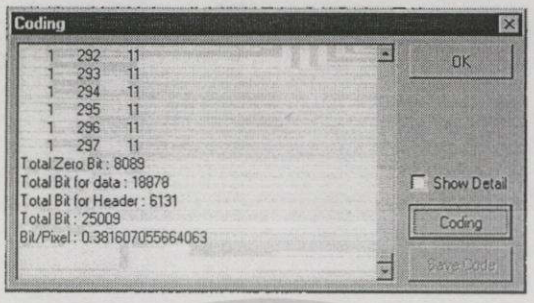
```
var
  m, n : Integer;
  a, b, c, ZeroPoint, mem , aa, bb : Longint;
  aaa, bbb : PArray;
  ValList, CntList, QuaList, LevList, HisList : TList;
  Plong1, Plong2, Plong3, Plong4 : ^Longint;
begin
  ZeroPoint := -1;
  try
    a := $800000 - StrToInt(TholdEdit.Text);
    b := $800000 + StrToInt(TholdEdit.Text);
    with MainForm.ActiveMDIChild as TMDIChild do
      begin
        Getmem(aaa, SizeOf( Longint)*
MainForm.res.Height* MainForm.res.Width);
        Getmem(bbb, SizeOf( Longint)*
MainForm.res.Height* MainForm.res.Width);
        ValList := TList.Create;
        ValList.Clear;
        CntList := TList.Create;
        CntList.CLear;
        QuaList := TList.Create;
        QuaList.Clear;
        LevList := TList.Create;
        LevList.Clear;
        HisList := TList.Create;
        HisList.Clear;

        c := 0;
        for m := 0 to MainForm.res.Height-1 do
          for n := 0 to MainForm.res.Width-1 do
            begin
              if (Image1.Picture.Bitmap.Canvas.Pixels
[n,m] < a) or (Image1.Picture.Bitmap.Canvas.Pixels
[n,m] > b) then
                begin
                  aaa^[c] :=
Image1.Picture.Bitmap.Canvas.Pixels[n,m] -
$800000;
                  bbb^[c] := aaa^[c];
                  end
                else
                  begin
                    aaa^[c] := 0;
                    bbb^[c] := 0;
                    end;
              inc(c);
              end;
        end;
      QSort(aaa,c-1);
      b := -1;
      for a := 0 to c-2 do
        if aaa^[a+1] - aaa^[a] <> 0 then
          begin
            ValList.Add(@aaa[a]);
            New(Plong2);
            Plong2^ := a - b;
            b := a;
            CntList.Add(Plong2);
            if aaa[a] = 0 then
              ZeroPoint := ValList.Count-1;
            end;
      if aaa^[c-1] - aaa^[c-2] <> 0 then
        begin
          ValList.Add(@aaa[c-1]);
          New(Plong2);
          Plong2^ := 1;
          CntList.Add(Plong2);
          end;

      Memo1.Clear;
      for a := 0 to ValList.Count-1 do
        begin
          Plong2 := ValList.Items[a];
          Plong3 := CntList.Items[a];
          Memo1.Lines.Add( Format('%3d %8d %8d
%10d',
            [a,Plong2^,Plong3^, Plong2^*Plong3^]));
          end;
        Memo1.Lines.Add('Total : '+IntToStr
(ValList.Count));

        if ZeroPoint > 0 then
          begin
            Plong1 := ValList.Items[ZeroPoint+1];
            New(Plong2);
            Plong2^ := 0;
            Plong4 := CntList.Items[ZeroPoint];
            New(Plong3);
            Plong3^ := Plong4^;
            QuaList.Add(Plong1);
            LevList.Add(Plong2);
            HisList.Add(Plong3);

            c := 0;
            b := 0;
            a := ZeroPoint;
            while a < ValList.Count-1 do
              begin
                Inc(a);
                Plong1 := ValList.Items[a];
                Plong2 := CntList.Items[a];
                c := c + (Plong1^*Plong2^);
                b := b + Plong2^;
                if (c > Strtoint(QuanEdit.Text)) or (a =
ValList.Count-1) then
                  begin
                    if (b <> Plong2^) and (c > Strtoint
(QuanEdit.Text)) then
                      begin
                        c := c - (Plong1^*Plong2^);
                        b := b - Plong2^;
                        Dec(a);
                        end;
                    New(Plong3);
                    Plong3^ := Round(c / b);
                    LevList.Add(Plong3);
                    QuaList.Add(Plong1);
                    New(Plong4);
                    Plong4^ := b;
                    HisList.Add(Plong4);

                    c := 0;
                    b := 0;
                    end
                  end;
            Plong1 := QuaList.Last;
            aa := Plong1^+1;
            QuaList.Delete(QuaList.Count-1);
            QuaList.Add(@aa);

            mem := LevList.Count;

            c := 0;
            b := 0;
            a := ZeroPoint;
            while a > 0 do
              begin
                Dec(a);
                Plong1 := ValList.Items[a];
                Plong2 := CntList.Items[a];
                c := c + (Plong1^*Plong2^);
                b := b + Plong2^;
                if (-c > Strtoint(QuanEdit.Text)) or (a = 0)
then
                  begin
                    if (b <> Plong2^) and (-c > Strtoint
(QuanEdit.Text)) then
                      begin
                        c := c - (Plong1^*Plong2^);
```

```
            b := b - Plong2^;
            Inc(a);
            end;
          New(Plong3);
          Plong3^ := Round(c / b);
          LevList.Add(Plong3);
          QuaList.Add(Plong1);
          New(Plong4);
          Plong4^ := b;
          HisList.Add(Plong4);
          c := 0;
          b := 0;
          end
        end;
      Plong1 := QuaList.Last;
      bb := Plong1^-1;
      QuaList.Delete(QuaList.Count-1);
      QuaList.Add(@bb);

      c := 0;
      for m := 0 to MainForm.res.Height-1 do
       for n := 0 to MainForm.res.Width-1 do
        begin
         if bbb^[c] = 0 then
          MainForm.res.Canvas.Pixels[n,m] :=
$800000
          else
           if bbb^[c] > 0 then
            begin
             a := 0;
             while a < mem do
              begin
               Inc(a);
               Plong1 := QuaList.Items[a];
               if Plong1^ > bbb^[c] then
                begin
                 Plong2 := LevList.Items[a];
                 MainForm.res.Canvas.Pixels
[n,m] := Plong2^ + $800000;
                 a := LevList.Count;
                end;
              end
             else
              begin
               a := mem-1;
               while a < LevList.Count-1 do
                begin
                 Inc(a);
                 Plong1 := QuaList.Items[a];
                 if Plong1^ < bbb^[c] then
                  begin
                   Plong2 := LevList.Items[a];
                   MainForm.res.Canvas.Pixels
[n,m] := Plong2^ + $800000;
                   a := LevList.Count;
                  end;
                end;
              end;
          Inc(c);
          end;

      end;

    for a := 0 to QuaList.Count-1 do
     begin
      Plong1 := QuaList.Items[a];
      Plong2 := LevList.Items[a];
      Plong3 := HisList.Items[a];

      Memo1.Lines.Add( Format('%3d %8d %8d
%8d',
       [a,Plong1^,Plong2^,Plong3^]));
      end;
    finally
     ValList.Free;
     QuaList.Free;
     for a := 0 to CntList.Count-1 do
      Dispose(CntList.Items[a]);
     CntList.Free;
     for a := 0 to LevList.Count-1 do
      Dispose(LevList.Items[a]);
     LevList.Free;
     for a := 0 to HisList.Count-1 do
      Dispose(HisList.Items[a]);
     HisList.Free;

     FreeMem(aaa);
     FreeMem(bbb);
     end;
end;

procedure TQuantizeDlg.NonUniform2Quan(Sender:
TObject);

 procedure QSort(var A : PArray; sz : Longint);

  procedure QuickSort(var A: PArray; iLo, iHi:
Longint);
   var
    Lo, Hi, Mid, T :  Longint;
   begin
    Lo := iLo;
    Hi := iHi;
    Mid := A^[(Lo + Hi) div 2];
    repeat
     while A^[Lo] < Mid do Inc(Lo);
     while A^[Hi] > Mid do Dec(Hi);
      if Lo <= Hi then
       begin
        T := A^[Lo];
        A^[Lo] := A^[Hi];
        A^[Hi] := T;
        Inc(Lo);
        Dec(Hi);
        end;
    until Lo > Hi;

    if Hi > iLo then QuickSort(A, iLo, Hi);
    if Lo < iHi then QuickSort(A, Lo, iHi);
   end;

 begin
  QuickSort(A, 0, sz);
 end;

var
 m, n : Integer;
 a, b, c, ZeroPoint, mem , aa, bb : Longint;
 aaa, bbb, SumA, NewA : PArray;
 ValList, CntList : TList;
 Plong1, Plong2, Plong3, Plong4 : ^Longint;
begin
 ZeroPoint := -1;
 aaa := nil;
 bbb:= nil;
 SumA := nil;
 NewA := nil;
 try
  a := $800000 - StrToInt(TholdEdit.Text);
  b := $800000 + StrToInt(TholdEdit.Text);
  with MainForm.ActiveMDIChild as TMDIChild do
   begin
    Getmem(aaa, SizeOf( Longint)*
MainForm.res.Height* MainForm.res.Width);
    Getmem(bbb, SizeOf( Longint)*
MainForm.res.Height* MainForm.res.Width);
    ValList := TList.Create;
```

```
      ValList.Clear;
      CntList := TList.Create;
      CntList.CLear;

      c := 0;
      for m := 0 to MainForm.res.Height-1 do
       for n := 0 to MainForm.res.Width-1 do
        begin
         if (Image1.Picture.Bitmap.Canvas.Pixels
[n,m] < a) or (Image1.Picture.Bitmap.Canvas.Pixels
[n,m] > b) then
          begin
           aaa^[c] := (StrToInt(QuanEdit.Text) *
Rround((Image1.Picture.Bitmap.Canvas.Pixels[n,m]-
$800000) / StrToInt(QuanEdit.Text)));
           bbb^[c] := aaa^[c];
           end
          else
           begin
            aaa^[c] := 0;
            bbb^[c] := 0;
            end;
         inc(c);
         end;
      end;
     QSort(aaa,c-1);
     b := -1;
     for a := 0 to c-2 do
      if aaa^[a+1] - aaa^[a] <> 0 then
       begin
        ValList.Add(@aaa[a]);
        New(Plong2);
        Plong2^ := a - b;
        b := a;
        CntList.Add(Plong2);
        end;
     if aaa^[c-1] - aaa^[c-2] <> 0 then
       begin
        ValList.Add(@aaa[c-1]);
        New(Plong2);
        Plong2^ := 1;
        CntList.Add(Plong2);
        end;

     Memo1.Clear;
     for a := 0 to ValList.Count-1 do
      begin
       Plong2 := ValList.Items[a];
       Plong3 := CntList.Items[a];
       Memo1.Lines.Add( Format('%3d %8d %8d
%10d',
        [a,Plong2^,Plong3^, Plong2^*Plong3^]));
       end;
     Memo1.Lines.Add('Total : '+IntToStr
(ValList.Count));

     Getmem(SumA, SizeOf( Longint)* ValList.Count);
     Getmem(NewA, SizeOf( Longint)* ValList.Count);

     for a := 0 to ValList.Count-1 do
      SumA^[a] := 0;

     with MainForm.ActiveMDIChild as TMDIChild do
      begin
       c := 0;
       for m := 0 to MainForm.res.Height-1 do
        for n := 0 to MainForm.res.Width-1 do
         begin
          if bbb^[c] <> 0 then
           for a := 0 to ValList.Count-1 do
            begin
             Plong1 := ValList.Items[a];
             if bbb^[c] = Plong1^ then
              begin
               SumA^[a] := SumA^[a] +
(Image1.Picture.Bitmap.Canvas.Pixels[n,m]-
$800000);
                Break;
                end;
              end;
           Inc(c);
           end;
       end;
     for a := 0 to ValList.Count-1 do
      begin
       Plong1 := CntList.Items[a];
       NewA^[a] := Round(SumA^[a] / Plong1^);
       end;

     for a := 0 to ValList.Count-1 do
      begin
       Plong1 := ValList.Items[a];
       Plong2 := CntList.Items[a];
       Memo1.Lines.Add( Format('%3d %8d %8d
%8d',
        [a,Plong1^,Plong2^,NewA^[a]]));
       end;

     c := 0;
     for m := 0 to MainForm.res.Height-1 do
      for n := 0 to MainForm.res.Width-1 do
       begin
        if bbb^[c] = 0 then
         MainForm.res.Canvas.Pixels[n,m] :=
$800000
         else
          for a := 0 to ValList.Count-1 do
           begin
            Plong1 := ValList.Items[a];
            if bbb^[c] = Plong1^ then
             begin
              MainForm.res.Canvas.Pixels[n,m] :=
NewA^[a] + $800000;
              Break;
              end;
            end;
        Inc(c);
        end;


    finally
     ValList.Free;
     for a := 0 to CntList.Count-1 do
      Dispose(CntList.Items[a]);
     CntList.Free;

     FreeMem(aaa);
     FreeMem(bbb);
     FreeMem(SumA);
     FreeMem(NewA);
     end;
end;
end.
```

Coding.pas , Coding.dfm

เป็นส่วนการเข้ารหัสข้อมูล และแสดงอัตราการลดขนาดข้อมูลที่ได้



รูปแสดงหน้าต่างของการทำ Coding โดยในรูปจะแสดงผลลัพท์ที่ได้จากการเข้ารหัสค่าสัมประสิทธิ์

โปรแกรม Coding.dfm

```
object CodingDlg: TCodingDlg
  Left = 231
  Top = 196
  BorderStyle = bsDialog
  Caption = 'Coding'
  ClientHeight = 193
  ClientWidth = 399
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  Position = poScreenCenter
  PixelsPerInch = 96
  TextHeight = 14
  object OKBtn: TButton
    Left = 311
    Top = 5
    Width = 81
    Height = 27
    Caption = 'OK'
    Default = True
    ModalResult = 1
    TabOrder = 0
  end
  object CodingBtn: TButton
    Left = 311
    Top = 129
    Width = 81
    Height = 27
    Cancel = True
    Caption = 'Coding'
    TabOrder = 1
    OnClick = CodingBtnClick
  end
  object SaveCodeBtn: TButton
    Left = 311
    Top = 161
    Width = 81
    Height = 27
    Cancel = True
    Caption = 'Save Code'
    Enabled = False
    TabOrder = 2
  end
  object Memo1: TMemo
    Left = 4
    Top = 4
    Width = 297
    Height = 185
    ScrollBars = ssVertical
    TabOrder = 3
  end
  object CheckBox1: TCheckBox
    Left = 312
    Top = 104
    Width = 81
    Height = 17
    Caption = 'Show Detail'
    State = cbChecked
    TabOrder = 4
  end
end
```

โปรแกรม Coding.pas

```
unit Coding;

interface

uses Windows, SysUtils, Classes, Graphics, Forms,
Controls, StdCtrls,
  Buttons, ExtCtrls;

type
  TCodingDlg = class(TForm)
    OKBtn: TButton;
    CodingBtn: TButton;
    SaveCodeBtn: TButton;
    Memo1: TMemo;
    CheckBox1: TCheckBox;
    procedure CodingBtnClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  CodingDlg: TCodingDlg;

implementation

uses  Main, ChildWin;

{$R *.DFM}

type
```

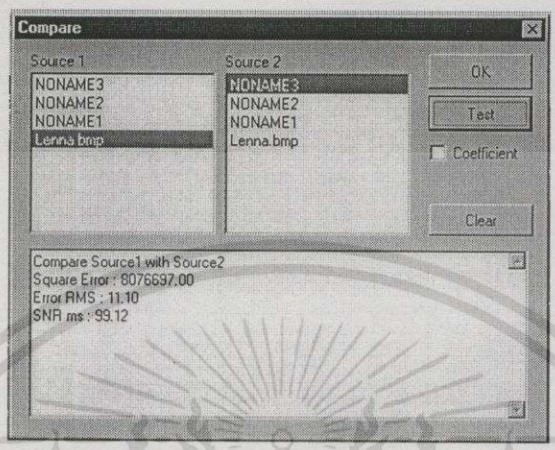
```
  TArray = array[0..3000000] of Longint;
  PArray = ^Tarray;

procedure TCodingDlg.CodingBtnClick(Sender:
TObject);

  procedure QSort(var A : PArray; sz : Longint);

    procedure QuickSort(var A: PArray; iLo, iHi:
Longint);
    var
      Lo, Hi, Mid, T :  Longint;
    begin
      Lo := iLo;
      Hi := iHi;
      Mid := A^[(Lo + Hi) div 2];
      repeat
       while A^[Lo] < Mid do Inc(Lo);
        while A^[Hi] > Mid do Dec(Hi);
         if Lo <= Hi then
           begin
             T := A^[Lo];
             A^[Lo] := A^[Hi];
             A^[Hi] := T;
             Inc(Lo);
             Dec(Hi);
             end;
    until Lo > Hi;

    if Hi > iLo then QuickSort(A, iLo, Hi);
    if Lo < iHi then QuickSort(A, Lo, iHi);
    end;

  begin
    QuickSort(A, 0, sz);
  end;

  function Huffman(var Hist, Code, Len : PArray;
Count : Longint) : Longint;
  var
    v, HistTemp : PArray;
    M, i, j, loc, ctemp, Ltemp, temp, sum : Longint;
  begin
    try
      Getmem(v, SizeOf( Longint)* Count);
      Getmem(HistTemp, SizeOf( Longint)* Count);

      for j := 0 to Count-1 do
       begin
         HistTemp^[j] := Hist^[Count-(j+1)];
         if HistTemp^[j] = 0 then
           Count := j;
         end;
      for j := 0 to Count-1 do
       Hist^[j] := HistTemp^[j];

      for i :=  0 to (Count-2)-1 do
       v^[i] := 0;
      M := Count;
      for i := 0 to (Count-2)-1 do
       begin
         HistTemp^[M-2] := HistTemp^[M-1] +
HistTemp^[M-2];
         loc := M - 2;
         for j := 0 to (M-1)-1 do
          if HistTemp^[M-2] >= HistTemp^[j] then
            begin
              loc := j;
              Break;
              end;
         temp := HistTemp^[M-2];
         for j := M-2 downto loc do
          HistTemp^[j] := HistTemp^[j-1];
         HistTemp^[loc] := temp;
         Dec(M);
         v^[(Count-3)-i] := loc;
         end;

      for j := 0 to Count-1 do
       begin
         Code^[j] := 0;
         Len^[j] := 1;
         end;
      Code^[0] := 0;
      Code^[1] := 1;
      M := 1;
      for i := 0 to (Count-2)-1 do
       begin
         if v^[i] = M then
          begin
            Code^[M+1] := ( Code^[M] shl 1) + 1;
            Code^[M] := Code^[M] shl 1;
            Inc(Len^[M]);
            Len^[M+1] := Len^[M];
            end
          else
            begin
              ctemp := Code^[v^[i]];
              Ltemp := Len^[v^[i]];
              for j := v^[i] to M-1 do
               begin
                 Code^[j] := Code^[j+1];
                 Len^[j] := Len^[j+1];
                 end;
              Code^[M] := ctemp;
              Len^[M] := Ltemp;
              Code^[M+1] := ( Code^[M] shl 1) + 1;
              Code^[M] := Code^[M] shl 1;
              Inc(Len^[M]);
              Len^[M+1] := Len^[M];
              end;
          Inc(M);
          end;
      finally
       Freemem(v);
       Freemem(HistTemp);
       end;

      sum := 0;
      for j := 0 to Count-1 do
       sum := sum + Hist^[j] * Len^[j];
      Huffman := sum;
    end;

  var
    m, n : Integer;
    a, b, c, s1, s2, ss1, ss2   : Longint;
    aaa, bbb, Hist, Code, Len, ZHist, ZCode, ZLen :
PArray;
    ValList, CntList, ZroList, ZValList, ZCntList : TList;
    Plong1, Plong2, Plong3, Plong4 : ^Longint;
    flag : Boolean;
  begin
    try
     with MainForm.ActiveMDIChild as TMDIChild do
      begin
        Getmem(aaa, SizeOf( Longint)*
Image1.Picture.Height* Image1.Picture.Width);
        Getmem(bbb, SizeOf( Longint)*
Image1.Picture.Height* Image1.Picture.Width);
        ValList := TList.Create;
        ValList.Clear;
        CntList := TList.Create;
        CntList.CLear;
        ZroList := TList.Create;
        ZroList.Clear;

        c := 0;
        flag := True;
        for m := 0 to Image1.Picture.Height-1 do
```

```
      for n := 0 to Image1.Picture.Width-1 do
        if (Image1.Picture.Bitmap.Canvas.Pixels[n,m]
<> $800000) then
          begin
           aaa^[c] :=
Image1.Picture.Bitmap.Canvas.Pixels[n,m] -
$800000;
           bbb^[c] := aaa^[c];
           Inc(c);
           if not flag then
            begin
              New(Plong1);
              Plong1^ := a;
              ZroList.Add(Plong1);
              flag := True;
              end;
           end
          else
           if flag then
            begin
              aaa^[c] := 0;
              bbb^[c] := aaa^[c];
              Inc(c);
              flag := False;
              a := 0;
              end
            else
              Inc(a);

    end;
   if not flag then
    begin
      New(Plong1);
      Plong1^ := a;
      ZroList.Add(Plong1);
      end;

   QSort(aaa,c-1);

   b := -1;
   for a := 0 to c-2 do
    if aaa^[a+1] - aaa^[a] <> 0 then
      begin
       ValList.Add(@aaa[a]);
       New(Plong2);
       Plong2^ := a - b;
       b := a;
       CntList.Add(Plong2);
       end;
   if aaa^[c-1] - aaa^[c-2] <> 0 then
      begin
       ValList.Add(@aaa[c-1]);
       New(Plong2);
       Plong2^ := 1;
       CntList.Add(Plong2);
       end;

   Memo1.Clear;

   if CheckBox1.Checked then
    for a := 0 to ValList.Count-1 do
      begin
       Plong2 := ValList.Items[a];
       Plong3 := CntList.Items[a];
       Memo1.Lines.Add( Format('%3d %8d %8d
%10d',
         [a,Plong2^,Plong3^, Plong2^*Plong3^]));
       end;

   Memo1.Lines.Add('Total Code Level : '+IntToStr
(ValList.Count));

   Getmem(Hist, Sizeof(Longint)*CntList.Count+1);
   Getmem(Code, Sizeof(Longint)*CntList.Count+1);
   Getmem(Len, Sizeof(Longint)*CntList.Count+1);

   for a := 0 to CntList.Count-1 do
    begin
     Plong1 := CntList.Items[a];
     Hist^[a] := Plong1^;
     end;

   Qsort(Hist,CntList.Count-1);

   s1 := Huffman(Hist,Code,Len,CntList.Count);

   Memo1.Lines.Add('Coding for Non zero');
   ss1 := 0;
   for a := 0 to CntList.Count-1 do
    begin
     Memo1.Lines.Add( Format('%8d %8d %8d',
      [ Hist^[a], Code^[a], Len^[a]]));
     ss1 := ss1 + Len^[a];
     end;
   Memo1.Lines.Add('Total Code Bit : '+IntToStr
(s1));

{ do for zero value}
     ZValList := TList.Create;
     ZValList.Clear;
     ZCntList := TList.Create;
     ZCntList.CLear;

   for a := 0 to ZroList.Count-1 do
    begin
     Plong1 := ZroList.Items[a];
     aaa^[a] := Plong1^;
     end;
   c := ZroList.Count;
   QSort(aaa,c-1);

   b := -1;
   for a := 0 to c-2 do
    if aaa^[a+1] - aaa^[a] <> 0 then
     begin
      ZValList.Add(@aaa[a]);
      New(Plong2);
      Plong2^ := a - b;
      b := a;
      ZCntList.Add(Plong2);
      end;
   if aaa^[c-1] - aaa^[c-2] <> 0 then
     begin
      ZValList.Add(@aaa[c-1]);
      New(Plong2);
      Plong2^ := 1;
      ZCntList.Add(Plong2);
      end;
   if CheckBox1.Checked then
    for a := 0 to ZValList.Count-1 do
     begin
      Plong2 := ZValList.Items[a];
      Plong3 := ZCntList.Items[a];
      Memo1.Lines.Add( Format('%3d %8d %8d
%10d', [a,Plong2^,Plong3^, Plong2^*Plong3^]));
      end;

   Memo1.Lines.Add('Total Zero Count : '+IntToStr
(ZValList.Count));

   Getmem(ZHist,
Sizeof(Longint)*ZCntList.Count+1);
   Getmem(ZCode,
Sizeof(Longint)*ZCntList.Count+1);
   Getmem(ZLen,
Sizeof(Longint)*ZCntList.Count+1);


   for a := 0 to ZCntList.Count-1 do
    begin
```

```
        Plong1 := ZCntList.Items[a];
        ZHist^[a] := Plong1^;
        end;

    Qsort(ZHist,ZCntList.Count-1);

    s2 :=
Huffman(ZHist,ZCode,ZLen,ZCntList.Count);

    Memo1.Lines.Add('Coding for Zero');
    ss2 := 0;
    for a := 0 to ZCntList.Count-1 do
     begin
       Memo1.Lines.Add( Format('%8d %8d %8d',
         [ ZHist^[a], ZCode^[a], ZLen^[a]]));
       ss2 := ss2 + ZLen^[a];
       end;
    Memo1.Lines.Add('Total Zero Bit : '+IntToStr
(s2));

    b := s1+s2;
    Memo1.Lines.Add('Total Bit for data : '+IntToStr
(b));
    c := (12*4)+ss1+ss2+(
(ZCntList.Count+CntList.Count)*20);
{16 * 4 is [size_x size_y code_count zero_count]}
    Memo1.Lines.Add('Total Bit for Header :
'+IntToStr(c));

    Memo1.Lines.Add('Total Bit : '+IntToStr(b+c));
    Memo1.Lines.Add('Bit/Pixel : '+floatToStr((b+c)/
(m*n)));


  finally
    ValList.Free;
    for a := 0 to CntList.Count-1 do
      Dispose(CntList.Items[a]);
    CntList.Free;
    for a := 0 to ZroList.Count-1 do
      Dispose(ZroList.Items[a]);
    ZroList.Free;

    ZValList.Free;
    for a := 0 to ZCntList.Count-1 do
      Dispose(ZCntList.Items[a]);
    ZCntList.Free;

    FreeMem(aaa);
    FreeMem(bbb);

    FreeMem(Hist);
    FreeMem(Code);
    FreeMem(Len);

    FreeMem(ZHist);
    FreeMem(ZCode);
    FreeMem(ZLen);

    end;
end;


end.
```

Compare.pas , Compare.dfm

สำหรับการเปรียบเทียบความแตกต่างระหว่างข้อมูลสองชุด โดยจะแสดงออกมาเป็น E_{rms} และค่าของ SNR_{ms}



รูปแสดงหน้าต่างของ การเปรียบเทียบข้อมูล โดยในรูปจะแสดงผลการเปรียบเทียบข้อมูลภาพต้นแบบ (LENA.bmp) กับข้อมูลภาพที่ได้หลังจากผ่านขบวนลดขนาดข้อมูลแล้ว (NONAME3)

โปรแกรม Compare.dfm

```
object CompareDlg: TCompareDlg
 Left = 249
 Top = 160
 BorderStyle = bsDialog
 Caption = 'Compare'
 ClientHeight = 307
 ClientWidth = 414
 Font.Charset = DEFAULT_CHARSET
 Font.Color = clWindowText
 Font.Height = -11
 Font.Name = 'MS Sans Serif'
 Font.Style = []
 Position = poScreenCenter
 OnActivate = FormActivate
 PixelsPerInch = 96
 TextHeight = 14
 object Label3: TLabel
  Left = 12
  Top = 8
  Width = 42
  Height = 14
  Caption = 'Source 1'
 end
 object Label4: TLabel
  Left = 164
  Top = 8
  Width = 43
  Height = 14
  Caption = 'Source 2'
 end
 object OKBtn: TButton
  Left = 323
  Top = 9
  Width = 81
  Height = 27
  Caption = 'OK'
  Default = True
  ModalResult = 1
  TabOrder = 0
 end
 object TestBtn: TButton
  Left = 323
  Top = 41
  Width = 81
  Height = 27
  Cancel = True
  Caption = 'Test'
  Enabled = False
  TabOrder = 1
  OnClick = TestBtnClick
 end
 object ListBox1: TListBox
  Left = 12
  Top = 24
  Width = 145
  Height = 125
  ItemHeight = 14
  TabOrder = 2
  OnClick = ListBox1Click
 end
 object ListBox2: TListBox
  Left = 164
  Top = 24
  Width = 145
  Height = 125
  ItemHeight = 14
  TabOrder = 3
  OnClick = ListBox1Click
 end
 object Memo1: TMemo
  Left = 12
  Top = 160
  Width = 389
  Height = 133
  ScrollBars = ssVertical
  TabOrder = 4
```

```
  end
  object ClearBtn: TButton
    Left = 324
    Top = 124
    Width = 81
    Height = 25
    Caption = 'Clear'
    TabOrder = 5
    OnClick = ClearBtnClick
  end
  object CheckBox1: TCheckBox
    Left = 324
    Top = 76
    Width = 69
    Height = 17
    Caption = 'Coefficient'
    TabOrder = 6
  end
end
```

โปรแกรม Compare.pas

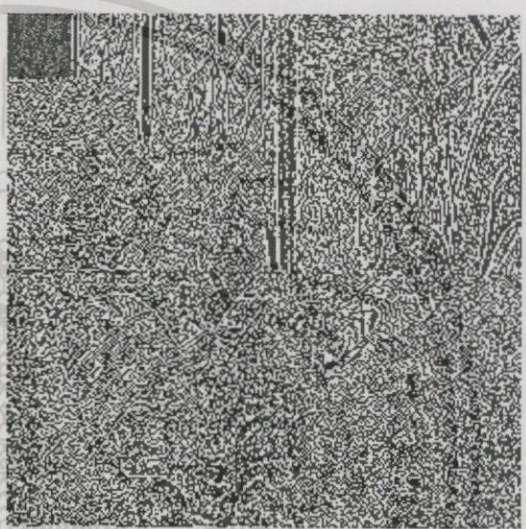
```
unit compare;

interface

uses Windows, SysUtils, Classes, Graphics, Forms,
Controls, StdCtrls,
  Buttons, ExtCtrls, Mask, Math;

type
  TCompareDlg = class(TForm)
    OKBtn: TButton;
    TestBtn: TButton;
    ListBox1: TListBox;
    ListBox2: TListBox;
    Label3: TLabel;
    Label4: TLabel;
    Memo1: TMemo;
    ClearBtn: TButton;
    CheckBox1: TCheckBox;
    procedure FormActivate(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure TestBtnClick(Sender: TObject);
    procedure ClearBtnClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  CompareDlg: TCompareDlg;

implementation

uses Main, ChildWin;

{$R *.DFM}

procedure TCompareDlg.FormActivate(Sender:
TObject);
var
  I : Integer;
begin
  ListBox1.Items.Clear;
  for I := 0 to MainForm.MDIChildCount - 1 do
    ListBox1.Items.Add(ExtractFileName
(MainForm.MDIChildren[I].Caption));
  ListBox2.Items := ListBox1.Items;
  ListBox1Click(Sender);
end;

procedure TCompareDlg.ListBox1Click(Sender:
TObject);
begin
  TestBtn.Enabled := (ListBox1.ItemIndex > -1) and
(ListBox2.ItemIndex > -1);
end;

procedure TCompareDlg.TestBtnClick(Sender:
TObject);
var
  ref : TBitmap;
  ep2, sum, psnr : Single;
  m, n : Integer;
begin
  with MainForm.MDIChildren[ListBox1.ItemIndex] as
TMDIChild do
    ref := Image1.Picture.Bitmap;
  with MainForm.MDIChildren[ListBox2.ItemIndex] as
TMDIChild do
    begin
      ep2 := 0.0;
      sum := 0.0;
      for m := 0 to ref.Height-1 do
        for n := 0 to ref.Width-1 do
          if CheckBox1.Checked then
            begin
              ep2 := ep2 +
IntPower(Image1.Picture.Bitmap.Canvas.Pixels[n,m] -
ref.Canvas.Pixels[n,m], 2);
              sum := sum +
IntPower(Image1.Picture.Bitmap.Canvas.Pixels[n,m] -
$800000, 2);
            end
            else
              begin
                ep2 := ep2 +
IntPower((Image1.Picture.Bitmap.Canvas.Pixels[n,m]
and $FF)  - (ref.Canvas.Pixels[n,m] and $FF), 2);
                sum := sum +
IntPower(Image1.Picture.Bitmap.Canvas.Pixels[n,m]and
$FF, 2);
              end;
      Memo1.Lines.Add('Compare Source1 with
Source2');
      Memo1.Lines.Add(Format('Square Error : %f',
[ep2]));
      Memo1.Lines.Add(Format('Error RMS : %f', [Sqrt
(ep2 / (ref.Width * ref.Height))]));
      if ep2 <> 0.0 then
        Memo1.Lines.Add(Format('SNR ms : %f',[sum /
ep2]))

{     mse := mse / (ref.Width * ref.Height);
      psnr := 10*log10(IntPower(256,2)/mse);
}
    end;
end;

procedure TCompareDlg.ClearBtnClick(Sender:
TObject);
begin
  Memo1.Lines.Clear;
end;

end.
```

ภาคผนวก ค

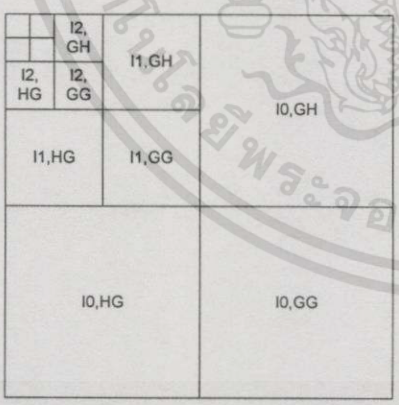
ขบวนการลดขนาดข้อมูล จากบทความ [1] "Image Compression Using the 2-D Wavelet Transform", A.S. Lewis and G. Knowles, IEEE Trans. Image Processing, vol.1, no.2, pp. 244-250, 1992.

จากการแปลง wavelet โดยใช้ mother wavelet แบบ Daubechies 4 ของข้อมูลภาพ

รูปที่ 1 ข้อมูลภาพ Lena.BMP



รูปที่ 2 ข้อมูลของค่าสัมประสิทธิ์ wavelet



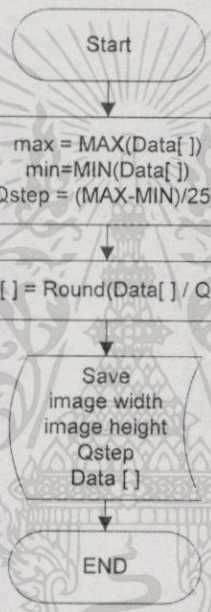
รูปที่ 3 แสดงการจัดเรียงข้อมูล

การลดขนาดของข้อมูลค่าสัมประสิทธิ์ wavelet นี้ จะแบ่งเป็น

สำหรับข้อมูล ส่วนที่เป็น low frequency band (ส่วนซ้ายบนของรูปที่ 3 ซึ่งมีขนาด 32x32 จุดข้อมูล) จะทำการเก็บโดยใช้รหัส 8 bit ข้อมูลจะถูกแบ่งออกเป็นช่วงๆ ที่เท่ากัน 256 ช่วง

สำหรับข้อมูล ในส่วนที่เป็น high frequency band ที่เหลือนั้น จะทำการเก็บข้อมูลโดยใช้หลักการของ zero-tree coding ร่วมกับการ quantize โดยพิจารณาจาก การมองเห็นของ ตามนุษย์ แล้วเก็บข้อมูลโดยใช้รหัสความยาวไม่คงที่

ขบวนการของการเก็บข้อมูลในส่วนที่ 1 นั้น จะสามารถแสดงได้ดังนี้



สำหรับข้อมูลในส่วนของ ค่าสัมประสิทธิ์ จะเก็บโดยใช้หลักการของ zero-tree นั้น จะได้จาก function การทำงานแบบ เรียกตัวเองของ

```
SendTree( I, r, s, x, y)
        If( THRESHOLD < ThresholdFunction( I^{r,s}(x+{0,1},y+{0,1}))) {
                SendToken(BlockNotEmpty);
                SaveCoefficient(I^{r,s}(x+{0,1},y+{0,1}));
                SendTree( I, r-1, s, 2x, 2y);
                SendTree( I, r-1, s, 2(x+1), 2y);
                SendTree( I, r-1, s, 2x, 2(y+1));
```

```
SendTree( I, r-1, s, 2(x+1), 2(y+1));
} else SendToken(BlockEmpty);
```

เมื่อ I : ข้อมูลค่าสัมประสิทธิ์

r : ระดับ

s : Subband ที่ใช้

สามารถอธิบายการทำงานได้คือ function SendTree นี้ จะทำการหาระดับการมองเห็น ของค่าสัมประสิทธิ์ สี่ จุดข้อมูล คำนวณจาก function ThresholdFunction ซึ่งถ้าระดับการมองเห็นมีค่าต่ำกกว่า ค่า THRESHOLD ที่ตั้งไว้ ก็จะไม่เก็บข้อมูลของค่าสัมประสิทธิ์ ในกลุ่มนี้ต่อไป โดยจะเก็บเป็นข้อมูลของ BlockEmpty

ถ้าค่าระดับการมองเห็นที่ได้ มีค่ามากกว่า ค่า THRESHOLD ที่ตั้งไว้ ก็จะทำการเก็บค่าสัมประสิทธิ์สี่จุดข้อมูลที่ normalize กับระดับการมองเห็น จากนั้นจะทำการเรียก function SendTree อีกครั้ง สำหรับข้อมูลสัมประสิทธิ์ในระดับที่ต่ำลงไปจนถึงระดับที่ r = 0

สำหรับค่าของระดับการมองเห็นนั้น จะคำนวณโดยพิจารณาจาก

Band sensitivity

หรือความไวในการรับรู้ของข้อมูลที่อยู่ใน Subband ต่างๆ ค่าใช้นี้จะได้จากการทดลอง เพิ่ม noise เข้าไปใน subband ต่างๆ แล้วทำการสังเกตผลที่เกิดขึ้น ได้ออกมาเป็นค่าคงที่ดังนี้

$$\text{frequency}(r,s) = \begin{cases} \sqrt{2} & s = GG \\ 1 & \text{otherwise} \end{cases} \begin{cases} 1.00 & r = 0 \\ 0.32 & r = 1 \\ 0.16 & r = 2 \\ 0.10 & r = 3 \end{cases}$$

Background luminance

การหาค่าระดับความสว่างจะพิจารณาจาก ค่าของข้อมูลที่อยู่ใน low frequency subband โดยจะคำนวณได้ดังนี้

$$\text{luminance}(r,x,y) = 3 + \frac{1}{256}\sum_{i=0}^{1}\sum_{j=0}^{1} I^{3,HH}(i+1+\frac{x}{2^{3-r}}, j+j+\frac{y}{2^{3-r}})$$

Texture

หรือลายเส้นของภาพ จะคำนวณได้จากสมการ

$$\text{texture}(r,x,y) = \sum_{k=1}^{3-r} 16^{-k} \sum_{s}^{GG,GH,HG} \sum_{i=0}^{1}\sum_{j=0}^{1} (I^{k+r,s}(i+\frac{x}{2^k}, j+\frac{y}{2^k}))^2$$

$$+16^{3-r}\,\text{var}(I^{3,HH}(\{1,2\}+\frac{x}{2^{3-r}}, \{1,2\}+\frac{y}{2^{3-r}}))$$

จากตัวแปลทั้งสามนี้ จะนำมาคำนวณเป็นค่าระดับการมองเห็น และใช้เป็นค่าของ quantization step ด้วย คือ

$$\text{qstep}(r,s,x,y) = q_0 * \text{frequency}(r,s) * \text{luminance}(r,x,y) * \text{texture}(r,x,y)^{0.034}$$

ในการเข้ารหัสข้อมูล ที่ใช้ระหัสแบบความยาวไม่คงที่ จะมีการกำหนดระหัสที่ใช้งานดังนี้

Quantized level	Code Bit
.	.
-5	111001
-4	11101
-3	1111
-2	101
-1	011
0	00
1	010
2	100
3	1101
4	11001
5	110001
.	.

จากผลการทดลองที่ได้ในการลดขนาดข้อมูลภาพต้นแบบทั้งสี่ภาพให้ผลดังต่อไปนี้

ผลการทดลองจากการทำ Threshold ของค่าสัมประสิทธิ์โดยใช้ภาพต้นแบบที่ 1 (Lena.bmp)

ตารางที่ 1 ผลการทดลองที่ค่า Q0=5

Threshold	*5*	*10*	*20*	*30*	*40*	*50*	*70*	*100*	*200*	*500*	*1000*	*1500*	*2000*	*2500*
Erms	8.74	8.81	8.89	9.04	9.2	9.39	9.72	10.22	11.55	13.58	14.95	16.01	16.71	17.15
SNRms	160.7	158.06	155.14	150.18	144.9	138.74	129.37	117	91.2	65.79	54.09	47	43.05	40.86
bit count	50547	42605	36048	32415	30221	28422	26463	24485	20964	16871	14574	13033	12083	11527
Bpp	0.77129	0.6501	0.55005	0.49461	0.46114	0.43369	0.40379	0.37361	0.31989	0.25743	0.22238	0.19887	0.18437	0.17589
Ratio	10.3723	12.3058	14.5442	16.1742	17.3485	18.4466	19.8121	21.4126	25.009	31.0763	35.9742	40.2277	43.3905	45.4835

ตารางที่ 2 ผลการทดลองที่ค่า Q0=10

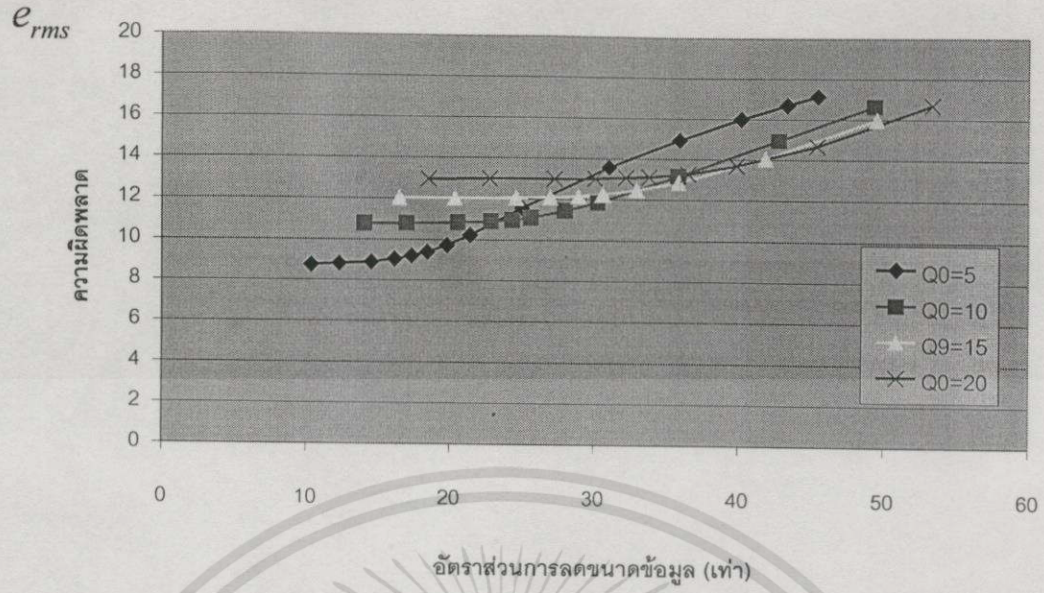
Threshold	*5*	*10*	*20*	*30*	*40*	*50*	*70*	*100*	*200*	*500*	*1000*
Erms	10.77	10.78	10.84	10.89	10.96	11.09	11.43	11.83	13.16	14.95	16.66
SNRms	105.64	105.47	104.27	103.28	101.94	99.49	93.43	87.06	70.08	54.04	43.33
Bit count	37236	30825	25501	22936	21536	20455	18713	17302	14616	12248	10615
Bpp	0.56818	0.47035	0.38911	0.34998	0.32861	0.31212	0.28554	0.26401	0.22302	0.18689	0.16197
Ratio	14.0801	17.0085	20.5595	22.8587	24.3447	25.6313	28.0173	30.3022	35.8708	42.806	49.3912

ตารางที่ 3 ผลการทดลองที่ค่า Q0=15

Threshold	*5*	*10*	*20*	*30*	*40*	*50*	*70*	*100*	*200*	*500*
Erms	12.01	12.03	12.05	12.08	12.14	12.26	12.47	12.86	14.06	15.99
SNRms	84.9	84.54	84.29	83.82	82.93	81.22	78.45	73.63	61.36	47.13
bit count	31790	25766	21308	19440	18115	17113	15894	14599	12517	10583
Bpp	0.48508	0.39316	0.32513	0.29663	0.27641	0.26112	0.24252	0.22276	0.19099	0.16148
Ratio	16.4922	20.3481	24.6052	26.9695	28.9422	30.6368	32.9865	35.9126	41.8861	49.5406

ตารางที่ 4 ผลการทดลองที่ค่า Q0=10

Threshold	*5*	*10*	*20*	*30*	*40*	*50*	*70*	*100*	*200*	*500*
Erms	12.96	12.99	12.99	13.03	13.08	13.12	13.29	13.69	14.67	16.71
SNRms	72.74	72.49	72.48	72.01	71.38	70.95	68.97	64.87	56.23	43.11
bit count	28314	23018	19224	17420	16257	15492	14340	13141	11537	9822
Bpp	0.43204	0.35123	0.29333	0.26581	0.24806	0.23639	0.21881	0.20052	0.17604	0.14987
Ratio	18.5169	22.7773	27.2726	30.0969	32.25	33.8425	36.5612	39.8971	45.444	53.3789



แสดงความสัมพันธ์ ของอัตราส่วนการลดขนาดข้อมูลที่ได้ (Ratio) กับความผิดพลาด (e_{rms}) ที่ได้

และสำหรับภาพผลลัพท์ที่ได้จากขบวนการลดขนาดข้อมูลแล้วจะเป็นดังนี้

ภาพ Lena ที่อัตราการลดข้อมูล 19 เท่า

Erms =9.47 (q_0=5 Threshold=55)

ภาพ Lenaที่อัตราการลดข้อมูล 30 เท่า

Erms =11.83 (q_0=10 Threshold=100)

ภาพ Lena ที่อัตราการลดข้อมูล 40 เท่า
Erms =13.6 (q_0=15 Threshold=150)

สรุปผลการทดลองที่ได้จะเห็นได้ว่า การใช้ zero-tree จะช่วยให้การเก็บข้อมูลของค่าสัมประสิทธิ์มีประสิทธิภาพมาก กว่าการเก็บข้อมูล โดยใช้ run-length coding มาก และขบวนการ ลดข้อมูลโดยพิจารณาจากระดับการมองเห็นนี้ จะทำให้ภาพที่ได้ มีคุณภาพดีกว่าการลดข้อมูลที่พิจารณาเพียงแต่ลักษณะทางสถิติของค่าข้อมูลเพียงอย่างเดียว

จากขบวนการที่ใช้ในการทดลองนี้ จะเห็นได้ว่ามีค่าตัวแปรที่ผู้ใช้จะต้องเป็นผู้กำหนดเอง สองตัวแปรด้วยกันคือ ค่าของ q_0 (Quantize) และค่าของ Threshold จากกราฟผลการทดลองที่ได้ จะเห็นได้ว่าการเลือกค่าของ q_0 และ Threshold นี้จะขึ้นอยู่กับอัตราการลดขนาดข้อมูลที่ต้องการและลักษณะของภาพที่ใช้

ประวัติผู้เขียน

ชื่อ นายชินภัทร นันทจิวากรชัย

เกิดวันที่ 8 กุมภาพันธ์ 2513 จังหวัดขอนแก่น

การศึกษา ปีการศึกษา 2528-2530

ระดับ มัธยมศึกษาตอนปลาย สายวิทยาศาสตร์

โรงเรียนสาธิตมัธยมศึกษา มหาวิทยาลัยขอนแก่น

ปีการศึกษา 2531-2534

ระดับ ปริญญาตรีวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมอิเลคทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง