การใส่ลายน้ำในรูปภาพแบบไม่ใช้ต้นฉบับในการตรวจสอบโดยใช้การเข้ารหัสแก้ไขความผิดพลาดแบบเทอร์โบ

นาย วิทิต พงศ์พิโรดม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2543
ISBN 974-347-205-3

BLIND IMAGE WATERMARKING USING TURBO ERROR CORRECTING CODE

Mr. Vitid Pongpirodom

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Electrical Engineering

Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2000
ISBN 974-347-205-3

หัวข้อวิทยานิพนธ์	การใส่ลายน้ำในรูปภาพแบบไม่ใช้ต้นฉบับในการตรวจสอบโดยใช้การเข้ารหัสแก้ไขความผิดพลาดแบบเทอร์โบ
โดย	นาย วิทิต พงศ์พิโรดม
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษา	อาจารย์ สุวิทย์ นาคพีระยุทธ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร. สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

……ประธานกรรมการ
(ศาสตราจารย์ ดร. ประสิทธิ์ ประพิณมงคลการ)

…….. อาจารย์ที่ปรึกษา
(อาจารย์ สุวิทย์ นาคพีระยุทธ)

……กรรมการ
(รองศาสตราจารย์ ดร. วาทิต เบญจพลกุล)

วิทิต พงศ์พิโรดม : การใส่ลายน้ำในรูปภาพแบบไม่ใช้ต้นฉบับในการตรวจสอบโดยใช้การเข้ารหัสแก้ไขความผิดพลาดแบบเทอร์โบ. (Blind Image Watermarking using Turbo Error Correcting Code) อ. ที่ปรึกษา : อ. สุวิทย์ นาคพีระยุทธ , 60 หน้า. ISBN 974-347-205-3.

งานวิจัยนี้เป็นการศึกษาวิธีการฝังข้อมูลลงในรูปภาพแบบ Public watermarking ซึ่งสามารถดึงข้อมูลจากลายน้ำโดยไม่ใช้รูปต้นฉบับได้และนำเอารหัสเทอร์โบมาช่วยในการแก้ไขบิตที่ผิดพลาด ขั้นตอนการตรวจจับจะนำเอาเทคนิคของ image restoration มาใช้ในการประมาณรูปต้นฉบับ การฝังลายน้ำจะฝังในโดเมนของการแปลงโคซายน์ งานวิจัยยังได้หาวิธีกำหนดช่วงสัมประสิทธิ์ที่เหมาะสมในการฝังลายน้ำของแต่ละรูปที่ระดับการมองเห็นเดียวกัน และทดสอบความทนทานของลายน้ำซึ่งเป็นการนำรูปที่ผ่านการฝังลายน้ำไปเปลี่ยนแปลงให้ต่างจากเดิมด้วยวิธีต่างๆก่อนตรวจจับลายน้ำ ในการทดสอบความทนทานแต่ละวิธีจะหาจำนวนบิตที่สามารถฝังลงไปได้มากที่สุดที่อัตราความผิดพลาดของบิตไม่เกิน 10^{-3} งานวิจัยนี้พบว่ารหัสเทอร์โบสามารถนำมาแก้ไขบิตที่ผิดพลาดได้เป็นอย่างดีโดยสามารถเพิ่มจำนวนบิตได้มากกว่า 3 เท่าขึ้นไปเมื่อเทียบกับกรณีไม่มีการเข้ารหัสหรือใช้รหัสแฮมมิง (7,4) แทนการเข้ารหัสเทอร์โบ เนื่องจากรหัสเทอร์โบทำงานได้ดีสำหรับสัญญาณที่มี SNR ต่ำมากซึ่งการใส่ลายน้ำมีลักษณะเช่นนี้

ภาควิชา ..... ลายมือชื่อนิสิต .....

สาขาวิชา ..... ลายมือชื่ออาจารย์ที่ปรึกษา .....

ปีการศึกษา 2543

4270539821 : MAJOR ELECTRICAL ENGINEERING

KEY WORD: WATERMARKING / TURBO CODE /

VITID PONGPIRODOM : THESIS TITLE. (BLIND IMAGE WATERMARKING USING TURBO ERROR CORRECTING CODE) THESIS ADVISOR : SUVIT NAKPEERAYUTH, 60 pp. ISBN 974-347-205-3.

This research studies how to embed information into image using public watermarking, where bit can be extracted without original image. Turbo code was used for bit error correction. Image restoration technique was used to estimate the orginal image for extracting information bits. Information bits were embeded in DCT domain. This research also investigated the method to determine the optimum range of zigzag DCT coefficients to be embedded for each image at the same visibility level. The watermarking robustness for various image processing methods was tested and the maximum embedded bits at 10^{-3} bit error rate was found.

The turbo code can increase the maximum number of embedded bits more than 3 times when compared to the cases of no coding and using Hamming code (7,4) instead of Turbo code. This is because the turbo code is more efficient at low SNR level and image watermarking is usually under this condition.

Department ……. Student's signature……

Field of study ……. Advisor's signature .....

Academic year 2000

กิตติกรรมประกาศ

ขอบคุณ นาย พิสิฐ วนิชชานันท์ และ นาย อาทิตย์ จำปาศรี สำหรับคำอธิบายเรื่องรหัสเทอร์โบและเพื่อนๆพี่ๆในห้องปฏิบัติการวิจัยสื่อสารสำหรับกำลังใจที่มีให้

สุดท้ายนี้ขอขอบคุณ คุณพ่อ คุณแม่และน้อง สำหรับการสนับสนุนและกำลังใจในการเรียนและการทำวิจัยตลอดมา

วิทิต พงศ์พิโรดม

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันอินเตอร์เนตได้เป็นที่กล่าวถึงอย่างมากในการที่ช่วยให้ผู้คนสามารถติดต่อทำธุรกิจหรือแลกเปลี่ยนข้อมูล ผู้คนสามารถหาข้อมูลต่างๆที่อยากรู้ได้ด้วยปลายนิ้วผ่านทางโฮมเพจต่างๆ อย่างไรก็ตามการที่อินเตอร์เนตสามารถช่วยให้การแลกเปลี่ยนหรือดูข้อมูลเป็นไปได้โดยง่ายก็ทำให้เกิดปัญหาต่างๆตามมาซึ่งก็คือการละเมิดทรัพย์สินทางปัญญาในรูปแบบต่างๆเช่นโปรแกรมคอมพิวเตอร์ เพลง หรือภาพถ่ายดิจิตอล ซึ่งก็ได้มีข้อถกเถียงกันว่าจะป้องกันอย่างไรดี

วิธีหนึ่งซึ่งช่วยในการแก้ปัญหานี้คือวิธีที่เรียกว่า Digital Watermarking [1-4] ซึ่งจะเป็นกระบวนการที่จะใส่หรือฝัง (embed) สัญญาณหนึ่งไปในอีกสัญญาณหนึ่งซึ่งก็คือข้อมูลต้นฉบับ (host data) โดยมักจะเป็นข้อมูลเกี่ยวกับเจ้าของ,ผู้ผลิต,ลูกค้า หรือรายละเอียดใดใดก็ตามเกี่ยวกับลิขสิทธิ์ของข้อมูลต้นฉบับนั้น โดยที่ข้อมูลที่ฝังไปนั้นจะไม่สามารถเห็นหรือสังเกตุได้ยกเว้นเมื่อนำไปทำการตรวจจับที่เหมาะสม วิธีการใส่ลายน้ำสามารถทำได้กับข้อมูลภาพ,เสียงและภาพเคลื่อนไหวโดยที่วิทยานิพนธ์เรื่องนี้จะทำการศึกษาถึงการใส่ลายน้ำในภาพดิจิตอล

ลายน้ำแบบดิจิตอลแบ่งออกเป็น 3 ประเภทคือ

1. Private watermarking จะต้องใช้รูปต้นฉบับในการตรวจสอบลายน้ำแบ่งเป็น
   - Type I เป็นการฝังบิตข้อมูลลงไปจริงๆ
   - Type II เป็นการตอบคำถามเพียงว่ารูปที่นำมาตรวจสอบมีหรือไม่มีลายน้ำเช่นงานของ Cox [5]
2. Semi-private watermarking ไม่ต้องใช้รูปต้นฉบับในการตรวจสอบลายน้ำและเป็นการตอบคำถามเพียงว่ารูปที่นำมาทดสอบมีหรือไม่มีลายน้ำซึ่งงานวิจัยส่วนมากจะอยู่ในจำพวกนี้เช่นงานของ Piva [6], Pitas [7]
3. Public watermarking เป็นแบบที่ยากที่สุดเพราะไม่ต้องมีรูปต้นฉบับในการตรวจสอบลายน้ำและเป็นการฝังบิตข้อมูลลงไปจริงๆซึ่งประเภทนี้จะมีความทนทานน้อยกว่า 2 แบบแรกเช่นงานของ Girod [8], Lisa [9]

ลายน้ำจะต้องมีคุณสมบัติดังต่อไปนี้ [3]

1. ความปลอดภัยและกุญแจ (Watermark Security and Keys)

สำหรับระบบที่ต้องการความปลอดภัยอาจมีการใช้กุญแจ (key) ในการฝังหรือการดึงลายน้ำเช่น กรณีที่ใช้เลขสุ่มเทียม (pseudorandom number) เป็นลายน้ำอาจใช้ค่า seed ของตัวกำเนิดเลขสุ่มเทียม (pseudorandom number generator) เป็นกุญแจได้

2. ความทนทาน (Robustness)

เป็นลักษณะสำคัญที่จำเป็นต้องมีในทุกระบบลายน้ำจะต้องถูกฝังแบบเอาออกยากหรือไม่สามารถเอาออกได้ถ้าไม่รู้ตำแหน่งที่มีลายน้ำอยู่หรือถ้าพยายามจะเอาออกก็จะทำให้รูปถูกทำลายอย่างมากก่อนที่จะเอาลายน้ำออกได้หมด โดยทั่วไปลายน้ำควรจะมีความทนทานต่อ

- การประมวลผลสัญญาณทั่วไป เช่น JPEG compression, Low pass filtering, Median filtering, Histogram equalization, Dithering

- ความเพี้ยนเชิงเรขาคณิต เช่น Rotation, Translation, Cropping, Resizing

3. การมองไม่เห็น (Imperceptibility)

ลักษณะสำคัญอีกประการหนึ่งก็คือลายน้ำที่ฝังลงไปจะต้องมองไม่เห็น ซึ่งความต้องการนี้จะเป็นไปในทางตรงข้ามกับความต้องการในข้อ 2 คือความทนทานดังนั้นในการออกแบบระบบจึงต้องมีการชดเชยข้อดีข้อเสีย (Tradeoff) ระหว่างความต้องการทั้งสองให้เหมาะสม

ในวิทยานิพนธ์นี้จะได้มีการปรับปรุงการใส่ลายน้ำจากที่มีข้อเสียของแต่ละวิธีคือ

1. private watermarking เช่นวิธีของ Cox จะต้องใช้รูปต้นฉบับในการตรวจสอบและเป็นเพียงการตรวจว่ามีหรือไม่มีลายน้ำเท่านั้น

2. semi-private watermarking เช่นวิธีของ Piva ไม่ต้องใช้รูปต้นฉบับแต่เป็นเพียงการตรวจว่ามีหรือไม่มีลายน้ำเท่านั้น

3. public watermarking ซึ่งเป็นการฝังบิตข้อมูลลงไปจริงๆแต่ความทนทานจะน้อยกว่าสองแบบแรก เช่นวิธีของ Lisa สนใจแต่การฝังบิตให้ได้มากที่สุดโดยไม่สนใจความทนทาน

ในการฝังลายน้ำลงในภาพก็เหมือนกับการมอดูเลตลายน้ำลงไปบนข้อมูลต้นฉบับซึ่งในขั้นตอนการตรวจจับลายน้ำก็จะมีการดีมอดูเลตซึ่งก็จะมีการให้บิตที่ผิดพลาดออกมาดังนั้นจึงได้มีการคิดนำเอารหัสแก้ไขความผิดพลาดมาใช้ด้วย [9,10] โดยรหัสแก้ไขความผิดพลาดที่ได้นำมาใช้จะใช้รหัสเทอร์โบเนื่องจากให้อัตราความผิดพลาดของบิตที่ต่ำที่ SNR น้อยๆ

จึงมีแนวคิดในการปรับปรุงโดยการตรวจจับลายน้ำโดยจะใช้วิธี public watermarking นั่นคือจะไม่ใช้รูปต้นฉบับในการตรวจสอบและเป็นการฝังบิตข้อมูลลงไปจริงๆโดยขั้นตอนการตรวจสอบจะใช้รหัสเทอร์โบช่วยในการแก้ไขความผิดพลาด โดยรูปที่ผ่านการฝังลายน้ำแล้วจะนำไปทดสอบความทนทานด้วยการประมวลผลแบบต่างๆด้วยเช่น JPEG compression ที่ Quality

ต่างๆกัน, Median filtering, Histogram equalization และอื่นๆก่อนจะทำการตรวจจับลายน้ำเพื่อทดสอบหาอัตราความผิดพลาด

1.2 วัตถุประสงค์ของการวิจัย

เพื่อปรับปรุงวิธีการใส่ลายน้ำแบบที่ไม่ใช้ต้นฉบับโดยใช้รหัสเทอร์โบช่วยเพื่อเพิ่มจำนวนบิตที่ฝังสำหรับการทดสอบความทนทานหนึ่ง ๆที่ระดับการมองเห็นคงที่

1.3 ขั้นตอนการดำเนินงาน

1. ศึกษาลักษณะพื้นฐาน, วิธีการ, ข้อดีและข้อเสียของวิธีใส่ลายน้ำในรูปภาพแบบต่าง ๆ ในงานวิจัยที่ผ่านมา
2. ปรับปรุงวิธีใส่ลายน้ำ
3. เขียนโปรแกรมเพื่อป้องกันการละเมิดลิขสิทธิ์รูปภาพ
4. ทดสอบความทนทานต่อการประมวลผลภาพต่างๆ
5. สรุปผลการทดสอบและเขียนวิทยานิพนธ์

1.4 เป้าหมายและขอบเขตของวิทยานิพนธ์

1. นำเอารหัสเทอร์โบมาช่วยในการตรวจสอบลายน้ำเพื่อลดความผิดพลาด
2. ทดสอบความทนทานต่อการประมวลผลแบบต่างๆ
3. หาจำนวนบิตที่ใส่ได้มากที่สุดเมื่อทดสอบความทนทานโดยใช้ JPEG compression ที่ Quality Factor 80 โดยมี bit error rate ไม่เกิน 10^{-3} เมื่อกำหนดให้ความแรงในการฝังเท่ากันในระดับการมองเห็นคงที่

1.5 ประโยชน์ที่คาดว่าจะได้รับ

สามารถใช้โปรแกรมที่พัฒนาขึ้นเพื่อป้องกันการละเมิดลิขสิทธิ์ข้อมูลรูปภาพได้

บทที่ 2
การฝังลายน้ำ

บทนี้จะกล่าวถึงหลักการพื้นฐานของการใส่ลายน้ำและการใส่ลายน้ำด้วยวิธีต่าง ๆโดย ก่อนอื่นจะอธิบายถึงคำศัพท์บางคำที่ใช้ในวิทยานิพนธ์นี้

1. ลายน้ำ หมายถึง บิตข้อมูลที่ต้องการฝังเข้าไปในภาพ

2. การแปลงโคซายน์ (Discrete Cosine Transform) หมายถึง เทคนิคที่ใช้ในการแสดงรูปแบบข้อมูลในรูปของผลรวมของค่าโคซายน์

3. HVS (Human Visual System) หมายถึง ระบบการมองเห็นของมนุษย์

4. Image Restoration หมายถึง การทำให้ภาพที่เปลี่ยนแปลงไปเป็นเหมือนเดิม

5. AWGN (Additive White Gaussian Noise) หมายถึง สัญญาณรบกวนแบบบวกเพิ่มที่มีสเปกตรัมความถี่แบบต่อเนื่องเท่ากันทุกความถี่และการแจกแจงของแอมพลิจูดของสัญญาณรบกวนเป็นแบบเกาส์

6. MSE (Mean Square Error) หมายถึง ค่าเฉลี่ยของความแตกต่างของสัญญาณรบกวนยกกำลังสอง

7. JPEG (Joint Photographic Experts Group) หมายถึง มาตรฐานการบีบอัดภาพแบบหนึ่งโดยเป็นแบบมีการสูญเสียเมื่อทำการบีบอัด มาจากชื่อคณะกรรมการของ ITU ที่กำหนดมาตรฐานนี้

8. Histogram Equalization หมายถึง เทคนิคหนึ่งที่ทำให้ภาพมีการแจกแจงของความเข้มของจุดภาพแบบยูนิฟอร์มโดยทำการแปลงความเข้มของแต่ละจุดภาพใหม่แบบไม่เชิงเส้น

9. Hamming code (7,4) หมายถึง รหัสตรวจวัดและแก้ความผิดพลาดแบบหนึ่งโดยบิตข้อมูล 4 บิตจะถูกเข้ารหัสได้บิตออกมา 7 บิต

2.1 หลักการฝังและตรวจจับลายน้ำทั่วไป

1. ออกแบบสัญญาณลายน้ำ W ซึ่งจะถูกฝังเข้าไปในสัญญาณต้นฉบับโดยทั่วไปลายน้ำจะถูกสร้างโดยอาศัยรหัสกุญแจ K ร่วมกับข้อมูลลายน้ำ I ซึ่งก็คือรหัสไบนารีที่เรียงกันซึ่งอาจเป็นชื่อเจ้าของหรือบริษัท

$$W = f_0(I, K) \tag{2.1}$$

และบางครั้งอาจจะขึ้นกับสัญญาณต้นฉบับ X ที่จะถูกฝังด้วย

$$W = f_0(I, K, X) \tag{2.2}$$

2. ออกแบบวิธีการฝังลายน้ำซึ่งจะทำการฝังลายน้ำที่สร้าง W ลงในข้อมูลต้นฉบับ X เพื่อให้ได้ข้อมูลที่ถูกฝังแล้ว Y ดังแสดงในรูปที่ 2.1

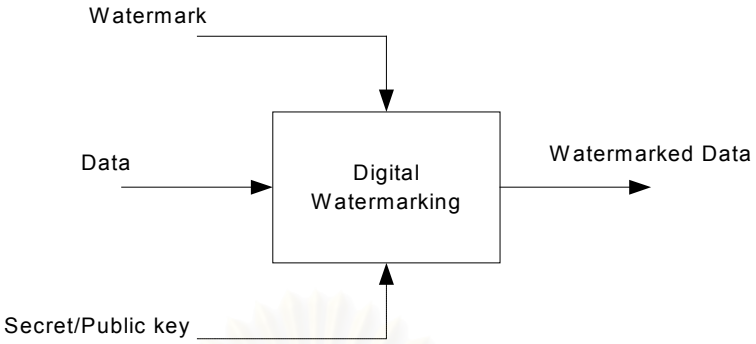
$$Y = f_1(X, W) \tag{2.3}$$

3. ออกแบบวิธีการถอดเอาข้อมูลลายน้ำจากสัญญาณที่ผสมกันอยู่โดยใช้รหัสกุญแจและใช้ต้นฉบับช่วย ดังแสดงในรูปที่ 2.2

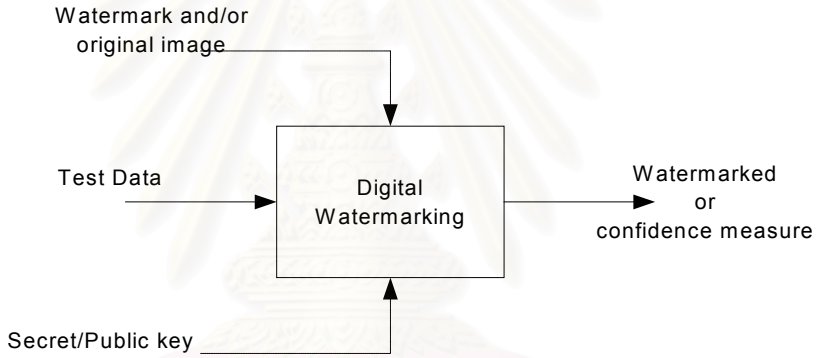
$$\hat{I} = g(X, Y, K) \tag{2.4}$$

หรือไม่ใช้ต้นฉบับช่วย

$$\hat{I} = g(Y, K) \tag{2.5}$$



รูปที่ 2.1 วิธีการฝังลายน้ำทั่วไป



รูปที่ 2.2 วิธีการถอดลายน้ำทั่วไป

2.2 งานวิจัย Digital watermarking ที่ผ่านมา

คำว่า Digital Image Watermarking ได้เกิดขึ้นเป็นครั้งแรกเมื่อปี 1993 เมื่อ Tirkel [11] ได้เสนอเทคนิคขึ้นมา 2 แบบในการฝังข้อมูลลงในภาพ ซึ่งวิธีเหล่านี้จะอาศัยการเปลี่ยนแปลง LSB ของภาพ

Image Watermarking ถ้าแบ่งตามโดเมนในการซ่อนข้อมูลก็แบ่งได้เป็น Spatial domain [7-13] และ Frequency domain [5,6,13] โดยการฝังลายน้ำใน spatial domain ได้เริ่มมีการพัฒนามาก่อน แต่เนื่องจากยังไม่ค่อยมีความทนทานมากนักจึงได้มีการพัฒนามาฝังข้อมูลใน frequency domain

Frequency domain watermarking ได้เริ่มนำมาใช้โดย Cox และ Boland [14] วิธีของ Cox จะใช้หลักการของ spread spectrum communication เพื่อที่จะฝังบิตเพียงบิตเดียวในภาพ (ตัดสินว่ามีหรือไม่มีลายน้ำที่กำหนดไว้เท่านั้น) และยังต้องใช้ภาพต้นฉบับในการตรวจสอบ Smith [15] พูดถึงวิธีที่ต้องใช้ภาพต้นฉบับว่าเป็นข้อจำกัดต่อการนำไปใช้ประโยชน์จริงซึ่งต่อมาได้มีการพัฒนาไปสู่วิธีที่ไม่ต้องใช้ต้นฉบับในการตรวจสอบ

งานวิจัยในการฝังลายน้ำส่วนมากจะใช้หลักการของ Spread Spectrum ใน SS watermarking ซึ่งก็จะคล้ายกับ spread sprectrum communication โดยที่สัญญาณ narrow band จะถูกส่งเป็นสัญญาณที่มีแบนด์วิดท์ขนาดใหญ่มากทำให้พลังงานของสัญญาณในแต่ละความถี่มีค่าน้อยมาก เช่นเดียวกันกับที่ลายน้ำจะถูก spread ไปในหลายความถี่ทำให้พลังงานในแต่ละความถี่มีค่าน้อยและทำให้ตำแหน่งในการฝังไม่เป็นที่สังเกตได้ สัญญาณลายน้ำที่จะถูกฝังลงไปจะเป็นเลขสุ่มเทียมมีพลังงานต่ำและจะถูกตรวจจับโดยอาศัยการหาค่าสหสัมพันธ์ ของสัญญาณลายน้ำที่ตรวจจับได้กับสัญญาณลายน้ำที่ฝัง ถ้าค่าสหสัมพันธ์มีค่ามากกว่า Threshold ที่กำหนดแสดงว่าสามารถถอดออกมาได้อย่างถูกต้อง

2.2.1 การฝังลายน้ำโดยวิธีของ Cox [5]

วิธีของ Cox เป็นประเภท Private watermarking Type I เนื่องจากต้องใช้รูปต้นฉบับในการตรวจสอบและบอกเพียงว่ารูปนี้ฝังลายน้ำหรือไม่เท่านั้น โดย Cox จะอาศัยหลักการของ Spread spectrum ในการใส่ลายน้ำโดยวิธีดังนี้

การฝังลายน้ำ

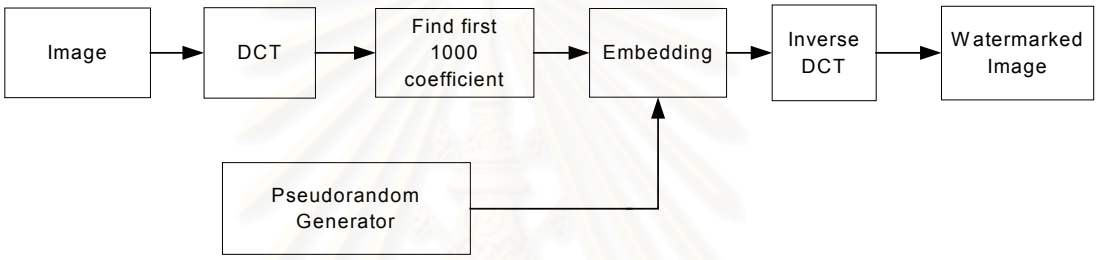
ลายน้ำ $X = x_1,...,x_n$ เป็นลำดับจำนวนสุ่มเทียมที่มีการแจกแจงปกติมีค่าเฉลี่ยเป็นศูนย์ค่าความแปรปรวนเป็นหนึ่ง $N(0,1)$ ลายน้ำถูกฝังลงไปในรูปภาพ V ในส่วนที่มีความสำคัญต่อการมองเห็นมากที่สุด (most perceptive modes of the image) หรือก็คือค่าสัมประสิทธิ์การแปลงโคซายน์กลุ่มที่มีค่ามากที่สุดเพื่อให้มีความทนทานต่อกระบวนการบีบอัดแบบสูญเสีย (lossy compression) และวิธีการประมวลผลภาพต่างๆวิธีซึ่งเสนอโดย Cox ใส่ลายน้ำโดยทำการแปลงโคซายน์กับรูปต้นฉบับทั้งรูป แล้วเปลี่ยนแปลงค่าสัมประสิทธิ์ซึ่งมีค่ามากที่สุด 1000 ตัวแรก (ไม่รวม DC coefficient) โดยมีเทคนิคการใส่ที่เสนอดังนี้

$$v'_i = v_i + \alpha x_i \qquad (2.6)$$

$$v'_i = v_i(1 + \alpha x_i) \tag{2.7}$$

$$v'_i = v_i e^{\alpha x_i} \tag{2.8}$$

โดยที่ α คือความแรงในการฝังลายน้ำซึ่งสามารถปรับค่าได้เพื่อให้ได้ความทนทานและการมองไม่เห็นที่พอเหมาะ ถ้า α มีค่ามากลายน้ำจะมีความทนทานสูงแต่ในขณะเดียวกันก็จะทำให้เกิดการมองเห็นมากไปด้วยและ v_i คือส่วนประกอบของสเปกตรัมที่ผ่าน DCT ซึ่งโดยทั่วไปจะนิยมใช้สมการที่ (2.7) จากนั้นทำการแปลงกลับโคซายน์ก็จะได้รูปที่ฝังลายน้ำแล้ว V'

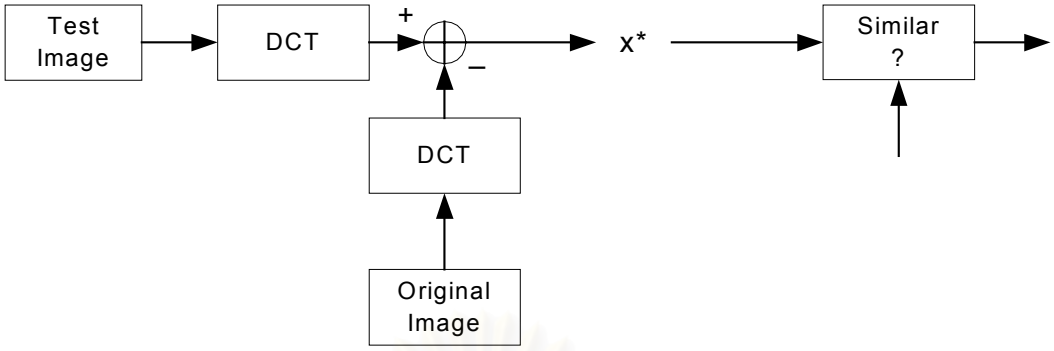


รูปที่ 2.3 การฝังลายน้ำวิธีของ Cox

การตรวจจับลายน้ำ

การตรวจจับจะต้องใช้รูปต้นฉบับ โดยนำมาลบออกจากรูปทดสอบแล้วทำการแปลงโคซายน์กับผลต่างนั้นซึ่งพิจารณาสมการจะได้ค่าผลต่างคือ $\alpha v_i x_i$ เมื่อหารด้วย $\alpha_i v_i$ (เนื่องจากมีรูปต้นฉบับ) ก็จะสามารถหาค่าประมาณของลายน้ำที่ตรวจจับได้ในรูปทดสอบคือ $X*$ ซึ่งนำมาหาค่าดัชนีความเหมือนกัน (similarity index) กับลายน้ำที่ฝังลงไปได้ดังนี้

$$sim(X, X^*) = \frac{X * X}{\sqrt{X * X *}} \tag{2.9}$$



รูปที่ 2.4 การตรวจจับลายน้ำวิธีของ Cox

2.2.2 การฝังลายน้ำโดยวิธีของ Piva [6]

วิธีของ Piva เป็นประเภท Semi-private watermarking เนื่องจากไม่ต้องใช้รูปต้นฉบับในการตรวจสอบและบอกเพียงว่ารูปนี้ฝังลายน้ำหรือไม่เท่านั้น โดย Piva จะอาศัยหลักการของ Spread spectrum ในการใส่ลายน้ำเหมือน Cox แต่จะดัดแปลงดังนี้

ลักษณะของลายน้ำ

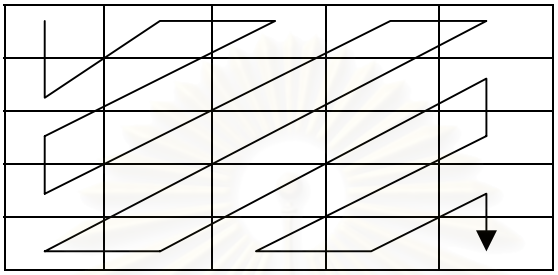
ลายน้ำ $X = \{x_1, x_2, ..., x_M\}$ เป็นลำดับจำนวนสุ่มเทียม (pseudorandom number sequence) ความยาว M สร้างโดย multiplicative congruential algorithm แต่ละค่า x_i เป็นเลขจำนวนจริงแบบสุ่ม (random real number) ที่มีการแจกแจงปกติ (normal distribution) มีค่าเฉลี่ยเป็นศูนย์และมีค่าความแปรปรวนเป็นหนึ่ง

การฝังลายน้ำ

1. จากรูปต้นฉบับ gray scale image I ขนาด N x N นำมาทำการแปลงโคซายน์ทั้งรูปแล้วเรียงสัมประสิทธิ์ที่ได้แบบซิกแซ็ก T ดังแสดงในรูปที่ 2.5
2. ฝังลายน้ำโดยการเปลี่ยนแปลงค่าสัมประสิทธิ์ที่เรียงแบบซิกแซ็กแล้วตัวที่ $L+1$ ถึง $L+M$ ($T = [t_1, t_s, ..., t_M]$) ซึ่งก็คือสัมประสิทธิ์ในกลุ่มแรกๆที่มีความถี่ต่ำเพื่อให้ลายน้ำทนทานต่อกระบวนการต่างๆแต่ข้าม L ตัวแรกไปเพื่อไม่ให้เกิดการมองเห็นอย่างชัดเจน ลายน้ำ X จะถูกฝังไปใน T ตามสมการที่ (2.10)

$$t'_i = t_i + \alpha | t_i | x_i \quad (2.10)$$

จากนั้นเวคเตอร์ T' จะถูกใส่กลับเข้าไปในลำดับแบบซิกแซ็กจากนั้นทำการแปลงกลับโคซายน์ก็จะได้รูปที่ถูกฝังด้วยลายน้ำ (Watermarked Image) I'



รูปที่ 2.5 การจัดเรียงสัมประสิทธิ์แบบซิกแซก

การตรวจจับลายน้ำ

รูปทดสอบ I^* ขนาด N x N ทำการแปลงโคซายน์ทั้งรูปแล้วจัดเรียงสัมประสิทธิ์แบบซิกแซ็ก เวคเตอร์ $T^* = \{t^*_{L+1}, t_{L+2}*, ..., t^*_{L+M}\}$ เป็นเวคเตอร์ของสัมประสิทธิ์ตัวที่ $L+1$ ถึง $L+M$ ของรูปทดสอบหาค่าสหสัมพันธ์ (correlation) z ระหว่างสัมประสิทธิ์ DCT ของรูปทดสอบกับลายน้ำที่จะนำมาตรวจสอบ Y ได้ดังนี้

$$z = \frac{Y.T^*}{M} = \frac{1}{M}\sum_{i=1}^{M} y_i t^*_{L+i} \quad (2.11)$$

ค่า z จะใช้ตัดสินว่าในรูป I^* มีลายน้ำ Y อยู่หรือไม่โดยเปรียบเทียบกับค่าจุดเริ่มเปลี่ยน (threshold) T_z

การหาค่าจุดเริ่มเปลี่ยน T_z

กรณีที่รูปทดสอบ I^* เหมือนกับรูปที่ถูกฝังด้วยลายน้ำ I' และเมื่อละจำนวนสัมประสิทธิ์ที่ถูกข้าม L จะได้ว่า

$$t^*_i = t'_i = t_i + \alpha |t_i| x_i \quad (2.12)$$

$$z = \frac{1}{M}\sum_{i=1}^{M}\left(t_i y_i + \alpha|t_i|x_i y_i\right) \tag{2.13}$$

สามารถหาค่าเฉลี่ยและค่าความแปรปรวนของ z ภายใต้สมมติฐานที่ว่า t_i และ x_i มีค่าเฉลี่ยเป็นศูนย์และเป็นอิสระต่อกัน ได้ดังนี้

$$\mu_z = \begin{cases} \alpha\mu_{|t|} & \text{if X = Y} \\ 0 & \text{if X} \neq \text{Y} \\ 0 & \text{if no mark is present} \end{cases} \tag{2.14}$$

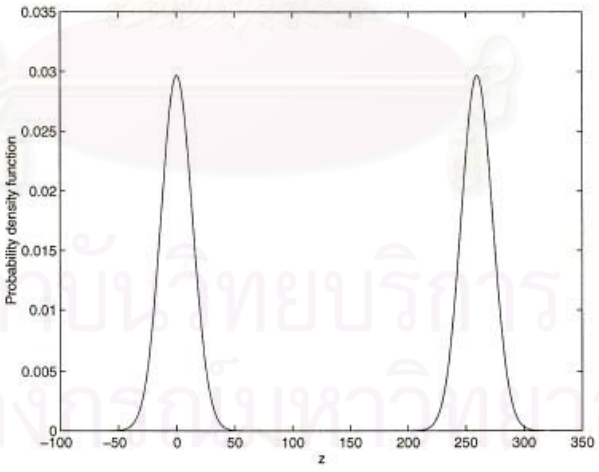
$$\sigma_z^2 \approx \frac{\sigma_t^2}{M} \tag{2.15}$$

กรณีลายน้ำที่นำมาตรวจสอบไม่ตรงกับลายน้ำในรูปทดสอบ : z_1 $\mu_{z_1} = 0$

กรณีลายน้ำที่นำมาตรวจสอบตรงกับลายน้ำในรูปทดสอบ : z_2 $\mu_{z_2} = \alpha\mu_{|t|}$

โดย $$\sigma_{z_1}^2 = \sigma_{z_2}^2 \approx \frac{\sigma_t^2}{M} \tag{2.16}$$

เขียนการแจกแจงของตัวแปรสุ่ม z_1 และ z_1 ได้ดังรูป



รูปที่ 2.6 การแจกแจงของตัวแปรสุ่ม z_1 และ z_2

เพื่อให้ความน่าจะเป็นของการตรวจวัดความผิดพลาด (error probability) มีค่าต่ำระยะทางระหว่างการกระจายของตัวแปรทั้งสองหรือค่า $\frac{\mu_z}{\sigma_z} = \frac{\alpha\mu_{|t|}}{\frac{\sigma_t^2}{M}} = M\alpha\frac{\mu_{|t|}}{\sigma_t^2}$ ควรมีค่ามากและเนื่องจากเมื่อจำนวนสัมประสิทธิ์ที่ถูกข้าม L เพิ่มขึ้น $\mu_{|t|}$ และ $\sigma^2{}_t$ จะลดลงแต่ $\sigma^2{}_t$ จะลดลงด้วยอัตราที่เร็วกว่าดังนั้นจึงควรใช้ค่า L มากๆและจากสมการจะเห็นว่าค่า M ก็ควรมีค่ามากเช่นกัน สามารถหาค่าจุดเริ่มเปลี่ยน T_z ได้ดังนี้

$$T_z = \frac{\mu_{z_2}}{2} = \frac{\alpha\mu_{|t|}}{2} = \frac{\alpha}{2M}\sum_{i=1}^{M}|t_i'| \tag{2.17}$$

แต่จากผลการทดสอบพบว่าถ้ารูปภาพถูกรบกวนด้วยวิธีการที่จงใจหรือไม่ก็ตามจะทำให้ σ_{z_2} เพิ่มขึ้นอย่างมาก นั่นคือ z_1 และ z_2 ยังมีการกระจายแบบเดิมแต่ z_2 จะมีค่าความแปรปรวนสูงขึ้น ดังนั้นค่าจุดเริ่มเปลี่ยนที่ใช้จึงควรเข้าใกล้ศูนย์มากกว่าที่จะอยู่ตรงกลาง Piva ได้เสนอให้ใช้ค่าจุดเริ่มเปลี่ยนเป็น

$$T_z = \frac{\alpha}{3M}\sum_{i=1}^{M}|t^*{}_i| \tag{2.18}$$

ซึ่งให้ผลการทดสอบเป็นที่น่าพอใจ

Visual Masking

หลังจากที่ได้รูปที่ถูกฝังด้วยลายน้ำ I' แล้วสามารถปรับให้ลายน้ำเข้ากับรูปที่ถูกฝังมากขึ้นเพื่อช่วยเพิ่มการมองไม่เห็น (invisibility) ได้โดยใช้ spatial masking characteristics ของ Human Visual System (HVS) โดยในขั้นตอนของการใส่ลายน้ำ รูปต้นฉบับ I และรูปที่ถูกฝังด้วยลายน้ำ I' จะถูกนำมาบวกกันทีละจุดภาพดังนี้

$$y_{ij}'' = y_{ij}(1-\beta_{ij}) + \beta_{ij}y_{ij}' = y_{ij} + \beta_{ij}(y_{ij}' - y_{ij}) \tag{2.19}$$

การหาค่า β_{ij} สำหรับจุดภาพ y_{ij} พิจารณาบล็อกขนาด R x R หาค่าความแปรปรวนของจุดภาพนั้นแล้วทำการนอร์มอลไลซ์ (normalized) ด้วยความแปรปรวนที่มากที่สุด β_{ij} คือค่าความแปรปรวนนอร์มอลไลซ์ (normalized variance) ของจุดภาพ y_{ij} บริเวณที่ทนต่อสัญญาณ

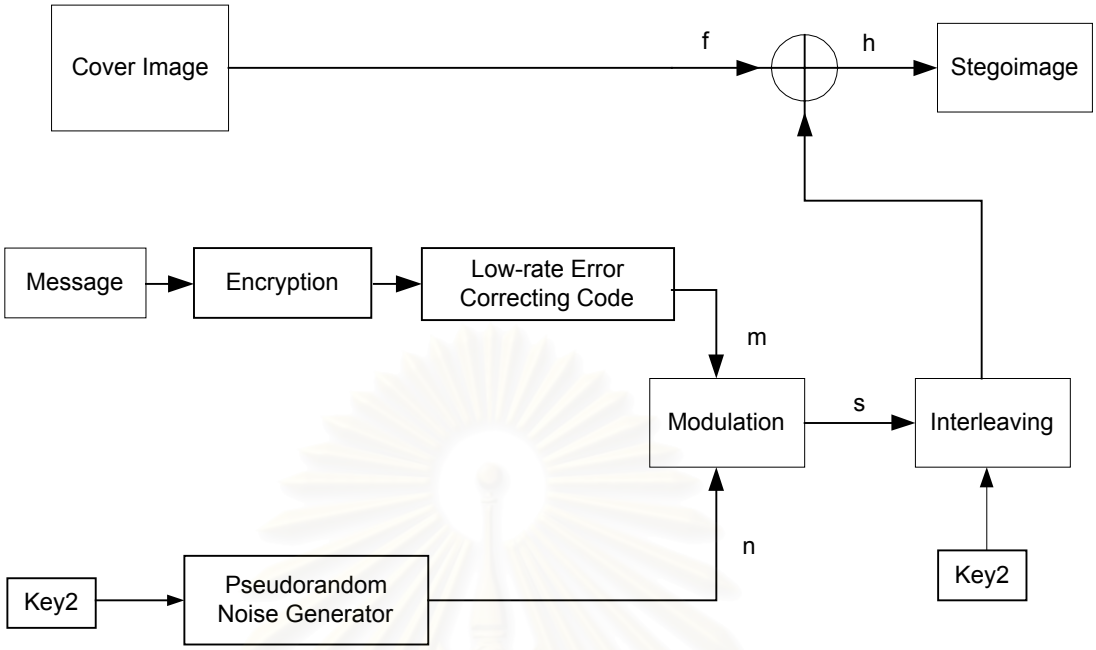
รบกวนได้มาก (บริเวณที่มีรายละเอียดสูง) $\beta_{ij} \approx 1$ ดังนั้น $y''_{ij} \approx y'_{ij}$ นั่นคือสามารถใส่ลายน้ำได้เต็มที่ ส่วนบริเวณที่ทนต่อสัญญาณรบกวนได้น้อย $\beta_{ij} \approx 0$ จะได้ $y''_{ij} \approx y_{ij}$ สามารถใส่ลายน้ำได้เพียงเล็กน้อยเท่านั้น จะเห็นได้ว่าเมื่อใช้วิธีนี้จะช่วยให้สามารถเพิ่มความเข้มของลายน้ำ α ได้โดยไม่ทำให้เกิดการมองเห็นมาก และทำให้ความพยายามที่จะเอาลายน้ำออกทำได้ยากขึ้นด้วย

2.2.3 การฝังลายน้ำโดยวิธีของ Lisa [9]

วิธีของ Lisa เป็นประเภท Public watermarking เนื่องจากไม่ต้องใช้รูปต้นฉบับในการตรวจสอบและเป็นการฝังบิตข้อมูลลงไปจริงๆโดย Lisa จะอาศัยหลักการของ Spread spectrum ในการใส่ลายน้ำเหมือน Cox แต่จะทำใน Spatial Domain และใช้รหัสแก้ไขความผิดพลาดร่วมด้วย

การซ่อนข้อมูล

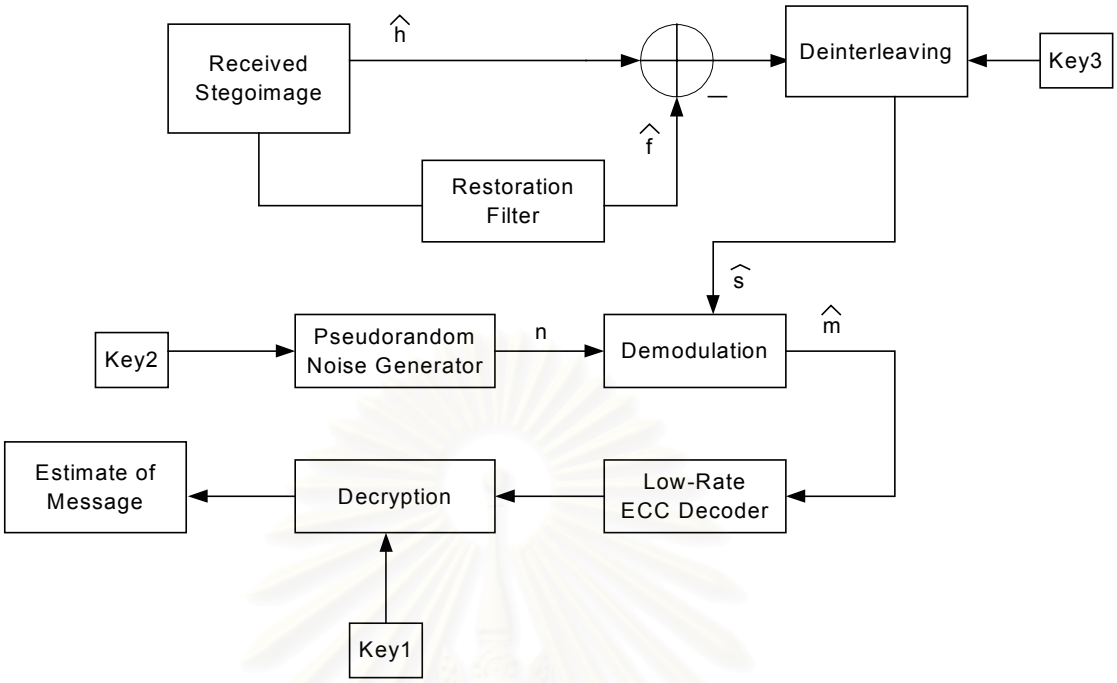
ในที่นี้ Lisa Marvel เรียกวิธีที่เสนอว่าการซ่อนข้อมูล (Data Hiding) เนื่องจากว่าคำนึงถึงเฉพาะปริมาณข้อมูลที่จะฝังลงไปให้ได้มากที่สุดโดยไม่คำนึงถึงความทนทาน ข้อมูลที่จะใส่ซึ่งจะเป็นบิตศูนย์หรือหนึ่งเรียงกันจำนวนหนึ่งจะถูกเข้ารหัสลับด้วยกุญแจตัวที่ 1 หลังจากนั้นจะนำมาเข้ารหัสด้วยรหัสแก้ไขความผิดพลาดก็จะได้ข้อมูลที่เข้ารหัสแล้ว m ในขณะเดียวกันก็จะทำการสร้างลำดับจำนวนสุ่มเทียม (pseudorandom number sequence) ได้ spread sequence n โดยใช้กุญแจตัวที่ 2 เสร็จแล้วนำ m และ n มามอดูเลตกันจากนั้นนำเอาค่าที่ผ่านการมอดูเลต s มาทำการวางสลับโดยใช้กุญแจตัวที่ 3 สัญญาณที่ได้นั้นก็จะนำมาบวกกับรูปต้นฉบับ f ก็จะได้ภาพที่ผ่านการใส่ลายน้ำมา h



รูปที่ 2.7 การซ่อนข้อมูลโดยวิธีของ Lisa

การดีเทกต์ข้อมูล

การดีเทกต์ข้อมูลในที่นี้จะไม่ใช้ภาพต้นฉบับการถอดรหัสขั้นแรกก็จะต้องใช้เทคนิคของ image restoration มาทำการประมาณภาพต้นฉบับจากภาพที่ทำการฝังข้อมูลที่ได้รับมาซึ่งอาจผ่านการเปลี่ยนแปลงมาแล้ว ซึ่งก็ทำได้เช่นใช้ Adaptive Wiener Filter ความต่างของทั้งสองนี้จะนำมาทำวางสลับกลับ (deinterleaving) โดยใช้กุญแจตัวที่สามเพื่อหาค่าประมาณของสัญญาณที่ใส่เข้าไป จากนั้นจะทำการสร้างลำดับจำนวนสุ่มเทียม (pseudorandom number sequence) โดยใช้กุญแจตัวที่ 2 นำสัญญาณทั้งสองมาดีมอดูเลตก็จะได้สัญญาณประมาณของข่าวสาร นำรหัสแก้ความผิดพลาดมาถอดรหัสข่าวสารนี้เสร็จแล้วนำไปถอดรหัสลับโดยใช้กุญแจตัวที่ 1 ก็จะได้ข้อมูลที่ใส่เข้าไป



รูปที่ 2.8 การถอดรหัสข้อมูลโดยวิธีของ Lisa

Image Restoration Filter ที่จะนำมาใช้ประกอบในขั้นตอนการถอดลายน้ำคือ Adaptive Wiener Filter (AW filter) โดยจะใช้ในการลดปริมาณของสัญญาณรบกวนที่บวกเข้าไปในภาพต้นฉบับ AW filter จะทำการรักษาสัญญาณไว้ขณะที่ทำการกำจัดสัญญาณรบกวนในภาพที่เสีย เนื่องจากภาพต้นฉบับและสัญญาณที่ใส่เข้าไปเป็นอิสระเชิงเส้นกันดังนั้นจะได้ความผิดพลาดกำลังสองเฉลี่ย (MSE) ต่ำสุดจากการฟิลเตอร์ภาพโดยใช้ AW filter ผลตอบสนองเชิงความถี่ของฟิลเตอร์นั้นขึ้นกับสเปกตรัมกำลังของภาพต้นฉบับและสัญญาณรบกวนตามสมการที่ (2.20) โดยที่ P_f คือสเปกตรัมกำลังงานของภาพต้นฉบับ f และ P_s คือสเปกตรัมกำลังงานของสัญญาณรบกวน s

$$H(\omega_1,\omega_2)=\frac{P_f(\omega_1,\omega_2)}{P_f(\omega_1,\omega_2)+P_s(\omega_1,\omega_2)} \qquad (2.20)$$

สเปกตรัมกำลังของ AWGN ซึ่งมีค่าคงที่และเป็นอิสระของ ω_1 และ ω_2 จะรู้ได้ที่เครื่องรับ แม้ว่าคุณลักษณะของสัญญาณรบกวนที่บวกเข้าไปจะไม่เปลี่ยนแต่คุณลักษณะของภาพจะเปลี่ยนไปตามแต่ละบริเวณของภาพ เช่นภาพที่มีฉากหลังเรียบกับฉากหน้ามีรายละเอียดมากสเปกตรัมกำลังงานของทั้งสองก็จะต่างกัน เพื่อชดเชยการเปลี่ยนคุณลักษณะของภาพ AW filter จะเป็น

ฟิลเตอร์ที่มีพารามิเตอร์เปลี่ยนแปลงตามแต่ละส่วนของภาพโดยการเปลี่ยนนี้อาจจะเป็นจุดต่อจุด หรือเป็นบล็อกต่อบล็อกก็ได้

สเปกตรัมกำลังของรูปต้นฉบับนั้นจะไม่ทราบที่เครื่องรับต้องประมาณจากรูปที่ฝังลายน้ำที่รับมา $\hat{h}$ พิจารณาที่บริเวณท้องถิ่น (local region) ของภาพบริเวณหนึ่งถ้าสมมุติว่าสัญญาณภาพต้นฉบับ $f(n_1, n_2)$ ของบริเวณนั้นคงที่จะสามารถจำลองรูปแบบตามสมการที่ (2.21) โดยที่ m_f และ σ_f คือค่าเฉลี่ยบริเวณนั้นและค่าเบี่ยงเบนมาตรฐานของภาพต้นฉบับ, w คือ white noise ที่มีค่าเฉลี่ยเป็นศูนย์และมีความแปรปรวนเป็นหนึ่ง

$$f(n_1, n_2) = m_f + \sigma_f w(n_1, n_2) \tag{2.21}$$

ในการประมาณค่าเหล่านี้จากภาพที่ใส่ลายน้ำจะพิจารณาว่าเมื่อค่าเฉลี่ยของสัญญาณรบกวนที่ใส่เป็นศูนย์ซึ่งเป็นกรณีที่ฝัง AWGN ลงไป m_f จะเหมือนกันค่าเฉลี่ยของบริเวณนี้ของภาพที่ใส่ลายน้ำ m_h และเนื่องจาก s เป็นการบวก σ_h^2 จะสามารถเขียนได้ตามสมการที่ (2.22) และค่าประมาณของ $\hat{\sigma}_f^2$ สามารถหาได้จากสมการที่ (2.23) โดยที่ $\sigma_{\hat{h}}^2$ คือความแปรปรวนบริเวณท้องถิ่นของภาพที่ได้รับมาตรวจสอบลายน้ำ ภายในบริเวณท้องถิ่นนี้ transfer function ของ AW Wiener filter จะเขียนได้ตามสมการที่ (2.24) และภาพที่กู้แล้ว $\hat{f}$ ก็จะหาได้ตามสมการที่ (2.25)

$$\sigma_h^2 = \sigma_f^2 + \sigma_s^2 \tag{2.22}$$

$$\hat{\sigma}_f^2(n, n_2) = \begin{cases} \sigma_{\hat{h}}^2(n_1, n_2) - \sigma_s^2, & if\ \sigma_{\hat{h}}^2(n_1, n_2) > \sigma_s^2 \\ 0, & otherwise, \end{cases} \tag{2.23}$$

$$H(\omega_1, \omega_2) = \frac{P_f(\omega_1, \omega_2)}{P_f(\omega_1, \omega_2) + P_s(\omega_1, \omega_2)} = \frac{\hat{\sigma}_f^2}{\hat{\sigma}_f^2 + \sigma_s^2} \tag{2.24}$$

$$\hat{f}(n_1, n_2) = m_{\hat{h}} + (\hat{h}(n_1, n_2) - m_{\hat{h}}) * \frac{\hat{\sigma}_f^2}{\hat{\sigma}_f^2 + \sigma_s^2} \delta(n_1, n_2) \tag{2.25}$$

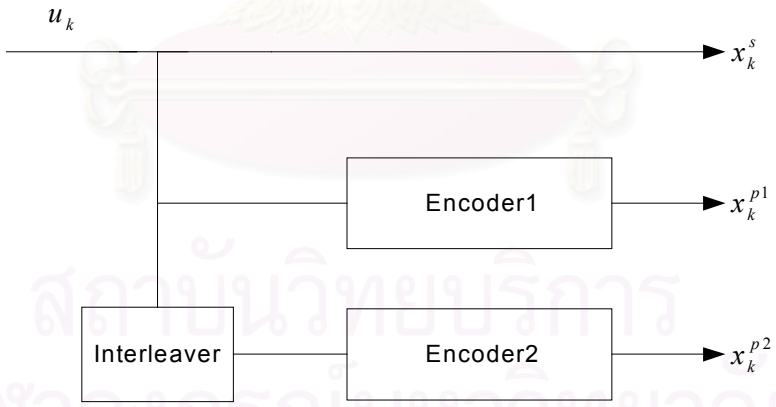
ภาพที่กู้ได้จะถูกเปลี่ยนแปลงตามความสัมพันธ์ระหว่าง $\hat{\sigma}_f^2$ ซึ่งประมาณได้จากบริเวณท้องถิ่นของภาพที่ใส่ลายน้ำและ σ_s^2 ถ้า σ_s^2 มากกว่า $\hat{\sigma}_f^2$ ก็จะมีการเปลี่ยนแปลงมาก แต่กลับกันถ้า $\hat{\sigma}_f^2$ มากกว่า σ_s^2 จะมีการเปลี่ยนแปลงน้อยมาก

บทที่ 3
รหัสเทอร์โบ

รหัสเทอร์โบ [16-19] ได้ถูกคิดค้นขึ้นโดย C. Berrou, A. Glavieus และ P. Thitimajshina การเข้ารหัสเทอร์โบเป็นการเข้ารหัสแบบหนึ่งที่ได้รับความนิยมมากสำหรับการเข้ารหัสช่องสัญญาณ (Channel Coding) เนื่องจากการเข้ารหัสนี้ให้อัตราความผิดพลาดของบิตที่ต่ำ หลักการของการเข้ารหัสเทอร์โบนั้นเป็นแบบที่นำเครื่องเข้ารหัสมาต่อขนานกัน (parallel concatenation) ส่วนการถอดรหัสนั้นจะนำเอา Bahl algorithm มาใช้ในการถอดรหัสตัว RSC code โดยที่ผลลัพธ์ที่ออกจากตัวถอดรหัสจะเป็นแบบ soft output ซึ่งจะนำไปใช้ในการป้อนกลับเพื่อให้มีการคำนวณแบบวนซ้ำหลายๆรอบเพื่อแก้ไขและเพิ่มความเชื่อมั่นในการตัดสินใจของระดับสัญญาณ

3.1 การเข้ารหัสเทอร์โบ

การเข้ารหัสเทอร์โบนั้นจะนำเอาตัวเข้ารหัสแบบคอนโวลูชันที่มีการป้อนกลับ(RSC:Recursive Systematic Convolutional encoder) 2 ตัวมาต่อกันแบบขนานโดย RSC encoder ซึ่งโดยปรกติ RSC encoder ทั้งสองจะเหมือนกันแต่ RSC encoder ตัวที่ 2 จะรับเอาบิตข้อมูลที่ผ่านการสลับโดยตัววางสลับ (interleaver)

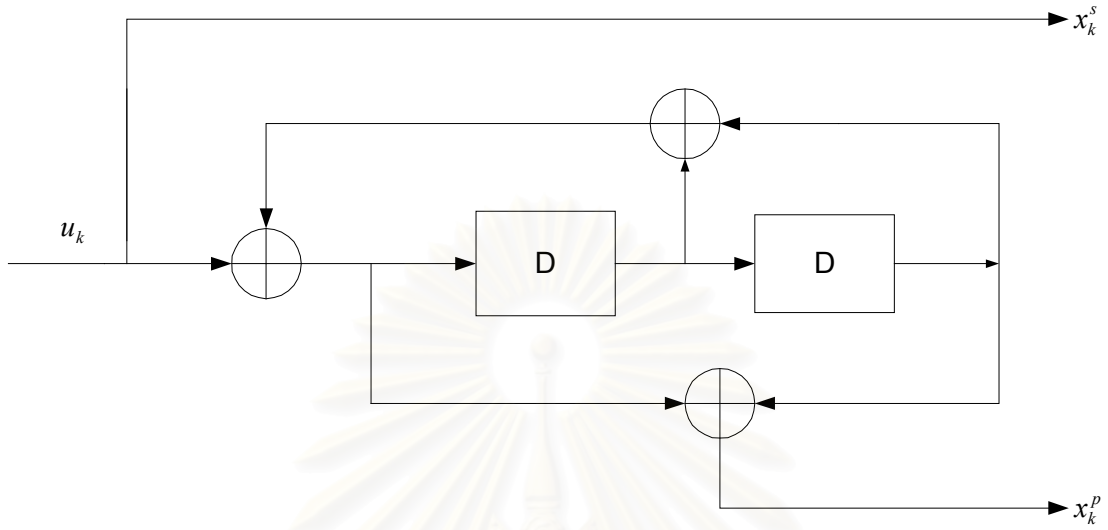


รูปที่ 3.1 ตัวเข้ารหัสเทอร์โบพื้นฐาน

3.1.1 Recursive Systematic Convolutional (RSC) Encoder

RSC encoder นั้นได้มาจากการที่ nonrecursive nonsystematic (conventional) convolutional encoder ถูกป้อนกลับผลลัพธ์ที่เข้ารหัสแล้วมายังฝั่งขาเข้า ตามรูปที่ 3.2 จะแสดง RSC encoder

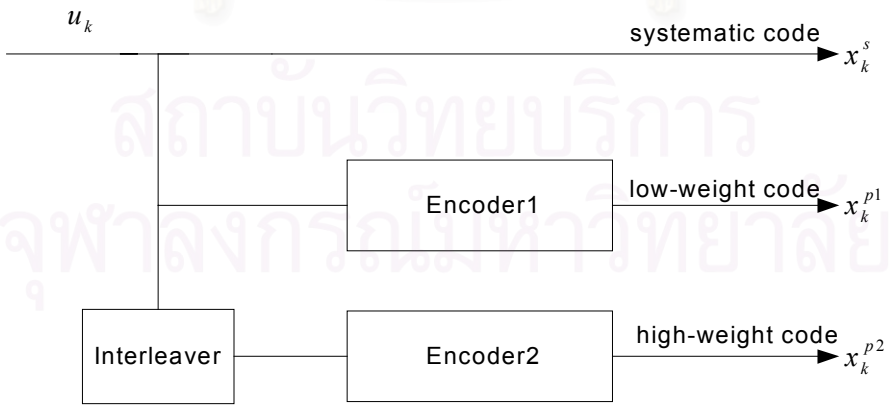
RSC encoder จะถูกแสดงด้วย G=$[1, g_2 / g_1]$ โดยผลลัพธ์ตัวที่ 1 (แสดงด้วย g_1) จะถูกป้อนกลับมาที่ขาเข้าโดย $g_1 = [111]$ และ $g_2 = [101]$



รูปที่ 3.2 RSC encoder ที่มีอัตราเข้ารหัส r=1/2 K=3

3.1.2 **ตัววางสลับ** (Interleaver)

ตัววางสลับเป็นส่วนสำคัญส่วนหนึ่งในตัวเข้ารหัสเทอร์โบโดยจะทำหน้าที่สุ่มข้อมูลด้านเข้าก่อนเข้า RSC encoder ตัวที่ 2 และยังช่วยเพิ่ม weight สำหรับรหัสตามรูป 3.3



รูปที่ 3.3 ตัววางสลับเพิ่ม code weight ของ RSC encoder 2 เมื่อเทียบกับ RSC encoder1

ตามรูปบิตข้อมูล u_k จะให้ ลำดับรหัส RSC แบบมีน้ำหนักต่ำ (low-weight recursive convolutional code sequence) x_k^{p1} สำหรับ RSC encoder ตัวที่ 1 เพื่อที่จะหลีกเลี่ยงให้ RSC encoder ตัวที่ 2 ให้ลำดับรหัส RSC แบบมีน้ำหนักต่ำออกมาเหมือนกัน ตัววางสลับจะทำการสลับบิตข้อมูลด้านเข้า u_k เพื่อที่จะให้ได้ลำดับที่ต่างออกไปโดยหวังว่าจะให้ลำดับรหัส RSC แบบมีน้ำหนักสูง (high-weight recursive convolutional code sequence) x_k^{p2} ดังนั้นน้ำหนัก ของรหัสเทอร์โบจะเป็นแบบกลางๆโดยรวมเอารหัสที่มีน้ำหนักต่ำของตัวเข้ารหัสตัวที่ 1 และรหัสที่มีน้ำหนักสูงของตัวเข้ารหัสตัวที่ 2

ตัววางสลับจะมีผลต่อประสิทธิภาพของรหัสเทอร์โบเพราะว่ามีผลโดยตรงต่อระยะทางของรหัส โดยหลีกเลี่ยงรหัสที่มีน้ำหนักต่ำ อัตราความผิดพลาด (BER) ของรหัสเทอร์โบจะสามารถดีขึ้นได้อย่างมีนัยสำคัญ ตัววางสลับที่นิยมใช้เช่น block interleaver, random interleaver

3.1.3 ขั้นตอนการเข้ารหัส

ตัวอย่างของการเข้ารหัสเทอร์โบถูกแสดงตามรูป บล็อกของตัววางสลับแสดงด้วย α และผลลัพธ์ของตัววางสลับคือ $\overline{u}_i$ ฟังก์ชั่น α จะบอกถึงการวางสลับตามนี้

$$\overline{u}_{\alpha(i)} = u_i \tag{3.1}$$

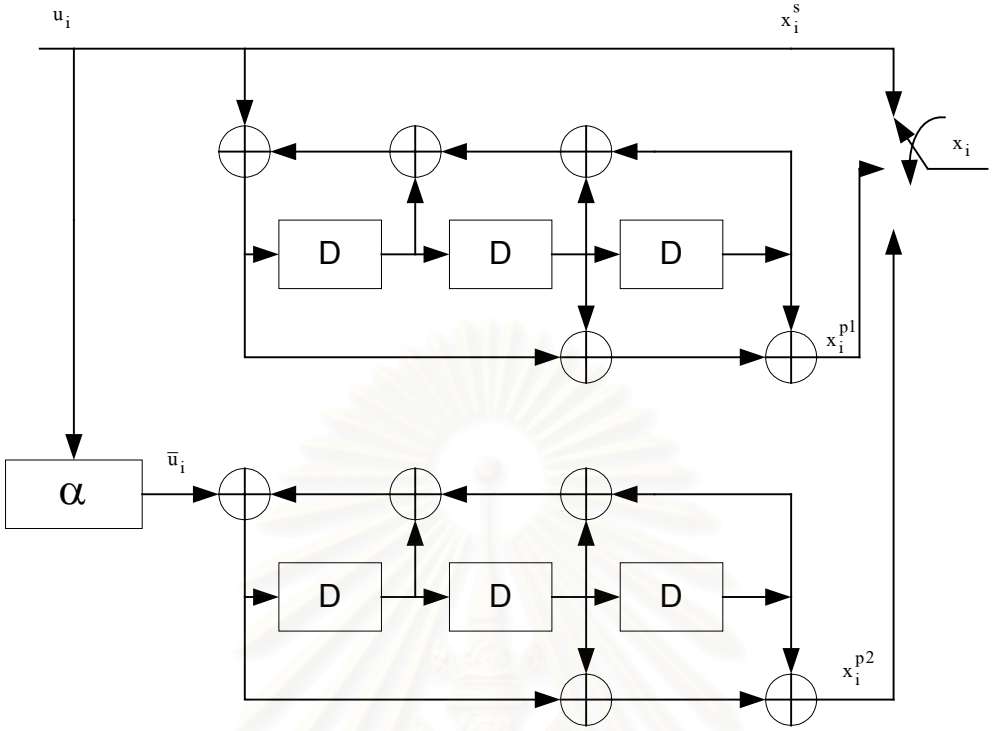
โดยที่ $i \in (0,...,L-1)$ และการวางสลับกลับ (deinterleaving) สามารถกำหนดได้ดังนี้

$$u_{\alpha^{-1}(i)} = \overline{u}_i \tag{3.2}$$

systematic output ของตัวเข้ารหัสเทอร์โบ x^s นั้นเอามาจาก RSC encoder ตัวบน และส่วนที่เป็น parity output x^{p1} และ x^{p2} เอามาจาก parity output ของ RSC encoder ตัวที่ 1 และ 2 ตามลำดับ ผลลัพธ์ทั้งสามสายนี้จะถูกมามัลติเพล็กซ์เป็นรหัส

$$x = (x_0^s, x_0^{p1}, x_0^{p2}, ..., x_{L-1}^s, x_{L-1}^{p1}, x_{L-1}^{p2}) \tag{3.3}$$

code rate ของรหัสเทอร์โบ (ไม่คำนึงถึง trellis termination) คือ 1/3 ซึ่งก็เหมือนกับรหัสคอนโวลูชัน โดยสามารถเพิ่มอัตราเข้ารหัสเทอร์โบได้โดยการทำ puncturing



รูปที่ 3.4 ตัวเข้ารหัสเทอร์โบ

3.2 การถอดรหัสเทอร์โบ

รหัสเทอร์โบนั้นจะใช้อัลกอริทึมในการถอดรหัสแบบ soft-decision ซึ่งเป็นไปได้เพราะตัวถอดรหัสเทอร์โบสร้างจากตัวถอดรหัสที่มีการวัดความน่าเชื่อถือของการใช้บิตร่วมกัน ซึ่งการที่มีการใช้ข้อมูลร่วมกันนั้นเป็นไปได้จากการวางสลับในตัวเข้ารหัสซึ่งจะสร้าง สายของ parity สองสายที่เกี่ยวข้องกันอย่างหลวมๆ

ตัวถอดรหัสนั้นจะมีอัลกอริทึมในการถอดรหัสสองแบบคือ Soft Output Viterbi Algorithm (SOVA) และ Bahl, Cocke, Jelinek และ Rajiv (BCJR) แต่ว่าเนื่องจากต้องการให้อัตราความผิดพลาดต่ำวิธี BCJR จึงนิยมใช้ในรหัสเทอร์โบเพราะว่าใช้กฎการถอดรหัสแบบ MAP (Maximum a posteriori) ขณะที่ SOVA นั้นจะเป็นวิธีประมาณของ MAP และจะให้ผลของอัตราความผิดพลาดแย่กว่าแต่จะมีความซับซ้อนน้อยกว่าเมื่อเทียบกับ BCJR

3.2.1 BCJR Algorithm

เพื่อให้เข้าใจการถอดรหัสเทอร์โบจำเป็นต้องเข้าใจถึงอัลกอริทึม BCJR พิจารณาตัวเข้ารหัสคอนโวลูชันอย่างมีระบบที่มีอัตราเข้ารหัส 1/2 ที่มีขนาดหน่วยความจำ ν โดยมีลำดับบิตข้อมูลความยาว N เพื่อให้ได้รหัสแบบบล็อก (2N,N) ถ้าตัวเข้ารหัสนั้นให้สถานะเริ่มต้นเป็นศูนย์

สำหรับการเริ่มต้นของบิตข้อมูลในแต่ละเฟรมรหัสแบบบล็อกก็จะเป็นเชิงเส้น เนื่องจากเป็นตัวเข้ารหัสแบบมีระบบผลลัพธ์ของตัวเข้ารหัสจะประกอบด้วยเฟรมข้อมูลด้านเข้า $u = x^s$ และลำดับของ parity x^P ทั้งบิตข้อมูลและลำดับของ parity จะเป็นเวกเตอร์ขนาด (1 x N) โดยใช้สัญลักษณ์ไบนารีในการส่ง

ถ้าเป็นการส่งสัญญาณแบบ BPSK บนช่องสัญญาณ AWGN สัญลักษณ์ที่ได้รับจากบิตข้อมูลด้านเข้าและ parity bit จะได้เป็น

$$y_k^s = \sqrt{E_s}(2 \cdot u_k - 1) + n_k^s \tag{3.4}$$

$$y_k^p = \sqrt{E_s}(2 \cdot x_k^p - 1) + n_k^p \tag{3.5}$$

ตัวยกที่ใช้จะใช้แยกระหว่างสัญลักษณ์ด้านเข้า (y_k^s) และสัญลักษณ์ parity (y_k^p) สำหรับช่องสัญญาณ AGWN สัญญาณรบกวนที่เพิ่มเข้าไป $(n_k^s$ และ $n_k^p)$ คือตัวแปรสุ่มแบบเกาส์ที่มีค่าเฉลี่ยเป็นศูนย์และมีความแปรปรวน $N_0/2$ และจะเขียนรหัสที่ได้รับดังนี้ $Y = [y^s | y^p]$

สำหรับรหัสเดี่ยวๆอัลกอริทึมของ BCJR จะลดอัตราความผิดพลาดของสัญลักษณ์สำหรับการถอดรหัสเทรลิสและรหัสแบบบล็อก โดยหลังจากได้รับ Y จะเป็นหน้าที่ของตัวถอดรหัสที่จะตัดสินว่าบิตข้อมูลด้านเข้า u_k น่าจะเป็นอะไรจากสัญลักษณ์ที่ได้รับ เนื่องจากบิตข้อมูลด้านเข้าอยู่ในรูปของไบนารี (0 และ 1) ดังนั้นจึงสะดวกที่จะใช้ Log-likelihood ratio (LLR) ในการตัดสินบิต โดย LLR สำหรับสัญลักษณ์ด้านเข้าที่เวลา k กำหนดเป็น

$$L(k) = \log\left[\frac{P(u_k = 1|Y)}{P(u_k = 0|Y)}\right] \tag{3.6}$$

จากสมการที่ (3.6) $P(u_k = i|Y)$ คือ posteriori probability ของ $u_k = i$ เมื่อรู้ข้อมูลที่ได้รับ Y ตัวถอดรหัสจะทำการประมาณหา $\hat{u}_k$ ของบิตข้อมูลโดยจะใช้การเปรียบเทียบ $L(k)$ กับค่า threshold สำหรับช่องสัญญาณ AWGN จะทำการตัดสินบิตดังนี้

$$\hat{u}_k = \begin{cases} 1 & if \quad L(k) \geq 0 \\ 0 & if \quad L(k) \geq 0 \end{cases} \tag{3.7}$$

ถ้าตัวเข้ารหัสคอนโวลูชันมีจำนวนหน่วยความจำ v แผนภาพเทรลิสของมันจะมี q สถานะเมื่อ $q = 2^v$ โดยสถานะเหล่านี้จะกำหนดตำแหน่งตั้งแต่ 0 ถึง q -1 เมื่อ S_k เป็นสถานะที่เวลา k สำหรับตัวเข้ารหัสแบบคอนโวลูชันแบบมีระบบที่มีอัตราเข้ารหัสเป็น 1/2 การเปลี่ยน

สถานะแต่ละครั้งจะมีเอาต์พุต 2 ตัวเกี่ยวข้องโดยที่เอาต์พุตเหล่านี้จะเกี่ยวเนื่องกับบิตข้อมูลขาเข้าและค่า parity ของตัวเข้ารหัส โดยทั่วไป u_k และ x_k^p คือบิตที่เกี่ยวข้องเมื่อมีเปลี่ยนสถานะระหว่าง S_{k-1} ไป S_k

จากอัลกอริทึมของ BJCR เขียน log-likelihood ratio ได้ว่า

$$L(k) = \frac{\sum_{m=0}^{(q-1)} \sum_{m'=0}^{(q-1)} \gamma^1(y_k^s, y_k^p, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)}{\sum_{m=0}^{(q-1)} \sum_{m'=0}^{(q-1)} \gamma^0(y_k^s, y_k^p, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)} \tag{3.8}$$

โดยที่

$$\alpha_k(m) = \frac{\sum_{m'=0}^{(q-1)} \sum_{i=0}^{1} \gamma^i(y_k^s, y_k^p, m', m) \cdot \alpha_{k-1}(m')}{\sum_{m=0}^{(q-1)} \sum_{m'=0}^{(q-1)} \sum_{i=0}^{1} \gamma^i(y_k^s, y_k^p, m', m) \cdot \alpha_{k-1}(m')} \tag{3.9}$$

$$\beta_k(m) = \frac{\sum_{m'=0}^{(q-1)} \sum_{i=0}^{1} \gamma^i(y_k^s, y_k^p, m, m') \cdot \beta_{k+1}(m')}{\sum_{m=0}^{(q-1)} \sum_{m'=0}^{(q-1)} \sum_{i=0}^{1} \gamma^i(y_k^s, y_k^p, m', m) \cdot \alpha_k(m')} \tag{3.10}$$

$$\begin{aligned} \gamma^i(r_k, r_k^p, m', m) = {} & p(y_k \mid u_k = i, S_k = m, S_{k-1} = m') \cdot \\ & p(y_k^p \mid u_k = i, S_k = m, S_{k-1} = m') \cdot \\ & P(u_k = i \mid S_k = m, S_{k-1} = m') \cdot \\ & P(S_k = m \mid S_{k-1} = m') \end{aligned} \tag{3.11}$$

ค่าของ $\alpha_k(m)$ แสดงถึงความเหมือนของสถานะ ที่เวลา k เมื่อคิดจากการกวาดไปข้างหน้าบนแผนภาพเทรลิส โดยจะหาจากการคำนวณแบบซ้ำและค่าของมันที่เวลา k ใดๆคือฟังก์ชันเฉพาะสัญลักษณ์ที่ได้รับ (systematic และ partiy) ที่เวลาน้อยกว่าหรือเท่ากับ k เท่านั้น นอกจากนี้เนื่องจากอยู่ในรูปของการคำนวณแบบซ้ำ ค่าเริ่มต้น $\alpha_0(m)$ จึงขึ้นกับตัวแปรของการเข้ารหัส ถ้าให้ตัวเข้ารหัสเริ่มต้นที่สถานะศูนย์สำหรับแต่ละเฟรมดังนั้น $\alpha_0(0) = 1$ และ $\alpha_0(0) = 0$ เมื่อ $m \neq 0$

ค่า $\beta_k(m)$ นั้นจะแสดงถึงความเหมือนของสถานะโดยจะขึ้นกับการกวาดไปข้างหลังบนแผนภาพเทรลิส มันจะอยู่ในรูปแบบของการคำนวณซ้ำไปข้างหลังโดยเริ่มจากเวลา $k = N$ โดยจะกำหนดค่าเริ่มต้นของ $\beta_k(m)$ เมื่อรู้สถานะสุดท้ายของตัวเข้ารหัส $S_N = m_N$ ได้ $\beta_N(m_N) = 1$ และ $\beta_N(m) = 0$ เมื่อ $m \neq m_N$

การคำนวณขั้นสุดท้ายของอัลกอริทึม BCJR คือหาค่าความน่าจะเป็นในการเปลี่ยนสถานะ $\gamma^i = (y_k^s, y_k^p, m', m)$ โดยที่ m' คือสถานะของตัวเข้ารหัสที่เวลา $k-1$ และ m คือสถานะที่เวลา k สัญลักษณ์ y_k^s และ y_k^p จะถูกสร้างขึ้นระหว่างการเปลี่ยนสถานะจากสถานะที่ m' ไปสถานะที่ m ต่อไปจะทำ $\gamma^i = (y_k^s, y_k^p, m', m)$ ให้อยู่ในรูปง่าย

ค่า $p(y_k^s | u_k = i, S_k = m, S_{k-1} = m')$ จะถูกลดรูปเป็น $p(y_k^s | u_k)$ เนื่องจาก y_k^s ขึ้นกับค่า u_k และสัญญาณรบกวนในช่องสัญญาณเท่านั้น และ $P(u_k = i | S_k = m, S_{k-1} = m') \cdot P(S_k = m | S_{k-1} = m')$ คือ $P(u_k = i)$ เมื่อการเปลี่ยนจากสถานะที่ m' ไปสถานะที่ m นั้นเป็นได้และเกี่ยวข้องกับ $u_k = i$ ไม่เช่นนั้นค่านี้ก็เป็นศูนย์ จะเขียน $\gamma^i(y_k^s, y_k^p, m', m)$ ใหม่ได้ดังนี้

$$\gamma(y_k^s, y_k^p, m', m) = p(y_k^s | u_k = i) \cdot P(u_k = i) \cdot p(y_k^p | u_k = i, S_k = m, S_{k-1} = m') \cdot \delta(m' \xrightarrow{u_k = i} m) \quad (3.12)$$

$$= p(y_k^s | u_k = i) \cdot P(u_k = i) \cdot \gamma'_i(y_k^p, m', m) \quad (3.13)$$

โดยที่ $\delta(m' \xrightarrow{u_k = i} m)$ คือฟังก์ชันที่มีค่าเป็น 1 เมื่อ $u_k = i$ ทำให้เกิดการเปลี่ยนจากสถานะที่ m' ไปสถานะที่ m เมื่อแทนค่าและทำให้ $L(k)$ ง่ายขึ้นจะได้

$$L(k) = \log\left[\frac{p(y_k^s | u_k = 1)}{p(y_k^s | u_k = 0)}\right] + \log\left[\frac{(u_k = 1)}{(u_k = 0)}\right] + \log\left[\frac{\sum_{m=0}^{(q-1)} \sum_{m'=0}^{(q-1)} \gamma'_1(y_k^p, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)}{\sum_{m=0}^{(q-1)} \sum_{m'=0}^{(q-1)} \gamma'_0(y_k^p, m', m) \cdot \alpha_{k-1}(m') \cdot \beta_k(m)}\right] \quad (3.14)$$

สามารถเขียน Log-likelihood ratio ได้เป็นสามส่วนดังนี้

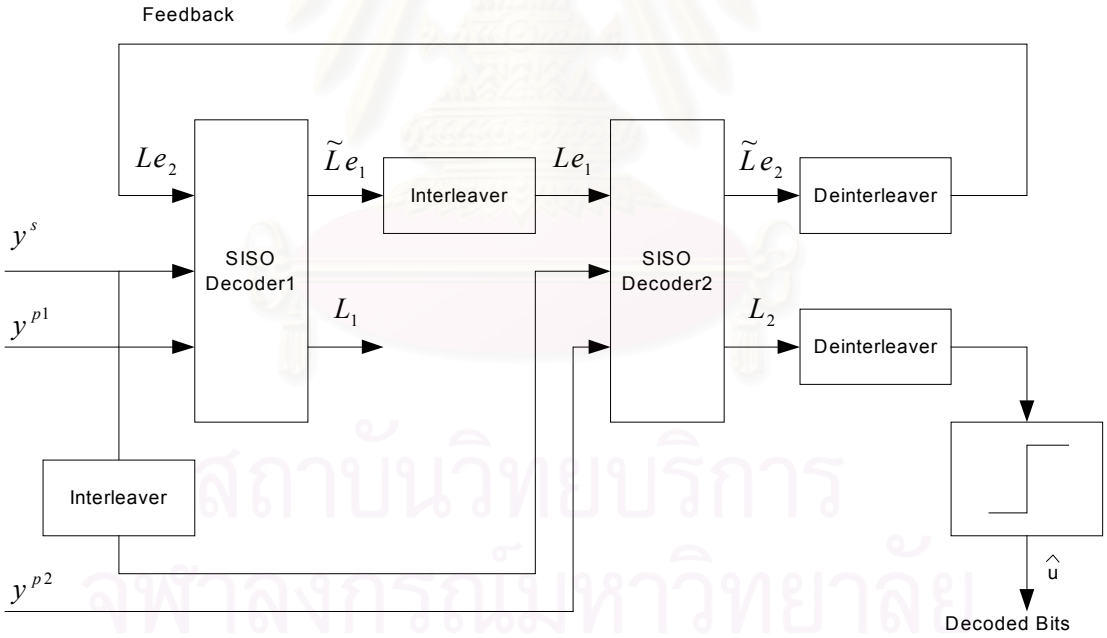
$$L(k) = L_{systematic} + L_{a\ priori} + L_{extrinsic} \quad (3.15)$$

$L_{systematic}$ นั้นขึ้นกับเฉพาะส่วนที่ได้รับมาจากสัญลักษณ์ที่เป็น systematic ที่เวลา k $L_{a\ priori}$ นั้นขึ้นกับ a priori information ของบิตข้อมูลเข้า u_k ส่วน $L_{extrinsic}$ คือความรู้ใหม่ที่

ได้จากบิตข้อมูลเข้า u_k โดยจะขึ้นกับ parity และ systematic information ทั้งหมดยกเว้น systematic value ที่เวลา k โดยการใช้สมการนี้ของ log-likelihood ratio ทำให้เราสามารถถอดรหัสเทอร์โบได้

3.2.2 ขั้นตอนการถอดรหัสเทอร์โบ

จากข้างต้นเราจะใช้ตัวถอดรหัสที่ใช้ MAP และได้เห็นแล้วว่า log-likelihood ratio นั้นสามารถแยกออกมาได้เป็น 3 ส่วนจากสมการที่ (3.15) จากรูป **SISO Decoder 1** และ **SISO Decoder 2** คือตัวถอดรหัสที่ใช้อัลกอริทึมของ BCJR สำหรับการถอดรหัสที่เข้ารหัสด้วยตัวเข้ารหัสเทอร์โบ 2 ตัวแยกกัน L_1 และ L_2 คือค่า log-likelihood ratio ที่คำนวณได้โดยใช้อัลกอริทึม BJCR ตามสมการที่ (3.15) อย่างไรก็ดีกุญแจสำคัญในการคำนวณซ้ำไปมาของเครื่องถอดรหัสเทอร์โบก็คือการส่งความน่าจะเป็นของบิตข้อมูลซึ่งอยู่ในรูปของ extrinsic information ที่ได้จาก BJCR



รูปที่ 3.5 ตัวถอดรหัสเทอร์โบ

extrinsic information นี้จะสร้างโดยตัวถอดรหัสอีกตัวเพื่อใช้ในการคำนวณหา a priori probability ที่แต่ละบิต ตัวอย่างเช่น **Decoder 1** จะใช้ extrinsic information จาก **Decoder 2** เพื่อใช้คำนวณ a priori probability $P(u_k = i)$ สำหรับแต่ละบิต

$$P(u_k = 1) = \left(\frac{e^{L_{e2}(k)}}{1 + e^{L_{e2}(k)}} \right) \tag{3.16}$$

$$P(u_k = 0) = 1 - \left(\frac{e^{L_{e2}(k)}}{1 + e^{L_{e2}(k)}} \right) \tag{3.17}$$

โดยค่า a priori information ใหม่นี้จะถูกใช้ในอัลกอริทึม BJCR สำหรับการกำหนดค่าเริ่มต้นนั้นจะให้ $L_{e_2}(k) = 0$ ทุกๆค่า k เมื่อเริ่มทำการคำนวณครั้งแรก จุดมุ่งหมายในการคำนวณแบบวนซ้ำไปมาก็คือเพื่อกำจัดความผิดพลาดของบิต

บทที่ 4
วิธีดำเนินการวิจัย

วิธีในการใส่ลายน้ำได้ทำการปรับปรุงการฝังลายน้ำใหม่โดยวิธีนี้จะเป็นแบบ Blind watermarking ซึ่งเป็นอีกชื่อเรียกหนึ่งของ Public watermarking คือไม่ต้องใช้ต้นฉบับในการตรวจสอบ โดยจะนำความรู้ของรหัสเทอร์โบ (turbo code), image restoration, spread spectrum communication มาใช้ด้วย หลักการของวิธีนี้มีอยู่ว่าเราจะฝังบิตข้อมูลซึ่งเป็นข้อมูลซึ่งแสดงถึงความเป็นเจ้าของภายในรูปสัญญาณที่มีลักษณะคล้ายสัญญาณรบกวนแล้วก็นำมาบวกเข้ากับรูปต้นฉบับที่ผ่านการทำแปลงโคซายน์และจัดเรียงสัมประสิทธิ์แบบซิกแซ็กก่อน โดยถ้ารักษาระดับให้อยู่ในระดับที่ต่ำก็จะไม่สามารถใช้ตาคนสังเกตเห็นได้ ส่วนการถอดรหัสเอาบิตข้อมูลที่ทำการฝังออกมาจะใช้ image restoration เพื่อทำการประมาณหารูปต้นฉบับจากรูปที่ทำการฝังลายน้ำแล้วเพราะจะไม่ใช้รูปต้นฉบับในการตรวจสอบ การประมาณนี้จะใช้เพื่อหาสัญญาณที่บวกเข้าไปกับรูปต้นฉบับแต่เนื่องจากสัญญาณรบกวนที่บวกเข้าไปนั้นมีพลังงานต่ำและการทำ image restoration นั้นไม่สมบูรณ์ดังนั้นบิตที่ถอดออกมาจึงมีความผิดพลาดเกิดขึ้นเพื่อที่แก้ไขความผิดพลาดนี้จึงได้อาศัยรหัสแก้ไขความผิดพลาดโดยบิตข้อมูลนั้นจะนำมาเข้ารหัสก่อนใส่ลายน้ำ โดยรหัสแก้ไขความผิดพลาดที่ใช้นั้นจะใช้รหัสเทอร์โบเนื่องจากมีประสิทธิภาพในการแก้ไขความผิดพลาดที่ดีที่ SNR ต่ำ โดยวิธีที่เสนอมีขั้นตอนดังนี้

4.1 การฝังลายน้ำ

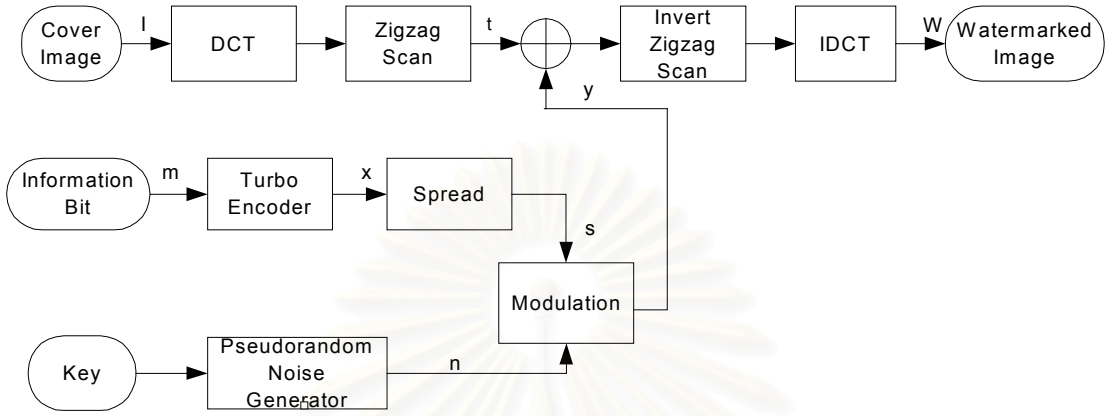
ขั้นตอนการฝังลายน้ำถูกแสดงในรูปที่ 4.1 ในระบบนี้บิตข้อมูลที่จะฝังเข้าไปในรูปภาพ m นั้นจะถูกนำมาเข้ารหัสเทอร์โบก่อนโดยมีอัตราการเข้ารหัส 1/3 ตามรูปที่ 3.4 เมื่อได้บิตข้อมูลที่เข้ารหัส x จึงนำมาทำการซ้ำบิตเดิมออกไปหลายๆครั้งโดยกำหนดค่า spreading factor ซึ่งบอกว่าจะทำซ้ำบิตข้อมูลที่เข้ารหัสกี่ครั้งเพื่อช่วยในการตัดสินใจได้ออกมาเป็น s ขณะที่ใช้กุญแจ (key) ทำการสร้างตัวแปรสุ่มแบบเทียมขึ้น n มีค่าเฉลี่ยเป็นศูนย์และความแปรปรวนเป็นหนึ่งซึ่งจะมีจำนวนเท่ากับ s นำเอา s และ n มามอดูเลตกันเสร็จแล้วจะได้สัญญาณที่จะนำไปบวกกับภาพ y ส่วนภาพต้นฉบับ I ก็จะทำการแปลงโคซายน์ (DCT) ทั้งรูปก่อนแล้วทำการจัดเรียงสัมประสิทธิ์แบบซิกแซ็ก (zigzag scan) เพื่อทำการเรียงความถี่จากความถี่ต่ำไปความถี่สูง ได้เป็น t จากนั้นจึงนำไปบวกกับสัญญาณที่สร้างโดยจะทำการบวกเข้าไปยังช่วงสัมประสิทธิ์ที่เหมาะสมเพื่อให้มีความทนทานและมองไม่เห็น โดย t' เป็นสัมประสิทธิ์หลังการบวก

แสดงการฝังลายน้ำดังนี้

$$t' = t + \alpha s n \tag{4.1}$$

เมื่อ α คือความแรงในการฝังลายน้ำ

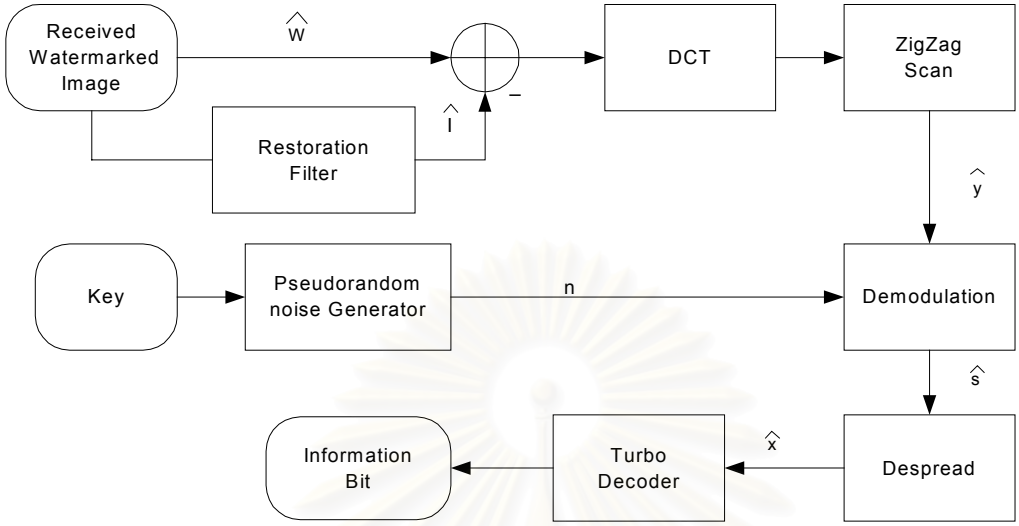
จากนั้นจะทำการซิกแซ็ก t' กลับและทำการแปลงกลับโคซายน์ (IDCT) ก็จะได้ภาพที่ผ่านการฝังลายน้ำ W



รูปที่ 4.1 ขั้นตอนการฝังลายน้ำ

4.2 การตรวจจับลายน้ำ

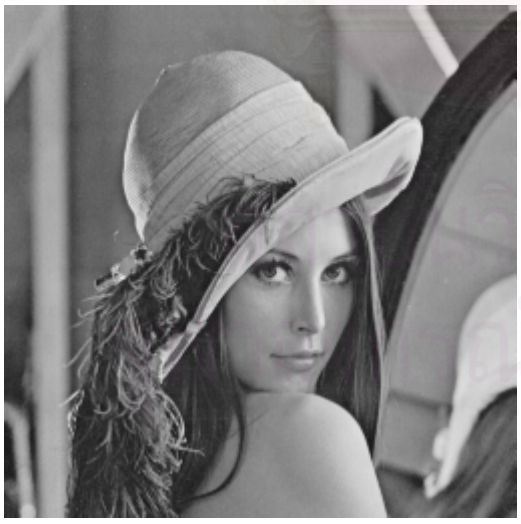
ขั้นตอนการตรวจจับลายน้ำแสดงดังรูปที่ 4.2 โดยภาพที่ได้รับ $\hat{W}$ เป็นภาพที่ฝังลายน้ำแล้วและอาจผ่านการทำ JPEG หรือกระบวนทดสอบความทนทานอื่นๆจะนำมาผ่าน image restoration filter ซึ่งจะใช้ wiener filter ขนาด 5x5 เพื่อที่จะให้ได้เป็นภาพประมาณของภาพต้นฉบับ $\hat{I}$ จากภาพที่ฝังลายน้ำจากนั้นนำผลต่างของทั้งสองนำเอาไปแปลงโคซายน์และเรียงสัมประสิทธิ์แบบซิกแซ็กก็จะได้ค่าประมาณของสัญญาณที่บวกเข้าไป $\hat{y}$ จากนั้นใช้กุญแจที่เหมือนกับที่ใช้ในตอนแรกทำการสร้างตัวแปรสุ่มแบบเทียม n แล้วมาดีมอดูเลตกับ $\hat{y}$ จะได้ $\hat{s}$ จากนั้นนำเอาสัญญาณที่ทำซ้ำกันหลายๆครั้งมารวมกันเป็นชุดเดียวได้ $\hat{x}$ แล้วนำไปถอดรหัสเทอร์โบเพื่อทำการถอดรหัสบิตข้อมูลที่ได้ m



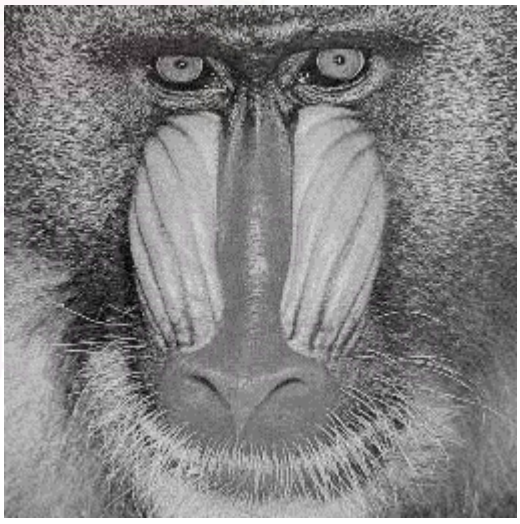
รูปที่ 4.2 ขั้นตอนการตรวจสอบลายน้ำ

4.3 ภาพที่ใช้ทดสอบ

ภาพที่นำมาทดสอบในการใส่ลายน้ำจะใช้ภาพ Gray Scale 8 บิต ของ Lena, Baboon, Boat และ Cameraman ขนาด 256 x 256 จุดภาพในการทดสอบ ดังแสดงในรูปที่ 4.3-4.6



รูปที่ 4.3 ภาพ Lena



รูปที่ 4.4 ภาพ Baboon

รูปที่ 4.5 ภาพ Cameraman

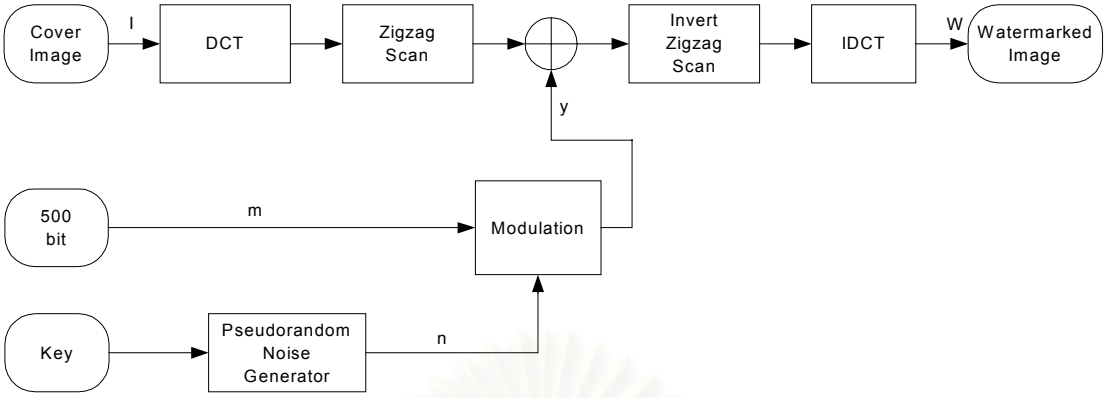
รูปที่ 4.6 ภาพ Boat

4.4 การหาช่วงสัมประสิทธิ์ในการใส่ลายน้ำ

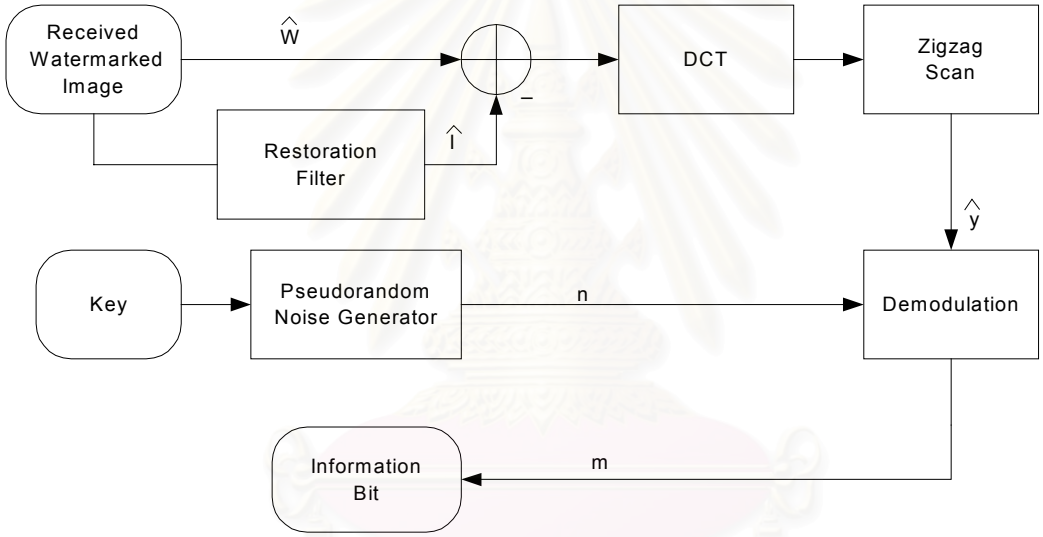
เนื่องจากว่าการที่ JPEG เป็นการทดสอบความทนทานที่สำคัญที่สุดเนื่องจากเป็นรูปแบบการบีบอัดภาพที่แพร่หลายมากในอินเตอร์เนต รูปภาพที่ถูกละเมิดลิขสิทธิ์อาจถูกนำไปผ่านการทำ JPEG แล้วแจกจ่ายไปทั่วโครงข่ายดังนั้นวิธีในการฝังลายน้ำจึงต้องให้มีความทนทานต่อ JPEG โดยจะทำการทดสอบ JPEG ที่ Quality Factor 80 เนื่องจากที่ระดับนี้รูปยังอยู่ในระดับที่คุณภาพไม่ต่างไปจากรูปก่อนทำ JPEG มากนัก โดยเราจะทำการทดสอบการหาช่วงสัมประสิทธิ์ในการใส่ลายน้ำ

ช่วงสัมประสิทธิ์ที่พูดถึงนี้หมายถึงช่วงสัมประสิทธิ์ของภาพที่ผ่านการแปลงโคซายน์และจัดเรียงสัมประสิทธิ์แบบซิกแซ็กซึ่งในภาพขนาด 256 x 256 จะมีจำนวน 65536 ตำแหน่ง โดยมีวิธีการหาตามรูปที่ 4.7 และ 4.8 ในการทดสอบหาช่วงสัมประสิทธิ์ในการใส่ลายน้ำนี้จะยังไม่ทำการเข้ารหัสเทอร์โบและการทำซ้ำรหัสเทอร์โบโดยจะเป็นการฝังบิตข้อมูลลงไปแบบหนึ่งบิตต่อหนึ่งตำแหน่ง การใส่ลายน้ำจะทำครั้งละ 1 บล็อกกำหนดให้ 1 บล็อกมี 500 ตำแหน่ง

โดยที่บิตข้อมูลในการฝังและตัวแปรสุ่มแบบเทียมที่สร้างขึ้นจะกำหนดให้เท่ากันแต่จะเปลี่ยนความแรงในการฝังลายน้ำจาก 1 จนถึง 35 และทำการตรวจสอบลายน้ำโดยจะทำการเปรียบเทียบบิตข้อมูลที่ถอดออกมาได้กับบิตข้อมูลที่ฝังเข้าไปโดยหาค่าสหสัมพันธ์ทำไปทีละบล็อกจนครบทุกตำแหน่ง ในการทดสอบนี้แต่และครั้งที่ทำจะมีการนำภาพที่ผ่านการฝังลายน้ำแล้วมาเปรียบเทียบกับภาพต้นฉบับเพื่อหาจุดเสียที่เกิดจากการฝังลายน้ำโดยใช้วิธีของ Girod [20] แล้วเก็บข้อมูลไว้

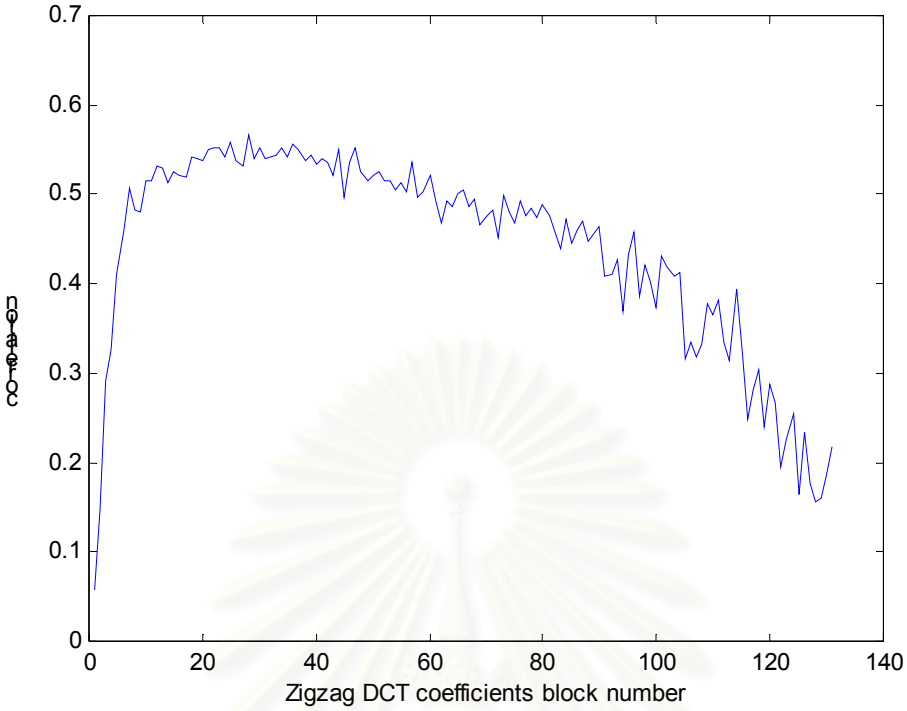


รูปที่ 4.7 การฝังลายน้ำเพื่อทดสอบหาช่วงสัมประสิทธิ์ที่เหมาะสมในการฝังลายน้ำ

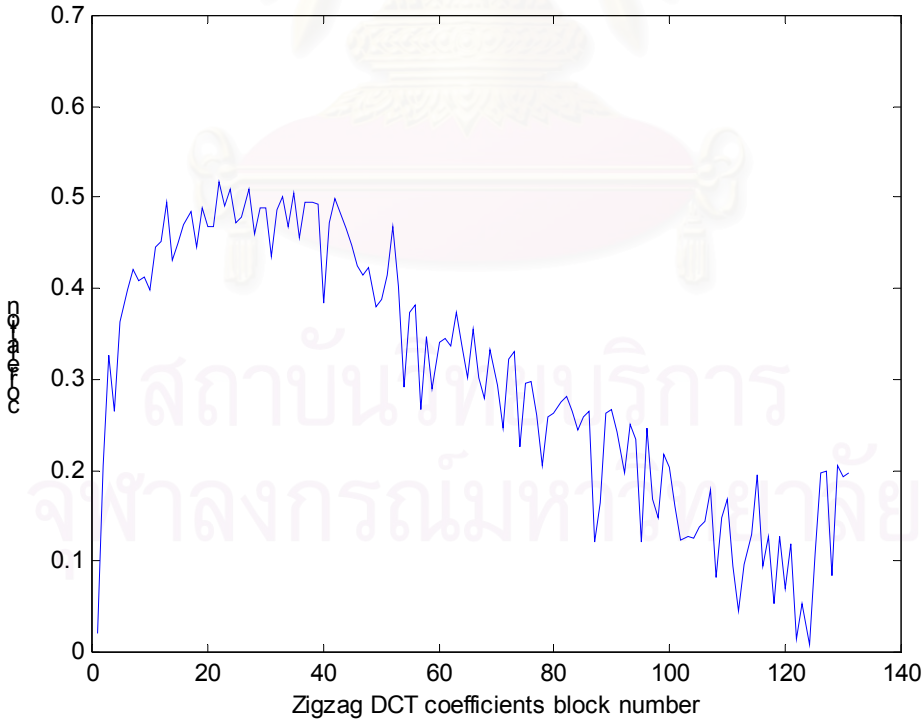


รูปที่ 4.8 การตรวจจับลายน้ำเพื่อทดสอบหาช่วงสัมประสิทธิ์ที่เหมาะสมในการฝังลายน้ำ

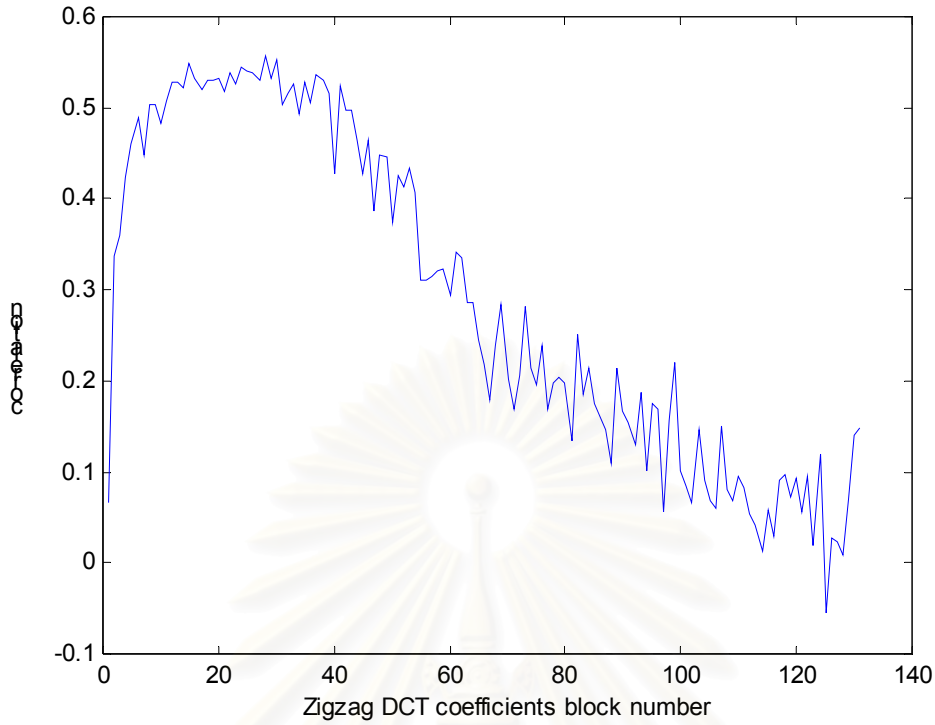
เมื่อทำจนครบแล้วเราก็จะสามารถนำข้อมูลที่ได้มาเขียนกราฟของจำนวนของจุดเสียในแต่และบล็อกเมื่อกำหนดให้ความแรงในการฝังคงที่และกราฟของค่าสหสัมพันธ์ในแต่ละบล็อกเมื่อกำหนดให้ความแรงในการฝังคงที่ จากนั้นจึงใช้การประมาณ (Interpolation) ทำการเขียนกราฟของค่าสหสัมพันธ์ในแต่ละบล็อกโดยกำหนดให้จำนวนจุดเสียคงที่โดยที่ค่าความแรงในการฝังนั้นเปลี่ยนแปลงได้ โดยจะกำหนดให้มีจุดเสีย 200 จุดจะได้กราฟของแต่ละรูปดังนี้



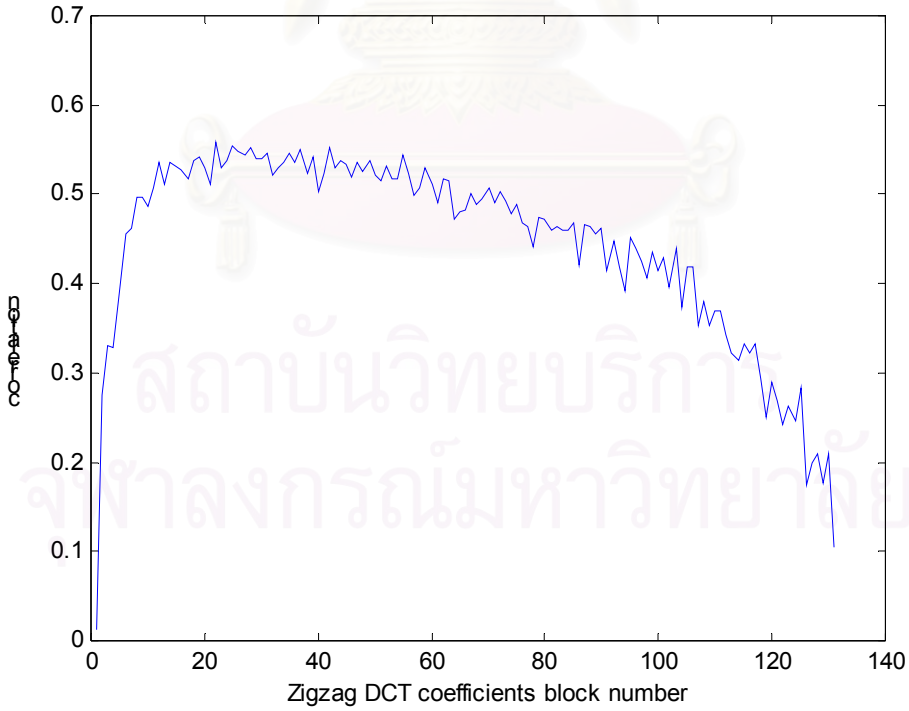
รูปที่ 4.9 กราฟของค่าสหสัมพันธ์ในแต่ละบล็อกของภาพ Lena โดยกำหนดให้มีจุดเสีย 200 จุด



รูปที่ 4.10 กราฟของค่าสหสัมพันธ์ในแต่ละบล็อกของภาพ Baboon โดยกำหนดให้มีจุดเสีย 200 จุด



รูปที่ 4.11 กราฟของค่าสหสัมพันธ์ในแต่ละบล็อกของภาพ Cameraman โดยกำหนดให้มีจุดเสีย 200 จุด



รูปที่ 4.12 กราฟของค่าสหสัมพันธ์ในแต่ละบล็อกของภาพ Boat โดยกำหนดให้มีจุดเสีย 200 จุด

พิจารณากราฟของค่าสหสัมพันธ์ (correlation) ในแต่ละบล็อกที่กำหนดให้จำนวนจุดที่เสีย 200 จุด ค่าสหสัมพันธ์จะแสดงถึงบิตที่ได้เมื่อผ่านการตรวจจับลายน้ำว่ามีความเหมือนกับบิตที่ฝังแค่ไหน การพิจารณาหาช่วงสัมประสิทธิ์ที่จะใส่ลายน้ำลงไปจะเลือกใช้ช่วงที่มีค่าสหสัมพันธ์ประมาณ 0.5 ขึ้นไป จากกราฟจะพิจารณาให้

ภาพ Lena จะใส่ที่ตำแหน่ง 5001–30000

ภาพ Baboon จะใส่ที่ตำแหน่ง 10001-20000

ภาพ Cameraman จะใส่ที่ตำแหน่ง 5001-20000

ภาพ Boat จะใส่ที่ตำแหน่ง 5001-32500

เมื่อได้ช่วงสัมประสิทธิ์ในการฝังลายน้ำแล้วต่อไปจะหาค่าความแรงในการฝังลายน้ำของรูปที่ใช้ทดสอบแต่ละรูปโดยกำหนดให้มีค่าความแรงในการฝังคงที่และรูปมีจุดเสียไม่เกิน 200 จุด ทำการทดสอบโดยใส่ลายน้ำในช่วงที่กำหนดโดยยังไม่มีการเข้ารหัสเทอร์โบและเป็นการฝังบิตข้อมูลแบบหนึ่งบิตต่อหนึ่งตำแหน่งเหมือนเดิม ใช้โปรแกรมทดสอบของ Girod ดูค่าความแรงในการฝังลายน้ำที่เหมาะสมที่จะให้จุดเสียไม่เกิน 200 จุดจะได้ค่าเริ่มต้นในทดสอบความทนทานของการฝังลายน้ำของแต่ละรูปดังนี้

ตารางที่ 4.1 ช่วงสัมประสิทธิ์และความแรงในการฝังลายน้ำของภาพแต่ละภาพ

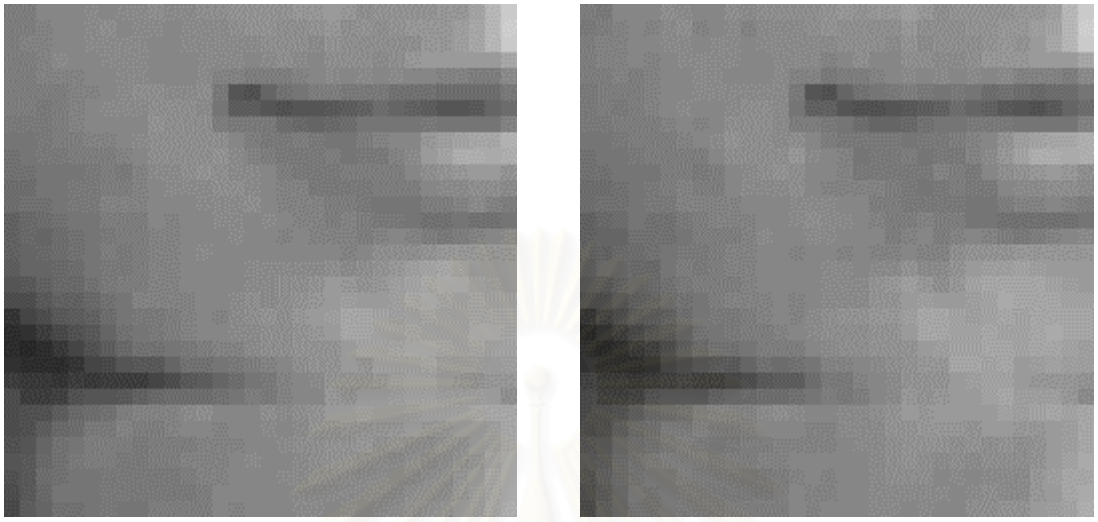
ภาพ	ช่วงสัมประสิทธิ์ที่ใส่ลายน้ำ	ความแรงในการฝัง α
Lena	5001-30000	2.3
Baboon	10001-20000	3.6
Cameraman	5001-20000	3.0
Boat	5001-32500	2.37

4.5 ตัวอย่างการฝังลายน้ำลงในรูปภาพ

รูปด้านล่างจะแสดงภาพ Lena ต้นฉบับและภาพ Lena ที่ผ่านการฝังลายน้ำแล้วโดยจะฝังลายน้ำไป 1000 บิต จากนั้นจะแสดงจุดเสียของภาพ Lena ที่ทำการฝังลายน้ำโดยใช้โปรแกรมของ Girod และทำการขยายในส่วนที่เป็นกรอบสี่เหลี่ยมของภาพทั้งสองเพื่อเปรียบเทียบกันโดยเลือกกรอบสี่เหลี่ยมที่มีจุดเสียเกิดขึ้น

รูปที่ 4.13 การเปรียบเทียบภาพ Lena ต้นฉบับ (ซ้าย) กับภาพที่ฝังลายน้ำ 1000 บิต (ขวา)

รูปที่ 4.14 ภาพแสดงจุดที่เสียของภาพที่ฝังลายน้ำ



รูปที่ 4.15 การเปรียบเทียบภาพขยายในกรอบสี่เหลี่ยมที่กำหนดของ Lena ต้นฉบับ (ซ้าย) กับภาพที่ฝังลายน้ำ 1000 บิต (ขวา)

บทที่ 5
ผลการทดสอบ

เมื่อได้ทำการฝังลายน้ำที่สมบูรณ์แล้วต่อไปจะทำการทดสอบความทนทานโดยจะทำการทดสอบโดยวิธีต่างๆโดยจะทำการหาจำนวนบิตที่ฝังได้มากที่สุดโดยจะยอมให้มี BER เมื่อทำการตรวจจับไม่เกิน 1×10^{-3} โดยบิตที่ฝังนั้นจะเป็นบิตที่มากที่สุดที่มีได้ในช่วงสัมประสิทธิ์ที่กำหนดเมื่อให้จำนวนครั้งในการซ้ำบิตเดิม (spreading factor) ค่าหนึ่งตามวิธีในบทที่ 4 โดยมีวิธีการหาจำนวนบิตข้อมูลที่มากที่สุด m ที่ฝังได้ในภาพเมื่อกำหนดค่า spreading factor ให้ดังนี้ จากการที่บิตข้อมูลที่หาได้จะถูกนำไปเข้ารหัสเทอร์โบที่มีอัตราการเข้ารหัสเป็น 1/3 และรหัสเทอร์โบมีการเติมบิตส่วนหางออกไป 18 บิตเพื่อทำให้สถานะสุดท้ายเป็นศูนย์ ดังนั้นจะหา m ได้จากสมการที่ (5.1)

จำนวนสัมประสิทธิ์ที่ใส่ได้ในภาพ = (3m +18) x spreading factor (5.1)

ตัวอย่าง

จากสมการที่ (5.1) หาจำนวนบิตข้อมูลมากที่สุดที่ฝังได้ในภาพ Lena เมื่อกำหนดให้ spreading factor เท่ากับ 14 ได้ m = ((25000/14) –18)/3 = 589 บิต เมื่อให้ spreading factor เท่ากับ 13 และ 12 จะได้ m มีค่า 635 บิต และ 688 บิต ตามลำดับ

ส่วนเมื่อไม่มีการเข้ารหัสเทอร์โบและใช้ Hamming code (7,4) แทนจะหา m ได้จากสมการที่ (5.2) และ (5.3) ตามลำดับ

จำนวนสัมประสิทธิ์ที่ใส่ได้ในภาพ = m x spreading factor (5.2)

จำนวนสัมประสิทธิ์ที่ใส่ได้ในภาพ = m x (7/4) x spreading factor (5.3)

เมื่อได้จำนวนบิตที่มากที่สุดเมื่อกำหนด spreading factor ค่าหนึ่งแล้วจะทำการทดสอบความทนทานแล้วหาค่า BER เมื่อทำการตรวจจับ จากนั้นเปลี่ยนค่า spreading factor แล้วทำเช่นเดิม นำผลที่ได้ทั้งหมดมาพิจารณาหาจำนวนบิตที่ฝังได้มากที่สุดโดยมี BER ไม่เกิน 1×10^{-3} ดังนั้นจึงทำให้การกำหนดจำนวนบิตที่ฝังได้ดังแสดงในตารางที่ 5.1 มีค่าดิสครีตเนื่องจากเปลี่ยนไปตาม spreading factor ที่กำหนด และทำให้ BER ในตารางมีค่าดิสครีตด้วย

5.1 การทดสอบความทนทานโดยการทำ JPEG ที่ Quality Factor 80

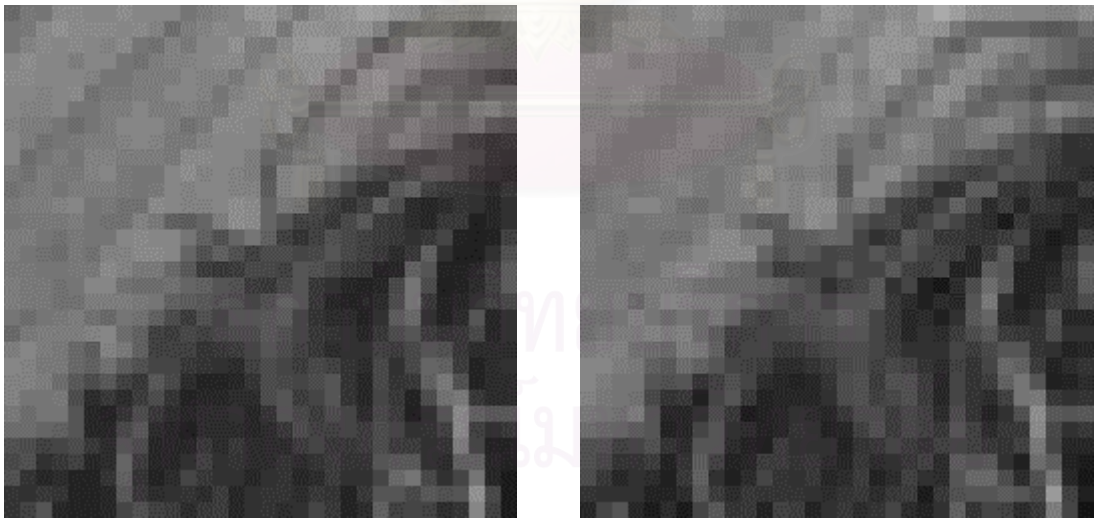
เมื่อทำการหาตำแหน่งในการฝังลายน้ำในแต่ภาพได้แล้วต่อไปจะทำการหาจำนวนบิตที่มากที่สุดที่จะทำการฝังในภาพได้โดยภาพที่ฝังลายน้ำแล้วจะนำไปบีบอัดด้วย JPEG ที่ Quality

Factor 80 ก่อนที่จะนำภาพไปตรวจสอบลายน้ำโดยให้มี BER ได้ไม่เกิน 1×10^{-3} JPEG เป็นวิธีการบีบอัดภาพแบบ Lossy compression คือว่าการบีบอัดนั้นจะทำให้ข้อมูลสูญหายไป โดย JPEG จะทำการกำจัดสัญญาณที่ไม่มีความสำคัญต่อการมองเห็นออก ถ้าลายน้ำสามารถทนทานต่อการบีบอัดนี้ได้ แสดงว่าลายน้ำนั้นฝังอยู่ในบริเวณที่มีความสำคัญของสเปกตรัม

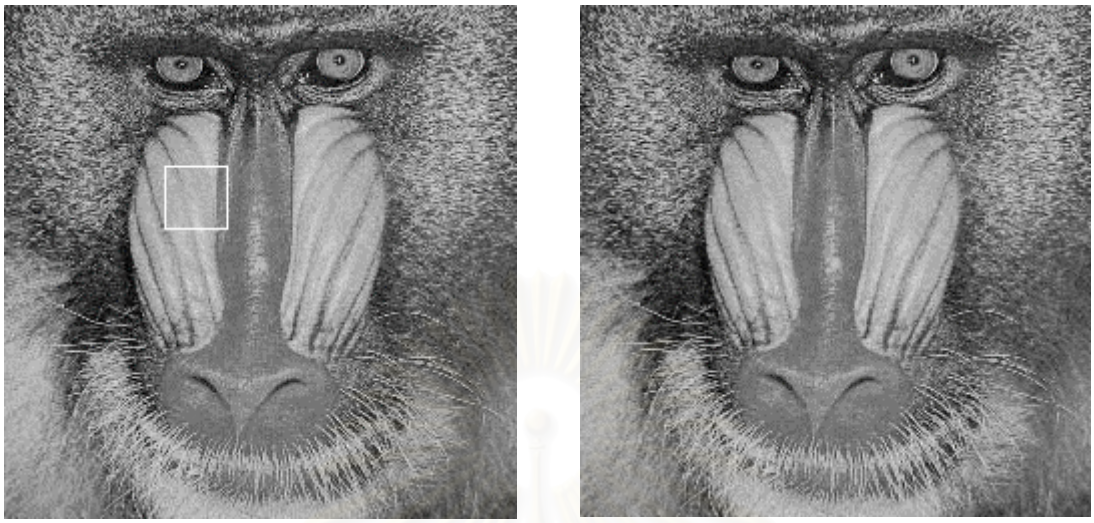
ตารางที่ 5.1 จำนวนบิตที่ฝังในภาพและ BER เมื่อทำการบีบอัดด้วย JPEG ที่ Quality Factor 80 ก่อนตรวจจับลายน้ำ

ภาพ	บิตที่ฝังได้	spreading factor	BER
Lena	589	14	4.46×10^{-5}
	635	**13**	**7.35×10^{-4}**
	688	12	4.87×10^{-3}
Baboon	60	50	4.84×10^{-4}
	62	**49**	**8.53×10^{-4}**
	63	48	1.17×10^{-3}
Cameraman	448	11	1.98×10^{-5}
	494	**10**	**7.19×10^{-5}**
	549	9	2.19×10^{-3}
Boat	452	20	2.35×10^{-4}
	476	**19**	**3.34×10^{-4}**
	503	18	2.46×10^{-3}

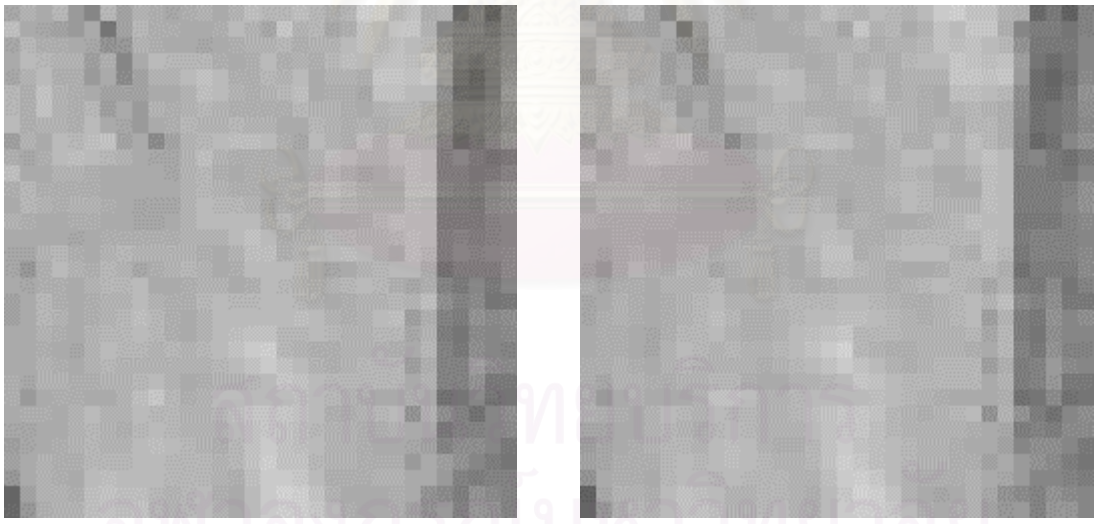
รูปที่ 5.1 การเปรียบเทียบภาพ Lena ที่ฝังลายน้ำ 635 บิต (ซ้าย) และภาพที่ฝังลายน้ำและทำการบีบอัดแบบ JPEG ที่ Quality Factor 80 (ขวา)



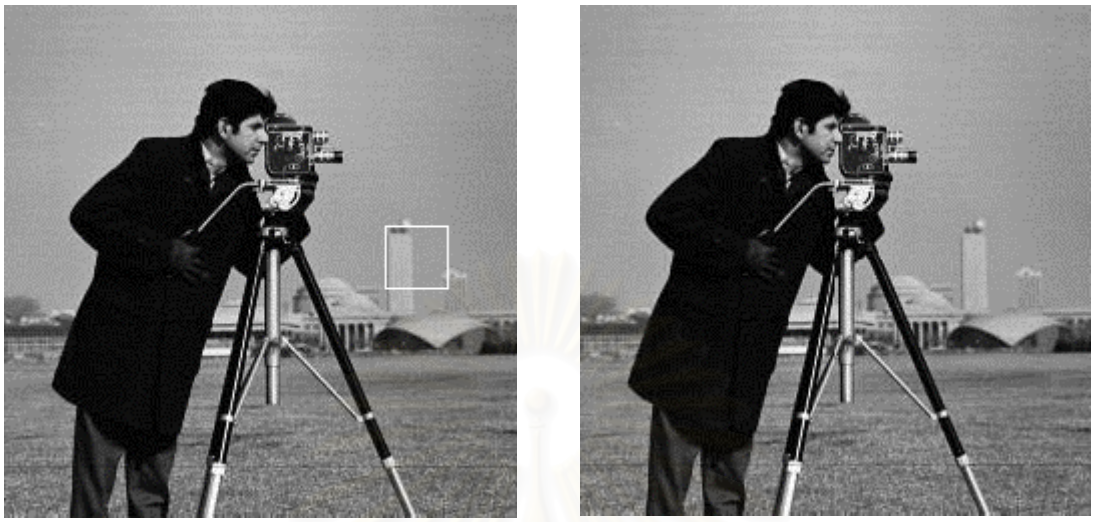
รูปที่ 5.2 การเปรียบเทียบภาพ Lena ที่ขยายในส่วนล้อมกรอบที่ฝังลายน้ำ 635 บิต (ซ้าย) และภาพที่ฝังลายน้ำและทำการบีบอัดแบบ JPEG ที่ Quality Factor 80 (ขวา)



รูปที่ 5.3 การเปรียบเทียบภาพ Baboon ที่ฝังลายน้ำ 62 บิต (ซ้าย) และภาพที่ฝังลายน้ำและทำการบีบอัดแบบ JPEG ที่ Quality Factor 80 (ขวา)



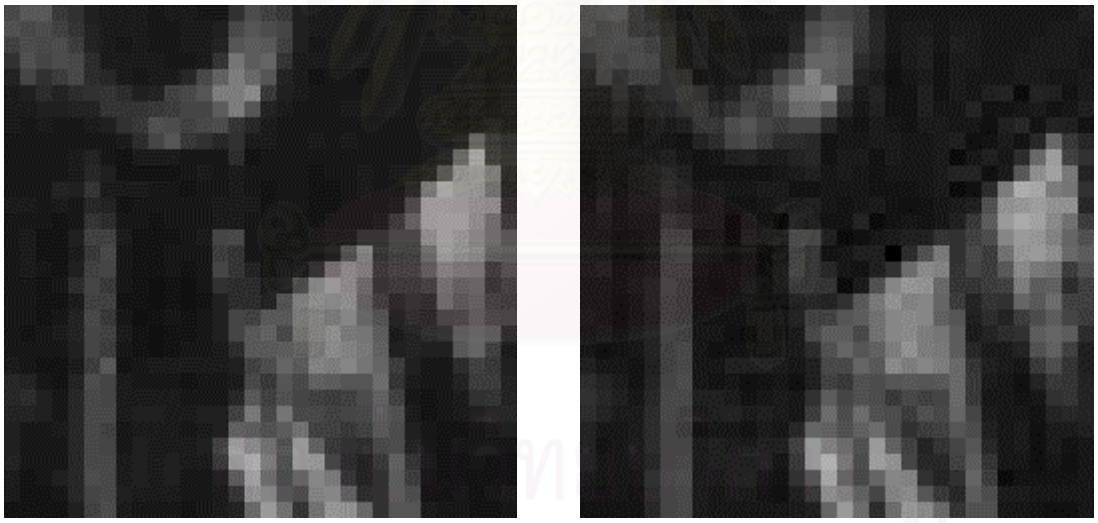
รูปที่ 5.4 การเปรียบเทียบภาพ Baboon ที่ขยายในส่วนล้อมกรอบที่ฝังลายน้ำ 62 บิต (ซ้าย) และภาพที่ฝังลายน้ำและทำการบีบอัดแบบ JPEG ที่ Quality Factor 80 (ขวา)



รูปที่ 5.5 การเปรียบเทียบภาพ Cameraman ที่ฝังลายน้ำ 494 บิต (ซ้าย) และภาพที่ฝังลายน้ำและทำการบีบอัดแบบ JPEG ที่ Quality Factor 80 (ขวา)

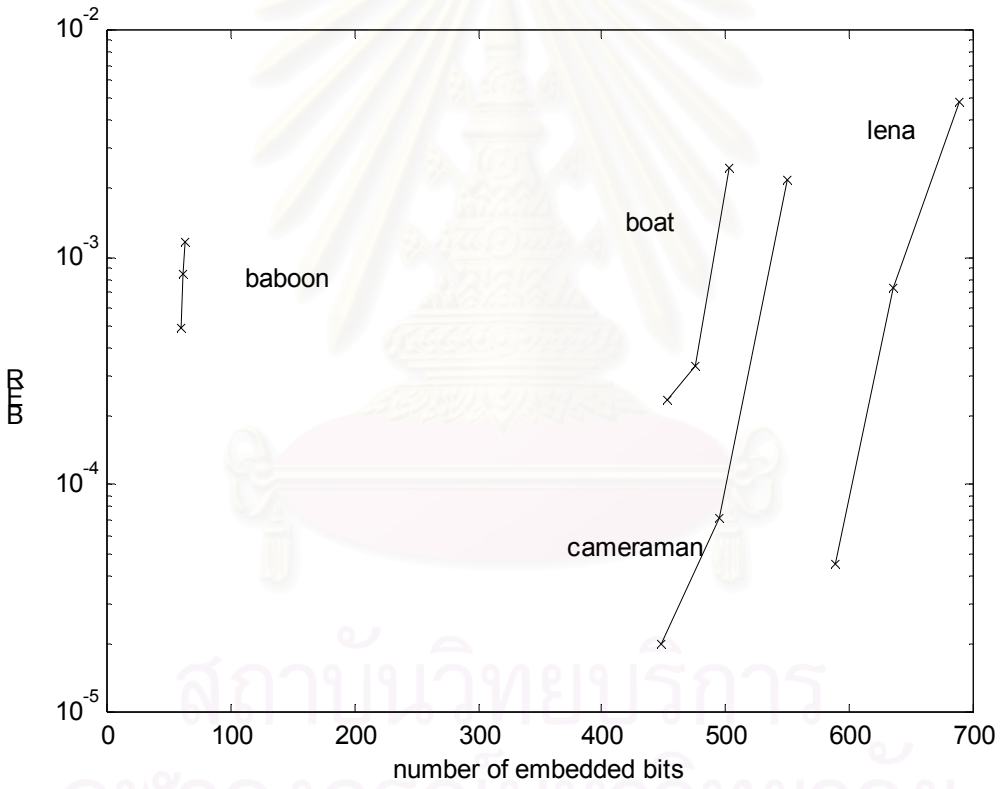
รูปที่ 5.6 การเปรียบเทียบภาพ Cameraman ที่ขยายในส่วนล้อมกรอบที่ฝังลายน้ำ 494 บิต (ซ้าย) และภาพที่ฝังลายน้ำและทำการบีบอัดแบบ JPEG ที่ Quality Factor 80 (ขวา)

รูปที่ 5.7 การเปรียบเทียบภาพ Boat ที่ฝังลายน้ำ 476 บิต (ซ้าย) และภาพที่ฝังลายน้ำและทำการบีบอัดแบบ JPEG ที่ Quality Factor 80 (ขวา)



รูปที่ 5.8 การเปรียบเทียบภาพ Boat ที่ขยายในส่วนล้อมกรอบที่ฝังลายน้ำ 476 บิต (ซ้าย) และภาพที่ฝังลายน้ำและทำการบีบอัดแบบ JPEG ที่ Quality Factor 80 (ขวา)

จากตารางสรุปและกราฟด้านล่างสรุปได้ว่าเมื่อทำการบีบอัดภาพด้วย JPEG ที่ Quality Factor 80 ภาพ Lena สามารถฝังบิตได้มากที่สุด 635 บิต, ภาพ Baboon ฝังบิตได้มากที่สุด 62 บิต, ภาพ Cameraman สามารถฝังบิตได้มากที่สุด 494 บิต, ภาพ Boat ฝังบิตได้มากที่สุด 476 บิต เนื่องจากเมื่อลดค่า spreading factor ซึ่งทำให้จำนวนบิตที่ฝังได้ในภาพมีค่ามากขึ้นแต่ค่า BER จะเกิน 1×10^{-3} แล้ว การที่ภาพ Baboon ฝังบิตได้น้อยมากเมื่อเทียบกับภาพอื่น เนื่องจากว่าภาพ Baboon มีส่วนประกอบสำคัญของภาพส่วนใหญ่อยู่ในความถี่สูงและ Wiener Filter ที่สร้างขึ้นอาจทำการลบรายละเอียดสำคัญที่อยู่ในความถี่สูงของภาพทำให้ภาพที่กู้คืนมาไม่สมบูรณ์

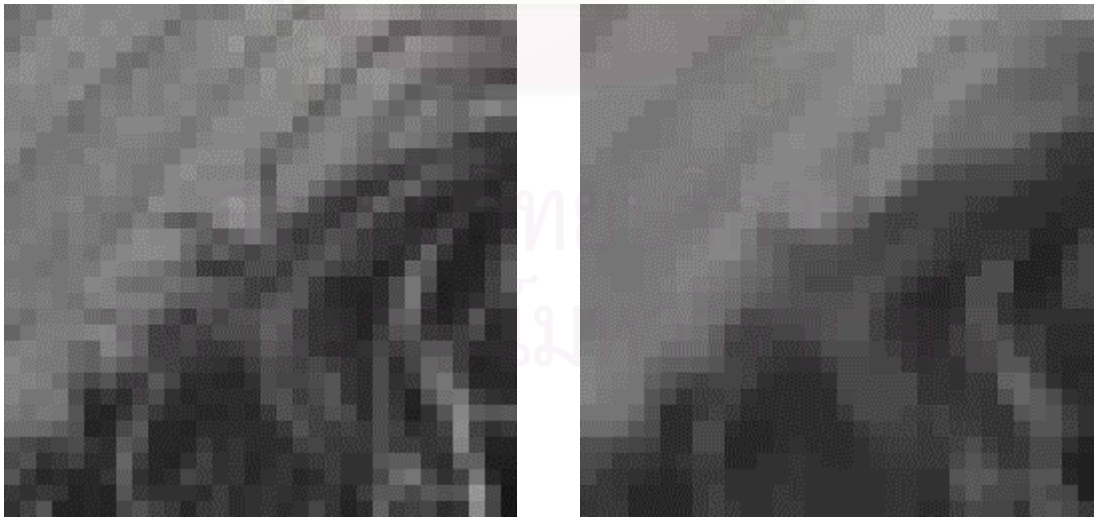


รูปที่ 5.9 กราฟแสดงความสัมพันธ์ของบิตที่ฝังกับ BER ที่ได้ของภาพ Lena, Baboon, Cameraman, Boat เมื่อทำการทดสอบความทนทานโดยการบีบอัดด้วย JPEG ที่ Quality Factor 80

5.2 การทดสอบความทนทานโดย Median Filtering

เมื่อทำการหาตำแหน่งในการฝังลายน้ำในแต่ละภาพได้แล้วต่อไปจะทำการหาจำนวนบิตที่มากที่สุดที่จะทำการฝังในภาพได้ โดยภาพที่ฝังลายน้ำแล้วจะนำไปผ่าน Median Filter ที่มีขนาดหน้าต่าง 3 x 3 ก่อนที่จะนำภาพไปตรวจสอบลายน้ำโดยให้มี BER ได้ไม่เกิน 1×10^{-3}

รูปที่ 5.10 การเปรียบเทียบภาพ Lena ที่ฝังลายน้ำ 232 บิต (ซ้าย) กับภาพที่ฝังลายน้ำบิตแล้วทำ Median Filtering ขนาด 3 x 3 (ขวา)



รูปที่ 5.11 การเปรียบเทียบภาพ Lena ที่ขยายในส่วนล้อมกรอบที่ฝังลายน้ำ 232 บิต (ซ้าย) และภาพที่ฝังลายน้ำแล้วทำ Median Filtering ขนาด 3 x 3 (ขวา)

ตารางที่ 5.2 จำนวนบิตที่ฝังในภาพ Lena และ BER เมื่อทำ Median Filtering ขนาด 3 x 3 ก่อนตรวจจับลายน้ำ

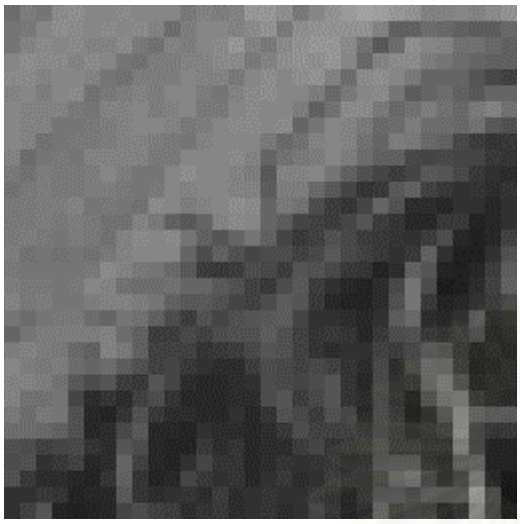
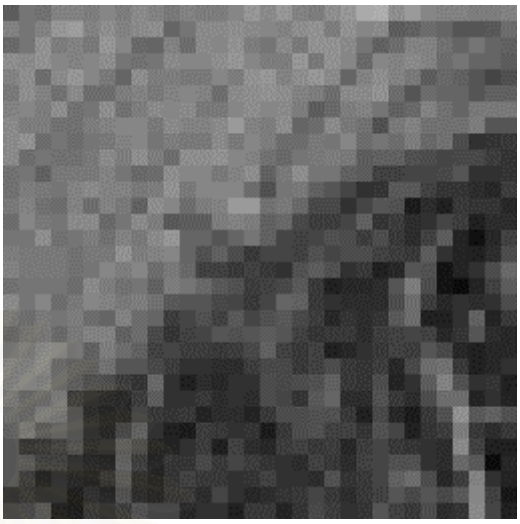
ภาพ	บิตที่ฝัง	spreading factor	BER
Lena	225	36	5.42×10^{-4}
	232	**35**	**7.59×10^{-4}**
	239	34	1.31×10^{-3}

จากตารางสรุปได้ว่าเมื่อทำ Median Filtering ขนาด 3 x 3 ภาพ Lena สามารถฝังบิตได้มากที่สุด 232 บิต เนื่องจากเมื่อลดค่า spreading factor เป็น 34 ทำให้จำนวนบิตที่ฝังได้ในภาพมีค่ามากขึ้นแต่ค่า BER จะเกิน 1×10^{-3}

5.3 การทดสอบความทนทานโดยใส่สัญญาณรบกวนแบบเกาส์เข้าไป

เมื่อทำการหาตำแหน่งในการฝังลายน้ำในแต่ภาพได้แล้วต่อไปจะทำการหาจำนวนบิตที่มากที่สุดที่จะทำการฝังในภาพได้ โดยภาพที่ฝังลายน้ำแล้วจะนำไปใส่สัญญาณรบกวนแบบเกาส์โดยมีค่าเฉลี่ยเป็นศูนย์และมีความแปรปรวน 0.001 เท่ากับมีค่า SNR 24.2 dB ก่อนที่จะนำภาพไปตรวจสอบลายน้ำโดยให้มี BER ได้ไม่เกิน 1×10^{-3}

รูปที่ 5.12 การเปรียบเทียบภาพ Lena ที่ฝังลายน้ำ 372 บิต (ซ้าย) กับภาพที่ฝังลายน้ำแล้วใส่สัญญาณรบกวนแบบเกาส์ (ขวา)

รูปที่ 5.13 การเปรียบเทียบภาพ Lena ที่ขยายในส่วนล้อมกรอบที่ฝังลายน้ำ 232 บิต (ซ้าย) และภาพที่ฝังลายน้ำแล้วใส่สัญญาณรบกวนแบบเกาส์ (ขวา)

ตารางที่ 5.3 จำนวนบิตที่ฝังในภาพ Lena และ BER เมื่อทำการฝังสัญญาณรบกวนแบบเกาส์ก่อนตรวจจับลายน้ำ

ภาพ	บิตที่ฝังได้	spreading factor	BER
Lena	356	23	1.92×10^{-4}
	372	**22**	**2.81×10^{-4}**
	390	21	2.83×10^{-3}

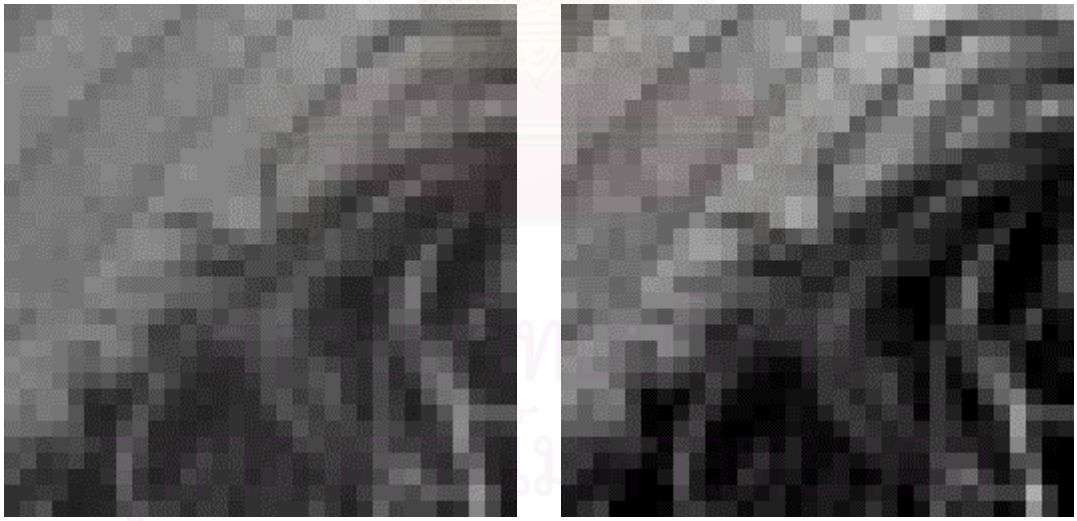
จากตารางสรุปได้ว่าเมื่อทำการฝังสัญญาณรบกวนแบบเกาส์ ภาพ Lena สามารถฝังบิตได้มากที่สุด 372 บิต เนื่องจากเมื่อลดค่า spreading factor เป็น 21 ทำให้จำนวนบิตที่ฝังได้ในภาพมีค่ามากขึ้นแต่ค่า BER จะเกิน 1×10^{-3}

5.4 การทดสอบความทนทานโดยทำ Histogram equalization

เมื่อทำการหาตำแหน่งในการฝังลายน้ำในแต่ภาพได้แล้วต่อไปจะทำการหาจำนวนบิตที่มากที่สุดที่จะทำการฝังในภาพได้ โดยภาพที่ฝังลายน้ำแล้วจะนำไปทำ Histogram equalization

ซึ่งก็คือการกระจาย histogram ของภาพออกให้สม่ำเสมอกันมากขึ้นก่อนที่จะนำภาพไปตรวจสอบลายน้ำโดยให้มี BER ได้ไม่เกิน 1×10^{-3}

รูปที่ 5.14 การเปรียบเทียบภาพ Lena ที่ฝังลายน้ำ 2077 บิต (ซ้าย) กับภาพที่ฝังลายน้ำแล้วทำ Histogram equalization (ขวา)



รูปที่ 5.15 การเปรียบเทียบภาพ Lena ที่ขยายในส่วนล้อมกรอบที่ฝังลายน้ำ 2077 บิต (ซ้าย) และภาพที่ฝังลายน้ำแล้วทำ Histogram equalization (ขวา)

ตารางที่ 5.4 จำนวนบิตที่ฝังในภาพ Lena และ BER เมื่อทำ Histogram equalization ก่อนตรวจจับลายน้ำ

ภาพ	บิตที่ฝัง	spreading factor	BER
Lena	**2077**	**4**	**0**
	2500	3	2.59×10^{-1}

จากตารางสรุปได้ว่าเมื่อทำ Histogram equalization ภาพ Lena สามารถฝังบิตได้มากที่สุด 2077 บิต เนื่องจากเมื่อลดค่า spreading factor เป็น 3 ทำให้จำนวนบิตที่ฝังได้ในภาพมีค่ามากขึ้นแต่ค่า BER จะเกิน 1×10^{-3}

5.5 การทดสอบความทนทานโดยทำการตัดรูปออก

เมื่อทำการหาตำแหน่งในการฝังลายน้ำในแต่ภาพได้แล้วต่อไปจะทำการหาจำนวนบิตที่มากที่สุดที่จะทำการฝังในภาพได้ โดยภาพที่ฝังลายน้ำแล้วจะนำไปทำการตัดบางส่วนของภาพออก (Crop) ซึ่งเหมือนกับว่าเป็นการสูญเสียข้อมูลบางส่วนโดยจะทำการตัดขอบออกให้เหลือตรงกลางโดยมีพื้นที่ครึ่งหนึ่งของรูปเดิม

รูปที่ 5.16 การเปรียบเทียบภาพ Lena ที่ฝังลายน้ำ 514 บิต (ซ้าย) กับภาพที่ฝังลายน้ำแล้วทำการตัดรูปบางส่วน

ตารางที่ 5.5 จำนวนบิตที่ฝังในภาพ Lena และ BER เมื่อทำการตัดรูปออกบางส่วนก่อนตรวจจับลายน้ำ

ภาพ	บิตที่ฝัง	spreading factor	BER
Lena	484	17	0
	514	**16**	**1.50×10^{-4}**
	549	15	1.82×10^{-3}

จากตารางสรุปได้ว่าเมื่อทำตัดภาพออกบางส่วนให้เหลือพื้นที่ครึ่งหนึ่ง ภาพ Lena สามารถฝังบิตได้มากที่สุด 514 บิต เนื่องจากเมื่อลดค่า spreading factor เป็น 15 ทำให้จำนวนบิตที่ฝังได้ในภาพมีค่ามากขึ้นแต่ค่า BER จะเกิน 1×10^{-3}

5.6 การทดสอบความทนทานโดยทำ JPEG ที่ Quality Factor 80 จากนั้นจึงทำการตัดภาพออกบางส่วน

เมื่อทำการหาตำแหน่งในการฝังลายน้ำในแต่ภาพได้แล้วต่อไปจะทำการหาจำนวนบิตที่มากที่สุดที่จะทำการฝังในภาพได้ โดยภาพที่ฝังลายน้ำแล้วจะนำไปทำการบีบอัดด้วย JPEG ที่ Quality Factor 80 จากนั้นจะนำไปตัดบางส่วนของภาพออกให้เหลือเฉพาะตรงกลางโดยมีพื้นที่ครึ่งหนึ่งของรูปเดิมก่อนที่จะนำภาพไปตรวจสอบลายน้ำโดยให้มี BER ได้ไม่เกิน 1×10^{-3}

รูปที่ 5.17 การเปรียบเทียบภาพ Lena ที่ฝังลายน้ำ 145 บิต (ซ้าย) กับภาพที่ฝังลายน้ำแล้วการบีบอัดด้วย JPEG ที่ Quality Factor 80 และตัดภาพออกบางส่วน

ตารางที่ 5.6 จำนวนบิตที่ฝังในภาพ Lena และ BER เมื่อทำการบีบอัดด้วย JPEG ที่ Quality Factor 80 ตามด้วยการตัดบางส่วนของภาพออกก่อนตรวจจับลายน้ำ

ภาพ	บิตที่ฝัง	spreading factor	BER
Lena	142	56	2.30×10^{-4}
	145	**55**	**4.90×10^{-4}**
	148	54	1.22×10^{-3}

จากตารางสรุปได้ว่าเมื่อทำการบีบอัดด้วย JPEG ที่ Quality Factor 80 จากนั้นจะนำไปตัดบางส่วนของภาพออกให้เหลือเฉพาะตรงกลางโดยมีพื้นที่ครึ่งหนึ่งของรูปเดิม ภาพ Lena สามารถฝังบิตได้มากที่สุด 145 บิต เนื่องจากเมื่อลดค่า spreading factor เป็น 54 ทำให้จำนวนบิตที่ฝังได้ในภาพมีค่ามากขึ้นแต่ค่า BER จะเกิน 1×10^{-3}

5.7 การหาบิตที่ฝังได้มากที่สุดในภาพเมื่อไม่มีการเข้ารหัสหรือใช้ Hamming code (7,4) แทนการเข้ารหัสเทอร์โบ

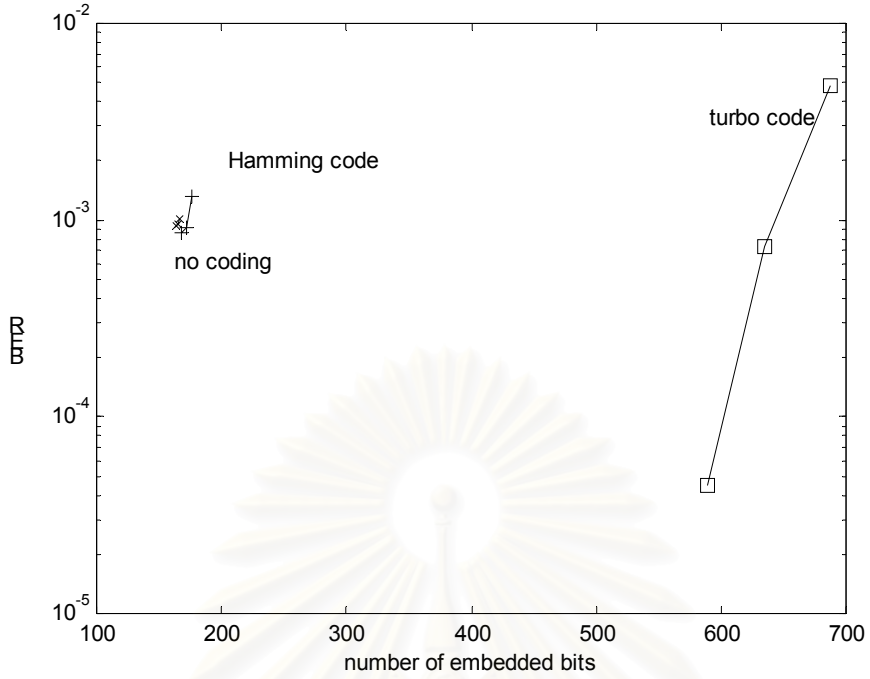
ต่อไปจะทำการทดสอบทำการเปรียบเทียบระหว่างการฝังลายน้ำโดยใช้รหัสเทอร์โบกับเมื่อไม่มีการเข้ารหัสหรือเปลี่ยนจากรหัสเทอร์โบมาเป็น Hamming code (7,4) โดยวิธีฝังจะเหมือนเดิมเพียงแต่ว่าถ้าไม่ฝังรหัสเทอร์โบก็จะตัดขั้นตอนการเข้ารหัสเทอร์โบออกการซ้ำบิตก็จะซ้ำได้มากขึ้น ส่วนถ้าเปลี่ยนไปใช้ Hamming code ก็คือจะเปลี่ยนจากการเข้ารหัสเทอร์โบมาเป็น Hamming code (7,4) ก็คือบิตข้อมูล 4 บิตจะถูกเข้าเป็นรหัส Hamming 7 บิต การซ้ำบิตก็จะทำได้มากขึ้นแต่น้อยกว่าเมื่อไม่เข้ารหัสใดๆเลย

เมื่อดูจากตารางที่ 5.7 ดูการทดสอบความทนทานโดยการบีบอัดแบบ JPEG ที่ Quality Factor 80 จะเห็นว่าภาพ Lena เมื่อใช้รหัสเทอร์โบในการฝังลายน้ำจะสามารถฝังได้มากที่สุด 635 บิต ในขณะที่เมื่อไม่มีการเข้ารหัสใดๆจะสามารถฝังได้มากที่สุด 165 บิต และเมื่อใช้ Hamming code (7,4) จะสามารถฝังได้มากที่สุด 172 บิต โดยทั้งสามแบบถ้ามีการเปลี่ยนค่า spreading factor ให้มีค่าลดลงจะทำให้บิตที่สามารถฝังได้เพิ่มขึ้นแต่จาก ตารางค่า BER จะมีค่าเกิน 1×10^{-3} แล้ว หมายความว่าบิตที่หาได้ของทั้ง 3 วิธีคือบิตที่มากที่สุดที่แต่ละวิธีจะสามารถฝังลงไปในรูปภาพ Lena ได้โดยมีค่า BER ไม่เกิน 1×10^{-3}

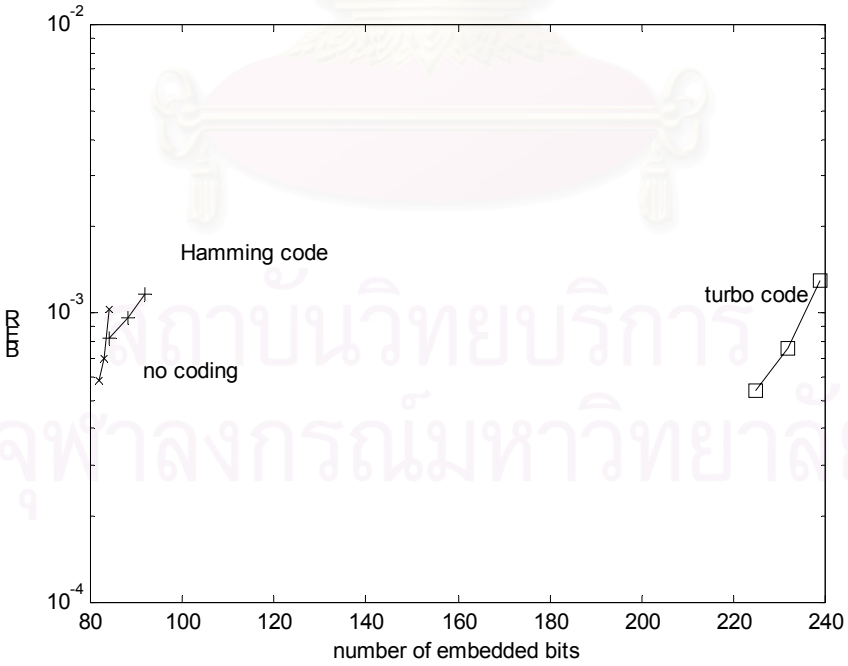
จะเห็นว่าในการทดสอบความทนทานทุกกรณีการนำบิตข้อมูลมาเข้ารหัสเทอร์โบก่อนที่จะฝังไปในภาพจะฝังบิตได้มากกว่าเมื่อไม่เข้ารหัสใดๆเลยและเปลี่ยนมาใช้ Hamming code (7,4) อย่างเห็นได้ชัด โดย Hamming code (7,4) จะดีกว่าเมื่อไม่เข้ารหัสใดๆเลยอยู่เล็กน้อยเท่านั้น

ตารางที่ 5.7 การเปรียบเทียบบิตที่ฝังและค่า BER เมื่อทำการทดสอบความทนแทนแบบต่างๆของภาพ Lena โดยใช้การเข้ารหัสเทอร์โบ,ไม่มีการเข้ารหัส,เข้ารหัสด้วย Hamming code (7,4) เมื่อกำหนดให้มีจุดเสียไม่เกิน 200 จุดเหมือนกัน

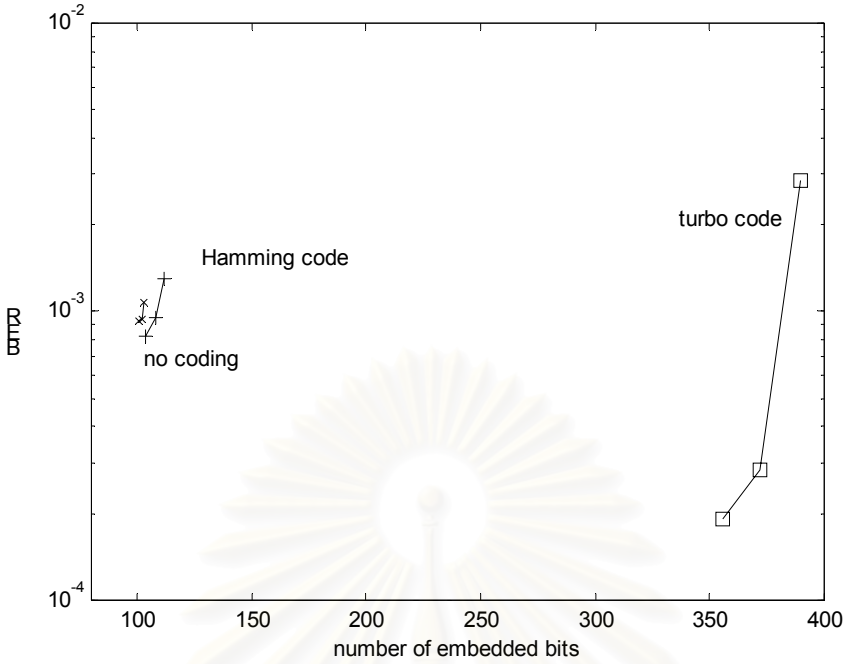
การทดสอบความทนทาน	รหัสเทอร์โบ		ไม่เข้ารหัสใดๆ		Hamming code (7,4)	
	บิตที่ฝัง (spreading factor)	BER	บิตที่ฝัง(spreading factor)	BER	บิตที่ฝัง (spreading factor)	BER
JPEG ที่ Q 80	589 (14)	4.46×10^{-5}	164 (152)	9.31×10^{-4}	168 (85)	8.70×10^{-4}
	635 (13)	**7.35×10^{-4}**	**165 (151)**	**9.45×10^{-4}**	**172 (84)**	**9.21×10^{-4}**
	688 (12)	4.87×10^{-3}	166 (150)	1.02×10^{-3}	176 (83)	1.32×10^{-3}
Median Filtering	225 (36)	5.42×10^{-4}	82 (304)	5.85×10^{-4}	84 (170)	8.20×10^{-4}
	232 (35)	**7.59×10^{-4}**	**83 (303)**	**7.00×10^{-4}**	**88 (169)**	**9.70×10^{-4}**
	239 (34)	1.31×10^{-3}	84 (302)	1.03×10^{-3}	92 (168)	1.17×10^{-3}
Noise Adding	356 (23)	1.92×10^{-4}	101 (247)	9.30×10^{-4}	104 (137)	8.20×10^{-4}
	372 (22)	**2.81×10^{-4}**	**102 (246)**	**9.40×10^{-4}**	**108 (136)**	**9.51×10^{-4}**
	390 (21)	2.83×10^{-3}	103 (245)	1.08×10^{-3}	112 (135)	1.30×10^{-3}
Cropping	484 (17)	0	134 (186)	5.87×10^{-4}	140 (102)	9.30×10^{-4}
	514 (16)	**1.50×10^{-4}**	**135 (185)**	**8.69×10^{-4}**	**144 (101)**	**9.60×10^{-4}**
	549 (15)	1.82×10^{-3}	136 (184)	1.08×10^{-3}	148 (100)	1.36×10^{-3}
Histogram Equalization			510 (49)	7.40×10^{-4}	524 (27)	6.63×10^{-4}
	2077 (4)	**0**	**520 (48)**	**7.47×10^{-4}**	**528 (26)**	**9.02×10^{-4}**
	2500 (3)	2.59×10^{-1}	531 (47)	1.03×10^{-3}	532 (25)	4.01×10^{-3}
JPEG ที่ Q 80 และ Cropping	142 (56)	2.30×10^{-4}	60 (416)	7.80×10^{-4}	60 (238)	7.70×10^{-4}
	145 (55)	**4.90×10^{-4}**	**61 (415)**	**9.00×10^{-4}**	**64 (237)**	**8.40×10^{-4}**
	148 (54)	1.22×10^{-3}	62 (414)	1.15×10^{-3}	68 (236)	1.10×10^{-3}



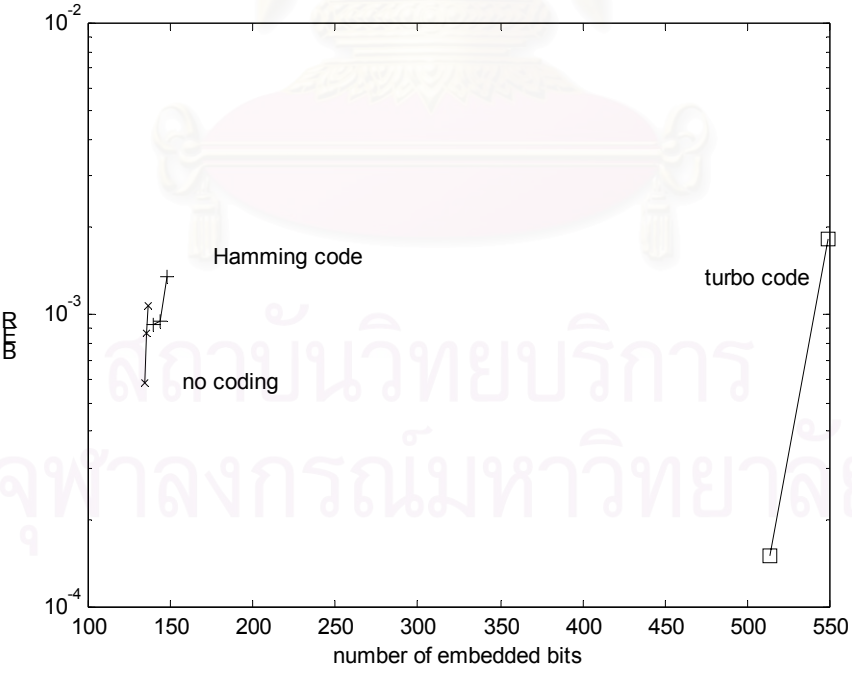
รูปที่ 5.18 กราฟแสดงความสัมพันธ์ของบิตที่ฝังกับ BER ที่ได้ของภาพ Lena เมื่อทำการทดสอบความทนทานโดยทำการบีบอัดแบบ JPEG ที่ Quality Factor 80 โดยเปรียบเทียบกันระหว่างรหัสเทอร์โบ, ไม่มีการเข้ารหัส, Hamming code (7,4)



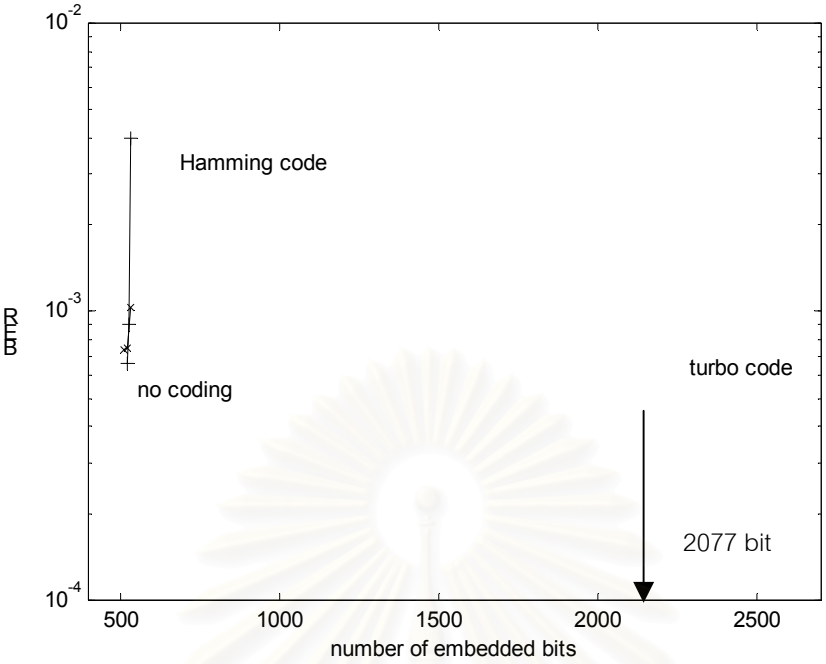
รูปที่ 5.19 กราฟแสดงความสัมพันธ์ของบิตที่ฝังกับ BER ที่ได้ของภาพ Lena เมื่อทำการทดสอบความทนทานโดยใช้ Median Filtering โดยเปรียบเทียบกันระหว่างรหัสเทอร์โบ, ไม่มีการเข้ารหัส, Hamming code (7,4)



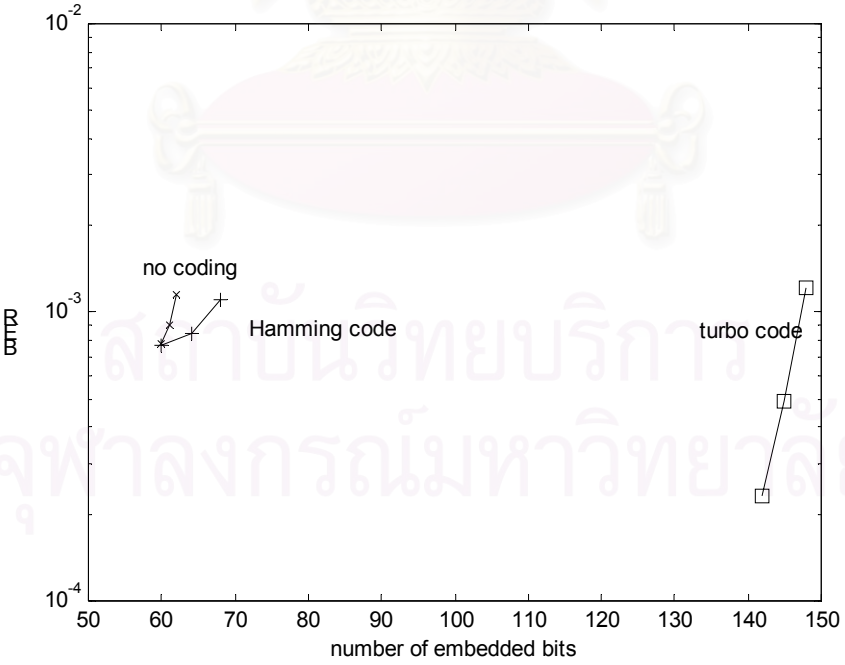
รูปที่ 5.20 กราฟแสดงความสัมพันธ์ของบิตที่ฝังกับ BER ที่ได้ของภาพ Lena เมื่อทำการทดสอบความทนทานโดยใส่สัญญาณรบกวนแบบเกาส์ โดยเปรียบเทียบกันระหว่างรหัสเทอร์โบ, ไม่มีการเข้ารหัส, Hamming code (7,4)



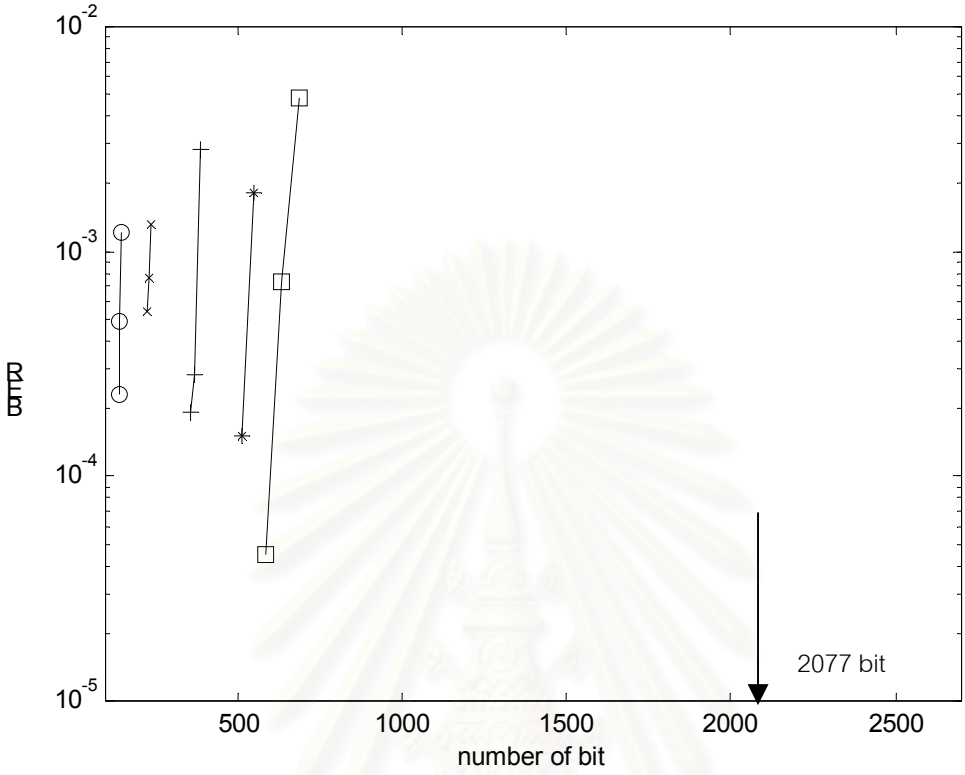
รูปที่ 5.21 กราฟแสดงความสัมพันธ์ของบิตที่ฝังกับ BER ที่ได้ของภาพ Lena เมื่อทำการทดสอบความทนทานโดยการตัดภาพออกบางส่วน โดยเปรียบเทียบกันระหว่างรหัสเทอร์โบ, ไม่มีการเข้ารหัส, Hamming code (7,4)



รูปที่ 5.22 กราฟแสดงความสัมพันธ์ของบิตที่ฝังกับ BER ที่ได้ของภาพ Lena เมื่อทำการทดสอบความทนทานโดยการทำ Histogram Equalization โดยเปรียบเทียบกันระหว่างรหัสเทอร์โบ, ไม่มีการเข้ารหัส, Hamming code (7,4)



รูปที่ 5.23 กราฟแสดงความสัมพันธ์ของบิตที่ฝังกับ BER ที่ได้ของภาพ Lena เมื่อทำการทดสอบความทนทานโดยการบีบอัดด้วย JPEG ที่ Quality Factor 80 และตัดภาพออกบางส่วนโดยเปรียบกันระหว่างรหัสเทอร์โบ, ไม่มีการเข้ารหัส, Hamming code (7,4)



รูปที่ 5.24 กราฟแสดงความสัมพันธ์ของบิตที่ฝังกับ BER ที่ได้ของภาพ Lena เมื่อทำการทดสอบความทนทานโดยการบีบอัดด้วย JPEG ที่ Quality Factor 80 ,Median Filtering, ใส่สัญญาณรบกวนแบบเกาส์, Histogram Equalization, ตัดภาพออก, บีบอัดด้วย JPEG ที่ Quality Factor 80 และตัดภาพออก

จุด □ แสดง JPEG ที่ Q 80

จุด x แสดง Median Filtering

จุด + แสดง การใส่สัญญาณรบกวนแบบเกาส์

จุด * แสดง การตัดภาพออก

จุด o แสดง JPEG ที่ Q 80 และตัดภาพออก

→ แสดง Histogram Equalization

บทที่ 6
บทสรุปและข้อเสนอแนะ

6.1 บทสรุป

งานวิจัยนี้ปรับปรุงวิธีการใส่ลายน้ำโดยในปัจจุบันวิธีการใส่ลายน้ำส่วนมากจะเป็นแบบ semi-private watermarking คือไม่ต้องใช้รูปต้นฉบับและเป็นการตัดสินว่ามีลายน้ำหรือไม่เท่านั้น โดยงานวิจัยนี้ได้คิดวิธีการใส่ลายน้ำแบบ public watermarking คือไม่ต้องใช้รูปต้นฉบับในการตรวจสอบลายน้ำและเป็นการฝังบิตข้อมูลลงไปจริงๆ โดยในการตรวจสอบลายน้ำนั้นบิตที่ตรวจวัดได้จะมีความผิดพลาด ดังนั้นจึงมีการนำเอารหัสแก้ไขความผิดพลาดมาใช้โดยจะเลือกเอารหัสเทอร์โบมาใช้เนื่องจากแก้ไขความผิดพลาดได้ดีที่ SNR ต่ำๆ โดยได้ศึกษาในประเด็นต่างๆดังต่อไปนี้

1. หาวิธีการฝังและตรวจจับลายน้ำโดยใช้รหัสเทอร์โบในการแก้ไขบิตที่ผิดพลาด
2. หาช่วงสัมประสิทธิ์ในการฝังลายน้ำในรูป Lena, Baboon, Cameraman, Boat
3. ทดสอบความทนทานต่อการประมวลผลแบบต่างๆเพื่อหาว่าสามารถฝังลายน้ำได้สูงสุดกี่บิตโดยกำหนดให้มี BER ไม่เกิน 1×10^{-3} ดังต่อไปนี้

การทดสอบความทนทาน	ภาพที่ทดสอบ
JPEG ที่ Quality Factor 80	Lena, Baboon, Cameraman, Boat
Median Filterng	Lena
ใส่สัญญาณรบกวนแบบเกาส์	Lena
Histogram Equalization	Lena
ตัดรูปออกบางส่วน	Lena
JPEG ที่ Quality Factor 80 และตัดรูปออกบางส่วน	Lena

เมื่อพิจารณาเฉพาะการทดสอบความทนทานโดยนำภาพที่ฝังลายน้ำมาทำการบีบอัดด้วย JPEG ที่ Q 80 แล้วหาว่าสามารถฝังบิตได้มากที่สุดกี่บิตโดยกำหนดให้มี BER ไม่เกิน 1×10^{-3} โดยทดสอบกับรูปมาตรฐาน 4 รูป จากผลการทดสอบพบว่าภาพ Baboon สามารถฝังบิตได้น้อยกว่าภาพอื่นๆมากซึ่งอาจเนื่องจากภาพ Baboon มีส่วนประกอบที่อยู่ในความถี่สูงอยู่มากแต่ image restoration filter ทำงานได้ดีในภาพที่มีส่วนประกอบอยู่ในความถี่ต่ำ ซึ่งเมื่อนำมาใช้กับภาพ Baboon อาจทำการลบข้อมูลสำคัญของภาพไปด้วยทำให้การสร้างภาพต้นฉบับนั้นผิดพลาดมาก แม้ว่าจะใส่ได้มากโดยสังเกตไม่ได้แต่ก็ถูกทำลายได้ง่ายเช่นกัน

เมื่อพิจารณาการทดสอบความทนทานแบบอื่นๆโดยกำหนดให้ฝังลายน้ำในภาพ Lena จากผลการทดสอบเมื่อพิจารณาจากภาพที่ผ่านการทดสอบความทนทานและภาพขยายจะเห็นแนวโน้มว่าเมื่อภาพที่ใส่ลายน้ำได้ถูกเปลี่ยนแปลงให้ต่างจากของเดิมมากขึ้นก็จะทำให้บิตที่ใส่ได้นั้นมีจำนวนน้อยลง เช่นการทดสอบความทนทานสองอย่างร่วมกันจะทำให้ใส่บิตได้น้อยลงกว่าการทดสอบความทนทานแบบเดียว หรือ Median Filtering ที่ดูจากภาพขยายจะเห็นว่าจุดภาพนั้นเปลี่ยนแปลงไปจากเดิมก่อนทำ Filtering อย่างมากจนสังเกตได้ก็สามารถใส่บิตได้น้อย

เมื่อเปรียบการใช้รหัสเทอร์โบกับเมื่อไม่มีการเข้ารหัสใดๆและใช้รหัส Hamming code (7,4) จะเห็นได้อย่างชัดเจนว่ารหัสเทอร์โบนั้นเหมาะสมกับการนำไปใช้ในการฝังลายน้ำอย่างมาก เนื่องจากสามารถฝังบิตได้มากกว่าอีก 2 แบบมากประมาณ 3 เท่าขึ้นไปและจะเห็นว่า Hamming code นั้นดีกว่าเมื่อไม่มีการเข้ารหัสใดๆเพียงเล็กน้อยเท่านั้น

งานวิจัยนี้ได้พยายามหาว่าบิตที่ใส่ได้มากที่สุดในภาพควรมีค่าประมาณเท่าใดเมื่อทำการทดสอบความทนทานแบบต่างๆและหวังว่าต่อไปจะสามารถพัฒนาให้ดีขึ้นกว่านี้ได้อีก ปัญหาที่เกิดจากการทำงานคือการหาระดับการมองเห็นที่ใช้ในการบอกว่าควรจะฝังลายน้ำลงไปได้แรงมากที่สุดเท่าไรจึงจะเหมาะสม ซึ่งปัญหานี้ยังเป็นปัญหาที่พบในงานวิจัยเรื่อง watermarking ในขณะนี้ โดยในงานวิจัยได้พยายามกำหนดให้ระดับการมองเห็นของลายน้ำที่ฝังเท่ากันในแต่ละรูปโดยใช้โปรแกรมของ Girod ซึ่งจะบอกจุดที่เสียของภาพที่ฝังลายน้ำโดยจะนำจำนวนจุดเหล่านี้มากำหนดระดับการมองเห็นในงานวิจัย ส่วนในการทดสอบความทนทานจะไม่สามารถกำหนดให้การทดสอบความทนทานแต่ละวิธีให้มีระดับการเปลี่ยนแปลงที่เท่ากันได้

งานวิจัยนี้สามารถนำไปใช้ในการฝังบิตข้อมูลแสดงความเป็นเจ้าของลงไปในรูปภาพดิจิตอลได้ โดยงานวิจัยจะทำการหากระบวนการในการฝังบิตข้อมูลลงในรูปภาพให้ได้จำนวนบิตข้อมูลมากที่สุดภายใต้ระดับการมองเห็นและการทดสอบความทนทานที่กำหนด ทำให้สามารถรู้ได้ว่ารูปภาพแต่ละรูปสามารถฝังบิตข้อมูลได้เท่าไหร่ มีข้อดีคือเป็นการฝังบิตลงไปจริงๆและไม่ต้องใช้ต้นฉบับในการตรวจสอบและยังมีการใช้รหัสเทอร์โบมาช่วยแก้ไขความผิดพลาด ส่วนข้อเสียของงานวิจัยนี้คือจะมีความทนทานน้อยกว่า private watermarking และ semi-private watermarking และการใช้รหัสเทอร์โบทำให้โปรแกรมต้องใช้เวลาทำงานนานขึ้นด้วย รวมทั้งวิธีในงานวิจัยนี้ยังไม่สามารถใช้กับภาพสีได้ซึ่งจะต้องมีการปรับปรุงต่อไป

6.2 ข้อเสนอแนะ

งานวิจัยในขั้นต่อไปคือ

1. งานวิจัยที่เกี่ยวข้องกับ image restoration คืออาจใช้เทคนิคอื่นที่ทำให้การจำลองรูปภาพต้นฉบับนั้นได้ผลที่เหมือนกับภาพต้นฉบับมากขึ้น

2. งานวิจัยที่เกี่ยวข้องกับการปรับปรุงการเข้ารหัสเทอร์โบโดยอาจปรับปรุงในส่วนตัวสลับหรือปรับปรุงขั้นตอนการถอดรหัสเพื่อดูผลว่าจะสามารถฝังบิตได้มากขึ้นหรือไม่
3. งานวิจัยที่เกี่ยวข้องกับการวัดระดับการมองเห็นของภาพที่ใส่ลายน้ำลงไปว่าแตกต่างจากภาพต้นฉบับมากเพียงใด
4. งานวิจัยในการฝังลายน้ำลงในภาพเคลื่อนไหว

รายการอ้างอิง

1. J. Zhao and E. Koch. A Generic Digital Watermarking Model. Comput & Graphics. Vol. 22, No. 4 (1998): 397-403.
2. I. Cox, M. L. Miller and Andrew L. Mckellips. Watermarking as Communications with Side Information. Proceeding of the IEEE Special Issue on Protection of Multimedia Content. Vol 87, No. 7 (July 1999): 1127-1141.
3. F. Hartung and M. Kutter. Multimedia Watermarking Techniques. Proceeding of the IEEE Special Issue on Protection of Multimedia Content. Vol. 87, No. 7 (July 1999): 1079-1107.
4. F. Peticolas and R. Anderson. Information Hiding–A Survey. Proceeding of the IEEE Special Issue on Protection of Multimedia Content. Vol. 87, No. 7 (July1999): 1062-1078.
5. I.Cox, J. Kilian, T. Leighton and T. Shamoon. Secure Spread Spectrum Watermarking for Images, Audio and Video. Proc. IEEE Int. Conf. Image Processing (ICIP 96). (September 1996): 243-246.
6. M. Barni, F. Bartolini, V. Cappellini and A. Piva. A DCT-Domain System for Robust Image Watermarking. Signal Processing (Special Issue on Watermarking). (May 1998): 357-372.
7. N. Nikolaidis and I. Pitas. Robust Image Watermarking in the Spatial Domain. Signal Processing. (May 1998): 385-403.
8. F. Hartung and B. Girod. Digital Watermarking of Uncompressed and Compressed Video. Signal Processing (Special Issue on Copyright Protection and Access Control for Multimedia Services). (1998): 283-301.
9. Lisa M. Marvel, Charles G. Boncelet and Charles T. Retter. Reliable Blind Information Hiding for Images. International Workshop on Information Hiding. (April 1998): 48-61.
10. Lisa M. Marvel. Image Steganography for Hidden Communication. Ph.D. Thesis,U. of Delaware. (May 1999).
11. A. Tirkel, G. Rankin, R. Van Schyndel, W. Ho, N. Mee and C. Osborne. Electronic Watermark. Proc. DICTA 1993. (December 1993): 666-672.

12. G. C. Langelaar, J. C. A. Van Der Lubbe and J. Biemond. Copy Protection for Multimedia Data Based on Labeling Techniques. Proc. 17th Symposium on Information Theory. (May 1996): 159-165.

13. Won-Gyum Kim, Chan-Woon Lee and Won Don Lee. A Watermarking Scheme for both Spatial and Frequency Domain to extract the Seal Image without the Original Image. ISSPA 99. (22-25 August 1999): 293-296.

14. F. M. Boland, J. J. K. Ruanaidh and W. J. Dowling. Watermarking Digital Images for Copyright Protection. IEE Proceedings on Vision, Signal and Image Processing. (August 1996): 250-256.

15. J. Smith and B. Comiskey. Modulation and Information Hiding in Images. Proc. First International Workshop on Information Hiding, Lecture Notes on Computer Science. (June 1996): 207-226.

16. C. Berrou, A. Glavieus and P. Thitimajshina. Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes. IEEE Trans. Commun. (October 1996): 1261-1271.

17. B. Sklar. A Primer on Turbo Code Concepts. IEEE Communications Magazines. (December 1997): 94-102.

18. Matthew C. Valenti. Iterative Detection and Decoding for Wireless Communications. Ph.D. Dissertation, Virginia Tech. (July 1999).

19. William E. Ryan. A Turbo Code Tutorial. http://www.ece.arizona.edu/~ryan/.

20. B. Girod. The Information Theoretical Significance of Spatial and Temporal Masking in Video Signals. Proc. of the SPIE Human Vision, Visual Processing and Digital Display. Vol. 1077 (1989): 178-187.

ประวัติผู้เขียนวิทยานิพนธ์

นาย วิทิต พงศ์พิโรดม เกิดเมื่อวันที่ 18 มีนาคม พ.ศ. 2521 ที่ อ.เมือง จ.ลำปาง สำเร็จการศึกษาชั้นมัธยมปลายจากโรงเรียนบุญวาทย์วิทยาลัย จ.ลำปาง สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้าจาก จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2541 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2542