ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)

ปริญญา

วิศวกรรมไฟฟ้า	วิศวกรรมไฟฟ้า
สาขา	**ภาควิชา**

เรื่อง การออกแบบการสลับลำดับข้อมูลแบบสุ่มสำหรับการสื่อสารข้อมูลภาพ MPEG-4 ในช่องสัญญาณไร้สายภายในอาคาร

Design of Random Interleavers for MPEG-4 Image Indoor Wireless Transmission System

นามผู้วิจัย นางสาวอิสรีย์ สรรพสิริโสภณ

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(ผู้ช่วยศาสตราจารย์ศรีจิตรา เจริญลาภนพรัตน์, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(รองศาสตราจารย์ณัฏฐกา หอมทรัพย์, Ph.D.)

หัวหน้าภาควิชา .....

(รองศาสตราจารย์มงคล รักษาพัชรวงศ์, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรอง

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, Ph.D.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

การออกแบบการสลับลำดับข้อมูลแบบสุ่มสำหรับการสื่อสารข้อมูลภาพ MPEG-4 ในช่องสัญญาณไร้สายภายในอาคาร

Design of Random Interleavers for MPEG-4 Image Indoor Wireless Transmission System

โดย

นางสาวอิสรีย์ สรรพสิริโสภณ

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)

พ.ศ. 2551

อิสรีย์ สรรพสิริโสภณ 2551: การออกแบบการสลับลำดับข้อมูลแบบสุ่มสำหรับการสื่อสารข้อมูลภาพ MPEG-4 ในช่องสัญญาณไร้สายภายในอาคาร ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า) สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: ผู้ช่วยศาสตราจารย์ศรีจิตรา เจริญลาภนพรัตน์, Ph.D. 98 หน้า

งานวิจัยนี้เป็นการปรับปรุงระบบส่งข้อมูลภาพ MPEG-4 ในช่องสัญญาณไร้สายภายในอาคาร โดยออกแบบการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม เพื่อทดสอบกับช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า และ แบบเร็ว ผลการทดลองที่ได้จากการประมวลผลพบว่า การนำกระบวนการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม มาใช้กับช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า และ แบบเร็ว สามารถเพิ่มประสิทธิภาพให้กับระบบได้สูงถึง 17 dB และ 2.31 dB ตามลำดับ นอกจากนี้ได้ทำการปรับปรุงวิธีการออกแบบการสลับลำดับข้อมูลแบบสุ่ม เพื่อลดเวลาในการประมวลผลจาก 43.73 วินาทีต่อภาพ เป็น 4.07 วินาทีต่อภาพ ซอฟต์แวร์ประยุกต์ที่นำมาใช้ในการจำลองระบบถูกพัฒนาด้วยโปรแกรมคอมพิวเตอร์ภาษาจาวา

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Itsaree Sapasirisopon 2008: Design of Random Interleavers for MPEG-4 Image Indoor Wireless Transmission System. Master of Engineering (Electrical Engineering), Major Field: Electrical Engineering, Department of Electrical Engineering. Thesis Advisor: Assistant Professor Srijidtra Charoenlarpnopparut, Ph.D. 98 pages.

This research improves MPEG-4 image indoor wireless transmission system using random and semi-random interleavers. The system is tested for slow and fast flat fading channel. The simulation results in PSNRs show the better improvement of about 17 dB and 2.31 dB for slow and fast flat fading channel, respectively. In addition, our designed random interleaver is modified in order to decrease the processing time. It can be reduced from 43.73 second per image to 4.07 second per image. The application software which is used to simulate our system, is developed by JAVA computer program.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ข้าพเจ้าขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ศรีจิตรา เจริญลาภนพรัตน์ ประธานกรรมการที่ปรึกษา ที่ได้ช่วยเหลือในการวางแผนงานวิจัย ให้คำปรึกษาแนะนำและตรวจสอบแก้ไขข้อบกพร่อง ขอกราบขอบพระคุณ รองศาสตราจารย์ ดร.ณัฏฐกา หอมทรัพย์ และ รองศาสตราจารย์ ดร.มงคล รักษาพัชรวงศ์ ที่ช่วยเหลือตรวจสอบพร้อมแนะนำการจัดทำรูปแบบผลการทดลองในวิทยานิพนธ์ให้สำเร็จลุล่วงไปด้วยดี

ข้าพเจ้าขอกราบขอบพระคุณ คุณแม่ คุณยายและคุณตาของข้าพเจ้าที่สนับสนุนและให้กำลังใจ ขอบคุณคุณพิรัลณญา อินปา ที่เป็นกำลังใจ แนะนำและช่วยแก้ไขการทำวิทยานิพนธ์ฉบับนี้ให้สำเร็จลุล่วงไปได้ด้วยดี และหากวิทยานิพนธ์ฉบับนี้มีข้อบกพร่องประการใด ข้าพเจ้ายินดีรับข้อเสนอแนะและขออภัยมา ณ ที่นี้ด้วย

อิสรีย์ สรรพสิริโสภณ

ตุลาคม 2551

สารบัญ

หน้า

สารบัญตาราง

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

คำอธิบายสัญลักษณ์และคำย่อ

EZW	=	Embedded Zerotree Wavelet
DWT	=	Discrete Wavelet Transform
ML	=	Maximum Likelihood
MAP	=	Maximum a posteriori
HFS	=	Higher Frequency Subband
LFS	=	Lowest Frequency Subband
DPCM	=	Differential Pulse Code Modulation
ZTR	=	Zerotree Root
IZ	=	Isolated Zero
VZTR	=	Valued Zerotree Root
VAL	=	Value
CRC	=	Cyclic Redundancy Check
SOT	=	Starting Spatial Orientation Tree
MLSE	=	Maximum Likelihood Sequence Estimator
CPM	=	Continuous Phase Modulation
TCM	=	Trellis-coded Modulation
AWGN	=	Additive White Gaussian Noise
PSNR	=	Peak Signal to Noise Ratio
WER	=	Word Error Rate

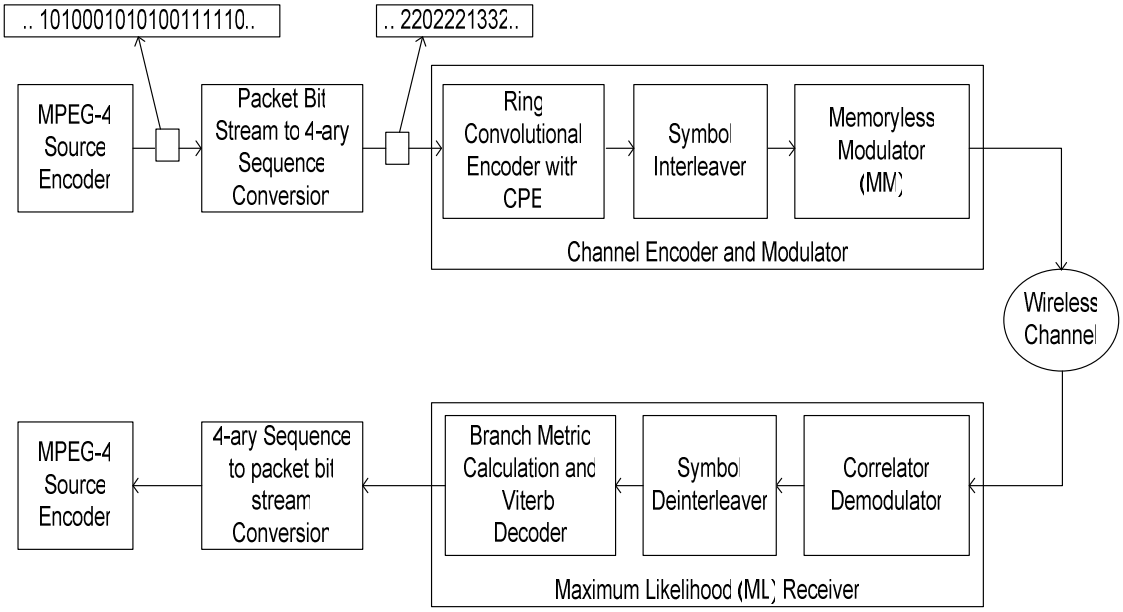
การออกแบบการสลับลำดับข้อมูลแบบสุ่มสำหรับการสื่อสารข้อมูลภาพ MPEG-4 ในช่องสัญญาณไร้สายภายในอาคาร

Design of Random Interleavers for MPEG-4 Image Indoor Wireless Transmission System

คำนำ

ปัจจุบันระบบสื่อสารข้อมูลดิจิตอลเป็นเทคโนโลยีที่มีความสำคัญ และ นำมาใช้งานเพื่อความสะดวกในการรับส่งข้อมูล และ มัลติมีเดีย ผ่านคลื่นแม่เหล็กไฟฟ้า เส้นใยนำแสง และตัวกลางอื่นๆ ดังนั้น คุณสมบัติของช่องสัญญาณ ตัวกลางการส่งผ่านข้อมูล และคุณภาพของอุปกรณ์ จึงเป็นปัจจัยที่ส่งผลโดยตรงต่อระบบ ซึ่งอาจทำให้ข้อมูลที่ได้รับเกิดความผิดเพี้ยนได้ วิธีการป้องกัน คือ การเข้ารหัสข้อมูล และ การเพิ่มกำลังส่งเครื่องส่งต้นทาง นอกจากนี้ยังมีความผิดพลาดของข้อมูล ที่เกิดจากสัญญาณรบกวนที่มีลักษณะเป็นกลุ่มก้อนในช่วงเวลาสั้นๆ หรือเรียกว่า ความผิดพลาดแบบต่อเนื่อง อาจเกิดจากปัจจัยภายนอกระบบ เช่น สภาพอากาศ เป็นต้น ดังนั้นวิธีการป้องกันข้อมูล และ การเพิ่มกำลังส่ง อาจไม่สามารถแก้ความผิดพลาดของข้อมูลแบบต่อเนื่องได้ จึงมีนักวิจัยนำวิธีการสลับลำดับข้อมูลมาใช้ เพื่อลดจำนวนความผิดพลาดของข้อมูลแบบต่อเนื่อง เช่น Dolinar and Divsalar (1995) นำกระบวนการสลับลำดับข้อมูลมาใช้กับรหัสเทอร์โบเพื่อกระจายค่าน้ำหนักหรือ Weight โดยพบว่าสามารถแก้ปัญหาดังกล่าวได้ และ ทำให้ประสิทธิภาพของระบบดีขึ้น นอกจากนี้ การสลับลำดับข้อมูลถูกนำมาประยุกต์ใช้งานกับการส่งสัญญาณโทรทัศน์ระบบดิจิตอล ระบบสื่อสารผ่านดาวเทียม และ ระบบโทรศัพท์เคลื่อนที่

งานวิจัยนี้สืบเนื่องมาจากงานวิจัยของ Mahapakulchai and Van Dyck (2004) โดยมีแนวคิด คือ ต้องการลดการถูกทำลายของข้อมูลภาพ ที่เกิดจากสัญญาณรบกวนบนช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณหรือ Flat Fading โดยการออกแบบการสลับลำดับข้อมูลแบบสุ่ม และกึ่งสุ่ม เพื่อนำมาปรับใช้กับระบบ การวัดประสิทธิภาพของระบบ วิเคราะห์จากค่าเฉลี่ย Peak Signal to Noise Ratioหรือ PSNR และ Word Error Rateหรือ WER ภาพที่ 1 แสดงโครงสร้างของระบบส่งข้อมูลภาพ ประกอบด้วย 5 ส่วนหลัก คือ



ภาพที่ 1 โครงสร้างระบบส่งข้อมูลภาพในช่องสัญญาณไร้สาย

ส่วนแรก การเข้าและถอดรหัสข้อมูลภาพแบบ MPEG-4 เป็นวิธีการบีบอัดข้อมูลภาพที่มีประสิทธิภาพ โดยแยกข้อมูลภาพออกเป็น 2 ส่วน คือ ส่วนที่มีความถี่ต่ำหรือ LFS และ ส่วนที่มีความถี่สูงหรือ HFS กลุ่มข้อมูล LFS เป็นตัวแทนโครงสร้างหลักของข้อมูลภาพ และมีความสำคัญมาก ถ้าสูญเสียข้อมูลในส่วนนี้ไป จะทำให้ประสิทธิภาพของระบบลดลง ส่วนกลุ่มข้อมูล HFS เป็นตัวแทนรายละเอียดของข้อมูลภาพ ดังนั้นการเข้าและถอดรหัสข้อมูลภาพแบบเทลลิส อาจทำแยกกัน เพื่อให้การส่งข้อมูลภาพผ่านช่องสัญญาณมีความผิดพลาดน้อยที่สุด ส่วนที่สอง คือ การเปลี่ยนข้อมูลภาพจากบิตเป็นสัญลักษณ์ โดยที่ 1 สัญลักษณ์เท่ากับ 2 บิต เป็นการออกแบบเพื่อสามารถเข้ารหัสแบบริงได้ เนื่องจากการเข้ารหัสแบบริงสามารถออกแบบร่วมกับการมอดูเลตแบบเฟสของสัญญาณมีความต่อเนื่อง และ เหมาะสมกับการส่งสัญญาณผ่านช่องสัญญาณไร้สาย ส่วนที่สาม คือ การสลับลำดับข้อมูลภาพและการสลับลำดับข้อมูลภาพกลับ เป็นการเปลี่ยนตำแหน่งข้อมูลเพื่อทำให้ข้อมูลเดิมที่อยู่ติดกันแยกออกจากกัน โดยลดจำนวนความผิดพลาดแบบต่อเนื่องของข้อมูลภาพ ในช่วงเวลาสั้นๆ ประกอบด้วยการสลับลำดับข้อมูลภาพแบบสุ่ม และ กึ่งสุ่ม ส่วนที่สี่ คือ การเข้าและถอดรหัสช่องสัญญาณแบบเทลลิสประกอบด้วย 2 ส่วน คือ การเข้ารหัสช่องสัญญาณริงคอนโวลูชันนอล และ การมอดูเลตแบบที่เฟสของสัญญาณมีความต่อเนื่อง โดยใช้ทฤษฎีการตรวจจับสัญญาณแบบ ML และ MAP เพื่อหาขนาดในแต่ละเส้นทางบนเทลลิสร่วมกับการถอดรหัสวีเทอร์บี และในส่วนสุดท้าย คือ ช่องสัญญาณแบบที่มีการเลือนหายทางขนาดของสัญญาณ เกิดขึ้นใน 2 ลักษณะ คือ แบบช้า และ แบบเร็ว

งานวิจัยนี้ได้ทำการออกแบบการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม โดยมีวิธีการสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence และ การเรียกใช้ฟังก์ชันแบบสุ่มที่มีอยู่ในโปรแกรมภาษาจาวา จำลองช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า และแสดงเวลาในการประมวลผลของระบบ

วัตถุประสงค์

1. ศึกษาและออกแบบการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม ในระบบส่งข้อมูลภาพผ่านช่องสัญญาณไร้สาย ที่คำนึงถึงความยาวของกลุ่มข้อมูล และ การกระจายลำดับข้อมูล เพื่อให้เหมาะสมกับระบบที่ออกแบบไว้แล้ว

2. สามารถนำการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม มาใช้ร่วมกับระบบส่งข้อมูลภาพที่มีอยู่แล้วได้

3. ศึกษาค้นคว้า และ เขียนโปรแกรมการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม โดยนำมาใช้กับระบบส่งข้อมูลภาพ เพื่อวัดประสิทธิภาพของระบบ

4. สามารถนำการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม ทดสอบกับช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า เพื่อปรับปรุงประสิทธิภาพของระบบ

5. เพื่อให้เกิดผลงาน และ พัฒนาโปรแกรมของระบบส่งข้อมูลภาพ สามารถนำไปใช้งานได้จริง

การตรวจเอกสาร

1. บทนำ

ระบบสื่อสารข้อมูลภาพต้นแบบ ประกอบด้วย 3 ส่วนหลักๆ ส่วนแรกการจำลองการเข้าและถอดรหัสข้อมูลภาพ ส่วนที่สองการจำลองการเข้าและถอดรหัสช่องสัญญาณ และ ส่วนสุดท้าย การจำลองช่องสัญญาณไร้สาย วิธีการเข้าและถอดรหัสข้อมูลภาพ EZW เป็นวิธีการที่ให้ประสิทธิภาพสูง เริ่มต้นสร้างโดย Shapiro (1993) เป็นการบีบอัดข้อมูลแบบ MPEG-4 การเข้ารหัสช่องสัญญาณ ได้นำงานวิจัยของ Ungerboeck (1982) โดยศึกษาและออกแบบร่วมกับการเข้ารหัสคอนโวลูชันกับการมอดูเลต CPFSK จากนั้นเริ่มออกแบบระบบส่งข้อมูลภาพโดย Mahapakulchai and Van dyck (2000) เป็นการนำการถอดรหัสแบบ MAP เข้ามาใช้กับระบบส่งข้อมูลภาพ พบว่าประสิทธิภาพของระบบโดยเฉลี่ยดีขึ้น และ Mahapakulchai and Van dyck (2004) มีการเปลี่ยนแปลงรูปภาพที่ใช้ในการทดสอบ ให้มีความหลากหลายมากขึ้น นอกจากนี้ Mahapakulchai (2007) ออกแบบการเข้ารหัสช่องสัญญาณแบบริงคอนโวลูชั่นใช้ร่วมกับการถอดรหัสแบบ MAP เพื่อหาตัวถอดรหัสที่ดีที่สุด

ระบบส่งข้อมูลภาพ เริ่มนำวิธีการสลับลำดับข้อมูลมาใช้เพื่อเพิ่มประสิทธิภาพของระบบ โดยเริ่มจาก Mahapakulchai and Thongnumpen (2007) นำกระบวนการสลับลำดับข้อมูลแบบบล็อกมาใช้กับระบบ แต่เนื่องจากความยาวของข้อมูลภาพไม่คงที่ ดังนั้นวิธีการนี้จึงมีความยุ่งยากในการคำนวณขนาดของเมตริกซ์การสลับ Mahapakulchai and Sapasirisopon (2008a) จึงได้ทำการออกแบบการสลับลำดับข้อมูลแบบสุ่ม ผลที่ได้มีประสิทธิภาพใกล้เคียงกันแต่สามารถลดปัญหาความยุ่งยากของการออกแบบเมตริกซ์การสลับ และ Mahapakulchai and Sapasirisopon (2008b) นำการถอดรหัสแบบ MAP มาทดสอบกับระบบที่มีการสลับลำดับข้อมูลแบบสุ่ม ผลที่ได้มีประสิทธิภาพดีขึ้นเล็กน้อย นอกจากนี้ ยังมีการปรับปรุงประสิทธิภาพของระบบอย่างต่อเนื่อง โดยนำการสลับลำดับข้อมูลแบบกึ่งสุ่ม S = 10 และ S = 17 มาใช้กับระบบส่งภาพ

การปรับปรุงระบบส่งข้อมูลภาพของโครงงานวิจัยนี้ เริ่มจากศึกษาทฤษฎีของกระบวนการสลับลำดับข้อมูล มีอยู่ 3 วิธี คือ การสลับลำดับข้อมูลแบบบล็อก การสลับลำดับข้อมูลแบบคอนโวลูชั่น และ การสลับลำดับข้อมูลแบบสุ่ม เพื่อกระจายลำดับของข้อมูลที่ส่งผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า ซึ่งอาจทำให้ข้อมูลมีลักษณะของความผิดพลาดแบบต่อเนื่อง ดังนั้นการกระจายข้อมูลออกจากกัน ทำให้ข้อมูลที่ได้จากการเข้ารหัสไม่มี

ความสัมพันธ์ซึ่งกันและกัน ดังนั้นตัวถอดรหัสสามารถแก้ไขความผิดพลาดได้ ถ้าไม่มีการกระจายข้อมูลออกจากกันในกรณีที่มีความผิดพลาดแบบต่อเนื่อง จะทำให้ตัวถอดรหัสตัดสินใจผิดพลาด ส่งผลให้ข้อมูลเกิดความผิดพลาด จากนั้นออกแบบกระบวนการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม โดยอาศัยหลักการของงานวิจัย Dolinar and Divsalar (1995) และ Fragouli and Wesel (1999)

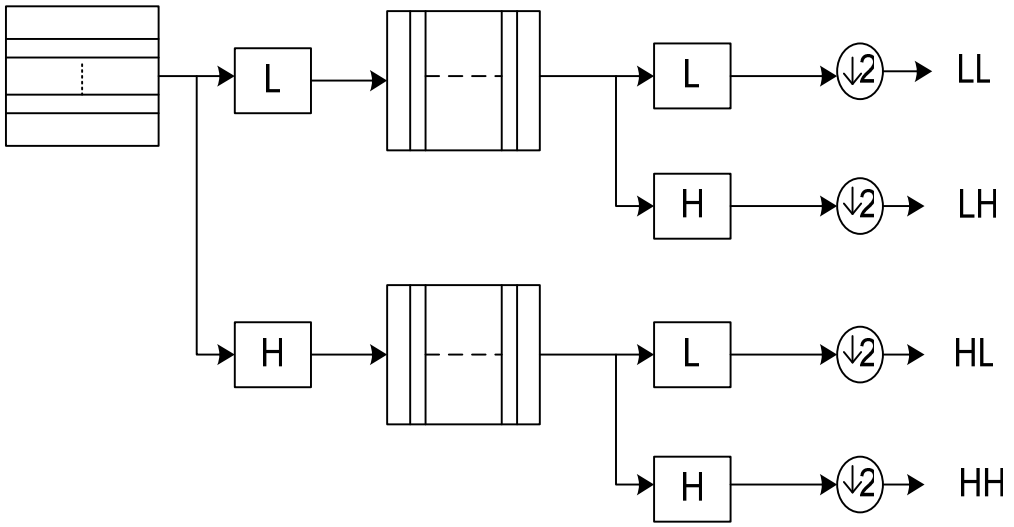
2. ระบบส่งข้อมูลภาพ

2.1 การเข้าและถอดรหัสข้อมูลภาพ

การเข้าและถอดรหัสข้อมูลภาพแบบ MPEG-4 เป็นวิธีการที่นิยมใช้ในการบีบอัดข้อมูลภาพ เนื่องจากมีประสิทธิภาพในการบีบอัดข้อมูลภาพสูง มีวิธีการเริ่มจากการแยกข้อมูลออกเป็น 2 ส่วน คือ กลุ่มข้อมูล LFS และ กลุ่มข้อมูล HFS กลุ่มข้อมูล LFS มีหน้าที่เป็นตัวแทนโครงสร้างหลักของข้อมูล กลุ่มข้อมูล HFS มีหน้าที่เป็นตัวแทนรายละเอียดของภาพ ดังนั้นการป้องกันข้อมูลในส่วนนี้ทำโดยการเข้าและถอดรหัสช่องสัญญาณแบบเทลลิส เพื่อให้สามารถส่งผ่านช่องสัญญาณได้ และมีความผิดพลาดน้อยที่สุด ซึ่งอาจทำแยกและใช้วิธีการต่างกัน

เทคนิค EZW เป็นอัลกอริทึมที่ใช้ในการบีบอัดข้อมูลภาพ โดยแบ่งข้อมูลภาพออกเป็นส่วนๆ ข้อมูลภาพที่ทำการแปลงแล้ว จะแทนด้วยสัมประสิทธิ์เวฟเลต เทคนิคนี้มีประสิทธิภาพในการนำข้อมูลภาพกลับ และมีความสามารถในการแก้ปัญหาหลายรูปแบบ ทำให้ได้รับความนิยมในด้านการสื่อสารข้อมูล และ มัลติมีเดีย ในการแปลงสัมประสิทธิ์เวฟเลตของข้อมูลภาพดิจิตอล โดยวิธีการแปลงแบบ Discrete Wavelet Transform หรือ DWT เริ่มจากแบ่งข้อมูลภาพในช่วงความถี่ย่านที่ต่างกัน จากนั้นใช้เทคนิค Iterated Filtering และ Downsampling เป็นตัวกรองเพื่อแยกความถี่ของข้อมูลภาพในย่านความถี่ที่ต้องการ โดยใช้สเกลในการวิเคราะห์สัมประสิทธิ์เวฟเลตของกระบวนการ Downsampling ซึ่งมีการกำหนดสเกล 2 แบบ คือ Coarse และ Fine ใช้ในการเรียกสเกลที่มีขนาดใหญ่ และ สเกลที่มีขนาดเล็ก ตามลำดับ

กระบวนการแบ่งแยกข้อมูลภาพแบบ 2 มิติ ประกอบด้วย 2 ขั้นตอน เริ่มจากการแบ่งข้อมูลภาพในแนวนอนหรือ แถว ผ่านตัวกรอง จากนั้นแบ่งข้อมูลภาพในแนวตั้งฉากหรือ คอลัมน์ ผ่านตัวกรอง โดยตัวกรองมีการกรองความถี่ 2 แบบ คือ ตัวกรองความถี่ต่ำหรือ Low-pass Filter และ ตัวกรองความถี่สูงหรือ High-pass Filter สามารถแสดงขั้นตอนการแบ่งข้อมูลภาพดังภาพที่ 2

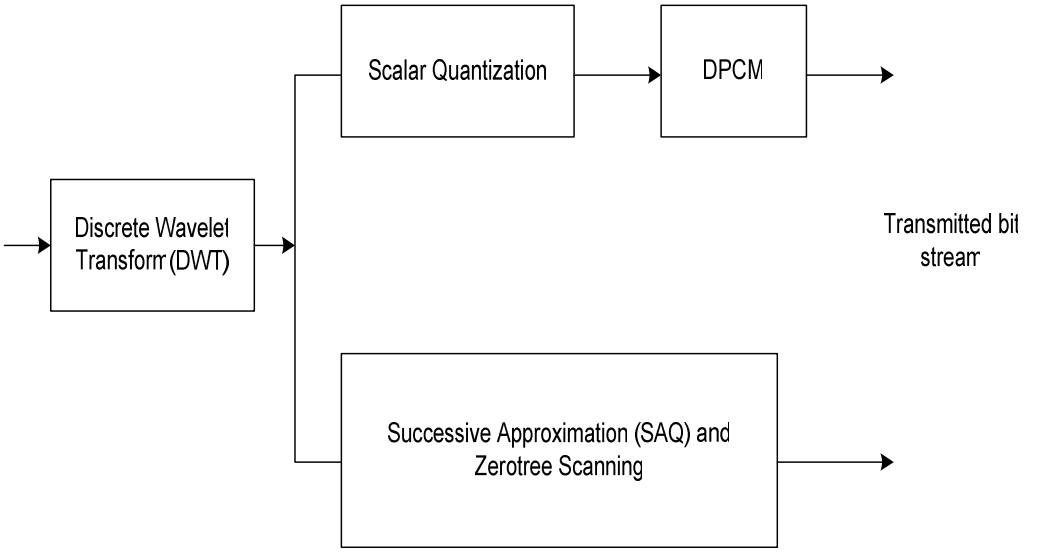


ภาพที่ 2 ขั้นตอนการแบ่งข้อมูลภาพ

รหัส MPEG-4 EZW เป็นอัลกอริทึมที่มีต้นแบบมาจาก EZW และ SPIHT โดยทำการเข้ารหัสจากการแปลงภาพด้วย DWT โดยใช้ Daubechies (9,15) Tap Biorthogonal Filters ที่มีการแบ่งข้อมูลออกเป็น 5 ระดับ เพื่อแยกข้อมูลภาพออกเป็นสัมประสิทธิ์ LFS และ HFS เป็นผลทำให้การป้องกันความผิดพลาดของสัมประสิทธิ์มีประสิทธิภาพต่างกัน

2.1.1 การเข้ารหัสข้อมูลภาพ

การเข้ารหัสข้อมูลภาพ เพื่อแปลงข้อมูลภาพสัมประสิทธิ์ LFS เริ่มจากกระบวนการ Scalar Quantization จากนั้นส่งค่าสัมประสิทธิ์ที่ได้ไปยังกระบวนการ Differential Pulse Code Modulation หรือ DPCM และ การเข้ารหัสข้อมูลภาพสัมประสิทธิ์ HFS จะใช้กระบวนการ Successive Approximation Quantization และ Zerotree Scanning สามารถแสดงขั้นตอนการเข้ารหัสของทั้ง 2 กระบวนการดังภาพที่ 3

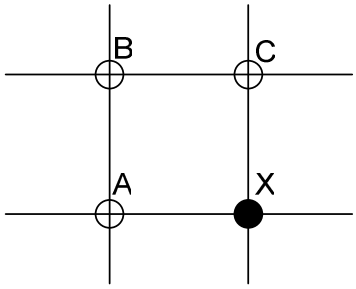


ภาพที่ 3 บล็อกไดอะแกรมการเข้ารหัส EZW

การเข้ารหัสข้อมูลภาพสัมประสิทธิ์ LFS เริ่มจากการนำสัมประสิทธิ์ LFS ผ่านกระบวนการควอนไตซ์โดย Scalar Quantization จากนั้นส่งไปยังกระบวนการ DPCM เพื่อทำนายค่าสัมประสิทธิ์ของการควอนไตซ์ (X) จากสัมประสิทธิ์ใกล้เคียงอีก 3 ตัว ซึ่งแทนด้วยสัญลักษณ์ A, B และ C ดังภาพที่ 4 โดยการหาค่าสัมประสิทธิ์ Residual หาได้จากผลต่างระหว่างค่าที่ทำนาย $\hat{X}$ กับ X โดยกำหนดเงื่อนไขดังนี้

$$\begin{aligned} &\text{If} && \left(|A-B| < |A-C|\right) \\ &\text{then} && \hat{X} = C \\ &\text{else} && \hat{X} = A \\ &\text{The residual} && = X - \hat{X} \end{aligned}$$

โดยทั่วไปการเข้ารหัส MPEG-4 จะใช้ค่าน้อยที่สุดเป็นการตั้งค่าออฟเซตของสัมประสิทธิ์ Residual ซึ่งมีค่าเป็นบวก จากนั้นผ่านกระบวนการเข้ารหัส Adaptive Arithmetic ทำการเลื่อนสัมประสิทธิ์ Residual จากค่าออฟเซตที่ใช้ในการส่ง โดยเลือกจากความสำคัญของสัมประสิทธิ์

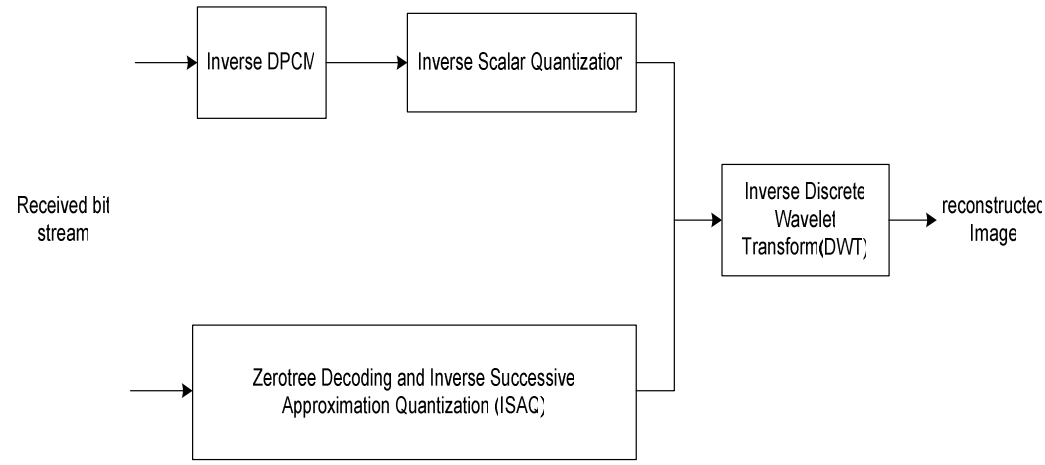


ภาพที่ 4 ตำแหน่งการควอนไตซ์ของสัมประสิทธิ์ X และใกล้เคียง

การเข้ารหัสข้อมูลภาพสัมประสิทธิ์ HFS แบ่งออกเป็น 2 แบบ คือ Single Quantization และ Multi-Quantization ในงานวิจัยนี้ใช้การเข้ารหัสภาพแบบ Single Quantization โดยใช้ Multi-level Quantizer และทำการควอนไตซ์เพียงครั้งเดียว โดยลำดับข้อมูลที่ได้จะไม่มีคุณสมบัติของ Embedded Code เนื่องจากบิตที่มีความสำคัญไม่ได้ทำการส่งเป็นบิตแรก โครงสร้างทรีเริ่มต้นจากสเกล Coarsest ที่มีสัมประสิทธิ์ในย่านความถี่ HL_3, LH_3และ HH_3 ที่ SOT เดียวกัน โดยใช้บิตข้อมูล SOTs เพื่อทำการสแกนลำดับ ดังนั้นสัมประสิทธิ์จะถูกแบ่งเป็น 4 แบบ คือ Zerotree Root หรือ ZTR, Isolated Zero หรือ IZ, Valued Zerotree Root หรือ VZTR และ Value หรือ VAL ตามเงื่อนไขดังนี้ คือ ถ้าสัมประสิทธิ์มีค่าน้อยกว่าขีดแบ่ง สัมประสิทธิ์ที่ได้จะอยู่ในกลุ่ม ZTR ถ้าสัมประสิทธิ์ที่ได้อย่างน้อย 1 ตัว มีค่ามากกว่าขีดแบ่ง สัมประสิทธิ์ที่ได้จะอยู่ในกลุ่ม IZ ถ้าสัมประสิทธิ์ที่ได้ไม่ตรงตามเงื่อนไขข้างต้น สัมประสิทธิ์ที่ได้จะอยู่ในกลุ่ม VZTR หรือ VAL โดยที่สัมประสิทธิ์ที่ได้มีค่ามากกว่าขีดแบ่งอยู่ในกลุ่ม VZTR และถ้าสัมประสิทธิ์ที่ได้อย่างน้อย 1 ตัว มีค่ามากกว่าขีดแบ่งจะอยู่ในกลุ่ม VAL อย่างไรก็ตามสัมประสิทธิ์ที่อยู่ในรูปแบบของ VAL และ IZ ต้องมีความสอดคล้องกับรูปแบบของทรี เมื่อทำกระบวนการดังกล่าวเสร็จสิ้นแล้ว จะวนกลับมาเพื่อเริ่มทำกระบวนของทรีถัดไป

การเข้ารหัสข้อมูลภาพสัมประสิทธิ์ HFS โดยใช้ Multi-Quantization จะนำค่าสัมประสิทธิ์เวฟเลตเปรียบเทียบกับขีดแบ่ง จากนั้นทำการกำหนดรูปแบบของสัญลักษณ์ 4 แบบ คือ ZTR, IZ, VZTR และ VAL โดยใช้หลักการในการกำหนดรูปแบบของสัญลักษณ์แบบเดียวกับการเข้ารหัสข้อมูลภาพแบบ Single Quantization และทำการปรับค่าขีดแบ่งโดยกำหนดให้เป็นครึ่งหนึ่งของ ขีดแบ่ง Current เพื่อใช้ในการสแกนรอบถัดไป

2.1.2 การถอดรหัสข้อมูลภาพ



ภาพที่ 5 บล็อกไดอะแกรมการถอดรหัส EZW

การถอดรหัสข้อมูลภาพสัมประสิทธิ์ LFS จะเป็นการเลื่อนกลับโดยใช้ระดับของการ Quantization โดยผ่านกระบวนการ Inverse DPCM ซึ่งมีการกำหนดเงื่อนดังนี้

$$\text{If} \quad \left(|A-B| < |A-C|\right)$$

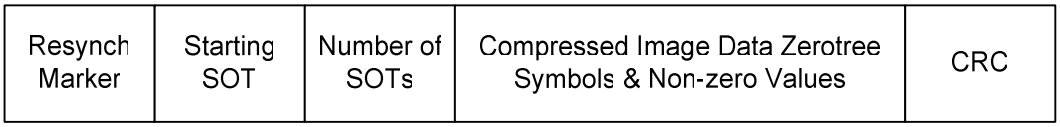
$$\text{then} \quad \hat{X} = C$$

$$\text{else} \quad \hat{X} = A$$

$$X = \hat{X} + \text{The residual}$$

จากนั้นทำกระบวนการ Inverse Quantization เพื่อนำสัมประสิทธิ์ LFS กลับคืนมา

การถอดรหัสข้อมูลภาพสัมประสิทธิ์ HFS เริ่มจากนำขีดแบ่งเริ่มต้น และ สัญลักษณ์ซีโร่ทรีเข้ามาในกระบวนการ จากนั้นเซตค่าสัมประสิทธิ์ทั้งหมดให้เป็นศูนย์ ถ้าสัญลักษณ์ที่รับมานั้นเป็น VZTR และ VAL จะใช้ค่าขีดแบ่งเพิ่มค่าของสัมประสิทธิ์นั้น แต่ถ้าสัมประสิทธิ์ที่รับมาเป็น ZTR และ IZ ค่าของสัมประสิทธิ์นั้นจะมีค่าเท่าเดิม (โดยที่ ZTR และ VZTR จะไม่มีการถอดรหัส) และต้องปรับค่าของขีดแบ่งก่อนที่จะทำการถอดรหัสครั้งต่อไป โดยค่าที่ใช้กำหนดค่าตามเดิมของการเข้ารหัส จากนั้นใช้กระบวนการ Inverse DWT เพื่อแปลงสัมประสิทธิ์เวฟเลตกลับเป็นรูปภาพตามเดิม



ภาพที่ 6 โครงสร้างพื้นฐานของกลุ่มข้อมูลภาพสัมประสิทธิ์ HFS

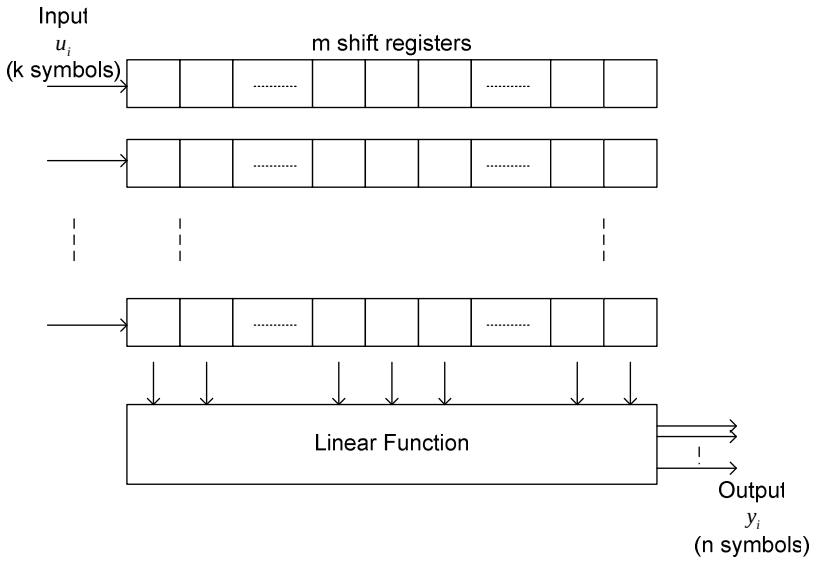
ภาพที่ 6 แสดงโครงสร้างพื้นฐานของกลุ่มข้อมูลภาพสัมประสิทธิ์ HFS สำหรับ Single Quantization โดย 20 บิตแรกแทน Header ของการ Resynchronization Maker ที่ใช้เป็นตัวบอกจุดเริ่มต้นของแต่ละกลุ่มข้อมูล 10 บิตถัดไปเป็นบิต SOT เพื่อใช้บอกจุดเริ่มต้นของแต่ละทรี 8 บิตถัดไปเป็นบิตของ SOTs ใช้บอกจำนวนของทรีที่อยู่ในแต่ละกลุ่มข้อมูล และ 4 บิตสุดท้าย คือ Cyclic Redundancy Check ใช้ในการตรวจเช็คความถูกต้องของการส่งกลุ่มข้อมูล การใช้ Multi-Quantization จะใช้บิตของ SOTs และ CRC จำนวน 10 บิต และ 4 บิต ตามลำดับ สำหรับข้อมูลที่มีการถอดรหัสแบบ MAP จะมีการแยกส่วนของซีโร่ทรี และ Nonzero Value ออกจากกัน

2.2 การเข้าและถอดรหัสช่องสัญญาณ

การเข้ารหัสช่องสัญญาณ เป็นการเพิ่มบิตพิเศษหรือ Redundancy เข้าไปกับข้อมูล เพื่อป้องกันบิตของข้อมูล วิธีการเข้ารหัสแบบเทลลิส Ugrelidze and Shavgulidze (1994) โดยการออกแบบการรวมกันของการเข้ารหัสคอนโวลูชั่นกับการมอดูเลตแบบ M-ary Phase Shift Keying หรือ MPSK ซึ่งการออกแบบร่วมกันจะให้ประสิทธิภาพดีกว่าเมื่อเทียบกับการออกแบบแยกกัน จากผลงานนี้จึงเกิดงานวิจัยที่เกี่ยวข้องกับการออกแบบการเข้ารหัสช่องสัญญาณแบบเทลลิสขึ้นอย่างกว้างขวาง เช่น Yang and Taylor (n.d.) ออกแบบการเข้ารหัสคอนโวลูชั่น และ การมอดูเลตแบบ Continuous Phase Frequency-Shift Keying หรือ CPFSK และ Rimoldi (1988) ออกแบบโดยใช้การเข้ารหัสริงคอนโวลูชั่นแทนการใช้การเข้ารหัสคอนลูชั่นแบบฐานสอง เนื่องจากการใช้การเข้ารหัสริงคอนโวลูชั่นสามารถนำมาใช้กับการมอดูเลตแบบเฟสต่อเนื่องได้อย่างเป็นธรรมชาติ ซึ่งแนวคิดนี้ได้รับการพัฒนามาจากงานวิจัย Mahapakulchai and Van Dyck (2000) โดยศึกษาการแยกโครงสร้างภายในของการมอดูเลตแบบเฟสต่อเนื่อง เพื่อสามารถใช้การเข้ารหัสริงคอนโวลูชั่น และการมอดูเลตแบบที่ไม่มีหน่วยความจำ

2.2.1 การเข้ารหัสช่องสัญญาณ

การเข้ารหัสคอนโวลูชั่น (n,k,m) ภายใต้ Finite Field F โดยกำหนดให้ k คือ อินพุต n คือ เอาต์พุต และ m คือ จำนวนสถานะของ Shift Register ซึ่งแสดงบล็อกไดอะแกรมของการเข้ารหัสคอนโวลูชั่น ดังภาพที่ 7



ภาพที่ 7 บล็อกไดอะแกรมการเข้ารหัสคอนโวลูชั่น

รหัสคอนโวลูชั่นสามารถแทนด้วย Generator Matrix โดยมีอัตราส่วนการเข้ารหัส k/n และมีความยาวของรหัส $(m+1)k$ สามารถแสดงค่า อินพุต และ เอาต์พุต ที่กำหนดเป็น Semi-Finite ในเทอมของ Unit Time Delay (D) ดังสมการ

$$u(D) = u_0 + u_1 D + u_2 D^2 + \ldots \qquad (1)$$

และ

$$y(D) = y_0 + y_1 D + y_2 D^2 + \ldots \qquad (2)$$

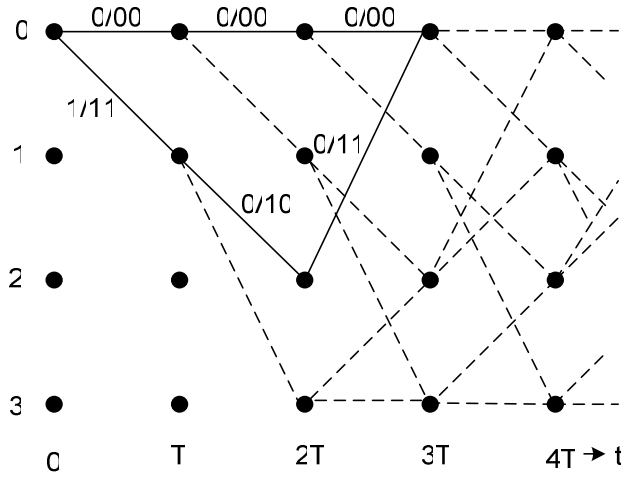
โดยที่ $u_i = (u_{0,i}, u_{1,i}, \ldots, u_{k-1,i})$ และ $y = (y_{0,i}, y_{1,i}, \ldots, y_{n-1,i})$ คือ อินพุต และ เอาต์พุต ตามลำดับ Generator Matrix $G(D)$ ของรหัสคอนโวลูชั่นสามารถกำหนดให้เป็น rank-k, ซึ่งเป็นเมตริกซ์พหุ-

นามขนาด $k \times n$ หรือเป็นฟังก์ชันตรรกยะหรือ Rational ในเทอมของ D ดังนั้นความสัมพันธ์ของอินพุต และ เอาต์พุต คือ $y(D) = u(D)G(D)$ ข้อมูลที่ได้จะเป็นรหัสคำหรือ Codeword ที่สร้างจาก $G(D)$ ซึ่งมีค่าสอดคล้องกับค่าอินพุต

ระยะทางของรหัสคอนโวลูชั่นจะกำหนดให้เป็นเมตริกซ์ของระยะห่างระหว่างรหัสคำแต่ละคู่ เช่น เมื่อพิจารณาจำนวนบิตที่แตกต่างกันระหว่างรหัสคำ 00110 และ 11000 คือ ระยะห่างระหว่างรหัสคำมีค่าเท่ากับ 4 เรียกว่า ระยะทางแฮมมิ่ง หรือ $d_{\min}$ ในรหัสบล็อกการหาระยะทางสั้นที่สุดเป็นการกำหนดให้รหัสคำที่ต่อเนื่องกันเป็นเมตริกซ์ที่มีค่าน้อยๆ ถ้าเป็นรหัสเชิงเส้นค่า $d_{\min}$ สามารถหาค่าได้จากรหัสคำอ้างอิง โดยกำหนดรหัสคำที่เป็นศูนย์ทั้งหมด เพื่อใช้ในการคำนวณหาประสิทธิภาพในการแก้ไขความผิดพลาดของข้อมูล การควบคุมความผิดพลาดของข้อมูลจะขึ้นอยู่กับวิธีการถอดรหัส การพิจารณาโดยใช้ $d_{\min}$ เป็นเกณฑ์ที่ใช้กับการถอดรหัสที่มีความยาวของรหัสช่วงเวลาหนึ่งเหมือนกับรหัสบล็อก แต่ถ้าการถอดรหัสใช้เทคนิค Maximum Likelihood Sequence Estimator (MLSE) ค่าระยะทางสั้นที่สุดระหว่างรหัสคำที่เกิดขึ้นต่อเนื่องกันจะใช้ d_{free} ในการพิจารณา ดังนั้นจำนวนของรหัสคำที่เป็นไปได้ทั้งหมดของรหัสคอนโวลูชั่นจะขึ้นอยู่กับความยาวของช่วงเวลาที่ทำการสังเกต พิจารณาจากการเลือกรหัสคำอ้างอิง โดยใช้รหัสคำที่เป็นศูนย์ทั้งหมด จากภาพที่ 8 แสดงไดอะแกรมเทลลิสของระยะ d_{free} โดยสร้างจากสถานะไดอะแกรม ซึ่งเป็นรหัสเฉพาะของข้อมูลนี้จะเกิดขึ้นเมื่อช่วงเวลาที่สังเกต $t \geq 3T$ และมีระยะทางเท่ากับ 5 เป็นเส้นทางสั้นที่สุดที่ลู่ออก และกลับมายังเส้นทางของรหัสคำอ้างอิง

การเท่ากันของการเข้ารหัสคอนโวลูชั่นเป็นการเข้ารหัสของข้อมูล 2 รูปแบบ โดยมีค่ารหัสคำเหมือนกัน เรียกว่าเป็นการเข้ารหัสชนิดเดียวกัน ในการเข้ารหัสบล็อกที่เท่ากันนั้น ต้องมีการสร้างโดเมนของรหัสคำเดียวกัน และ การจับคู่สถานะต้องเป็นตัวเดียวกัน โดยกำหนดให้การเข้ารหัสคอนโวลูชั่นมีอัตราการเข้ารหัส k/n โดยมีเมตริกซ์ที่เท่ากัน คือ $G(D)$ และ $G'(D)$ ซึ่งมีความสัมพันธ์ดังสมการ $G(D) = T(D)G'(D)$ โดยเมตริกซ์ $T(D)$ มีขนาด $b \times b$

รหัสริงคอนโวลูชั่นเป็นการใช้รหัสเชิงเส้นบนรหัสริงของจำนวนเต็มที่ modulo M ซึ่ง $Z_M = \{0,1,...,M-1\}$ เนื่องจากเป็นการรวมรหัสโดย M -ary Phase Modulation รหัสคอนโวลูชั่นที่มีอัตราการเข้ารหัส k/n บน Z_M มีโครงสร้างคล้ายกับรหัสไบนารี่คอนโวลูชั่น ยกเว้นสัมประสิทธิ์ที่เป็นสมาชิกของ Z_M



ภาพที่ 8 ไดอะแกรมเทลลิสของการเข้ารหัสคอนโวลูชั่น

2.2.2 การถอดรหัสช่องสัญญาณ

การถอดรหัสคอนโวลูชั่นแบ่งได้ 2 แบบ คือ การถอดรหัสพีชคณิต และ การถอดรหัสแบบความน่าจะเป็น โดยการถอดรหัสแบบความน่าจะเป็น หรือ การถอดรหัสแบบวีเทอร์บี เป็นวิธีที่ดีที่สุด เนื่องจากนำมาแก้ปัญหาการถอดรหัสแบบ ML และ MAP ของสถานะที่มีจำกัดแบบไม่ต่อเนื่องทางเวลาที่ใช้ในกระบวนการ Markov ซึ่งกระบวนการ Markov เป็นค่าความน่าจะเป็นของสถานะ x_k ที่เวลา k ใดๆ และกำหนดให้ทุกๆสถานะจากเวลา 0 ถึง $k-1$ ขึ้นอยู่กับสถานะที่ x_{k-1} ที่เวลา $k-1$ ดังสมการที่ (3)

$$P\left(x_k \left| x_0, x_1, ..., x_{k-1}\right.\right) = P\left(x_k \left| x_{k-1}\right.\right) \qquad (3)$$

การเข้ารหัสคอนโวลูชั่นสามารถแสดงกระบวนการ Markov ที่ไม่ต่อเนื่องทางเวลาและมีสถานะจำกัด โดยมีสถานะไดอะแกรมแสดงเส้นทางทุกๆเส้นทางที่มีการเชื่อมต่อระหว่างสถานะก่อนหน้า x_{k-1} ที่เวลา $k-1$ และ สถานะปัจจุบัน x_k ที่เวลา k ซึ่ง Transition Probability $P\left(x_k \left| x_{k-1}\right.\right)$ จะขึ้นอยู่กับ Transition Probability $P\left(u_k \left| u_{k-1}\right.\right)$ โดย u_k คือ บิตของข้อมูล การคำนวณหาเหตุการณ์ที่เกิดขึ้นของการถอดรหัสแบบ MAP จะทำการหาเหตุการณ์ที่เกิดขึ้นอย่างต่อเนื่องกันของสถานะเพื่อให้ $P(x|z)$ มีค่ามากที่สุด โดยที่ z คือเหตุการณ์ที่สังเกตได้โดยสัมพันธ์กับสถานะของเหตุการณ์ที่ x เป็นการหาค่าเหตุการณ์ที่เกิดร่วมกัน โดยที่ $P(x,z) = P(x|z)P(z) = P(z|x)P(x)$ ต้องมีค่ามากที่สุด ความน่าจะเป็นของเหตุการณ์ x ที่เกิดร่วมกันแสดงดังสมการที่ (4)

$$P(x) = P(x_0, x_1, ..., x_{K-1})$$

$$= P(x_{K-1}|x_{K-2}, ..., x_0)P(x_{K-2}|x_{K-3}, ..., x_0)\cdots P(x_1|x_0)P(x_0) \quad (4)$$

เนื่องจากคุณสมบัติของกระบวนการ Markov ในสมการที่ (4) สามารถลดรูปได้ดังสมการที่ (5)

$$P(x) = P(x_{K-1}|x_{K-2})P(x_{K-2}|x_{K-3})\cdots P(x_1|x_0)P(x_0) \quad (5)$$

การใช้กระบวนการ Markov และ การใช้คุณสมบัติของสัญญาณรบกวนแบบที่ไม่มีหน่วยความจำ สามารถแสดงดังสมการที่ (6)

$$\arg\max_X P(z|x)P(x) = \arg\max_X \prod_{k=1}^{K-1} P(z_k|x_k, x_{k-1})P(x_k|x_{k-1}) \quad (6)$$

ฟังก์ชัน Natural log (ln(.)) คือ ฟังก์ชันโมโนโทนิกหรือ Monotonic Function ซึ่งสามารถนำไปใช้กับสมการที่ (6) โดยไม่ทำให้ผลลัพธ์ทางขวาของสมการที่ (6)มีค่าเปลี่ยนแปลง จะได้เป็นสมการที่ (7)

$$\arg\max_X \sum_{k=1}^{K-1} \ln P(z_k|x_k, x_{k-1}) + \ln P(x_k|x_{k-1}) \quad (7)$$

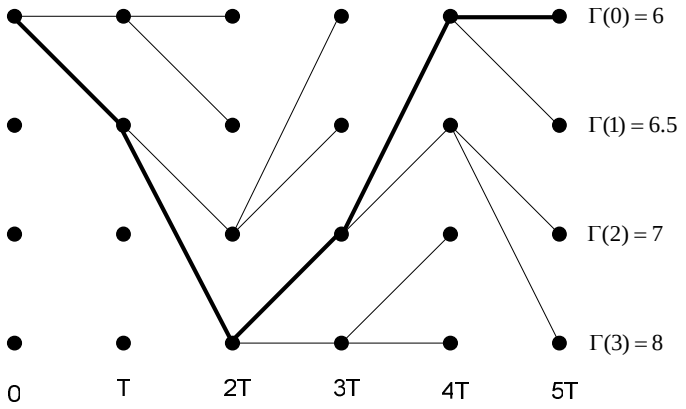
นอกจากนี้ถ้ามีการกำหนดให้แต่ละเมตริกซ์ของแต่ละเส้นทางเป็น

$$\lambda(x_k, x_{k-1}) \underline{\Delta} - \ln P(z_k|x_k, x_{k-1}) - \ln P(x_k|x_{k-1}) \quad (8)$$

ทำให้สมการที่ (7) สามารถเปลี่ยนรูปสมการอย่างง่ายดังสมการที่ (9)

$$\arg\min_X = \sum_{k=1}^{K-1} \lambda(x_k, x_{k-1}) \quad (9)$$

จากสมการที่ (9) จะเห็นได้ว่าอัลกอริทึมของการถอดรหัสวีเทอร์บีมีความเกี่ยวเนื่องในการหาค่าระยะทางที่สั้นที่สุดในไดอะแกรมเทลลิส



ภาพที่ 9 ตัวอย่างเส้นทางสำรวจช่วงเวลา T = 5 และ เส้นทางที่ตัดสินใจเลือก

ถ้าทำการตัดสินใจเลือกที่ช่วงเวลานี้ เส้นทางที่เลือกจะกลับไปยังสถานะที่ 0 ที่มีค่าน้อยที่สุด $(\Gamma(0)=0)$ ภาพที่ 9 แสดงการเลือกเส้นทางที่สั้นที่สุด เมื่อทำการถอดรหัสแล้วจะได้บิตข้อมูล คือ 1,1,0,0,0

การออกแบบรหัสคอนโวลูชั่น การตัดสินใจเลือกที่ดีต้องกำหนดเกณฑ์ในการตัดสิน โดยต้องเลือกรหัสที่มีความน่าจะเป็นของความผิดพลาดน้อยๆ บนช่วงกำลังของสัญญาณรบกวน เนื่องจากการกระทำของรหัสคอนโวลูชั่นที่กำลังของสัญญาณรบกวนต่ำๆ จะขึ้นอยู่กับอัลกอริทึมการถอดรหัส และ d_{free} ดังนั้นเกณฑ์การวัด คือ การเปรียบเทียบรหัสกับฐานหลักของ d_{free} ต่างๆ

ตารางที่ 1 แสดงการเข้ารหัสไบนารี่คอนโวลูชันนอล เมื่อมีความซับซ้อนของการเข้ารหัสคอนโวลูชั่นเพิ่มขึ้น เนื่องจากจำนวนของดีเลย์เพิ่มขึ้น ทำให้รหัสคอนโวลูชั่นในเทอมของ d_{free} มีค่าเพิ่มขึ้นด้วย รหัสคอนโวลูชั่นได้นำมาใช้กับการเข้ารหัสช่องสัญญาณ โดยใช้เทคนิคการมอดูเลตของการออกแบบรหัสคอนโวลูชั่น และบางครั้งจะรวมกับสัญญาณรูปคลื่นที่สร้างจากตัวกล้ำสัญญาณ ซึ่งทำให้การเก็บค่าของรหัสมีความสำคัญ และไม่ทำให้ประสิทธิภาพของแบนด์วิดท์ลดลง สัญญาณรูปคลื่นที่แทนด้วยรหัสคำจะให้รหัสคำที่ได้มีรูปแบบแตกต่างกัน ทำให้ระบบมีความยืดหยุ่นต่อสัญญาณรบกวน

ตารางที่ 1 การเข้ารหัสไบนารี่คอนโวลูชั่นที่มีอัตราการเข้ารหัส 1/2 และ ระยะแฮมมิ่ง

Number of delays	Best encoders	d_{free}
2	$\left[D^2+D+1,\ D^2+1\right]$	5
3	$\left[D^3+D^2+D+1,\ D^2+D+1\right]$	6
4	$\left[D^4+D^2+D+1,\ D^4+D^3+1\right]$	7
5	$\left[D^5+D^3+D^2+D+1,\ D^5+D^4+D^2+1\right]$	8
6	$\left[D^6+D^3+D^2+D+1,\ D^6+D^5+D^3+D^2+1\right]$	10
7	$\left[D^7+D^4+D^3+D^2+D+1,\ D^7+D^6+D^5+D^2+1\right]$	10
8	$\left[D^8+D^7+D^5+D^3+D^2+D+1,\ D^8+D^4+D^3+D^2+1\right]$	12

2.3 การมอดูเลตช่องสัญญาณ

กระบวนการ CPM เป็นวิธีการมอดูเลตข้อมูลดิจิตอลแบบไม่เชิงเส้นโดยการใช้หน่วยความจำ เนื่องจากเฟสของสัญญาณ CPM ทำให้สัญญาณที่ได้มีความต่อเนื่องกัน เพื่อทำการส่งไปยังช่องสัญญาณ ดังนั้น CPM จึงเป็นวิธีการที่เหมาะสำหรับช่องสัญญาณที่มีอยู่จำกัด และมีการเพิ่มประสิทธิภาพในการใช้แบนด์วิดท์โดยแอมพลิจูดเท่าเดิม ทำให้มีความต้านทานต่อสัญญาณรบกวนในระบบสื่อสารภายในช่องสัญญาณแบบไม่เป็นเชิงเส้น ตัวอย่างของการนำ CPM ไปใช้งานสามารถพบได้ในระบบสื่อสารดาวเทียม

-ในการส่งสัญญาณข่ายการเชื่อมโยงลงจะใช้กำลังในการส่งสูง ทำให้เกิดการลดทอนของสัญญาณมากขึ้นด้วย และ

-ในย่านความถี่ UHF (300-3000 MHz) และ VHF (30-300 MHz) ที่นำมาใช้งานส่วนมากจะถูกทำลายโดย Fading
การทำการมอดูเลตสามารถแก้ปัญหานี้ได้ โดยการสร้างสัญญาณ Envelope โดยทั่วไประบบ CPM จะส่งสัญญาณในรูปแบบของ

$$sm(t,\alpha)=\sqrt{\frac{2E}{T}}\cos\left(2\pi f_0 t+\varphi(t,\underline{\alpha})+\varphi_0\right) \qquad (10)$$

โดยที่ $\underline{\alpha}$ คือ เหตุการณ์ของสัญลักษณ์แบบ M -ary ที่ต่างกัน และ $\alpha_i \in \{\pm 1, \pm 3, ..., \pm(M-1)\}$ โดยกำหนดเฟสที่ใช้ในการส่ง ดังสมการที่ (11)

$$\varphi(t,\underline{\alpha})=2\pi h\sum_{i=-\infty}^{\infty}\alpha_i f(t-iT) \qquad -\infty<t<\infty \qquad (11)$$

โดยที่ h คือ Modulation Index และ $f(t)$ คือ สัญญาณแถบความถี่ฐานในเทอมของเฟส โดยมีผลตอบสนองในเทอมของความถี่ $g(t)$ โดย $f(t)=\int_{-\infty}^{t} g(\tau)d\tau$, -∞<t<∞ วิธีการ CPM จะแสดงว่า $g(t)$ ต้องให้ผลตอบสนองที่ดีที่ $g(t)=0,\ t>LT$ และ $g(t)\neq 0,\ 0<t<LT$ โดยสามารถแสดงเฟสที่ใช้ในการส่ง ดังสมการที่ (12)

$$\varphi(t,\underline{\alpha})=\pi h\sum_{i=-\infty}^{n-L}\alpha_i+2\pi h\sum_{i=n-L+1}^{n}\alpha_i f(t-iT),\quad nT\le t\le(n+1)T \qquad (12)$$

การกำหนดค่า h, M และ $g(t)$ หรือ $f(t)$ โดยส่วนใหญ่จะใช้ $g(t)$ ในรูปแบบของ Rectangular (L-REC)

$$g(t)=\begin{cases}\dfrac{1}{2LT}, & 0\le t\le LT\\ 0, & otherwise\end{cases} \qquad (13)$$

หรือฟังก์ชัน Rised Cosine (L-RC)

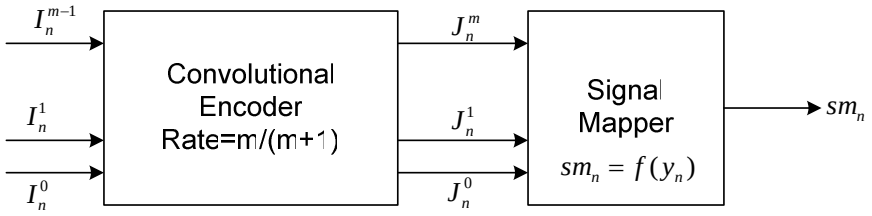
$$g(t)=\begin{cases}\dfrac{1}{2LT}\left[1-\cos\left(\dfrac{2\pi t}{LT}\right)\right], & 0\le t\le LT\\ 0, & otherwise\end{cases} \qquad (14)$$

การมอดูเลตจะเป็นแบบ Full Response CPM เมื่อ $L=1$ และเป็นแบบ Partial Response CPM เมื่อ $L>1$ ส่วนวิธีการ CPFSK จะเป็นกรณีพิเศษของ การมอดูเลตแบบ Full Response โดยผลตอบสนองทางเฟสของสัญญาณแถบความถี่ฐานจะเป็นแบบเชิงเส้น หรือ $g(t)=1-REC$ การใช้

แบนด์วิดท์ขึ้นอยู่กับการเลือก h, $g(t)$ และ จำนวนของสัญญาณ M การเลือกใช้ h ที่มีค่ามากๆ จะทำให้ความต้องการของแบนด์วิดท์มากด้วย และการเลือกใช้ Raised Cosine Frequency จะใช้แบนด์วิดท์น้อยกว่า Rectangular Frequency Pulse นอกจากนี้การมอดูเลตแบบ Partial Response จะใช้แบนด์วิดท์น้อยกว่า Full Response

2.3.1 การมอดูเลตรหัสเทลลิสของกระบวนการ CPM

TCM เป็นเทคนิคที่มีการรวมการเข้ารหัสและมอดูเลตไว้ด้วยกัน เพื่อใช้ในการส่งข้อมูลดิจิตอล เนื่องจากเป็นรหัสที่ไม่มีการเข้ารหัสแบบมอดูเลชั่นหลายระดับหรือ Multilevel Modulation โดยไม่ทำให้ประสิทธิภาพในการใช้แบนด์วิดท์ลดลง วิธี TCM จึงนำมาใช้กับช่องสัญญาณที่มีอยู่อย่างจำกัด โดยทำการสร้างเหตุการณ์ที่เกิดขึ้นอย่างต่อเนื่องของรหัสสัญญาณเพื่อรวมการเข้ารหัส Finite-State และ การเลือกสัญญาณมอดูเลต หลักการของการออกแบบวิธี TCM คือ การออกแบบกระบวนการเข้ารหัส และ กระบวนการมอดูเลตร่วมกัน โดยการขยายขีดสูงสุดของระยะทางสั้นที่สุด ที่เรียกว่า “free distance” ของรหัสสัญญาณที่เกิดขึ้นอย่างต่อเนื่องกัน ทำให้การสร้างรหัสสัญญาณ ที่มี free distance ระหว่างสัญญาณอัตราข่าวสาร และ ค่าของแบนด์วิดท์เดียวกันมีค่ามากกว่าที่ระยะเวลาสั้นสุด จากภาพที่ 10 เป็นบล็อกไดอะแกรมของวิธี TCM ข่าวสารที่เกิดขึ้นต่อเนื่องกันจะกำหนดให้เป็น $\underline{I} = (I_0, I_1, ..., I_n, ...)$ โดยที่ $I_n = \left(I_n^0, I_n^1, ..., I_n^{m-1}\right)$ การเข้ารหัสคอนโวลูชั่นที่มีอัตราการเข้ารหัส $\frac{m}{m+1}$ เพื่อสร้างลำดับเหตุการณ์ $\underline{J} = (J_0, J_1, ..., J_n, ...)$ โดยที่ $J_n = (J_n^0, J_n^1, ..., J_n^m)$ จากนั้นจะนำรหัสที่ได้ไปจับคู่กับสัญญาณ $\underline{sm} = (sm_0, sm_1, ..., sm_n, ...)$



ภาพที่ 10 บล็อกไดอะแกรมกระบวนการ TCM

2.3.2 รหัสโพลิโนเมียลริงคอนโวลูชั่นเทลลิส

ก่อนทำกระบวนการเข้าและถอดรหัสสัญญาณแบบเทลลิส TCM จะเปลี่ยนข้อมูลจาก 1 บิตเป็นสัญลักษณ์ โดยที่สัญลักษณ์ 1 สัญลักษณ์เท่ากับ 2 บิต เพื่อสามารถให้ผ่านการเข้ารหัสแบบริงได้ การเข้ารหัสแบบริง มีผลดี คือ สามารถออกแบบให้ร่วมกับการมอดูเลตแบบที่เฟสของสัญญาณมีความต่อเนื่อง เช่น การมอดูเลตแบบ CPM การเข้ารหัสช่องสัญญาณแบบเทลลิสประกอบด้วย 2 ส่วน คือ การเข้ารหัสช่องสัญญาณแบบริงคอนโวลูชั่น และ การมอดูเลตแบบที่เฟสของสัญญาณมีความต่อเนื่อง เนื่องจากเป็นการมอดูเลตที่เหมาะสมกับช่องสัญญาณไร้สาย ซึ่งมีช่วงกว้างแถบความถี่ที่ต่ำ นอกจากนี้การเลือกใช้การเข้ารหัสช่องสัญญาณแบบริงคอนโวลูชั่นสามารถรวมกับการมอดูเลตได้อย่างเป็นธรรมชาติ การถอดรหัสช่องสัญญาณแบบเทลลิสเลือกใช้การถอดรหัสแบบวีเทอร์บี เป็นเทคนิคที่สามารถตรวจจับบิตที่ผิดพลาดได้ดีที่สุด นอกจากนี้การใช้คำนวณค่าขนาดในแต่ละเส้นทางบนเทลลิสจะใช้ทฤษฎีการตรวจจับสัญญาณแบบ ML และ MAP

สำหรับการนำไปใช้ในช่องสัญญาณไร้สายนั้น ต้องการกระบวนการมอดูเลตที่มีการใช้ช่วงความถี่อย่างมีประสิทธิภาพ ซึ่งจะใช้การมอดูเลตแบบที่เฟสของสัญญาณมีความต่อเนื่อง โดยมีดัชนีการมอดูเลต 1/4 ซึ่งสัญญาณที่ได้นั้นมีการใช้ช่วงความถี่เท่ากับกระบวนการ MSK โดยกระบวนการมอดูเลตแบบที่เฟสของสัญญาณมีความต่อเนื่องจะแยกออกเป็น CPE และ MM ซึ่งทั้งหมดจะรวมกันอยู่ภายในกระบวนการเข้ารหัสช่องสัญญาณแบบริงคอนโวลูชันนอลที่อัตรา = 1/2

2.4 ช่องสัญญาณ

ช่องสัญญาณแบ่งออกเป็น ช่องสัญญาณแบบ Additive White Gaussian Noise หรือ AWGN และ ช่องสัญญาณที่มีการเลือนหายของสัญญาณ ดังนี้

2.4.1 ช่องสัญญาณ Additive White Gaussian Noise หรือ AWGN

โดยทั่วไปสัญญาณรบกวนที่เกิดจากความร้อน สามารถอธิบายได้จากกระบวนการสุ่มแบบ Gaussian และมี Spectral Density ที่เกิดอย่างชัดเจนบนช่วงของความถี่ โดยสัญญาณรบกวนนั้นคล้ายกับแสงสีขาวหรือ White Noise สามารถแสดง Power Spectral Density ของสัญญาณรบกวนสีขาวได้ดังสมการที่ (15)

$$S_n(\omega) = \frac{N_0}{2} \ \ watts/Hz \tag{15}$$

และสามารถแสดง Autocorrelation ของสัญญาณรบกวนสีขาวได้ดังสมการที่ (16)

$$R_{nn}(\tau) = F^{-1}\left(S_n(\omega)\right) = \frac{N_0}{2}\delta(\tau) \tag{16}$$

โดยที่ $F^{-1}(\cdot)$ คือ การแปลง Inverse Fourier จากสมการที่ (16) ทำให้ทราบว่าการ Sample ของสัญญาณ 2 สัญญาณที่ต่างกันของ Zero Mean Gaussian White Noise จะไม่สัมพันธ์กัน และ เป็นอิสระต่อกัน ในระบบสื่อสารข้อมูลดิจิตอลนั้น ข้อมูลจะถูกส่งโดยการใช้รูปคลื่น M สัญญาณ ซึ่งสามารถเขียนแทนโดย $s_m(t),\ m = 1, 2, ..., M$ โดยที่แต่ละสัญญาณจะถูกส่งแทนด้วยสัญลักษณ์ในช่วงเวลา T และถูกรบกวนโดย AWGN ดังนั้นสัญญาณที่ได้รับสามารถแสดงได้ดังสมการที่ (17)

$$r(t) = s_m(t) + n_w(t),\ 0 \le t \le T \tag{17}$$

โดยที่ $n_w(t)$ แทนด้วยฟังก์ชันของ AWGN ซึ่งมี PSD ดังสมการที่ (15) และสามารถแสดงสมการของสัญญาณแถบความถี่ฐาน $s_m(t)$ ดังสมการที่ (18)

$$s_m(t) = \mathrm{Re}\left[s_{ml}(t) e^{j\omega_c t} \right] \tag{18}$$

โดยที่ $\mathrm{Re}[x]$ คือ จำนวนจริงของค่า x และ $s_{ml}(t)$ เป็นจำนวนเชิงซ้อนของ Envelope ของ $s_m(t)$ ดังนั้นสามารถแทนสัญญาณย่านความถี่ต่ำที่ได้รับ ดังสมการที่ (19)

$$r_l(t) = s_{ml}(t) + n_{wl}(t),\ 0 \le t \le T \tag{19}$$

โดยที่ $n_{wl}(t)$ คือ Envelope ที่เป็นจำนวนเชิงซ้อนของ $n_w(t)$

2.4.2 ช่องสัญญาณที่มีการเลือนหายของสัญญาณ

การสร้างการจำลองช่องสัญญาณแบบไร้สาย โดยศึกษาคุณลักษณะของช่องสัญญาณที่มีการเลือนหายของสัญญาณ การเลือนหายของสัญญาณมี 2 แบบ คือ การเลือนหายของสัญญาณในเชิงขนาดหรือ Flat Fading และการเลือนหายของสัญญาณในเชิงความถี่หรือ Selective Fading นอกจากนี้ การเลือนหายของสัญญาณในเชิงขนาดยังเกิดขึ้นใน 2 ลักษณะคือ แบบช้า และ แบบเร็ว ถ้าการเปลี่ยนแปลงทางขนาดของสัญญาณที่ทุกๆ M สัญญาณข้อมูล โดยที่ M = 2,3,… เรียกว่าเป็นการเลือนหายของสัญญาณในเชิงขนาดแบบช้า ในทางตรงกันข้ามถ้าการเปลี่ยนแปลงทางขนาดของสัญญาณทุกค่าๆ ของสัญญาณข้อมูล เรียกว่าเป็นการเลือนหายของสัญญาณในเชิงขนาดแบบเร็ว

ในส่วนของช่องสัญญาณไร้สายแบบเป็นการเลือนหายของสัญญาณในเชิงขนาดแบบช้า การจำลองช่องสัญญาณนี้จะสมมติให้ พลังงานของสัญญาณมีค่าน้อยมากเมื่อเปรียบเทียบกับพลังงานของเส้นทางการกระเจิงหรือ Scattered Path ข้อสมมตินี้ไม่ตรงกับคุณสมบัติของช่องสัญญาณไร้สายภายในอาคาร เนื่องจากโดยทั่วไปแล้ว ภายในอาคารพลังงานของสัญญาณไม่ได้มีค่าน้อยมาก ดังนั้นจึงใช้การจำลองช่องสัญญาณแบบนี้โดยใช้ทฤษฎีของตัวแปรสุ่ม ที่มีการกระจายแบบไรเชียล ซึ่งมีรายละเอียดดังนี้

ถ้ากำหนดให้สัญญาณแถบความถี่ฐานเป็น $s_1(t)$ จะได้ว่าที่สัญญาณภาครับหลังจากผ่านเครื่องกรองความถี่ต่ำแล้ว สามารถเขียนได้ดังสมการที่ (20)

$$r_l(t) = \left(1 + \beta e^{-j\phi}\right) s_l(t) + n_{wl}(t),\ 0 \le t \le T \qquad (20)$$

โดยที่ $n_{wl}(t)$ เป็นสัญญาณรบกวนที่มีกระบวนการสุ่มแบบ Gaussian ที่มีความแปรปรวน = $N_0/2$ เนื่องจากกำหนดให้ช่องสัญญาณเป็นแบบช้า ดังนั้นจะสมมติให้สามารถประมาณค่าของ ϕ ได้ถูกต้อง ด้วยเหตุนี้จึงสามารถกำหนดให้เป็นศูนย์ที่ทำการจำลองสัญญาณได้ ส่วนตัวแปร β กำหนดดังนี้

$$\beta = \sqrt{\left[\frac{f_i(t)}{\sqrt{2S}}\right]^2 + \left[\frac{f_q(t)}{\sqrt{2S}}\right]^2} \qquad (21)$$

โดยที่ $f_i(t)$ และ $f_q(t)$ คือ In-Phase และ Quadrature Phase ของเส้นทางการกระเจิง ซึ่งกำหนดให้เป็นตัวแปรสุ่มแบบ Gaussian ที่มีค่าเฉลี่ยเป็นศูนย์ ความแปรปรวน = σ^2 และ $S = \frac{E_s}{T}$ คือค่าพลังงานของสัญญาณ ถ้ากำหนดให้

$$\gamma_b = (1+\beta)^2 E_s / N_0 \tag{22}$$

จะได้ว่า The Probability Density Function ของ γ_b คือ

$$p(\gamma_b) = e^{-\gamma-(1+\gamma)\gamma_0/\bar{\gamma}_0} I_0\left(2\sqrt{\frac{\gamma\gamma_b(1+\gamma)}{\bar{\gamma}_b}}\right)$$

$$\bar{\gamma}_b = \frac{1+\gamma}{\gamma}\frac{E_s}{N_0} \tag{23}$$

$$\gamma = \frac{S}{\sigma^2}$$

โดยที่ $\gamma = \frac{S}{\sigma^2}$ เป็นอัตราส่วนระหว่างกำลังส่งในเส้นทางตรง และ กำลังส่งในเส้นทางการกระเจิงซึ่งก็คือการกระจายแบบไรเชียล

2.4.3 การถอดรหัสแบบ MAP บนช่องสัญญาณเป็นการเลือนหายของสัญญาณ

การสมมติให้เป็นเป็นเป็นการเลือนหายของสัญญาณในเชิงขนาดแบบช้าจะกำหนดให้การเลื่อนทางเฟสมีค่าเป็น 0 ดังนั้นค่าสัญญาณความถี่ต่ำที่ได้รับในสมการที่ (21) จะถูกลดรูปดังสมการที่ (24)

$$r_l(t) = (1+\beta)s_{ml}(t) + n_{wl}(t) \tag{24}$$

โครงสร้างการถอดรหัสแบบ MAP ของกระบวนการ CPFSK นั้น จะสังเกตจากการ Sample ของข้อมูล $Z_{k,i}$ ในช่วงเวลา $kT \le t \le (k+1)T$ สามารถแสดงได้ดังสมการ $z_{k,i} = (1+\beta)sm_{k,i} + n_{k,i}$ โดยที่ $i = 1, 2, ..., N$ ในช่องสัญญาณเป็นการเลือนหายของสัญญาณในเชิงขนาดแบบช้า

พารามิเตอร์ β คือ ค่าคงที่ระหว่างช่วงเวลาของแต่ละสัญญาณ ดังนั้นข้อมูลสามารถรวมกันได้ภายในเมตริกซ์ของการถอดรหัส

สำหรับ CSI ความน่าจะเป็นแบบมีเงื่อนไขสามารถแสดงได้ดังสมการที่ (25)

$$P\left(z_k \left| x_{2,k+1}, x_{2,k} \right.\right) = P\left(z_k \left| sm_k \right.\right) = \frac{1}{\left(\sqrt{\pi N_0}\right)^N} \exp\left[-\sum_{i=1}^{N} \frac{\left(z_{k,i} - \alpha sm_{k,i}\right)^2}{N_0} \right] \qquad (25)$$

โดยที่ $\alpha = (1+\beta)$ เมื่อนำสมการที่ (25) แทนในสมการที่ (8) จะได้การคำนวณค่าขนาดในแต่ละเส้นทางบนเทลลิส $\lambda(sm_k)$ คือ

$$\lambda\left(sm_k\right) = -\sum_{i=1}^{N} z_{k,i} sm_{k,i} - \frac{N_0}{2\alpha} \ln P\left(x_{2,k+1} \left| x_{2,k} \right.\right) \qquad (26)$$

ซึ่งถ้าไม่รวมพจน์สุดท้ายจะเป็นการถอดรหัสแบบ ML จากสมการที่ (26) และ คุณสมบัติของกระบวนการ Markov จะได้สมการการคำนวณค่าขนาดในแต่ละเส้นทางบนเทลลิส คือ

$$\lambda\left(sm_k\right) = -\sum_{i=1}^{N} z_{k,i} sm_{k,i} - \frac{N_0}{2\alpha} \ln P\left(u_k \left| u_{k-1} \right.\right) \qquad (27)$$

ในส่วนของการเข้ารหัสภายนอกจะใช้ความน่าจะเป็นแบบมีเงื่อนไขของ $P\left(z_k \left| S_{0,k+1}, S_{0,k} \right.\right)$ แทน $P\left(z_k \left| x_{2,k+1}, x_{2,k} \right.\right)$ ซึ่งทำให้มีผลกับขนาดของเป็นการเลือนหายของสัญญาณ และ จากความน่าจะเป็นแบบมีเงื่อนไขของ $P\left(z_k \left| S_{0,k+1}, S_0, k \right.\right)$ กับความสัมพันธ์ของสมการที่ (25) จะได้สมการการคำนวณค่าขนาดในแต่ละเส้นทางบนเทลลิส ดังนี้

$$\lambda\left(sm_k\right) = -\alpha_1 \sum z_{k,i} sm_{k,i} - \alpha_2 \sum z_{k,i} sm_{k,i} - \frac{N_0}{2} \ln P\left(S_{0,k+1} \left| S_{0,k} \right.\right) \qquad (28)$$

โดยที่ α_1 และ α_2 เป็นค่าขนาดของเป็นการเลือนหายของสัญญาณของช่วงเวลาของ 2 สัญญาณ

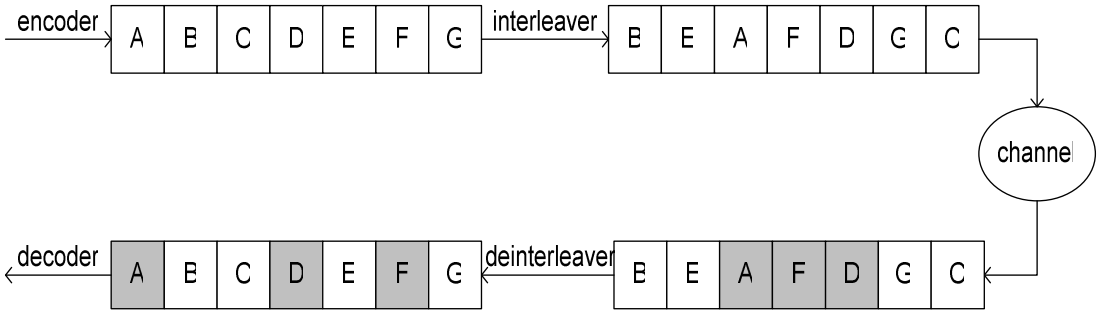
2.5 การถอดรหัสของการส่งภาพ MPEG-4 และช่องสัญญาณ Source-Controlled

การถอดรหัสช่องสัญญาณที่เป็น Source-Controlled สำหรับการส่งข้อมูลภาพโดยมีการเข้ารหัสริงคอนโวลูชั่น และการมอดูเลตสัญญาณ CPFSK เป็นการมอดูเลตแบบ Full Response CPM ที่มีความสัมพันธ์กับสัญลักษณ์ของสัมประสิทธิ์ HFS เพื่อต่อต้านกับสัญญาณรบกวน โดยกระบวนการมอดูเลตสัญญาณ CPFSK ประกอบด้วย 2 กระบวนการ คือ CPE และ MM สำหรับกระบวนการ CPE นั้นแยกออกเป็น 2 กระบวนการ คือ CE และ CPE โดยทุกกลุ่มข้อมูลจะมีการกำหนดค่าใหม่ทุกครั้ง หลักการของกระบวนการ MM จะเปลี่ยนรหัสสัญลักษณ์เป็นสัญญาณแถบความถี่ฐานประกอบด้วย สัญญาณ In-Phase และ สัญญาณ Quadrature Phase โดยไม่มีการมอดูเลตร่วมกับคลื่นพาห์ จากนั้นสัญญาณเบสแบนด์จะส่งผ่านไปยังช่องสัญญาณไร้สาย เพื่อรบกวนระบบโดยสัญญาณรบกวน ทำให้เกิดความผิดเพี้ยนของสัญญาณ จากนั้นจะส่งไปยังการถอดรหัสวีเทอร์บีและ ถอดรหัสภาพ เพื่อวิเคราะห์ประสิทธิภาพของระบบจากการหาค่า PSNR และ WER

3. การปรับปรุงระบบส่งข้อมูลภาพ

3.1 การสลับลำดับข้อมูลภาพ

การสลับลำดับข้อมูล เป็นเทคนิคที่ใช้ในการแก้ปัญหาข้อมูลที่มีความผิดพลาดแบบต่อเนื่อง Proakis (1995) โดยเปลี่ยนแปลงตำแหน่งข้อมูลเดิมให้แยกออกจากกัน ทำให้ความผิดพลาดของข้อมูลที่เกิดจากสัญญาณรบกวนที่มีลักษณะเป็นกลุ่มก้อนในช่วงเวลาสั้นๆมีค่าลดลง ดังนั้นข้อมูลจึงไม่มีความสัมพันธ์ซึ่งกันและกัน ตัวถอดรหัสสามารถแก้ไขความผิดพลาดได้ ภาพที่ 11 แสดงการกระจายตำแหน่งของสัญลักษณ์ เริ่มจากการสลับลำดับสัญลักษณ์ เพื่อให้ตำแหน่งของสัญลักษณ์เดิมแยกออกจากกัน เมื่อสัญลักษณ์ถูกส่งไปยังช่องสัญญาณ เกิดความผิดพลาดของสัญลักษณ์แบบต่อเนื่อง คือ A, F และ D เมื่อทำการสลับลำดับกลับ สัญลักษณ์ที่เคยมีความผิดพลาดแบบต่อเนื่องมีการกระจายออกจากกัน ทำให้ตัวถอดรหัสสามารถแก้ไขความผิดพลาดได้

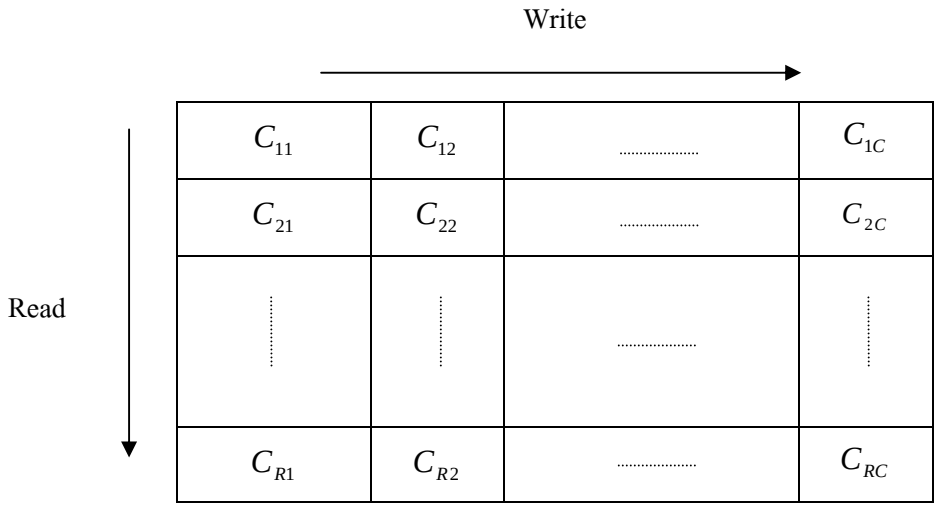


ภาพที่ 11 ขั้นตอนการสลับลำดับสัญลักษณ์

จากหลักการข้างต้นจึงนำเทคนิคนี้ใช้กับระบบส่งข้อมูลภาพต้นแบบ กระบวนการสลับลำดับข้อมูล และหลักการเบื้องต้น Dolinar and Divsalar (1995) แบ่งได้ 4 วิธี คือ การสลับลำดับข้อมูลแบบบล็อก การสลับลำดับข้อมูลแบบคอนโวลูชั่น การสลับลำดับข้อมูลแบบสุ่ม และ การสลับลำดับข้อมูลแบบกึ่งสุ่ม งานวิจัยนี้สนใจวิธีของกระบวนการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม โดยศึกษาบทบาทและหลักการของกระบวนการจาก Fragouli and Wesel (1999) จากนั้นจึงออกแบบเทคนิคดังกล่าวให้เหมาะสมกับระบบส่งข้อมูลภาพ

3.1.1 การสลับลำดับข้อมูลแบบบล็อก

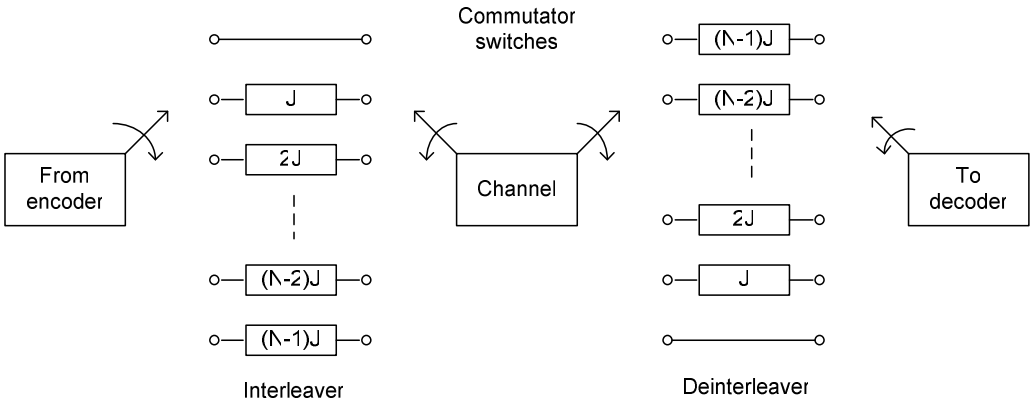
วิธีการสลับลำดับข้อมูลแบบบล็อก เริ่มจากการกำหนดขนาดของอาร์เรย์ให้เหมาะสมกับจำนวนของสัญลักษณ์ ซึ่งหาได้จาก $\sqrt{M \times N}$ หรือมีขนาด M แถว และ N คอลัมน์ จากนั้นนำสัญลักษณ์ที่ได้จากการเข้ารหัสเขียนลงในอาร์เรย์ทีละคอลัมน์จากซ้ายไปขวาและจากบนลงล่าง เมื่อนำสัญลักษณ์เขียนในอาร์เรย์ครบแล้ว ผ่านไปยังกระบวนการมอดูเลต โดยอ่านสัญลักษณ์ทีละแถวจากบนลงล่างและจากซ้ายไปขวา จากนั้นส่งไปยังช่องสัญญาณ ทางภาครับสัญลักษณ์ที่ได้จากช่องสัญญาณ จะผ่านกระบวนการดีมอดูเลต และ สลับลำดับกลับ โดยเขียนสัญลักษณ์ลงในอาร์เรย์ทีละแถว จากนั้นอ่านสัญลักษณ์ทางคอลัมน์ทีละคอลัมน์ จะได้ลำดับของสัญลักษณ์เดิมกลับมา เพื่อส่งไปยังการถอดรหัสข้อมูล ภาพที่ 12 เป็นการสลับลำดับข้อมูลแบบบล็อก



ภาพที่ 12 กระบวนการสลับลำดับแบบบล็อก

3.1.2 การสลับลำดับข้อมูลแบบคอนโวลูชั่น

วิธีการสลับลำดับแบบคอนโวลูชั่น Proakis (1995) มีหลักการเริ่มจาก สัญลักษณ์จะถูกเลื่อนไปภายในช่องว่างของรีจิสเตอร์ N ตัว ที่ต่อเนื่องกัน โดยจัดเตรียมพื้นที่ เพื่อเก็บค่าสัญลักษณ์ J สัญลักษณ์ เรียงลำดับก่อน-หลัง รีจิสเตอร์ศูนย์ ไม่มีการเก็บค่าสัญลักษณ์ไว้ เนื่องจากเมื่อรับสัญลักษณ์เข้ามาจะมีการส่งออกทันที และ เมื่อมีสัญลักษณ์ตัวใหม่เข้ามาสวิตซ์จะเปลี่ยนตำแหน่งไปยังรีจิสเตอร์ตัวถัดไป และ สัญลักษณ์ตัวใหม่จะถูกแทนที่เก็บไว้ในรีจิสเตอร์นั้น ส่วนสัญลักษณ์ก่อนหน้านี้จะเลื่อนตำแหน่งออกมาเป็นลำดับของสัญลักษณ์ที่สลับลำดับแล้ว เพื่อผ่านกระบวนการมอดูเลต เมื่อสวิตซ์ทำการเปลี่ยนการติดต่อไปยังรีจิสเตอร์ตัวสุดท้าย (N-1) สวิตซ์จะวนกลับไปสัมผัสกับรีจิสเตอร์ตัวที่ศูนย์อีกครั้งเพื่อเริ่มต้นใหม่ ในส่วนของกระบวนการสลับลำดับกลับ จะมีการกระทำกับสัญลักษณ์ที่ตรงกันข้าม



ภาพที่ 13 กระบวนการสลับลำดับแบบคอนโวลูชั่น

3.1.3 การสลับลำดับข้อมูลแบบสุ่ม

ทฤษฎีที่เกี่ยวข้องกับการสลับลำดับข้อมูลแบบสุ่ม Dolinar and Divsalar (1995) สามารถอธิบายหลักการของการสลับลำดับข้อมูลได้ดังนี้

การสลับลำดับข้อมูล ทำโดยนำเวกเตอร์ข้อมูลคูณกับเมตริกซ์เอกลักษณ์ P สมการที่ (29)

$$d_{INT} = dP \qquad (29)$$

โดยที่ d คือ เวกเตอร์ของข้อมูล และ P คือ เมตริกซ์เอกลักษณ์แบบสุ่มขนาด $L_T \times L_T$

การสลับลำดับข้อมูลกลับ ทำโดยนำข้อมูลที่ได้จากการสลับลำดับข้อมูลคูณกับการสลับเปลี่ยนของเมตริกซ์เอกลักษณ์แบบสุ่ม P_T สมการที่ (30)

$$d = d_{INT} P^T \qquad (30)$$

การใช้หลักการสลับเปลี่ยนเมตริกซ์มีค่าเท่ากับการทำผกผันเมตริกซ์ ซึ่งสามารถพิสูจน์ได้ดังนี้

$$PP^T = P^T P = I$$

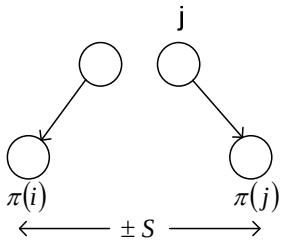
หรือ $$P^{-1} = P^T$$

3.1.4 การสลับลำดับข้อมูลแบบกึ่งสุ่ม

หลักการของกระบวนการสลับลำดับแบบกึ่งสุ่ม Fragouli and Divsalar (1999) และ Koutsouvelis and Dimakis คือ การสร้างตัวเลขแบบสุ่มจำนวน i ตัว แบบไม่ซ้ำ โดยที่ $0 \le i \le N-1$ ซึ่งตำแหน่งของข้อมูลเริ่มต้นที่ตำแหน่ง i และ j คือตำแหน่งของข้อมูลที่มีระยะห่างน้อยกว่าหรือเท่ากับพารามิเตอร์ S และ ตำแหน่งของข้อมูลที่ผ่านกระบวนการสลับลำดับแบบกึ่งสุ่ม $\pi(i)$ และ $\pi(j)$ ต้องมีระยะห่างมากกว่าหรือน้อยกว่า S หรือ $\pm S$ โดยกำหนดให้ i และ j แทนตำแหน่งของข้อมูลเริ่มต้น และกำหนดให้ $\pi(i)$ และ $\pi(j)$ แทนตำแหน่งของข้อมูลที่ผ่านกระบวนการสลับลำดับข้อมูลแบบกึ่งสุ่ม ซึ่งมีความสัมพันธ์ดังสมการ

$$|i-j| \le S \xrightarrow{\text{interleaver}} |\pi(i)-\pi(j)| > S \qquad (31)$$

โดยทั่วไปจะกำหนดให้ค่าพารามิเตอร์ $S < \sqrt{N/2}$ เนื่องจากค่าที่ใช้มีความเหมาะสมทางด้านเวลาในการสร้างตัวเลขสุ่มแบบกึ่งสุ่ม เพราะถ้า S ยิ่งเพิ่มมากขึ้นจะทำให้เวลาที่ใช้ในการสร้างตัวเลขสุ่มแบบกึ่งสุ่มเพิ่มขึ้น โดยที่ N คือขนาดความยาวของข้อมูล



ภาพที่ 14 ระยะห่างของการสลับลำดับแบบกึ่งสุ่ม

3.2 การสร้างตัวเลขแบบสุ่ม

การสร้างตัวเลขแบบสุ่มมีหลายวิธี ซึ่งจะใช้วิธีใดขึ้นอยู่กับความง่าย และ การกระจายของตัวเลขที่สุ่มได้ การสร้างตัวเลขแบบสุ่ม เริ่มจากการกำหนดค่าเริ่มต้นของการสุ่ม เพื่อให้ได้ลำดับของการสุ่มรูปแบบเดิม และ การกำหนดตัวเลขในการสุ่มว่าต้องการค่าของการสุ่มในช่วงใด เพื่อที่จะได้ตัวเลขแบบสุ่ม $x = \{x_n\}_{n=0}^{m-1}$ โดยวิธีการสร้างตัวเลขแบบสุ่มที่ใช้ในงานวิจัยนี้มี 2 วิธี คือ

การใช้อัลกอริทึม Linear Congruence และ การเรียกใช้ฟังก์ชันสุ่มในโปรแกรมภาษาจาวา ซึ่งทั้ง 2 วิธีนี้ มีหลักการดังนี้

3.2.1 วิธีการสร้างตัวเลขแบบสุ่มโดยอัลกอริทึม Linear Congruence

การสลับลำดับข้อมูลแบบสุ่มจำเป็นต้องสร้างตัวเลขแบบสุ่มเพื่อใช้เป็นค่าในการกำหนดตำแหน่งของสัญลักษณ์ โดยการคำนวณลำดับของตัวเลขแบบสุ่ม $x = \{x_n\}_{n=0}^{m-1}$ หาได้จากสมการ

$$x_{n+1} = (ax_n + c) \bmod m, \quad n = 0,1,...,m-1 \qquad (32)$$

โดยที่ $x_n \in \{0,1,...,m-1\}$, x_0 เป็นจำนวนเต็มใดๆในเซต $\{0,1,...,m-1\}$ a และ c เป็นค่าคงที่โดยมีขั้นตอนการเลือกค่าของ a และ c ดังนี้

ขั้นตอนที่ 1 ให้ b_0 เป็นผลคูณของจำนวนเฉพาะที่เป็นตัวร่วมของ m และ ไม่ซ้ำกัน
ขั้นตอนที่ 2 ให้ $b = vb_0$ โดยที่ v เป็นจำนวนเฉพาะ ดังนั้น $b = vb_0 \le m$
ขั้นตอนที่ 3 คำนวณหาค่า a ได้จาก $a = b+1$
ขั้นตอนที่ 4 เลือก c โดยที่ c ต้องเป็นจำนวนเฉพาะสัมพัทธ์กับ m และ $1 < c < m$
ขั้นตอนที่ 5 ให้ x_0 เป็นตัวเลขใดๆ ที่ $0 \le x_0 < m$

จะได้ลำดับของการจัดตำแหน่ง $x = \{x_n\}_{n=0}^{m-1}$ ซึ่งเป็นการสร้างตัวเลขแบบสุ่ม โดยใช้ตัวเลขแบบสุ่มลำดับที่ n สร้างตัวเลขตัวถัดไปลำดับที่ $n+1$

จำนวนเฉพาะสัมพัทธ์ คือ gcd(m,n) = 1 หมายความว่าตัวหารร่วมมากของจำนวนเต็ม m และ n ไม่มีตัวหารที่มากกว่า 1 ร่วมกันเลย จึงเรียก m และ n ว่าจำนวนเฉพาะสัมพันธ์

<u>ตัวอย่างที่ 1</u> ถ้า m = 10 สามารถสร้างเมตริกซ์แบบสุ่มได้ดังนี้
คำนวณหาค่า b_0 = (2)(5) = 10 จะได้ตัวประกอบจำนวนเฉพาะของ m คือ m = (2)(5) จากนั้น $b = vb_0$ = (1)(10) = 10 เมื่อเลือก v = 1 จะได้ $a = b+1$ = 11 สำหรับค่า m = 10, ค่า c ที่นำมาใช้ได้คือ 3 และ 7 เนื่องจาก c ต้องเป็นจำนวนเฉพาะสัมพัทธ์กับ m และ $1 < c < 10$ ในการสร้างการสุ่มตัวเลขโดยใช้หลักการหาค่าลำดับที่ $n+1$ จากค่าลำดับที่ n โดย n = 0,1,2,…,9 ในสมการที่ (32) โดยเลือก $c = 7$ และ $x_0 = 1$ จะได้ตัวเลขแบบสุ่มคือ x = {1,8,5,2,9,6,3,0,7,4}

3.2.2 การสร้างตัวเลขแบบสุ่มโดยเรียกใช้ฟังก์ชันในโปรแกรมภาษาจาวา

การสร้างลำดับตัวเลขแบบสุ่ม โดยการเรียกใช้ฟังก์ชันแบบสุ่มจากโปรแกรมภาษาจาวา ดังนี้

```
Random rand = new Random(SMseed)
```

กำหนดค่าเริ่มต้น เพื่อใช้ในการตรวจสอบลำดับการสุ่ม ทำให้ลำดับของตัวเลขที่สุ่มได้เป็นลำดับการสุ่มชุดเดิม และกำหนดจำนวนของตัวเลขที่ต้องการ ในที่นี้กำหนดให้มีค่าเท่ากับ [ขนาดของข้อมูล - 1] โดยสร้างอาร์เรย์เก็บค่าตัวเลขที่ต้องการไว้ คือ number[] จากนั้นจะทำการสุ่มตัวเลขนั้นทีละตัวดังนี้

```
pointer = rand.nextInt(size-i)+i
```

โดยค่า pointer คือ ค่าที่สุ่มเกิดจากการนำค่าขนาดลบกับจำนวนของตำแหน่งในปัจจุบัน เพื่อเป็นการสุ่มแบบเอกรูปหรือ Uniform ได้บวกกับค่าของตำแหน่งปัจจุบัน จะได้การสุ่มของตัวเลขที่ไม่ซ้ำกัน จากนั้นจะผ่านไปยังกระบวนการสับค่าหรือ swap เพื่อทำการสลับค่าพารามิเตอร์ระหว่าง (number, point, i) เมื่อทำกระบวนการเสร็จจะได้ลำดับตัวเลขแบบสุ่ม

3.3 การจำลองช่องสัญญาณ

ทฤษฎีเป็นการเลือนหายของสัญญาณในเชิงขนาด คือ ช่องสัญญาณไร้สายที่ทำให้สัญญาณข้อมูลมีการเปลี่ยนแปลงในเชิงขนาด โดยไม่มีการสูญเสียคุณสมบัติในเชิงความถี่ ช่องสัญญาณไร้สายแบบเลือนหายของสัญญาณในเชิงขนาดแบบช้า ที่มีค่าคงที่ภายใน M ช่วงสัญญาณข้อมูล โดยที่ M = 2, 3,… ในการตรงกันข้าม ช่องสัญญาณไร้สายแบบการเลือนหายของสัญญาณในเชิงขนาดแบบเร็ว มีค่าของขนาดเปลี่ยนไปทุกๆช่วงของสัญญาณข้อมูล

อุปกรณ์และวิธีการ

อุปกรณ์

1. คอมพิวเตอร์
2. โปรแกรมภาษาจาวา
3. รูปภาพ Lena

วิธีการ

งานวิจัยนี้นำระบบการส่งข้อมูลภาพต้นแบบ Mahapakulchai and Van Dyck (2004) โดยนำกระบวนการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่มใช้กับระบบ โครงสร้างของระบบการส่งข้อมูลภาพแสดงดังภาพที่ 1 ซึ่งการออกแบบของระบบจะแบ่งเป็น 3 ส่วน คือ การออกแบบของระบบต้นแบบ การออกแบบโดยการปรับปรุงระบบเดิมจากการใช้กระบวนการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม และ การจำลองช่องสัญญาณที่มีการเลือนหายของสัญญาณในเชิงขนาดแบบช้า

1. การออกแบบระบบการส่งข้อมูลภาพของระบบต้นแบบ

การเข้าและถอดรหัสข้อมูลภาพแบบ MPEG-4 ของรูปภาพ Lena เป็นการบีบอัดข้อมูลที่มีประสิทธิภาพ เริ่มจากการแยกข้อมูลออกเป็น 2 ส่วน คือ กลุ่มสัมประสิทธิ์ LFS และ HFS การเปลี่ยนข้อมูลภาพจากบิตเป็นสัญลักษณ์ แต่ละกลุ่มสัญลักษณ์มีความยาวไม่เท่ากัน ประกอบด้วย Zerotree, Non-Zero Values, Cyclic Redundancy Check และ Header ทั้งหมด 102 กลุ่ม โดยแบ่งเป็นกลุ่มสัมประสิทธิ์ LFS 1 กลุ่ม และ กลุ่มสัมประสิทธิ์ HFS 101 กลุ่ม ความยาวเฉลี่ยประมาณ 325 สัญลักษณ์ต่อกลุ่ม

การเปลี่ยนข้อมูลภาพจากบิตเป็นสัญลักษณ์โดยสัญลักษณ์ 1 ค่าเท่ากับ 2 บิต มีสัญลักษณ์ 4 รูปแบบ คือ 0, 1, 2 และ 3 บิต 00 แทนด้วยสัญลักษณ์ 0 บิต 01 แทนด้วยสัญลักษณ์ 1 บิต 10 แทนด้วยสัญลักษณ์ 2 และบิต 11 แทนด้วยสัญลักษณ์ 3 เพื่อให้เหมาะสมกับการเข้ารหัสแบบริงคอนโวลูชั่น

การเข้ารหัสช่องสัญญาณแบบเทลลิส ประกอบด้วย 2 ส่วน คือการเข้ารหัสช่องสัญญาณแบบริงคอนโวลูชั่น และ การมอดูเลตแบบที่เฟสของสัญญาณมีความต่อเนื่อง การถอดรหัสวีเทอร์บี การคำนวณหาเส้นทางบนเทลลิสใช้ทฤษฎีการตรวจจับสัญญาณแบบ ML และ MAP การเข้ารหัสของกระบวนการ CPE ที่มีอัตรา = 1/2 ดังสมการที่ (34)

$$G_{CPE} = \begin{bmatrix} 1 & 0 & 3 & 1 \\ 3D & D & 1 & 0 \end{bmatrix} \tag{34}$$

งานวิจัยนี้ได้นำการเข้ารหัสโพลีนอเมียลริงคอนโวลูชั่น 32 ขั้น โดยการทำกระบวนการ RCE ที่อัตรา = 1/2 คูณกับการทำกระบวนการ CPE ที่อัตรา = 2/4 จะได้ Generator Matrix ของการเข้ารหัสแบบริงคอนโวลูชั่น สมการที่ (35)

$$\begin{aligned} G_{all}(D) &= \begin{bmatrix} 1+D+2D^3 \\ 2+D \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 3 & 1 \\ 3D & D & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1+3D+3D^2+2D^3 \\ 2D+D^2 \\ 1+2D^3 \\ 1+D+2D^3 \end{bmatrix} \end{aligned} \tag{35}$$

การคำนวณหาเส้นทางบนเทลลิสโดยใช้ทฤษฎีการตรวจจับสัญญาณแบบ MAP จะใช้กับสัญลักษณ์ซีโร่ทรีเท่านั้น ดังนั้นจึงมีการเพิ่มบิตตัวบ่งชี้หรือ Flag เข้าไป เพื่อใช้เป็นบิตตรวจสอบเงื่อนไขในการคำนวณหาเส้นทางบนเทลลิส

การจำลองช่องสัญญาณไร้สาย มีรูปแบบการเลือนหายของสัญญาณในเชิงขนาดที่เกิดขึ้นใน 2 ลักษณะ คือ แบบช้า และ แบบเร็ว โดยในระบบต้นแบบได้มีการจำลองช่องสัญญาณไร้สายแบบการเลือนหายของสัญญาณในเชิงขนาดแบบเร็วอยู่ก่อนแล้ว

2. การปรับปรุงระบบการส่งข้อมูลภาพ

การปรับปรุงระบบข้อมูลภาพเริ่มจากการใช้กระบวนการสลับลำดับข้อมูลแบบบล็อก Mahapakulchai and Thongnumpen (2007) โดยนำผลการทดลองมาแสดงไว้ดังตารางผลการทดลองที่ 4 และ 5 จากกระบวนการดังกล่าวพบว่ามีความยุ่งยากเนื่องจากต้องหาเมตริกซ์การสลับของความยาวข้อมูลในแต่ละกลุ่ม เนื่องจากข้อมูลมีความยาวไม่คงที่ และการออกแบบขนาดของเมตริกซ์การสลับอาจมีตำแหน่งว่างอยู่ ดังนั้นจึงต้องมีการเพิ่มบิตพิเศษเข้าไปภายในพื้นที่ว่างนั้น และในการสลับลำดับข้อมูลกลับต้องทำการแยกบิตพิเศษออกจากรหัสข้อมูล ถ้าออกแบบไม่ดีอาจทำให้บิตพิเศษที่เติมเข้าไปนั้นรวมเข้าไปอยู่กับรหัสข้อมูลได้ ซึ่งอาจทำให้เกิดความผิดพลาดของข้อมูล

จากเหตุผลข้างต้น จึงนำการสลับลำดับข้อมูลแบบสุ่มเข้ามาใช้กับระบบ เพื่อลดความยุ่งยาก เริ่มต้นใช้อัลกอริทึม Linear Congruence ในการสร้างตัวเลขแบบสุ่ม เพื่อสร้างเมตริกซ์ P เป็นการออกแบบขั้นพื้นฐานให้เหมาะสมกับระบบส่งภาพต้นแบบ วิธีการนี้ เป็นการสิ้นเปลืองพื้นที่ในหน่วยความจำ และ ใช้เวลานาน ดังนั้นจึงมีการปรับปรุงการออกแบบในส่วนนี้ใหม่โดยใช้หลักการเดิม จากนั้นปรับปรุงการสร้างตัวเลขแบบสุ่ม จากการเรียกใช้ฟังก์ชันแบบสุ่มในโปรแกรมภาษาจาวา โดยกำหนดค่าเริ่มต้นและจำนวนตัวเลขที่ต้องการในการสุ่ม เพื่อให้สามารถนำไปใช้งานได้จริง นอกจากนี้ยังมีการจำลองช่องสัญญาณแบบการเลือนหายของสัญญาณในเชิงขนาดแบบช้าเพื่อใช้ในการทดสอบการสลับลำดับข้อมูลแบบสุ่ม และ กึ่งสุ่ม และในส่วนสุดท้าย เป็นการแสดงเวลาในการประมวลผล

2.1 การออกแบบการสลับลำดับข้อมูลภาพแบบสุ่ม

2.1.1 การออกแบบโดยใช้อัลกอริทึม Linear Congruence

การออกแบบโดยใช้อัลกอริทึม Linear Congruence มีขั้นตอนการออกแบบอยู่ 4 ขั้นตอนดังนี้

1. การสร้างตัวเลขแบบสุ่ม
2. การสร้างเมตริกซ์เอกลักษณ์แบบสุ่ม P
3. การสลับเปลี่ยนเมตริกซ์เอกลักษณ์แบบสุ่ม P^T
4. การสลับลำดับข้อมูล

5. การสลับลำดับข้อมูลกลับ

เริ่มจากการสร้างตัวเลขแบบสุ่มโดยใช้อัลกอริทึม Linear Congruence จากสมการที่ (32) ซึ่งในการออกแบบกำหนดค่าพารามิเตอร์ต่างๆ ดังนี้ x_0 หรือค่าเริ่มต้นมีค่าเท่ากับ 1 ในสัญลักษณ์กลุ่มแรก และ เพิ่มค่าอีก 1 ในแต่ละรอบของกลุ่มสัญลักษณ์ถัดไป พารามิเตอร์ c_0 หรือค่าที่ต้องการให้ลำดับการสุ่มในแต่ละรอบมีการกระโดดเป็นระยะทางเท่าไร ซึ่งในที่นี้กำหนดให้กลุ่มสัญลักษณ์ที่ 7, 8 และ 12 มีค่า c_0 = 17 นอกนั้นกำหนดให้ c_0 = 13 โดยค่า c_0 และ ขนาดของข้อมูลแต่ละกลุ่มมีความสัมพันธ์กัน โดยต้องเป็นจำนวนเฉพาะสัมพัทธ์ และกำหนดให้พารามิเตอร์ a = size+1 โดยที่ size คือขนาดของกลุ่มสัญลักษณ์ วิธีการเลือกค่าพารามิเตอร์เป็นการออกแบบเพื่อใช้สำหรับระบบข้อมูลภาพในงานวิจัยนี้ เมื่อได้ตัวเลขแบบสุ่ม $x = \{x_n\}_{n=0}^{m-1}$ ขั้นตอนถัดไปเป็นการสร้างเมตริกซ์เอกลักษณ์แบบสุ่ม P โดยใช้ลำดับตัวเลขแบบสุ่มในการกำหนดค่าทางแถวของเมตริกซ์ ตัวเลขแบบสุ่มที่สร้างไว้เป็นตัวกำหนดเงื่อนไขในแต่ละแถวของเมตริกซ์แบบสุ่ม P ดังนี้ ถ้าค่าของตัวเลขที่สุ่มได้ตรงกับตำแหน่งในคอลัมน์ใดให้กำหนดคอลัมน์นั้นมีค่าเท่ากับ 1 คอลัมน์ที่เหลือของแถวนั้นกำหนดให้มีค่าเท่ากับ 0 ลำดับของการสุ่ม หมายถึงแถว และ ตัวเลขที่สุ่มได้ คือ ตำแหน่งของคอลัมน์ที่มีค่าเท่ากับ 1 คอลัมน์ที่เหลือ มีค่าเท่ากับ 0 เมื่อนำลำดับตัวเลขแบบสุ่มทุกตัวทำเงื่อนไขจนครบจะได้เมตริกซ์เอกลักษณ์แบบสุ่ม P โดยขนาดของเมตริกซ์มีขนาดเท่ากับจำนวนข้อมูลแต่ละกลุ่มสัญลักษณ์ จากนั้นทำการสลับลำดับข้อมูลแบบสุ่มตามสมการที่ (29) และขั้นตอนสุดท้ายทำการสลับลำดับข้อมูลกลับโดยสมการที่ (30)

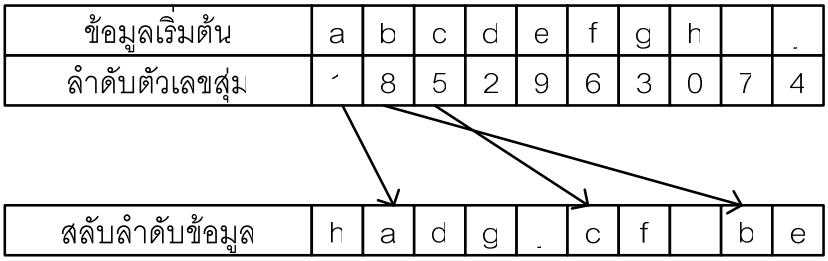
<u>ตัวอย่างที่ 2</u> การสลับลำดับข้อมูลแบบสุ่มของข้อมูล d = {a, b, c, d, e, f, g, h, i, j} มีขั้นตอนดังนี้ จากข้อมูลทั้งหมด 10 ตัว โดยใช้ลำดับตัวเลขแบบสุ่มจากตัวอย่างที่ 1 คือ x = {1, 8, 5, 2, 9, 6, 3, 0, 7, 4} จากนั้นนำลำดับตัวเลขแบบสุ่มที่ได้สร้างเมตริกซ์เอกลักษณ์แบบสุ่มตามเงื่อนไขในหัวข้อ (2.1.1) จะได้

$$P=\begin{bmatrix}0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0&0&0\\0&0&0&1&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&1&0&0&0&0&0\end{bmatrix}$$ และ $$P^{\mathrm{T}}=\begin{bmatrix}0&0&0&0&0&0&0&1&0&0\\1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&0&0&1\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&1&0\\0&1&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\end{bmatrix}$$

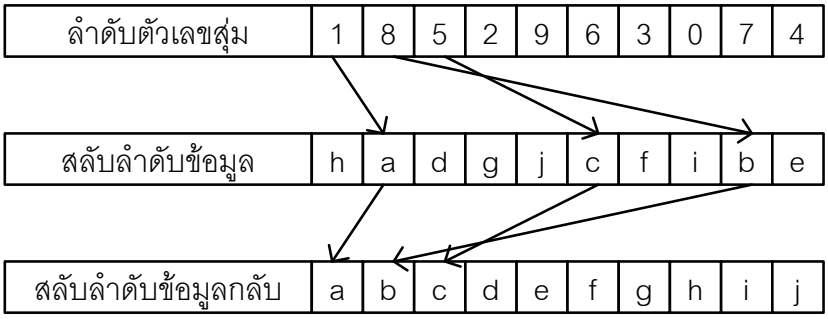
กระบวนการสลับลำดับข้อมูล จากสมการที่ (29) คือ $d_{INT} = dP$ จะได้ข้อมูลที่ผ่านการสลับลำดับดังนี้ d_{INT} = {h, a, d, g, j, c, f, i, b,e}

ในขั้นตอนสุดท้ายทำการสลับลำดับข้อมูลกลับ จากสมการที่ (30) คือ $d = d_{INT} \times P^T$ จะได้ข้อมูลเริ่มต้นกลับคืนดังนี้ d = {a, b, c, d, e, f, g, h, i, j}

อย่างไรก็ตามการสร้างเมตริกซ์ P เพื่อใช้ในการสลับลำดับข้อมูล ต้องใช้อาร์เรย์ที่มีขนาดใหญ่มาก ทำให้สิ้นเปลืองเวลาและพื้นที่ในการสร้างเมตริกซ์ P ไม่เหมาะสมกับการนำไปใช้งานในระบบจริง ดังนั้นจึงเปลี่ยนวิธีการกำหนดรูปแบบในการสลับลำดับข้อมูลใหม่แต่ยังคงหลักการเดิมของกระบวนการนี้ เริ่มจากนำลำดับของตัวเลขที่ทำการสุ่มได้ เป็นตัวเลขที่ชี้ตำแหน่งของข้อมูลเริ่มต้นไปยังตำแหน่งของข้อมูลที่ทำการสลับลำดับ เมื่อทำกระบวนการดังกล่าวครบทุกค่าแล้ว จะได้ข้อมูลของการสลับลำดับ และในส่วนของการสลับลำดับข้อมูลกลับ เริ่มจากการใช้ลำดับของตัวเลขแบบสุ่มชุดเดียวกันเป็นตัวชี้ไปยังตำแหน่งที่ตรงกับตัวเลขสุ่มของข้อมูลที่สลับลำดับแล้วเพื่อดึงข้อมูลที่อยู่ในตำแหน่งนั้นมาวางไว้ในตำแหน่งของข้อมูลการสลับลำดับกลับ ดังภาพที่ 15 และ 16 โดยวิธีการนี้ได้นำไปใช้กับการสลับลำดับข้อมูลแบบสุ่มโดยใช้ฟังก์ชันแบบสุ่มในโปรแกรมภาษาจาวา และ การสลับลำดับข้อมูลแบบกึ่ง ซึ่งวิธีการออกแบบจะอยู่ในส่วนถัดไป



ภาพที่ 15 การสลับลำดับโดยไม่สร้างเมตริกซ์ P



ภาพที่ 16 การสลับลำดับกลับโดยไม่สร้างเมตริกซ์ P^T

2.1.2 การออกแบบโดยใช้ฟังก์ชันสุ่มในโปรแกรมภาษาจาวา

วิธีการออกแบบโดยเรียกใช้ฟังก์ชันแบบสุ่มในโปรแกรมภาษาจาวามี 3 ขั้นตอน

1. การสร้างตัวเลขแบบสุ่ม
2. การสลับลำดับข้อมูล
3. การสลับลำดับข้อมูลกลับ

เริ่มจากการสร้างลำดับตัวเลขแบบสุ่ม โดยการเรียกใช้ฟังก์ชันสุ่มจากโปรแกรมภาษาจาวา ซึ่งได้อธิบายไว้แล้วในหัวข้อ (3.2.2) เมื่อทำกระบวนการเสร็จจะได้ลำดับตัวเลขแบบสุ่ม ในขั้นตอนต่อไปคือการสลับลำดับข้อมูลและในส่วนสุดท้ายคือการสลับลำดับข้อมูลกลับซึ่งทั้ง 2 กระบวนการนี้จะใช้วิธีการเดียวกับวิธีการที่ใช้ในหัวข้อ (2.1.1)

อย่างไรก็ตามการสลับลำดับข้อมูลแบบนี้ มีขั้นตอนไม่ซับซ้อนแต่ลำดับตัวเลขที่ได้จากการสุ่มโดยใช้ฟังก์ชันในโปรแกรมภาษาจาวา ไม่สามารถกำหนดการกระจายของตัวเลขเหล่านั้นได้ จึงปรับปรุงวิธีการใหม่ โดยการเพิ่มเงื่อนไขในการตรวจสอบเข้าไปในกระบวนการ เรียกว่า การสลับลำดับข้อมูลแบบกึ่งสุ่ม ซึ่งรายละเอียดอยู่ในหัวข้อถัดไป

2.2 การออกแบบการสลับลำดับข้อมูลแบบกึ่งสุ่ม

วิธีการออกแบบการสลับลำดับข้อมูลแบบกึ่งสุ่ม มี 5 ขั้นตอน

1. สร้างตัวเลขแบบสุ่ม
2. จัดเรียงตัวเลขแบบสุ่ม
3. ตรวจสอบเงื่อนไข
4. การสลับลำดับข้อมูล
5. การสลับลำดับข้อมูลกลับ

การสร้างตัวเลขแบบสุ่มจะใช้วิธีการเดียวกับหัวข้อที่ (2.1.2) จากนั้นทำการจัดเรียงลำดับตัวเลขแบบสุ่มโดยใช้การจัดเรียงโครงสร้างแบบต้นไม้ เพื่อให้มีการกระจายของลำดับตัวเลขแบบสุ่มมากขึ้น เมื่อทำการจัดเรียงเสร็จแล้ว ขั้นตอนถัดไปจะนำข้อมูลตัวเลขแบบสุ่มที่ได้มาทำการตรวจสอบเงื่อนไข โดยกำหนดให้พารามิเตอร์ S = 10 และ 17 ซึ่งแสดงได้ดังสมการที่ (31)

$$|i-j| \le S \xrightarrow{\text{interleaver}} |\pi(i)-\pi(j)| > S$$

เมื่อลำดับของตัวเลขแบบสุ่มตรงตามเงื่อนไขแล้ว ขั้นตอนต่อไปคือการสลับลำดับข้อมูล และในส่วนสุดท้าย คือ การสลับลำดับข้อมูลกลับ ซึ่งทั้ง 2 กระบวนการนี้จะใช้วิธีการเดียวกับวิธีการที่ใช้ในหัวข้อ (2.1.1)

2.3 การจำลองช่องสัญญาณแบบการเลือนหายของสัญญาณในเชิงขนาดแบบช้า

วิธีการปรับเปลี่ยนช่องสัญญาณจากเดิมแบบการเลือนหายของสัญญาณในเชิงขนาดแบบเร็ว โดยมีการกำหนดค่า แอลฟา 1 ค่ามีผลต่อสัญลักษณ์ 1 สัญลักษณ์ เปลี่ยนเป็นช่องสัญญาณที่มีการเลือนหายของสัญญาณแบบการเลือนหายของสัญญาณในเชิงขนาดแบบช้า โดยการกำหนดค่า

แอลฟา 1 ค่ามีผลต่อสัญลักษณ์ในช่วงเวลา M สัญลักษณ์ โดยที่ M = 2, 3,… ในระบบนี้ กำหนดให้ M = 5 ทำให้ช่องสัญญาณจากเดิมที่ไม่มีหน่วยความจำกลายเป็นช่องสัญญาณที่มีหน่วยความจำ สัญญาณที่ได้รับจากช่องสัญญาณแบบนี้ มีผลทำให้การถอดรหัสวีเทอร์บีมีประสิทธิภาพในการแก้ไขความผิดพลาดลดลง ค่าแอลฟา แสดงดังสมการที่ (24) คือ พจน์ของ $(1+\beta)$ ที่ทำให้สัญญาณที่ได้รับมีการลดทอนของสัญญาณ

2.4 การแสดงเวลาในการประมวลผล

การแสดงเวลาในการประมวลผล เริ่มจากการดึงเวลาจากคอมพิวเตอร์ขึ้นมาใช้ในการประมวลผลของระบบ ดังนี้

Locale Ic = new Locale(“th” , “TH”)

String[]patterns={“H:mm:ss:SSS”}

ฟังก์ชันด้านบนประกอบด้วย การเรียกวัตถุทางเวลาของพื้นที่ในประเทศไทยขึ้นมาใช้ จากนั้นกำหนดรูปแบบของตัวอักษรที่จะให้แสดงค่าโดย H หมายถึงหน่วยชั่วโมง mm หมายถึงหน่วยนาที ss หมายถึงหน่วยวินาที และ SSS หมายถึงหน่วยมิลลิวินาที

3. การทดสอบระบบข้อมูลภาพ

การทดสอบระบบข้อมูลภาพ ทำโดยนำการสลับลำดับสัญลักษณ์แบบสุ่ม และ กึ่งสุ่ม มาใช้กับระบบข้อมูลภาพแสดงในภาพที่ 1 รูปภาพที่ใช้ทำการทดสอบ คือ รูปภาพ Lena ขนาด 512×512 จุดภาพ จำนวน 200 รูปภายในช่วง $\bar{\gamma}_b$ = 3.75, 5.00 และ 6.25 dB โดยผ่านกระบวนการบีบอัดข้อมูลภาพด้วยวิธีการ EZW เพื่อแยกข้อมูลออกเป็น 2 ส่วน คือส่วนข้อมูลสัมประสิทธิ์ LFS และ HFS ทำการแบ่งข้อมูลออกเป็น 5 ระดับ จากนั้นนำส่วนของข้อมูล HFS ผ่านกระบวนการควอนไตซ์จะได้บิตข้อมูล และแบ่งบิตข้อมูลเป็นกลุ่มซึ่งแต่ละกลุ่มมีความยาวของบิตข้อมูลไม่เท่ากัน โดยความยาวเฉลี่ยประมาณ 325 สัญลักษณ์ต่อกลุ่ม ในส่วนของการเข้ารหัสช่องสัญญาณ เลือกใช้การเข้ารหัสช่องสัญญาณแบบริงคอนโวลูชั่น ที่มีจำนวน 32 ขั้น และ อัตรา = 1/2 ขั้นตอนถัดไปเป็นการสลับลำดับสัญลักษณ์แบบสุ่ม จะมีการนำวิธีการลำดับสัญลักษณ์แบบสุ่ม และ กึ่งสุ่ม ใช้กับระบบข้อมูลภาพ ที่ทดสอบกับช่องสัญญาณแบบการเลือนหายของสัญญาณในเชิงขนาดแบบช้า และ แบบเร็ว มีการกระจายแบบไรเชียล ที่มี γ = -43 dB, -33 dB และ -23 dB โดยที่ถ้า $\gamma \to -\infty$ จะทำให้

ช่องสัญญาณมีการกระจายแบบเรเร่หรือ Rayleigh ขั้นตอนถัดไปเป็นการคำนวณหาเส้นทางบนเทลลิส โดยใช้การตรวจจับความผิดพลาดแบบ ML และ MAP จากนั้นทำการสลับลำดับข้อมูลกลับเพื่อทำการถอดรหัสวีเทอร์บี และขั้นตอนสุดท้ายคือการถอดรหัสข้อมูลภาพ ในกรณีที่ไม่มีการรบกวนของสัญญาณรบกวนเมื่อส่งข้อมูลภาพผ่านช่องสัญญาณทั้ง 2 แบบ ประสิทธิภาพของระบบวัดค่า PSNR เท่ากับ 31.83 dB และมีอัตราส่วนของการบีบอัดข้อมูลภาพ คือ 31:1

ภาพที่ 17 รูปภาพต้นแบบ Lena ที่ไม่มีสัญญาณรบกวน

4. การวิเคราะห์ประสิทธิภาพของระบบ

4.1 การคำนวณค่า Peak Signal to Noise Ratio (PSNR)

PSNR คือ ค่าสูงสุดของอัตราส่วนระหว่างสัญญาณและสัญญาณรบกวน บอกถึงประสิทธิภาพของระบบ ระบบที่ดีต้องมีค่า PSNR ใกล้เคียงกับต้นแบบมากที่สุด การหาค่า PSNR เป็นดังสมการที่ (37)

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left\|I(i,j)-K(i,j)\right\|^2 \qquad (36)$$

$$\text{PSNR} = 10\ \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \qquad (37)$$

MAX_I = ค่าจุดภาพสูงสุดของรูปภาพ คือ 2^B-1 ซึ่ง B คือ จำนวนของ Bits/Sample และ $\left\|I(i,j)-K(i,j)\right\|$ คือ ค่าความผิดพลาดของผลต่างระหว่างภาพต้นแบบและภาพที่รับได้

4.2 การคำนวณค่า Word Error Rate (WER)

WER คือ อัตราส่วนระหว่างจำนวนความผิดพลาดของสัญลักษณ์ที่ได้รับและจำนวนสัญลักษณ์ทั้งหมด ระบบที่ดีต้องมีค่า WER ต่ำหรือเข้าใกล้ศูนย์ การหาค่า WER เป็นดังสมการที่ (38)

$$\text{WER} = \text{จำนวนสัญลักษณ์ที่ผิดพลาด/จำนวนสัญลักษณ์ทั้งหมด} \qquad (38)$$

5. สถานที่ทำการทดลองและระยะเวลาในการทดลอง

สถานที่ทำการทดลอง คือ ห้องปฏิบัติการ Cubic Space ชั้น 5 ห้อง 505/5 ตึก 60 ปี คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ รวมระยะเวลาทั้งสิ้น 1 ปี

ผลและวิจารณ์

ผล

การทดสอบประสิทธิภาพของระบบ ทำโดยนำการสลับลำดับข้อมูลแบบสุ่ม และ แบบกึ่งสุ่มใช้กับระบบส่งข้อมูลภาพ รูปภาพที่ใช้ในการทดสอบระบบ คือ รูปภาพ Lena ขนาด 512×512 จุดภาพโดยใช้เทคนิค EZW ในการบีบอัดข้อมูลภาพ แบ่งข้อมูลภาพออกเป็น 5 ระดับ และแยกออกเป็นสัมประสิทธิ์ LFS และ HFS จากนั้นส่งไปยังกระบวนการควอนไตซ์จะได้บิตของข้อมูลภาพ และแบ่งบิตของข้อมูลภาพออกเป็นกลุ่มทั้งหมด 102 กลุ่ม มีความยาวแต่ละกลุ่มไม่เท่ากัน บิตของข้อมูลภาพในแต่ละกลุ่มจะถูกแทนด้วยสัญลักษณ์ความยาวเฉลี่ยประมาณ 325 สัญลักษณ์ต่อกลุ่ม การเข้ารหัสช่องสัญญาณจะใช้การเข้ารหัสริงคอนโวลูชั่น 32 ขั้น ดังสมการที่ (34) จากนั้นทำการสลับลำดับของแต่ละกลุ่มสัญลักษณ์ โดยทดสอบระบบ 2 กรณี คือ การสลับลำดับสัญลักษณ์ของสัมประสิทธิ์ HFS และ การสลับลำดับสัญลักษณ์ของสัมประสิทธิ์ LFS และ HFS จากนั้นมอดูเลตสัญญาณ และ ส่งผ่านไปยังช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า ทางด้านรับ ประกอบด้วย การดีมอดูเลต การสลับลำดับสัญลักษณ์กลับ การคำนวณหาเส้นทางบนเทลลิส และ การถอดรหัสวีเทอร์บี ปัจจัยการวิเคราะห์ผลการทดลองเพื่อวัดประสิทธิภาพของระบบส่งข้อมูลภาพ ประกอบด้วย ค่าเฉลี่ย PSNR, WER และ เวลาที่ใช้ในการประมวลผลภาพ ในกรณีที่ไม่มีสัญญาณรบกวนต่อระบบจะมีค่า PSNR = 31.83 dB โดยมีอัตราส่วนการบีบอัดข้อมูล 31:1 เมื่อพิจารณาในขณะที่ไม่มีการนำกระบวนการสลับลำดับข้อมูลเข้ามาใช้กับระบบ เวลาในการประมวลผลประมาณ 2.14 วินาทีต่อภาพ

เริ่มจากการทดสอบระบบแบ่งเป็น 2 กรณี คือ การทดสอบช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า ที่ γ = -23 dB และ $\bar{\gamma}_b$ = 5 dB โดยกำหนดค่าแอลฟา 1 ค่ามีผลต่อสัญลักษณ์ M สัญลักษณ์ โดยที่ M = 1 จะมีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ M = 2,3,… จะมีการเลือนหายทางขนาดของสัญญาณแบบช้า ในกรณีที่ไม่มีการสลับลำดับสัญลักษณ์ โดยใช้การถอดรหัสแบบ ML และ ทดสอบความสำคัญของกลุ่มสัญลักษณ์ LFS ต่อระบบข้อมูลภาพ

1. การทดสอบช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า ที่ γ = -23 dB และ $\bar{\gamma}_b$ = 5 dB ในกรณีที่ไม่มีการสลับลำดับสัญลักษณ์โดยใช้การถอดรหัสแบบ ML

ตารางที่ 2 เป็นผลการทดสอบช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า ที่ γ = -23 dB และ $\bar{\gamma}_b$ = 5 dB โดยกำหนดการสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ M สัญลักษณ์ ซึ่งมี 2 กรณี คือ กรณีที่การสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ 1 สัญลักษณ์ เป็นการทดสอบกับช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ กรณีที่การสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ = 2, 3, 4, 5 และ 10 เป็นการทดสอบกับช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า โดยไม่มีการสลับลำดับสัญลักษณ์ทั้ง 2 กรณี ใช้การถอดรหัสแบบ ML แสดงค่าเฉลี่ย PSNR และ WER

ตารางที่ 2 ค่าเฉลี่ย PSNR ของการส่งภาพผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า โดยกำหนดค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ M สัญลักษณ์ M = 1, 2, 3, 4, 5 และ 10 ที่ γ = -23 dB และ $\bar{\gamma}_b$ = 5 dB โดยไม่มีการสลับลำดับสัญลักษณ์ ใช้การถอดรหัสแบบ ML

ช่วงของสัญลักษณ์	PSNR(dB)	WER
1	28.8008	0.000254
2	16.4203	0.004811
3	13.7364	0.009451
4	12.3815	0.013805
5	11.4732	0.017996
10	10.5295	0.032808

จากตารางที่ 2 เมื่อพิจารณาค่าเฉลี่ย PSNR ในกรณีที่การสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์เพิ่มขึ้น มีผลทำให้ค่าเฉลี่ย PSNR ลดลง และ ค่าเฉลี่ย WER เพิ่มขึ้น จากผลการทดลอง พบว่า ค่าแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์เพิ่มขึ้น ทำให้จำนวนความผิดพลาดของสัญลักษณ์เพิ่มขึ้น และ ประสิทธิภาพในการนำภาพกลับมีค่าลดลง การทดสอบช่องสัญญาณในกรณีนี้อาจทำให้สัญลักษณ์ที่อยู่ติดกันเกิดความผิดพลาดแบบต่อเนื่อง เมื่อไม่มีการสลับลำดับสัญลักษณ์ การถอดรหัสวีเทอร์บีจึงมีประสิทธิภาพในการนำภาพกลับลดลง

2. การทดสอบความสำคัญของสัญลักษณ์ LFS กับระบบส่งข้อมูลภาพ ในช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า ที่ γ = -23 dB และ $\bar{\gamma}_b$ = 5 dB กรณีที่ไม่มีการสลับลำดับสัญลักษณ์โดยใช้การถอดรหัสแบบ ML

ตารางที่ 3 เป็นผลการทดสอบช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า ที่ γ = -23 dB และ $\bar{\gamma}_b$ = 5 dB โดยกำหนดการสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ M สัญลักษณ์ ซึ่งมี 2 กรณี คือ กรณีที่การสุ่มค่าของแอลฟา 1 ค่า มีผลต่อ 1 สัญลักษณ์ เป็นการทดสอบกับช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ กรณีที่การสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ = 2, 3, 4, 5 และ 10 เป็นการทดสอบกับช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ในกรณีที่ไม่มีการสลับลำดับสัญลักษณ์ ใช้การถอดรหัส ML แสดงค่าเฉลี่ย PSNR และ WER ของระบบที่เกิดการสูญหายของสัญลักษณ์ LFS และ HFS จากคอลัมน์แรก และ คอลัมน์สุดท้ายตามลำดับ ในระบบที่ไม่มีสัญญาณรบกวนการสูญเสียสัญลักษณ์ LFS มีค่า PNRS = 30.3537 dB และ เมื่อมีการสูญเสียสัญลักษณ์ HFS มีค่า PSNR = 19.3665 dB

ตารางที่ 3 ค่าเฉลี่ย PSNR ของการส่งภาพผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า โดยกำหนดค่าของแอลฟา 1 ค่ามีผลต่อสัญลักษณ์ M สัญลักษณ์ M = 1, 2, 3, 4, 5 และ 10 ที่ γ = -23 dB และ $\bar{\gamma}_b$ = 5 dB ไม่มีการสลับลำดับสัญลักษณ์ ใช้การถอดรหัสแบบ ML ในกรณีที่เกิดการสูญหายของสัญลักษณ์ LFS และ HFS

ช่วงของสัญลักษณ์	การสูญหายสัญลักษณ์ LFS PSNR(dB) (WER)	การสูญหายสัญลักษณ์ HFS PSNR(dB) (WER$\times 10^{-4}$)
1	22.8928 (0.712512)	18.5127 (0.26)
2	16.3009 (0.712625)	15.4719 (1.3)
3	13.9395 (0.712687)	13.9474 (2.4)
4	12.3240 (0.712780)	12.4977 (3.8)
5	11.4266 (0.712831)	11.4994 (4.6)
10	10.6779 (0.713247)	10.6419 (8.9)

จากตารางที่ 3 ค่าแอลฟา 1 ค่า ต่อช่วงของสัญลักษณ์ที่เพิ่มขึ้น กรณีที่เกิดการสูญหายของสัญลักษณ์ LFS ทำให้สัญลักษณ์ HFS ถูกทำลาย ค่าเฉลี่ย PSNR ลดลง ค่าเฉลี่ย WER เปลี่ยนแปลงน้อยมาก จากผลการทดลอง พบว่า การที่สัญลักษณ์ LFS ถูกทำลายมีผลต่อประสิทธิภาพของระบบไม่ทำให้จำนวนความผิดพลาดของสัญลักษณ์เพิ่มขึ้น เนื่องจากจำนวนสัญลักษณ์ LFS มีจำนวนน้อยมากเมื่อเทียบกับจำนวนสัญลักษณ์ HFS

3. ผลการทดลองการสลับลำดับสัญลักษณ์แบบบล็อกบนช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็วมีการกระจายแบบไรเชียล

ตารางที่ 4 และ 5 เป็นผลการทดลองที่ได้จากการสลับลำดับสัญลักษณ์แบบบล็อก โดยนำมาใช้เป็นผลการทดลองอ้างอิง เพื่อเปรียบเทียบประสิทธิภาพกับระบบที่มีการสลับลำดับสัญลักษณ์แบบสุ่ม โดยส่งรูปภาพ Lena จำนวน 250 รูป ผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็วมีการกระจายแบบไรเชียล โดยกำหนดการสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ 1 สัญลักษณ์ ที่ γ = -43 dB และ -23 dB พิจารณาที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB ใช้การถอดรหัสแบบ ML ค่าเฉลี่ย PSNR และ WER ของคอลัมน์แรกได้จากระบบที่ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด คอลัมน์ถัดไปได้จากระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS และคอลัมน์สุดท้ายได้จากระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด

จากตารางที่ 4 และ 5 เมื่อพิจารณาค่าเฉลี่ย PSNR ของคอลัมน์สุดท้ายมีค่าลดลงเล็กน้อยเมื่อเปรียบเทียบกับคอลัมน์แรก และคอลัมน์ที่สอง เมื่อพิจารณาค่าเฉลี่ย WER ของคอลัมน์สุดท้ายเปรียบเทียบกับคอลัมน์แรก และคอลัมน์ที่สอง มีค่าเพิ่มขึ้นเล็กน้อย จากผลการทดลอง พบว่า การนำการสลับลำดับสัญลักษณ์แบบบล็อก ผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดแบบเร็วมาใช้กับระบบส่งข้อมูลภาพ มีผลทำให้จำนวนความผิดพลาดของสัญลักษณ์เพิ่มขึ้น และ ประสิทธิภาพในการนำภาพกลับมีค่าลดลง

ตารางที่ 4 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับโดยใช้การถอดรหัสแบบ ML ในช่องสัญญาณที่มีการเลือนหายทางขนาดแบบเร็ว โดยที่ γ = -43dB

$\bar{\gamma}_b$ (dB)	PSNR(dB) ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-3})	PSNR(dB) ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-3})	PSNR(dB) ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-3})
3.75	23.54	23.26	22.97
	(1.5)	(1.6)	(1.6)
5.00	28.09	27.85	27.17
	(0.41)	(0.46)	(0.47)
6.25	30.08	29.92	29.74
	(0.0147)	(0.0167)	(0.0168)

ที่มา: Mahapakulchai and Thongnumpen (2007)

ตารางที่ 5 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับโดยใช้การถอดรหัสแบบ ML ในช่องสัญญาณที่มีการเลือนหายทางขนาดแบบเร็ว โดยที่ γ = -23dB

$\bar{\gamma}_b$ (dB)	PSNR(dB) ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-3})	PSNR(dB) ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-3})	PSNR(dB) ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-3})
2.50	20.82	20.67	20.02
	(2.7)	(3)	(3)
3.75	26.67	26.44	25.79
	(0.67)	(0.73)	(0.74)
5.00	29.61	29.43	29.18
	(0.21)	(0.23)	(0.23)

ที่มา: Mahapakulchai and Thongnumpen (2007)

4. ผลการทดลองการสลับลำดับสัญลักษณ์แบบสุ่มบนช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็วมีการกระจายแบบไรเชียล ในกรณีที่สร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence เพื่อสร้างเมตริกซ์ P

ตารางที่ 6, 7 และ 8 เป็นผลการทดลองจากการสลับลำดับสัญลักษณ์แบบสุ่ม ที่สร้างตัวเลขแบบสุ่มโดยอัลกอริทึม Linear Congruence เพื่อสร้างเมตริกซ์เอกลักษณ์แบบสุ่ม P ส่งรูปภาพ Lena จำนวน 300 รูป ผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว โดยกำหนดการสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ 1 สัญลักษณ์ ที่ γ = -43 dB, -33 dB และ -23 dB พิจารณาที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB โดยใช้การถอดรหัสแบบ ML ค่าเฉลี่ย PSNR และ WER ของคอลัมน์แรกได้จากระบบที่ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด คอลัมน์ถัดไปได้จากระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS และคอลัมน์สุดท้ายได้จากระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด

ตารางที่ 6 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ โดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence เพื่อสร้างเมตริกซ์ P ในระบบส่งข้อมูลภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีเลือนหายทางขนาดของสัญญาณแบบเร็ว ที่ γ = -43dB

$\bar{\gamma}_b$ (dB)	PSNR(dB) ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-3})	PSNR(dB) ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-3})	PSNR(dB) ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-3})
3.75	20.6283	20.6183	20.6296
	(2.214)	(2.205)	(2.279)
5.00	25.4973	25.4303	25.5932
	(0.6713)	(0.6838)	(0.6809)
6.25	28.9896	29.0085	28.5821
	(0.2326)	(0.2306)	(0.2280)

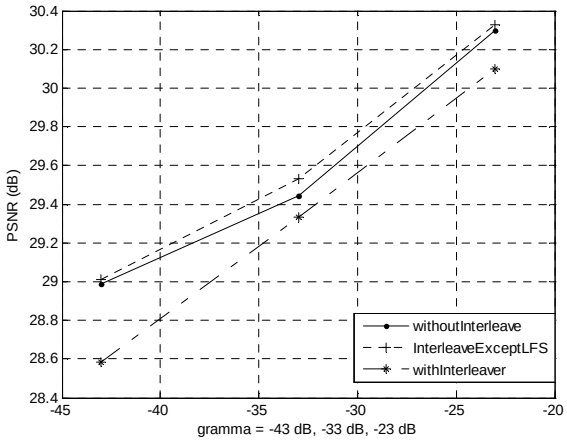
ตารางที่ 7 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ โดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence เพื่อสร้างเมตริกซ์ P ในระบบส่งข้อมูลภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีเลือนหายทางขนาดของสัญญาณแบบเร็ว ที่ γ = -33dB

$\bar{\gamma}_b$ (dB)	PSNR(dB) ไม่ใช้การสลับลำดับ กลุ่มสัญลักษณ์ทั้งหมด (WER$\times 10^{-3}$)	PSNR(dB) ใช้การสลับลำดับกลุ่ม สัญลักษณ์ HFS (WER$\times 10^{-3}$)	PSNR(dB) ใช้การสลับลำดับ กลุ่มสัญลักษณ์ทั้งหมด (WER)
3.75	21.7046	21.7569	21.257
	(1.734)	(1.712)	(1.721)
5.00	26.5760	26.4498	26.6163
	(0.5229)	(0.513)	(0.513)
6.25	29.4453	29.5295	29.3304
	(0.1822)	(0.1765)	(0.1824)

ตารางที่ 8 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ โดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence เพื่อสร้างเมตริกซ์ P ในระบบส่งข้อมูลภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีเลือนหายทางขนาดของสัญญาณแบบเร็ว ที่ γ = -23dB

$\bar{\gamma}_b$ (dB)	PSNR(dB) ไม่ใช้การสลับลำดับ กลุ่มสัญลักษณ์ (WER$\times 10^{-4}$)	PSNR(dB) ใช้การสลับลำดับกลุ่ม สัญลักษณ์ HFS (WER$\times 10^{-4}$)	PSNR(dB) ใช้การสลับลำดับ กลุ่มสัญลักษณ์ (WER$\times 10^{-4}$)
3.75	24.9629	24.9578	24.9154
	(8.197)	(8.082)	(7.95)
5.00	28.8008	28.7534	28.4569
	(2.537)	(2.521)	(2.599)
6.25	30.2969	30.3269	30.0979
	(1.060)	(1.052)	(1.059)

จากตารางที่ 6, 7 และ 8 เมื่อพิจารณาค่าเฉลี่ย PSNR คอลัมน์สุดท้าย กับ คอลัมน์แรก จากตารางที่ 6 และ 7 ที่ $\bar{\gamma}_b$ = 5 dB มีค่าเพิ่มขึ้นเล็กน้อย ค่าเฉลี่ย PSNR อยู่ที่ 25-26 dB และเมื่อเปรียบเทียบตารางที่ 6 และ 8 ของคอลัมน์ที่สอง กับคอลัมน์แรก ที่ $\bar{\gamma}_b$ = 6.25 dB มีค่าเพิ่มขึ้นเล็กน้อย ค่าเฉลี่ย PSNR อยู่ที่ 28-29 dB จากนั้นเมื่อพิจารณาค่าเฉลี่ย WER ของตารางที่ 6 ของ $\bar{\gamma}_b$ = 6.25 dB มีค่าลดลงสามารถทำให้ลดจำนวนความผิดพลาดของสัญลักษณ์ได้ จากผลการทดลองพบว่า การนำการสลับลำดับสัญลักษณ์แบบสุ่มใช้กับช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว มีผลทำให้จำนวนความผิดพลาดของสัญลักษณ์ลดลง และ สามารถเพิ่มประสิทธิภาพในการนำภาพกลับ ซึ่งเป็นการเปลี่ยนแปลงค่าเพียงเล็กน้อยให้กับระบบส่งภาพ ในบางรูปแบบของสัญญาณรบกวน ใช้เวลาในการประมวลผลประมาณ 43.73 วินาทีต่อภาพ



ภาพที่ 18 กราฟแสดงค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับสัญลักษณ์แบบสุ่ม โดยสร้างเมตริกซ์ P ในระบบส่งภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว ที่ $\bar{\gamma}_b$ = 6.25 dB ระหว่าง γ = -43, -33 และ -23 dB

จากกราฟเป็นการเปรียบเทียบค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับแบบสุ่ม เพื่อสร้างเมตริกซ์ P ในระบบใช้การถอดรหัสแบบ ML ที่ $\bar{\gamma}_b$ = 6.25 dB ระหว่าง γ = -43, -33, และ -23 dB พิจารณาประสิทธิภาพของระบบ 3 กรณี คือ กรณีแรกแสดงถึงประสิทธิภาพของระบบที่ไม่ใช้การสลับลำดับแบบสุ่มทั้งหมด ค่าเฉลี่ย PSNR = 28.9896 dB, 29.4453 dB และ 30.2969 dB กรณีที่สองแสดงถึงประสิทธิภาพของระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS ค่าเฉลี่ย PSNR = 29.0085 dB, 29.5295 dB และ 30.3269 dB และกรณีสุดท้าย คือ การใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ค่าเฉลี่ย PSNR = 28.5821 dB, 29.3304 dB และ 30.0979 dB

(ก)

(ข)

ภาพที่ 19 การนำรูปภาพ Lena กลับโดยใช้การถอดรหัสแบบ ML (ก) $\bar{\gamma}_b$ = 3.75dB ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ประสิทธิภาพของระบบ PSNR = 26.44 dB WER = 0.00088 (ข) $\bar{\gamma}_b$ = 3.75 dB ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมดโดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence เพื่อสร้างเมตริกซ์ P ประสิทธิภาพของระบบ PSNR = 28.75 dB WER = 0.00053

จากภาพที่ 19 (ก) และ (ข) เป็นการเปรียบเทียบการนำรูปภาพ Lena กลับ ระหว่างระบบที่ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด กับ ระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมดที่มี $\bar{\gamma}_b$ = 3.75 dB และ γ = -23 dB สามารถเห็นความแตกต่างของภาพที่ได้ทั้ง 2 แบบ ที่ทำให้ประสิทธิภาพของภาพ เพิ่มขึ้นประมาณ 2.31dB ดังภาพที่ 15 (ข) จะเห็นความแตกต่างได้อย่างชัดเจนบริเวณรูปหน้าของภาพ

5. ผลการทดลองการสลับลำดับสัญลักษณ์แบบสุ่มบนช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็วมีการกระจายแบบไรเชียล ในกรณีที่สร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence โดยไม่สร้างเมตริกซ์ P

ตารางที่ 9, 10 และ 11 เป็นผลการทดลองจากการสลับลำดับกลุ่มสัญลักษณ์แบบสุ่ม ที่สร้างตัวเลขแบบสุ่มโดยอัลกอริทึม Linear Congruence เพื่อนำลำดับของตัวเลขที่ทำการสุ่มได้ เป็นตัวเลขที่ชี้ตำแหน่งสัญลักษณ์เริ่มต้นไปยังตำแหน่งของการสลับลำดับสัญลักษณ์ โดยส่งรูปภาพ Lena จำนวน 200 รูป ผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว โดยกำหนดการสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ 1 สัญลักษณ์ ที่ γ = -43 dB, -33 dB และ -23 dB พิจารณาที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB ค่าเฉลี่ย PSNR และ WER ของคอลัมน์แรกได้จากระบบที่มีการใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS และคอลัมน์สุดท้ายได้จากระบบที่มีการใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ใช้การถอดรหัสแบบ ML และ MAP

ตารางที่ 9 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ โดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence ไม่สร้างเมตริกซ์ P ในระบบส่งภาพใช้การถอดรหัสแบบ ML และ MAP และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว ที่ γ = -43 dB

$\bar{\gamma}_b$ (dB)	PSNR(dB) การสลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-3})		PSNR(dB) การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-3})	
	ถอดรหัสแบบ ML	ถอดรหัสแบบ MAP	ถอดรหัสแบบ ML	ถอดรหัสแบบ MAP
3.75	20.63	20.51	20.52	20.39
	(2.2)	(2.6)	(2.2)	(2.6)
5.00	25.49	25.41	25.82	25.74
	(0.65)	(0.73)	(0.65)	(0.73)
6.25	28.99	28.87	28.75	28.67
	(0.22)	(0.25)	(0.22)	(0.25)

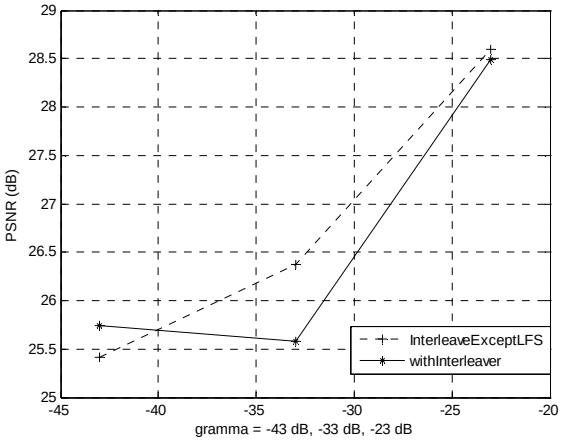
ตารางที่ 10 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ โดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence ไม่สร้างเมตริกซ์ P ในระบบส่งภาพใช้การถอดรหัสแบบ ML และ MAP และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว ที่ γ = -33 dB

$\overline{\gamma}_b$ (dB)	PSNR(dB) การสลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-3})		PSNR(dB) การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-3})	
	ถอดรหัสแบบ ML	ถอดรหัสแบบ MAP	ถอดรหัสแบบ ML	ถอดรหัสแบบ MAP
3.75	21.82	21.61	21.73	21.50
	(1.7)	(2)	(1.7)	(2)
5.00	26.53	26.38	26.72	25.58
	(0.5)	(0.58)	(0.49)	(0.58)
6.25	29.45	29.34	29.29	29.19
	(0.17)	(0.2)	(0.17)	(0.2)

ตารางที่ 11 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ โดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence ไม่สร้างเมตริกซ์ P ในระบบส่งภาพใช้การถอดรหัสแบบ ML และ MAP และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว ที่ γ = -23 dB

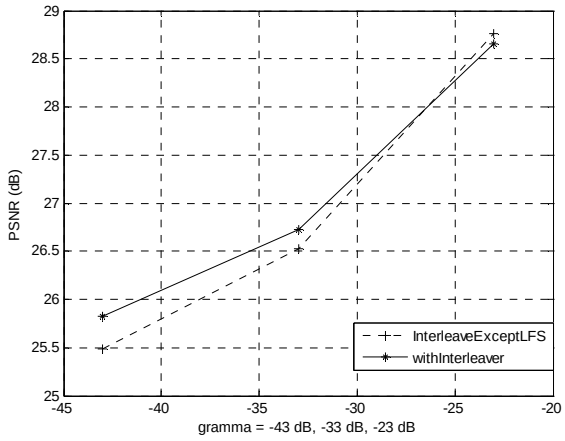
$\overline{\gamma}_b$ (dB)	PSNR(dB) การสลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-3})		PSNR(dB) การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-3})	
	ถอดรหัสแบบ ML	ถอดรหัสแบบ MAP	ถอดรหัสแบบ ML	MAP decoding
3.75	25.004	24.88	25.39	25.24
	(0.79)	(0.9)	(0.78)	(0.9)
5.00	28.76	28.60	28.65	28.49
	(0.24)	(0.28)	(0.23)	(0.27)
6.25	30.56	30.47	30.47	30.39
	(0.08)	(0.1)	(0.09)	(0.1)

จากตารางที่ 9, 10 และ 11 เปรียบเทียบการถอดรหัสแบบ ML และ MAP เมื่อพิจารณาค่าเฉลี่ย PSNR พบว่า ในทุกกรณีที่ทำการทดลอง การถอดรหัสแบบ MAP มีค่าน้อยกว่าการถอดรหัสแบบ ML เมื่อพิจารณาค่าเฉลี่ย WER ในทุกกรณีที่ทำการทดลอง การถอดรหัสแบบ MAP มีค่ามากกว่าการถอดรหัสแบบ ML จากผลการทดลอง พบว่า การนำการสลับลำดับกลุ่มสัญลักษณ์แบบสุ่ม ผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็วมาใช้กับระบบส่งภาพ กรณีนี้ เมื่อวิเคราะห์จากค่าเฉลี่ย PSNR มีผลทำให้ประสิทธิภาพของการถอดรหัสแบบ ML ดีกว่า MAP ซึ่งไม่สอดคล้องกับทฤษฎี จากผลการทดลองอาจเป็นไปได้ว่า หลังจากการสลับลำดับกลุ่มสัญลักษณ์กลับ ค่าความน่าจะเป็นของสัญลักษณ์ซีโร่ทรี ไม่สามารถจับคู่กันเหมือนเดิม เนื่องจากก่อนการคำนวณหาเส้นทางบนเทลลิสตำแหน่งของสัญลักษณ์มีการเปลี่ยนแปลงจากการสลับลำดับกลุ่มสัญลักษณ์



ภาพที่ 20 กราฟแสดงค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับสัญลักษณ์แบบสุ่ม โดยไม่สร้างเมตริกซ์ P ในระบบส่งข้อมูลภาพใช้การถอดรหัสแบบ MAP และช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว ที่ $\bar{\gamma}_b$ = 5 dB ระหว่าง γ = -43, -33 และ -23 dB

จากกราฟเป็นการเปรียบเทียบค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับแบบสุ่ม โดยไม่สร้างเมตริกซ์ P ในระบบใช้การถอดรหัสแบบ MAP ที่ $\bar{\gamma}_b$ = 5 dB ระหว่าง γ = -43, -33 และ -23 dB พิจารณาประสิทธิภาพของระบบ 2 กรณี คือ กรณีแรกแสดงถึงประสิทธิภาพของระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS ค่าเฉลี่ย PSNR = 25.41 dB, 26.38 dB และ 28.60 dB และกรณีสุดท้าย คือ การใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ค่าเฉลี่ย PSNR = 25.74 dB, 25.58 dB และ 28.49 dB



ภาพที่ 21 กราฟแสดงค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับแบบสัญลักษณ์แบบสุ่ม โดยไม่สร้างเมตริกซ์ P ในระบบส่งภาพใช้การถอดรหัสแบบ ML และช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว ที่ $\bar{\gamma}_b$ = 5 dB ระหว่าง γ = -43, -33 และ -23 dB

จากกราฟเป็นการเปรียบเทียบค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับแบบสุ่ม โดยไม่สร้างเมตริกซ์ P ในระบบใช้การถอดรหัสแบบ ML ที่ $\bar{\gamma}_b$ = 5 dB ระหว่าง γ = -43, -33 และ -23 dB พิจารณาประสิทธิภาพของระบบ 2 กรณี คือ กรณีแรกแสดงถึงประสิทธิภาพของระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS ค่าเฉลี่ย PSNR = 25.49 dB, 26.53 dB และ 28.76 dB และกรณีสุดท้าย คือ การใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ค่าเฉลี่ย PSNR = 25.82 dB, 26.72 dB และ 28.65 dB

(ก)

(ข)

ภาพที่ 22 การนำรูปภาพ Lena กลับโดยใช้การสลับลำดับสัญลักษณ์แบบสุ่ม ไม่สร้างเมตริกซ์ P ก) $\bar{\gamma}_b$ = 5 dB ใช้การถอดรหัสแบบ ML ประสิทธิภาพของระบบ PSNR = 27.28 dB WER = 0.00038 ข) $\bar{\gamma}_b$ = 5 dB ใช้การถอดรหัสแบบ MAP ประสิทธิภาพของระบบ PSNR = 28.51dB WER = 0.00032

จากภาพที่ 22 (ก) และ (ข) เป็นการเปรียบเทียบรูปภาพ Lena ของการนำภาพกลับ ระหว่างการถอดรหัสแบบ ML และ MAP ตามลำดับ ที่มี $\bar{\gamma}_b$ = 5 dB และ γ = -23 dB ความแตกต่างของภาพที่ได้รับอยู่บริเวณหมวกและไหล่ของภาพ

6. ผลการทดลองการสลับลำดับสัญลักษณ์แบบสุ่มบนช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้ามีการกระจายแบบไรเชียลในกรณีที่สร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence โดยไม่สร้างเมตริกซ์ P

ตารางที่ 12, 13 และ 14 เป็นผลการทดลองจากการสลับลำดับสัญลักษณ์แบบสุ่ม ที่สร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence เพื่อนำลำดับของตัวเลขที่ทำการสุ่มได้ เป็นตัวเลขที่ชี้ตำแหน่งสัญลักษณ์เริ่มต้นไปยังตำแหน่งของการสลับลำดับสัญลักษณ์ โดยส่งรูปภาพ Lena จำนวน 200 รูป ผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า กำหนดให้การสุ่มค่าของแอลฟา 1 ค่ามีผลต่อช่วงของสัญลักษณ์ 5 สัญลักษณ์ ที่ γ = -43 dB, -33 dB และ -23 dB พิจารณาที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB โดยการถอดรหัสแบบ ML ค่าเฉลี่ย PSNR และ WER ของคอลัมน์แรกได้จากระบบที่ไม่มีการสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด คอลัมน์ที่สองได้จากระบบที่มีการสลับลำดับกลุ่มสัญลักษณ์ HFS คอลัมน์ที่สามได้จากระบบที่มีการสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด และคอลัมน์สุดท้าย คือ ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ เป็นผลต่างระหว่างคอลัมน์ที่สามและคอลัมน์แรก

ตารางที่ 12 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ โดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence ไม่สร้างเมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -43 dB

$\bar{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS ($WER \times 10^{-2}$)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ($WER \times 10^{-2}$)	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	8.7952	9.1166	20.7836	11.9885
	(0.0481)	(0.43)	(0.31)	
5.00	9.6469	10.1846	25.6567	16.0098
	(0.0319)	(0.17)	(0.08)	
6.25	10.7828	11.6285	29.0810	18.2982
	(0.0215)	(0.08)	(0.02)	

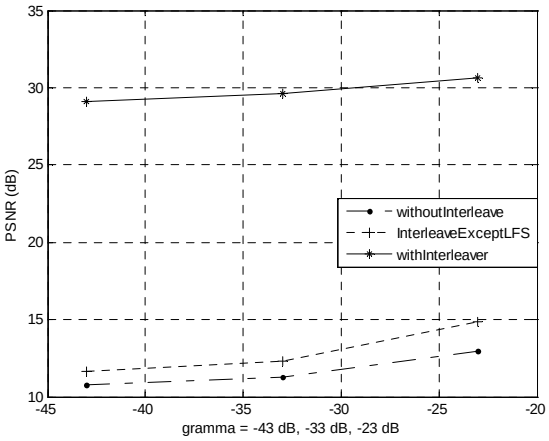
ตารางที่ 13 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ โดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence ไม่ใช้เมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -33 dB

$\overline{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	8.9992	9.3758	21.8371	12.8379
	(0.0431)	(0.34)	(0.23)	
5.00	10.0141	10.6269	26.5801	16.5660
	(0.0281)	(0.14)	(0.06)	
6.25	11.2581	12.2883	29.6165	18.3583
	(0.0184)	(0.06)	(0.02)	

ตารางที่ 14 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ โดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence ไม่ใช้เมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -23 dB

$\overline{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	9.9578	10.5214	25.0229	15.0652
	(0.0296)	(0.18)	(0.11)	
5.00	11.3875	12.4585	29.02486	17.6373
	(0.0180)	(0.07)	(0.03)	
6.25	12.9878	14.8342	30.6197	17.6320
	(0.0109)	(0.04)	(0.01)	

จากตารางที่ 12, 13 และ 14 เมื่อพิจารณาค่าเฉลี่ย PSNR ที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB เปรียบเทียบระหว่างคอลัมน์ที่สองกับคอลัมน์แรก มีค่าแตกต่างกันเล็กน้อย และคอลัมน์ที่สามเปรียบเทียบกับคอลัมน์แรกมีค่าแตกต่างกันประมาณ 11-17 dB เมื่อพิจารณาค่าเฉลี่ย WER คอลัมน์ที่สองกับคอลัมน์แรก มีค่าแตกต่างกัน ในขณะที่ค่าเฉลี่ย PSNR แตกต่างกันเพียงเล็กน้อย และคอลัมน์ที่สามเปรียบเทียบกับคอลัมน์ที่สอง มีค่าแตกต่างกันเล็กน้อย ในขณะที่ค่าเฉลี่ย PSNR ต่างกันประมาณ 11-18 dB อาจเนื่องจากกลุ่มสัญลักษณ์ LFS ถูกทำลายอย่างมาก จากผลการทดลองพบว่า การนำการสลับลำดับสัญลักษณ์แบบสุ่ม มาใช้กับช่องสัญญาณช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า มีผลทำให้จำนวนความผิดพลาดของสัญลักษณ์ลดลง และ สามารถเพิ่มประสิทธิภาพในการนำภาพกลับ ใช้เวลาในการประมวลผลประมาณ 4.0697 วินาทีต่อภาพ



ภาพที่ 23 กราฟแสดงค่าเฉลี่ย PSNR ของการนำภาพกลับ ใช้การสลับลำดับสัญลักษณ์แบบสุ่ม โดยไม่สร้างเมตริกซ์ P ในระบบส่งภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ $\bar{\gamma}_b$ = 6.25 dB ระหว่าง γ = -43, -33 และ -23 dB

จากกราฟเป็นการเปรียบเทียบค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับแบบสุ่ม เพื่อสร้างเมตริกซ์ P ในระบบใช้การถอดรหัสแบบ ML ที่ $\bar{\gamma}_b$ = 6.25 dB ระหว่าง γ = -43, -33 และ -23 dB พิจารณาประสิทธิภาพของระบบ 3 กรณี คือ กรณีแรกแสดงถึงประสิทธิภาพของระบบที่ไม่ใช้การสลับลำดับแบบสุ่มทั้งหมด ค่าเฉลี่ย PSNR = 10.7828 dB, 11.2581 dB และ 12.9878 dB กรณีที่สองแสดงถึงประสิทธิภาพของระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS ค่าเฉลี่ย PSNR = 11.6285 dB, 12.2883 dB และ 14.8342 dB และกรณีสุดท้าย คือ การใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ค่าเฉลี่ย PSNR = 29.0810 dB, 29.6165 dB และ 30.6197 dB

(ก)

(ข)

ภาพที่ 24 การนำรูปภาพ Lena กลับ โดยสร้างตัวเลขแบบสุ่มจากอัลกอริทึม Linear Congruence ไม่สร้างเมตริกซ์ P โดยใช้การถอดรหัสแบบ ML ก) $\bar{\gamma}_b$ = 5 dB ไม่ใช้การสลับลำดับสัญลักษณ์ทั้งหมด ประสิทธิภาพของระบบ PSNR = 12.40 dB WER = 0.01447 ข) $\bar{\gamma}_b$ = 5 dB ใช้การสลับลำดับสัญลักษณ์ทั้งหมด ประสิทธิภาพของระบบ PSNR = 31.36 dB WER = 0.00006

จากภาพที่ 24 (ก) และ (ข) เป็นการเปรียบเทียบรูปภาพ Lena ของการนำภาพกลับ โดยใช้การถอดรหัสแบบ ML ที่มี $\bar{\gamma}_b$ = 5 dB และ γ = -23 dB ความแตกต่างของภาพที่ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด เห็นเป็นภาพลางๆ เป็นโครงสร้างของภาพไม่สามารถเก็บรายละเอียดของภาพได้ เห็นความแตกต่างของภาพที่ได้รับอย่างชัดเจน

7. ผลการทดลองการสลับลำดับสัญลักษณ์แบบสุ่มบนช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้ามีการกระจายแบบไรเชียล ในกรณีที่สร้างตัวเลขแบบสุ่มโดยเรียกใช้ฟังก์ชันจากโปรแกรมภาษาจาวา โดยไม่สร้างเมตริกซ์ P

ตารางที่ 15, 16 และ 17 เป็นผลการทดลองจากการสลับลำดับสัญลักษณ์แบบสุ่ม ที่สร้างตัวเลขแบบสุ่มโดยเรียกใช้ฟังก์ชันสุ่มจากโปรแกรมภาษาจาวา เพื่อนำลำดับของตัวเลขที่ทำการสุ่มได้ เป็นตัวเลขที่ชี้ตำแหน่งสัญลักษณ์เริ่มต้นไปยังตำแหน่งของการสลับลำดับสัญลักษณ์ โดยส่งรูปภาพ Lena จำนวน 200 รูป ผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า กำหนดให้การสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ 5 สัญลักษณ์ ที่ γ = -43 dB, -33 dB และ -23 dB พิจารณาที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB โดยใช้การถอดรหัสแบบ ML ค่าเฉลี่ย PSNR และ WER ของคอลัมน์แรกได้จากระบบที่ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด คอลัมน์ที่สองได้จากระบบที่มีการสลับลำดับกลุ่มสัญลักษณ์ HFS คอลัมน์ที่สามได้จากระบบที่มีการสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด และคอลัมน์สุดท้าย คือ ประสิทธิภาพที่เพิ่มขึ้น เป็นผลต่างระหว่างคอลัมน์ที่สามและคอลัมน์แรก

ตารางที่ 15 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ สร้างตัวเลขแบบสุ่มโดยใช้ฟังก์ชันสุ่มจากโปรแกรมภาษาจาวา ไม่สร้างเมตริกซ์ P ระบบส่งข้อมูลภาพภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -43 dB

$\bar{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดักลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	8.7952	9.1003	20.4943	11.6991
	(0.0481)	(0.39)	(0.27)	
5.00	9.6469	10.17601	25.4260	15.779
	(0.0319)	(0.17)	(0.08)	
6.25	10.7828	11.6179	28.6212	17.8384
	(0.0215)	(0.08)	(0.03)	

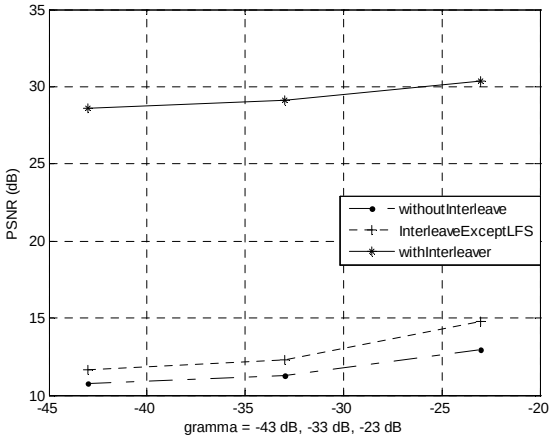
ตารางที่ 16 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ สร้างตัวเลขแบบสุ่มโดยใช้ฟังก์ชันสุ่มจากโปรแกรมภาษาจาวา ไม่สร้างเมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -33 dB

$\overline{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	8.9992	9.3547	21.4388	12.4396
	(0.0431)	(0.32)	(0.21)	
5.00	10.0141	10.6184	26.3377	16.3226
	(0.0281)	(0.14)	(0.06)	
6.25	11.2581	12.2818	29.1364	17.8783
	(0.0184)	(0.07)	(0.02)	

ตารางที่ 17 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ สร้างตัวเลขแบบสุ่มโดยใช้ฟังก์ชันสุ่มจากโปรแกรมภาษาจาวา ไม่สร้างเมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -23 dB

$\overline{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	9.9578	10.5091	24.6456	14.688
	(0.0296)	(0.17)	(0.09)	
5.00	11.3875	12.4500	28.5182	17.1307
	(0.0180)	(0.07)	(0.03)	
6.25	12.9878	14.7826	30.3307	17.3429
	(0.0109)	(0.04)	(0.01)	

จากตารางที่ 15, 16 และ 17 เมื่อพิจารณาค่าเฉลี่ย PSNR ที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB เปรียบเทียบระหว่างคอลัมน์ที่สองกับคอลัมน์แรก มีค่าแตกต่างกันเล็กน้อย และคอลัมน์ที่สามเปรียบเทียบกับคอลัมน์แรกมีค่าแตกต่างกันประมาณ 11-17 dB เมื่อพิจารณาค่าเฉลี่ย WER คอลัมน์ที่สองกับคอลัมน์แรก มีค่าแตกต่างกัน ในขณะที่ค่าเฉลี่ย PSNR แตกต่างกันเพียงเล็กน้อย และคอลัมน์ที่สามเปรียบเทียบกับคอลัมน์ที่สอง มีค่าแตกต่างกันเล็กน้อย ในขณะที่ค่าเฉลี่ย PSNR ต่างกันประมาณ 11-17 dB ซึ่งอาจเนื่องจากสัญลักษณ์ LFS ถูกทำลายอย่างมาก จากผลการทดลองพบว่า การนำการสลับลำดับสัญลักษณ์แบบสุ่ม มาใช้กับช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า มีผลทำให้จำนวนความผิดพลาดของสัญลักษณ์ลดลง และ สามารถเพิ่มประสิทธิภาพในการนำภาพกลับ ใช้เวลาในการประมวลผลประมาณ 4.2406 วินาทีต่อภาพ



ภาพที่ 25 กราฟแสดงค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับสัญลักษณ์แบบสุ่ม ไม่สร้างเมตริกซ์ P ในระบบส่งภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ $\bar{\gamma}_b$ = 6.25 dB ระหว่าง γ = -43, -33 และ -23 dB

จากกราฟเป็นการเปรียบเทียบค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับแบบสุ่ม เพื่อสร้างเมตริกซ์ P ในระบบใช้การถอดรหัสแบบ ML ที่ $\bar{\gamma}_b$ = 6.25 dB ระหว่าง γ = -43, -33 และ -23 dB พิจารณาประสิทธิภาพของระบบ 3 กรณี คือ กรณีแรกแสดงถึงประสิทธิภาพของระบบที่ไม่ใช้การสลับลำดับแบบสุ่มทั้งหมด ค่าเฉลี่ย PSNR = 10.7828 dB, 11.2581 dB และ 12.9878 dB กรณีที่สองแสดงถึงประสิทธิภาพของระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS ค่าเฉลี่ย PSNR = 11.6179 dB, 12.2818 dB และ 14.7826 dB และกรณีสุดท้าย คือ การใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ค่าเฉลี่ย PSNR = 28.6212 dB, 29.1364 dB และ 30.3307 dB

(ก)

(ข)

ภาพที่ 26 การนำรูปภาพ Lena กลับ ใช้การสลับลำดับแบบสุ่ม โดยฟังก์ชันสุ่มโปรแกรมภาษาจาวา ไม่สร้างเมตริกซ์ P ในระบบใช้การถอดรหัสแบบ ML ก) $\bar{\gamma}_b$ = 5dB ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ประสิทธิภาพ ของระบบ PSNR = 7.82 dB WER = 0.01729 ข) $\bar{\gamma}_b$ = 5dB ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ประสิทธิภาพของระบบ PSNR = 31.05 dB WER = 0.00018

จากภาพที่ 26 (ก) และ (ข) เป็นการเปรียบเทียบรูปภาพ Lena ของการนำภาพกลับ โดยใช้การถอดรหัสแบบ ML ที่มี $\bar{\gamma}_b$ = 5 dB และ γ = -23 dB ความแตกต่างของภาพที่ไม่ใช้กระบวนการสลับลำดับสัญลักษณ์ทั้งหมดไม่สามารถแสดงเค้าโครงและรายละเอียดของภาพได้ มีความแตกต่างของภาพที่ได้รับอย่างชัดเจน

8. ผลการทดลองการสลับลำดับสัญลักษณ์แบบกึ่งสุ่มบนช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้ามีการกระจายแบบไรเชียล ในกรณีที่สร้างตัวเลขแบบสุ่มโดยเรียกใช้ฟังก์ชันสุ่มจากโปรแกรมภาษาจาวา โดยไม่สร้างเมตริกซ์ P และ กำหนดเงื่อนไขในการตรวจสอบ S = 10

ตารางที่ 18, 19 และ 20 เป็นผลการทดลองจากการสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม ที่สร้างตัวเลขแบบสุ่มโดยเรียกใช้ฟังก์ชันสุ่มจากโปรแกรมภาษาจาวา โดยใช้โครงสร้างทรีในการกระจายตัวเลขสุ่ม และนำมาตรวจสอบเงื่อนไข S = 10 เพื่อนำลำดับของตัวเลขสุ่มที่ได้ เป็นตัวเลขที่ชี้ตำแหน่งสัญลักษณ์เริ่มต้นไปยังตำแหน่งของการสลับลำดับสัญลักษณ์ โดยส่งรูปภาพ Lena จำนวน 200 รูป ผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า กำหนดให้การสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ 5 สัญลักษณ์ ที่ γ = -43 dB, -33 dB และ -23 dB พิจารณาที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB โดยใช้การถอดรหัสแบบ ML ค่าเฉลี่ย PSNR และ WER ของคอลัมน์แรกได้จากระบบที่ไม่มีการสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด คอลัมน์ที่สองได้จากระบบที่มีการสลับลำดับกลุ่มสัญลักษณ์ HFS คอลัมน์ที่สามได้จากระบบที่มีการสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด และคอลัมน์สุดท้าย คือ ประสิทธิภาพที่เพิ่มขึ้น เป็นผลต่างระหว่างคอลัมน์ที่สามและคอลัมน์แรก

ตารางที่ 18 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ สร้างตัวเลขแบบกึ่งสุ่ม S = 10 โดยใช้ฟังก์ชันสุ่มโปรแกรมภาษาจาวา ไม่สร้างเมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -43 dB

$\bar{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	8.7952	9.1050	20.8681	12.0729
	(0.0481)	(0.40)	(0.27)	
5.00	9.6469	10.1775	25.6588	16.0118
	(0.0319)	(0.17)	(0.08)	
6.25	10.7828	11.6207	28.5324	17.7495
	(0.0215)	(0.08)	(0.03)	

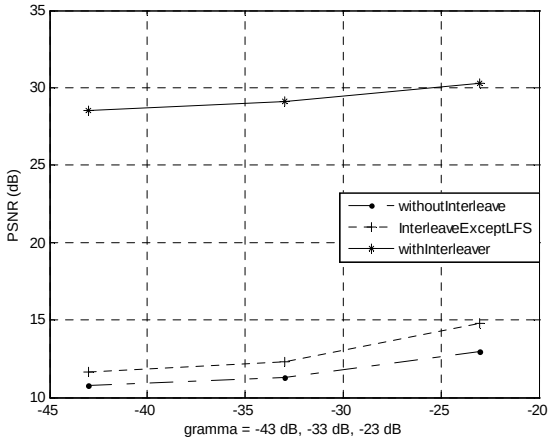
ตารางที่ 19 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ สร้างตัวเลขแบบกึ่งสุ่ม S = 10 โดยใช้ฟังก์ชันสุ่มโปรแกรมภาษาจาวา ไม่สร้างเมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -33 dB

$\overline{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	8.9992	9.3622	21.9662	12.9670
	(0.0431)	(0.32)	(0.21)	
5.00	10.0141	10.6187	26.4119	16.3979
	(0.0281)	(0.14)	(0.06)	
6.25	11.2581	12.2901	29.1239	17.8658
	(0.0184)	(0.07)	(0.02)	

ตารางที่ 20 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ สร้างตัวเลขแบบกึ่งสุ่ม S = 10 โดยใช้ฟังก์ชันสุ่มโปรแกรมภาษาจาวา ไม่ใช้เมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -23 dB

$\overline{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	9.9578	10.5146	25.1313	15.1735
	(0.0296)	(0.18)	(0.10)	
5.00	11.3875	12.4676	28.54889	17.1614
	(0.0180)	(0.07)	(0.03)	
6.25	12.9878	14.8084	30.2511	17.2633
	(0.0109)	(0.04)	(0.01)	

จากตารางที่ 18, 19 และ 20 เมื่อพิจารณาค่าเฉลี่ย PSNR ที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB เปรียบเทียบระหว่างคอลัมน์ที่สองกับคอลัมน์แรก มีค่าแตกต่างกันเล็กน้อย และคอลัมน์ที่สามเปรียบเทียบกับคอลัมน์แรกมีค่าแตกต่างกันประมาณ 12-17 dB เมื่อพิจารณาค่าเฉลี่ย WER คอลัมน์ที่สองกับคอลัมน์แรก มีค่าแตกต่างกัน ในขณะที่ค่าเฉลี่ย PSNR แตกต่างกันเพียงเล็กน้อย และคอลัมน์ที่สามเปรียบเทียบกับคอลัมน์ที่สอง มีค่าแตกต่างกันเล็กน้อย ในขณะที่ค่าเฉลี่ย PSNR ต่างกันประมาณ 12-17 dB ซึ่งอาจเนื่องจากสัญลักษณ์ LFS ถูกทำลายอย่างมาก จากผลการทดลองพบว่า การนำการสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม มาใช้กับช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า มีผลทำให้จำนวนความผิดพลาดของสัญลักษณ์ลดลง และ สามารถเพิ่มประสิทธิภาพในการนำภาพกลับ ใช้เวลาในการประมวลผลภาพประมาณ 4.38 วินาทีต่อภาพ



ภาพที่ 27 กราฟแสดงค่าเฉลี่ย PSNR ของการนำภาพกลับ ใช้การสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม S = 10 ไม่สร้างเมตริกซ์ P ในระบบส่งข้อมูลภาพใช้การถอดรหัสแบบ ML และช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ $\bar{\gamma}_b$ = 6.25 dB ระหว่าง γ = -43, -33 และ -23 dB

จากกราฟเป็นการเปรียบเทียบค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับแบบกึ่งสุ่ม เพื่อสร้างเมตริกซ์ P ในระบบใช้การถอดรหัสแบบ ML ที่ $\bar{\gamma}_b$ = 6.25 dB ระหว่าง γ = -43, -33 และ -23 dB พิจารณาประสิทธิภาพของระบบ 3 กรณี คือ กรณีแรกแสดงถึงประสิทธิภาพของระบบที่ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ค่าเฉลี่ย PSNR = 10.7828 dB, 11.2581 dB และ 12.9878 dB กรณีที่สองแสดงถึงประสิทธิภาพของระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS ค่าเฉลี่ย PSNR = 11.6207 dB, 12.2901 dB และ 14.8084 dB และกรณีสุดท้าย คือ การใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ค่าเฉลี่ย PSNR = 28.5324 dB, 29.1239 dB และ 30.2511 dB

(ก)

(ข)

ภาพที่ 28 การนำรูปภาพ Lena กลับ ใช้การสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม S = 10 ไม่สร้างเมตริกซ์ P ในระบบใช้การถอดรหัสแบบ ML ก) $\bar{\gamma}_b$ = 5dB ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ประสิทธิภาพของระบบ PSNR = 10.79dB WER = 0.0174 ข) $\bar{\gamma}_b$ = 5dB ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมดประสิทธิภาพของระบบ PSNR = 31.51dB WER = 0.00009

จากภาพที่ 28 (ก) และ (ข) เป็นการเปรียบเทียบรูปภาพ Lena ของการนำภาพกลับ โดยใช้การถอดรหัสแบบ ML ที่มี $\bar{\gamma}_b$ = 5 dB และ γ = -23 dB ความแตกต่างของภาพที่ไม่ใช้การสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม เห็นเป็นภาพลางๆสีขาวเป็นโครงสร้างของภาพ ไม่สามารถเก็บรายละเอียดของภาพได้ เห็นความแตกต่างของภาพที่ได้รับอย่างชัดเจน

9. ผลการทดลองการสลับลำดับสัญลักษณ์แบบกึ่งสุ่มบนช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้ามีการกระจายแบบไรเชียล ในกรณีที่สร้างตัวเลขแบบสุ่มโดยเรียกใช้ฟังก์ชันสุ่มจากภาษาจาวา โดยไม่สร้างเมตริกซ์ P และ กำหนดเงื่อนไขในการตรวจสอบ S = 17

ตารางที่ 21, 22 และ 23 เป็นผลการทดลองจากการสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม ที่สร้างตัวเลขแบบสุ่มโดยเรียกใช้ฟังก์ชันสุ่มจากโปรแกรมภาษาจาวา โดยใช้โครงสร้างทรีในการกระจายตัวเลขสุ่ม และนำมาตรวจสอบเงื่อนไข S = 17 เพื่อนำลำดับของตัวเลขสุ่มที่ได้ เป็นตัวเลขที่ชี้ตำแหน่งสัญลักษณ์เริ่มต้นไปยังตำแหน่งของการสลับลำดับสัญลักษณ์ โดยส่งรูปภาพ Lena จำนวน 200 รูป ผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า กำหนดให้การสุ่มค่าของแอลฟา 1 ค่า มีผลต่อช่วงของสัญลักษณ์ 5 สัญลักษณ์ ที่ γ = -43 dB, -33 dB และ -23 dB พิจารณาที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB โดยใช้การถอดรหัสแบบ ML ค่าเฉลี่ย PSNR และ WER ของคอลัมน์แรกได้จากระบบที่ไม่มีการสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด คอลัมน์ที่สองได้จากระบบที่มีการสลับลำดับกลุ่มสัญลักษณ์ HFS คอลัมน์ที่สามได้จากระบบที่มีการสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด และคอลัมน์สุดท้าย คือ ประสิทธิภาพที่เพิ่มขึ้น เป็นผลต่างระหว่างคอลัมน์ที่สามและคอลัมน์แรก

ตารางที่ 21 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ สร้างตัวเลขแบบกึ่งสุ่ม S = 17 โดยใช้ฟังก์ชันสุ่มโปรแกรมภาษาจาวา ไม่สร้างเมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -43 dB

$\bar{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	8.7952	9.1042	19.9783	11.1831
	(0.0481)	(0.39)	(0.27)	
5.00	9.6469	10.1732	24.9093	15.2623
	(0.0319)	(0.17)	(0.08)	
6.25	10.7828	11.6191	28.1629	17.38
	(0.0215)	(0.08)	(0.03)	

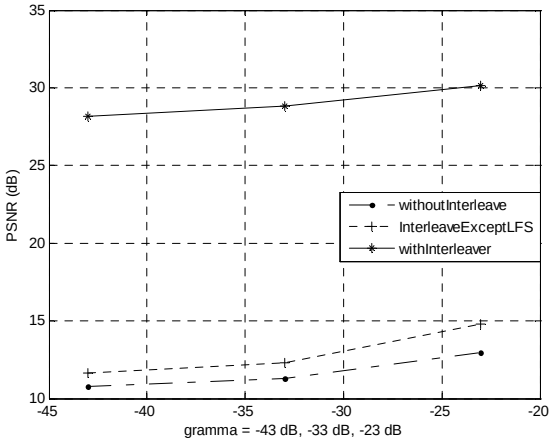
ตารางที่ 22 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ สร้างตัวเลขแบบกึ่งสุ่ม S = 17 โดยใช้ฟังก์ชันสุ่มโปรแกรมภาษาจาวา ไม่สร้างเมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -33 dB

$\overline{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	8.9992	9.3622	21.1272	12.2733
	(0.0431)	(0.32)	(0.21)	
5.00	10.0141	10.6172	25.7050	15.6909
	(0.0281)	(0.14)	(0.06)	
6.25	11.2581	12.2662	28.8334	17.5751
	(0.0184)	(0.07)	(0.02)	

ตารางที่ 23 ค่าเฉลี่ย PSNR ของการนำรูปภาพ Lena กลับ สร้างตัวเลขแบบกึ่งสุ่ม S = 17 โดยใช้ฟังก์ชันสุ่มโปรแกรมภาษาจาวา ไม่สร้างเมตริกซ์ P ระบบส่งภาพใช้การถอดรหัสแบบ ML และ ช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ γ = -23 dB

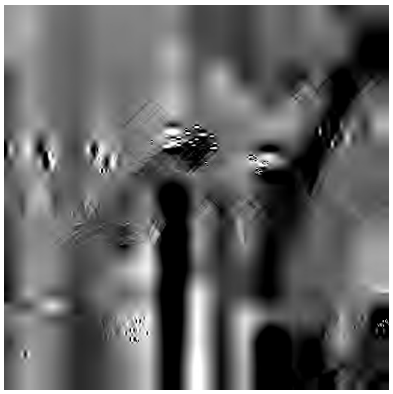
$\overline{\gamma}_b$ (dB)	PSNR(dB) ไม่สลับลำดับกลุ่มสัญลักษณ์ทั้งหมด (WER)	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ HFS (WER×10^{-2})	PSNR(dB) สลับลำดับกลุ่มสัญลักษณ์ (WER×10^{-2})	PSNR(dB) ประสิทธิภาพที่เพิ่มขึ้นต่อระบบ
3.75	9.9578	10.5081	24.3912	14.4335
	(0.0296)	(0.18)	(0.10)	
5.00	11.3875	12.4384	28.1643	16.7768
	(0.0180)	(0.07)	(0.03)	
6.25	12.9878	14.7955	30.1303	17.1426
	(0.0109)	(0.04)	(0.01)	

จากตารางที่ 21, 22 และ 23 เมื่อพิจารณาค่าเฉลี่ย PSNR ที่ $\bar{\gamma}_b$ = 3.75 dB, 5.00 dB และ 6.25 dB เปรียบเทียบระหว่างคอลัมน์ที่สองกับคอลัมน์แรก มีค่าแตกต่างกันเล็กน้อย และคอลัมน์ที่สามเปรียบเทียบกับคอลัมน์แรกมีค่าแตกต่างกันประมาณ 11-17 dB เมื่อพิจารณาค่าเฉลี่ย WER คอลัมน์ที่สองกับคอลัมน์แรก มีค่าแตกต่างกัน ในขณะที่ค่าเฉลี่ย PSNR แตกต่างกันเพียงเล็กน้อย และคอลัมน์ที่สามเปรียบเทียบกับคอลัมน์ที่สอง มีค่าแตกต่างกันเล็กน้อย ในขณะที่ค่าเฉลี่ย PSNR ต่างกันประมาณ 11-17 dB อาจเนื่องจากสัญลักษณ์ LFS ถูกทำลายอย่างมาก จากผลการทดลองพบว่า การนำการสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม มาใช้กับช่องสัญญาณที่มีที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า มีผลทำให้จำนวนความผิดพลาดของสัญลักษณ์ลดลง และ สามารถเพิ่มประสิทธิภาพในการนำภาพกลับ ใช้เวลาในการประมวลผลภาพประมาณ 4.38 วินาทีต่อภาพ



ภาพที่ 29 กราฟแสดงค่าเฉลี่ย PSNR ของการนำภาพกลับ ใช้การสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม S = 17 ไม่สร้างเมตริกซ์ P ในระบบส่งภาพใช้การถอดรหัสแบบ ML บนช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ที่ $\bar{\gamma}_b$ = 6.25 dB ระหว่าง γ = -43, -33 และ -23 dB

จากกราฟเป็นการเปรียบเทียบค่าเฉลี่ย PSNR ของการนำภาพกลับ โดยใช้การสลับลำดับแบบกึ่งสุ่ม เพื่อสร้างเมตริกซ์ P ในระบบใช้การถอดรหัสแบบ ML ที่ $\bar{\gamma}_b$ = 6.25 dB ระหว่าง γ = -43, -33 และ -23 dB พิจารณาประสิทธิภาพของระบบ 3 กรณี คือ กรณีแรกแสดงถึงประสิทธิภาพของระบบที่ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ค่าเฉลี่ย PSNR = 10.7828 dB, 11.2581 dB และ 12.9878 dB กรณีที่สองแสดงถึงประสิทธิภาพของระบบที่ใช้การสลับลำดับกลุ่มสัญลักษณ์ HFS ค่าเฉลี่ย PSNR = 11.6191 dB, 12.2662 dB และ 14.7955 dB และกรณีสุดท้าย คือ การใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ค่าเฉลี่ย PSNR = 28.1629 dB, 28.8334 dB และ 30.1303 dB



(ก)

(ข)

ภาพที่ 30 การนำรูปภาพ Lena กลับ ใช้การสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม S = 17 ไม่สร้างเมตริกซ์ P ระบบใช้การถอดรหัสแบบ ML ก) $\bar{\gamma}_b$ = 5 dB ไม่ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ประสิทธิภาพของระบบ PSNR = 12.22 dB WER = 0.02087 ข) $\bar{\gamma}_b$ = 5 dB ใช้การสลับลำดับกลุ่มสัญลักษณ์ทั้งหมด ประสิทธิภาพของระบบ PSNR = 31.51 dB WER = 0.00006

จากภาพที่ 30 (ก) และ (ข) เป็นการเปรียบเทียบรูปภาพ Lena ของการนำภาพกลับ โดยใช้การถอดรหัสแบบ ML ที่มี $\bar{\gamma}_b$ = 5 dB และ γ = -23 dB ความแตกต่างของภาพที่ไม่ใช้การสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม เห็นเค้าโครงของภาพบริเวณโค้งดำทางขวาของภาพ แต่ไม่สามารถเก็บรายละเอียดของภาพได้ เห็นความแตกต่างของภาพที่ได้รับอย่างชัดเจน

วิจารณ์

1. การทดสอบระบบบนช่องสัญญาณที่มีที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว โดยใช้การสลับลำดับสัญลักษณ์แบบสุ่ม มีค่าเฉลี่ย PSNR ใกล้เคียง และ ลดลงเล็กน้อย เมื่อเปรียบเทียบกับระบบที่ไม่ใช้การสลับลำดับสัญลักษณ์ อาจเนื่องจากช่องสัญญาณที่มีลักษณะที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว เป็นการเปลี่ยนแปลงในเชิงขนาดมีค่าเปลี่ยนไปภายในช่วงสัญญาณ โดยกำหนดให้การสุ่มค่าของแอลฟา 1 ค่ามีผลต่อช่วงของสัญลักษณ์ 1 สัญลักษณ์ ดังนั้นรูปแบบของสัญญาณรบกวนจึงมีลักษณะการกระจายอยู่ภายในกลุ่มสัญลักษณ์ การสลับลำดับสัญลักษณ์ อาจทำให้สัญลักษณ์ที่มีความผิดพลาดแบบกระจายกลายเป็นสัญลักษณ์ที่มีความผิดพลาดแบบต่อเนื่องได้

2. การทดสอบระบบบนช่องสัญญาณแบบที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า โดยใช้การสลับลำดับสัญลักษณ์แบบสุ่ม และ กึ่งสุ่ม สามารถเพิ่มประสิทธิภาพให้กับระบบได้สูงถึง 17 dB เนื่องจากช่องสัญญาณแบบที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า เป็นการเปลี่ยนแปลงเชิงขนาด มีค่าคงที่ภายในช่วงสัญญาณ โดยกำหนดให้การสุ่มค่าของแอลฟา 1 ค่ามีผลต่อช่วงของสัญลักษณ์ 5 สัญลักษณ์ ดังนั้นอาจทำให้รูปแบบของสัญญาณรบกวนที่สร้างขึ้นมีลักษณะเป็นกลุ่ม ทำให้เกิดความผิดพลาดของสัญลักษณ์แบบต่อเนื่อง เมื่อนำการสลับลำดับสัญลักษณ์แบบสุ่ม และ กึ่งสุ่ม มาใช้กับระบบสามารถกระจายความผิดพลาดของสัญลักษณ์แบบต่อเนื่องให้ลดลงได้ และเมื่อเปรียบเทียบค่าเฉลี่ย WER ของระบบที่มีการสลับลำดับสัญลักษณ์เฉพาะกลุ่มสัญลักษณ์ HFS และ ระบบที่มีการสลับลำดับกลุ่มสัญลักษณ์ทั้ง 2 แบบ มีค่าต่างกันน้อยมาก อาจเนื่องมาจากการทำลายสัญลักษณ์ LFS บางส่วนแทบไม่มีผลต่อการเปลี่ยนแปลงของค่าเฉลี่ย WER เพราะมีจำนวนสัญลักษณ์น้อยมากเมื่อเทียบกับจำนวนกลุ่มสัญลักษณ์ HFS ดังตารางที่ 3

3. การถอดรหัสแบบ MAP มีประสิทธิภาพน้อยกว่า ML อาจเนื่องมาจาก การไม่เข้าคู่กันระหว่างค่าทางสถิติของสัญลักษณ์ซีโร่ทรี และ ลำดับของสัญลักษณ์ที่ถูกสลับลำดับแล้ว เพราะการสลับลำดับสัญลักษณ์ ทำให้ตำแหน่งของสัญลักษณ์เปลี่ยนไปจากเดิม ดังนั้นการคำนวณหาเส้นทางบนเทลลิส ในการตรวจจับสัญญาณแบบ MAP เกิดความผิดพลาดได้

4. การทดสอบการประมวลผลของการสลับลำดับสัญลักษณ์แบบสุ่ม โดยไม่สร้างเมตริกซ์ P สามารถลดเวลาในการประมวลผลได้ประมาณ 39.66 วินาทีต่อภาพ

สรุปและข้อเสนอแนะ

สรุป

การทดสอบระบบส่งข้อมูลภาพ ประกอบด้วย การเข้ารหัสภาพ การเปลี่ยนข้อมูลภาพจากบิตเป็นสัญลักษณ์ การเข้ารหัสริงคอนโวลูชันนอล การสลับลำดับสัญลักษณ์ และ การมอดูเลตสัญญาณ เพื่อส่งไปยังช่องสัญญาณไร้สาย ด้านรับ ประกอบด้วย การดีมอดูเลต การสลับลำดับสัญลักษณ์กลับ การคำนวณหาเส้นทางบนเทลลิส การถอดรหัสวีเทอร์บี การเปลี่ยนข้อมูลภาพจากสัญลักษณ์เป็นบิต และการถอดรหัสภาพ การทดสอบระบบ โดยใช้และไม่ใช้การสลับลำดับสัญลักษณ์แบบสุ่ม และ กึ่งสุ่ม ในการส่งภาพผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบเร็ว และ แบบช้า มีการกระจายแบบไรเชียล สามารถสรุปได้ว่า

1. ในกระบวนการสลับลำดับสัญลักษณ์ โดยสร้างและไม่สร้างเมตริกซ์ P ใช้เวลาในการประมวลผล 43.73 วินาทีต่อภาพ และ 4.07 วินาทีต่อภาพ โดยการไม่สร้างเมตริกซ์ P สามารถลดเวลาในการประมวลผลประมาณ 39.66 วินาทีต่อภาพ

2. การใช้การสลับลำดับสัญลักษณ์แบบสุ่ม กับ การถอดรหัสแบบ MAP มีประสิทธิภาพดีขึ้นเพียงเล็กน้อย ประมาณ 1 dB

3. เมื่อเปรียบเทียบระบบที่ใช้ และไม่ใช้การสลับลำดับสัญลักษณ์แบบสุ่ม และ กึ่งสุ่ม เมื่อส่งผ่านช่องสัญญาณที่มีการเลือนหายทางขนาดของสัญญาณแบบช้า ระบบใช้การถอดรหัสแบบ ML และ ใช้การสลับลำดับสัญลักษณ์แบบสุ่ม และ กึ่งสุ่ม สามารถเพิ่มประสิทธิภาพให้กับระบบสูงถึง 17 dB และจำนวนความผิดพลาดของสัญลักษณ์มีค่าลดลง

4. เวลาที่ใช้ในการประมวลผลเมื่อไม่ใช้และใช้การสลับลำดับสัญลักษณ์แบบสุ่ม และ กึ่งสุ่ม เพิ่มขึ้นประมาณ 2 เท่า ของการประมวลผลต่อภาพ โดยการไม่ใช้การสลับลำดับสัญลักษณ์ ใช้เวลาในการประมวลผลประมาณ 2.14 วินาทีต่อภาพ

ข้อเสนอแนะ

1. การใช้การสลับลำดับสัญลักษณ์แบบสุ่มที่มีการสร้างเมตริกซ์ P ใช้เวลาในการประมวลผลประมาณ 43.73 วินาทีต่อภาพ จึงทำการปรับปรุงโดยไม่สร้างเมตริกซ์ P สามารถลดเวลาในการประมวลผล โดยใช้เวลาประมวลผลประมาณ 4.07 วินาทีต่อภาพ ดังนั้นเพื่อเพิ่มประสิทธิภาพทางเวลาในการประมวลผล จึงควรเลือกวิธีการสร้างตัวเลขแบบสุ่มที่มีการกระจายตำแหน่งของสัญลักษณ์ที่เหมาะสมกับข้อมูล เพื่อลดกระบวนการการสลับลำดับ

2. การออกแบบการสลับลำดับสัญลักษณ์แบบกึ่งสุ่ม ควรหาค่าพารามิเตอร์ S ให้เหมาะสมกับระบบ จะทำให้ระบบมีประสิทธิภาพเพิ่มขึ้น

เอกสารและสิ่งอ้างอิง

A.J. Viterbi. 1967. Error Bounds for Convolutional Codes and Asymptotically Optimum Decoding Algorithm, pp. 260-269. **vol. IT-13.** IEEE Trans. Inform. Theory.

B.E. Rimoldi. 1988. A decomposition approach to CPM, pp. 260-270. **vol. 34, no. 2.** IEEE Trans. Info. Theory.

Christine Fragouli and Richard D. Wesel. 1999. Semi-Random Interleaver Design Criteria, pp. 2352-2356. IEEE Global Telecommunications Conference.

G. Ungerboeck. 1982. Channel Coding with Multilevel/Phase Signals, **vol. IT-28, no. 1.** IEEE Trans. Information Theory.

J.G. Proakis. 1995. Digital Communications. McGraw-Hill.

J.M. Shapiro. 1993. Embedded Image Coding using Zerotrees of Wavelete Coefficients, pp. 3445-3462. **vol. 41.** IEEE Trans. Signal Proc.

K.V. Koutsouvelis and C.E. Dimakis. Generating Turbo Code S-Random Interleavers with Application of the Bubble Search Sorting Method, Aristotle University of Thessaloniki, GREECE.

R.H. Yang and D.P. Taylor. Trellis-Coded Continuous-Phase Frequency-Shift Keying with Ring Convolutional Codes.

S. Dolinar and D. Divsalar. 1995. Weight Distributions for Turbo Codes Using Random and Nonrandom Permu-tations. TDA Progress Report 42-122.

S. Mahapakulchai. 2007. MAP Decoding for Polynomial Ring Convolutional Trellis Codes for MPEG-4 Image Transmission System over Rician Fadind Channels, pp. 655-658. **vol. 2.** ECTI-CON 2007.

S. Mahakulchai and I. Sapasirisopon. 2008. Design of Random Interleavers for the Variable Length of MPEG-4 Packets in Rician Fading Channels, pp. 409-412. Proceedings of the 2008 Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON 2008) International Conference, Thailand.

S. Mahapakulchai and I. Sapasirisopon. 2008. MAP Decoding with Random Interleaver for MPEG-4 Image Indoor Wireless Transmission System, to be presented at APCC 2008, JAPAN.

S. Mahapakulchai and R.E. Van Dyck. 2001. Design of Ring Convolutional Trellis Codes for MAP Decoding for MPEG-4 Imagery, Proc. IEEE Inter. Conf. Comm. (ICC2001).

S. Mahapakulchai and R.E. Van Dyck. 2004. Design of Ring Convolutional Trellis Codes for MAP Decoding of MPEG-4 Images, pp. 1033-1037. **vol. 52, no. 7.** IEEE Trans. Comm.

S. Mahapakulchai and R.E. Van Dyck. 2000. Source-Controlled Channel Decoding of MPEG-4 Imagery using CPFSK and Ring Convolutional Codes, IEEE Electro-Information Technology Conf., Chicago.

S. Mahapakulchai and Warutt Thongnumpen. 2007. Design of Block Interleavers in MPEG-4 Image Indoor Wireless Transmission System, International Conference on Engineering, Applied Sciences and Technology (ICEAST2007).

ภาคผนวก

1. โปรแกรมภาษาจาวาการสลับลำดับข้อมูลแบบบล็อก

```
import java.io.*;
import java.lang.*;
import java.math.*;
import java.awt.Event.*;
public class BlockInterleaver
{
        int row;                                    // กำหนดขนาดของแถว
        int col;                                    // กำหนดขนาดของคอลัมน์
        int[][][] matrix_data;                      // กำหนดเมตริกซ์ข้อมูล
        boolean[][] matrix_data;                    // กำหนดเมตริกซ์ข้อมูล
        double[][][] matrix_data3;                  // กำหนดเมตริกซ์ข้อมูล
        int[][] isymsarray;                         // กำหนดเมตริกซ์ข้อมูลซิมเบิล
        int[][] isymsarray_intl;                    // อาร์เรย์เอาพุต
        double[][] isymsarray2;                     // อาร์เรย์อินพุต temp
        double[][] isymsarray_intl2;                // อาร์เรย์เอาพุต
        int[][] isymsarray_deintl_;                 // อาร์เรย์เอาพุต
        double[][] isymsarray_deintl;               // อาร์เรย์เอาพุต
        int count;                                  // กำหนดตัวนับ
        static int nsyms;                           // กำหนดจำนวนของซิมเบิล
        int array_length;
        boolean[] bflagarray;                       // ข้อมูลBranch_Flag
        boolean[] bflagarray_intl;                  // อาร์เรย์เอาพุต
        boolean[] bflagarray_deintl;                // อาร์เรย์เอาพุต

public static void main(String[] args)             //ใช้ในการทดสอบภายในคลาส
throws IOException
{
        int row = 4;                                //กำหนดจำนวนแถว
        int col = 6;                                //กำหนดจำนวนคอลัมน์
        int a_length;                               //ความยาวของข้อมูล
```

```
        int pin,po;                                 //จำนวนของอาร์เรย์ที่เก็บข้อมูล
        int[] pout = new int[2];                    //ค่าที่คำนวณได้จากการหาขนาดบล็อก
        a_length = row*col;                         //ขนาดอาร์เรย์
        nsyms = 16;                                 // สร้างอินพุต
        BlockInterleaver BIntl    = new BlockInterleaver(row,col); //ประกาศวัตถุของคลาส
        int[][] isymsarray        = new int[a_length][2];       //กำหนดอาร์เรย์รับอินพุต
        int[][] isymsarray_intl   = new int[a_length][2];       //กำหนดอาร์เรย์ส่งข้อมูล
        double[][] isymsarray_intl2  = new double[a_length][8];     //กำหนดอาร์เรย์รับข้อมูล
        int[][] isymsarray_deintl_= new int[a_length][2];           //กำหนดอาร์เรย์รับข้อมูล
        double[][] isymsarray_deintl = new double[a_length][8];     //กำหนดอาร์เรย์รับข้อมูล
        int[]   bflagarray        = new int[a_length];          //กำหนดอาร์เรย์ข้อมูลflag
        int[]   bflagarray_intl   = new int[a_length];          //กำหนดอาร์เรย์flagสลับข้อมูล
        int[]   bflagarray_deintl = new int[a_length];          //กำหนดอาร์เรย์flagสลับข้อมูลกลับ
        pin = 901;                                  //กำหนดค่าpin
        po = BIntl.getblocksize(pin);               //เรียกใช้ฟังก์ชันgetblocksize
        pout = BIntl.getblocksize_opt(pin);         //เรียกใช้ฟังก์ชันgetblocksize_opt
        System.out.println(pout[0]);                //สั่งพิมพ์ค่า
        System.out.println(pout[1]);                //สั่งพิมพ์ค่า

BlockInterleaver(int row, int col)                  //ประกาศคลาสConstructor
throws IOException
{
        this.row = row;                             //กำหนดค่าแถว
        this.col = col;                             //กำหนดค่าคอลัมน์
        array_length = row*col;                     //กำหนดขนาดบล็อก
        matrix_data  = new int[row][col][4];                // อินพุต
        matrix_data2 = new boolean[row][col];               // อินพุต
        matrix_data3 = new double[row][col][8];             // อินพุต
        isymsarray        = new int[array_length][2];       // อินพุตซิมเบิลสลับลำดับ
        isymsarray_intl   = new int[array_length][2];       //เอาพุตซิมเบิลสลับลำดับ
        isymsarray_intl2  = new double[array_length][8];  //เอาพุตซิมเบิลสลับลำดับ
```

```
        isymsarray_deintl_= new int[array_length][2];       //เอาพุตซิมเบิลสลับลำดับ
        isymsarray_deintl = new double[array_length][8]; //เอาพุตซิมเบิลสลับลำดับ
        bflagarray        = new boolean[array_length];          // กำหนดอินพุตflag
        bflagarray_intl   = new boolean[array_length];       // กำหนดอาร์เรย์flag
        count = 0;                                          //กำหนดค่าcount
} // end Constructor

public int getblocksize(int size)               //หาขนาดของบล็อก
{
        double sqrt;
        sqrt = Math.sqrt(size);
        size = (int)Math.ceil(sqrt);
        return(size);
 }

public int[] getblocksize_opt(int sizein)    //กำหนดขนาดของบล็อกเป็นแถวและคอลัมน์
{
        double sqrt;
        int size,size2;
        int sizeout[] = new int[2];
        sqrt  = Math.sqrt(sizein);
        size  = (int)Math.ceil(sqrt);
        size2 = size-1;
        if ((size*size2)>=sizein)
        {
sizeout[0] = size2;
                sizeout[1] = size;
        }
        else
        {
sizeout[0] = size;
```

```
                sizeout[1] = size;
        }
        return(sizeout);
    }

public int[][] interleaver(int[][] isymsarray_input)    // ฟังก์ชันอินเตอร์ลีฟชนิดจำนวนเต็ม
{
        create_matrix(isymsarray_input, row, col);//นำซิมเบิลเรียงในอาร์เรย์
        transpost(matrix_data, row, col);              // ทำการสลับแถวและหลักของซิมเบิล
        isymsarray_intl = matrixtoarray(matrix_data, row, col);      // เรียงข้อมูลออกเป็นอาร์เรย์
        return isymsarray_intl;                        // ส่งค่าซิมเบิลกลับไปยังคลาสที่เรียกใช้งาน
    }

public boolean[] interleaver_bflag(boolean[] isymsarray_input)    // ฟังก์ชันอินเตอร์ลีฟFlag
{
        create_matrix2(isymsarray_input, row, col);          //นำซิมเบิลเรียงในอาร์เรย์
        transpost2(matrix_data2, row, col);          // ทำการสลับแถวและหลักของซิมเบิล
        bflagarray_intl = matrixtoarray2(matrix_data2, row, col);    // เรียงข้อมูลออก
        return bflagarray_intl;                        // ส่งค่าซิมเบิลกลับไปยังคลาสที่เรียกใช้งาน
}

public double[][] interleaver2(double[][] isymsarray_input)  // ฟังก์ชันสลับลำดับข้อมูล
{
        create_matrix3(isymsarray_input, row, col);          //นำซิมเบิลเรียงในอาร์เรย์
        transpost3(matrix_data3, row, col);                  // ทำการสลับแถวและหลักของซิมเบิล
        isymsarray_deintl = matrixtoarray3(matrix_data3, row, col);// เรียงข้อมูลออก
        return isymsarray_deintl;            // ส่งค่าซิมเบิลกลับไปยังคลาสที่เรียกใช้งาน
}
public int[][] deinterleaver_(int[][] isymsarray_input)  // กระบวนการสลับลำดับกลับ
{
        create_matrix(isymsarray_input, col, row);//นำซิมเบิลเรียงในอาร์เรย์
```

```
        transpost(matrix_data, col, row);              // ทำการสลับแถวและหลักของซิมเบิล
        isymsarray_deintl_ = matrixtoarray(matrix_data, col, row); // เรียงข้อมูลออก
        return isymsarray_deintl_;                     // ส่งค่ากลับ
}

public double[][] deinterleaver(double[][] isymsarray_input)  //สลับลำดับกลับ
{
        create_matrix3(isymsarray_input, col, row);//นำซิมเบิลเรียงในอาร์เรย์
        transpost3(matrix_data3, col, row);            // ทำการสลับแถวและหลักของซิมเบิล
        isymsarray_deintl = matrixtoarray3(matrix_data3, col, row);          // เรียงข้อมูลออก
        return isymsarray_deintl;                      // ส่งค่าซิมเบิลกลับไปยังคลาสที่เรียกใช้งาน
}

public boolean[] deinterleaver_bflag(boolean[] isymsarray_input) // สลับลำดับFlagกลับ
{
        create_matrix2(isymsarray_input, col, row);          //นำซิมเบิลเรียงในอาร์เรย์
        transpost2(matrix_data2, col, row);                  // ทำการสลับแถวและหลักของซิมเบิล
        bflagarray_intl = matrixtoarray2(matrix_data2, col, row);    // เรียงข้อมูลออก
        return bflagarray_intl;                              // ส่งค่าซิมเบิลกลับไปยังคลาสที่เรียกใช้งาน
}

public void create_matrix(int[][] isymarray_input,int row,int col)    //นำซิมเบิลเรียงในอาร์เรย์
{
        int index = 0;
        matrix_data = new int[row][col][8];
        for (int i = 0 ; i < row ; i++)
        {
                for (int j = 0 ; j < col ; j++)
                {
matrix_data[i][j] = isymarray_input[index];
                        index += 1;
```

```
                }
        }
}

public void transpost(int matrix_t[][][],int row,int col)    // ทำการสลับแถวและหลักของซิมเบิล
{
        int buffer[][][] = new int[col][row][8];
        for (int i = 0 ; i < row ; i++)
        {
for (int j = 0 ; j < col ; j++)
                {
                        buffer[j][i] = matrix_t[i][j];
                }
        }
        matrix_data = buffer;
}

public int[][] matrixtoarray(int matrix_t[][][],int row,int col)    // เรียงข้อมูลออกเป็นอาร์เรย์
{
        int index = 0;
        int[][] arrayoutput = new int[row*col][8];
        for (int i = 0 ; i < col ; i++)
        {
        for (int j = 0 ; j < row ; j++)
                {
                        arrayoutput[index] = matrix_data[i][j];
                        index += 1;
                }
        }
        return arrayoutput;
}
```

```
public void create_matrix2(boolean[] isymarray_input,int row,int col)//นำซิมเบิลเรียงในอาร์เรย์
{
	int index = 0;
	matrix_data2 = new boolean[row][col];
	for (int i = 0 ; i < row ; i++)
	{
		for (int j = 0 ; j < col ; j++)
		{
			matrix_data2[i][j] = isymarray_input[index];
			index += 1;
		}
	}
}

public void transpost2(boolean matrix_t[][],int row,int col) //ทำการสลับแถวและหลักซิมเบิล
{
	boolean buffer[][] = new boolean[col][row];
	for (int i = 0 ; i < row ; i++)
	{
for (int j = 0 ; j < col ; j++)
		{
			buffer[j][i] = matrix_t[i][j];
		}
	}
	matrix_data2 = buffer;
}

public boolean[] matrixtoarray2(boolean matrix_t[][],int row,int col) // เรียงข้อมูลออก
{
	int index = 0;
	boolean[] arrayoutput = new boolean[row*col];
```

```
        for (int i = 0 ; i < col ; i++)
        {
for (int j = 0 ; j < row ; j++)
                {
                        arrayoutput[index] = matrix_data2[i][j];
                        index += 1;
                }
        }
        return arrayoutput;
}

public void create_matrix3(double[][] isymarray_input,int row,int col)//นำซิมเบิลเรียงในอาร์เรย์
{
        int index = 0;
        matrix_data3 = new double[row][col][8];
        for (int i = 0 ; i < row ; i++)
        {
                for (int j = 0 ; j < col ; j++)
                {
                        matrix_data3[i][j] = isymarray_input[index];
                        index += 1;
                }
        }
}

public void transpost3(double matrix_t[][][],int row,int col) //ทำการสลับแถวและหลักของซิมเบิล
{
        double buffer[][][] = new double[col][row][8];
        for (int i = 0 ; i < row ; i++)
        {       for (int j = 0 ; j < col ; j++)
                {
```

```
                buffer[j][i] = matrix_t[i][j];
                }
        }
        matrix_data3 = buffer;
}

public double[][] matrixtoarray3(double matrix_t[][][],int row,int col)//เรียงข้อมูลออกเป็นอาร์เรย์
{
        int index = 0;
        double[][] arrayoutput = new double[row*col][8];
        for (int i = 0 ; i < col ; i++)
        {       for (int j = 0 ; j < row ; j++)
                {
                        arrayoutput[index] = matrix_data3[i][j];
                        index += 1;
                }
        }
        return arrayoutput;
}
} // End BlockInterleaver
```

ที่มา: Mahapakulchai and Thongnumpen (2007)

2. โปรแกรมภาษาจาวาการสลับลำดับข้อมูลแบบสุ่ม

```
import java.io.*;
import java.lang.*;
import java.math.*;
import java.awt.Event.*;

public class RandomInt
{
        int size;                       // ขนาดข้อมูล
        int [] randomNumber;            // แบบอาร์เรย์ตัวสุ่ม
        int [][] isymsarray;            // อาร์เรย์ซิมเบิล
        int [][] isymsarray_intl;       // อาร์เรย์การสลับลำดับ
        double[][] isymsarray_intl2;    // อาร์เรย์ก่อนทำการสลับลำดับกลับ
        double[][] isymsarray_deintl;   // อาร์เรย์สลับลำดับกลับ
        int[][] data_int;               // อาร์เรย์ซิมเบิล
        double[][] data_deint;          // อาร์เรย์ทำการสลับลำดับกลับ
        int[][] isymsarray_deintl2;     // อาร์เรย์หลังทำการสลับลำดับกลับ
public static void main(String[] args) throws IOException
{
        int size=20;
        int Randseed=1;
        RandomInt randInt    = new RandomInt(size);
        int[][] isymsarray_intl   = new int[size][2];
        int[][] isymsarray        = new int[size][2];
        int[][] isymsarray_deintl2 = new int[size][2];
}       // end main


RandomInt (int size)      throws IOException      //ประกาศคลาสConstructor
{
        this.size = size;
```

```
        randomNumber = new int[size];              // ตัวเลขสุ่ม
        isymsarray  = new int[size][2];            //อินพุต
        isymsarray_deintl = new double[size][8];  //ข้อมูลสลับลำดับกลับ
        isymsarray_intl  = new int[size][2];       //ข้อมูลสลับลำดับ
        isymsarray_intl2 = new double[size][8];   //ข้อมูลสลับลำดับกลับ
}       //End Constructor

public int[][] interleaver(int[][] isymsarray_input, int Rseed_Int )    //สลับลำดับข้อมูล
{
        random_intl(size,Rseed_Int);                        //เรียกใช้ฟังก์ชันสุ่มตัวเลข
        isymsarray_intl = seqence(random,isymsarray_input); //ผ่านกระบวนการอินเตอร์ลีฟ
        return isymsarray_intl;                             //ส่งค่ากลับไปยังคลาสที่เรียกใช้งาน
}
public double[][] deinterleaver(double[][] isymsarray_input1)  //สลับลำดับกลับ
{
        isymsarray_deintl = Reseqence(random, isymsarray_input1); //การสลับลำดับ
        return isymsarray_deintl;                   // ส่งค่ากลับไปยังคลาสที่เรียกใช้งาน
}

public int[][] deinterleaver_data(int[][] isymsarray_input2) //โปรแกรมทดสอบสลับลำดับกลับ
 {
        isymsarray_deintl2 = Reseqence_data(random, isymsarray_input2);//นำข้อมูลกลับ
        return isymsarray_deintl2;                  //ส่งค่ากลับ
 }

public void random_intl(int size,int x0)            //การสลับลำดับข้อมูล
 {
        int c0;                                     //ประกาศตัวแปร
        int m=size;                                 //ประกาศตัวแปร
        int a=m+1;                                  //ประกาศตัวแปร
        int xn;                                     //ประกาศตัวแปร
```

```
        if ((size==598)||(size==572)||(size==624)) // กำหนดค่าc0ให้แพ็กเก็จ 7, 8, 12
                c0=17;
        else                                      // กำหนดค่า c0 ให้กับแพ็กเก็จที่เหลือ
                c0=13;
        for (int k = 0 ; k < m ; k++)             // ลูปของการสุ่มตัวเลขจาก 0 ถึง m
        {
                If (k==0)                         // การสุ่มครั้งแรกมีค่าเท่ากับseedเริ่มต้น
                        xn=x0;                    // ค่าตัวเลขแบบสุ่มที่ได้ค่าเท่ากับ seed เริ่มต้น
                else                              //สุ่มตัวเลขโดยใช้อัลกอริทึม
                {
                        xn = (a*x0+c0)%m;         // ค่าตัวเลขแบบสุ่มที่ได้
                        x0 = xn;                  // ค่าตัวเลขแบบสุ่มมีค่าเท่ากับค่า x0
                }
                randomNumber[k]=xn;               // เก็บค่าตัวเลขที่สุ่มได้ไว้ในอาร์เรย์
        }
}

public int[][] seqence(int[]RandomNum ,int [][]i_matrix)      // การสลับลำดับข้อมูล
{
        int[][] data_int = new int[size][2];
        for (int i = 0 ; i < size ; i++)
        {
                for (int j = 0 ; j < 2 ; j++)
                {
                        for (int l=0;l<size;l++)
                        {
                                If (l==RandomNum[i])   // ตัวชี้ที่ตำแหน่ง l เท่ากับตัวเลขสุ่ม
                                {
                                        data_int[i][j] = i_matrix[l][j]; // ซิมเบิลตำแหน่ง l ไว้ที่ i
                                }
                        }
```

```
                }
        }
        return data_int;                                    // ส่งค่ากลับ
}

public double [][]Reseqence(int[]ReRandomNum ,double [][]i_matrix1) // การสลับลำดับกลับ
        {
                double[][] data_deint = new double[size][8];
                for (int i = 0 ; i < sizeR ; i++)
                {
                        for (int j = 0 ; j < 8 ; j++)
                        {
                                for(int l=0;l<size;l++)
                                {
                                        if (l==ReRandomNum[i])//ตัวชี้ที่ตำแหน่ง 1 ตัวเลขสุ่ม
                                        {
                                                data_deint[l][j] = i_matrix1[i][j];
                                        }
                                }
                        }
                }
                return data_deint;                                  // ส่งค่ากลับ
}

public int [][]Reseqence_data(int[]ReRandomNum ,int [][]i_matrix1)  // การสลับลำดับข้อมูลกลับ
{
int[][]data_deint = new int[size][2];
for (int i = 0 ; i < size ; i++)
{
        for (int j = 0 ; j < 2 ; j++)
        {
```

```
                for(int l=0;l<size;l++)
                {
                        if(l==ReRandomNum[i])
                        {
                        data_deint[l][j] = i_matrix1[i][j];
                        }
                }
        }
}
return data_deint;
}
}                                       //end class
```

3. โปรแกรมภาษาจาวาการสลับลำดับข้อมูลแบบกึ่งสุ่ม

```
import java.io.*;
import java.lang.*;
import java.math.*;
import java.awt.Event.*;
import java.util.*;
import java.text.*;

public class SemiRandomML
{
        int size;                               //ขนาดข้อมูล
        int []number;                           //เก็บตัวเลขสุ่ม
        int [][] isymsarray;                    //รับข้อมูลภาพ
        int [][] isymsarray_intl;               //ข้อมูลสลับลำดับ
        double[][] isymsarray_intl2;            //รับข้อมูล
        double[][] isymsarray_deintl;           //ข้อมูลสลับลำดับกลับ
        int[][] isymsarray_deintl2;             //ทดสอบข้อมูลสลับลำดับกลับ
        int []report2;                          //ข้อมูลที่จัดเรียงแล้ว
        int[] Numberbuild;                      //สร้างโครงสร้างทรี
        int[] numberShrink;                     //กระจายตัวเลขสุ่ม

public static void main(String[] args)         //ทดสอบภายในคลาส
throws IOException
{
        int size=500;                           //ขนาดข้อมูล
        SemiRandomML SMrandInt    = new SemiRandomML(size);//กำหนดวัตถุ
        int[][] isymsarray_intl   = new int[size][2]; //ข้อมูลสลับลำดับ
        int[][] isymsarray        = new int[size][2];  //รับข้อมูลภาพ
        double[][] isymsarray_intl2  = new double[size][8]; //ข้อมูลสลับลำดับ
        double[][] isymsarray_deintl = new double[size][8]; //ข้อมูลสลับลำดับกลับ
        int[][] isymsarray_deintl2 = new int[size][2];        //ข้อมูลสลับลำดับกลับ
```

```
int []number=new int[size];                    //เก็บตัวเลขสุ่ม
int []Numberbuild=new int[size];               //สร้างโครงสร้างทรี
int []numberShrink=new int[size];              //กระจายตัวเลขสุ่ม
int[][]data_int = new int[size][2];            //ข้อมูลสลับลำดับ
int []my=new int[size];                        //สลับค่าข้อมูล
int sseed=2;                                   //กำหนดค่า
for(int i=0;i<size;i++)                        //สร้างอินพุต
{
for(int j=0;j<2;j++)
{
                    isymsarray[i][j]=i;
}
}
isymsarray_intl=SMrandInt.interleaver(isymsarray,sseed);//สลับลำดับ
System.out.print("DataInterleaver=");          //สั่งพิมพ์
System.out.print('\n');                        //ขึ้นบรรทัดใหม่
for(int j=0;j<size;j++)                        //ลูปสั่งพิมพ์
{
        for(int y=0;y<2;y++)
        {
        System.out.print(isymsarray_intl[j][y]);
        System.out.print(',');
        }
}
System.out.print('\n');                        //ขึ้นบรรทัดใหม่
isymsarray_deintl2=SMrandInt.deinterleaver_data(isymsarray_intl);//สลับลำดับกลับ
System.out.print("DataInterleaver=");          //สั่งพิมพ์
System.out.print('\n');                        //ขึ้นบรรทัดใหม่
for(int x=0;x<size;x++)                        //ลูปสั่งพิมพ์
{
        for(int z=0;z<2;z++)
```

```
                {
                System.out.print(isymsarray_deintl2[x][z]);
                System.out.print(',');
                }
        }
                System.out.print('\n');                 //ขึ้นบรรทัดใหม่
}               //end main                              //จบการทดสอบ

SemiRandomML(int size)                                  //สร้างConstructor
throws IOException
{
        this.size = size;
        number = new int[size];                         //เก็บตัวเลขสุ่ม
        Numberbuild=new int[size];                      //สร้างโครงสร้างทรี
        numberShrink=new int[size];                     //กระจายตัวเลขสุ่ม
        report2=new int[size];                          //ข้อมูลที่จัดเรียงแล้ว
        isymsarray       = new int[size][2];            //รับข้อมูล
        isymsarray_intl   = new int[size][2];           //ข้อมูลสลับลำดับ
        isymsarray_intl2  = new double[size][8];        //รับข้อมูล
        isymsarray_deintl = new double[size][8];        //ข้อมูลสลับลำดับกลับ
}       //End Constructor

 public int[][] interleaver(int[][]isymsarray_input,int SMseed) //สลับลำดับ
{
        int[] report1=new int[size];                    //เก็บค่าโครงสร้างทรี
        RandomNumber(size,SMseed);                      //สร้างตัวเลขสุ่ม
        report1=buildHeap(number,size);                 //สร้างโครงสร้างทรี
        report2=ShrinkHeap(report1,size);               //กระจายตัวเลขสุ่ม
        isymsarray_intl = seqence(isymsarray_input,report2,size);//สลับลำดับ
        return isymsarray_intl;                         //คืนค่ากลับ
}
```

```
public double[][] deinterleaver(double[][] isymsarray_input3)//สลับลำดับกลับ
{
        isymsarray_deintl = Reseqence(report2, isymsarray_input3);//สลับลำดับกลับ
        return isymsarray_deintl;                       //คืนค่ากลับ
}


public int[][] deinterleaver_data(int[][] isymsarray_input2)//สลับลำดับกลับ
{
        isymsarray_deintl2 = Reseqence_data(report2, isymsarray_input2,size);//สลับลำดับกลับ
        return isymsarray_deintl2;                      //คืนค่ากลับ
}

public void RandomNumber(int size,int SMseed)           //สร้างตัวเลขสุ่ม
{
        Random rand=new Random(SMseed);                 //เรียกฟังก์ชันสุ่ม
        int pointer;                                    //ตัวเลขสุ่ม
        int tmp;                                        //ค่าเปรียบเทียบ
        for(int i=0;i<size;i++)                         //ลูปสร้างตัวเลข
        {
                number[i] = i;
        }

        for(int i=0;i<size;i++)                         //ลูปตรวจสอบ
        {
                pointer = rand.nextInt(size-i)+i;       //ตัวเลขสุ่ม
                swap(number, pointer, i);               //สลับค่า
        }
}
```

```
public int[] buildHeap(int[]Numberbuild,int size)          //สร้างโครงสร้างทรี
{
        int n = 1;                                         //
        int parent;                                        //ตัวหลักเพื่อเปรียบเทียบ
        int child;                                         //ตัวรองนำมาเปรียบเทียบ
        int output;                                        //ค่าทีเปรียบเทียบ
        while (n < size)                                   //เงื่อนไขคำสั่ง
        {
                n++;                                       //เพิ่มค่าn
                child = n - 1;                             //กำหนดสมการ
                parent = (child - 1) / 2;                  //กำหนดสมการ
                output=compareTo(Numberbuild[parent],Numberbuild[child],size);//เปรียบเทียบ
                if(parent>=0&&output!=0)                   //สร้างโครงสร้างทรี
                {
                        swap(Numberbuild, parent, child);//สลับค่า
                        child=parent;
                        parent=(child-1)/2;
                }
        }
    return Numberbuild;                                    //ส่งค่ากลับ
  }

public int[] ShrinkHeap(int[]numberShrink,int size)        //กระจายตัวเลขสุ่ม
{
        int m=0;                                           //ค่าเริ่มต้น
        int output2;                                       //ค่าที่เปรียบเทียบ
        while (m<size)                                     //เงื่อนไขคำสั่ง
        {
                m++;                                       //เพิ่มค่าm
                for(int n=m-1;n<size-1;n++)                //ลูปตรวจสอบค่า
                {
```

```
                    if(m!=n)                    //เงื่อนไขm!=n
                    {
                        output2=compareTo(numberShrink[m],numberShrink[n],size);
                        if(output2!=0)          //ลูปตรวจสอบค่า
                            swap(numberShrink,m,n);//สลับค่า
                        else break;             //หยุด
                    }
                }
            }
            return numberShrink;                //ส่งค่ากลับ
        }

public int compareTo(int first,int second,int size)     //เปรียบเทียบค่า
{
        int s=17;                               //กำหนดS=17
        if (second-first>s)                     //ผลต่างมากกว่าS
        return -1;                              //ส่งค่ากลับ
        else if (first-second>s)                //ผลต่างมากกว่าS
        return 1;                               //ส่งค่ากลับ
        else                                    //นอกเงื่อนไข
        return 0;                               //ส่งค่ากลับ
}
 public int[] swap(int[] my,int i,int j)        //สลับค่า
{
        int temp;                               //ค่าเปรียบเทียบ
        temp = my[i];                           //ให้ temp = my[i]
        my[i] = my[j];                          //ให้ my[i] = my[j]
        my[j] = temp;                           //ให้ my[j] = temp
        return my;                              //ส่งค่ากลับ
}
}                                   //end class   //จบคลาส
```

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นางสาวอิสรีย์ สรรพสิริโสภณ
วัน เดือน ปี ที่เกิด	27 เมษายน 2527
สถานที่เกิด	จังหวัดสระบุรี
ประวัติการศึกษา	วิศวกรรมศาสตร์บัณฑิต (วิศวกรรมไฟฟ้า) มหาวิทยาลัยรังสิต
ตำแหน่งหน้าที่การงานปัจจุบัน	นักศึกษาปริญญาโท มหาวิทยาลัยเกษตรศาสตร์ บางเขน
สถานที่ทำงานปัจจุบัน	อาคารวิศวกรรมศาสตร์ 60 ปี ชั้น 5 ห้อง 505/5
ผลงานดีเด่นและรางวัลทางวิชาการ	- ได้รับการตีพิมพ์ผลงานเรื่อง “Design of Random Interleavers for the Variable Length of MPEG-4 Packets in Rician Fading Channels” ใน ECTI-CON 2008 International Conference, pp. 409-412, May, 2008 - ได้รับการตีพิมพ์ผลงานเรื่อง “MAP Decoding with Random Interleaver for MPEG-4 Image Indoor Wireless Transmission System”, presented at APCC 2008, JAPAN, Oct. 2008
ทุนการศึกษาที่ได้รับ	-