บทที่ 2

การอัดข้อมูลรูปภาพ

การอัดข้อมูลรูปภาพเป็นการปรับแต่งหรือจัดเก็บข้อมูลรูปภาพให้อยู่ในรูปแบบที่ไม่เปลืองเนื้อที่ในการจัดเก็บและยังช่วยประหยัดเวลาในการรับส่งข้อมูลรูปภาพอีกทางหนึ่ง

การอัดข้อมูลรูปภาพมีอยู่หลายวิธีขึ้นอยู่กับรูปแบบของรูปภาพและวัตถุประสงค์ของการใช้งาน โดยสิ่งสำคัญที่จะต้องคำนึงถึงในการอัดข้อมูลรูปภาพคือ

- เวลาที่ใช้ในการอัดข้อมูลและการกระจายกลับต้องอยู่ในเกณฑ์ที่ผู้ใช้พอใจ
- รูปภาพที่กระจายกลับจากข้อมูลที่ถูกอัด คุณภาพต้องไม่ด้อยลงไปมากเกินควร
- รูปแบบของข้อมูลที่ถูกอัดต้องเหมาะสมแก่การเก็บ หรือการส่งถ่ายข้อมูล

ในบทนี้จะกล่าวถึงวิธีการพื้นฐานของการอัดข้อมูลเฉพาะรูปภาพประเภทความเข้มต่อเนื่อง ซึ่งประกอบด้วยวิธีพื้นฐานดังต่อไปนี้ (Andrian 1991)

วิธีการพื้นฐานของการอัดข้อมูลรูปภาพ

1. วิธีการอัดข้อมูลโดยหลักสถิติ(Statistical Compression)

ข้อมูลในระบบคอมพิวเตอร์มีพื้นฐานการเก็บในระบบของเลขฐานสองหรือบิตศูนย์ หนึ่ง เท่านั้น ดังนั้นค่าความเข้มในแต่ละจุดบนจอภาพหรือพิกเซล(pixel) ที่มีได้จะขึ้นอยู่กับจำนวนบิตที่ใช้แทนรหัสความเข้มต่อ 1 พิกเซล ตัวอย่างเช่น จอภาพ VGA ขนาดความละเอียด 640 x 480 และ 16 ระดับความเข้ม แสดงว่าใช้จำนวนบิต 4 บิต แทนรหัสความเข้มตั้งแต่ค่า 0-15 ดังนั้นจะต้องใช้หน่วยความจำทั้งสิ้น 4 x 640 x 480 หรือเท่ากับ 150 กิโลไบท์ เพื่อเก็บภาพตามขนาดดังกล่าว แต่ถ้าพิจารณาจากค่าทางสถิติของโอกาสที่จะเกิดระดับความเข้มในแต่ละระดับ จะพบว่าแต่ละระดับความเข้มมีโอกาสเกิดขึ้นมากน้อยต่างกัน ด้วยเหตุนี้จึงทำให้สามารถลดขนาดในการจัดเก็บลงได้โดยการจัดรหัสที่แทนระดับความเข้มใหม่ตามโอกาสการเกิดที่มากน้อยต่างกันนั้น เช่นเมื่อพบว่ารูปภาพมีระดับความเข้มเป็นดังนี้

พบว่ามีอยู่ 4 ระดับความเข้มที่เกิดขึ้นในรูปภาพบ่อยที่สุดหรือคิดเป็น 60% จะแทนด้วยรหัสเพียง 3 บิตคือ

000 001 010 011

และมีอีก 4 ระดับความเข้มที่เกิดขึ้นในรูปภาพรองลงมาหรือคิดเป็น 30% ก็จะแทนด้วยรหัส 4 บิตดังนี้คือ

1000 1001 1010 1011

ส่วนระดับความเข้มในรูปภาพที่เหลือซึ่งมีโอกาสเกิดขึ้น 10% จะแทนด้วยรหัส 5 บิตคือ

11000 11001 11010 11011 11100 11101 11110 11111

จะสังเกตได้ว่าในการกำหนดรหัสใหม่นี้ ส่วนต้นของรหัสที่ยาวกว่าจะไม่ซ้ำกับส่วนของรหัสที่สั้นกว่า เมื่อแทนด้วยรหัสตามนี้แล้วขนาดของแฟ้มข้อมูลจะลดขนาดจาก 150 กิโลไบท์ลงเหลือ

$(0.6 \times 3) + (0.3 \times 4) + (0.1 \times 5) \times 640 \times 480 = 131.25$ กิโลไบท์

และจำนวนบิตเฉลี่ยที่ใช้จะลดลงจาก 4 บิตเหลือ 3.5 บิตเท่านั้น

2. Spatial Compression

Spatial Compression เป็นการอัดข้อมูลโดยอาศัยความซ้ำของข้อมูลที่อยู่ติดกัน ดังตัวอย่างที่จะแสดงต่อไปนี้

ตัวเลขข้างล่างแสดงระดับความเข้มในกรอบขนาด 6x4 จะสังเกตเห็นได้ว่ามีบางค่าในตำแหน่งติดกัน มีระดับความเข้มซ้ำกัน

1	2	1	1	1	1
1	3	4	4	4	4
1	1	3	3	3	5
1	1	1	1	3	3

เมื่อนำมาเรียงลำดับทั้ง 24 ค่าจะได้ลำดับดังนี้

1 2 1 1 1 1 1 3 4 4 4 4 1 1 3 3 3 5 1 1 1 1 3 3

เทคนิคของการเข้ารหัสแบบนี้ วิธีหนึ่งคือ Run Length coding ซึ่งเหมาะสำหรับการอัดข้อมูลที่มีค่าซ้ำกันมากๆในตำแหน่งที่ติดกัน การเข้ารหัสด้วยเทคนิค

นี้กระทำได้โดยการจัดคู่ลำดับของค่าความเข้มและความซ้ำ โดยให้ค่าแรกแทนระดับความเข้ม และค่าที่สองแทนจำนวนความซ้ำที่เกิดขึ้น จากข้อมูล 24 ค่าข้างต้น จะเข้ารหัสได้ดังนี้

(1,1) (2,1) (1,5) (3,1) (4,4) (1,2) (3,3) (5,1) (1,4) (3,2)

เมื่อนำมาเรียงลำดับเพื่อการจัดเก็บจะได้ดังนี้

1 1 2 1 1 5 3 1 4 4 1 2 3 3 5 1 1 4 3 2

ซึ่งจะเห็นได้ว่าลดจำนวนข้อมูลในการจัดเก็บลงเหลือเพียง 20 ค่าจากจำนวนข้อมูลเดิม 24 ค่า

นอกจากนี้อาจจะใช้วิธี Run Length Coding ร่วมกับวิธีการอัดข้อมูลตามหลักสถิติ ตามที่กล่าวมาแล้วเพื่อเป็นการเพิ่มประสิทธิภาพ

3. <u>การเข้ารหัสแบบใช้เส้นแสดงรูปร่าง (Contour Coding)</u>

การเข้ารหัสแบบใช้เส้นแสดงรูปร่างเป็นการอัดข้อมูลด้วยการเข้ารหัสเส้นแสดงรูปร่างของพื้นที่ของพิกเซลที่มีระดับความเข้มเดียวกันแทนการเก็บระดับความเข้มทุกตำแหน่งในพื้นที่นั้น วิธีนี้จึงเหมาะกับรูปภาพที่มีระดับความเข้มเดียวกันเป็นบริเวณกว้างดังตัวอย่างข้อมูลต่อไปนี้

1 1 1 1 1 1 1 1 2 2 2 2 2 2 2
1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1 1 1 2 2 2

จากข้อมูลข้างต้นจะเห็นได้ว่าเส้นแสดงรูปร่างของพื้นที่ที่มีระดับความเข้มต่างกันเป็นดังนี้

0 0 0 0 0 0 0 0 1 0 0 0 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 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

จุดเริ่มต้นของขอบคือจุด (8,0) แล้วย้ายมายังจุดต่างๆ ดังนี้

(7,1) (6,2) (7,3) (8,3) (9,3) (10,3) (11,4)

ให้ตำแหน่งการเปลี่ยนแปลงทิศทางของเส้นแสดงรูปร่าง ซึ่งเป็นไปได้ 8 ทิศทาง กำหนดด้วยรหัสดังต่อไปนี้

จากข้อมูลข้างต้นเราสามารถเข้ารหัสเส้นแสดงรูปร่างด้วยรหัสทิศทาง โดยเริ่มจากจุด(8,0) ได้รหัสใหม่ดังนี้

6 6 4 3 3 3 4

ซึ่งแต่ละทิศทางแทนด้วยจำนวนบิตเพียง 3 บิต เพราะฉะนั้นจำนวนบิตที่ใช้ทั้งหมดจะคำนวณได้จากจำนวนบิตที่ใช้แทนตำแหน่งเริ่มต้นของเส้นแสดงรูปร่างรวมกับจำนวนบิตที่ใช้แทนทิศทางและจำนวนบิตที่ใช้แทนค่าความเข้มระหว่างเส้นแสดงรูปร่างดังนี้

จำนวนบิตที่ใช้ = จำนวนบิตแทนตำแหน่ง(8,0) + (7 x 3) บิตแทนทิศทาง
+ จำนวนบิตที่ใช้แทนค่าความเข้มระหว่างเส้นแสดงรูปร่าง

4. <u>การอัดแบบควอนไทซิ่ง(Quantizing Compression)</u>

เทคนิคควอนไทซิ่งเป็นการลดจำนวนของระดับความเข้มลงจากเดิมเพื่อลดขนาดรหัสที่ใช้แทนค่าความเข้มเช่น จากรูปภาพเดิมที่มีระดับความเข้มเท่ากับ 256 ระดับ หากลดจำนวนระดับความเข้มลงเหลือ 16 ระดับ จะมีผลให้ใช้จำนวนบิตที่แทนระดับ

ความเข้มลดลงจาก 8 บิตต่อหนึ่งระดับความเข้ม เหลือเพียง 4 บิตต่อหนึ่งระดับความเข้ม

ให้ P เป็นจำนวนพิกเซลของรูปภาพที่ต้องการอัดข้อมูลให้เหลือ N ระดับความเข้ม เริ่มต้นพิจารณาด้วยการสร้างฮิสโตแกรม(Histogram)ของระดับความเข้มในภาพ แล้วแบ่งระดับความเข้มใหม่มีจำนวน N ระดับ โดยการเฉลี่ยให้จำนวนพิกเซลในแต่ละช่วงมีจำนวนเท่าๆกันคือช่วงละ P/N พิกเซล แล้วใช้ค่ามัธยฐานของแต่ละช่วงเป็นตัวแทนของระดับความเข้มใหม่ในช่วงนั้นๆ แทน

ตัวอย่าง สมมติให้รูปภาพซึ่งมี 10 ระดับความเข้มมีรูปแบบดังต่อไปนี้

2 9 6 4 8 2 6 3 8 5 9 3 7
3 8 5 4 7 6 3 8 2 8 4 7 3
3 8 4 7 4 9 2 3 8 2 7 4 9
3 9 4 7 2 7 6 2 1 6 5 3 0
2 0 4 3 8 9 5 4 7 1 2 8 3

เมื่อต้องการอัดข้อมูลรูปภาพนี้ โดยลดระดับความเข้มให้เหลือ 4 ระดับความเข้ม หรือให้เหลือ 2 บิตต่อพิกเซล เริ่มด้วยการสร้างฮิสโตแกรมของข้อมูลข้างต้นจะได้ลักษณะฮิสโตแกรมดังต่อไปนี้

ระดับความเข้ม	จำนวนพิกเซล
0	**
1	**
2	*********
3	**********
4	*********
5	****
6	*****
7	********
8	*********
9	******

ถ้าต้องการแบ่งระดับความเข้มเป็น 4 ระดับจากพิกเซลทั้งหมด 65 พิกเซล จะแบ่งได้ช่วงละ 16.25 พิกเซล ดังนั้นช่วงที่เหมาะสมที่สุดจะเป็นดังนี้

จำนวนพิกเซล ในแต่ละช่วง	ระดับ ความเข้ม	จำนวนพิกเซล
	0	**
13	1	**
	2	*********
20	3	**********
	4	*********
	5	****
17	6	*****
	7	********
15	8	*********
	9	******

ใช้ค่ามัธยฐานของแต่ละช่วงแทนความเข้มใหม่ตามลำดับดังนี้คือ 2, 3,6 และ 8 ดังนั้นรูปใหม่ที่ได้จะเป็นดังนี้

0 3 2 1 3 0 2 1 3 2 3 1 2
1 3 2 1 2 2 1 3 0 3 1 2 1
1 3 1 2 1 3 0 1 3 0 2 1 3
1 3 1 2 0 2 2 0 0 2 2 1 0
0 0 1 1 3 3 2 1 2 0 0 3 0

ในการเก็บด้วยรหัสใหม่ซึ่งมีจำนวนระดับความเข้มน้อยลง จำนวนบิตที่ใช้แทนค่าระดับความเข้มจึงลดลงไปด้วย แต่ก็จำเป็นต้องมีตารางเก็บค่าระดับความเข้มใหม่กับรหัสที่ใช้ด้วย

5. การเข้ารหัสการแปลง

จากที่ได้กล่าวมาแล้วในบทที่ 1 ว่าการเข้ารหัสการแปลงคือการอัดข้อมูลในพื้นที่การแปลง ซึ่งได้จากการแปลงเชิงเส้น เพื่อแปลงข้อมูลจากพื้นที่สัญญาณให้อยู่ในพื้นที่การแปลง ข้อมูลหรือค่าสัมประสิทธิ์แต่ละตัวในพื้นที่การแปลงจะมีสหสัมพันธ์น้อยกว่าข้อมูลในพื้นที่สัญญาณทำให้สามารถลดการจัดเก็บข้อมูลรายละเอียดที่ซ้ำซ้อนกันลงได้ นอกจากนี้ในพื้นที่การแปลงจะมีข้อมูลสัมประสิทธิ์ที่มีพลังงานสูงรวมอยู่เป็นกลุ่ม ลักษณะเช่นนี้เรียกว่า Energy Compaction property จึงสามารถเลือกเฉพาะสัมประสิทธิ์กลุ่มที่มีพลังงานสูงซึ่งมีความสำคัญต่อการแปลงกลับของภาพมาเข้ารหัสแทนการเข้ารหัสสัมประสิทธิ์ทุกตัว จึงมีผลให้ใช้เนื้อที่ในการเก็บและเวลาที่ใช้ในการส่งข้อมูลน้อยลง

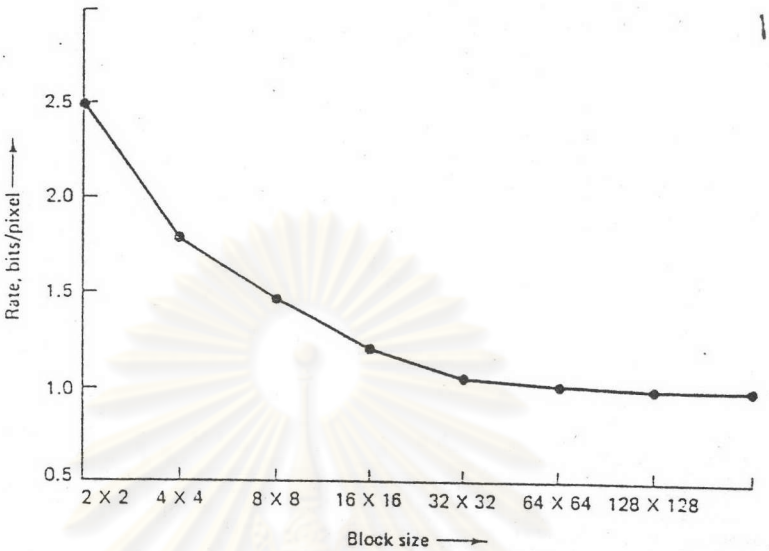
ลำดับต่อไปจะเป็นขั้นตอนการเข้ารหัสการแปลงโดยละเอียด(Wintz 1972, Gregary 1991)

5.1 ขั้นตอนการเข้ารหัสการแปลง

5.1.1 แบ่งรูปภาพเป็นพื้นที่สี่เหลี่ยมขนาดย่อย

เพื่อให้สามารถทำการเข้ารหัสการแปลงได้สะดวกขึ้นจะแบ่งรูปภาพออกเป็นพื้นที่สี่เหลี่ยมขนาดย่อยหลายๆรูป แล้วทำการเข้ารหัสการแปลงในพื้นที่ย่อยเหล่านี้แทนรูปภาพซึ่งมีขนาดใหญ่กว่ามากๆ เป็นการช่วยลดเวลาในการคำนวณของการแปลงเชิงเส้น นอกจากนี้การเข้ารหัสการแปลงในแต่ละพื้นที่สี่เหลี่ยมย่อยจะช่วยลดความยุ่งยากซับซ้อนเมื่อนำมาทำโดยฮาร์ดแวร์ รูปที่ 9 แสดงความสัมพันธ์ระหว่างขนาดของพื้นที่สี่เหลี่ยมย่อยและจำนวนบิตต่อพิกเซลของการเข้ารหัสการแปลงกับข้อมูลที่ได้จากการสุ่มแบบ Gaussian จากกราฟแสดงให้เห็นว่าการเข้ารหัสการแปลงกับพื้นที่สี่เหลี่ยมขนาดใหญ่มากไม่มีผลช่วยให้อัตราส่วนการอัดเพิ่มมากขึ้น

JPEG ได้เลือกใช้พื้นที่สี่เหลี่ยมขนาด 8 x 8 เพื่อประหยัดเวลาในการคำนวณ โดยผลจากการวิจัยพบว่าการเพิ่มขนาดของพื้นที่สี่เหลี่ยมมากกว่านี้ไม่ช่วยให้อัตราส่วนการอัดดีขึ้น(Anil n.d.)



รูปที่ 9 แสดงความสัมพันธ์ระหว่างขนาดของพื้นที่สี่เหลี่ยมย่อยและจำนวนบิตต่อพิกเซลของการเข้ารหัสการแปลงกับข้อมูลที่ได้จากการสุ่มแบบ Gaussian

5.1.2 ทำการแปลงเชิงเส้นกับข้อมูลในแต่ละพื้นที่สี่เหลี่ยมย่อย

หัวใจสำคัญของขบวนการอัดข้อมูลแบบเข้ารหัสการแปลงนี้ก็คือ การแปลงเชิงเส้นซึ่งใช้กันมากในการวิเคราะห์สัญญาณดิจิตอล ตัวอย่างเช่นการแปลงระดับความต่างศักย์ในโดเมนของเวลามาเป็นโดเมนของความถี่ เพื่อให้ง่ายต่อการวิเคราะห์ โดยที่สามารถแปลงกลับรูปแบบเดิมได้อย่างสมบูรณ์ สำหรับการแปลงข้อมูลรูปภาพมาอยู่ในโดเมนของความถี่ซึ่งสามารถตัดค่าสัมประสิทธิ์บางส่วนทิ้งได้โดยไม่มีผลกระทบต่อรูปภาพที่แปลงกลับมากนัก โดยอาศัยหลักการนี้จึงนำมาใช้กับการอัดข้อมูลรูปภาพ

การแปลงเชิงเส้นที่ใช้ในการทำการเข้ารหัสการแปลงในสัญญาณแบบดิจิตอลมีอยู่หลายวิธีเช่น Principal Component Transform (Karhunen-Loeve Transform), Discrete Fourier Transform, Discrete Cosine Transform, Hadamard หรือ Harr Transform (Ahmed 1974, Lynch 1991, Peason 1991) ฯลฯ Karhunen-Loeve Transform นับว่าเป็นการแปลงที่ให้ผลดีที่สุดในแง่ที่ให้ค่าเฉลี่ยของความผิดพลาดกำลังสอง (Mean Square Error) ต่ำที่สุดเมื่อเปรียบเทียบกับการแปลงชนิดอื่นๆในกรณีที่ต้องแปลงข้อมูลกลับมาอยู่ในพื้นที่สัญญาณโดยใช้ข้อมูลในพื้นที่การแปลงเพียงบางส่วน แต่วิธีนี้ไม่เหมาะสมที่จะนำมาใช้ในทางปฏิบัติ

เนื่องจากต้องการข้อมูลทางสถิติซึ่งบางครั้งไม่สามารถทราบได้ หรือไม่สามารถประมาณได้ถูกต้อง นอกจากนั้นยังเป็นการคำนวณที่ซับซ้อนยุ่งยากอีกด้วย

การแปลงที่นิยมใช้ในการทำการเข้ารหัสการแปลง คือ Discrete Cosine Transform (DCT) ซึ่งเป็นการแปลงเชิงเส้นวิธีหนึ่งที่มีคุณสมบัติเด่นคือ ในทางทฤษฎี DCT จะให้ประสิทธิภาพในการอัดข้อมูลใกล้เคียงกับ Optimum Transformation ส่วนในทางปฏิบัติ ขั้นตอนวิธีในการคำนวณหาค่าสัมประสิทธิ์ของ DCT สามารถคำนวณอย่างลัดได้ด้วยวิธี Fast Cosine Transform (FCT)ในทำนองเดียวกับ Fast Fourier Transform (FFT) ซึ่งมีจำนวนครั้งในการคำนวณน้อย จึงช่วยประหยัดเวลาได้เป็นอย่างมาก นอกจากนี้แล้วค่าสัมประสิทธิ์ของ DCT ยังเป็นเลขจำนวนจริง ทำให้ง่ายต่อการนำไปใช้งาน

DCT ใน 2 มิติ สำหรับภาพขนาด N x N pixel แสดงให้เห็นด้งสมการต่อไปนี้

$$X(n,m) = 2/N\ e(n,m)\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}x(k,l)\ C_{2N}^{(2k+1)n}\ C_{2N}^{(2l+1)m} \qquad (1)$$

และ Inverse Discrete Cosine Transform (IDCT) แสดงให้เห็นด้งสมการต่อไปนี้

$$x(k,l) = 2/N\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}e(n,m)\ X(n,m)\ C_{2N}^{(2k+1)n}\ C_{2N}^{(2l+1)m} \qquad (2)$$

เมื่อ $n,m,k,l = 0,1,\ldots,N-1$; $C_{2N}^{(2k+1)n} = \cos[\pi(2k+1)n/(2N)]$;

$e(n,m) = e(n)*e(m)$;

$$e(n) = \begin{cases} 1/\sqrt{2} & \text{ถ้า } n \text{ เท่ากับ } 0 \\ 1 & \text{ถ้า } n \text{ ไม่เท่ากับ } 0 \end{cases}$$

5.1.3 การคัดเลือกสัมประสิทธิ์

สัมประสิทธิ์ที่ถูกคัดเลือกเพื่อทำการเข้ารหัสต่อไปก็คือ สัมประสิทธิ์ที่มีค่าสัมบูรณ์สูงๆ เพราะสัมประสิทธิ์เหล่านี้มีความสำคัญต่อการกระจายกลับสู่รูปภาพหลังการทำ Inverse Transformation การคัดเลือกค่าสัมประสิทธิ์ทำได้ 2 วิธีคือ การคัดเลือกสัมประสิทธิ์แบบเข้ารหัสตามโซน(Zonal Coding)และใช้การคัดเลือกสัมประสิทธิ์ด้วยการเข้ารหัสแบบขีดแบ่ง

5.1.3.1 การเข้ารหัสตามโซน

การคัดเลือกสัมประสิทธิ์แบบเข้ารหัสตามโซน(Anil n.d.) เป็นการคัดเลือกเฉพาะค่าสัมประสิทธิ์ที่อยู่ภายในโซนที่กำหนดไว้เพื่อนำมาเข้ารหัส วิธีการนี้มีข้อจำกัดคือต้องมีการกำหนดโซนความถี่ให้เหมาะสมกับความถี่ของรูปภาพไว้ล่วงหน้า ดังนั้นการคัดเลือกสัมประสิทธิ์ของภาพความถี่ต่างๆ ด้วยโซนที่เจาะจงเพียงโซนเดียวจึงมักไม่ให้ผลดี

5.1.3.2 การเข้ารหัสแบบขีดแบ่ง

การคัดเลือกสัมประสิทธิ์โดยการเข้ารหัสแบบขีดแบ่ง(Anil n.d.) เป็นการคัดเลือกเฉพาะค่าสัมประสิทธิ์ที่มีค่าสูงกว่าค่าขีดแบ่งที่ตั้งไว้มาเข้ารหัส วิธีการนี้มีข้อดีคือสามารถใช้ได้กับภาพที่มีความถี่ต่างๆกันเนื่องจากสัมประสิทธิ์ที่มีความสำคัญสูงต่อการกระจายกลับสู่รูปภาพเดิมที่ตำแหน่งต่างๆ จะถูกคัดเลือกไว้ได้โดยไม่ขึ้นอยู่กับความถี่ของภาพ แต่วิธีนี้ก็มีข้อเสียคือ จะต้องส่งตำแหน่งของสัมประสิทธิ์ไปพร้อมกับตัวสัมประสิทธิ์ด้วยเสมอ จึงมีผลทำให้อัตราส่วนการอัด(Compression Ratio) ไม่ดีเท่าการคัดเลือกสัมประสิทธิ์แบบเข้ารหัสตามโซน

5.1.4 การควอนไทซ์ค่าสัมประสิทธิ์

เมื่อคัดเลือกสัมประสิทธิ์ที่ต้องการได้แล้ว จะทำการควอนไทซ์ค่าสัมประสิทธิ์เหล่านั้นเพื่อให้สามารถแทนค่าได้ด้วยจำนวนบิตที่กำหนดไว้ก่อนที่จะทำการเข้ารหัสต่อไปโดยต้องคำนึงถึงคุณภาพของรูปเป็นสำคัญ ตัวอย่างเช่นผลจากการแปลงดีซีทีของข้อมูลรูปภาพขนาด 8 บิตหรือ 256 ระดับความเข้ม จะให้ค่าสัมประสิทธิ์การแปลงอยู่ในช่วง

ระหว่าง -1,024 ถึง 1,023 หรือจะต้องใช้จำนวนบิตเท่ากับ 11 บิตในการแทนค่า วิธีการลดจำนวนบิตเหล่านี้ทำได้ด้วยการควอนไทซ์ซึ่งเป็นการลดความถูกต้องของค่าสัมประสิทธิ์ลงไป ขั้นตอนนี้เองที่ทำให้การอัดข้อมูลนี้เป็นการอัดข้อมูลแบบย้อนกลับไม่สมบูรณ์บางครั้งเรียกการทำควอนไทซ์นี้ว่าเป็นการทำ Many-to-one Mapping ตัวอย่างของการควอนไทซ์แบบแบ่งเป็นลำดับขั้นเป็นดังนี้

$$F^{Q}(u,v) = \text{Integer Round } (F(u,v)/Q(u,v))$$

การ Dequantization ก็ทำได้โดย

$$F^{Q'}(u,v) = F^{Q}(u,v) \times Q(u,v)$$

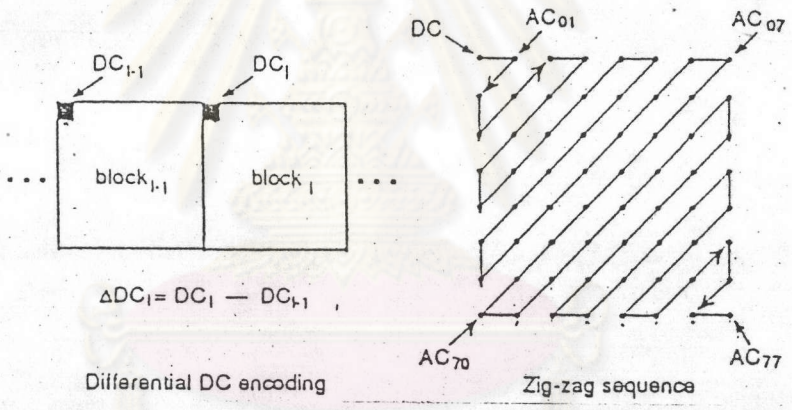
เนื่องจากสัมประสิทธิ์เหล่านี้มีค่าความแปรปรวนสูง การใช้ค่าควอนไทซ์เพียงค่าเดียวจึงให้ผลไม่ดีเท่าที่ควร ปัญหานี้แก้โดยทำการควอนไทซ์ด้วยค่า ส่วนเบี่ยงเบนมาตรฐาน เสมือนเป็นการทำ Normalized Coefficient เพื่อให้ใช้จำนวนบิตในการจัดเก็บเท่ากันหมด

JPEG ใช้วิธีการสร้างตารางค่าสำหรับควอนไทซ์ในขณะที่เรียกใช้โปรแกรม โดยได้ทำการทดลองเพื่อหาค่าที่เหมาะสมเพื่อใช้ในการควอนไทซ์จนกระทั่ง ISO ได้พัฒนาขึ้นเป็นมาตรฐานใช้สำหรับ JPEG โดยสามารถเลือกระดับคุณภาพได้ ตารางควอนไทซ์นี้สามารถเลือกระดับคุณภาพของผลจากการอัดได้ ถ้าเลือกระดับคุณภาพดีอัตราส่วนการอัดก็จะด้อยลงไป ในทางกลับกันถ้าเลือกระดับคุณภาพต่ำอัตราส่วนการอัดก็จะดี ทั้งนี้ขึ้นอยู่กับความพอใจของผู้ใช้งาน

5.1.5 การจัดเก็บหรือส่งสัมประสิทธิ์ดีซี

สัมประสิทธิ์ดีซีเป็นค่าที่มีความสำคัญมากต่อคุณภาพของรูปภาพที่ได้จากการกระจายกลับ เพราะแสดงถึงค่าเฉลี่ยความเข้มของข้อมูลทั้งพื้นที่สี่เหลี่ยมและยังมีค่าสูงกว่าสัมประสิทธิ์เอซีมาก ข้อมูลสัมประสิทธิ์ดีซีจะถูกจัดเก็บหรือส่งแยกจากขบวนการเข้ารหัสของสัมประสิทธิ์เอซี ภาพความถี่ต่ำมีค่าสัมประสิทธิ์ดีซีในแต่ละพื้นที่สี่เหลี่ยมที่ติดกันไม่ห่างกันมากเนื่องจากแต่ละพื้นที่สี่เหลี่ยมมีระดับความเข้มใกล้เคียงกัน จึงสามารถจัดเก็บหรือส่งสัมประสิทธิ์ดีซีโดยใช้ค่าผลต่างระหว่างพื้นที่สี่เหลี่ยมที่ติดกันได้เพื่อลดขนาดของข้อมูลที่จัดเก็บหรือส่ง เทคนิคในการจัดลำดับบล็อคเพื่อหาค่าผลต่างวิธีหนึ่งคือการจัดลำดับแบบ Zig-Zag

ดังรูปที่ 10 ทั้งนี้ก็เพื่อให้ได้ความเข้มที่ต่อเนื่องกันมากที่สุด ภาพความถี่สูงมีค่าสัมประสิทธิ์ดีซีในแต่ละพื้นที่สี่เหลี่ยมแตกต่างกันมากทำให้การใช้วิธีเดียวกับภาพความถี่ต่ำที่ได้กล่าวมาแล้วนั้นไม่เหมาะสม งานวิจัยครั้งนี้จึงจำเป็นต้องส่งสัมประสิทธิ์ดีซีไปโดยตรงเพื่อให้เป็นตามวัตถุประสงค์



รูปที่ 10 แสดงลำดับการจัดเก็บแบบ Zig-Zag

5.1.6 <u>การเข้ารหัส</u>

ขั้นตอนสุดท้ายสำหรับการทำการเข้ารหัสการแปลงคือการเข้ารหัสแบบย้อนกลับได้กับค่าสัมประสิทธิ์ที่ได้รับการคัดเลือก JPEG ได้เสนอการเข้ารหัสในขั้นตอนนี้ไว้ 2 วิธีคือ ฮัฟแมน และ Arithmetic coding