การส่งภาพที่ถูกลดขนาดด้วยวิธีโซนเกรสซีฟแบบมีการเข้ารหัสของโซน

PROGRESSIVE TRANSMISSION OF COMPRESSIONED IMAGE USING ZONAL CODING

กฤช ปานุโสรน
KRIT PANUSORONE

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
จุฬาลงกรณ์มหาวิทยาลัย
พ.ศ. 2536

ISBN 974-621-093-9

การส่งภาพที่ถูกลดข้อมูลด้วยวิธีโปรเกรสซีพแบบการเข้ารหัสของโซน

PROGRESSIVE TRANSMISSION OF COMPRESSIONED IMAGE USING ZONAL CODING

กฤษณ์ ภาณุโสภณ
KRIT PANUSOPONE

อาจารย์ที่ปรึกษา
รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์
ASSOC.PROF.DR.FUSAK CHEEVASUVIT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2536

ISBN 974-621-093-9

PROGRESSIVE TRANSMISSION OF COMPRESSIONED IMAGE USING ZONAL CODING

KRIT PANUSOPONE

ADVISOR

ASSOC.PROF.DR.FUSAK CHEEVASUVIT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

GRADUATE SCHOOL

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

1993

ISBN 974-621-093-9

บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การส่งภาพที่ถูกลดข้อมูลด้วยวิธีโปรเกรสซีพแบบการเข้ารหัสของโซน
PROGRESSIVE TRANSMISSION OF COMPRESSIONED IMAGES
USING ZONAL CODING

ชื่อนักศึกษา นายกฤษณ์ ภาณุโสภณ รหัสประจำตัว 35620001
หลักสูตร วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชา วิศวกรรมไฟฟ้า
ภาควิชา วิศวกรรมไฟฟ้า
อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
รศ.ดร.ฟูศักดิ์	ชีวสุวิทย์	
ผศ.ดร.กอบชัย	เดชหาญ	
อาจารย์เกษตร์	ศิริสันติสัมฤทธิ์	
ผศ.ดร.บุญวัฒน์	อัตชู	
รศ.ดร.มนัส	สังวรศิลป์	

ค่าระดับคะแนนที่เป็นเอกฉันท์จากคณะกรรมการสอบ GOOD
วัน/เดือน/ปี ที่สอบ 16 พฤษภาคม 2537 เวลา 12.00 น. ถึงเวลา 13.00 น.
สถานที่สอบ ห้อง (A-307) คณะวิศวกรรมศาสตร์

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์)
คณบดีบัณฑิตวิทยาลัย

วันที่ 30 เดือน พฤษภาคม พ.ศ. 2537

หมายเหตุ การวัดผลวิทยานิพนธ์ให้ใช้ค่าระดับคะแนนดังนี้

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

หัวข้อวิทยานิพนธ์ การส่งภาพที่ถูกลดข้อมูลด้วยวิธีโปรเกรสซีพแบบการเข้ารหัสของโซน
นักศึกษา นายกฤษณ์ ภาณุโสภณ
อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์
ระดับการศึกษา วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า
ภาควิชา วิศวกรรมไฟฟ้า สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2536

บทคัดย่อ

ปัญหาหลักในการประมวลผลภาพในขณะนี้คือเรื่องขนาดของข้อมูล ข้อมูลที่ใช้แสดงภาพเป็นข้อมูลที่จัดว่ามีขนาดใหญ่มากเมื่อเทียบกับข้อมูลชนิดอื่น ดังนั้นในการใช้งานที่มีความจำเป็นต้องมีการจัดเก็บหรือการสื่อสารจึงมีปัญหาเกี่ยวกับเวลาและค่าใช้จ่ายในการทำงานมากขึ้นในกรณีการสื่อสารภาพที่ใช้คนเป็นผู้ตัดสินใจ มักจะต้องการได้รับภาพที่ไม่คมชัดมากนักเพราะไม่ต้องการที่จะเสียเวลารอคอยนาน ในกรณีนี้การส่งภาพแบบโปรเกรสซีพจึงเป็นวิธีการสื่อสารภาพแบบใหม่ที่ทำให้ผู้ใช้สามารถเข้าใจภาพได้ในเวลาอันสั้น ในการส่งตามระบบนี้ทางด้านส่งจะทยอยส่งข้อมูลไปให้ทางด้านรับ โดยจะเริ่มส่งข้อมูลในอัตราบิทต่ำๆก่อน เมื่อผู้ใช้ได้รับภาพแล้วยังต้องการข้อมูลเพิ่มเติมก็สามารถขอให้ทางด้านส่งทำการส่งข้อมูลที่มีรายละเอียดเพิ่มขึ้นตามมาจนกระทั่งผู้ใช้พอใจ อย่างไรก็ตามเพื่อให้ผู้ใช้สามารถเห็นภาพที่มีรายละเอียดชัดเจนมากขึ้นในการรับภาพแต่ละครั้ง วิทยานิพนธ์ฉบับนี้จึงได้ทำการเปลี่ยนวิธีการจัดเรียงลำดับความสำคัญของข้อมูลใหม่ให้เหมาะกับลักษณะการกระจายพลังงานของข้อมูล จากการส่งภาพตามวิธีการใหม่ภาพที่ได้รับจะมีความคมชัดมากขึ้นในการรับภาพแต่ละครั้ง

THESIS TITLE	PROGRESSIVE TRANSMISSION OF COMPRESSIONED IMAGE USING ZONAL CODING
STUDENT	MR.KRIT PANUSOPONE
THESIS ADVISOR	ASSOC.PROF.DR.FUSAK CHEEVASUVIT
LEVEL OF STUDY	MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING
DEPARTMENT	ELECTRICAL ENGINEERING KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
YEAR	1993

Abstract

An enormous amount of data to represent an image is the crucial obstacle for all digital image processing techniques. From the large amount of image information, some applications which need to store or transmit image data confront with higher processing time and overhead cost. For Interactive Image Transmission that the end user is human eye usually needs only quite clear image in a short period. Progressive Image Transmission will be the optimum alternative for this constraint. In Progressive Image Transmission, a low-grade image is first transmitted with low bit-rate. Upon the user's request, the image can progressively be improved with furthur transmission. This thesis, however, arranges the priority area of transmission data to match energy distribution of data. The results show the preferable performance in each of the transmission.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงลงได้ด้วยความช่วยเหลือและสนับสนุนจากบุคคลหลายฝ่าย ในเบื้องแรกผู้เขียนขอกราบขอบพระคุณคุณพ่อและคุณแม่ที่ช่วยปลูกฝังแนวความคิดและให้กำลังใจจนผู้เขียนเกิดมีมานะในการศึกษาต่อจนถึงระดับนี้ นอกจากนี้ท่านทั้งสองยังคอยห่วงใยและให้ความช่วยเหลือในด้านต่างๆแก่ผู้เขียนตลอดเวลา

ผู้เขียนขอกราบขอบพระคุณ รองศาสตราจารย์ ดร. ฟูศักดิ์ ชีวสุวิทย์ อาจารย์ที่ปรึกษาซึ่งผู้เขียนถือว่าเป็นครูผู้ให้โลกทัศน์ใหม่ในด้านการวิจัยและด้านการประมวลผลภาพแก่ตัวผู้เขียน ยิ่งกว่านี้ท่านอาจารย์ยังได้กรุณาให้คำปรึกษาและชี้แนะแนวทางแก้ไขปัญหาต่างๆ จนทำให้ผู้เขียนมีความสามารถที่จะทำการวิจัยจนสำเร็จลุล่วงได้

ท้ายที่สุดผู้เขียนขอขอบคุณพี่ๆ,เพื่อนๆ และน้องๆ ทุกคนที่มีส่วนช่วยเหลือผู้เขียนในด้านต่างๆ โดยเฉพาะพี่ที่ห้องวิจัยที่ได้เป็นแบบอย่างที่ดีแก่ผู้เขียน และคุณกรรธิมา สาริกาที่คอยเป็นธุระให้ผู้เขียนในยามที่ผู้เขียนติดภารกิจและยังช่วยตรวจทานและขัดเกลาเนื้อหาในวิทยานิพนธ์ฉบับนี้ให้ด้วย

สารบัญ

สารบัญรูป

บทที่ 1

บทนำ

ในปัจจุบันการรับ-ส่งข้อมูลภาพผ่านสายสื่อสารระหว่างจุดต่างๆกำลังได้รับความนิยมเป็นอย่างมาก เพราะการส่งด้วยวิธีนี้สามารถลดระยะเวลาและค่าใช้จ่ายโดยเฉลี่ยลงไปได้มาก อย่างไรก็ตามข้อมูลที่มีรายละเอียดสูง เช่น รูปภาพ ยังคงต้องใช้ระยะเวลาในการสื่อสารนานมาก โดยเฉพาะอย่างยิ่งเมื่อนำไปประยุกต์ใช้กับการสื่อสารแบบโต้ตอบ(Interactive Communication) ระยะเวลาสื่อสารที่ยาวนานย่อมแสดงว่าผู้ใช้ต้องรอเป็นเวลานานกว่าจะตัดสินใจว่าเป็นภาพที่ต้องการหรือไม่ ดังนั้นจึงได้มีความพยายามจากหลายฝ่ายที่จะปรับปรุงคุณภาพในการสื่อสารผ่านช่องสัญญาณความถี่แคบนี้ วิธีหนึ่งซึ่งกำลังได้รับความสนใจเป็นอย่างมากในการแก้ปัญหาเรื่องเวลาในการส่งรูปภาพก็คือ การส่งภาพแบบโปรเกรสซีพซึ่งจะให้ผู้ใช้เป็นคนตัดสินใจว่าต้องการรายละเอียดเพิ่มเติมอีกหรือไม่ ถ้าหากภาพนั้นเป็นภาพที่ไม่ต้องการก็สามารถสั่งให้หยุดภาพที่กำลังแสดงอยู่ และให้ส่งรายละเอียดของภาพถัดไปมาแทน ทำให้ผู้ที่รู้รายละเอียดของภาพอยู่บ้างแล้วสามารถเลือกใช้ภาพจากกระบวนการนี้ได้อย่างรวดเร็วกว่าวิธีปกติมาก

1.1วัตถุประสงค์ของวิทยานิพนธ์

จุดประสงค์หลักในการส่งภาพแบบโต้ตอบคือการทำให้ผู้ใช้สามารถตัดสินใจเลือกภาพที่เหมาะสมโดยใช้เวลาอันสั้นที่สุด การส่งภาพแบบโปรเกรสซีพเป็นการส่งภาพเป็นชุดที่มีความคมชัดเพิ่มขึ้นเรื่อยๆ ประสิทธิภาพในการส่งวิธีนี้ขึ้นอยู่กับความสามารถในการจัดส่งภาพให้ผู้รับสามารถเข้าใจได้เร็วที่สุดหรือใช้การส่งน้อยครั้งที่สุดนั่นเอง ดังนั้นการส่งภาพแบบโปรเกรสซีพจึงเปรียบเสมือนเป็นการลดข้อมูลภาพอย่างหนึ่งด้วย การส่งภาพแบบโปรเกรสซีพที่ดีจึงต้องพยายามส่งภาพส่วนที่มีความสำคัญมากที่สุดไปให้ด้านรับก่อนนั่นเอง ในการจัดหาความสำคัญในภาพสามารถทำได้หลายวิธี แต่ในวิทยานิพนธ์ฉบับนี้ได้นำหลักการแยกแยะภาพกับการวิเคราะห์ข้อมูลในโดเมนความถี่มาประยุกต์ในการหาพื้นที่ที่สำคัญในภาพ โดยจะเริ่มจากการลดข้อมูลภาพที่ได้จากการแยกแยะหาส่วนที่มีความสำคัญในภาพ และทำการจัดเลือกส่งข้อมูลโดยวิเคราะห์ข้อมูลในโดเมนความถี่ ในหลักการนี้จะสามารถเลือกพื้นที่ส่วนสำคัญมาใช้ได้ดีกับภาพทั่วไปเนื่องจากระบบจะมีความอ่อนตัวสูง

1.2ขอบเขตของการวิจัย

ในส่วนของการวิจัยได้เสนอเทคนิคการลดข้อมูลภาพวิธีต่างๆ เพื่อที่จะนำไปประยุกต์ใช้กับการส่งภาพแบบโปรเกรสซีพให้มีประสิทธิภาพดียิ่งขึ้น รายละเอียดการวิจัยของวิทยานิพนธ์ฉบับนี้ได้จัดแบ่งออกเป็น 6 บท โดยที่แต่ละบทมีหัวข้อและเนื้อหาดังต่อไปนี้

บทที่ 1 เป็นบทนำ กล่าวถึงวัตถุประสงค์และขอบเขตของวิทยานิพนธ์

บทที่ 2 เป็นทฤษฎีเบื้องต้น ในบทนี้ได้กล่าวถึงทฤษฎีที่เกี่ยวข้องกับการประมวลผลภาพบางประการ เพื่อให้เกิดความเข้าใจในขั้นตอนที่เกี่ยวข้องและจำเป็นต้องนำไปใช้เป็นพื้นฐานในบทต่อๆไป โดยจะเริ่มตั้งแต่การเกิดภาพ การลดข้อมูลภาพทั่วๆไป ทฤษฎีการแปลง ทฤษฎีการแควนไตซ์ และการแยกแยะภาพ

บทที่ 3 เป็นการลดข้อมูลภาพโดยใช้การแปลง กล่าวถึงความสัมพันธ์ของข้อมูลภาพระหว่างโดเมนSpatialกับโดเมนความถี่ และได้กล่าวถึงองค์ประกอบที่นำมาใช้กับการลดข้อมูลภาพโดยใช้การแปลงนอกจากนี้ยังได้สรุปวิธีการและหลักการเบื้องต้นของการลดข้อมูลภาพโดยใช้การแปลงที่นิยมใช้กันด้วย

บทที่ 4 เป็นการส่งภาพแบบโปรเกรสซีพ กล่าวถึงลักษณะทั่วไปของการส่งภาพแบบโปรเกรสซีพ รวมทั้งวิธีการส่งภาพแบบโปรเกรสซีพต่างๆที่มีใช้กันอยู่ ทั้งวิธีการส่งภาพแบบโปรเกรสซีพตามลักษณะโครงสร้างปิรามิด และวิธีการส่งภาพแบบโปรเกรสซีพโดยใช้การแปลง

บทที่ 5 เป็นระบบการสื่อสารภาพแบบโปรเกรสซีพ กล่าวถึงวิธีการและเทคนิคการส่งภาพแบบโปรเกรสซีพที่ได้มีการปรับปรุงขึ้นมา โดยได้แยกอธิบายเป็น3ส่วนหลักคือ ส่วนการแยกแยะภาพเพื่อให้ได้พื้นที่ที่มีความเป็นเนื้อเดียวกันมากที่สุด ส่วนการเข้ารหัสภาพสำหรับเตรียมข้อมูลให้มีขนาดเล็กที่สุดสำหรับการสื่อสารในแต่ละครั้ง ส่วนการสื่อสารเป็นการอธิบายการเชื่อมต่อระบบและรูปแบบลักษณะของการส่งข่าวสาร

บทที่ 6 เป็นบทสรุป กล่าวถึงผลสรุปจากการวิจัยพร้อมทั้งได้เสนอแนะแนวทางการวิจัยที่จะพัฒนาต่อไป

ในส่วนของภาคผนวก ได้กล่าวถึงข่าวสารและEntropyที่เป็นหัวใจของทฤษฎีข้อมูล และสามารถใช้อธิบายข้อมูลชนิดต่างๆได้เป็นอย่างดี ซึ่งตัวแปรทั้งสองได้มีการอ้างถึงหลายครั้งในวิทยานิพนธ์ฉบับนี้ ส่วนในภาคผนวก ข ได้ใส่รายละเอียดของโปรแกรมการทำงานของระบบที่ได้ออกแบบขึ้นมาเพื่ออำนวยความสะดวกแก่ผู้ที่จะค้นคว้าวิจัยต่อไป

บทที่ 2

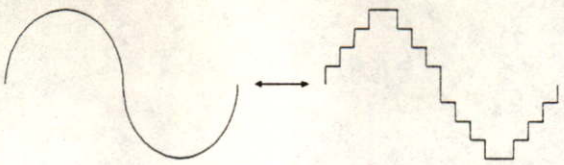
ทฤษฎีเบื้องต้น

2.1 ข้อมูลภาพ

การเกิดภาพโดยใช้เลนส์เป็นวิธีการพื้นฐานและพบทั่วไปในการถ่ายภาพ แสงที่ตกกระทบวัตถุและสะท้อนกลับมาที่เลนส์จะถูกรวบรวมที่เลนส์และนำมาแสดงยังจุดที่สอดคล้องกันกับวัตถุ ดังนั้นกระบวนการถ่ายภาพจึงเป็นการเปลี่ยนข้อมูลของวัตถุ 3 มิติมาเป็นข้อมูลของภาพใน 2 มิติ นอกจากนี้เรายังสามารถบันทึกหรือประมวลผลรูปแบบของแสงที่สะท้อนเข้ามาจากวัตถุได้โดยการใส่เซนเซอร์ลงไป โดยที่เซนเซอร์จะให้สัญญาณไฟฟ้าในรูปอนาลอกออกมา ในขั้นตอนนี้เซนเซอร์จะทำการสแกนหรือทำการวัดค่าผลรวมความเข้มของแสงที่จุดเล็กๆทีละจุดไปเรื่อยๆตามแนวทางที่กำหนดไว้หรือราสเตอร์(RASTER SCAN) ซึ่งปกติจะไล่จากซ้ายไปขวาและจากบนลงล่าง ค่าที่เซนเซอร์วัดได้นี้ถือว่าเป็นค่าความเข้มของภาพที่จุดนั้น กระบวนการถ่ายภาพทั้งหมดสามารถจัดให้เป็นเสมือนการทำคอนโวลูชั่นของความเข้มภาพ I(x,y) กับช่วงที่วัดที่จัดแบ่งเป็นลำดับ และภาพผลลัพธ์จะได้จากการสุ่มผลของการทำคอนโวลูชั่น สำหรับการถ่ายภาพนิ่งแล้วสัญญาณภาพจะถูกสแกนมาเพียงครั้งเดียว แต่ถ้าจัดภาพนิ่งที่ถ่ายจากมุมเดียวกันในเวลาต่างกันทีละน้อยมาจัดเรียงกันเป็นจำนวนมากและแสดงผลในช่วงเวลาอันสั้น ภาพที่เห็นจะมีลักษณะต่อเนื่องโดยที่ความกลมกลืนของลำดับภาพจะขึ้นอยู่กับความถี่ในการถ่ายภาพ

ดังนั้นข้อมูลภาพนิ่งจึงเป็นฟังก์ชั่น 2 มิติของความเข้มแสงหรือ f(x,y) เมื่อค่าหรือขนาดของฟังก์ชั่นที่พิกัด Spatial (x,y) จะแสดงความเข้มแสงของภาพที่จุดนั้น สำหรับภาพขาวดำเราจะแบ่งระดับความเข้มของภาพออกเป็น 256 ระดับสีเทา (Grey Scale) โดยที่ f(x,y)=0 จะแทนสีดำ และ f(x,y)=255 จะแทนสีขาว และค่าอื่นๆระหว่างขอบทั้งสองจะแทนค่าสีเทาที่ค่อยๆเปลี่ยนจากดำเป็นขาว เนื่องจากภาพที่ได้จากการสแกนเป็นภาพแบบอนาลอก โดยที่ค่าความเข้มในพิกัด Spatial หนึ่งจะมีลักษณะต่อเนื่อง(Continuous) ดังนั้นในการที่จะนำภาพมาประมวลผลที่คอมพิวเตอร์จึงจำเป็นต้องทำให้ภาพลักษณะต่อเนื่องกลายเป็นภาพดีสครีตเสียก่อน

ในการแสดงภาพดิจิตอลจากภาพอนาลอกจะต้องทำให้ข้อมูลแบ่งเป็นระดับต่างๆ ทั้งในพิกัด Spatial และที่ค่าความเข้มดังแสดงในรูป



รูปที่ 2.1 การแทนสัญญาณอนาลอกด้วยสัญญาณดิจิตอล

การดิจิไตซ์ (Digitization) เป็นวิธีการแปลงภาพต่อเนื่องให้เป็นดิจิตอลโดยจะเริ่มต้นจากการสุ่มภาพ (Sampling) ซึ่งตัวอย่างที่ได้จากการสุ่มภาพจะเป็นค่าระดับสีเทาของพิกัด Spatial ที่ห่างกันเป็นระยะที่เรียกว่าจุดภาพ (pixel) ซึ่งสามารถเขียนในรูปทางคณิตศาสตร์ดังนี้

$$f_s(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(x_i, y_i)\delta(x_i - i\delta(x), y_i - i\delta(y)) \qquad (2.1)$$

เมื่อ Delta Impulse Function คือ $\delta(x-n)$

หลังจากที่ผ่านการสุ่มมาแล้ว ค่าข้อมูลของภาพจะถูกควอนไทซ์ เพื่อให้ได้ค่าระดับสีเทาที่เป็นลำดับขั้น อย่างไรก็ตามการควอนไทซ์ในขั้นตอนนี้จะแยกแยะค่าของข้อมูลภาพไปอยู่ที่ระดับสีเทาต่างๆที่ห่างเท่ากัน

ภาพดิจิตอลที่นำมาประมวลผลในคอมพิวเตอร์สามารถเขียนในรูปเมตริกซ์ 2 มิติ โดยแต่ละมิติจะแทนพิกัดทาง Spatial และค่าสัมประสิทธิ์ของเมตริกซ์จะเท่ากับระดับสีเทาของจุดภาพนั้น ดังเช่นภาพดิจิตอลขนาด M*N จะสามารถเขียนในรูปเมตริกซ์ได้เป็น

$$\mathbf{I} = \begin{bmatrix} i(1,1) & i(1,2) & \cdots & i(1,M) \\ i(2,1) & i(2,2) & \cdots & i(2,M) \\ \vdots & \vdots & \ddots & \vdots \\ i(N,1) & i(N,2) & \cdots & i(N,M) \end{bmatrix} \qquad (2.3)$$

และสามารถเขียนในรูปเวกเตอร์ได้โดยการสแกนจากเมตริกซ์ทีละแถว(หรือทีละคอลัมน์) แล้วเขียนเรียงกันในแนวเดียว การเขียนข้อมูลในแบบเวกเตอร์จะอยู่ในรูปที่กะทัดรัดกว่าและสามารถนำไปประยุกต์ใช้กับผลของการประมวลผลสัญญาณเชิงดิจิตอล 1 มิติที่มีอยู่แล้วได้เลย

แม้ว่าจะมีการนำไปใช้งานต่างๆเป็นจำนวนมาก แต่เราสามารถแบ่งการประมวลผลกับภาพเป็นวิธีหลักๆ ได้ 2 แบบดังนี้

1.การประมวลผลเชิงจุด ซึ่งเป็นวิธีประมวลผลอย่างง่าย โดยจะทำการแปลงข้อมูลของภาพต้นแบบ และการแปลงจะขึ้นอยู่กับค่าระดับสีเทาของจุดภาพนั้นๆ เทคนิคที่จัดเป็นการประมวลผลเชิงจุด คือการควอนไทซ์

2.การประมวลผลเชิงกลุ่มข้างเคียง เป็นวิธีประมวลผลอีกวิธีหนึ่งซึ่งนอกจากจะนำค่าระดับสีเทาของจุดนั้นมาประมวลผลแล้วยังนำค่าระดับสีเทาของจุดภาพข้างเคียงมาประมวลผลด้วย ซึ่งจะต้องนำมาใช้ในการทำตัวกรองแบบต่างๆ

สำหรับภาพต้นแบบที่ใช้ในวิทยานิพนธ์ฉบับนี้ดังแสดงในรูปที่ 2.2 และ 2.3 โดยแต่ละภาพจะมีขนาด 128×128 จุดภาพ แม้ว่าขนาดภาพที่ใหญ่ขึ้นจะทำให้ภาพมีลักษณะเหมือนจริงมากยิ่งขึ้น แต่ก็ทำให้การประมวลผลยุ่งยากและใช้เวลานานมากขึ้นตามไปด้วย

รูปที่ 2.2 ภาพต้นแบบ "Krit"

รูปที่ 2.3 ภาพต้นแบบ "Baboon"

2.2 การลดข้อมูลภาพ

เนื่องจากภาพเป็นข้อมูลขนาดใหญ่ ดังนั้นในการจัดเก็บข้อมูลภาพต้องประสบปัญหาเกี่ยวกับเนื้อที่ในการจัดเก็บ แต่ในปัจจุบันเทคโนโลยีฮาร์ดแวร์จะพัฒนาไปมากทำให้สามารถเพิ่มขนาดหน่วยความจำได้มากกว่าเดิมมาก อาทิเช่น การเก็บข้อมูลใน CD-ROM เป็นต้น ดังนั้นปัญหาในการจัดเก็บจึงบรรเทาลงไป แต่ก็ยังเป็นอุปสรรคสำคัญในการนำภาพไปใช้งานต่างๆ นอกจากนี้ขนาดของภาพที่เทอะทะยังทำให้การสื่อสารภาพต้องใช้เวลานาน ซึ่งปัจจุบันความจุของช่องการสื่อสารมีความสำคัญและมีค่าใช้จ่ายสูงมาก ดังนั้นภาพที่มีรายละเอียดมากจะต้องเสียเวลาและค่าใช้จ่ายในการสื่อสารภาพมาก

อันที่จริงแล้ว ภาพเป็นข้อมูลที่มีขนาดใหญ่แต่ก็มีความซ้ำซ้อนสูงโดยธรรมชาติ ซึ่งสามารถเห็นได้จากจุดภาพที่อยู่ใกล้ชิดกันส่วนใหญ่จะมีค่าระดับสีเทาใกล้เคียงกัน ดังนั้นเมื่อเรารู้ค่าระดับสีเทาของจุดภาพหนึ่ง เราสามารถทำนายค่าระดับสีเทาของจุดภาพข้างเคียงได้ในเกือบทุกกรณี ซึ่งเราถือว่าข้อมูลลักษณะนี้จะมีตัวข่าวสารน้อยและในทาง Information Theory [1] แล้ว เราสามารถคำนวน Entropy [2] หรือจำนวนข่าวสารจริงที่มีอยู่ในแต่ละข้อมูล ซึ่ง Entropy ของข้อมูลภาพจะมีขนาดเล็กกว่าขนาดของข้อมูลภาพจริงๆมาก ดังตัวอย่างเช่น ภาพต้นแบบที่นำมาใช้ในวิทยานิพนธ์ที่ใช้เนื้อที่เก็บข้อมูล 8 บิทต่อ 1 จุดภาพ(bpp) แต่ในภาพที่ 2.2 จะมีEntropyเพียง 5.84 bpp ซึ่งแสดงว่าเราสามารถใช้เนื้อที่เพียง 5.84 bppในการเก็บข้อมูลภาพขนาด 8 bppอย่างถูกต้อง แต่ปัญหาหลักอยู่ที่ว่าจะทำอย่างไรจึงจะลดข้อมูลภาพให้อยู่ที่ Entropy Rate ได้ ดังนั้นจึงได้มีการคิดค้นวิธีการต่างๆเพื่อที่จะลดข้อมูลภาพลง (Image Compression) [3-6] ซึ่งเป็นการประยุกต์การลดข้อมูล (Data Compression) [7,8] มาใช้ ในการลดข้อมูลเราสามารถแบ่งเป็นวิธีหลักๆได้ 2 แบบคือ

2.2.1การลดข้อมูลโดยไม่มีความสูญเสีย

วิธีนี้จะมุ่งเน้นไปที่การรักษาความถูกต้องของข้อมูลไว้โดยพยายามลดความซ้ำซ้อนที่มีอยู่ให้หมดไปมากที่สุดหรือให้เท่ากับ Entropy Rate การลดข้อมูลวิธีนี้จะสามารถย้อนกลับได้ (Reversible) หรือสามารถคืนค่าข้อมูลจริงได้โดยทำการถอดรหัส วิธีนี้จะนิยมใช้กับข้อมูลที่มีความสำคัญมากและสูญเสียไม่ได้ เช่น ข้อมูลในคอมพิวเตอร์,ข้อมูลทางการแพทย์ เป็นต้น ในการประมวลผลภาพเราจะทำการลดข้อมูลวิธีนี้ภายหลังสุดเพื่อให้ข้อมูลอยู่ในรูปที่กะทัดรัดที่สุด เพื่อเป็นพื้นฐานสำหรับการศึกษาต่อไป จะอธิบายวิธีการลดข้อมูลโดยไม่มีความสูญเสีย 2 วิธีหลักดังนี้

1. การเข้ารหัสแบบ Huffman วิธีนี้ถูกพัฒนาขึ้นมาเพื่อการเข้ารหัสข้อมูลที่ไม่ขึ้นต่อกันทางสถิติเลยให้เป็นข้อมูลที่มีขนาดกะทัดรัดที่สุด โดยจะกำหนดขนาดของข้อมูลแต่ละตัวไม่เท่ากันเพื่อจะได้ใช้โอกาสของความไม่สม่ำเสมอได้เต็มที่ หรือกล่าวอีกนัยหนึ่ง พยายามจัดขนาดของข้อมูลให้ใช้

จำนวนบิทน้อยเมื่อข้อมูลนั้นมีความซ้ำซ้อนกันสูงและจัดจำนวนบิทให้มากสำหรับข้อมูลที่มีความซ้ำซ้อนกันต่ำโดยในวิธีการปฏิบัติจะมีขั้นตอนการทำงานคร่าวๆดังนี้

1.1 จัดเรียงความน่าจะเป็นในการพบข้อมูลแต่ละตัวในอันดับที่ลดลง

1.2 รวมความน่าจะเป็น2ตัวที่ต่ำสุดเข้าด้วยกันและจัดข้อมูลที่มีความน่าจะเป็นสูงกว่าอยู่ในขั้นบนจนกว่าจะรวมความน่าจะเป็นทั้งหมดเป็นหนึ่ง

1.3 กำหนดค่าศูนย์ให้กับสมาชิกตัวบนและค่าหนึ่งให้กับสมาชิกตัวล่างในการรวมตัวแต่ละคู่

1.4 กำหนดรหัสให้กับข้อมูลแต่ละตัวโดยบันทึกค่าของการรวมตัวของข้อมูลแต่ละตัวจนถึงการรวมตัวครั้งสุดท้าย

2. การเข้ารหัสแบบ Arithmatic เป็นการเข้ารหัสวิธีหนึ่งที่ให้ผลเข้าถึงค่าจำกัดของ Entropy วิธีนี้มีหลักการแตกต่างจากวิธีเข้ารหัสแบบ Huffman ที่พยายามกำหนดข้อมูลแต่ละตัวด้วยรหัสเฉพาะแต่ Arithmatic Coding นำสัญลักษณ์ของข้อมูลที่จะต้องการลดขนาดมาแทนด้วยเลขทศนิยมเพียงตัวเดียว ยิ่งข่าวสารที่จะนำมาลดข้อมูลมีความยาวและซับซ้อนมากขึ้นเท่าใด เลขทศนิยมของการเข้ารหัสก็จะละเอียดมากขึ้นเท่านั้น จุดประสงค์ของ Arithmatic Coding[9] คือการสร้างชุดสัญลักษณ์ให้อยู่ในช่วงๆหนึ่งตามข้อมูลที่นำมาเข้ารหัส การเข้ารหัสสัญลักษณ์ใหม่ถือเป็นการแบ่งช่วงของความน่าจะเป็นให้เป็นช่วงย่อยที่มีขนาดเล็กลงซึ่งช่วงย่อยนี้เป็นสัดส่วนกับความน่าจะเป็นของแต่ละชุดสัญลักษณ์

หลังจากที่ได้เข้ารหัสสัญลักษณ์จำนวนมาก ช่วง P ซึ่งเป็นผลรวมของความน่าจะเป็นที่สัญลักษณ์ทั้งหมดจะถูกใช้ และความละเอียดในแต่ละช่วงที่เท่ากับจำนวนบิทที่นำมาแสดงช่วงนั้นๆ จะมีค่าโดยประมาณเป็น $-\log_2 P$ และจาก

$$P = P_1 \times P_2 \times P_3 \times \ldots \times P_N \tag{2.4}$$

จำนวนบิทของความละเอียดหาได้จาก

$$-\log_2 P = -\{\log_2(P_1) + \log_2(P_2) + \log_2(P_3) + \ldots + \log_2(P_N)\} \tag{2.5}$$

ดังนั้นสามารถสรุปได้ว่าความยาวของสัญลักษณ์จะมีค่าใกล้เคียงกับความน่าจะเป็นในการเกิดสัญลักษณ์แต่ละตัว และค่าเฉลี่ยของจำนวนบิทที่ใช้แทนสัญลักษณ์ก็มีค่าเข้าใกล้ขีดจำกัดที่คำนวณจากEntropy

2.2.2 การลดข้อมูลภาพโดยยอมให้มีความสูญเสีย

การลดข้อมูลวิธีนี้มุ่งเน้นไปที่การลดข้อมูลให้มากที่สุดโดยยอมให้มีการสูญเสียข้อมูลไปบ้าง แม้ว่าวิธีนี้จะไม่สามารถย้อนกลับได้ (Irreversible) เนื่องจากในการถอดรหัสจะมีความสูญเสียจากการลดข้อมูลบ้าง แต่วิธีการนี้จะนิยมใช้ในการประมวลผลภาพทั่วไปเป็นอย่างมาก เนื่องจากสายตาคนซึ่งใช้เป็นตัวตัดสินใจเรื่องคุณภาพของภาพมีความตอบสนองต่อภาพได้ไม่ดีนัก ดังนั้นเราสามารถเลือกให้มีการสูญเสียข้อมูลในส่วนที่สายตาคนไม่สามารถตรวจจับได้ ทำให้สามารถใช้จำนวนบิทได้เล็กลงมากกว่า Entropy Rate เพราะมีการลดข่าวสารที่ไม่จำเป็นลงไป ในการลดข้อมูลภาพเราจะเริ่มต้นที่การลดข้อมูลโดยยอมให้มีความสูญเสียก่อนเพื่อพยายามที่จะลดข่าวสารให้ได้มากที่สุด วิธีการหลักที่ใช้ในการลดข้อมูลตามวิธีนี้แบ่งเป็น

1. Predictive Coding เป็นที่ทราบว่า ข้อมูลภาพมีความสัมพันธ์กันสูงมากระหว่างจุดภาพที่อยู่ใกล้เคียงกัน นั่นคือข่าวสารจำนวนมากของจุดที่กำลังพิจารณาสามารถทำนายได้จากข้อมูลของข่าวสารที่อยู่รอบๆ ซึ่งค่าจากการเข้ารหัสนี้สามรถคืนตัวได้จากการทำนายทางด้านรับ ดังจะเห็นได้จากเกรเดียนท์ของภาพซึ่งใช้แสดงลักษณะการเปลี่ยนแปลงในภาพนั้น พบว่าบริเวณทั่วไปเกรเดียนท์ของภาพจะมีค่าต่ำ แต่นอกจากบริเวณขอบแล้วข้อมูลที่อยู่รอบๆจะมีค่าคล้ายกัน

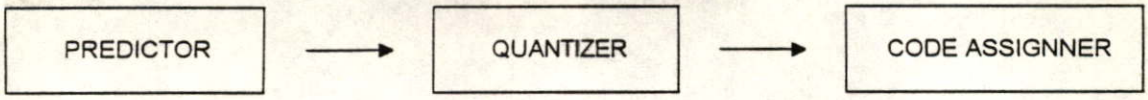
ตัวแปรที่สำคัญในการเข้ารหัสวิธีนี้คือ ตัวทำนาย ถ้าตัวทำนายนี้สามารถคาดหมายค่าของจุดต่อไปได้ใกล้เคียงมากแล้ว จะส่งผลให้ต้องส่งข้อมูลส่วนที่จำเป็นน้อยลง ตัวทำนายแบบพื้นฐานอย่างหนึ่งคือ ตัวทำนายเชิงเส้น ซึ่งตัวทำนายนี้จะประมาณค่าถัดไปจากค่าที่ผ่านมาและข้อมูลสถิติบางอย่างและสามารถกำหนดรูปแบบได้ดังนี้

$$\hat{x}_i = \rho x_{i-1} + (1-\rho)\overline{x} \tag{2.6}$$

เมื่อ $\hat{x}_i$ เป็นค่าประมาณของจุดภาพที่ตำแหน่ง i, x_{i-1} เป็นค่าของข้อมูลที่ผ่านมาที่ตำแหน่ง i-1, $\overline{x}$ เป็นค่าเฉลี่ยของภาพ และ ρ เป็นค่าสัมประสิทธิ์แสดงความสัมพันธ์ระหว่างจุดภาพที่อยู่ติดกันและหาได้จาก

$$\rho = \frac{E(x_i, x_{i-1})}{E(x_i^2)} \tag{2.7}$$

E หมายถึงค่าคาดหมายและสามารถแทนได้โดยค่าเฉลี่ย ถ้าในภาพมีความสัมพันธ์ต่อกันสูงแล้ว ρ จะเข้าใกล้1 เมื่อ $\rho = 1$ ตัวทำนายจะหาความแตกต่างจากจุดภาพปัจจุบันและจุดภาพก่อนหน้าเท่านั้น



รูปที่ 2.4 แสดงส่วนประกอบของ Predictive Coding

2. Transform Coding การเข้ารหัสโดยการแปลง (Transformation) เป็นความพยายามที่จะลดความสัมพันธ์ระหว่างจุดภาพที่มีอยู่มากกว่าการเข้ารหัสวิธีอื่น เมื่อความสัมพันธ์ระหว่างกันถูกกำจัดไปแล้ว ข้อมูลของข่าวสารที่ซ้ำซ้อนจะไม่ต้องมาใช้ซ้ำอีก ซึ่งเทคนิคนี้[10] จะประกอบด้วยวิธีการหลัก 2 วิธี

2.1 กระบวนการเข้ารหัสทางสถิติเป็นการแปลงเชิงเส้นซึ่งจะแปลงชุดข้อมูลที่เกี่ยวข้องกันไปเป็นชุดข้อมูลที่ไม่เกี่ยวข้องกันมากขึ้น

2.2 กระบวนการเข้ารหัสทาง Psychovisual จะเป็นการควอนไทซ์และเข้ารหัสให้กับสัมประสิทธิ์โดยจะใช้การตอบสนองของสายตาคนต่อผลเชิง Subjective ของความผิดพลาดในการ Quantize มากำหนดความละเอียดของการเข้ารหัส

การเข้ารหัสวิธีนี้สามารถลดข้อมูลได้มากที่สุดเมื่อเทียบกับวิธีอื่นๆที่สร้างความผิดพลาดให้กับภาพพอๆกัน อย่างไรก็ตาม Transform Coding จะมีวิธีการทำงานที่ยุ่งยากซับซ้อนมากกว่า

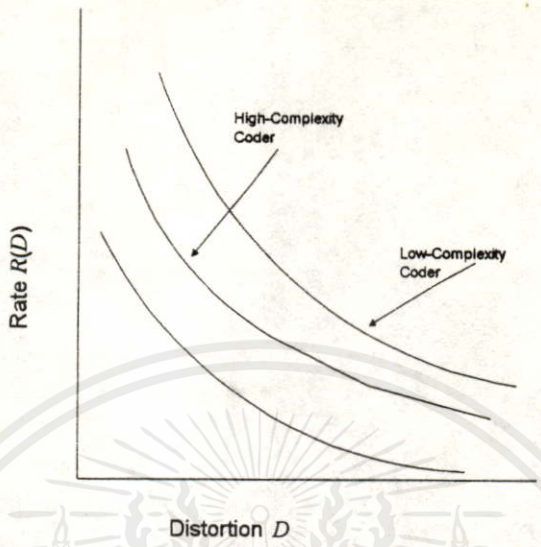
ในการลดข้อมูลแบบไม่มีความสูญเสียนั้น จะมีตัวแปรหนึ่งที่ใช้เป็นขีดจำกัดในการลดข้อมูลข่าวสาร ซึ่งตัวแปรนี้ถูกกำหนดจาก Entropy ของข่าวสารนั้น แต่ในการใช้งานบางประเภทยอมให้เกิดความสูญเสียบางส่วนที่กลับคืนมาไม่ได้ ทำให้เกิดปัญหาใหม่เกิดขึ้นในกรณีนี้คือ จำนวนบิทต่ำที่สุดที่ต้องใช้เข้ารหัสภาพโดยที่ภาพผลลัพธ์ต้องมีค่าสูญเสียอยู่ในระดับที่น่าพอใจ ปัญหานี้ได้มีการศึกษาอย่างกว้างขวางใน Rate Distortion Theory[11] และ Rate Distortion Theory ได้กำหนดขีดจำกัดของประสิทธิภาพในการลดข้อมูลภาพแบบมีความสูญเสียตามระดับของความถูกต้องที่ต้องการ สำหรับการวัดความผิดเพี้ยนทั่วไปของข้อมูลตามทฤษฎีจะกำหนด Rate Distortion Function, R(D) ที่มีคุณสมบัติดังต่อไปนี้

1.สำหรับความผิดเพี้ยน D ที่กำหนดให้แล้ว เป็นไปได้ที่จะหาหลักการเข้ารหัสที่ใช้ข้อมูลขนาดใกล้เคียงกับ R(D) เพื่อให้ได้ความผิดเพี้ยนเฉลี่ยเป็น D

2. เป็นไปไม่ได้ที่จะหารหัสที่สร้างข้อมูลให้มีความผิดเพี้ยน D เป็นอย่างต่ำ โดยใช้อัตราข้อมูลต่ำกว่า R(D)

รูปที่ 2.5 แสดง Rate Distortion Function ที่เป็นไปได้จริง[6] ขนาดของข้อมูลที่ต้องนำมาใช้ในการเข้ารหัสโดยไม่มีความผิดเพี้ยนของข่าวสารจะเป็นค่าของ R ที่ D=0 และขนาดของข้อมูลนี้จะน้อย

กว่าหรือเท่ากับEntropyของข่าวสารขึ้นอยู่กับความผิดเพี้ยนที่กำหนด ซึ่งโดยทั่วไปแล้ววิธีการเข้ารหัสที่ซับซ้อนของข้อมูลที่มีรูปแบบทางสถิติที่ดีจะมีประสิทธิภาพเข้าใกล้ขีดจำกัดของ R(D)



รูปที่ 2.5 แสดง Rate-Distortion Function

ในการลดข้อมูลโดยยอมให้มีความสูญเสียนี้ ประสิทธิภาพของเทคนิคต่างๆไม่สามารถสรุปได้จากอัตราการลดข้อมูลเพียงอย่างเดียว แต่ยังต้องพิจารณาถึงความสูญเสียข้อมูลที่เกิดขึ้นว่าอยู่ในช่วงที่ยอมรับได้ไหม ซึ่งเกณฑ์ที่จะนำมาใช้วัดคุณภาพของข้อมูลภาพจะแบ่งได้เป็น 2 หลักหญ่ๆ [12]คือ

1. วิธีทางนามธรรม (Subjective Criterion) เป็นวิธีพื้นฐานแต่ให้ผลน่าเชื่อถือมาก วิธีนี้จะใช้สายตาของผู้ไม่เกี่ยวข้องทำการตัดสินคุณภาพของภาพในฐานะที่เป็นผู้ใช้ธรรมดา และยังใช้สายตาของผู้เชี่ยวชาญที่มีความสามารถในการสังเกตความบกพร่องเล็กน้อยที่ผู้ใช้ทั่วไปอาจมองข้าม ในการตัดสินใจอาจจะใช้กฎเกณฑ์ต่างๆดังนี้

1.1 ตัดสินใจจากภาพทดสอบแล้วให้ระดับคะแนน ซึ่งเมื่อผู้ทดสอบได้มองภาพทดสอบแล้วจะระบุระดับคุณภาพตามที่ระบุไว้ โดยอาจมีชุดภาพมาตรฐานที่มีระดับคะแนนระบุไว้อ้างอิงด้วยก็ได้ ระดับคุณภาพสามารถแสดงได้ทั้งระดับความถูกต้องและระดับความผิดพลาดโดยข้อมูลต่อไปนี้จะแสดงระดับความถูกต้องและระดับความผิดพลาด

Overall Goodness Scale

5.Excellent

4.Good

3.Fair

2.Poor

1.Unsatisfactory

Impairment Scale

1.Not Perceptible

2.Just Perceptible

3.Definitely Perceptible but only slightly impairment to picture

4.Impairment to picture but not objectionable

5.somewhat objectionable

6.Definitely objectionable

7.Extremely objectionable

1.2 ตัดสินใจจากชุดภาพทดสอบแล้วเรียงลำดับคุณภาพของภาพ ซึ่งผู้ใช้จะเรียงลำดับภาพทดสอบตามคุณภาพของภาพที่เห็น ผู้ทดสอบจะตัดสินใจในลักษณะเปรียบเทียบว่าดีกว่าหรือแย่กว่า ต่อไปจะแสดงระดับคุณภาพเปรียบเทียบ

Group Goodness Scale

7.Best in group

6.Well above average for this group

5.Slightly above average for this group

4.Average for this group

3.Slightly below average for this group

2.Well below average for this group

1.Worst in group

2. วิธีทางรูปธรรม (Quantitative Criterion) ในวิธีนี้จะทำการคำนวนหาความผิดพลาดระหว่างภาพ 2 ภาพ โดยภาพหนึ่งจะถือว่าเป็นภาพต้นแบบ F และอีกภาพหนึ่งถือว่าเป็นภาพทดสอบ F' ดังนั้นจากภาพทั้งสองสามารถหาภาพผลต่าง E หรือความผิดพลาดได้จาก

$$F' = F + E \tag{2.8a}$$

หรือ

$$E = F' - F \tag{2.8b}$$

และในการหาค่าผิดพลาดเชิงตัวเลข จะนิยมใช้ค่าเฉลี่ยกำลังสองซึ่งได้จากการเฉลี่ยหรือการรวมภาพผลต่างที่ถูกยกกำลังสอง ดังนั้นค่า Mean Square Error(MSE) จะหาได้จาก

$$MSE = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} e^2(x,y)$$

$$MSE = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} [F'(x,y) - F(x,y)]^2 \tag{2.9}$$

แต่ในการใช้งานบางประเภท ผู้ใช้อาจต้องการทราบอัตราส่วนของสัญญาณต่อสัญญาณต่อสัญญาณรบกวน (Signal per Noise Ratio,SNR) ซึ่งหาได้จากการนำค่าเฉลี่ยของภาพต้นแบบที่ถูกยกกำลังสองแล้วหารด้วย MSE ดังนี้

$$SNR = 10\log_{10}\frac{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F^2(x,y)}{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} E^2(x,y)} \quad \text{dB} \tag{2.10}$$

อย่างไรก็ตาม ถ้าอาศัยเฉพาะความผิดพลาดเชิงตัวเลขมาตัดสินใจคุณภาพของภาพโดยตรงแล้ว อาจทำให้เกิดความผิดพลาดขึ้นได้ เพราะในบางกรณีที่ภาพทดสอบ 2 ภาพมีความผิดพลาดเท่ากันแต่ภาพหนึ่งมีความผิดพลาดในช่วงความถี่ต่ำซึ่งสายตาคนสังเกตไม่ชัด ในขณะที่อีกภาพหนึ่งมีความผิดพลาดในช่วงความถี่สูงซึ่งเห็นเด่นชัด ในกรณีนี้ภาพแรกดูจะเป็นภาพที่มีคุณภาพดีกว่า

2.3 การแปลง(Transformation)

การแปลงเป็นวิธีการทางคณิตศาสตร์วิธีหนึ่ง[13,14,15] ซึ่งเป็นที่ใช้กันอย่างแพร่หลายในการประมวลผลข้อมูลภาพ โดยการแปลงมักจะถูกใช้ในการประมวลผลภาพเป็นตัวเปลี่ยนข้อมูลทางโดเมน Spatial ไปเป็นข้อมูลในโดเมนอื่น ซึ่งส่วนใหญ่จะเป็นโดเมนความถี่ ข้อมูลภาพในโดเมนความถี่มีลักษณะเด่น 3 ประการที่การประมวลผลภาพต้องการ

1.ค่าที่ความถี่ศูนย์ (DC Component) เป็นค่าความเข้มเฉลี่ยของภาพ

2.ค่าที่ความถี่สูง (High Frequency Components) เป็นค่าที่บอกถึงขนาดและทิศทางของขอบในภาพ

3.ค่าที่ความถี่ต่ำ (Low Frequency Components) เป็นค่าที่บอกลักษณะโดยคร่าวๆของภาพ

ต่อไปจะกล่าวถึงคุณสมบัติที่สำคัญของการแปลงที่จะนำมาใช้ในการประมวลผลภาพ คือ คุณสมบัติ Separable และ Unitary

ถ้าพิจารณาข้อมูลภาพ 2 มิติ $f(x,y)$ ขนาด $N \times N$ ที่ถูกแปลงโดยตัวแปลงไปข้างหน้า (Forward Transform) ให้ได้ข้อมูลหลังการแปลง F(u,v) ที่มีมิติ $N \times N$ สามารถคำนวนได้จาก

$$F(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)a(x,y,u,v) \qquad (2.11)$$

เมื่อ $a(x,y,u,v)$ เป็นโครง(kernel)ของการแปลงไปข้างหน้า

เราสามารถคืนข้อมูลภาพ $f(x,y)$ กลับมาได้โดยทำการแปลงย้อนกลับกับข้อมูลหลังการแปลง $F(u,v)$ ซึ่งจะย้ายข้อมูลจากโดเมนความถี่มาเป็นโดเมน Spatial ซึ่งคำนวนได้จาก

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v)b(x,y,u,v) \qquad (2.12)$$

เมื่อ $b(x,y,u,v)$ เป็นโครงของการแปลงย้อนกลับ

การแปลงนี้จะมีคุณสมบัติ Unitary ถ้าสถานะ Orthonormal เหล่านี้เป็นจริง

$$\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} a(x,y,u,v)a^{*}(i,j,u,v) = \delta(x-i,y-j)$$

$$\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} b(x,y,u,v)b^{*}(i,j,u,v) = \delta(x-i,y-j)$$

$$\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} a(x,y,u,v)a^{*}(x,y,k,l) = \delta(u-k,v-l)$$

$$\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} b(x,y,u,v)b^{*}(x,y,k,l) = \delta(u-k,v-l) \qquad (2.13)$$

และตัวแปลงจะมีคุณสมบัติ Separable ถ้าสามารถเขียนให้อยู่ในรูป

$$a(x,y,u,v) = a_1(x,u)a_2(y,v) \qquad (2.14)$$

ถ้าตัวแปลงมีคุณสมบัติ Separable แล้ว การคำนวนจะสามารถทำได้เป็น2ขั้นตอนคือ

1.ทำการแปลง1มิติกับแต่ละแถวของข้อมูลภาพ $f(x,y)$ จะได้

$$F(u,y) = \sum_{x=0}^{N-1} f(x,y)a_1(x,u) \qquad (2.15)$$

2.ทำการแปลง1มิติครั้งที่2 กับแต่ละคอลัมน์ของข้อมูลจากขั้นที่1 $F(u,y)$ และจะได้

$$F(u,v) = \sum_{y=0}^{N-1} F(u,y)a_2(y,v) \qquad (2.16)$$

เมื่อเขียนการแปลง 2 มิติในรูปเมตริกซ์และกำหนดให้โครงของการแปลงเป็น Separable แล้ว

$$[\mathbf{F}] = [\mathbf{A}][\mathbf{f}][\mathbf{A}] \qquad (2.17)$$

เมื่อ$[\mathbf{F}]$ เป็นเมตริกซ์ของข้อมูลหลังการแปลง

$[\mathbf{f}]$ เป็นเมตริกซ์ของข้อมูลภาพ

$[\mathbf{A}]$ เป็นเมตริกซ์ของการแปลง

ถ้าทำการคูณก่อนและหลัง$[\mathbf{F}]$ด้วยเมตริกซ์การแปลงกลับ$[\mathbf{B}]$

$$[\hat{\mathbf{f}}] \equiv [\mathbf{B}][\mathbf{f}][\mathbf{B}] = [\mathbf{B}][\mathbf{A}][\mathbf{f}][\mathbf{A}][\mathbf{B}] \tag{2.18}$$

เมื่อ$[\hat{\mathbf{f}}]$เป็นการประมาณของ$[\mathbf{f}]$

และถ้าให้เมตริกซ์การแปลงย้อนกลับ$[\mathbf{B}]$เป็นอินเวอร์สของเมตริกซ์การแปลงไปข้างหน้า$[\mathbf{A}]$ หรือ$[\mathbf{B}] = [\mathbf{A}]^{-1}$แล้ว

$$[\hat{\mathbf{f}}] = [\mathbf{A}]^{-1}[\mathbf{A}][\mathbf{f}][\mathbf{A}][\mathbf{A}]^{-1} \tag{2.19}$$

แต่

$$[\mathbf{A}]^{-1}[\mathbf{A}] = [\mathbf{A}][\mathbf{A}]^{-1} = [\mathbf{I}] \tag{2.20}$$

ดังนั้น

$$[\hat{\mathbf{f}}] = [\mathbf{f}] = [\mathbf{A}]^{-1}[\mathbf{F}][\mathbf{A}]^{-1} \tag{2.21}$$

นั่นคือ$f(x,y)$และ$F(u,v)$สามารถแสดงเป็นคู่ของการแปลง2มิติ ถ้า$[\mathbf{A}]$มีอินเวอร์ส ถ้า$[\mathbf{A}]$เป็นเมตริกซ์ Unitary ที่เป็นจำนวนจริงแล้วจะได้ว่า

$$[\mathbf{A}]^{-1} = [\mathbf{A}]^{\mathbf{T}} \tag{2.22}$$

และท้ายที่สุด ถ้า$[\mathbf{A}]$เป็นเมตริกซ์ Orthogonal ที่สมมาตรแล้วจะได้

$$[\mathbf{A}]^{-1} = [\mathbf{A}] \tag{2.23}$$

จากสมการข้างต้นมีจุดมุ่งหมายที่จะแสดงให้เห็นว่า ถ้าการแปลงมีคุณสมบัติ Separable แล้ว การแปลงข้อมูลภาพซึ่งเป็นข้อมูล 2 มิติจะสามารถคิดได้ทีละมิติ ซึ่งจะทำให้ลดขั้นตอนการคำนวนลงไปได้มาก และถ้าการแปลงมีคุณสมบัติ Unitary แล้ว การแปลงไปข้างหน้าจะเหมือนกับการแปลงย้อนกลับ ซึ่งเป็นการลดความยุ่งยากในขั้นตอนการทำงาน

นอกจากนี้ยังมีคุณสมบัติที่สำคัญอีกอย่างหนึ่งที่ใช้ในการแสดงลักษณะของการแปลงคือ คุณสมบัติของอนุรักษ์พลังงาน คุณสมบัติการอนุรักษ์พลังงานเป็นการแสดงว่าพลังงานในโดเมน Spatial จะเท่ากับพลังงานในโดเมนความถี่ ดังนั้นเมื่อนำข้อมูลมาทำการแปลงไปข้างหน้าและแปลงย้อนกลับ แล้วจะได้ข้อมูลภาพต้นแบบโดยไม่มีความผิดเพี้ยนกลับคืนมา เนื่องจากกระบวนการแปลงเป็นกระบวนการอนุรักษ์พลังงานหรือไม่มีการสูญเสีย ซึ่งสามารถพิจารณาได้จากการหาความสัมพันธ์แบบทั่วไปของ Parseval[13]

$$F(u,v)F^*(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)a(x,y,u,v)$$
$$F(u,v)F^*(u,v) \approx \sum_{\alpha=0}^{N-1}\sum_{\beta=0}^{N-1} f(\alpha,\beta)a^*(\alpha,\beta,u,v) \qquad (2.24)$$

ทำการกระจายเทอมทางขวามือ

$$F(u,v)F^*(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}[f(x,y)]^2 a(x,y,u,v)a^*(x,y,u,v)$$
$$+\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\sum_{\alpha=0}^{N-1}\sum_{\beta=0}^{N-1} f(x,y)f(\alpha,\beta)a(x,y,u,v)a^*(\alpha,\beta,u,v) \quad ;x\neq\alpha, y\neq\beta \qquad (2.25)$$

เมื่อคำนวนทุกค่าของ u และ v แล้วจะสรุปได้เป็น

$$\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}|F(u,v)|^2 = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}[f(x,y)]^2 \qquad (2.26)$$

ถ้า a_k^* แทนคอลัมน์ที่kของเมตริกซ์คอนจูเกททรานสโพส A^{*^T} เราสามารถเขียนให้เป็น

$$A_{k,l}^* = a_k^* a_l^{*^T} \qquad (2.27)$$

และกำหนดให้ค่า Inner Product ของการคูณของ 2 เมตริกซ์ F และ G ขนาด $N \times N$ เป็น

$$<\mathbf{F},\mathbf{G}> = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f(m,n)g^*(m,n) \qquad (2.28)$$

ดังนั้นจึงสามารถแสดงสมการการแปลงย้อนกลับ ได้ดังนี้

$$\mathbf{U} = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} v(k,l)\mathbf{A}^{*}_{k,l} \tag{2.29}$$

$$v(k,l) = <\mathbf{U}, \mathbf{A}^{*}_{k,l}> \tag{2.30}$$

จาก (2.29) แสดงรูปภาพ $\mathbf{U}$ เป็นการรวมตัวเชิงเส้นของเมตริกซ์ $\mathbf{A}^{*}_{k,l}$ขนาด N^2 และจะเรียกเมตริกซ์นี้ว่า Basis Image สัมประสิทธิ์ของการแปลง$v(k,l)$ก็หาได้จากค่า Inner Product ของ Basis Image ที่(k,l) Basis Image ของการแปลงแต่ละแบบไม่เหมือนกัน แต่จะมีรูปแบบคงที่เมื่อขนาดของข้อมูลคงที่ ดังนั้นจึงนิยมที่จะคำนวน Basis Image เป็นค่าคงที่เก็บไว้ก่อน

2.3.1 Discrete Cosine Transform

DCT[16,17] เป็นการแปลงที่มีการใช้สำหรับการลดข้อมูลภาพอย่างแพร่หลาย เพราะมันมีประสิทธิภาพใกล้เคียงกับ KLT มากเมื่อนำไปใช้กับข้อมูลรูปแบบ Markov อันดับที่1ที่มีความสัมพันธ์กันสูง ซึ่งข้อมูลภาพสามารถจัดให้เป็นข้อมูลในลักษณะนี้ได้เช่นกัน นอกจากนี้ DCT ยังสามารถทำการประมวลโดยใช้อัลกอริธึมแบบเร็วได้ เพราะ DCT มีความสัมพันธ์อย่างใกล้ชิดกับ DFT ถ้าเริ่มพิจารณาจากข้อมูลใน1มิติขนาดNที่มีค่าเฉพาะในช่วง $0 \le n \le N-1$ ในการหาความสัมพันธ์ของDCTต้องเชื่อมโยงชุดข้อมูล $x(n)$ ขนาด N กับชุดข้อมูลใหม่ $y(n)$ ขนาด 2N และ DFT ของข้อมูลในชุดนี้ $Y(k)$ ขนาด 2N ท้ายสุดจะทำการเชื่อมโยง $Y(k)$ กับ $C_x(k)$ ซึ่งเป็นค่า DCT ของข้อมูล $x(n)$ นี้หรือเขียนในรูป

$$x(n) \overset{N}{\leftrightarrow} y(n) \overset{2N}{\leftrightarrow} Y(k) \overset{DFT\ 2N}{\leftrightarrow} C_x(k) \overset{N}{\leftrightarrow}$$

รูปที่ 2.6 แสดงความสัมพันธ์ระหว่าง DCT กับ DFT

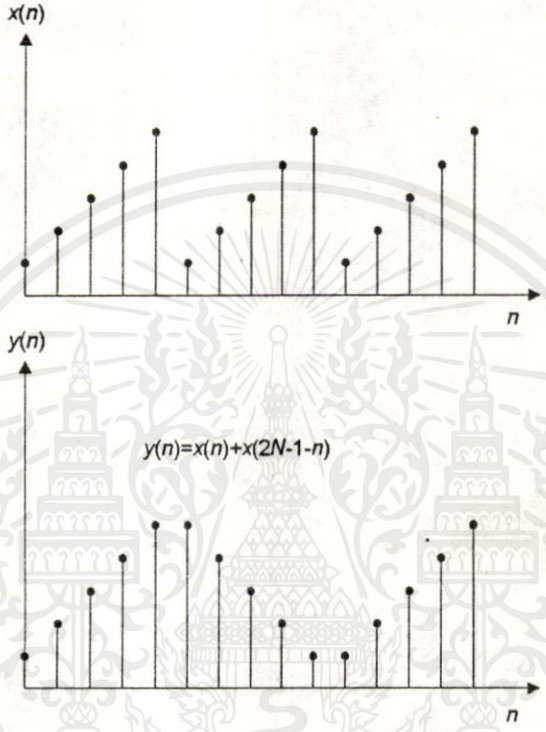
โดยชุดข้อมูล $x(n)$ เกี่ยวข้องกับ $y(n)$ ดังนี้

$$y(n) = x(n) + x(2N-1-N)$$

$$= \begin{cases} x(n) & 0 \le n \le N-1 \\ -x(2N-1-n) & N \le n \le 2N-1 \end{cases} \tag{2.31}$$

จะเห็นว่าชุดข้อมูลของ $y(n)$ สมมาตรกันที่จุด $n = N - \frac{1}{2}$ ถ้าสัญญาณ $x(n)$ เป็นสัญญาณคาบของ $\tilde{x}(n)$ แล้ว $\tilde{x}(n)$ อาจเกิดความไม่ต่อเนื่องเทียมขึ้น เพราะส่วนเริ่มต้นและส่วนจบ

ของ $x(n)$ ถูกเชื่อมกันอยู่ซ้ำๆ ในขณะที่สัญญาณคาบ $\tilde{y}(n)$ ที่มีการซ้ำข้อมูล $y(n)$ ทุก2Nจุดจะไม่มีลักษณะดังกล่าว ลักษณะทั้ง2อย่างแสดงในรูปที่ 2.7 ในการทำ DFT กับสัญญาณคาบ $\tilde{x}(n)$ จะทำให้เกิดองค์ประกอบความถี่สูงเทียมขึ้นมาและมีผลต่อประสิทธิภาพของการแปลง ความพยายามที่จะเพิ่มประสิทธิภาพของการแปลงโดยการละเลยองค์ประกอบเหล่านี้ทำให้เกิดความผิดเพี้ยนในขั้นตอนการแปลงกลับที่บริเวณขอบจนสังเกตเห็น(Blocking Artifacts)



รูปที่ 2.7 แสดงลักษณะที่ต่อเนื่องกันของ DCT

เนื่องจากชุดข้อมูล $y(n)$ เป็นสมมาตรคู่ของค่าจริง(ในกรณีพิจารณาเฉพาะค่าความเข้มจุดภาพเท่านั้น) ข้อมูลหลังการแปลง DFT คือ $Y(k)$ ขนาด2Nจุดจะเทียบเท่ากับข้อมูลหลังการแปลง DCT ซึ่งเป็น $C_x(k)$ ขนาดNจุด เพราะ $Y(k)$ ก็มีคุณสมบัติสมมาตรคู่ของค่าจริงเช่นเดียวกัน

การแปลง DCT 2 มิติของข้อมูล $f(j,k)$ ขนาด $N \times N$ หาได้จาก

$$F(u,v) = \frac{4c(u)c(v)}{N^2} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} f(j,k) \cos[\frac{(2j+1)u\pi}{2N}] \cos[\frac{(2k+1)v\pi}{2N}] \quad (2.32)$$

และการแปลงกลับของ DCT 2 มิติจะอยู่ในรูป

$$f(j,k) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} c(u)c(v)F(u,v) \cos[\frac{(2j+1)u\pi}{2N}] \cos[\frac{(2k+1)v\pi}{2N}] \quad (2.33)$$

เมื่อค่าถ่วงน้ำหนัก

$$c(w) = \frac{1}{\sqrt{2}} \quad ;w = 0$$
$$= 1 \quad ;w = 1,2,...,N-1$$

2.3.2 ประสิทธิภาพของการแปลง

เนื่องจากการแปลงเป็นส่วนประกอบที่มีผลต่อการลดข้อมูลอย่างสำคัญ ดังนั้นเพื่อที่จะออกแบบวิธีการลดข้อมูลภาพให้มีประสิทธิภาพสูง จึงจำเป็นต้องเลือกตัวแปลงที่เหมาะสม ปัจจัยที่สำคัญของการแปลงที่ดีคือ จะต้องมีความสามารถในการลดข้อมูลระหว่างจุดภาพลงให้มากที่สุด[18] หรือการอัดพลังงานของข้อมูลให้อยู่ในบริเวณที่เล็กที่สุด[19] จากการที่การแปลงเป็นการกระทำต่อข้อมูลเป็นกลุ่ม เราจึงสนใจที่จะพิจารณาคุณสมบัติของสัมประสิทธิ์หลังการแปลงเป็นกลุ่มโดยรวมไม่ใช่พิจารณาจากแต่ละสัมประสิทธิ์ โดยวิธีนี้เราจะใช้ข้อมูลทางสถิติแบบMarkovอันดับที่1 ซึ่งจะแทนลักษณะของข้อมูลภาพส่วนใหญ่ได้เป็นอย่างดี เมื่อมีการระบุค่าความสัมพันธ์ระหว่างข้อมูลซึ่งเป็นการแสดงคุณสมบัติอันดับที่1(mean)และอันดับที่2(variance)ของข้อมูล ทำให้สามารถหาเมตริกซ์ covariance ได้ดังนี้

$$COV(X) = E[(X - \overline{X})(X - \overline{X})^T] \tag{2.34}$$

เมื่อ X เป็นเวกเตอร์ข้อมูลและ $\overline{X}$ เป็นเวกเตอร์เฉลี่ย

หรืออาจเขียนเมตริกซ์ Covariance อีกรูปหนึ่งได้ดังนี้

$$COV(X) = E(XX^T) - \overline{X}\overline{X}^T \tag{2.35}$$

ในทำนองเดียวกัน เราสามารถเขียนเมตริกซ์ Covariance ของข้อมูลในโดเมนความถี่ได้เป็น

$$COV(Y) = E[(Y - \overline{Y})(Y - \overline{Y})^T] \tag{2.36}$$

หรือ

$$COV(Y) = E(YY^T) - \overline{Y}\,\overline{Y}^T \tag{2.37}$$

เมื่อ Y เป็นเวกเตอร์ข้อมูลหลังการแปลงและ $\overline{Y}$ เป็นเวกเตอร์ของการเฉลี่ย

ถ้า[T]เป็นเมตริกซ์การแปลง Orthogonal ที่เราสนใจแล้วจะได้ว่า

$$\mathbf{Y} = [\mathbf{T}]\mathbf{X}$$ ตฤ ล ภ ว □ □ ·-

จากสมการข้างต้น สามารถหาความสัมพันธ์ระหว่างเมตริกซ์ข้อมูลกับเมตริกซ์ของโดเมนความถี่

$$COV(Y) = [\mathbf{T}][COV(\mathbf{X})][\mathbf{T}]^{\mathbf{T}} \qquad (2.39)$$

ซึ่งจะมีรูปแบบเหมือนกับการทำการแปลง2มิติกับข้อมูลภาพ จากนั้นจะทำการหาประสิทธิภาพในการลดความสัมพันธ์ จากการแปลงโดยการคำนวนความสัมพันธ์ระหว่างข้อมูลที่ลดลงในเมตริกซ์ Covariance ของโดเมนความถี่เทียบกับในโดเมน Spatial ถ้าสามารถลดความสัมพันธ์ระหว่างข้อมูลลงได้ทั้งหมดแล้ว เมตริกซ์ Covariance ในโดเมนความถี่จะอยู่ในรูป Diagonal เพราะสัมประสิทธิ์ของเมตริกซ์ Covariance ในโดเมนความถี่ที่อยู่นอกแกน Diagonal จะแสดงถึงความสัมพันธ์ระหว่างข้อมูลในขณะที่สัมประสิทธิ์ในแนว Diagonal จะแสดงค่าความผันผวนของข้อมูลที่เกี่ยวข้องการคำนวนหาประสิทธิภาพของการลดความสัมพันธ์หาได้จากอัตราส่วนของ Hilbert-Schmidt Norm ระหว่างเมตริกซ์ $[\mathbf{C_y} - \mathbf{D_y}]$ และ $[\mathbf{C_x} - \mathbf{I}]$ เมื่อ $\mathbf{C_y}$ เป็นเมตริกซ์ Covariance ในโดเมนความถี่ของเมตริกซ์ Covariance ของข้อมูล $\mathbf{C_x}$ และเมตริกซ์ $\mathbf{D_y}$ เป็นเมตริกซ์ Diagonal ซึ่งสัมประสิทธิ์ที่ไม่เป็นศูนย์จะเป็นไปตามองค์ประกอบตามแนว Diagonal ของ $\mathbf{C_y}$ และ $\mathbf{I}$ เป็นเมตริกซ์I dentity ที่สอดคล้องกันกับ $\mathbf{C_x}$ หรือสรุปว่า

$$\eta_c = \frac{\textbf{Sum of squares of off- diagonal elements of } \mathbf{C_y}}{\textbf{Sum of squares of off- diagonal elements of } \mathbf{C_x}} \qquad (2.40)$$

นอกจากนี้ยังต้องกล่าวถึงประสิทธิภาพอีกประการหนึ่งของการแปลงซึ่งจะถูกนำไปใช้งานมากในการลดข้อมูลภาพ นั่นคือประสิทธิภาพของการอัดแน่นพลังงาน เนื่องจากพลังงานของข้อมูลก่อนและหลังการแปลงจะเท่ากันเสมอ แต่พลังงานของข้อมูลหลังการแปลงจะมีค่ามากเฉพาะสัมประสิทธิ์2-3ตัวแรกและจะลดลงอย่างรวดเร็ว ตามที่จะกล่าวในบทต่อไป ดังนั้นจะคำนวณหาประสิทธิภาพของการอัดแน่นพลังงานสามารถคำนวณจากค่า Mean Square Error ต่ออัตราส่วนของการลดข้อมูล M/N^2 เมื่อ Mเป็นจำนวนสัมประสิทธิ์ที่ใช้จากจำนวนข้อมูลทั้งหมดN^2 ค่าMSEสามารถหาได้จากผลรวมของวาเรียนซ์จากสัมประสิทธิ์ของการแปลงที่ถูกละเลย

จากคำจำกัดความข้างต้น ทำให้สามารถจัดลำดับเปรียบเทียบการแปลงแต่ละชนิดได้ จากข้อมูลในบทที่ 3 จะเห็นได้อย่างชัดเจนว่าในกรณีของข้อมูลทางสถิติที่มีความสัมพันธ์กันอยู่สูงนั้น DCT สามารถอัดแน่นพลังงานให้อยู่ในบริเวณแคบกว่า DFT และ DST(Discrete Sine Transform)

2.4 การควอนไทซ์

การควอนไทซ์เป็นหัวใจของการแปลงจากอนาลอกไปเป็นดิจิตอล ถ้าอธิบายอย่างง่ายที่สุดแล้ว Quantizer จะตรวจข้อมูลอินพุทที่ได้รับและเลือกค่าประมาณที่ได้จากฐานข้อมูลที่กำหนดไว้ล่วงหน้าที่ใกล้เคียงที่สุดให้ โดยปกติแล้วข้อมูลอินพุทจะอยู่ในรูปอนาลอกคือมันจะมีค่าอยู่ในช่วงค่าต่อเนื่องและข้อมูลเอาท์พุทจะอยู่ในรูปดิจิตอลซึ่งจะถูกระบุให้เป็นตัวเลขจำนวนเต็ม{1,2,3,...,N} เมื่อNเป็นขนาดของชุดข้อมูลเอาท์พุท ถ้าจะอธิบายให้ละเอียดขึ้น เราอาจจะกำหนดให้ Quantizer,Q ขนาด N จุดใน1มิติ (Scalar) เป็น Mapping $Q:R \rightarrow C$ เมื่อ Rเป็นเส้นของเลขจริงและ

$$C \equiv \{y_1, y_2, y_3, \ldots, y_n\} \subset \Re \tag{2.41}$$

เป็นชุดข้อมูลเอาท์พุทหรือ Codebook ที่มีขนาด $|C| = N$ ค่าข้อมูลเอาท์พุท y_i คือReproduction value ในขั้นตอนต่อไปจะทำการกำหนดความละเอียด(Resolution)หรือ Code Rate,r ของ Quantizer เป็น $r = \log_2 N$ ซึ่งค่านี้จะใช้วัดจำนวนบิทที่ต้องใช้ในการระบุค่าของการควอนไทซ์ที่ถูกต้องและค่าความละเอียดจะใช้บอกความแม่นยำในการคาดหมายค่าข้อมูลอินพุท

ในการสร้างQuantizerขนาดNจุด จะต้องมีการแบ่งเส้นจริง$\Re$ ออกเป็นNเซลเรียกว่าR_i เมื่อ i=1,2,3,...,N และค่าของเซลที่i หาได้จาก

$$R_i = \{x \in \Re; Q(x) = y_i\} \equiv Q^{-1}(y_i) \tag{2.42}$$

โดยที่แต่ละเซลจะมีคุณสมบัติดังนี้

$$U_i R_i = \Re \text{ และ } R_i \cap R_j = \varnothing \text{ เมื่อ } i \neq j \tag{2.43}$$

เซลที่ไม่มีขอบเขตจะถูกเรียกว่าเซลเกินพิกัด(Overload)) ส่วนเซลที่มีขอบเขตจะถูกเรียกว่าเซลกรานูลาร์ พื้นที่ที่ประกอบจากเซลที่ไม่มีขอบเขตเรียกว่าพื้นที่เกินพิกัด

Quantizerจะเป็นแบบRegularถ้า

1.แต่ละเซลของ R_i เป็นช่วงของ (x_{i-1}, x_i) กับจุดปลายอีกด้านหนึ่งหรือทั้งสองด้าน

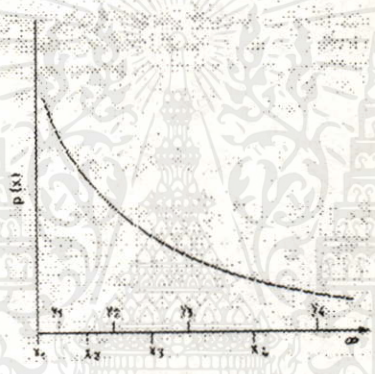
2.$y_i \in (x_{i-1}, x_i)$

ค่าของ x_i จะถูกเรียกว่าBoundary pointsหรือdecision point

Regular Quantizer จะมีลักษณะที่ว่า เมื่อมีข้อมูลอินพุท2ตัว aและb เมื่อ $a \langle b$ ถูกควอนไทซ์ได้ค่าข้อมูลเอาท์พุท w เดียวกันแล้ว ค่าอินพุทอื่นที่มีค่าระหว่าง aกับb เมื่อถูกควอนไทซ์แล้วจะให้ค่าเอาท์พุท w เช่นกัน ดังนั้นสำหรับ Regular Quantizer ช่วงสำหรับแต่ละจุดปลาย (end points) จะเรียงกันตามลำดับดังนี้

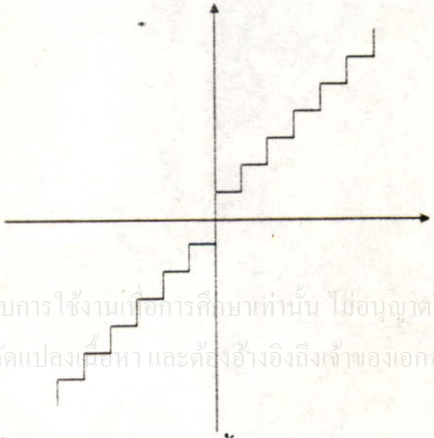
$$x_0 \langle y_1 \langle x_1 \langle y_2 \langle x_2 \langle \ldots \langle y_N \langle x_N \tag{2.44}$$

รูปต่อไปแสดงเส้นจริง $\Re$ ของ Regular Quantizer



รูปที่ 2.8 แสดงเส้นจริงของการแควนไตซ์

สำหรับการควอนไทซ์ที่นิยมใช้ในทางปฏิบัติจะกำหนดให้ Quantizer เป็น Symmetric ซึ่งหมายความว่า Q(x)=-Q(-x) และพบว่า Quantizer จะมีลักษณะเป็นขั้นบันไดตามที่เห็นในภาพต่อไปนี้



รูปที่ 2.9 แสดงลักษณะขั้นบันไดของ Quantizer

Quantizerแต่ละตัวถูกมองเสมือนเป็นการรวมผลจากกระบวนการ2อย่างที่ต่อเนื่องกันคือ การเข้ารหัส ε และการถอดรหัส D การเข้ารหัสคือการ Mapping $\varepsilon: R \to I$ เมื่อI={1,2,3,...,N} และการถอดรหัสเป็นการ Mapping $D: I \to C$ ดังนั้นถ้า $Q(x) = y_i$ แล้ว $\varepsilon(x) = i$ และ $D(i) = y_i$ จากความหมายนี้พบว่า $Q(x) = D(\varepsilon(x))$ ในบางครั้งการถอดรหัสถูกเรียกว่า" Inverse Quantizer"

ในอดีตได้มีนักวิจัยเป็นจำนวนมากพยายามออกแบบ Quantizer เพื่อให้ประมาณค่าข้อมูลอินพุทได้ใกล้เคียงที่สุด แต่วิธีที่สามารถปฏิบัติได้ง่าย,ให้ผลทางคณิตศาสตร์ที่ดีและเป็นที่นิยมที่สุดได้แก่ วิธีที่คิดค้นขึ้นโดย Lloyd[20] และพัฒนาโดย Max[21] Quantizer ที่ถูกออกแบบโดยใช้ข้อกำหนดดังกล่าวเรียกว่า Lloyd-Max Quantizer ซึ่งทั้ง Lloyd และ Max ได้วิเคราะห์สัญญาณรบกวนจากการควอนไทซ์โดยใช้ MSE และได้พยายามลดสัญญาณรบกวนนี้ลงโดยกำหนดให้ความน่าจะเป็นของความหนาแน่นสัญญาณในช่วงหนึ่งมีค่าไม่คงที่และกำหนดให้อนุพันธ์ของค่าความผิดเพี้ยนเมื่อเทียบกับระดับ y_i และช่วง x_i เท่ากับศูนย์ ทำให้ได้ Quantizer ในลักษณะ Nonuniform และค่าการจัดระดับจะได้จาก

1.ขอบของแต่ละช่วงจะต้องอยู่กึ่งกลางระหว่างขอบของเซลข้างเคียง

$$x_i = \frac{y_i + y_{i+1}}{2} \tag{2.45}$$

2.จุดขอบควรจะอยู่ที่จุดเซนทรอยด์ของpdfของข้อมูลอินพุตตลอดช่วงของข้อมูลนั้น

$$\int_{x_{i-1}}^{x_i} (x - y_i)\, p(x) dx = 0 \tag{2.46}$$

2.5 การเซกเมนต์ภาพ(Image Segmentation)

จุดประสงค์ของการเซกเมนต์ภาพคือการแยกองค์ประกอบของภาพไปเป็นส่วนประกอบย่อยๆ ที่สัมพันธ์กับลักษณะทางกายภาพของรูปนั้น ส่วนประกอบที่ถูกแยกออกมานั้นจะถูกนำไปตีความและ รู้จำต่อไป ในการแยกแยะภาพถือว่าวัตถุมีผิวเรียบเป็นเอกพันธ์(Homogeneous) โดยความเข้มของภาพที่ผิวหน้านี้มีค่าใกล้เคียงกัน นอกจากนี้จะเกิดการเปลี่ยนความเข้มทันทีทันใดที่บริเวณขอบของวัตถุ อย่างไรก็ตามสมมุติฐานนี้ก็ไม่สามารถใช้ได้กับทุกกรณี ในบางครั้งขอบทางกายภาพระหว่างวัตถุที่คล้ายกันก็จะไม่ปรากฏเป็นขอบของภาพอย่างชัดเจน

โดยทั่วไปแล้ว การเซกเมนต์ภาพได้ถูกนำมาใช้อย่างกว้างขวางในฐานะที่เป็นตัวเลือกวัตถุมาทำการวิเคราะห์ภาพ (Image Analysis) ต่อไป การเซกเมนต์ภาพทำงานในวิธีเดียวกันกับที่ตาของคนแยกลักษณะเด่นออกมาจากภาพที่มองเห็น นอกจากนี้การเซกเมนต์ภาพยังทำการแบ่งภาพออก

เป็นส่วนๆที่มีความหมายทางกายภาพและแยกพื้นที่ที่เป็นเอกพันธ์สำหรับลักษณะบางอย่างอีกด้วย ถึงแม้ว่ามีลักษณะหลายอย่างที่สามารถนำมาใช้ในการตัดสินใจความเป็นเอกพันธ์ในพื้นที่ ในการเซกเมนต์ภาพส่วนใหญ่จะสนใจเพียงลักษณะสำคัญ2อย่างคือความคล้าย (Similarity) และความต่าง (Discontinuity)

วิธีการเซกเมนต์ภาพหลักๆแบ่งเป็น 2 วิธีคือ วิธีอาศัยขอบเป็นหลักใช้ตรวจสอบความแตกต่างเพื่อหาความไม่ต่อเนื่องภายในก่อน จากนั้นจะเชื่อมต่อผลที่ได้ให้ยาวขึ้นเพื่อให้ได้เป็นขอบของภาพ อีกวิธีหนึ่งคือวิธีอาศัยพื้นที่เป็นหลักนำคุณสมบัติความคล้ายมาทำการค้นหาพื้นที่ของภาพที่มีคุณสมบัติเป็นเอกพันธ์ ในหัวข้อต่อไปจะอธิบายถึงวิธีการทำงานของการเซกเมนต์ภาพตามวิธีการทั้ง 2 แบบ

2.5.1 การเซกเมนต์ภาพโดยอาศัยขอบเป็นหลัก

ได้แก่วิธีตรวจหาขอบ ในการหาขอบจะสันนิษฐานว่าที่ขอบจะมีการเปลี่ยนแปลงความเข้มจนทำให้เกิดความเข้มไม่ต่อเนื่องเป็นลักษณะของ Step Function ในภาพ2มิติขอบจะมีทิศทางเช่นเดียวกับภาพและความเข้มตามแนวขอบจะเป็น Uniform วิธีการหาขอบสามารถทำได้หลายวิธีด้วยกันเช่น

1. Gradient Method เป็นวิธีที่ง่ายที่สุดในการหาขอบ ถ้าค่าเกรเดียนท์ที่ขอบความเข้มภาพมีค่ามากกว่าค่าเธ็รสโฮลด์แล้วถือว่ามีขอบเกิดขึ้น ซึ่งค่าเกรเดียนท์ของภาพ f(x,y) ที่ตำแหน่ง (x,y) คือเวกเตอร์

$$\nabla \mathbf{f} = \quad \nabla \mathbf{f} = \begin{bmatrix} \mathbf{G}_x \\ \mathbf{G}_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.47}$$

จากคณิตศาสตร์การวิเคราะห์เวกเตอร์แสดงว่าเวกเตอร์เกรเดียนท์จะชี้ทิศทางที่มีอัตราการเปลี่ยนแปลงมากที่สุดของ f ที่ตำแหน่ง (x,y) ในการหาขอบจำเป็นต้องรู้ขนาดของขอบและทิศทาง ซึ่งขนาดความเข้มของขอบหาได้จากขนาดของเวกเตอร์เกรเดียนท์หรือค่าเกรเดียนท์ ∇f และทิศทางของขอบหาได้จากมุมของเวกเตอร์เกรเดียนท์ $\theta_g(x,y)$ ซึ่งหาได้จาก

$$\nabla f = [G_x^2 + G_y^2]^{1/2}$$
$$\theta_g(x,y) = \tan^{-1}(\frac{G_y}{G_x}) \tag{2.48}$$

แต่ในการหาขอบในทางปฏิบัติ มักใช้วิธีประมาณค่าเกรเดียนท์จากค่าสัมบูรณ์ของเกรเดียนท์ในแต่ละทิศทาง

$$\nabla f \approx |G_x| + |G_y| \tag{2.49}$$

นอกจากนี้การคำนวนหาค่าเกรเดียนท์ในแต่ละทิศทางซึ่งได้จากการทำอนุพันธ์ย่อย ยังสามารถประมาณได้จากการ Mask บริเวณข้างเคียง ซึ่งได้มีการคิด Mask แบบต่างๆ เช่น Sobel Operator Mask

$$\begin{aligned} G_x &= (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \\ G_y &= (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \end{aligned} \tag{2.50}$$

เมื่อกำหนดพื้นที่ข้างเคียงขนาด 3×3 ของตำแหน่งที่หาขอบ

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix} \tag{2.51}$$

ทำให้สามารถกำหนด Mask แบบต่างๆในรูปของเมตริกซ์ได้ดังนี้

Sobel Operator

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.52}$$

Roberts Operator

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{2.53}$$

Prewitt Operator

$$G_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.54}$$

2. Second Derivative Method จากวิธีการหาขอบโดยใช้อนุพันธ์อันดับที่1ที่ได้กล่าวมามักมีปัญหาเมื่อใช้กับลักษณะความเข้มในช่วงลาดเอียงที่ไม่เป็นStep ซึ่งพบเมื่อวัตถุลาดเอียงถูกส่องด้วยแสงจากแหล่งเดียว ปัญหานี้จะหมดไปเมื่อใช้ Second Derivative Method สำหรับขอบที่มีลักษณะกระโดดแล้วอนุพันธ์อันดับที่2ที่จุดขอบนั้นจะเป็นศูนย์แต่จะมีค่าบวกและค่าลบอยู่คนละด้านของขอบ จากนั้นการหาขอบทำได้จากการทำ Zero Crossing การหาอนุพันธ์อันดับสองนี้มักจะทำโดย.ใช้ Laplacian Operator

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{2.55}$$

เพื่อให้สามารถนำไปใช้งานได้ง่ายขึ้น เราอาจจะประมาณค่าอนุพันธ์อันดับ2นี้เป็นการ Mask ของข้อมูลข้างเคียงดังนี้

$$1)\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad 2)\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad 3)\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \tag{2.56}$$

เนื่องจากการหาขอบวิธีนี้มีการตอบสนองไวต่อขอบมาก ดังนั้นสัญญาณรบกวนที่มีอยู่ในภาพก็อาจถูกเสริมขึ้น นอกจากนี้ยังไม่สามารถหาทิศทางของขอบได้อีกด้วย ในทางปฏิบัติจึงนิยมใช้Zero Crossing เพื่อหาตำแหน่งของขอบแทน

2.5.2 การเซกเมนต์โดยอาศัยพื้นที่เป็นหลัก

วิธีนี้คล้ายกับวิธีการแบ่งพื้นที่ หลักการนี้จะหาเซทของจุดภาพที่เชื่อมต่อกัน ซึ่งข้อมูลในเซทนี้มีลักษณะพื้นฐานบางอย่างเหมือนกัน อาจสันนิษฐานว่ามีส่วนสัมพันธ์กับวัตถุ วิธีการหลักที่จัดอยู่ตามหลักการนี้มีดังนี้

1. Thresholdingเป็นเทคนิคในการแบ่งแยกพื้นที่อย่างง่ายที่สุด ทุกจุดบนภาพที่มีคุณสมบัติของภาพที่อยู่ในช่วงใดช่วงหนึ่งจะถูกจัดเป็นกลุ่มๆหนึ่ง วิธีการนี้เหมาะกับลักษณะภาพที่ประกอบด้วยวัตถุที่เป็นเนื้อเดียวกันที่วางอยู่กับฉากหลังที่มีความเข้มสูงและเป็นเนื้อเดียวกัน เช่น ตัวอักษรบน □ □□ □□□ □ นที่ออกจากกัน ระดับเธ็รสโฮลด์นี้อาจถูกเลือกขึ้นมาจากประสบการณ์หรืออาจกำหนดให้มันโดยอ้างอิงจากฮิสโตแกรมของความเข้มก็ได้ ถ้า ฮีสโตแกรมแยกออกเป็นหลายๆกลุ่มอย่างเด่นชัด กลุ่มเหล่านี้จะเป็นตัวแสดงคุณสมบัติที่แตกต่างกันของภาพนั้นๆ และค่าเธ็รสโฮลด์สามารถกำหนดได้จากบริเวณรอยต่อระหว่างModeของกลุ่มข้อมูลในฮีสโตแกรม ในกรณีที่มีจุดยอดเล็กๆจำนวนมากซึ่งเกิด

เวณรอยต่อระหว่างModeของกลุ่มข้อมูลในฮีสโตแกรม ในกรณีที่มีจุดยอดเล็กๆจำนวนมากซึ่งเกิดจากสัญญาณรบกวนสามารถกำจัดได้โดยการทำฮีสโตแกรมให้เรียบขึ้น

2. Region Growing ในการเซกเมนต์ภาพถือว่าบริเวณข้างเคียงเป็นบริเวณที่สำคัญมาก จุดภาพที่อยู่เคียงข้างกันมักจะมีคุณสมบัติทางสถิติที่คล้ายกันของจุดรอบข้างมาเชื่อมต่อกัน ในวิธีนี้จะพิจารณาภาพเป็นบริเวณย่อยจำนวนมาก ซึ่งบริเวณเหล่านี้ถือว่ามีคุณสมบัติเป็นUniform จากนั้นพื้นที่ติดกันจะถูกนำมาพิจารณาความเป็นเนื้อเดียวกันร่วมกัน การรวมตัวกันจะสิ้นสุดลงเมื่อพื้นที่ข้างเคียงไม่สามารถบรรลุลักษณะความคล้ายกันได้ มีหลายวิธีที่สามารถนำมาใช้ตั้งกฎเกณฑ์ในการตัดสินใจที่จะรวมพื้นที่เข้าด้วยกันซึ่งโดยมากจะอ้างอิงกับการวัดกำลังขอบ กำลังขอบที่ตำแหน่งใดๆขึ้นกับการวัดค่าที่พื้นที่ติดๆกัน ถ้าความแตกต่างนี้มากกว่าค่าเธรสโฮลด์ขอบที่ตำแหน่งนั้นถือว่าเข้มและในกรณีกลับกันขอบที่มีความแตกต่างต่ำจะถือว่าจาง พื้นที่สามารถนำมารวมกันได้เมื่อมีขอบจางกั้นไว้ อย่างไรก็ตาม การเซกเมนต์ภาพด้วยวิธีนี้อาจเกิดความผิดพลาดจากผลของการสะท้อนแสงในบางพื้นที่ ทำให้ขอบบริเวณนั้นจางลงและพื้นที่อาจถูกรวมกัน ลักษณะนี้เป็นเช่นเดียวกับอาการที่เส้นขอบหักในการหาขอบ

2.5.3 ความเป็นเอกพันธ์ของพื้นที่(Region Homogeneity)

ในเทคนิคการเซกเมนต์ภาพโดยใช้พื้นที่ มักจะนำลักษณะความเป็นเอกพันธ์มาใช้ในการตัดสินใจ ในหัวข้อนี้จึงได้นำวิธีการวัดความเป็นเนื้อเดียวกันของพื้นที่มาอธิบายดังต่อไปนี้

1. ความคล้ายกันในบริเวณ เมื่อแบ่งบริเวณ$S = S_0$ออกเป็นชิ้นๆ$S_1, S_2, S_3, \ldots$โดยวิธีต่างๆ เช่น Quad-Tree จากนั้นคำนวนค่าทางสถิติของทั้งบริเวณSและS_iเป็นเวกเตอร์ลักษณะของS_i ถ้าเวกเตอร์ลักษณะเหล่านี้มีค่าใกล้เคียงกันมากพอแล้วจะถือว่าบริเวณSมีความเป็นเนื้อเดียวกัน ค่าทางสถิติที่นำมาคิดเป็นได้ทั้งค่าอันดับที่1 เช่น ค่าเฉลี่ยและค่าความผันผวน

2. ลักษณะเดียวกันของพื้นที่ ให้V(s)เป็นการวัดความแตกต่างของลักษณะในพื้นที่S เช่น Variance หรือ Standard Deviation ดังนั้นเราอาจใช้ตัวแปรที่อ้างอิงค่าV(s)สำหรับการวัดลักษณะเดียวกันของพื้นที่ เช่น $1/(1+V(s))$ นั่นคือถ้าSเป็นพื้นที่ที่มีค่าคงที่ ผลที่ได้จะเป็น1 นอกจากนี้ยังอาจพิจารณาลักษณะเดียวกันของพื้นที่ได้จากการเทียบค่าของพื้นที่Sกับค่าคงที่หนึ่ง โดยความผิดพลาดที่เราสนใจอาจเป็นค่ามากที่สุดหรือMSE

3.ความสามารถในการประมาณ ซึ่งอาจใช้หาความเป็นเอกพันธ์ของพื้นที่Sได้โดยการพิจารณาความสามารถในการประมาณพื้นที่Sโดยใช้ฟังก์ชั่นมาตรฐาน

บทที่ 3

การลดข้อมูลภาพโดยใช้การแปลง

3.1 ลักษณะของข้อมูลภาพภายหลังการแปลง

การแปลงคือการคูณภาพอย่างต่อเนื่องของเวกเตอร์องค์ประกอบข้อมูลกับชุดของ Basis Vetor ซึ่งอาจพิจารณาว่าเป็นค่าถ่วงน้ำหนัก เมื่อรวมค่าที่ได้จากการถ่วงน้ำหนักที่จุดต่างๆเข้าด้วยกัน แล้วนำผลรวมที่ได้มาหารด้วยScaling Factor กระบวนการนี้มีลักษณะเทียบเคียงกับการทำFeature Extraction เช่น ค่าจากการปลงจะเป็นบวกและมีขนาดใหญ่เมื่อข้อมูลที่นำมาทำการแปลงมีค่าเท่าๆ กัน ในขณะที่ค่าจากการแปลงจะเป็นลบค่ามากเมื่อข้อมูลมีขนาดพอๆกันต่มีเครื่องหมายกลับไปกลับมา ซึ่งจะเป็นวิธีเดียวกับที่การแปลงฟูร์เรียร์ทำการวิเคราะห์Spectral ซึ่งBasis Vectorในกรณีนี้จะเป็นฟังก์ชั่นของไซน์และโคไซน์ที่ความถี่สูงขึ้น

ก่อนที่จะพูดถึงลักษณะของข้อมูลจะขออธิบายตัวแปรที่จำเป็นต้องใช้ในการอธิบายข้อมูล[14] โดยจะเริ่มต้นจากค่าเฉลี่ยของข้อมูลซึ่งจะหาง่ายๆได้จาก

$$\overline{x} = \frac{1}{N \times M} \sum_{n=1}^{N} \sum_{m=1}^{M} x_{nm} \tag{3.1}$$

เมื่อ x_{nm} เป็นค่าข้อมูลที่ประกอบด้วย Nแถวและ Mคอลัมน์

ในกรณีที่ทราบโอกาสของการเกิดความเข้ม l ของแต่ละจุดภาพแล้วจะหาค่าเฉลี่ยจาก

$$\overline{x} = \sum_{l=1}^{2^B} x_l p(x_l) \tag{3.2}$$

เมื่อ 2^B เป็นความละเอียดของภาพและBคือจำนวนบิทที่ใช้เก็บข้อมูลแต่ละจุด ส่วนค่าความผันผวนของภาพจะหาได้จาก

$$\sigma^2 = \frac{1}{N \times M} \sum_{n=1}^{N} \sum_{m=1}^{M} (x_{nm} - \overline{x})^2 \tag{3.3}$$

หรือจะหาได้จาก

$$\sigma^2 = \sum_{l=1}^{2^B} (x_l - \overline{x})^2 \, p(x_l) \tag{3.4}$$

และเมื่อทำการหาค่ารากที่สองของค่าความผันผวนจะได้ค่าเบี่ยงเบนมาตรฐาน(Standard Deviation) จากนั้นเราอาจพิจารณาความสัมพันธ์ระหว่างค่าเฉลี่ยและค่าความผันผวนเพื่อหาพลังงานที่จุดนั้นได้ดังนี้

$$\begin{aligned} \sigma^2 &= E(x - \overline{x})^2 \\ \sigma^2 &= E(x^2 + \overline{x}^2 - 2x\overline{x}) \\ \sigma^2 &= E(x^2) + \overline{x}^2 - 2\overline{x}E(x) \\ \sigma^2 &= E(x^2) - \overline{x}^2 \end{aligned} \tag{3.5}$$

ดังนั้นอาจพิจารณาค่าความผันผวน σ^2 เป็นค่าความแตกต่างระหว่างพลังงานทั้งหมดกับพลังงานเฉลี่ยต่อจุดภาพ $\overline{x}^2$ หรือสรุปว่า

$$E(x^2) = \sigma^2 + \overline{x}^2 \tag{3.6}$$

พลังงานทั้งหมด = พลังงาน"AC" + พลังงาน"DC"

ค่าความผันผวนของภาพจึงเป็นข้อมูลสำคัญที่จะใช้บอกปริมาณการเปลี่ยนแปลงขนาดของแต่ละจุดภาพ[10] ซึ่งหมายถึงระดับกิจกรรมในภาพ แต่มันไม่สามารถชี้ถึงลักษณะการเปลี่ยนแปลงอย่างชัดเจนดังจะเห็นได้จากภาพต้นแบบที่ใช้ในวิทยานิพนธ์

ถ้าพิจารณาถึงภาพภายหลังการแปลงโดยทั่วไปแล้วจะพบว่าข้อมูลจะมีค่ามากเฉพาะในบริเวณมุมบนซ้ายเท่านั้น ดังนั้นเพื่อให้สะดวกและง่ายในการคำนวนหาความสัมพันธ์ทางสถิติของข้อมูลภาพหลังการแปลงซึ่งยังคงมีขนาดใหญ่พอๆกับข้อมูลภาพต้นแบบ เราจะพิจารณาข้อมูลภาพเป็นตัวแปรสุ่มทางสถิติที่มีรูปแบบมาตรฐานซึ่งทำให้หาค่าเมตริกซ์ Covariance ได้โดยใช้สมการ

$$COV[X] = E[(X - \overline{X})(X - \overline{X})] \tag{3.7}$$

ในการศึกษาลักษณะทางสถิติของข้อมูลภาพจะสามารถกำหนดรูปแบบของเมตริกซ์ Covariance ได้ตาม2สมมุติฐานใหญ่[22]ดังนี้

1.Separable Covariance Model รูปแบบนี้ถือว่ารูปแบบทางสถิติได้จากผลคูณของ Covariance1มิติ 2ชุดเข้าด้วยกันดังนี้

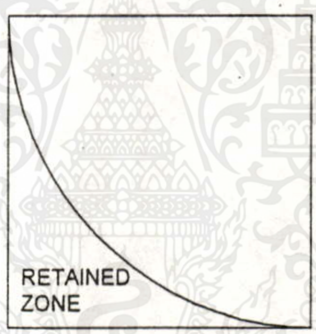
$$r(m,n;m',n') = r_1(m,m')r_2(n,n') \tag{3.8}$$

และรูปแบบนี้จะหาได้จาก

$$r(m,n) = \sigma^2 \rho_1^{|m|} \rho_2^{|n|} \quad |\rho_1| \langle 1 \quad |\rho_2| \langle 1 \tag{3.9}$$

ในที่นี้ σ^2 แสดงค่าความผันผวนของ Random Field และ $\rho_1 = {r(1,0)}/{\sigma^2}, \rho_2 = {r(0,1)}/{\sigma^2}$ เป็นความสัมพันธ์1ขั้นถัดไปในทิศทาง mและn

การกระจายพลังงานของรูปแบบนี้จะเป็นดังรูปที่ 3.1



รูปที่ 3.1 แสดงการกระจายพลังงานตาม Separable Covariance Model

2.Isotropic Covariance Model พิจารณารูปแบบทางสถิติให้เหมาะสมกับภาพตามที่เป็นจริงมากกว่า ซึ่งรูปแบบนี้จะกำหนดให้

$$r(m,n) = \sigma^2 \exp\{-\sqrt{\alpha_1 m^2 + \alpha_2 n^2}\} \tag{3.10}$$

เมื่อ $\alpha_1 = \alpha_2 = \alpha$ แล้ว $r(m,n)$ จะเป็นฟังก์ชั่นของระยะEuclidian $d \equiv \sqrt{m^2 + n^2}$ หรือ

$$r(m,n) = \sigma^2 \rho^d \tag{3.11}$$

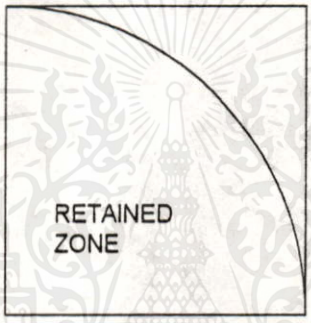
และ

$$\rho = \exp(-|\alpha|) \tag{3.12}$$

ตัวแปรของรูปแบบนี้จะขึ้นกับความสัมพันธ์ใน1ขั้นถัดไปดังนี้

$$\alpha_1 = -\ln \rho_1 \text{ และ } \alpha_2 = -\ln \rho_2 \tag{3.13}$$

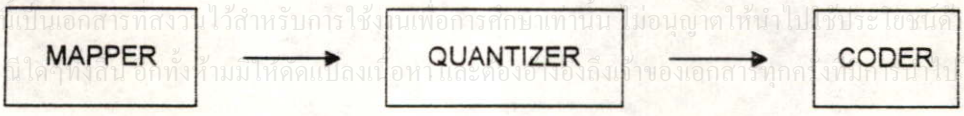
ถ้าเรานำรูปแบบนี้มาหาการกระจายพลังงานของมันจะได้ผลตามรูปข้างล่าง



รูปที่ 3.2 แสดงการกระจายพลังงานตาม Isotropic Covariance Model

3.2 องค์ประกอบของTransform Coding

จากพื้นฐานของการลดข้อมูลภาพโดยใช้การแปลงที่ได้กล่าวไปในหัวข้อ 2.2 ในตอนนี้จะมาพูดถึงรายละเอียดในการทำงานในวิธีการลดข้อมูลโดยใช้การแปลง โดยจะแบ่งออกเป็น 3 กระบวนการหลัก ซึ่งจะเริ่มจากการแปลงเชิงเส้นที่ย้อนกลับได้ไปทำการ Map ข้อมูลไปยังสัมประสิทธิ์ในโดเมนความถี่ ซึ่งจะถูกแควนไตซ์และ Entropy Code ต่อไป ในภาพทั่วไปแล้วสัมประสิทธิ์จำนวนมากจะมีค่าน้อยและสามารถผ่านการแควนไตซ์แบบหยาบได้โดยมีการสูญเสียเล็กน้อย รูปที่ 3.3 จะแสดงองค์ประกอบหลักของการแปลง



รูปที่ 3.3 แสดงองค์ประกอบหลักของการแปลง

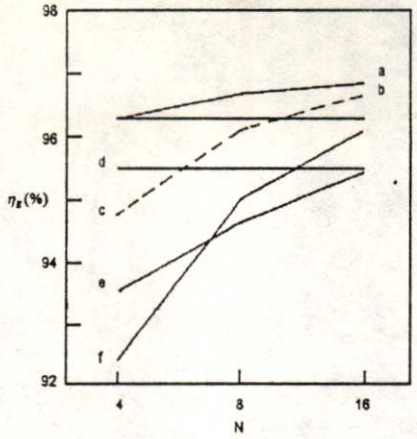
3.2.1Mapper

เป็นการแปลงข้อมูลจากโดเมน Spatial ไปอยู่ในโดเมนใหม่ ซึ่งมีความสัมพันธ์ต่อกันน้อยกว่า ดังนั้นข้อมูลในโดเมนใหม่จะอยู่ในรูปที่เหมาะจะทำการลดข้อมูลมากขึ้น ในกระบวนการนี้จะใช้การแปลงทำการลดความสัมพันธ์ของข้อมูลแต่ละจุดลงมา ข้อมูลภายหลังการแปลงที่อยู่ในโดเมนความถี่จะมีการกระจายพลังงานตามโครงสร้างทางกายภาพ ซึ่งก็คือ การรวบรวมพลังงานให้มาอัดตัวอยู่ในพื้นที่หนึ่งเป็นพิเศษ ถึงแม้ว่าพลังงานก่อนและหลังการแปลงจะไม่มีการสูญหายไปก็ตาม[13] ดังนั้นข้อมูลในช่วงนี้ยังคงมีรายละเอียดครบถ้วนและยังไม่มีการลดข้อมูลในขั้นนี้เลย จากการพิจารณาการแปลงระบบต่างๆจะพบว่าการแปลงแต่ละแบบจะมี Basis Image เฉพาะตัวของมันเอง ดังนั้นข้อมูลภายหลังการแปลงก็คือ ข้อมูลภาพที่ถูกถ่วงน้ำหนักด้วย Basis Image ที่ความถี่ต่างๆ

อย่างไรก็ตามปัญหาสำคัญของกระบวนการนี้ คือ ปัญหาเรื่องจำนวนการประมวลผล เนื่องจากการคำนวนการแปลงให้ได้แต่ละสัมประสิทธ์จะต้องทำการถ่วงน้ำหนักข้อมูลทั้งหมดเสียก่อน ซึ่งเห็นได้ว่าจำนวนการดำเนินงานทางคณิตศาสตร์จะเพิ่มขึ้นเป็นกำลังสองของขนาดที่เพิ่มขึ้น แม้ว่าในการทำงานเราสามารถคำนวน Basis Image เป็นค่าคงที่เก็บไว้ก่อนได้ เพราะสำหรับการแปลงแต่และชนิดที่ประมวลผลกับข้อมูลขนาดคงที่แล้ว Basis Image ของการแปลงนั้นจะเหมือนกัน แต่ปัญหาเรื่องการคำนวนก็ยังเป็นอุปสรรคสำคัญทำให้ไม่สามารถประมวลผลได้อย่างรวดเร็ว เพื่อแก้ไขปัญหานี้จึงได้มีการดัดแปลงโดยทำการแยกภาพทั้งหมดออกเป็นส่วนๆให้มีขนาดเล็กลง ผลของการลดขนาดลงทำให้จำนวนการคำนวนลดลงไปด้วย ในการใช้งานส่วนใหญ่จึงทำการแบ่งภาพเป็นบลอกย่อยที่มีขนาดใหญ่พอสมควรเพื่อไม่ให้ความสัมพันธ์ระหว่างบลอกติดๆกันลดต่ำมากเกินไป ซึ่งจะทำให้ขอบของบลอกย่อยๆไม่ต่อเนื่องและเห็นเป็นเส้นชัดเจนในภาพผลลัพธ์ (Block Artifact) นอกจากนี้ยังนิยมเลือกขนาดของบลอกย่อยให้เป็นค่ายกกำลังของ 2 เพื่อให้ทำการแปลงได้ง่ายขึ้น รูปต่อไปจะแสดงผลของการเลือกขนาดบลอกย่อยแต่ละแบบที่มีผลต่อความผิดพลาดในภาพผลลัพธ์ โดยจะกำหนดให้ภาพต้นแบบเป็นรูปแบบทางสถิติของ Markov 2 มิติอันดับ 1 และความผิดพลาดจะเกิดจากการละทิ้งข้อมูลอื่นที่ไม่ได้เป็นข้อมูลที่มีค่าความผันผวนสูงสุด25%แรก ความผิดพลาดนี้จะคำนวนได้จาก

$$MSE = \frac{1}{N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} E[F^2(u,v)][1-T(u,v)] \qquad (3.14)$$

เมื่อ $T(u,v)$เป็นตัวเลือกข้อมูล



รูปที่ 3.4 แสดงความสัมพันธ์ของการลดข้อมูลกับขนาดของบล็อกย่อย
โดย N แทนขนาดของบล็อก, η_E แทนประสิทธิภาพในการอัดแน่นพลังงาน
a) DCT, b) Slant, c) KLT, d) WHT, e) DFT และ f) DST

จากข้อมูลในรูปพบว่าประสิทธิภาพในการอัดแน่นพลังงานดีขึ้นเมื่อใช้ขนาดของบล็อกมากขึ้น ยกเว้นเมื่อใช้ WHT โดย DFT และ DST ที่มีประสิทธิภาพเพิ่มขึ้นมากที่สุดเมื่อมีการเพิ่มขนาดของบล็อก นอกจากนี้จะพบว่า DCT มีประสิทธิภาพในการอัดแน่นพลังงานใกล้เคียงกับเมื่อใช้ KLT มาก แต่เป็นที่น่าสังเกตว่า ประสิทธิภาพของการแปลงทุกชนิดจะมีค่าเปลี่ยนแปลงน้อยเมื่อขนาดของบล็อกมีขนาดตั้งแต่ 16×16 ขึ้นไป ดังนั้นเพื่อให้สะดวกในทางปฏิบัติ[4]แล้ว ขนาดของบล็อก $8 \times 8, 16 \times 16$ เป็นขนาดที่นิยมกันมาก

3.2.2 Quantizer

กระบวนการควอนไทซ์นับว่าเป็นกระบวนการที่มีความสำคัญมากที่สุดในการลดข้อมูลภาพโดยใช้การแปลง เนื่องจากข้อมูลในโดเมนความถี่ภายหลังการแปลงจะต้องถูกเลือกนำมาใช้ให้มีประสิทธิภาพมากที่สุดในขั้นตอนนี้ การลดข้อมูลส่วนใหญ่จะเกิดขึ้นระหว่างกระบวนการนี้ สังเกตได้ว่าการควอนไทซ์เป็นขั้นตอนที่ไม่สามารถย้อนกลับได้ (Irreversible) เพราะในขั้นตอนการ Dequantize ไม่สามารถระบุตำแหน่งที่แน่นอนและเป็นการประมาณจากข้อมูลภาพต้นแบบ โดยที่ความผิดพลาดจากการควอนไทซ์หาได้จาก

$$\varepsilon_q^2 = \sum_{i,j} nq_{i,j} S_{i,j}^2 \quad i,j = 0,1,2,\ldots,N-1 \tag{3.15}$$

เพื่อเป็นการประมาณข้อมูลให้ใกล้เคียงที่สุด จำเป็นต้องรู้รูปแบบของข้อมูลนั้นเพื่อนำมาออกแบบ Quantizer ให้เหมาะสม[23] เนื่องจากข้อมูลภาพภายหลังการแปลงส่วนใหญ่จะมีลักษณะไม่สม่ำเสมอ (Nonuniform) และค่าข้อมูลส่วนใหญ่(โดยเฉพาะในเทอมAC) มักมีค่าใกล้ศูนย์เป็นส่วนใหญ่และ

ลดลงเรื่อยๆ จึงนิยมออกแบบ Quantizer ให้มีขนาดไม่เท่ากัน โดยแต่ละช่วงการประมาณจะกว้างที่ค่ามากและการประมาณจะถี่มากหรือช่วงแคบบริเวณที่ค่าใกล้ศูนย์ สามารถสรุปได้ดังนี้

1.ข้อมูลในเทอมDCไม่ได้แสดงรูปแบบทางสถิติอย่างใดออกมาอย่างเด่นชัดหรือพบว่าการกระจายมีค่าพุ่งสูงบริเวณหนึ่งบริเวณใด แต่ถ้านำการกระจายนี้มาเทียบเคียงกับรูปแบบทางสถิติที่มีจะอยู่พบว่าใกล้เคียงกับการกระจายแบบ Gaussian มากที่สุด ในทางปฏิบัติแล้วจะกำหนดบิทสำหรับความละเอียดให้ข้อมูลในเทอมDCอย่างน้อย 8บิท

2.ข้อมูลในเทอมAC เนื่องจากข้อมูลภายหลังการแปลงได้มาจากการบวกและการลบของค่าถ่วงน้ำหนักของจุดภาพต่างๆได้อย่างเหมาะสม ถ้าข้อมูลนี้เป็น Stationary และขนาดข้อมูลNมีขนาดใหญ่มาก จะสามารถสรุปจาก Central Limit Theorem ได้ว่าข้อมูลภายหลังการแปลงนี้จะมีการกระจายแบบ Gaussian อย่างไรก็ตามในความเป็นจริงแล้วข้อมูลภาพส่วนใหญ่ไม่เป็น Stationary อย่างสมบูรณ์ และขนาดข้อมูลก็จะมีขนาดไม่ใหญ่มาก(น้อยกว่า16×16) ลักษณะการกระจายข้อมูลของภาพส่วนใหญ่จึงมีลักษณะคล้ายกับลักษณะของGaussianพอๆกับลักษณะ Laplacian

ในการลดข้อมูลมีจุดมุ่งหมายที่จะลดเนื้อที่ของข้อมูลให้มากที่สุด ดังนั้นจากจำนวนบิทที่มีอยู่จึงต้องนำมาใช้ให้เกิดประโยชน์สูงสุด การกำหนดบิทสำหรับความละเอียด (Bit Allocation) ได้ถูกนำมาใช้[24] เพื่อกำหนดความแม่นยำในการแสดงค่าข้อมูลภายหลังการแปลง เพื่อให้การควอนไทซ์สามารถเก็บรายละเอียดได้มากที่สุดจำเป็นต้องกำหนดความแม่นยำให้กับข้อมูลส่วนที่มีความสำคัญสูง ในขณะที่ข้อมูลส่วนที่มีความสำคัญต่ำอาจกำหนดให้มีบิทสำหรับความละเอียดต่ำหรืออาจละทิ้งข้อมูลส่วนนั้นไปเลยเพื่อเป็นการประหยัดเนื้อที่ก็ได้ เนื่องจากข้อมูลภาพในทางทฤษฎีจะมีการกระจายแบบ Gaussian จาก Rate Distortion Theory[11] ระบุว่าถ้าตัวแปรแบบ Gaussian ที่มีค่าความผันผวนเท่ากับ σ^2 และเรายอมให้มีความผิดเพี้ยน MSE เท่ากับ D แล้ว จำเป็นต้องใช้ข้อมูลที่มีขนาดไม่น้อยไปกว่า $\frac{1}{2}\log_2(\sigma^2/D)$ บิท หรือสามารถอธิบายว่าข่าวสารของตัวแปร Gaussian เป็นสัดส่วนโดย ตรงกับ $\log_2\sigma^2$

ถ้า$\sigma^2(u,v)$เป็นค่าความผันผวนของข้อมูลหลังการแปลงที่(u,v)แล้วพบว่า $F(u,v)/\sigma(u,v)$ เป็นข้อมูลที่มีค่าความผันผวนเท่ากับ 1 และกำหนดให้$d_{u,v}(b_{u,v})$เป็นค่าMSEจากการควอนไทซ์ของQuantizerที่ใช้บิทสำหรับค่าละเอียดเท่ากับ$b_{u,v}$ ซึ่งมีค่าเท่ากับ$k2^{-\rho b_{u,v}}$ ส่วนค่าผิดเพี้ยนจากการแควนไตซ์ข้อมูลทั้งภาพจะหาได้จาก[24]

$$MSE = \frac{1}{n^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}[F(u,v) - F_q(u,v)]^2$$
$$MSE = \frac{1}{n^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\sigma^2(u,v)d_{u,v}(b_{u,v}) \quad (3.16)$$

การกำหนดบิทสำหรับความละเอียดสำหรับข้อมูลภาพพยายามจะใช้จำนวนบิทให้น้อยที่สุด โดยยังมีความผิดพลาดตามที่กำหนด $(\sigma^2(u,v)2^{-2b_{u,v}} = c)$ ถ้าสัมประสิทธิ์หลังการแปลงแต่ละตัวมีการกระจายความผิดพลาดเท่าๆกันแล้ว

$$b_{u,v} = \frac{1}{2}(\log_2 \sigma^2_{u,v} - \log_2 c)$$
$$2M = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}(\log_2 \sigma^2_{u,v} - \log_2 c) \quad (3.17)$$

เมื่อกำหนดอัตราความผิดพลาดที่ยอมรับได้ c ไว้คงที่จะได้

$$2M = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\log_2 \sigma^2_{u,v} - n^2 \log_2 c \quad (3.18)$$

นำค่า $\log_2 c$ ไปแทนในสมการข้างต้นจะได้

$$b_{u,v} = \frac{1}{2}[\log_2 \sigma^2_{u,v} - \frac{1}{n^2}(\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\log_2 \sigma^2_{u,v} - 2M)]$$
$$b_{u,v} = \frac{M}{n^2} + \frac{1}{2}(\log_2 \sigma^2_{u,v} - \frac{1}{n^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\log_2 \sigma^2_{u,v}) \quad (3.19)$$

ถ้ากำหนดให้อัตราการเก็บข้อมูลเป็น M/n^2 อาจเขียนสมการที่แล้วใหม่เป็น

$$b_{u,v} = \frac{1}{2}\log_2 \sigma^2_{u,v} - \frac{1}{2n^2}(\log_2 \prod_{0}^{N-1}\sigma^2_{u,v} - 2M) \quad (3.20)$$

ซึ่งจะอยู่ในรูป

$$b_{u,v} = \frac{1}{2}\log_2(\sigma^2_{u,v}/D) \quad (3.21)$$

โดยที่

$$\log_2 D = \frac{1}{n^2}(\log_2 \prod_{0}^{N-1} \sigma_{u,v}^2 - 2M) \qquad (3.22)$$

3.2.3 ตัวเข้ารหัส

ขั้นตอนนี้เป็นการลดความซ้ำซ้อนที่ยังเหลืออยู่ของข้อมูลที่ผ่านการควอนไทซ์ วิธีการที่นำมาใช้ในการลดข้อมูลจะเป็นวิธีการลดข้อมูลโดยไม่มีความสูญเสีย เพื่อให้สามารถทำการลดข้อมูลได้อย่างเต็มที่ จึงต้องกระทำวิธีการ2อย่างต่อเนื่องกันคือ

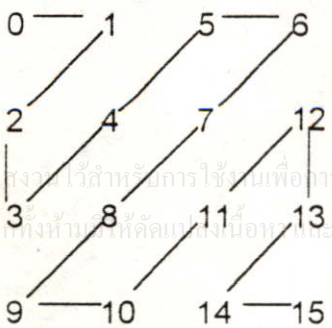
1.จัดเรียงลำดับสัมประสิทธิ์ของข้อมูลใหม่(Scanning) เพื่อที่จะหาลำดับของการเก็บข้อมูลให้มีความสัมพันธ์ต่อกันมากที่สุด ซึ่งจะทำให้สามารถลดข้อมูลในขั้นต่อไปได้ง่ายขึ้น[25]รูปแบบของการจัดเรียงลำดับที่มีประโยชน์ในการลดข้อมูลต่อไปมี5แบบหลักคือ

1.1วิธีการจัดเรียงให้ขึ้นกับอันดับความสำคัญของแต่ละสัมประสิทธิ์ตามค่า Normalized MSE (NMSE) ของการประมาณภาพนั้น เมื่อตัดสัมประสิทธิ์ที่มีความสำคัญต่ำทิ้ง ซึ่งค่าNMSEจะหาได้จาก

$$NMSE = \frac{\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}[f(j,k) - \hat{f}(j,k)]^2}{\sum_{j=0}^{N-1}\sum_{k=0}^{N-1} f(j,k)^2} \qquad (3.23)$$

อย่างไรก็ตามวิธีการนี้จะมีลำดับการเก็บข้อมูลเปลี่ยนไปตามข้อมูลภาพ ทำให้ต้องส่งลำดับข้อมูลนี้ไปให้ด้านรับด้วยทุกครั้ง

1.2วิธีการจัดเรียงแบบZigZag เนื่องจากค่าlogของค่าความผันผวนจากข้อมูลภายหลังการแปลงที่มีจากภาพทั่วไปจะมีการลดลงอย่างเป็นเชิงเส้นตามแนวZigZagที่แสดงในรูปต่อไป เมื่อดึงข้อมูลตามแนวZigZagนี้มาเรียงกันใหม่แล้วพบว่าค่าข้อมูลมีความใกล้เคียงกันมากขึ้น



รูปที่ 3.5 แสดงลำดับการเรียงข้อมูลตามวิธีที่ 2

1.3วิธีการจัดเรียงโดยเน้นแนวนอน ในบางภาพที่มีรายละเอียดหรือขอบตามแนวนอนมากเป็นพิเศษแล้วจะพบว่าค่าข้อมูลตามแนวนอนภายหลังการแปลงก็มีค่ามากขึ้นเช่นกัน ดังนั้นการเรียงข้อมูลโดยจะดึงข้อมูลที่มีความสัมพันธ์กันมาอยู่ใกล้กันมากกว่า จึงทำการเรียงสัมประสิทธิ์จากคอลัมน์แรกถึงคอลัมน์สุดท้ายของแต่ละแถวมาเรียงกัน

1.4วิธีการจัดเรียงโดยเน้นแนวตั้ง ในบางภาพที่มีรายละเอียดหรือขอบตามแนวตั้งมากเป็นพิเศษแล้วจะพบว่าค่าข้อมูลตามแนวตั้งภายหลังการแปลงก็มีค่ามากขึ้นเช่นกัน ดังนั้นวิธีการนี้จะทำการเรียงสัมประสิทธิ์จากแถวแรกถึงแถวสุดท้ายของแต่ละคอลัมน์มาเรียงกัน

1.5วิธีการจัดเรียงโดยเน้นแนวนอนและแนวตั้ง จะมีประโยชน์สำหรับภาพที่มีรายละเอียดมากทั้งตามแนวนอนและตามแนวตั้ง วิธีการนี้จะเลือกสัมประสิทธิ์สลับกันระหว่างสัมประสิทธิ์ตามแนวตั้งและตามแนวนอน

2.ทำการลดความซ้ำซากที่ยังเหลืออยู่ (Entropy Coding) ซึ่งเราสามารถแยกข้อมูลภายหลังการแปลงมาลดข้อมูลได้2แบบคือ

2.1ข้อมูลDC เป็นข้อมูลที่สำคัญและมีค่ามากเพียงตัวเดียวในแต่ละบลอก ดังนั้นจึงไม่นิยมที่จะลดข้อมูลลงอีก กล่าวคือจะกำหนดเนื้อที่อย่างต่ำ8บิทสำหรับข้อมูลDC

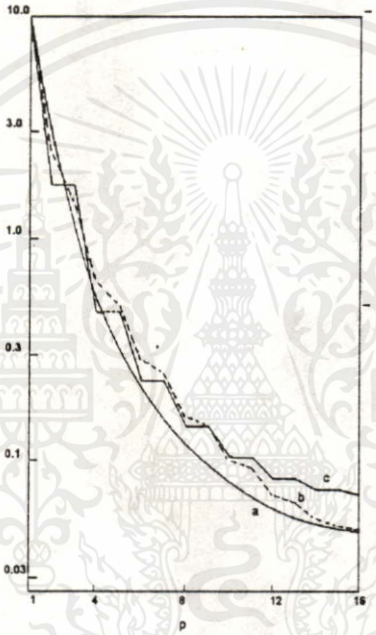
2.2ข้อมูลAC เป็นข้อมูลที่ไม่มีความสม่ำเสมอและค่าส่วนใหญ่จะเป็นศูนย์ นอกจากนี้ค่าข้อมูลของสัมประสิทธิ์ที่เหลือก็มักจะมีค่าน้อยใกล้ศูนย์เช่นกัน ในทางปฏิบัติจึงนิยมใช้การเข้ารหัสแบบ Run Length ซึ่งถือว่าเป็นตัวทำนายอันดับที่ศูนย์อย่างหนึ่ง การเข้ารหัสวิธีนี้จะลดข้อมูลที่ซ้ำซ้อนลงไป โดยการส่งความยาวของช่วงข้อมูลที่ซ้ำซ้อนไปแทน ดังนั้นวิธีการนี้จึงเหมาะกับข้อมูลที่มีค่าหนึ่งค่าใดซ้ำๆกันเป็นจำนวนมาก การเข้ารหัสแบบ Run Length จะทำการแยกข้อมูลที่จะส่งออกมาจากข้อมูลที่จะละทิ้ง ส่วนขนาดของข้อมูลที่จะเข้ารหัสจะใช้การกำหนดบิทแบบไม่เท่ากัน (Variable Length Coding) ซึ่งอาจจะใช้วิธีแบบ Huffman หรือ Arithmatic Coding การเข้ารหัสแบบ Huffman ดูจะเป็นวิธีที่ปฏิบัติได้ง่ายกว่า อย่างไรก็ตามจำเป็นต้องทราบตารางของการเข้ารหัสก่อน นอกจากนี้ Arithmatic Coding ยังให้อัตราการลดข้อมูลมากกว่าด้วย

ภายใน Entropy Coding จะมีการทำงานหลัก4อย่างคือ รูปแบบทางสถิติ,Adaptor,บริเวณจัดเก็บและตัวเข้ารหัส ซึ่งวิธีการเชื่อมต่อการทำงานต่างๆนั้นจะขึ้นกับชนิดของ Entropy Coding รูปแบบทางสถิติจะแปลงรูปแบบ Descriptor ไปเป็นสัญลักษณ์ที่ถูกกำหนดโอกาสการเกิดขึ้นไว้แล้ว ส่วนAdaptorรับผิดชอบสำหรับการกำหนดรหัสข้อมูล (Code Word) ใน Huffman Coding หรือประมาณความน่าจะเป็นที่ตัวเข้ารหัสต้องการใน Arithmatic Coding บริเวณจัดเก็บบรรจุตารางรหัสHuffmanหรือค่าประมาณความน่จะเป็นของ Arithmatic Coding ซึ่งข้อมูลที่อยู่ในบริเวณจัดเก็บจะช่วยให้ตัวเข้ารหัสเปลี่ยนสัญลักษณ์ไปเป็นจำนวนบิทข้อมูลที่ถูกเข้ารหัส

3.3วิธีการเข้ารหัสโดยใช้การแปลงแบบต่างๆ

3.3.1 Zonal Coding[25,26]

เนื่องจากข้อมูลภาพส่วนใหญ่มีความสัมพันธ์กันระหว่างจุดภาพสูง ส่งผลให้พลังงานของข้อมูลภาพภายหลังการแปลงมีแนวโน้มที่จะกระจุกตัวอยู่ในช่วงความถี่หนึ่ง รูปข้างล่างแสดงสัดส่วนของพลังงานภายในเขตวงกลมบริเวณมุมบนซ้ายของข้อมูลภาพภายหลังการแปลง จากรูปนี้พบว่าพลังงานของข้อมูลภายหลังการแปลงมากกว่า95%บรรจุอยู่ในสัมประสิทธิ์ไม่เกิน1%ของข้อมูลภาพทั้งหมด ซึ่งเกิดจากคุณสมบัติการอัดพลังงานของการแปลง



รูปที่ 3.6 แสดงคุณสมบัติการอัดพลังงานของการแปลง

a) DCT, b) DFT และ c) WHT

ข้อมูลภายหลังการแปลงส่วนใหญ่จะมีโครงสร้างคล้ายกัน เนื่องจากต้องผ่านการแปลงที่มีคุณสมบัติการอัดพลังงานเหมือนกัน ดังนั้นเราสามารถพิจารณารายละเอียดของภาพได้จากพื้นที่ส่วนหนึ่งที่มีขนาดเล็กเมื่อเทียบกับข้อมูลทั้งหมด ซึ่งพื้นที่นี้ต้องครอบคลุมข่าวสารหรือพลังงานเกือบทั้งหมดของภาพเอาไว้ นอกจากนี้ยังต้องเป็นมาตรฐานสำหรับภาพทั่วไปอีกด้วย ในการกำหนดขอบเขตของข้อมูลนี้จำเป็นต้องหาพื้นที่ทางกายภาพซึ่งสามารถเก็บรายละเอียดได้มากที่สุดก่อน การออกแบบพื้นที่ทางกายภาพนี้ขึ้นอยู่กับลักษณะการกระจายพลังงานของสัมประสิทธิ์ภายหลังการแปลง[25,26]

การเลือกพื้นที่ของการเข้ารหัส (Zonal Sampling) ขึ้นอยู่กับคุณสมบัติของการแปลงและโครงสร้างเมตริกซ์ Covariance ของข้อมูล การประมาณลักษณะโครงสร้างของเมตริกซ์ Covariance แต่ละ

แบบก็จะให้โครงสร้างพื้นที่การเข้ารหัสที่เหมาะสมต่างกันไป อย่างไรก็ตามข้อมูลภายหลังการแปลงมักถูกกำหนดให้มีรูปแบบเป็นดังนี้

1.ตามโครงสร้างเมตริกซ์ Covariance ของ Markov ที่แยกจากกัน[27] ที่กำหนดตามสมการจากลักษณะการกระจายพลังงานนี้สามารถกำหนดพื้นที่การเข้ารหัสได้ตามรูปที่ 3.1

2.ตามโครงสร้างเมตริกซ์ Covariance แบบ Isotropic[4] ซึ่งเป็นไปตามสมการที่ 3.10 พื้นที่การเข้ารหัสตามลักษณะการกระจายพลังงานตามโครงสร้างนี้สามารถกำหนดได้ตามรูปที่ 3.2

ในการเข้ารหัสตามแบบพื้นที่จะทำการเลือกข้อมูลเฉพาะในส่วนที่รูปแบบกำหนด และจะไม่สนใจข้อมูลที่นอกพื้นที่นี้หรือกำหนดให้มีค่าเป็นศูนย์ สัมประสิทธิ์ที่ถูกเลือกจะถูกนำไปทำการควอนไทซ์ต่อไป โดยที่ Quantizer มักนิยมกำหนดจำนวนบิทสำหรับความละเอียดเอาไว้ล่วงหน้า เนื่องจากตำแหน่งของการเลือกสัมประสิทธิ์และจำนวนบิทสำหรับความละเอียดได้ระบุเป็นมาตรฐานที่รู้กันทั้งทางด้านรับและด้านส่ง ค่ามาตรฐานเหล่านี้จึงไม่จำเป็นต้องส่งไปพร้อมกับข้อมูลที่เข้ารหัสแล้ว เทคนิคที่เรียกว่า Block Quantization[28] ซึ่งเป็นที่นิยมใช้กันอย่างแพร่หลายในยุคที่ริเริ่มการเข้ารหัสโดยใช้การแปลงได้ทำการแบ่งภาพทั้งหมดออกเป็นบล็อกย่อยๆที่มีขนาดเล็กลงจำนวนมากเพื่อช่วยลดจำนวนการคำนวน จากนั้นก็จะเข้ารหัสโดยวิธี Zonal Coding กับข้อมูลทุกบล็อกเหมือนๆกัน แต่เนื่องจากธรรมชาติของภาพที่ไม่เหมือนกันทำให้จำนวนบิทสำหรับความละเอียดที่กำหนดให้มีค่าเหมือนๆกันอาจไม่เหมาะกับข้อมูลภาพบางบล็อกที่ต่างออกไปมากๆ ปัญหานี้สามารถขจัดทิ้งไปได้โดยกำหนดให้แต่ละบล็อกสามารถปรับตัวโดยการปรับจำนวนบิทสำหรับความละเอียดให้ต่างออกไป[29,30] เนื่องจากการกำหนดให้แต่ละบล็อกสามารถปรับตัวเองได้ ทำให้สามารถเข้ารหัสภาพต่างๆได้เหมาะสมมากขึ้นและส่งผลให้สามารถลดข้อมูลได้มากขึ้นด้วย ทำให้มีนักวิจัยจำนวนมากใช้ความพยายามที่จะกำหนดการเข้ารหัสให้ปรับตัวได้ตามลักษณะของภาพซึ่งเรียกว่า การเข้ารหัสแบบอ่อนตัว (Adaptive Transform Coding[31])

3.3.2Threshold Coding

การทำเธรสโฮลด์เป็นกระบวนการแยกแยะข้อมูลที่อยู่รวมกันเป็นกลุ่มให้แยกจากกันอย่างชัดเจน การทำเธร็สโฮลด์เป็นที่นิยมในการแยกวัตถุที่สนใจออกจาก Background ในวิธีการแยกแยะภาพ อย่างไรก็ตามการเข้ารหัสภาพก็มีการประยุกต์การทำเธร็สโฮลด์เช่นเดียวกัน การเข้ารหัสแบบThresholdจะทำการเลือกข้อมูล (Threshold Sampling) ส่วนที่มีความหมายสำคัญของภาพนั่นคือบริเวณที่มีพลังงานสูงนั่นเอง ในทางปฏิบัติจะทำการเลือกเข้ารหัสเฉพาะข้อมูลที่มีค่าสูงกว่าระดับเธร็สโฮลด์ ดังนั้นในการกำหนดระดับเธร็สโฮลด์จึงมักเป็นการประนีประนอมระหว่างจำนวนข้อมูลที่จะนำไปใช้กับความผิดพลาดที่เกิดจากการละทิ้งข้อมูล ระดับThresholdที่ดีที่สุดสามารถกำหนดได้จาก Error Criterion ที่กำหนดไว้

ปัญหาสำคัญของการเข้ารหัสแบบThresholdคือตำแหน่งของสัมประสิทธิ์ที่เลือกไปใช้จะต้องถูกส่งไปยังด้านรับด้วย เพราะในแต่ละบล็อกจะเลือกตำแหน่งข้อมูลไม่เหมือนกันทำให้การเข้ารหัสวิธีนี้ต้องหาทางลดข้อมูลในส่วนของตำแหน่งสัมประสิทธิ์ด้วย และวิธีที่นิยมใช้คือ การเข้ารหัสของค่าศูนย์แบบ Run Length เพื่อให้สามารถควบคุมอัตราการลดข้อมูลได้ การเข้ารหัสแบบ Threshold วิธีนี้อาจจะกำหนดจำนวนข้อมูลที่ต้องการส่งขนาดคงที่ไว้แล้วเลือกข้อมูลที่มีความสำคัญมากที่สุดไล่ลงไปเรื่อยๆจนครบจำนวน

3.3.3Second-Generation Coding

จากเทคนิคการเข้ารหัสภาพที่ได้กล่าวถึงในหัวข้อก่อนๆ นับเป็นความพยายามที่จะลดความสัมพันธ์ของข้อมูลทางสถิติที่มีอยู่ตาม Information Theory แต่ผลของการเข้ารหัสยังไม่เป็นที่น่าพอใจ นักกล่าวคือ การเข้ารหัสในยุคแรกสามารถลดข้อมูลได้ถึงจุดอิ่มตัวที่ประมาณ1bpp ในการลดข้อมูลภาพเกินจากจุดนี้จะได้ภาพผลลัพธ์ไม่ชัดเจนรวมทั้งมีความไม่ต่อเนื่องที่ขอบของบล็อกอย่างเด่นชัด (Blocking Effect) จากความพยายามที่จะลดข้อมูลภาพให้มีขนาดเล็กลงกว่าแต่เดิมจึงจำเป็นต้องย้อนกลับมาพิจารณาลักษณะของภาพและลักษณะการนำภาพใช้งานอีกครั้ง เนื่องจากสายตาคนถูกใช้งานข้อมูลภาพ ดังนั้นถ้าสามารถออกแบบการเข้ารหัสให้สอดคล้องกับ Human Visual System(HVS) แล้วก็น่าจะลดข้อมูลภาพได้มากกว่า ความพยายามลดข้อมูลในลักษณะนี้ถูกเรียกว่าเป็นการเข้ารหัสในยุคที่สอง[32] ข้อแตกต่างอีกอย่างหนึ่งของการเข้ารหัสทั้ง2ยุคก็คือ โดยปกติแล้วการเข้ารหัสภาพต้องทำหน้าที่2อย่างคือ แปลงข้อมูลภาพให้เป็นชุดข่าวสาร และทำการกำหนดจำนวนบิทสำหรับความละเอียดให้กับข่าวสารนั้น ซึ่งการเข้ารหัสในยุคแรกจะเน้นไปที่ขั้นตอนที่สอง ส่วนการเข้ารหัสในยุคที่สองจะมุ่งเน้นไปที่ขั้นตอนที่1

เทคนิคการเข้ารหัสในยุคที่สองสามารถจัดแบ่งออกเป็น2แนวทางใหญ่ๆตามการดำเนินงานของแต่ละเทคนิค โดยกลุ่มแรกจะใช้ Local Operator โดยจะใช้ Filter Bank ทำการคอนโวลูชั่นกับข้อมูลภาพเช่นเดียวกับการเข้ารหัสในแบบเดิม แต่ Impulse Response ของ Filter นี้จะถูกออกแบบให้ดึงคุณลักษณะที่ขึ้นกับสิ่งแวดล้อมออกมา เอาท์พุทของ Filter เหล่านี้จะถูกนำมารวมกันอย่างเหมาะสมเพื่อให้ได้ข่าวสารที่จะนำมาเข้ารหัส Pyramid Coding[33] ก็ได้นำหลักการนี้มาประยุกต์ใช้ในการลดข้อมูล

แนวทางที่สองใช้วิธีการที่แตกต่างออกไป โดยพยายามที่จะอธิบายรูปภาพในเทอมของ Contour และ Texture เนื่องจากวัตถุในธรรมชาติจะเห็นเด่นชัดโดยขอบ (Contour) ล้อมรอบพื้นผิวของมัน (Texture) ซึ่งทำให้สรุปได้ว่าการแยกContourออกมาเป็นพื้นฐานที่สำคัญในการเห็นวัตถุ แต่การหาContourโดยทำการหาขอบมีข้อบกพร่องที่ไม่สามารถแยกContourที่เป็นขอบของวัตถุกับที่เป็นของสัญญาณรบกวนทั่วไป ในขณะที่การแยกหาContourโดยวิธีการแยกแยะภาพดูจะเป็นวิธีที่

เหมาะสมกับการเข้ารหัสมากกว่า ต่อไปจะยกตัวอย่างขั้นตอนการเข้ารหัสตามแนวทางนี้เพื่อเป็นพื้นฐานสำหรับการส่งภาพโปรเกรสซีพที่มีประสิทธิภาพสูง

การเข้ารหัสภาพโดยใช้บล็อกขนาดไม่คงที่[34]

เนื่องจากผลการทำงานของ Block Quantization ให้ผลที่ไม่ดีนักในทางปฏิบัติ เพราะรายละเอียดในส่วนต่างๆของภาพไม่เท่ากันและรายละเอียดในภาพจะยิ่งแตกต่างกันมาก โดยบริเวณที่มีรายละเอียดสูงก็ถูกทำให้ขาดความคมชัดลงไป ส่วนบริเวณที่มีรายละเอียดต่ำก็จะถูกเน้นขึ้นมา แม้ว่าปัจจุบันได้มีวิธีการปรับปรุงการทำงานให้มีการอ่อนตัวมากขึ้น (Adaptive Transform Coding) แต่ผลที่ได้ยังคงให้ผลที่ไม่คมชัดบริเวณขอบและลักษณะโดยรวมยังเห็นเป็นขอบของบล็อกอย่างเด่นชัด ในการปรับปรุงผลของการทำงานให้ดียิ่งขึ้น[35]ได้จัดแบ่งบล็อกใหม่ให้มีขนาดไม่เท่ากัน เพื่อช่วยให้Backgroundซึ่งเป็นบริเวณที่มีรายละเอียดน้อยมีความต่อเนื่องกันมากขึ้นและช่วยทำให้ขอบของข้อมูลที่ต้องการรักษาความคมชัดไว้มีความผิดพลาดน้อยลง

ในขั้นตอนการแบ่งบล็อกของภาพ ได้มีการใช้การแยกแยะภาพเข้ามาช่วยในการแยกภาพออกเป็นส่วนๆที่เป็นเนื้อเดียวกันในตัวเอง จากนั้นจะทำการลดข้อมูลโดยใช้การแปลงกับข้อมูลทีละขนาด ซึ่งเมื่อทำการแปลงกับข้อมูลที่มีความเป็นเนื้อเดียวกันแล้ว ข้อมูลภายหลังการแปลงจะมีพลังงานอัดแน่นมากและมีพลังงานในย่านความถี่สูงน้อยมาก ทำให้สามารถเลือกเฉพาะข้อมูลในย่านความถี่ต่ำได้ ส่งผลให้สามารถลดข้อมูลได้มากขึ้นโดยภาพผลลัพธ์ยังมีคุณภาพดี นอกจากนี้วิธีนี้ยังมีความยืดหยุ่นสูง เพราะจะจัดแบ่งบล็อกใหม่ให้เหมาะกับรายละเอียดในภาพทุกๆครั้ง อย่างไรก็ตามในการทำงานต้องมีการส่งรายละเอียดเกี่ยวกับขนาดบล็อกของข้อมูลไปให้ทางด้านรับด้วยเพื่อให้ทางด้านรับทำการประมาณได้ถูกต้อง

บทที่ 4

การส่งภาพแบบโปรเกรสซีพ

4.1ลักษณะทั่วไป

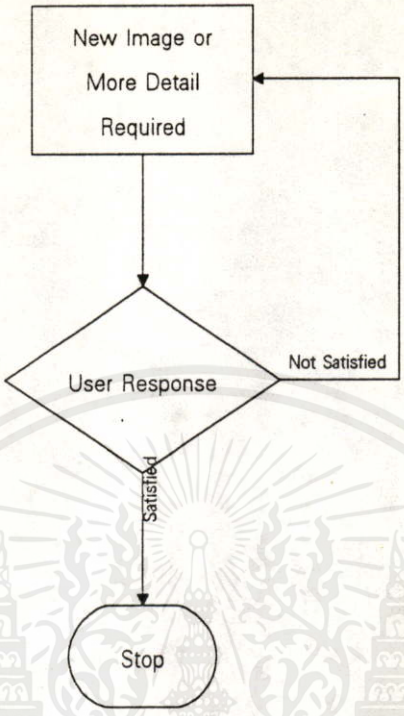
การส่งภาพแบบโปรเกรสซีพเป็นวิธีการจัดเรียงข่าวสารภายในภาพใหม่ โดยมีจุดประสงค์เพื่อการเพิ่มประสิทธิภาพของการสื่อสารผ่านสายสัญญาณความถี่ต่ำ[36-40] ภาพที่ถูกส่งออกไปจะเป็นชุดของการประมาณที่ดีขึ้นเรื่อยๆของภาพต้นแบบ การส่งภาพแบบโปรเกรสซีพจะใช้ความสามารถในการเข้าใจภาพของผู้ใช้มาช่วยลดเวลาในการสื่อสารลงได้มาก โดยเฉพาะในการสื่อสารแบบโต้ตอบที่มีมนุษย์เป็นผู้ใช้ขั้นสุดท้ายของภาพที่ได้รับ ในบางกรณีผู้ใช้อาจทราบลักษณะทั่วไปของภาพที่กำลังค้นหา การส่งภาพแบบโปรเกรสซีพจะช่วยให้ผู้ใช้สามารถตัดสินใจได้อย่างรวดเร็วว่าภาพที่เห็นเป็นภาพที่ต้องการหรือไม่ โดยการส่งภาพแบบโปรเกรสซีพจะทำการแบ่งส่งภาพไปเรื่อยๆและจากการส่งภาพแต่ละครั้งภาพที่ได้รับก็จะมีรายละเอียดมากขึ้นเรื่อยๆ ด้วยวิธีการนี้ผู้ใช้อาจทำการตัดสินใจเลือกภาพเลยหรือเรียกขอข้อมูลเพิ่มเติมในกรณีที่ยังไม่แน่ใจก็ได้ โฟลว์ชาร์ตที่แสดงข้างล่างเป็นการสรุปวิธีการทำงานของการส่งภาพแบบโปรเกรสซีพนี้ เมื่อพิจารณาเปรียบเทียบกับการนำภาพที่ถูกลดข้อมูลมาส่งโดยตรง ข้อมูลทั้งหมดซึ่งเป็นข้อมูลที่มีขนาดใหญ่จะถูกส่งออกมาพร้อมกันและทางด้านรับก็จะเห็นข้อมูลทั้งหมดในเวลาเดียวกัน ในกรณีที่ไม่เป็นภาพที่ต้องการข้อมูลที่ถูกส่งออกมาทั้งหมดก็เป็นการสูญเปล่า ในขณะที่การส่งภาพแบบโปรเกรสซีพนั้นผู้ใช้จะเห็นภาพในการส่งทุกๆครั้งและสามารถตัดสินใจได้ในระหว่างการสื่อสาร ทำให้ไม่ต้องรอให้เห็นภาพทั้งหมดอย่างละเอียดตามวิธีทั่วไป สำหรับเทคนิคที่นำมาใช้กับการส่งภาพแบบโปรเกรสซีพสามารถเป็นได้ทั้งแบบมีความสูญเสียและแบบไม่มีความสูญเสียจากการส่งภาพผลลัพธ์ในครั้งสุดท้าย กล่าวโดยสรุปแล้วการส่งภาพแบบโปรเกรสซีพจะมีข้อดีดังนี้

1.มีอัตราการใช้ข้อมูลต่ำในการประมาณขั้นแรกๆ ซึ่งผู้ใช้สามารถตัดสินใจยุติการส่งหรือให้ส่งข้อมูลเพิ่มเติม ในขั้นนี้มีการลดข้อมูลเป็นอย่างมาก

2.สามารถนำข้อมูลที่ส่งไปก่อนหน้ามาใช้ได้อย่างสมบูรณ์ ทำให้ข่าวสารที่จำเป็นต้องส่งเพิ่มมีขนาดเล็ก

3.มีความสามารถที่จะส่งภาพโดยใช้ข้อมูลขนาดเล็กแต่ให้ภาพที่มีคุณภาพดี

4.วิธีการเข้ารหัสและถอดรหัสทำได้ค่อนข้างเร็วและสามารถนำไปใช้งานกับอุปกรณ์ฮาร์ดแวร์ได้



รูปที่ 4.1 แสดงโฟลว์ชาร์ตในการส่งภาพแบบโปรเกรสซีพ

การส่งภาพแบบโปรเกรสซีพกำลังเป็นที่สนใจอย่างมากในการนำมาใช้ร่วมกับการสื่อสารแบบโต้ตอบ เช่น Telebrowsing[41] หรือการค้นหาภาพจากฐานข้อมูลจากระยะไกลๆ ที่มีความต้องการเลือกภาพที่ต้องการจากภาพจำนวนมากจึงต้องการการตัดสินใจที่รวดเร็ว นอกจากนี้ยังมีการใช้กันมากใน Teleconference หรือการประชุมทางไกล ซึ่งผู้ใช้ต้องการเห็นภาพลางๆของผู้ร่วมประชุมก่อนโดยไม่เสียเวลารอคอยยาวนาน

4.1.1 การจัดลำดับความสำคัญของภาพ

ลักษณะสำคัญพื้นฐานของการส่งภาพแบบโปรเกรสซีพคือการจัดลำดับความสำคัญของภาพ (Image Hierachies) ซึ่งในทางปฏิบัติสามารถอธิบายได้ดังนี้คือ ที่ความสำคัญของภาพที่ลำดับต่างๆกันนั้น แต่ละจุดภาพในภาพนั้นจะมีความคมชัดหรือคุณภาพต่างกันไป โครงสร้างสำคัญที่มักถูกนำมาใช้ในการจัดลำดับความสำคัญของภาพก็คือ โครงสร้างแบบปิรามิด

ในการแสดงภาพที่ได้จากการส่งแบบโปรเกรสซีพสามารถทำได้2วิธีคือ

1.การแสดงโดยให้ภาพมี Resolution ต่างกัน ภาพผลลัพธ์ในการส่งแต่ละครั้งจะมี Spatial Resolutionแตกต่างกันไป ซึ่งการแสดงภาพวิธีนี้สอดคล้องกับโครงสร้างปิรามิด โดยที่ฐานของปิรามิด

มิดเท่ากับภาพต้นแบบ ในขณะที่ปิรามิดอยู่ในอันดับที่สูงขึ้น ภาพก็จะมีขนาดเล็กลงและมี Spatial Resolution ลดลงด้วย

2.การแสดงโดยให้ภาพมี Resolution คงที่ ภาพผลลัพธ์ในการส่งแต่ละครั้งจะมีขนาดเท่ากันกับภาพต้นแบบและค่าของแต่ละจุดภาพในการส่งแต่ละครั้งมีการปรับปรุงภายหลังการส่งแต่ละครั้ง อย่างไรก็ตามภาพผลลัพธ์จากการแสดงโดยให้ภาพมีความคมชัดคงที่สามารถนำมาแสดงเป็นภาพที่มี Resolution ต่างกันได้โดยการทำอินเทอร์โปเลชั่น

ข้อแตกต่างที่สำคัญในการแสดงภาพทั้ง 2 วิธีก็คือ อัตราการเพิ่มบิท (Incremental Bit Rate) ซึ่งก็คือ จำนวนบิทที่ต้องนำไปใช้ในการส่งข้อมูลอีกระดับหนึ่งเมื่อเทียบกับจำนวนจุดภาพทั้งหมดของภาพต้นแบบ สำหรับการส่งโดยให้ภาพมีความคมชัดต่างกันแล้วอัตราการเพิ่มบิทสำหรับแต่ละระดับจะมีค่าคงที่ อย่างไรก็ตามการเพิ่มบิทอย่างคงที่ไม่ได้ส่งผลต่อการปรับปรุงคุณภาพอย่างคงที่ด้วย

เพื่อให้เข้าใจหลักการส่งภาพแบบโปรเกรสซีพให้ดียิ่งขึ้น จำเป็นต้องทราบลักษณะโครงสร้างปิรามิด[44]เป็นอย่างดี ดังนั้นต่อไปนี้จะอธิบายโครงสร้างนี้อย่างละเอียด โครงสร้างปิรามิดของภาพคือการนำภาพ $f_k(i,j)$ ที่ k=0,1,2,...,n มาเรียงใหม่เป็นชุดโดยที่แต่ละภาพมีขนาดเท่ากับ $2^k \times 2^k$ ซึ่งไม่เท่ากันและเป็นหน้าจั่ว ภาพชั้นบนสุด(k=0)มีขนาดเป็น1 ในขณะที่ภาพชั้นล่างสุด(k=n) จะมีขนาด $2^n \times 2^n$ เท่ากับของภาพจริง ส่วนชั้นระหว่างกลางเป็นภาพที่ขนาดและความคมชัดลดหลั่นกันไป ภาพที่ชั้นกลางถูกสร้างจากชั้นล่างขึ้นมา จุดภาพ $f_k(i,k)$ของชั้น k ถูกสร้างขึ้นจากจุดข้างเคียงทั้ง4 $f_{k+1}(i',j')$ที่ระดับ k+1

$$f_k(i,j) = g(f_{k+1}(2i,2j), f_{k+1}(2i,2j+1), f_{k+1}(2i+1,2j), f_{k+1}(2i+1,2j+1))$$
$$i,j = 0,1,2,\ldots,2^k - 1 \qquad (4.1)$$

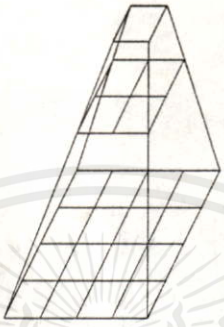
โดยที่ $g(\bullet)$ เป็น Mapping Function

จากการทำ Mapping ของภาพในระดับหนึ่งไปยังอีกระดับหนึ่งเทียบเท่ากับการทำ Low-Pass Filter ดังนั้นรายละเอียดและข้อมูลของภาพจะค่อยๆถูกตัดทิ้งไปเมื่ออยู่ในระดับที่สูงขึ้น ซึ่งชุดของภาพในลักษณะนี้จะเหมาะสมกับการลดข้อมูลและการสื่อสารมาก เพราะข้อมูลที่ได้เป็นลักษณะหยาบๆของภาพ

ถ้าสมมุติให้ภาพมีขนาดเท่ากับ $2^n \times 2^n$ หรือ 4^n จุดภาพ ดังนั้นจึงมีโครงสร้างปิรามิดสูงสุด n+1 ชั้น k=0,1,2,...,n เนื่องจากในแต่ละชั้นประกอบไปด้วยข้อมูลที่มีขนาดไม่เกิน $2^k \times 2^k$ หรือ 4^k ดังนั้นต้องใช้ข้อมูลมากที่สุดในการแสดงโครงสร้างปิรามิดนี้คือ

$$N = \sum_{k=0}^{n} 2^k \times 2^k = \frac{1-(2^{n+1} \times 2^{n+1})}{1-(2 \times 2)} = \frac{4}{3}(2^n \times 2^n) - \frac{1}{3} \approx \frac{4}{3}4^n \qquad (4.2)$$

ซึ่งหมายความว่า ข้อมูลของโครงสร้างปิรามิดมีขนาดเป็น $\frac{4}{3}$ เท่าของขนาดภาพที่จะแสดงนั้น



รูปที่ 4.2 แสดงลักษณะโครงสร้างปิรามิด

4.1.2 อินเทอร์โปเลชั่น

อินเทอร์โปเลชั่นเป็นวิธีการทางคณิตศาสตร์ที่ใช้ในการประมาณค่าฟังก์ชั่นที่จุดต่างๆหรือรูปแบบของฟังก์ชั่น เมื่อทราบข้อมูลของฟังก์ชั่นนั้นแล้วบางค่า ในการประมวลผลภาพมีการนำอินเทอร์โปเลชั่นเข้ามาใช้เพื่อช่วยในการเปลี่ยนลักษณะทางกายภาพของข้อมูลภาพ เช่น การขยายหรือการย่อภาพ การเปลี่ยนรูปร่างของภาพ(หมุนซ้าย-ขวา) ในการนี้จำเป็นต้องมีการเปลี่ยนแปลงพิกัดของภาพใหม่และค่าของข้อมูลที่พิกัดจะหาได้จากการประมาณข้อมูลภาพเดิมโดยใช้ Interpolation Function

ในหัวข้อต่อไปนี้จะพูดถึงลักษณะของ Interpolation Function แบบต่างๆที่มีใช้กันอยู่[4]ดังต่อไปนี้

1.Zero Order Interpolation หรือ Nearest Neighbor Assignment เป็นการอินเทอร์โปเลชั่นที่เร็วและง่ายที่สุดเนื่องจากไม่ต้องใช้การคำนวนเลย ค่าของข้อมูลภาพที่ตำแหน่งใหม่(x', y')จะเท่ากับค่าของข้อมูลภาพเดิม(x,y) ที่อยู่ใกล้ที่สุด วิธีนี้เทียบเท่ากับการยืดข้อมูลภาพเดิมที่มีขนาด $M \times N$ ไปเป็น $2M \times 2N$ โดยการเติมค่าศูนย์ในจุดที่เพิ่มขึ้นและข้อมูลภาพใหม่นี้จะถูกทำการคอนโวลูชั่นด้วย $\mathbf{H}$ ที่มีรูปแบบดังนี้

$$\mathbf{H} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad (4.3)$$

ซึ่งเท่ากับว่า

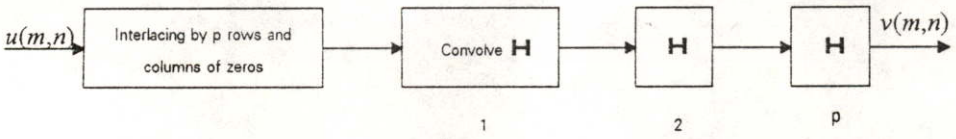
$$v(m,n) = u(k,l) \quad k \cong \mathbf{Int}[\frac{m}{2}] \quad l \cong \mathbf{Int}[\frac{n}{2}] \quad m,n = 0,1,2,... \tag{4.4}$$

อย่างไรก็ตาม การอินเทอร์โปเลชั่นวิธีนี้ให้ความไม่ต่อเนื่องตรงจุดที่ห่างจากข้อมูลภาพเดิม 1.5 จุดภาพ เนื่องจากผลของการตัดทิ้ง

2.First Order Interpolation หรือ Bilinear เป็นการอินเทอร์โปเลชั่นที่ให้ข้อมูลที่มีความกลมกลืนกันมากขึ้น โดยค่าของข้อมูลใหม่จะถูกประมาณจากข้อมูลเดิมที่อยู่รอบๆ การอินเทอร์โปเลชั่นแบบ Bilinear เทียบเท่ากับการยืดข้อมูลภาพเดิมออกไปโดยการเพิ่มค่าศูนย์ในจุดที่เพิ่มขึ้นแล้วทำการคอนโวลูชั่นด้วย$\mathbf{H}$ที่มีรูปแบบดังนี้

$$\mathbf{H} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix} \tag{4.5}$$

3.Higher Order Interpolation วิธีนี้สามารถทำได้โดยการยืดภาพต้นแบบจาก1จุดไปเป็นpจุดและpคอลัมน์ตามอันดับpของการอินเทอร์โปเลชั่น จากนั้นทำการคอนโวลูชั่นกับฟังก์ชั่นการอินเทอร์โปเลชั่น$\mathbf{H}$อีกpครั้ง เช่น การอินเทอร์โปเลชั่นอันดับที่3(p=3) หรือ Cubic Spline Interpolation ใช้การประมาณของฟังก์ชั่นซิงก์($\sin[\pi x]/\pi x$) ที่เป็นฟังก์ชั่นการอินเทอร์โปเลชั่นที่ดีที่สุดทางทฤษฎี โดยการประมาณวิธีนี้มีขั้นตอนการอินเทอร์โปเลชั่นดังต่อไปนี้



รูปที่ 4.3 *p*th order interpolation

4.2การส่งภาพแบบโปรเกรสซีพตามลักษณะโครงสร้างปิรามิด

การส่งภาพแบบโปรเกรสซีพมีรูปแบบที่สอดคล้องกับลักษณะโครงสร้างปิรามิดมาก ดังนั้นจึงมีนักวิจัยหลายท่านได้นำโครงสร้างปิรามิดมาใช้ในการแสดงภาพที่มีความคมชัดต่างกันเพื่อเตรียมทำ

การส่งแบบโปรเกรสซีพต่อไป ถึงแม้ว่าขนาดข้อมูลของโครงสร้างปิรามิดจะมีขนาดเป็น $4/3$ เท่าของขนาดข้อมูลภาพ แต่ในทางปฏิบัติแล้วสามารถคาดหวังได้ว่าผู้ใช้จะพอใจกับการแสดงภาพในครั้งต้นๆ ซึ่งหมายถึงข้อมูลที่มีขนาดเล็กกว่าข้อมูลของภาพจริงมากเสียเป็นส่วนใหญ่ ดังนั้นเพื่อให้สามารถใช้งานได้อย่างมีประสิทธิภาพ การส่งภาพแบบโปรเกรสซีพไม่เพียงแต่จะเน้นถึงขนาดข้อมูลในการแสดงภาพที่ควรจะต้องเล็กลงแล้วยังต้องพยายามที่จะปรับปรุงคุณภาพของภาพในการส่งแต่ละครั้งให้มีคุณภาพดีจนผู้ใช้สามารถคาดคะเนรายละเอียดของข้อมูลที่เหลือได้ด้วย ต่อจากนี้จะขอยกตัวอย่างหลักการส่งภาพแบบโปรเกรสซีพตามลักษณะโครงสร้างปิรามิดที่มีประสิทธิภาพดีบางวิธี ซึ่งความแตกต่างหลักๆจะอยู่ที่ Mapping Function

1.Mean Pyramid[37] เป็นลักษณะการสร้างปิรามิดแบบพื้นฐาน โดย Mapping Function ที่ใช้เป็นฟังก์ชั่นการเฉลี่ย ดังนั้นภาพที่อยู่ชั้นบนก็คือค่าเฉลี่ยของภาพในชั้นถัดลงไปนั่นเอง ซึ่งสามารถเขียนในรูปสมการได้เป็น

$$X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} = \frac{X_{k,i,j} + X_{k,i+1,j} + X_{k,i,j+1} + X_{k,i+1,j+1}}{4} \quad i,j = 1,3,\ldots,2^k - 1 \tag{4.6}$$

โครงสร้างปิรามิดนี้มีภาพที่ชั้นล่างสุดเป็นภาพต้นแบบนั่นเอง ส่วนภาพที่ชั้นบนสุดเป็นค่าเฉลี่ย ของทั้งภาพนั้น อย่างไรก็ตามโครงสร้างแบบนี้ยังมีปัญหาในการนำไปใช้งานหลายอย่างคือ

1.1ในการเก็บข้อมูลของแต่ละชั้นอย่างถูกต้อง จำเป็นต้องใช้จำนวนบิทสำหรับเก็บแต่ละข้อมูลมากกว่าข้อมูลภาพเดิม เพราะข้อมูลภาพเดิมจะเป็นเลขจำนวนเต็มแต่ข้อมูลใน Mean Pyramid เป็นเลขทศนิยม

1.2จำนวนข้อมูลทั้งหมดในโครงสร้าง Mean Pyramid จะมากกว่าข้อมูลทั้งหมดในภาพต้นแบบอยู่ 33.3%

1.3การหาค่าเฉลี่ยในพื้นที่เล็กๆ (2×2) มีผลเพียงเล็กน้อยต่อลักษณะการกระจายของโอกาสที่จะมีค่าของข้อมูลต่างๆ ดังนั้น Entropy อันดับ1 ของ Mean Pyramid, H_m จะมีค่าไม่แตกต่างจาก Entropy อันดับ1ของภาพต้นแบบ H_0 มากนัก

$$H_m \approx H_0 \tag{4.7}$$

อย่างไรก็ตามข้อบกพร่องดังกล่าวได้ถูกแก้ไขปรับปรุงขึ้นมาบ้างโดย

1.1ประมาณค่าของข้อมูลในปิรามิดให้มี Resolution เท่ากับของภาพจริง หรือการปัดเศษข้อมูลในปิรามิดให้เป็นเลขจำนวนจริงเหมือนภาพจริงซึ่งเท่ากับ

$$\hat{X}_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} = \frac{\hat{X}_{k,i,j} + \hat{X}_{k,i+1,j} + \hat{X}_{k,i,j+1} + \hat{X}_{k,i+1,j+1}}{4} \qquad i,j = 1,3,...,2^k - 1$$

(4.8)

เมื่อ$[\hat{\alpha}]$เป็นการปัดเศษของ$\alpha + 0.5$ และเราเรียกโครงสร้างนี้ว่า Truncated Mean Pyramid

1.2ทำการละทิ้งข้อมูลบางส่วนที่ซ้ำซ้อนกันลงไป จากสมการ สามารถนำมาจัดเรียงใหม่เป็น

$$\hat{X}_{k,i,j} = 4\hat{X}_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - \hat{X}_{k,i,j+1} - \hat{X}_{k,i+1,j} - \hat{X}_{k,i,j+1} - \hat{X}_{k,i+1,j+1} \quad (4.9)$$

นั่นก็คือ เมื่อทราบข้อมูลในชั้นปัจจุบันแล้วการแสดงภาพในชั้นถัดไปสามารถใช้ข้อมูลเพียง3จุดเพื่อแสดงข้อมูล4จุดได้ วิธีนี้เรียกว่า Reduced-Sum Pyramid และขนาดโครงสร้างนี้คือ

$$1 + \frac{3}{4}\sum_{k=1}^{n} 2^k \times 2^k = 1 + \frac{3}{4} \times 2 \times 2 \times \frac{1 - 2^n \times 2^n}{1 - 2 \times 2} = 2^n \times 2^n \qquad (4.10)$$

1.3แม้ว่า Mean Pyramid มีค่า Entropy พอๆกับของภาพต้นแบบ แต่ถ้านำข้อมูลมาแสดงในรูปแบบใหม่ เช่น ค่าผลต่างของแต่ละระดับใช้แทนค่าเฉลี่ยของตัวมันเอง ซึ่งค่าผลต่างนี้หาได้จาก

$$D_{k,i,j} = \hat{X}_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - \hat{X}_{k,i,j}$$

$$D_{k,i,j+1} = \hat{X}_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - \hat{X}_{k,i,j+1}$$

$$D_{k,i+1,j} = \hat{X}_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - \hat{X}_{k,i+1,j}$$

$$D_{k,i+1,j+1} = \hat{X}_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - \hat{X}_{k,i+1,j+1} \qquad (4.11)$$

โครงสร้างนี้ถูกเรียกว่า Difference ละ Pyramid Entropy อันดับ1ของข้อมูลในโครงสร้างนี้ H_Dจะมีค่าน้อยกว่า Entropy ของข้อมูลภาพจริงมาก

$$H_D < H_0 \qquad (4.12)$$

2.Gaussian-Laplacian Pyramid[38] ใช้ Mapping Function ที่มีลักษณะเป็นรูป Gaussian มาทำการคอนโวลูชั่นเพื่อให้ได้ภาพประมาณในชั้นสูงขึ้นไป ข้อมูลในโครงสร้างปิรามิดจะเหมือนกับการประมาณโดยใช้ตัวกรองความถี่ต่ำ ซึ่งในการกรองนี้ให้ผลเทียบเท่ากับการทำคอนโวลูชั่นด้วยตระกูลของฟังก์ชั่นถ่วงน้ำหนักสมมาตรของข้อมูลภายใน และสมาชิกในตระกูลนี้มีลักษณะคล้ายกับการกระจายความน่าจะเป็นแบบ Gaussian ดังนั้นชุดข้อมูลที่เรียงซ้อนกันในปิรามิดจึงถูกเรียกว่า Gaussian Pyramid

ข้อมูลภาพใน Gaussian Pyramid ถูกสร้างขึ้นโดย

$$G_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} = \sum_{m,n} w(m,n) G_{k,i+m,j+n} \quad i,j = 1,3,\ldots,2^k - 1 \tag{4.13}$$

เมื่อw(m,n)เป็นฟังก์ชั่นถ่วงน้ำหนักที่มีลักษณะเป็น Gaussian และในทางปฏิบัติมักกำหนดให้ฟังก์ชั่นนี้เป็นตารางหน้าต่างขนาด 5×5 เพื่อให้ผลที่ดีโดยที่มีภาระในการคำนวนน้อย ดังนั้นการเลือกฟังก์ชั่นถ่วงน้ำหนักอาจพิจารณาให้ฟังก์ชั่นมีคุณสมบัติ Separable

$$w(m,n) = \hat{w}(m)\hat{w}(n) \tag{4.14}$$

ฟังก์ชั่น1มิติ$\hat{w}$ ความยาว5 จะถูก Normalize และกำหนดให้มีความสมมาตร

$$\begin{aligned} &\sum_{m=-2}^{2} \hat{w}(m) = 1 \\ &\hat{w}(i) = \hat{w}(-i) \quad i = 0,1,2 \end{aligned} \tag{4.15}$$

ข้อกำหนดอีกประการหนึ่งคือลักษณะการกระจายเท่าๆกัน ซึ่งยอมให้ข้อมูลภาพในปิรามิดที่ชุดนั้นมีผลรวมของการถ่วงน้ำหนักเท่ากับ0.25ของข้อมูลในชั้นสูงขึ้นไป1ลำดับ

ถ้ากำหนดให้ $\hat{w}(0) = a, \hat{w}(-1) = \hat{w}(1) = b, \hat{w}(-2) = \hat{w}(2) = c$

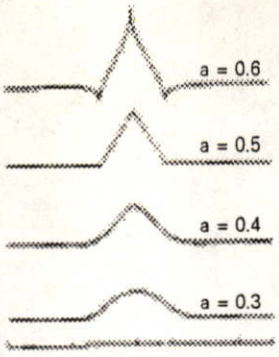
จากการกระจายเท่ากัน $a + 2c = 2b$

$$\hat{w}(0) = a$$

นั่นคือ $\hat{w}(-1) = \hat{w}(1) = \frac{1}{4}$ (4.16)

$$\hat{w}(-2) = \hat{w}(2) = \frac{1}{4} - \frac{a}{2}$$

รูปต่อไปจะแสดงลักษณะของฟังก์ชั่นถ่วงน้ำหนักที่ตัวแปรต่างๆ



รูปที่ 4.4 ลักษณะของฟังก์ชั่นถ่วงน้ำหนักที่ค่า a ต่างๆ

หลังจากที่ Gaussian Pyramid ถูกสร้างขึ้นมา ในขั้นต่อไปจะทำการจัดตั้ง Laplacian Pyramid ซึ่งข้อมูลในโครงสร้างใหม่นี้มีข้อมูลที่มีความสัมพันธ์กันลดลงกว่าข้อมูลของโครงสร้าง Gaussian Pyramid และ Entropy ของโครงสร้างใหม่จะน้อยกว่าของเดิมเช่นเดียวกัน ข้อมูลใน Laplacian Pyramid หาได้จากการหาความแตกต่างในแต่ละชั้นของ Gaussian Pyramid หรือ

$$L_{k,i,j} = G_{k,i,j} - \sum_{m,n} w(m,n) G_{k-1,[\frac{i-m}{2}],[\frac{j-n}{2}]} \qquad i,j = 1,2,...,2^k \tag{4.17}$$

Laplacian Pyramid มีข้อมูลที่สำคัญเพียงพอที่จะสร้าง Gaussian Pyramid กลับมาใหม่ได้โดย

$$G_{k,i,j} = L_{k,i,j} + \sum_{m,n} w(m,n) G_{k-1,[\frac{i-m}{2}],[\frac{j-n}{2}]} \qquad i,j = 1,2,...,2^k \tag{4.18}$$

ดังนั้นในการสื่อสารจะใช้ข้อมูลใน Laplacian Pyramid ส่งไปยังด้านรับ เพราะ Entropy ของ Laplacian Pyramid นี้มาสร้างเป็น Gaussian Pyramid ก่อนและประมาณภาพต้นแบบในขั้นต่อไป

4.3การส่งภาพแบบโปรเกรสซีพโดยใช้การแปลง

การส่งภาพแบบโปรเกรสซีพเป็นการส่งภาพซ้อนกันไปเรื่อยๆ โดยในการส่งครั้งแรกๆภาพจะมีลักษณะหยาบและค่อยๆมีความคมชัดมากขึ้นในการส่งครั้งต่อๆไป เมื่อพิจารณาภาพทั้งชุดแล้วพบว่าภาพในการส่งก่อนหน้าเหมือนกับเป็นภาพในการส่งครั้งต่อไปที่ผ่านตัวกรองความถี่ต่ำ ดังนั้นจึงสามารถทำการส่งภาพแบบโปรเกรสซีพได้อีกวิธีหนึ่ง โดยนำการแปลงมาใช้เป็นตัวกรองเพราะการ

มารถทำการส่งภาพแบบโปรเกรสซีพได้อีกวิธีหนึ่ง โดยนำการแปลงมาใช้เป็นตัวกรองเพราะการแปลงสามารถเทียบได้กับการทำ Frequency Decomposition นั่นเอง

เมื่อพิจารณาเปรียบเทียบระหว่างการส่งภาพแบบโปรเกรสซีพโดยใช้การแปลงกับการส่งภาพแบบโปรเกรสซีพโดยใช้โครงสร้างแบบปิรามิดแล้วพบว่าในแต่ละวิธีจะมีข้อดีข้อเสียต่างกันไปดังนี้คือ

ข้อดีของการส่งภาพแบบโปรเกรสซีพโดยใช้การแปลง

1.สามารถสร้างภาพในแบบโปรเกรสซีพได้ง่าย เพราะข้อมูลภาพภายหลังการแปลงในโดเมนความถี่มักมีพลังงานส่วนใหญ่อัดแน่นอยู่ในช่วงความถี่ต่ำ ดังนั้นในการส่งภาพวิธีนี้ก็จะเลือกข้อมูลที่มีความถี่ต่ำบางค่าไปก่อนทำให้เห็นภาพคร่าวๆ จากนั้นจะส่งข้อมูลในย่านความถี่สูงขึ้นตามไปเรื่อยๆ ทางด้านรับก็จะได้รับรายละเอียดในภาพมากขึ้นเรื่อยๆเช่นกัน นอกจากนี้ข้อมูลในการส่งภาพแต่ละครั้งจะไม่มีความซ้ำซ้อนกัน นั่นคือขนาดของข้อมูลทั้งหมดในการส่งวิธีนี้เท่ากับขนาดของภาพต้นแบบ ทำให้เวลาในการส่งภาพไม่มากไปกว่าวิธีการส่งภาพแบบปกติแน่นอน

2.ในการส่งภาพแต่ละครั้ง จะใช้ข้อมูลที่มีขนาดเล็กกว่า เนื่องจากข้อมูลที่ทำการส่งนี้ได้ผ่านการลดข้อมูลมาก่อนแล้ว ทำให้ข้อมูลที่ต้องใช้ในการส่งมีความสัมพันธ์ต่อเนื่องกันน้อยทำให้Entropyของข้อมูลลดลงและส่งผลให้สามารถส่งข้อมูลด้วยขนาดที่เล็กกว่าวิธีปกติได้

3.ในการรับภาพแต่ละครั้ง ทางด้านรับจะได้รับข้อมูลของภาพที่มีResolutionเท่ากับภาพต้นแบบเลย ดังนั้นจึงไม่จำเป็นต้องนำข้อมูลที่ได้รับมาทำการอินเทอร์โปเลชั่นก่อน

ข้อดีของการส่งภาพแบบโปรเกรสซีพโดยใช้โครงสร้างแบบปิรามิด

1.มีวิธีการทำงานที่ไม่ซับซ้อนและไม่ต้องใช้การคำนวนมากนัก เนื่องจาก Mapping Function ที่ใช้มักเป็นการกระทำทางคณิตศาสตร์ง่ายๆ

2.ในการรับภาพแต่ละครั้ง ทางด้านรับจะได้รับข้อมูลของภาพที่สามารถนำไปแสดงในแบบที่มี Resolution ต่างกันได้ทันที

3.ภาพที่ได้ในขั้นสุดท้ายคือภาพต้นแบบนั่นเอง ดังนั้นผู้ใช้สามารถเห็นภาพที่มีรายละเอียดเหมือนภาพจริงได้แน่นอน แต่ต้องระวังความผิดพลาดจากการส่งครั้งแรกๆซึ่งความผิดพลาดเล็กน้อยจะถูกสะสมกับความผิดพลาดในการส่งครั้งต่อไปจนอาจมีผลอย่างสำคัญในการส่งครั้งหลังๆ

การส่งภาพแบบโปรเกรสซีพตามเทคนิคนี้มีจุดมุ่งหมายที่จะส่งข้อมูลส่วนที่สำคัญที่สุดไปก่อน ดังนั้นข้อมูลทั้งหมดต้องถูกตัดแบ่งเป็นส่วนๆและนำมาจัดเรียงใหม่ตามลำดับความสำคัญของข้อมูล ในการจัดเรียงนี้พบว่าลักษณะของข้อมูลในโดเมนความถี่จะมีความเหมาะสมมากกว่า เพราะนอกจากจะถูกจัดเรียงตามลำดับความถี่ที่เพิ่มขึ้นแล้ว พลังงานส่วนใหญ่ยังรวมตัวอยู่ในย่านความถี่ต่ำอีกด้วย เนื่องจากข้อมูลที่ความถี่ต่ำแสดงโครงร่างของภาพ ดังนั้นถ้าเลือกเฉพาะข้อมูลส่วนนี้จะเห็นภาพที่ไม่คมชัดเหมือนผ่านตัวกรองความถี่ต่ำนั่นเอง ส่วนข้อมูลที่ความถี่สูงทำให้เกิดภาพในส่วนที่เป็นขอบหรือรายละเอียดเหมือนผ่านตัวกรองความถี่สูง-

ปัจจัยที่มีผลสำคัญต่อเรื่องคุณภาพของภาพในการส่งภาพขั้นต่างๆ ก็คือบริเวณข้อมูลที่นำไปใช้ ดังนั้นจึงได้มีความพยายามจัดเรียงข้อมูลใหม่ตามลักษณะความสำคัญ เช่น

1.การจัดเรียงสัมประสิทธิ์ของข้อมูลใหม่[24] โดยจะทำการเรียงอันดับความสำคัญของข้อมูลไว้อย่างคงที่ ในการส่งภาพจะนำข้อมูลตามลำดับในอนุกรมนั้นมาใช้งาน

2.การทำ Zonal Sampling กับแต่ละย่านความถี่[39] จากหัวข้อ 3.1 ทำให้เราทราบลักษณะข้อมูลหลังการแปลงว่ามีความหนาแน่นที่บริเวณความถี่ต่ำและลดลงอย่างรวดเร็วที่ความถี่สูงขึ้น นอกจากนี้ยังสามารถทำนายโครงสร้างของข้อมูลหลังการแปลงได้โดยโครงสร้างของเมตริกซ์ Covariance ที่แยกจากกันหรือโครงสร้างของเมตริกซ์C ovariance แบบ Isotropic จากโครงสร้างดังกล่าวทำให้สามารถเลือกส่วนที่มีความสำคัญในภาพได้ดียิ่งขึ้น

3.การเลือกสัมประสิทธิ์ที่มีความสำคัญมากที่สุดก่อน[40] วิธีนี้ถือว่าข้อมูลที่มีค่าความผันผวนสูงสุดมีความสำคัญมากที่สุด เพราะบริเวณนั้นจะมีพลังงานมากที่สุดและมีผลต่อภาพที่ประมาณขึ้นมากที่สุดด้วย วิธีนี้จะเริ่มจากการค้นหาข้อมูลที่มีค่าความผันผวนสูงสุด ซึ่งข้อมูลนี้ถูกจัดให้มีความสำคัญมากที่สุดและถูกส่งไปเป็นอันดับแรก วิธีนี้สามารถส่งข้อมูลส่วนที่สำคัญมากได้จริงแต่ยังมีปัญหาอยู่ที่ค่าใช้จ่ายในการส่งข้อมูลจะสูงมาก เพราะจำเป็นต้องส่งตำแหน่งของข้อมูลในทุกอันดับไปให้ด้านรับด้วย

วิธีที่ใช้การแปลงสำหรับการส่งภาพแบบโปรเกรสซีพ จะทำการเพิ่มข่าวสารในโดเมนความถี่สำหรับการส่งแต่ละครั้ง โดยวิธีการเลือกบริเวณสำหรับส่งข้อมูลจะทำการส่งข้อมูลทีละย่านความถี่คือจะส่งข้อมูลบริเวณความถี่ที่สูงขึ้นในการส่งครั้งต่อไป ส่วนวิธีการปรับระดับความละเอียดให้กับข้อมูลภาพจะส่งข้อมูลที่ไม่ละเอียดในโดเมนความถี่ทั้งหมดไป และในการส่งครั้งต่อไปจะเพิ่มความละเอียดของข้อมูลทั้งหมดส่งไปให้ด้านรับ ตั้งแต่รูปที่ 4.5.1 ถึง 4.5.8 เป็นผลการส่งภาพแบบโปรเกรสซีพตั้งแต่ครั้งที่ 1 ถึง 8โดยใช้การแปลง

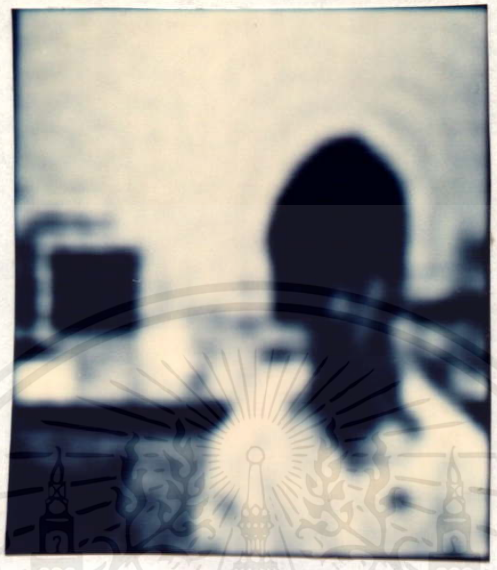
สำหรับความผิดพลาดที่เกิดขึ้นจากการส่งภาพแบบโปรเกรสซีพโดยใช้การแปลงในการควอนไทซ์นั้นสามารถถูกกำจัดไปได้ โดยมันจะถูกป้อนกลับมารวมกับข้อมูลที่จะมาควอนไทซ์ในครั้งต่อไป และความผิดพลาดนี้ (Residual Error) จะถูกทยอยส่งไปเรื่อยๆ ในทางทฤษฎีแล้วความผิดพลาดทั้งหมดในภาพผลลัพธ์จะเป็นศูนย์เมื่อทำการส่งถึงครั้งที่อนันต์ แต่ในทางปฏิบัติสามารถลดจำนวนการส่งลงได้โดยเมื่อส่งถึงระดับหนึ่งที่มากเพียงพอแล้วก็จะทำการเข้ารหัสข่าวสารกับResidual Errorที่ยังเหลืออยู่ แล้วส่งให้ด้านรับ ด้วยวิธีการนี้ทำให้การส่งภาพแบบโปรเกรสซีพในครั้งสุดท้ายจะได้ข้อมูลที่ไม่มีความสูญเสียเลย

รูปที่ 4.5.1 ผลการส่งภาพแบบโปรเกรสซีพครั้งที่ 1

รูปที่ 4.5.2 ผลการส่งภาพแบบโปรเกรสซีพครั้งที่ 2

รูปที่ 4.5.3 ผลการส่งภาพแบบโปรเกรสซีพครั้งที่ 3

รูปที่ 4.5.4 ผลการส่งภาพแบบโปรเกรสซีพครั้งที่ 4



รูปที่ 4.5.5 ผลการส่งภาพแบบโปรเกรสซีพครั้งที่ 5

รูปที่ 4.5.6 ผลการส่งภาพแบบโปรเกรสซีพครั้งที่ 6

รูปที่ 4.5.7 ผลการส่งภาพแบบโปรเกรสซีพครั้งที่ 7

รูปที่ 4.5.8 ผลการส่งภาพแบบโปรเกรสซีพครั้งที่ 8

บทที่ 5

ระบบการสื่อสารภาพแบบโปรเกรสซีพ

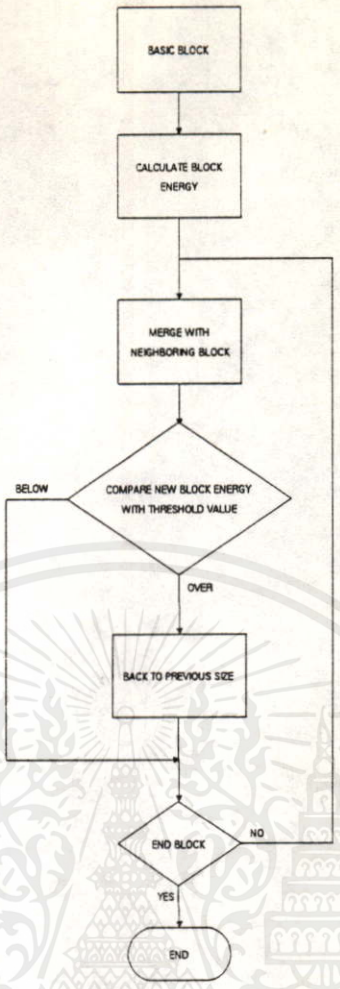
5.1 การออกแบบ

5.1.1 ส่วนแยกแยะภาพ

ในการแยกแยะภาพ เราเลือกใช้วิธี Region Growing เนื่องจากต้องการแยกพื้นที่ส่วนที่มีรายละเอียดมากซึ่งเป็นส่วนที่ผู้ใช้สังเกตได้ชัดเจนที่สุดออกมาพิจารณาความเป็นเนื้อเดียวกันเสียก่อน ถึงแม้ว่าการแยกแยะภาพแบบ Region Splitting สามารถปฏิบัติได้ง่ายกว่า เพราะข้อมูลภาพส่วนใหญ่จะมีความสัมพันธ์กันสูงดังนั้นพื้นที่บริเวณกว้างจึงมีความเป็นเนื้อเดียวกันสูงตามไปด้วย แต่บล็อกย่อยใหม่ที่ได้จะถูกจัดให้เป็นรูปร่างสี่เหลี่ยมจตุรัสเท่านั้น ทำให้ไม่สามารถติดตามรายละเอียดได้ดีและส่งผลให้ไม่สามารถจัดแบ่งภาพให้มีความเป็นเนื้อเดียวกันสูงสุดได้ ขั้นตอนการทำงานจะเริ่มจากการแบ่งภาพเป็นบล็อกย่อย ซึ่งบล็อกย่อยนี้ต้องมีขนาดเล็กเพียงพอที่จะแสดงความเป็นเนื้อเดียวกันภายในได้อย่างชัดเจน แต่ในขณะเดียวกันบล็อกนี้ก็ต้องไม่เล็กเกินไปจนไม่สามารถหาความสัมพันธ์ของข้อมูลเพื่อที่จะนำไปใช้ในการลดข้อมูลต่อไปได้ หลังจากทำหารทดสอบบล็อกย่อยขนาดต่างๆ ในวิทยานิพนธ์นี้เลือกใช้บล็อกย่อยขนาด 4×4 เป็นมาตรฐาน เพราะบล็อกขนาดนี้สามารถรักษาความเป็นเนื้อเดียวกันภายในภาพส่วนใหญ่ได้

ขั้นต่อไปจะนำบล็อกย่อยๆซึ่งต่อไปจะเรียกว่า Base มาทำการหาความเป็นเนื้อเดียวกันภายในจริงๆแล้วลักษณะรายละเอียดสามารถวัดได้หลายวิธี เช่น ค่าพิสัย,ค่าเฉลี่ย,ค่าสูงสุด,ค่าต่ำสุด เป็นต้น แต่ในวิทยานิพนธ์นี้เลือกใช้ค่าความผันผวนเป็นตัวเปรียบเทียบ เพราะจากความรู้ในบทก่อนๆทำให้ทราบว่าระดับของค่าผันผวนจะแสดงถึงพลังงานภายในบล็อก และระดับพลังงานภายในบล็อกสามารถระบุปริมาณรายละเอียดหรือกิจกรรมในบล็อกนั้นได้

หลังจากที่ได้ข้อมูลของแต่ละ Base ครบถ้วนแล้ว ภาพผลลัพธ์ที่ได้จะถูกแบ่งเป็นบล็อกขนาดไม่เท่ากัน โดยบริเวณที่เป็นขอบหรือบริเวณที่มีรายละเอียดสูงจะถูกแบ่งเป็นบล็อกขนาดเล็ก ส่วนบริเวณที่เป็น Background ถูกจัดให้มีบล็อกขนาดใหญ่ เพื่อให้เข้าใจกับการทำงานของส่วนนี้ให้ชัดเจนขึ้นขอให้พิจารณาจากโฟลว์ชาร์ตต่อไปนี้ประกอบด้วย



รูปที่ 5.1 โฟลว์ชาร์ตแสดงการแบ่งภาพ

Region Representation

เนื่องจากขนาดและตำแหน่งของบล็อกที่แบ่งแยกใหม่นี้ ทางด้านรับต้องรับทราบเพื่อจะได้ถอดรหัสข้อมูลที่ส่งมาให้เหมาะสม ดังนั้นจึงต้องสร้างเมตริกซ์ใหม่ขึ้นมาชนิดหนึ่งเรียกว่า เมตริกซ์แสดงการเชื่อมต่อ (Linkage Matrix) เพื่อเก็บตำแหน่งและขนาดของบล็อกใหม่ โดยเมตริกซ์นี้ต้องถูกส่งให้ทางด้านรับก่อนที่จะเริ่มส่งข้อมูลภาพ โดยเราจะแทนแต่ละ Base ด้วย $B_1, B_2, \ldots, B_n$ โดยที่ nคือจำนวนบล็อกย่อยทั้งหมดในภาพและเซตของ B คือเซตของ Base ในภาพต้นแบบ สำหรับบล็อกที่มีการรวมกันขึ้น บล็อกย่อยที่รวมกันจะมีการเชื่อมต่อกันด้วยสายเชื่อม(Link); $L_1, L_2, \ldots, L_n$ เพื่อให้การเก็บตำแหน่งมีขนาดกะทัดรัดในแต่ละบล็อกที่ถูกรวมกันจะมีการเชื่อมต่อกันเฉพาะในแนวตั้งฉากและแนวนอน และจะไม่มีการเชื่อมต่อระหว่างบล็อกใหม่ เมื่อเชื่อมต่อกันเสร็จทั้งภาพแล้วจะทำการให้รหัสการเชื่อมต่อสำหรับบล็อกย่อย โดยจะเริ่มพิจารณาจากต้นบล็อกหรือบล็อกย่อยที่อยู่ทางมุมบนด้านซ้ายของบล็อกใหม่ และกำหนดให้ค่า L ของบล็อกย่อยนั้นเป็น 1 จากนั้นจึงจะพิจารณาสายเชื่อมต่อหาบล็อกย่อยอื่นที่ถูกรวมให้อยู่ในบล็อกใหม่เดียวกัน โดยกำหนดให้ค่า L ของบล็อกย่อยที่อยู่ใต้ต้นบล็อกพอดีเป็น 3 ส่วนบล็อกย่อยอื่นที่มีสายเชื่อมกับต้นบล็อกทางแนวนอนจะ

ถูกกำหนดให้มีค่า L เป็น 2 จากรูปแบบดังกล่าว ถ้าเราใช้บล็อกย่อยขนาด 4×4 จะสามารถเขียนบล็อกใหม่ขนาด $16 \times 16, 8 \times 8, 8 \times 4, 4 \times 8, 4 \times 4$ ได้ดังรูปต่อไปนี้

1 → 2 → 2 → 2
↓ ↓ ↓ ↓
3 → 2 → 2 → 2
↓ ↓ ↓ ↓
3 → 2 → 2 → 2
↓ ↓ ↓ ↓
3 → 2 → 2 → 2

Linkage Matrix ของพื้นที่ขนาด 16×16

1 → 2
↓ ↓
3 → 2

Linkage Matrix ของพื้นที่ขนาด 8×8

1 → 2

Linkage Matrix ของพื้นที่ขนาด 4×8

1
↓
3

Linkage Matrix ขนาด 8×4

1

Linkage Matrix ขนาด 4×4

ถ้านำค่าใน Linkage Matrix มาใช้โดยตรงต้องใช้ข้อมูล 2 บิทต่อ 1 Base แต่จากการพิจารณาโอกาสในการเกิดค่าต่างๆใน Linkage Matrix แล้วจะพบว่า โอกาสในการเกิดค่า"2"มีมากที่สุดตามด้วย"1"และ"3"ตามลำดับ จากข้อมูลดังกล่าวทำให้สามารถลดข้อมูลของ Linkage Matrix เพิ่มเติมได้อีก โดยกำหนดให้แต่ละค่าของ Linkage Matrix แทนด้วยรหัสที่มีความยาวไม่คงที่ตามตารางต่อไปนี้

ค่าLink	รหัส
"1"	"0"
"2"	"10"
"3"	"110"

5.1.2ส่วนการเข้ารหัสภาพ

อันที่จริงแล้ว ภาพที่ได้หลังทำการแยกแยะภาพจะมีรูปร่างต่างๆมากมายแล้วแต่ขอบภายในภาพว่ามีอยู่ที่ไหน แต่เพื่อไม่ให้การทำงานของระบบมีความซับซ้อนมากเกินไป จึงได้จำกัดลักษณะของบล็อกย่อยหลังการแยกแยะภาพไว้ 5 ประเภทเท่านั้นคือ $16\times16, 8\times8, 8\times4, 4\times8, 4\times4$ โดยที่บล็อกขนาด 4×4 นำมาใช้กับบล็อกที่รายละเอียดมาก, บล็อกขนาด 4×8 นำมาใช้กับบล็อกที่มีรายละเอียดมากตามแนวนอน, บล็อกขนาด 8×4 นำมาใช้กับบล็อกที่มีรายละเอียดมากตามแนวตั้ง, บล็อกขนาด 8×8 นำมาใช้กับบล็อกที่มีรายละเอียดปานกลาง และบล็อกขนาด 16×16 นำมาใช้กับบล็อกที่มีรายละเอียดน้อย

เนื่องจากในการแบ่งภาพจากการทำ Region Growing กำหนดให้บล็อกย่อยมีขนาดต่างกัน 5 ขนาด ซึ่งบางขนาดก็ไม่อยู่ในรูปสี่เหลี่ยมจตุรัส จึงไม่สามารถทำการเข้ารหัสพร้อมกันทีละ 2 มิติได้ ในการทำงานจะกำหนดให้ทำการแปลงข้อมูลทีละมิติ โดยที่การแปลงแต่ละครั้งจะใช้ Basis Vector ไม่เท่ากัน ซึ่งการแปลงในลักษณะนี้สามารถทำได้ดังนี้

$$F(u,y) = c\sum_{x=0}^{n-1} f(x,y)\cos[\frac{\pi(2n+1)x}{2N}]$$
$$F(u,v) = c\sum_{y=0}^{n-1} F(u,y)\cos[\frac{\pi(2n+1)y}{2N}] \qquad (5.1)$$

ข้อมูลหลังการแปลงนี้จะมีความละเอียดมาก เพื่อให้เกิดผลในการลดข้อมูลจึงจำเป็นต้องรักษาความละเอียดไว้ให้ข้อมูลในส่วนที่มีความสำคัญมากหรือในบริเวณความถี่ต่ำ โดยการควอนไทซ์จะทำการลดความละเอียดของข้อมูลลงเพื่อจัดเก็บข้อมูลให้มีขนาดเล็กลงตามค่าบิทสำหรับความละเอียดที่ตำแหน่งนั้น

สำหรับในการควอนไทซ์ข้อมูลหลังการแปลงนี้ จะถือว่าข้อมูลDCและACมีการกระจายแบบ Gaussian[9] ดังนั้นในการสร้างค่าประมาณของการควอนไทซ์จะเลือกใช้วิธีของ Llyod-

Max สำหรับข้อมูลที่มีการกระจายแบบ Gaussian ซึ่งค่าการประมาณทั้งหมดสามารถรวบรวมเป็นตารางบรรจุในส่วนของโปรแกรมท้ายเล่ม

แม้ว่าในขั้นตอนการควอนไทซ์ถือว่าข้อมูลทั้งหมดมีการกระจายในลักษณะเดียวกัน แต่ในความเป็นจริงแล้วข้อมูลหลังการแปลงที่ตำแหน่งต่างๆมีค่าแตกต่างกันมาก ทำให้ต้องใช้ตารางควอนไทซ์เฉพาะตัว โดยแต่ละตารางก็จะมีความกว้างไม่เท่ากัน ในการทำงานจึงนิยมทำการควอนไทซ์กับข้อมูลที่เป็น Unit Variance กล่าวคือข้อมูลทั้งหมดจะถูกนำมาเปรียบเทียบในมาตราเดียวกันว่าข้อมูลแต่ละตัวมีการเบี่ยงเบนจากข้อมูลเฉลี่ยที่ตำแหน่งนั้นมากน้อยเท่าใด ซึ่งข้อมูลหลังการแปลงจะถูกถ่วงน้ำหนักด้วยค่าเบี่ยงเบนมาตรฐานที่ตำแหน่งนั้นๆ โดยที่ค่าเบี่ยงเบนมาตรฐานนี้สามารถหาได้จากการคำนวนจริงๆหรือการประมาณขึ้นมาก็ได้

5.1.3 ส่วนการสื่อสารข้อมูล

เมื่อได้รับข้อมูลที่มีขนาดเล็กลงหลังจากผ่านการควอนไทซ์มาแล้ว ในส่วนของการสื่อสารจะทำการจัดลำดับความสำคัญให้กับข้อมูลแต่ละส่วนเพื่อให้สอดคล้องกับการส่งภาพแบบโปรเกรสซีพ และการจัดเก็บข้อมูลในรูปแบบที่ทางด้านรับสามารถเข้าใจได้ ซึ่งในการจัดเรียงลำดับความสำคัญของข้อมูลเพื่อที่จะเลือกนำมาส่งในแต่ละครั้งได้มีการนำวิธี Zonal Sampling มาใช้กล่าวคือ สำหรับข้อมูลหลังการแปลงแต่ละบล็อกจะมีลักษณะทางสถิติใกล้เคียงกับ Isotropic Covariance Model ดังนั้นในการส่งข้อมูลแต่ละครั้งจะทำการเลือกหรือ Mask ข้อมูลแต่ละส่วนเอาไว้ตามลักษณะของ Isotropic Covariance Model และในการส่งครั้งถัดไปจะเลือกข้อมูลในส่วนที่เป็นความถี่สูงขึ้นและไม่จำเป็นต้องส่งข้อมูลในครั้งที่แล้วซ้ำอีก

ในระบบนี้ได้ทำการแบ่งข้อมูลหลังการแปลงออกเป็นหลายๆส่วนสำหรับการส่งหลายๆครั้งกับข้อมูลแต่ละขนาดดังต่อไปนี้

ข้อมูลขนาด 4×4

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

ข้อมูลขนาด 4×8

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

ข้อมูลขนาด 8×4

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

ข้อมูลขนาด 8×8

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

ข้อมูลขนาด 16×16

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

หลังจากที่ได้ทำการจัดเรียงลำดับความสำคัญให้กับข้อมูลเรียบร้อยแล้ว จะทำการจัดเก็บข้อมูลให้อยู่ในสภาพที่พร้อมจะสื่อสาร โดยเริ่มนำข้อมูลส่วนที่มีลำดับความสำคัญสูงสุดมาจัดเก็บก่อน อย่างไรก็ตามข้อมูลในขั้นตอนนี้ยังคงมีความเกี่ยวข้องกันอยู่บ้าง เพราะส่วนใหญ่จะมีค่าน้อยหรือค่าเป็นศูนย์ จึงเป็นไปได้ที่จะลดข้อมูลเพิ่มเติมโดยไม่ให้เกิดความสูญเสียอีก วิธีที่นิยมใช้กันก็คือใช้การเข้ารหัสแบบ Run Length ในการเก็บตำแหน่งข้อมูล และใช้การเข้ารหัสแบบ Huffman สำหรับเก็บ

ขนาดข้อมูลที่ตำแหน่งนั้นๆ อย่างไรก็ตามในวิทยานิพนธ์ฉบับนี้ได้เลือกใช้การเก็บข้อมูลให้มีขนาดไม่คงที่ (Variable Block-size Coding) ซึ่งขนาดเนื้อที่ในการเก็บจะมีค่าเท่ากับจำนวนบิทสำหรับความละเอียดของข้อมูลที่ตำแหน่งนั้นๆ

5.2 ผลการทำงาน

จากการทดลองโดยใช้ภาพต้นแบบในรูปที่ 2.2 และ 2.3 ซึ่งเป็นภาพที่มีขนาด 128×128 จุดภาพ ในแต่ละจุดภาพใช้หน่วยความจำขนาด 8 บิททำให้สามารถแสดงความละเอียดได้ทั้งหมด 256 ระดับ ดังนั้นภาพต้นแบบแต่ละรูปต้องใช้เนื้อที่ขนาด 16384 ไบท์ ในส่วนต่อไปนี้เป็นการทดลองนำภาพทั้งสองมาทำการจัดส่งตามแบบโปรเกรสซีพโดยใช้วิธีที่อธิบายในวิทยานิพนธ์ฉบับนี้ โดยตารางที่ 1 แสดงขนาดของพื้นที่ที่แบ่งได้และอัตราบิทสำหรับความละเอียดในแต่ละพื้นที่นี้และในตารางที่ 2 เป็นผลการลดข้อมูลและอัตราการสูญเสียที่เกิดขึ้นจากการเข้ารหัสในการส่งแต่ละครั้ง (PSNR) รูปที่ 5.1.1 และ 5.1.2 แสดงผลการแบ่งภาพ'KRIT'และภาพ'BABOON'ออกเป็นพื้นที่ย่อยๆตามวิธีที่เสนอ ตั้งแต่รูปที่ 5.2.1(a) ถึงรูปที่ 5.2.5(a) แสดงภาพผลลัพธ์จากการส่งภาพ "KRIT" แบบโปรเกรสซีพและภาพผลต่างของการส่งตั้งแต่ครั้งที่ 1 ถึง 5 ตามลำดับ ส่วนรูปที่ 5.2.1(b) ถึง 5.2.5(b) เป็นภาพผลต่างระหว่างภาพผลลัพธ์กับภาพต้นแบบในการส่งแต่ละครั้ง ในทำนองเดียวกันตั้งแต่รูปที่ 5.3.1(a) ถึงรูปที่ 5.3.5(a) แสดงภาพผลลัพธ์จากการส่งภาพ "BABOON" แบบโปรเกรสซีพและภาพผลต่างของการส่งตั้งแต่ครั้งที่ 1 ถึง 5 ตามลำดับ ส่วนรูปที่ 5.3.1(b) ถึง 5.3.5(b) เป็นภาพผลต่างระหว่างภาพผลลัพธ์กับภาพต้นแบบในการส่งแต่ละครั้ง

ขนาดพื้นที่	ภาพ 'Krit'		ภาพ 'Baboon'	
	จำนวนที่ใช้	อัตราบิท	จำนวนที่ใช้	อัตราบิท
16×16	24	0.05	0	0.00
8×8	47	0.33	46	0.28
8×4	17	0.81	27	0.56
4×8	52	1.00	139	0.66
4×4	316	1.94	511	1.81

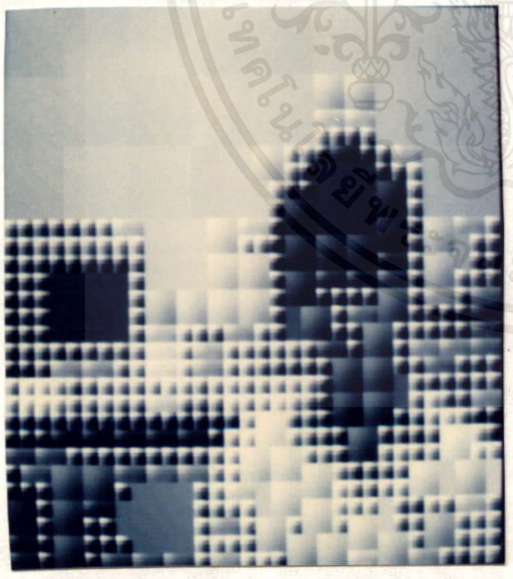
ตารางที่ 1 แสดงขนาดของพื้นที่ที่แบ่งได้และอัตราบิทสำหรับความละเอียดในแต่ละพื้นที่

การส่งครั้งที่	ภาพ 'Krit'		ภาพ 'Baboon'	
	อัตราบิท	PSNR(dB)	อัตราบิท	PSNR(dB)
1	0.23	15.78	0.35	18.06
2	0.49	22.93	0.67	22.33
3	0.68	25.69	0.92	24.24
4	0.79	26.96	1.09	24.99
5	0.98	31.37	1.33	25.99

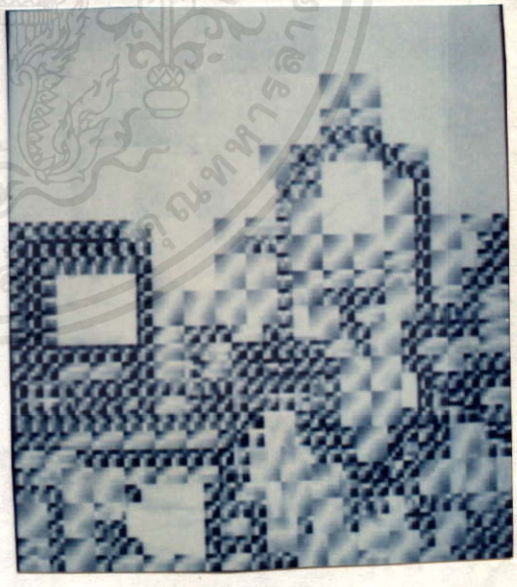
ตารางที่ 2 เป็นผลการลดข้อมูลและอัตราการสูญเสียที่เกิดขึ้นจากการเข้ารหัสในการส่งแต่ละครั้ง

รูปที่ 5.1.1 แสดงการแบ่งภาพ'KRIT'

รูปที่ 5.1.2 แสดงการแบ่งภาพ'BABOON'

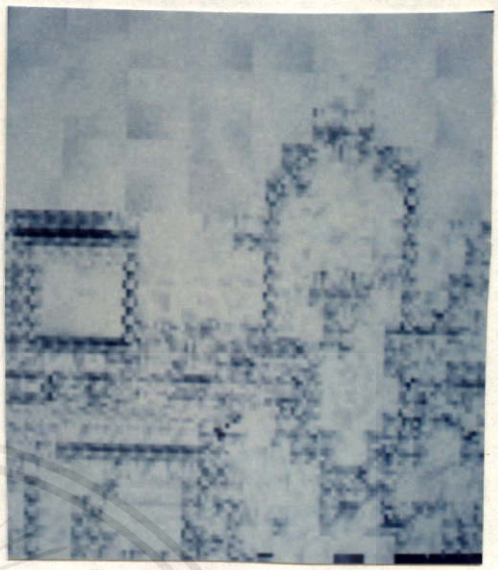


รูปที่ 5.2.1(a) ผลการส่งภาพครั้งที่ 1

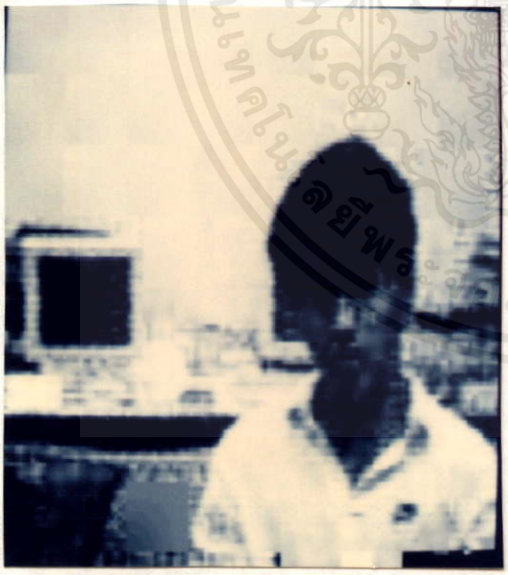


รูปที่ 5.2.1(b) ภาพผลต่างจากการส่งครั้งที่ 1

รูปที่ 5.2.2(a) ผลการส่งภาพครั้งที่ 2



รูปที่ 5.2.2(b) ภาพผลต่างจากการส่งครั้งที่ 2



รูปที่ 5.2.3(a) ผลการส่งภาพครั้งที่ 3

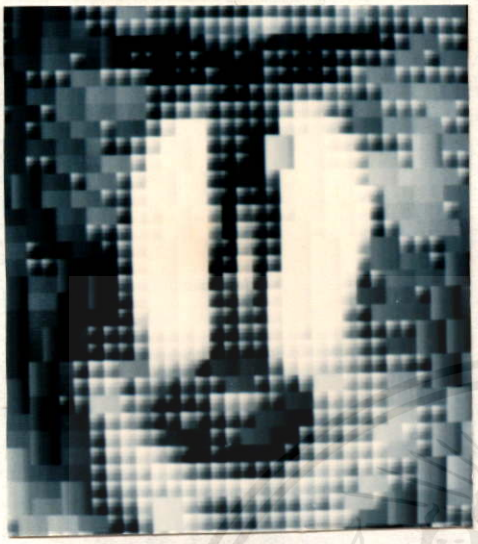
รูปที่ 5.2.3(b) ภาพผลต่างจากการส่งครั้งที่ 3

รูปที่ 5.2.4(a) ผลการส่งภาพครั้งที่ 4

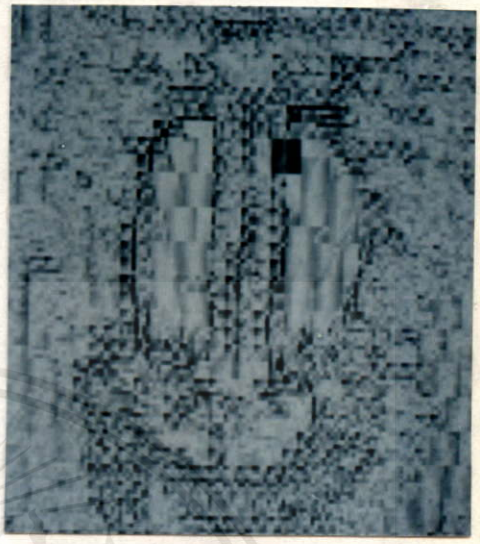
รูปที่ 5.2.4(b) ภาพผลต่างจากการส่งครั้งที่ 4

รูปที่ 5.2.5(a) ผลการส่งภาพครั้งที่ 5

รูปที่ 5.2.5(b) ภาพผลต่างจากการส่งครั้งที่ 5



รูปที่ 5.3.1(a) ผลการส่งภาพครั้งที่ 1



รูปที่ 5.3.1(b) ภาพผลต่างจากการส่งครั้งที่ 1

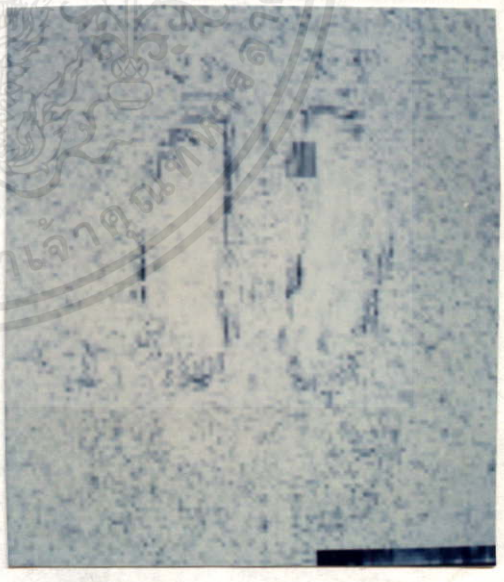
รูปที่ 5.3.2(a) ผลการส่งภาพครั้งที่ 2

รูปที่ 5.3.2(b) ภาพผลต่างจากการส่งครั้งที่ 2

รูปที่ 5.3.3(a) ผลการส่งภาพครั้งที่ 3

รูปที่ 5.3.3(b) ภาพผลต่างจากการส่งครั้งที่ 3

รูปที่ 5.3.4(a) ผลการส่งภาพครั้งที่ 4



รูปที่ 5.3.4(b) ภาพผลต่างจากการส่งครั้งที่ 4

รูปที่ 5.3.5(a) ผลการส่งภาพครั้งที่ 5

รูปที่ 5.3.5(b) ภาพผลต่างจากการส่งครั้งที่ 5

บทที่ 6

บทสรุป

6.1 สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้ได้ทำการวิจัยเกี่ยวกับการส่งภาพแบบโปรเกรสซีพ เพื่อหาวิธีการที่จะลดเวลาในการตัดสินใจเกี่ยวกับการสื่อสารภาพของผู้ใช้ให้เหลือน้อยที่สุด จากลักษณะเด่นของการส่งภาพแบบโปรเกรสซีพที่พยายามจัดส่งข้อมูลเป็นส่วนๆไปให้ผู้ใช้ตามลำดับความสำคัญในภาพทำให้ผู้ใช้สามารถตอบสนองได้อย่างรวดเร็ว การที่จะเพิ่มประสิทธิภาพของการส่งภาพแบบโปรเกรสซีพซึ่งจะช่วยให้ผู้ใช้สามารถตัดสินใจได้เร็วยิ่งขึ้น จึงขึ้นอยู่กับวิธีการจัดเรียงลำดับความสำคัญในภาพสำหรับการสื่อสารแต่ละครั้งนั่นเอง

ในขั้นตอนการทำงานมีการพัฒนาวิธีการลดข้อมูลภาพและวิธีการจัดเรียงลำดับความสำคัญในภาพขึ้นมาใหม่ โดยในส่วนของการลดข้อมูลภาพนี้ได้มีการประยุกต์วิธีการเซกเมนต์ภาพโดยอาศัยพื้นที่เป็นหลักมาใช้ร่วมกับการเข้ารหัสของโซนเพื่อให้ระบบมีความอ่อนตัวมากขึ้น ส่วนในการจัดเรียงลำดับความสำคัญในภาพได้ทำการแบ่งพื้นที่ออกเป็นโซนตามลักษณะของ Isotropic Covariance Model และได้จัดระดับความละเอียดให้กับพื้นที่ขนาดเล็กมากเป็นพิเศษ เพื่อให้สามารถจัดเก็บข่าวสารส่วนที่มีรายละเอียดสำคัญเอาไว้มากที่สุดในการส่งแต่ละครั้ง

จากการทดลองพบว่าภาพผลลัพธ์ในการส่งครั้งแรกๆ สามารถบอกลักษณะที่ทำให้ผู้ใช้เข้าใจได้ดียิ่งขึ้น แม้ว่าจะใช้จำนวนบิทต่ำ สำหรับภาพ"KRIT"ซึ่งเป็นภาพที่มีรายละเอียดน้อย ในการส่งครั้งที่ 1 ผู้ใช้จะเห็นภาพลักษณะเป็นขอบไม่ต่อเนื่องกันคือ บริเวณส่วนที่เป็นขอบภาพจะมีลักษณะเป็นบล็อกย่อยๆจำนวนมาก แต่ผู้ใช้สามารถที่จะคาดเดาลักษณะโดยรวมของภาพได้จากลักษณะไม่ต่อเนื่องกันนี้ ในการส่งครั้งที่ 2 และ 3 ภาพผลลัพธ์จะมีรายละเอียดชัดเจนมากขึ้นและมีลักษณะขอบลดลง ผู้ใช้ที่มีความคุ้นเคยกับภาพมาก่อนอาจเข้าใจภาพได้ตั้งแต่การส่งครั้งที่ 3 และในการส่งครั้งที่ 4 และ 5 ภาพผลลัพธ์จะมีลักษณะไม่ต่างจากภาพต้นแบบมากนัก อย่างไรก็ตามภาพผลลัพธ์ในครั้งสุดท้ายก็ยังมีความสูญเสียอยู่ในบางส่วนโดยเฉพาะบริเวณนัยน์ตา เมื่อเทียบกับภาพผลลัพธ์จากการส่งตามแบบเก่าแล้วเห็นได้ชัดเจนว่า ตามวิธีใหม่นี้มีการปรับปรุงรายละเอียดของการส่งครั้งแรกๆเป็นอย่างมาก นอกจากนี้ในการส่งแต่ละครั้งก็ใช้ข้อมูลขนาดเล็กลงด้วย สำหรับภาพ"BABOON"ซึ่งจัดเป็นภาพที่มีรายละเอียดมาก ทำให้ขั้นตอนการเซกเมนต์ภาพไม่สามารถจัดพื้นที่ให้มีขนาดใหญ่

กว่า 8×8 ได้ ดังนั้นเมื่อเทียบกับภาพ"KRIT"แล้ว ในการส่งแต่ละครั้งจึงต้องใช้จำนวนข้อมูลมากกว่า อย่างไรก็ตามภาพผลลัพธ์จากการส่งครั้งแรกก็มีลักษณะเป็นขอบเช่นเดียวกัน ส่วนในการส่งครั้งหลังลักษณะขอบที่ไม่ต่อเนื่องกันจะลดลงอย่างรวดเร็วและภาพในการส่งครั้งหลังมีความคมชัดเพิ่มขึ้นเรื่อยๆทำให้สามารถนำไปใช้ในการส่งข้อมูลภาพได้มีประสิทธิภาพสูง

6.2 ปัญหาที่เกิดขึ้นและข้อเสนอแนะ

เนื่องจากข้อมูลภาพเป็นข้อมูลที่มีความละเอียดสูง จึงต้องใช้เนื้อที่ขนาดใหญ่ในการจัดเก็บ ทำให้เป็นปัญหาในด้านการประมวลผลทั้งในเรื่องของเวลาในการทำงานและเรื่องความซับซ้อนในส่วนของหน่วยความจำของคอมพิวเตอร์ สำหรับในส่วนของวิธีการทำงานของระบบนั้นยังมีปัญหาในเรื่องการกำหนดรูปแบบของโซน เนื่องจากยังไม่มีการศึกษาผลของโซนอย่างจริงจัง ในส่วนของการวิจัยจึงได้เน้นวิธีลองผิดลองถูกเป็นหลัก นอกจากนี้ในการทำงานยังมีปัญหาในการแปลง เพราะพื้นที่ที่ได้จัดแบ่งสำหรับการแปลงนี้ถูกกำหนดให้มีลักษณะอ่อนตัวจึงอาจมีลักษณะไม่เป็นรูปสี่เหลี่ยมจตุรัส ดังนั้นในการทำงานจริงจึงต้องทำการแปลงทีละมิติโดยใช้ขนาดของ Basis Vector ซึ่งอาจไม่เท่ากัน

การวิจัยนี้ได้เน้นไปที่การออกแบบระบบการเข้ารหัสและการจัดเก็บข้อมูล โดยไม่ได้ทำการศึกษาในส่วนการสื่อสารเลย ดังนั้นในการทดลองจึงใช้วิธีการประมวลผลจากส่วนเข้ารหัสแล้วเก็บข้อมูลลงในไฟล์ จากนั้นจะนำข้อมูลในไฟล์นี้มาประมวลผลในส่วนถอดรหัสเพื่อที่จะแสดงผลต่อไป ในส่วนของการพัฒนาระบบต่อไปสามารถทำได้ในจุดของการควอนไทซ์โดยนำ Vector Quantization มาใช้แทนวิธีการควอนไทซ์แบบเดิม

Reference

[1] R.W.Hamming, "Coding and Information Theory", Prentice-Hall, New Jersey, 1986

[2] K.Thomas, "Entropy", Dr.Dobb's Journal, Feb 1991, pp.32-34

[3] A.N.Netravali and B.G.Haskell, "Digital Pictures, Representation and Compression", Plenum, New York, 1988

[4] A.K.Jain, "Image Data Compression, a review", Proc.IEEE, vol.69, No.3, Mar 1981, pp.349-389

[5] A.N.Netravali and J.O.Limb, "Picture Coding, a review", Proc.IEEE, vol.68, No.3, Mar 1980, pp.366-405

[6] M.Rabbani and P.W.Jones, "Digital Image Compression Techniques", SPIE, Optical Engineering Press, vol.TT-7, WA, 1991

[7] T.J.Lynch, "Data Compression: Techniques and Applications", Van Nostrand Reinhold, New York, 1985

[8] G.Held and T.R.Marshall, "Data Compression", John Wileys & Sons, New York, 1991

[9] M.R.Nelson, "Arithmatic Coding and Statistical Modeling", Dr.Dobb's Journal, Feb 1991, pp.16-29

[10] R.J.Clarke, "Transform Coding of Images", Academic Press, London, 1985

[11] L.D.Davisson, "Rate-Distortion Theory and Application", Proc.IEEE, vol.60, No.7, Jul 1972, pp.800-808

[12] Z.L.Budrikis, "Visual Fidelity Criterion and Modeling", Proc.IEEE, vol.60, Jul 1972, pp.771-779

[13] N.Ahmed and K.R.Rao, "Orthogonal Transform for Digital Signal Processing", Springer-Verlag, New York, 1975

[14] H.C.Andrews, "Two-Dimensional Transform", in Picture Processing and Digital Filtering, T.S.Huang, Ed., Springer-Verlag, New York, 1975, pp.21-68

[15] K.G.Beauchamp, "Transforms for Engineers: A guide to signal processing" Oxford University Press, Oxford, 1987

[16] N.Ahmed, T.Natarajan and K.R.Rao, "Discrete Cosine Transform", IEEE Trans.on Comput., Jan 1974, pp.90-93

[17] K.R.Rao and P.Yip, "Discrete Cosine Transform: Algorithms, advantages and applications", Academic Press, New York, 1990

[18] T.Natarajan and N.Ahmed, "Performance Evaluation for Transform Coding using a Nonseparable Covariance Model", IEEE Trans.on Commun., vol.COM-26, Feb 1978, pp.310-312

[19] P.Yip and K.R.Rao, "Energy Packing Efficiency for the Generalized Discrete Transforms", IEEE Trans.on Commun., vol.COM-26, Aug 1978, pp.1257-1262

[20] S.P.Lloyd, "Least Squares Quantization in PCM", IEEE Trans.on Inform.Theory, vol.IT-28, Mar 1982, pp.129-137

[21] J.Max, 'Quantizing for Minimum Distortion', IRE Trans. Inf. Theory, vol.IT-6, Mar 1960, pp.7-12

[22] A.K.Jain, "Advances in Mathematical Models for Image Processing", Proc.IEEE, vol.69, May 1981, pp.502-508

[23] R.C.Reininger and J.D.Gibson, "Distribution of the Two-Dimentional DCT coefficients for Images", IEEE Trans.on Commun., vol.COM-31, No.6, Jun 1983, pp.835-839

[24] K.N.Ngan, "Image Display Techniques using the Cosine Transform", IEEE Trans.on ASSP., vol.ASSP-32, No.1, Feb 1984, pp.159-166

[25] J.B.O'Neal,Jr. and T.R.Natarajan, "Coding Isotropic Images", IEEE Trans.on Inform.Theory, vol.IT-23, No.6, Nov.1977, pp.697-707

[26] S.Yuan and K.B.Yu, "Zonal Sampling and Bit Allocation of HT coefficients in Image Data Compression", IEEE Trans.on Commun., vol.COM-34, No.12, Dec 1986, pp.1246-1251

[27] E.L.Hall, "Computer Image Processing and Recognition", Academic Press, New York, 1979

[28] M.Tasto and P.A.Wintz, "Image Coding by Adaptive Block Quantization", IEEE Trans.on Commun.Tech., vol.COM-19, Dec 1971, pp.957-972

[29] W.H.Chen and W.K.Pratt, "Scene Adaptive Coder", IEEE Trans.on Commun., vol.COM-32, No.3, Mar 1984, pp.225-232

[30] W.H.Chen and C.H.Smith, "Adaptive Coding of Monochrome and Color Images", IEEE Trans.on Commun., vol.COM-25, Nov 1977, pp.1285-1292

[31] A.Habibi, "Survey of Adaptive Image Coding Technique", IEEE Trans.on Commun., vol.COM-25, No.11, Nov.1977, pp.1275-1284

[32] M.Kunt, A.Ikonomopoulis and M.Kocher, "Second Generation Image Coding Techniques" Proc.IEEE, vol.73, pp.549-574

[33] P.J.Burt and E.H.Adelson, "The Laplacian Pyramid as a Compact Image Code", IEEE Trans.on Commun., vol.COM-31, No.4, Apr 1983, pp.532-540

[34] K.Panusopone, K.Sarika and F.Cheevasuvit, "Variable Block-size Image Coding using Region Growing", in Proc.ICSPAT, Oct 1993, pp.702-708

[35] กฤษณ์ ภาณุโสภณ, มงคล จันทร์ชูเกียรติ และฟูศักดิ์ ชีวสุวิทย์, "การเข้ารหัสภาพโดยใช้บล็อกขนาดไม่คงที่" การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 16, DSP-003, 25-26 พฤศจิกายน 2536

[36] K.R.Sloan and S.L.Tanimoto, "Progressive Refinement of Raster Images", IEEE Trans.on Comput., vol.C-28, No.11, Nov 1979, pp.871-874

[37] S.E.Elnahas, K.H.Tzou, J.R.Cox,Jr. and R.L.Hill, "Progressive Coding and Transmission of Digital Diagnostic Pictures", IEEE Trans.on Medical Imaging, vol.MI-5, No.2, Jun 1986, pp.73-83

[38] L.Wang and M.Goldberg, "Progressive Transmission by Transform Coefficient Residual Error Quantization", IEEE Trans.on Commun., vol.COM-36, No.1, Jan 1988, pp.75-87

[39] กฤษณ์ ภาณุโสภณ และฟูศักดิ์ ชีวสุวิทย์, "การส่งภาพที่ถูกลดข้อมูลด้วยวิธีโปรเกรสซีพแบบการเข้ารหัสของโซน", การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 15, J-I, 3-4 ธันวาคม 2535

[40] Y.Huang, H.M.Dreizen and N.P.Galatsanos, "Prioritized DCT for Compression and Progressive Transmission of Images", IEEE Trans.on Image Proc., vol.1, No.4, Oct 1992, pp.477-487

[41] A.N.Netravali and E.G.Bowen, "A Picture Browsing System", IEEE Trans.on Commun., vol.COM-29, No.12, Dec 1981, pp.1969-1976

[42] T.Pavlidis, "Structural Pattern Recognition", Springer-Verlag, New York, 1977

ประวัติผู้เขียน

นายกฤษณ์ ภาณุโสภณ เกิดวันที่ 22 สิงหาคม พ.ศ. 2515 ที่กรุงเทพมหานคร จบการศึกษาระดับปริญญาตรีในหลักสูตรวิศวกรรมศาสตรบัณฑิตจากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังเมื่อปีพ.ศ. 2535 ปัจจุบันทำงานในตำแหน่งวิศวกรประจำโรงงานคาปาซิเตอร์ บริษัทเนชั่นแนลไทยจำกัด

ผลงานวิจัยที่ได้รับการตีพิมพ์

[1] กฤษณ์ ภาณุโสภณ และฟูศักดิ์ ชีวสุวิทย์, "การส่งภาพที่ถูกลดข้อมูลด้วยวิธีโปรเกรสซีพแบบการเข้ารหัสของโซน", การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 15, J-1, 3-4 ธันวาคม 2535

[2] K.Panusopone, K.Sarika and F.Cheevasuvit, "Variable Block-size Image Coding using Region Growing", in Proc.ICSPAT, Oct 1993, pp.702-708

[3] กฤษณ์ ภาณุโสภณ, มงคล จันทร์ฐเกียรติ และฟูศักดิ์ ชีวสุวิทย์, "การเข้ารหัสภาพโดยใช้บล็อกขนาดไม่คงที่" การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 16, DSP-003, 25-26 พฤศจิกายน 2536

ภาคผนวก ก. ข่าวสารและ Entropy

คำต่อไปนี้ "ความไม่แน่นอน","ความแปลกใจ"และ"ข่าวสาร" มีความเกี่ยวข้องกันบ้าง ก่อนเกิดเหตุการณ์หรือก่อนได้รับข่าวสารจะมีความไม่แน่นอนอยู่ เมื่อเหตุการณ์ปรากฏขึ้นจะเกิดความแปลกใจ และหลังจากเกิดเหตุการณ์แล้วก็จะได้รับข่าวสาร ปริมาณของค่าทั้งสามนี้จะเท่ากัน

ต่อไปจะมาพิจารณาฐานของlogที่นำมาใช้แสดง $I(p)$ เนื่องจากทุกฐานของlogมีความสัมพันธ์กันดังนี้

$$\log_a x = \frac{\log_b x}{\log_b a} = (\log_a b)(\log_b x) \qquad (1)$$

ดังนั้นเพื่อความสะดวกจึงกำหนดให้ใช้ฐานสำหรับ $I(p)$ เป็นเลขฐาน2 และค่าของ$I(p)$จะมีค่าเป็นบิท อย่างไรก็ตามบิทมี2ความหมาย คือเป็นหลักของเลขฐาน2 และเป็นหน่วยของข่าวสาร จึงต้องระวังไม่ให้เกิดความสับสน

สมมุติว่ามีข้อมูลตัวอักษรqสัญลักษณ์ $S_1, S_2, \ldots, S_q$ โดยที่แต่ละสัญลักษณ์จะมีความน่าจะเป็น $p(s_1) = p_1, p(s_2) = p_2, \ldots, p(s_q) = p_q$ เมื่อได้รับสัญลักษณ์เหล่านี้สักอย่างหนึ่งแล้วจะทราบข่าวสารได้มากเท่าใดจากสัญลักษณ์นี้ ตัวอย่างเช่นในกรณีที่ $p_1 = 1$ (และแน่นอนว่าความน่าจะเป็นของสัญลักษณ์ตัวอื่นจะเป็นศูนย์) นั่นคือมันจะไม่มีความแปลกประหลาด (หรือไม่มีข่าวสาร) เพราะสามารถคาดหมายข่าวสารที่จะเกิดขึ้นได้ แต่ในอีกทางหนึ่งถ้าความน่าจะเป็นของแต่ละสัญลักษณ์แตกต่างกันทั้งหมดแล้วเมื่อได้รับสัญลักษณ์ที่มีความน่าจะเป็นต่ำจะรู้สึกแปลกใจและได้รับข่าวสารมากกว่าเมื่อได้รับสัญลักษณ์ที่มีความน่าจะเป็นสูง โดยความคิดนี้ข่าวสารน่าจะเกี่ยวข้องในทางตรงข้ามกับความน่าจะเป็นของสัญลักษณ์

ถ้าต้องการสร้างฟังก์ชั่น $I(p)$ เพื่อใช้วัดปริมาณข่าวสารหรือความแปลกประหลาดในการเกิดเหตุการณ์ที่มีความน่าจะเป็น p ต้องคาดหวังให้ $I(p)$ เป็นดังนี้

1. $I(p) \geq 0$ เป็นการวัดสิ่งที่เป็นจำนวนจริงบวก

2. $I(p_1 p_2) = I(p_1) + I(p_2)$ สำหรับเหตุการณ์ที่ไม่ขึ้นแก่กัน

3. $I(p)$ เป็นฟังก์ชั่นที่ต่อเนื่องของp

จากเงื่อนไขที่สอง ทำให้สามารถหาข่าวสารของเหตุการณ์พิเศษ2อย่าง โดยที่ทั้งสองเหตุการณ์มีความน่าจะเป็นเท่ากันแล้ว ข่าวสารจากเหตุการณ์ทั้งสองคือ

$$I(p^2) = I(p) + I(p) = 2I(p) \qquad (2)$$

หรือในกรณีทั่วไป

$$I(p^n) = nI(p) \tag{3}$$

ถ้าเขียนในรูปซับซ้อนยิ่งขึ้น

$$\begin{aligned} p^n &= y, p = y^{1/n} \\ I(y) &= nI(y^{1/n}) \end{aligned} \tag{4}$$

ในทำนองเดียวกัน

$$I(y^{m/n}) = \frac{m}{n} I(y) \tag{5}$$

จากสมการข้างต้นแสดงให้เห็นว่าฟังก์ชั่น $I(p)$ มีคุณสมบัติเช่นเดียวกับฟังก์ชั่นlog ดังนั้นอาจเขียนฟังก์ชั่น $I(p)$ ใหม่เป็น

$$I(p) = k \log_b p \tag{6}$$

โดยต้องเลือกค่าถ่วงน้ำหนัก k และฐาน b ให้เหมาะสม เมื่อพิจารณาคุณสมบัติข้อที่ 1 ของฟังก์ชั่น $I(p)$ ซึ่งค่าpจะอยู่ในช่วง 0ถึง1 จึงจำเป็นต้องใช้ค่า k เป็นลบ ในที่นี้จะลองที่ -1

$$I(p) = -\log_b p = \log_b \frac{1}{p} \tag{7}$$

Entropy

ถ้าได้รับสัญลักษณ์ S_i ที่มีข่าวสารเท่ากับ $I(S_i)$ แล้ว ในการรับข้อมูลทั้งหมดจะได้รับข่าวสารเท่าใด ค่าข่าวสารของข้อมูลทั้งหมดอาจหาได้จากค่าเฉลี่ย โดยนำความน่าจะเป็นในการเกิดสัญลักษณ์ต่างๆ p_i มาร่วมพิจารณาด้วยหรือ

$$p_i I(s_i) = p_i \log_2 \frac{1}{p_i} \tag{8}$$

และสำหรับสัญลักษณ์ทั้งหมด

$$\sum_{i=1}^{q} p_i \log_2 \frac{1}{p_i} \tag{9}$$

ค่าข่าวสารของข้อมูลทั้งหมด $H_2(S)$ ถูกเรียกว่า Entropy ของระบบ S ที่ใช้สัญลักษณ์ S_i และมีความน่าจะเป็น p_i และเขียนในรูปมาตรฐาน (Radix r) เป็น

$$H_r(S) = H_2(S)\log_r 2 = \sum_{i=1}^{q} p_i \log_r (\frac{1}{p_i}) \tag{10}$$

ถ้าพิจารณาจากการกระจาย $p = (p_1, p_2, ..., p_q)$ ของสัญลักษณ์แต่ละตัว S_i แล้ว จะมีเลขตัวหนึ่งที่เป็น Entropy $H(S)$ ซึ่งเทียบได้กับค่าเฉลี่ยของการกระจายซึ่งอาจใช้สรุปลักษณะของการกระจายได้ Entropy $H(S)$ จะเป็นข่าวสารเฉลี่ยของสัญลักษณ์ S

ในการหาฟังก์ชั่น Entropy อีกวิธีหนึ่ง ถ้าข้อมูลที่ประกอบด้วย N สัญลักษณ์ จะสามารถคาดหวัง N_{p_1} ของสัญลักษณ์ตัวแรก, N_{p_2} ของตัวที่สอง เป็นเช่นนี้ไปเรื่อยๆ ความน่าจะเป็นPของข้อมูล N มีค่าเท่ากับ

$$\begin{aligned} p &= p_1^{(Np_1)} p_2^{(Np_2)} \cdots p_q^{(Np_q)} \\ &= [p_1^{(p_1)} p_2^{(p_2)} \cdots p_q^{(p_q)}]^N \end{aligned} \tag{11}$$

ดังนั้นจะมีข่าวสารเท่ากับ

$$\log(\frac{1}{p}) = N \sum_{i=1}^{q} p_i \log(\frac{1}{p_i}) \tag{12}$$

และข่าวสารของแต่ละตัวอักษรเป็น

$$H(S) = \sum (p_i \log \frac{1}{p_i}) \tag{13}$$

การพิจารณา Entropy ต้องระลึกอยู่เสมอว่า Entropy ของข้อมูลอาจไม่มีความหมายจนกว่าจะนำรูปแบบของข่าวสารนั้นมาร่วมพิจารณาด้วย

```
/*Encoder Part*/
#include  <dos.h>
#include  <alloc.h>
#include  <stdio.h>
#include  <math.h>
#define   sqr(A)    ((A)*(A))

void    AlloFai();
float  *fl1D(int  size);
float  **fl2D(int  size);
float  *float1D(int  ttl_base);
char huge *ch1D(int  siz_base);
char huge **ch2D(int  siz_base);
char huge ***ch3D(int  ttl_base,int  siz_base);
float huge *Mark1D(int  hor);
float huge **Mark2D(int  ver, int  hor);
float huge ***Mark3D(int size);
int    *int1D(int  siz_base);
int    **int2D(int  siz_base);
void    OpenFileFai();
void    ReadFileImage(int  ttl_base, int  siz_base);
void    Activ(int  ttl_base, int  siz_base);
void    Cluster(float  threshold, int  ttl_base);
float  COS(int  i, int  j, float  k);
void    CalTable();
void    DCT1D(int  x_pos, int  y_pos);
void    DCT2D();
void    Allocbit(float far ***Var);
void    WriteImage(char huge ***Image,int
ttl_base);
void    SetImage(float  threshold, int  ttl_base);
void    Quantize(int ttl_base);
void    Pack(int ttl_base);


char huge   ***Image;
float               *energy;
char                     *blk_stat;
int                 SizeImage;
int                 siz_base;
int                 sum_blk;
int                 index;
float               Array[16],Vector[16],Normal[5];
float far           ***Table;
float far           ***Coeff;
char                 ***Alloc;
float               threshold;


int          step[9] = {0,1,2,4,8,16,32,64,128};

int          mask[5][16][16] =
{{{1,1,2,2,3,3,3,3,4,4,4,4,5,5,5,5},

        {1,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5},

        {2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5},

        {2,2,3,3,3,3,3,3,4,4,4,4,5,5,5,5},

        {3,3,3,3,3,3,3,3,4,4,4,4,5,5,5,5},

        {3,3,3,3,3,3,4,4,4,4,4,4,5,5,5,5},

        {3,3,3,3,3,4,4,4,4,4,4,5,5,5,5,5},

        {3,3,3,3,4,4,4,4,4,4,5,5,5,5,5,5},

        {4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5},

        {4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5},

        {4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5},

        {4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5},

        {4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5},

        {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5},

        {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5},

        {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5},

        {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5}},

        {{1,2,2,3,3,4,4,5,0,0,0,0,0,0,0,0},

        {2,2,2,3,3,4,4,5,0,0,0,0,0,0,0,0},

        {2,2,3,3,3,4,4,5,0,0,0,0,0,0,0,0},

        {3,3,3,3,4,4,5,5,0,0,0,0,0,0,0,0},

        {3,3,3,4,4,5,5,5,0,0,0,0,0,0,0,0},

        {4,4,4,4,5,5,5,5,0,0,0,0,0,0,0,0},

        {4,4,4,5,5,5,5,5,0,0,0,0,0,0,0,0},

        {5,5,5,5,5,5,5,5,0,0,0,0,0,0,0,0},

        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}},
```

```
{{1,2,2,3,3,4,4,5,0,0,0,0,0,0,0,0},
{2,2,3,3,4,4,5,5,0,0,0,0,0,0,0,0},
{3,3,3,4,4,5,5,5,0,0,0,0,0,0,0,0},
{4,4,4,4,5,5,5,5,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}},
{{1,2,3,4,0,0,0,0,0,0,0,0,0,0,0,0},
{2,2,3,4,0,0,0,0,0,0,0,0,0,0,0,0},
{2,3,3,4,0,0,0,0,0,0,0,0,0,0,0,0},
{3,3,4,4,0,0,0,0,0,0,0,0,0,0,0,0},
{3,4,4,5,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,5,5,0,0,0,0,0,0,0,0,0,0,0,0},
{4,5,5,5,0,0,0,0,0,0,0,0,0,0,0,0},
{5,5,5,5,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}},
{{1,2,3,4,0,0,0,0,0,0,0,0,0,0,0,0},
{2,2,3,4,0,0,0,0,0,0,0,0,0,0,0,0},
{3,3,4,5,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,5,5,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}}};
static float qx1[1]   ={0.000000};
static float qx2[2]   ={0.000000, 0.981599};
static float qx3[4]   ={0.000000, 0.500549,
1.049957, 1.747927};
static float qx4[8]   ={0.000000, 0.258221,
0.522403, 0.799549, 1.099285, 1.437138,
1.843531, 2.400803};
static float qx5[16]  ={0.000000, 0.131971,
0.264715, 0.399039, 0.535817, 0.676035,
0.820850, 0.971674,
                      1.130294, 1.299072,
1.481284, 1.681731, 1.907981, 2.173234,
2.504430, 2.975926};
static float qx6[32]  ={0.000000, 0.066844,
0.133788, 0.200932, 0.268380, 0.336238,
0.404616, 0.473632,
                      0.543408, 0.614079,
0.685789, 0.758695, 0.832972, 0.908815,
0.986445, 1.066112,
```

```
                    1.148104, 1.232757,
1.320467, 1.411708, 1.507053, 1.607210,
1.713065, 1.825759,
                    1.946793, 2.078210,
2.222895, 2.385142, 2.571788, 2.794839,
3.078920, 3.492265};
static float qx7[64]  ={0.000000, 0.033660,
0.067332, 0.101029, 0.134765, 0.168553,
0.202404, 0.236333,
                    0.270353, 0.304478,
0.338721, 0.373097, 0.407621, 0.442308,
0.477172, 0.512232,
                    0.547502, 0.583002,
0.618747, 0.654759, 0.691056, 0.727658,
0.764589, 0.801871,
                    0.839529, 0.877587,
0.916074, 0.955019, 0.994453, 1.034410,
1.074925, 1.116037,
                    1.157788, 1.200223,
1.243391, 1.287346, 1.332147, 1.377857,
1.424549, 1.472299,
                    1.521197, 1.571338,
1.622833, 1.675805, 1.730394, 1.786759,
1.845085, 1.905583,
                    1.968503, 2.034136,
2.102832, 2.175010, 2.251183, 2.331987,
2.418223, 2.510926,
                    2.611463, 2.721700,
2.844280, 2.983146, 3.144589, 3.339681,
3.591183, 3.962314};
static float qx8[128] ={0.000000, 0.016893,
0.033787, 0.050685, 0.067587, 0.084496,
0.101413, 0.118340,
                    0.135278, 0.152229,
0.169194, 0.186176, 0.203175, 0.220195,
0.237235, 0.254299,
                    0.271387, 0.288502,
0.305645, 0.322818, 0.340022, 0.357261,
0.374534, 0.391845,
                    0.409196, 0.426587,
0.444022, 0.461502, 0.479029, 0.496605,
0.514232, 0.531913,
                    0.549649, 0.567443,
0.585298, 0.603215, 0.621196, 0.639245,
0.657363, 0.675554,
                    0.693819, 0.712162,
0.730585, 0.749092, 0.767684, 0.786365,
0.805138, 0.824006,
                    0.842972, 0.862040,
0.881213, 0.900495, 0.919889, 0.939398,
0.959028, 0.978782,
                    0.998664, 1.018679,
1.038830, 1.059123, 1.079562, 1.100153,
1.120901, 1.141811,
                    1.162888, 1.184139,
1.205570, 1.227187, 1.248997, 1.271007,
1.293224, 1.315656,
                    1.338311, 1.361197,
1.384324, 1.407700, 1.431335, 1.455240,
1.479425, 1.503902,
                    1.528684, 1.553782,
1.579211, 1.604985, 1.631120, 1.657631,
1.684536, 1.711855,
                    1.739606, 1.767811,
1.796493, 1.825676, 1.855387, 1.885655,
1.916509, 1.947985,
                    1.980117, 2.012946,
2.046514, 2.080870, 2.116064, 2.152155,
2.189206, 2.227287,
                    2.266477, 2.306863,
2.348545, 2.391633, 2.436256, 2.482558,
2.530707, 2.580897,
                    2.633354, 2.688348,
2.746197, 2.807286, 2.872088, 2.941187,
3.015322, 3.095448,
                    3.182829, 3.279188,
3.386976, 3.509848, 3.653649, 3.828680,
4.056155, 4.395057};

struct status
{
  unsigned int stat : 2;
};

void    AlloFai()
{
   printf("  Memory allocation is error.  \n");
   exit(1);
}

float  *fl1D(int  size)
{
  float  *a;

   if ( (a = (float *)calloc(size*4,4)) == NULL )
     AlloFai();
   return(a);
}

float  **fl2D(int  size)
{
  float  **b;
  int    i;

   if ( (b = (float **)malloc( size*sizeof(float*) ))
== NULL)
     AlloFai();
   for (i=0 ; i<size ; i++)
     b[i] = fl1D(size);
   return(b);
}

float  *float1D(int  ttl_base)
{
  float  *a;
```

```
    if ( (a = (float *)malloc(4*ttl_base)) == NULL)
      AlloFai();
    return(a);
}

char huge *ch1D(int  siz_base)
{
  char huge *a;

    if ( (a = (char huge*)farmalloc(siz_base)) ==
NULL)
      AlloFai();
    return(a);
}

char huge **ch2D(int  siz_base)
{
  char huge **b;
  int    i;

    if ( (b = (char huge**)farmalloc
(siz_base*sizeof(char huge*)) ) == NULL)
      AlloFai();
    for (i=0 ; i<siz_base ; i++)
      b[i] = ch1D(siz_base);
    return(b);
}

char huge ***ch3D(int  ttl_base,int  siz_base)
{
  char huge ***b;
  int    i;

    if ( (b = (char huge***)farmalloc
(ttl_base*sizeof(char huge**)) ) == NULL)
      AlloFai();
    for (i=0 ; i<ttl_base ; i++)
      b[i] = ch2D(siz_base);
    return(b);
}

int    *int1D(int  siz_base)
{
  int    *a;

    if (( a = (int *)farmalloc(siz_base*sizeof(int)) )
== NULL)
      AlloFai();
    return(a);
}

int    **int2D(int  siz_base)
{
  int    **b;
  int    i;

    if (( b = (int **)farmalloc (siz_base*sizeof(int*) )
) == NULL)
      AlloFai();
    for(i=0 ; i<siz_base ; i++)
      b[i] = int1D(siz_base);
    return(b);
}

float huge  *Mark1D(int  hor)
{
  float huge *a;

    if ((a = (float huge*)farcalloc(hor,4) ) == NULL)
      AlloFai();
    return(a);
}

float huge  **Mark2D(int  ver, int  hor)
{
  float huge **b;
  int    i;

    if ((b = (float huge**)farmalloc(ver*sizeof(float
huge*) ) ) == NULL)
      AlloFai();
    for (i=0 ; i<ver ; i++)
      b[i] = Mark1D(hor);
    return(b);
}

float huge  ***Mark3D(int size)
{
  float huge ***c;
  int    i;
    if ((c = (float huge***)farmalloc(size*sizeof(float
huge**) ) ) == NULL)
      AlloFai();
    for (i=0 ; i<5 ; i++)
      c[i] = Mark2D(16,16);
    return(c);
}

void   OpenFileFai()
{
    printf("Can't open file. \n");
    printf("Press any key to exit. \n");
    getch();
    exit(1);
}

void   ReadFileImage(int  ttl_base, int  siz_base)
{
  FILE     *fp;
  static   char     fname[] = "fname.dat";

  int   a,b,x,y,i,j;
  int   block,data,row,col,buffer;
```

```
    block  =  0;
    printf("  FILE NAME (IMAGE) ... ");
    scanf("%s",fname);
    fflush(stdin);
    printf("\n");
    if (( fp = fopen(fname,"rb") ) == NULL)
      OpenFileFai();

    Image              =    ch3D(ttl_base,siz_base);

    (int) SizeImage   =    sqrt((double)
ttl_base)*siz_base;
    for (a=0 ; a<SizeImage ; a++)
      for (b=0 ; b<SizeImage ; b++)
      {
            buffer  =  0;
            fread(&buffer,1,1,fp);
            row     =  a;
            col     =  b;
            data    =  siz_base-1;
            while(col>data || row>data)
            {
              if (col > data)
                col  -=  siz_base;
              if (row > data)
                row  -=  siz_base;
            }
            block   =  (sqrt((double)
ttl_base)*(a/siz_base))+(b/siz_base);

            Image[block][row][col]  =  buffer-
128;/*look*/
      }
      fclose(fp);

}

void  Activ(int  ttl_base, int  siz_base)
{
  int  Pixel_base;
  float  mean;
  register i,j,k;

    threshold   =  0.0;
    Pixel_base  =  sqr(siz_base);
    energy      =  float1D(ttl_base);
    for (i=0 ; i<ttl_base ; i++)
    {
      mean       =  0;
      energy[i]  =  0;

      for (j=0 ; j<siz_base ; j++)
          for (k=0 ; k<siz_base ; k++)
              mean   +=  Image[i][j][k]+127;

      mean /= Pixel_base;

      for (j=0 ; j<siz_base ; j++)
          for (k=0 ; k<siz_base ; k++)
            energy[i]  +=  sqr(Image[i][j][k]+127-
mean);

        threshold  +=  energy[i];
    }
    threshold  /=  ttl_base;
}

void  Cluster(float  threshold, int  ttl_base)
{
  int
row,X_blk,Y_blk,counter,count,count1,gap,limit1;
  register int  i,j,k;
  float  buffer;

  int  *data;

    sum_blk  =  0;
    if ((blk_stat = (char *)calloc(ttl_base,1)) ==
NULL)
      AlloFai();
    if ((data = (int*)calloc(ttl_base,2)) == NULL)
      AlloFai();
    row       =  SizeImage/siz_base;
    k  =  0;
    for (i=0 ; i<5 ; i++)
    {
        switch (i)
        {
          case 0 : X_blk = 4;
                   Y_blk = 4;
                   break;
          case 1 : X_blk = 2;
                   Y_blk = 2;
                   break;
          case 2 : X_blk = 2;/*type 3 x:4 y:8*/
                   Y_blk = 1;                  /*
1 -> 2 */
                   break;
          case 3 : X_blk = 1;/*type 2 x:8  y:4*/
                   Y_blk = 2;
                   for (j=0 ; j<(ttl_base-row) ;
j++)
                   {
                     buffer  = 0.0;            /*
1  */
                     buffer += energy[j];
/*  |  */
                     buffer += energy[j+row];
/*  v  */
                     if (buffer < threshold)   /*
3  */
                     {
                         sum_blk++;
                         blk_stat[j] = 0x01;
                         blk_stat[j+row] =
0x03;
```

```
                        energy[j] =
energy[j+row] = 1e7;
                    }
                }
                break;
        case 4 : X_blk = 1;
                Y_blk = 1;
                for (j=0 ; j<ttl_base ; j++)
                {
                  if (blk_stat[j] == 0)
                  {
                      sum_blk++;
                      blk_stat[j] = 0x01;
                      energy[j] = 1e7;
                  }
                }
                break;
    }
    gap  =  0;
    for (j=0 ; j<((row-X_blk+1)*(row-Y_blk+1))
; j++)
    {
      counter  =  0;
      buffer   =  0;
      k  =  j+gap;
      count  =  0;
      do
      {
        buffer +=  energy[k];
        data[counter]  =  k;
        if (blk_stat[k] > 0)
            break;
        blk_stat[k] = 0x03;
        count++;
        counter++;
        count1  =  1;
        limit1  =  1;
        k++;

        do
        {
            buffer += energy[k];
            if (buffer > threshold)
              limit1  =  0;
            data[counter]  =  k;
            if (blk_stat[k] > 0)
              break;
            blk_stat[k] = 0x02;
            k++;
            count1++;
            counter++;
        }
        while (count1<X_blk  && limit1 ==
1);

        k  +=  row-X_blk;
      }
      while (count<Y_blk && limit1 == 1);
      if (buffer < threshold)
      {
        sum_blk++;
        blk_stat[j+gap] = 0x01;
        for (buffer=0 ; buffer<counter ;
buffer++)
        {
            energy[data[buffer]] = 1e7;
        }
      }
      else
      {
        for (buffer=0 ; buffer<counter ;
buffer++)
            blk_stat[data[buffer]] = 0x00;
        }
        if (((k+X_blk)%row) == 0)
          gap  +=  X_blk-1;
      }
    }
    k  =  0;
    for (i=0 ; i<row ; i++)
    {
      for (j=0 ; j<row ; j++)
      {
        printf("%d",blk_stat[k]);
        k ++;
      }
      printf("\n");
    }
    printf("Total clustered blocks : %d\n",sum_blk);
}

float  COS(int  i, int  j, float  k)
{
  float  Pl;
  float  a;

    Pl  =  3.141592654;
    a   =  ((float)(i*Pl) * (float)(2.0*j + 1)  ) /
(float)(2.0*k);
  return(cos(a));
}

void   CalTable()
{
  int     i,j;

  for (i=0 ; i<4 ; i++)
    for (j=0 ; j<4 ; j++)
    {
      Table[4][i][j] = COS(i,j,4);
    }
  for (i=0 ; i<8 ; i++)
    for (j=0 ; j<8 ; j++)
    {
      Table[1][i][j] = COS(i,j,8);
```

```
    }
  for (i=0 ; i<16 ; i++)
    for (j=0 ; j<16 ; j++)
    {
        Table[0][i][j] = COS(i,j,16);
    }
}

void  DCT1D(int  x_pos, int  y_pos)
{
 int   x,y,z;

  switch (y_pos)
  {
    case 4  :  z = 4;   break;
    case 8  :  z = 1;   break;
    case 16 :  z = 0;   break;
  }

  if (x_pos < y_pos)
    x_pos = y_pos;
  if (x_pos > y_pos)
    y_pos = x_pos;

  for (x=0 ; x<y_pos ; x++)
    Vector[x] = 0.0;

  for (x=0 ; x<x_pos ; x++)
  {
    for (y=0 ; y<y_pos ; y++)
    {
        Vector[x] += Array[y]*Table[z][x][y];
    }
  }
}

void  DCT2D()
{
 int    m,n,u,v,row,sum;
 int
i,l,o,p,point2,point1,point,fix,pos,pos1,hor,ver,dumm
y[16];
 int    x_pos,y_pos,total,ttl0,ttl1,ttl2,ttl3,ttl4;
 float far  ***Var,scale1,scale2;
 FILE               *fp1;
 static  char       fname[] = "fname.dat";


  printf("  FILE NAME (BUFFER) ... ");
  scanf("%s",fname);
  fflush(stdin);
  printf("\n");
  if (( fp1 = fopen(fname,"wb") ) == NULL)
    OpenFileFail();


 Table  =  Mark3D(3);
 Coeff  =  Mark3D(5);
 Var    =  Mark3D(5);

 CalTable();
 ttl0   =  0;
 ttl1   =  0;
 ttl2   =  0;
 ttl3   =  0;
 ttl4   =  0;
 total  =  0;
 pos    =  -1;

 do
 {
   p      =  0;
   l      =  0;
   fix    =  0;
   ver    =  0;
   hor    =  0;
   row    =  SizeImage/siz_base;

   for (point=0 ; point<16 ; point++)
   {
       dummy[point] = 0;
   }

   do
   {
       pos++;
   }
   while (blk_stat[pos] == 2 || blk_stat[pos] ==
3);
   point  =  pos;

   do
   {
       dummy[p]  =  pos;
        p++;
       point1  =  point;
       hor = (fix == 1) ? 1 : 0;

       if (((fix == 1)&&(blk_stat[pos] == 3)) ||
(fix == 0))
       {
        do
        {
          hor++;
          pos++;
          dummy[p]  =  pos;
          p++;
          if ((pos-point) < row)
              point2  =  pos;
          fix  =  1;
        }
        while ((blk_stat[pos] == 2) && (pos <
sqr(row)));
        dummy[p-1]  =  0;
        p--;
       }
```

```
        else
        {
          pos  =  sqr(row);
        }

        l  +=  row;
        point1  +=  l;
        pos1 =  pos;
        pos  =  point1;
        ver++;
    }
    while ((blk_stat[pos] == 3) && (pos1 <
sqr(row)));

    hor--;
    pos  =  pos1;
    switch(ver)
    {
        case 4 :  index = 0;
                  break;
        case 2 :  index = (hor == 2) ? 1 : 2;
                  break;
        case 1 :  index = (hor == 1) ? 3 : 4;
                  break;
    }

    switch(index)
    {
        case 0 :  total++;
                  ttl0++;
                  x_pos   =  4;
                  y_pos   =  4;
                  break;
        case 1 :  total++;
                  ttl1++;
                  x_pos   =  2;
                  y_pos   =  2;
                  break;
        case 2 :  total++;
                  ttl2++;
                  x_pos   =  2;
                  y_pos   =  1;
                  break;
        case 3 :  total++;
                  ttl3++;
                  x_pos   =  1;
                  y_pos   =  2;
                  break;
        case 4 :  total++;
                  ttl4++;
                  x_pos   =  1;
                  y_pos   =  1;
                  hor  =  1;
                  break;
    }

    o   = 0;
    fix = 0;

    for (m=0 ; m<x_pos ; m++)
    {
        for (l=0 ; l<4 ; l++)
        {
         point = fix;
         for (n=0 ; n<y_pos ; n++)
         {
          for (p=0 ; p<4 ; p++)
            Array[(4*n)+p] =
(int)Image[dummy[point]][l][p]+127;
          point++;
         }
         DCT1D(4*x_pos,4*y_pos);
         for (n=0 ; n<(4*y_pos) ; n++)
           Coeff[index][o][n] = Vector[n];
         o++;
        }
        fix += (y_pos < x_pos)  ?  y_pos  :
x_pos;
    }

    for (m=0 ; m<(4*y_pos) ; m++)
    {
        for (n=0 ; n<(4*x_pos) ; n++)
          Array[n] = Coeff[index][n][m];
        DCT1D(4*y_pos,4*x_pos);
        for (n=0 ; n<(4*x_pos) ; n++)
        {
          scale1 = (n==0) ?  0.7071  :  1;
          scale2 = (m==0) ?  0.7071  :  1;
          Vector[n]          *=
((4*scale1*scale2)/(x_pos*y_pos));
          Coeff[index][n][m]   = Vector[n];
          Var[index][n][m]    += sqr(Vector[n]);
        }
    }
    for (m=0 ; m<(4*x_pos) ; m++)
        for (n=0 ; n<(4*y_pos) ; n++)
          fwrite(&Coeff[index][m][n],4,1,fp1);
    pos = point2-1; /*not sure +1?*/
}
while (total < sum_blk);
for (index=0 ; index<5 ; index++)
{
   switch(index)
   {
        case 0  :  fix = 256;  sum = ttl0;
                   break;
        case 1  :  fix = 64;   sum = ttl1;
                   break;
        case 2  :  fix = 32;   sum = ttl2;
                   break;
        case 3  :  fix = 32;   sum = ttl3;
                   break;
        case 4  :  fix = 16;   sum = ttl4;
                   break;
   }
   if (sum == 0)
```

```
            sum++;
        for (m=0 ; m<16 ; m++)
            for (n=0 ; n<16 ; n++)
              Var[index][m][n]  /=  ((sum*fix)/256.0);
    }
    fclose(fp1);

    printf("\n Type 0 has %d blocks\n",ttl0);
    printf("\n Type 1 has %d blocks\n",ttl1);
    printf("\n Type 2 has %d blocks\n",ttl2);
    printf("\n Type 3 has %d blocks\n",ttl3);
    printf("\n Type 4 has %d blocks\n",ttl4);
    Allocbit(Var);
}

void   Allocbit(float far ***Var)
{
  float  d,Rate;
  int    type,row,col,X_blk,Y_blk,i,j,total;

    Alloc  =  ch3D(5,16);
    for (type=0 ; type<5 ; type++)
    {
       Normal[type] = 0;
       switch (type)
       {
              case 0 : X_blk = 4;
                       Y_blk = 4;
                       Rate  =
0.1*16*X_blk*Y_blk; /*25.6*/
                       printf("\nEnter any key");
                       getch();
                       printf("\nType 0
allocation\n");
                       break;
              case 1 : X_blk = 2;
                       Y_blk = 2;
                       Rate  =
0.6*16*X_blk*Y_blk; /*38.4*/
                       printf("Enter any key");
                       getch();
                       printf("\nType 1
allocation\n");
                       break;
              case 2 : X_blk = 2;
                       Y_blk = 1;
                       Rate  =
1.0*16*X_blk*Y_blk; /*32*/
                       printf("Enter any key");
                       getch();
                       printf("\nType 2
allocation\n");
                       break;
              case 3 : X_blk = 1;
                       Y_blk = 2;
                       Rate  =
1.0*16*X_blk*Y_blk;
                       getch();
                       printf("\nType 3
allocation\n");
                       break;
              case 4 : X_blk = 1;
                       Y_blk = 1;
                       Rate  =
2.0*16*X_blk*Y_blk; /*32*/
                       getch();
                       printf("\nType 4
allocation\n");
                       break;
       }
       d = sqrt(Var[type][0][0]/100)/256;

       do
       {
            total  =  0;
            for (row=0 ; row<(4*X_blk) ; row++)
            {
               printf("\n");
               if (d == 0)
                 break;
               for (col=0 ; col<(4*Y_blk) ; col++)
               {
                  if (col == 0 && row == 0)
                  {
                     Alloc[type][row][col] = 8;
                     Var[type][row][col]  /= 100.0;
                  }
                  else
                     Alloc[type][row][col] = (int)(log(
sqrt(Var[type][row][col])

/ d ) / log(2));
                  if (Alloc[type][row][col] < 0)
                     Alloc[type][row][col] = 0;
                  if (Alloc[type][row][col] > 8)
                     Alloc[type][row][col] = 8;
                  total += Alloc[type][row][col];
                  if (Alloc[type][row][col] == 1)
                     if (Normal[type] <
sqrt(Var[type][row][col]))
                        Normal[type] =
sqrt(Var[type][row][col]);
                  printf(" %d",Alloc[type][row][col]);
               }
            }
            d += d;
       }
       while (total > Rate);
       printf("\nRate of %d is
%f\n",type,(float)total/(16*X_blk*Y_blk));
    }
}

float  qx(int  bit, int  i)
{
   switch (bit)
```

```
    {
      case 1 : return(qx1[i]);
      case 2 : return(qx2[i]);
      case 3 : return(qx3[i]);
      case 4 : return(qx4[i]);
      case 5 : return(qx5[i]);
      case 6 : return(qx6[i]);
      case 7 : return(qx7[i]);
      case 8 : return(qx8[i]);
    }
}

int    Compare(float  value, int  bit)
{
  int    i;

    for (i=0 ; i<step[bit] ; i++)
      if (value < qx(bit,i))
          break;
    return(i);
}

int    MaxQuant(float  value,int  bit)
{
    if (value < 0.0)
      return(-Compare(-value, bit));
    return(Compare(value, bit));
}

void  Quantize(int ttl_base)
{
  int     l,m,n,o,p,j,k,row;
  int
point2,point,fix,pos,pos1,hor,ver,dummy[16];
  int     x_pos,y_pos,total;
  float   point1;
  FILE    *fp3;
  static  char    fname[] = "fname.dat";

    printf("  FILE NAME (BUFFER) ... ");
    scanf("%s",fname);
    fflush(stdin);
    printf("\n");
    if (( fp3 = fopen(fname,"rb") ) == NULL)
      OpenFileFai();

    total  =  0;
    pos    =  -1;

    do
    {
      p      =  0;
      l      =  0;
      fix    =  0;
      ver    =  0;
      hor    =  0;
      row    =  SizeImage/siz_base;

      for (point=0 ; point<16 ; point++)
      {
          dummy[point] = 0;
      }

      do
      {
          pos++;
      }
      while (blk_stat[pos] == 2 || blk_stat[pos] ==
3);
      point  =  pos;

      do
      {
          dummy[p]  =  pos;
           p++;
          point1  =  point;
          hor = (fix == 1) ? 1 : 0;

          if (((fix == 1)&&(blk_stat[pos] == 3)) ||
(fix == 0))
          {
            do
            {
              hor++;
              pos++;
              dummy[p]  =  pos;
              p++;
              if ((pos-point) < row)
                  point2  =  pos;
              fix  =  1;
            }
            while ((blk_stat[pos] == 2) && (pos <
sqr(row)));
            dummy[p-1]  =  0;
            p--;
          }
          else
          {
            pos  =  sqr(row);
          }

          l  +=  row;
          point1  +=  l;
          pos1  =  pos;
          pos  =  point1;
          ver++;
      }
      while ((blk_stat[pos] == 3) && (pos1 <
sqr(row)));
      hor--;
      pos  =  pos1;
      switch(ver)
      {
          case 4 :  index = 0;
                    break;
```

```
        case 2 :  index = (hor == 2) ? 1 : 2;
                  break;
        case 1 :  index = (hor == 1) ? 3 : 4;
                  break;
    }

    switch(index)
    {
        case 0 :  total++;
                  x_pos  = 4;
                  y_pos  = 4;
                  break;
        case 1 :  total++;
                  x_pos  = 2;
                  y_pos  = 2;
                  break;
        case 2 :  total++;
                  x_pos  = 2;
                  y_pos  = 1;
                  break;
        case 3 :  total++;
                  x_pos  = 1;
                  y_pos  = 2;
                  break;
        case 4 :  total++;
                  x_pos  = 1;
                  y_pos  = 1;
                  hor  = 1;
                  break;
    }

    o   = 0;
    fix = 0;
    for (m=0 ; m<x_pos ; m++)
    {
        for (l=0 ; l<4 ; l++)
        {
         point = fix;
         for (n=0 ; n<y_pos ; n++)
         {
          for (p=0 ; p<4 ; p++)
          {
           point1   = Normal[index];
           if ((((4*m)+l)==0) && (((4*n)+p)==0))
           {
             point1 = Normal[index];
             Normal[index] = 100;
           }
           else
             Normal[index] = 100;

fread(&Coeff[index][(4*m)+l][(4*n)+p],4,1,fp3);
           if (Alloc[index][(4*m)+l][(4*n)+p] == 0)
             Image[dummy[point]][l][p] = 0;
           else
             Image[dummy[point]][l][p] =
MaxQuant((Coeff[index][(4*m)+l][(4*n)+p]
/(Normal[index]*pow(2.0,Alloc[index][(4*m)+l][(4*n)+
p]-1.0))),Alloc[index][(4*m)+l][(4*n)+p]);
             Normal[index] = point1;
           }
           point++;
          }
          o++;
         }
         fix += (y_pos < x_pos)  ?  y_pos  :
x_pos;
        }
        pos = point2-1;
    }
    while (total < sum_blk);
    fclose(fp3);
    printf("\nClose file already\n");
}

/*---set gray register---*/
set_gray_REG ()
{
  int i,j=0;

    for (i=64;i<128;i++)
    {
      wrcolor (i,j,j,j);
      j++;
    }
}

wrcolor (int regnum,int red,int green,int blue)
{
    _BX = regnum;
    _DH = red;
    _CH = green;
    _CL = blue;
    _AX = 0x1010;
    geninterrupt (0x10);
}

/*---set graphic mode---*/
setVGA  (char mode)
{
    _AH = 0;
    _AL = mode;
    geninterrupt (0x10);
}

/*---write dot---*/
put(c,p,x,y)
int c,p,x,y;
{
    _AH = 0x0C;
    _AL = c;
    _BH = p;
    _CX = x;
    _DX = y;
```

```
    geninterrupt(0x10);
}

void display(char huge ***Image ,int x, int y,int z)
{
  int i,j,k,row;

    row = SizeImage/siz_base;
    for (i=0;i<z;i++)
      for (j=0;j<y;j++)
          for (k=0;k<x;k++)
            put(Image[i][j][k]/4 +
64,0,((i%row)*x)+k+20,((i/row)*y)+j+20);
}

void WriteImage(char huge ***Image,int ttl_base)
{
  int  row,col;

    setVGA(0x13);
    set_gray_REG();
    display(Image,siz_base,siz_base,ttl_base);
}

void   Pack(int ttl_base)
{
  int     i,j,k;
  int     l,m,n,o,p,row;
  int
point2,point,fix,pos,pos1,hor,ver,dummy[16];
  int     x_pos,y_pos,total;
  int     stor_1,stor_2,stor_3,stor_4,stor_5;
  float     rate_1,rate_2,rate_3,rate_4,rate_5;
  float   point1;
   FILE       *fp2,*fp4,*fp5,*fp6,*fp7,*fp8,*fp9;
   static   char      fname[] = "fname.dat";
   static   char      fname4[] = "fname4.dat";
   static   char      fname5[] = "fname5.dat";
   static   char      fname6[] = "fname6.dat";
   static   char      fname7[] = "fname7.dat";
   static   char      fname8[] = "fname8.dat";
   static   char      fname9[] = "fname9.dat";

    printf("\n   FILE NAME (COMPRESS) ...   ");
    scanf("%s",fname);
    fflush(stdin);
    printf("\n");
    if (( fp2 = fopen(fname,"wb") ) == NULL)
      OpenFileFai();

    fwrite(&sum_blk,2,1,fp2);

    for (i=0 ; i<5 ; i++)
        fwrite(&Normal[i],4,1,fp2);

    for (i=0 ; i<5 ; i++)
        for (j=0 ; j<16 ; j++)
          for (k=0 ; k<16 ; k++)
            fwrite(&Alloc[i][j][k],1,1,fp2);

    for (i=0 ; i<ttl_base ; i++)
        fwrite(&blk_stat[i],1,1,fp2);

    fclose(fp2);

    printf("\n   FILE NAME (Step-1) ...   ");
    scanf("%s",fname5);
    fflush(stdin);
    printf("\n");
    if (( fp5 = fopen(fname5,"wb") ) == NULL)
      OpenFileFai();

    printf("\n   FILE NAME (Step-2) ...   ");
    scanf("%s",fname6);
    fflush(stdin);
    printf("\n");
    if (( fp6 = fopen(fname6,"wb") ) == NULL)
      OpenFileFai();

    printf("\n   FILE NAME (Step-3) ...   ");
    scanf("%s",fname7);
    fflush(stdin);
    printf("\n");
    if (( fp7 = fopen(fname7,"wb") ) == NULL)
      OpenFileFai();

    printf("\n   FILE NAME (Step-4) ...   ");
    scanf("%s",fname8);
    fflush(stdin);
    printf("\n");
    if (( fp8 = fopen(fname8,"wb") ) == NULL)
      OpenFileFai();

    printf("\n   FILE NAME (Step-5) ...   ");
    scanf("%s",fname9);
    fflush(stdin);
    printf("\n");
    if (( fp9 = fopen(fname9,"wb") ) == NULL)
      OpenFileFai();

    total  =  0;
    pos    =  -1;
    stor_1 =  0;
    stor_2 =  0;
    stor_3 =  0;
    stor_4 =  0;
    stor_5 =  0;

    do
    {
     p      =  0;
     l      =  0;
     fix    =  0;
     ver    =  0;
     hor    =  0;
     row    =  SizeImage/siz_base;
```

```
    for (point=0 ; point<16 ; point++)
    {
        dummy[point] = 0;
    }

    do
    {
        pos++;
    }
    while (blk_stat[pos] == 2 || blk_stat[pos] ==
3);
    point = pos;

    do
    {
        dummy[p] = pos;
          p++;
        point1 = point;
        hor = (fix == 1) ? 1 : 0;

        if (((fix == 1)&&(blk_stat[pos] == 3)) ||
(fix == 0))
        {
          do
          {
            hor++;
            pos++;
            dummy[p] = pos;
            p++;
            if ((pos-point) < row)
                point2 = pos;
            fix = 1;
          }
          while ((blk_stat[pos] == 2) && (pos <
sqr(row)));
          dummy[p-1] = 0;
          p--;
        }
        else
        {
          pos = sqr(row);
        }

        l += row;
        point1 += l;
        pos1 = pos;
        pos = point1;
        ver++;
    }
    while ((blk_stat[pos] == 3) && (pos1 <
sqr(row)));
    hor--;
    pos = pos1;
    switch(ver)
    {
        case 4 : index = 0;
                 break;
        case 3 : index = (hor == 3) ? 1 : 2;
                 break;
        case 2 : index = (hor == 2) ? 1 : 2;
                 break;
        case 1 : index = (hor == 1) ? 3 : 4;
                 break;
    }

    switch(index)
    {
        case 0 : total++;
                 x_pos = 4;
                 y_pos = 4;
                 break;
        case 1 : total++;
                 x_pos = 2;
                 y_pos = 2;
                 break;
        case 2 : total++;
                 x_pos = 2;
                 y_pos = 1;
                 break;
        case 3 : total++;
                 x_pos = 1;
                 y_pos = 2;
                 break;
        case 4 : total++;
                 x_pos = 1;
                 y_pos = 1;
                 hor = 1;
                 break;
    }

    o = 0;
    fix = 0;
    for (m=0 ; m<x_pos ; m++)
    {
      for (l=0 ; l<4 ; l++)
      {
       point = fix;
       for (n=0 ; n<y_pos ; n++)
       {
        for (p=0 ; p<4 ; p++)
          if (mask[index][(4*m)+l][(4*n)+p]!=0)
          {
            if (mask[index][(4*m)+l][(4*n)+p]<2)
            {

fwrite(&Image[dummy[point]][l][p],1,1,fp5);
                stor_1 +=
Alloc[index][(4*m)+l][(4*n)+p];
            }
            if (mask[index][(4*m)+l][(4*n)+p]<3)
            {
fwrite(&Image[dummy[point]][l][p],1,1,fp6);
                stor_2 +=
Alloc[index][(4*m)+l][(4*n)+p];
            }
```

```
                if (mask[index][(4*m)+l][(4*n)+p]<4)
                {

fwrite(&Image[dummy[point]][l][p],1,1,fp7);
                        stor_3 +=
Alloc[index][(4*m)+l][(4*n)+p];
                }
                if (mask[index][(4*m)+l][(4*n)+p]<5)
                {

fwrite(&Image[dummy[point]][l][p],1,1,fp8);
                        stor_4 +=
Alloc[index][(4*m)+l][(4*n)+p];
                }
                if (mask[index][(4*m)+l][(4*n)+p]<6)
                {

fwrite(&Image[dummy[point]][l][p],1,1,fp9);
                        stor_5 +=
Alloc[index][(4*m)+l][(4*n)+p];
                }
              }
            point++;
           }
           o++;
          }
          fix += (y_pos < x_pos) ? y_pos :
x_pos;
       }
       pos = point2-1;
     }
     while (total < sum_blk);
     rate_1 = (float)stor_1/(SizeImage*SizeImage);
     rate_2 = (float)stor_2/(SizeImage*SizeImage);
     rate_3 = (float)stor_3/(SizeImage*SizeImage);
     rate_4 = (float)stor_4/(SizeImage*SizeImage);
     rate_5 = (float)stor_5/(SizeImage*SizeImage);
     printf("\nStep-1 use %d bits (%f
bpp)",stor_1,rate_1);
     printf("\nStep-2 use %d bits (%f
bpp)",stor_2,rate_2);
     printf("\nStep-3 use %d bits (%f
bpp)",stor_3,rate_3);
     printf("\nStep-4 use %d bits (%f
bpp)",stor_4,rate_4);
     printf("\nStep-5 use %d bits (%f
bpp)",stor_5,rate_5);

    fclose(fp5);
    fclose(fp6);
    fclose(fp7);
    fclose(fp8);
    fclose(fp9);
    printf("\nClose file already\n");

     printf("\n   FILE NAME (LINKAGE MATRIX) ...
");
     scanf("%s",fname4);
     fflush(stdin);
     printf("\n");
     if (( fp4 = fopen(fname4,"wb") ) == NULL)
       OpenFileFai();

     for (i=0 ; i<ttl_base ; i++)
          fwrite(&blk_stat[i],1,1,fp4);
     fclose(fp4);
     printf("Pack already\n ");
     exit(0);
}

main()
{
   int       i,j,k,ttl_base;
   char      ans;

     printf("Enter your dimension\n");
     scanf("%d",&SizeImage);
     fflush(stdin);
     printf("Enter your basic block size\n");
     scanf("%d",&siz_base);
     fflush(stdin);
     ttl_base  =  sqr(SizeImage/siz_base);
     ReadFileImage(ttl_base,siz_base);
     printf("Successful Reading %d
Image\n",SizeImage);
     Activ(ttl_base,siz_base);
     printf("coreleft = %u\n",coreleft());

     printf("coreleft = %u\n",farcoreleft());

     printf("desired threshold value is
%f",threshold);
     fflush(stdin);
     Cluster(threshold,ttl_base);

     DCT2D();
     Quantize(ttl_base);
     printf("\nQuantize pass\n");

     Pack(ttl_base);
     for (i=0 ; i<ttl_base ; i++)
          for (j=0 ; j<siz_base ; j++)
            farfree(Image[i][j]);
          farfree(Image[i]);
     farfree(Image);

     printf("\nImage already compress!");
     printf("\nGood Luck Man");
     getch();
     exit(0);
}
```

```
/*Decoder Part*/
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#define     sqr(A)     ((A)*(A))

void  ReadFileImage(int ttl_base,int iterate);
void  DeQuantize();
void  DCT2D();
void  WriteImage(char huge ***Image,int ttl_base);

float            *energy,B_Rate;
char             *blk_stat;
int              SizeImage;
int              siz_base;
int              sum_blk;
int              index;
float            Array[16],Vector[16],Normal[5];
unsigned char    *block_1;
float far        ***Table;
float far        ***Coeff;
char             ***Alloc;
char huge        ***Image;
int              step[9] = {0,1,2,4,8,16,32,64,128};

int         mask[5][16][16] = {{{1,1,2,2,3,3,3,3,4,4,4,4,5,5,5,5},
            {1,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5},
            {2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5},
            {2,2,3,3,3,3,3,3,4,4,4,4,5,5,5,5},
            {3,3,3,3,3,3,3,4,4,4,4,4,5,5,5,5},
            {3,3,3,3,3,3,4,4,4,4,4,5,5,5,5,5},
            {3,3,3,3,3,4,4,4,4,4,5,5,5,5,5,5},
            {3,3,3,3,4,4,4,4,4,5,5,5,5,5,5,5},
            {4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5},
            {4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5},
            {4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5},
            {4,4,4,4,4,5,5,5,5,5,5,5,5,5,5,5},
            {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5},
            {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5},
            {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5},
            {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5}},
            {{1,2,2,3,3,4,4,5,0,0,0,0,0,0,0,0},
            {2,2,2,3,3,4,4,5,0,0,0,0,0,0,0,0},
            {2,2,3,3,3,4,4,5,0,0,0,0,0,0,0,0},
            {3,3,3,3,4,4,5,5,0,0,0,0,0,0,0,0},
            {3,3,3,4,4,5,5,5,0,0,0,0,0,0,0,0},
            {4,4,4,4,5,5,5,5,0,0,0,0,0,0,0,0},
            {4,4,4,5,5,5,5,5,0,0,0,0,0,0,0,0},
            {5,5,5,5,5,5,5,5,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}},
            {{1,2,2,3,3,4,4,5,0,0,0,0,0,0,0,0},
            {2,2,3,3,4,4,5,5,0,0,0,0,0,0,0,0},
            {3,3,3,4,4,5,5,5,0,0,0,0,0,0,0,0},
            {4,4,4,4,5,5,5,5,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
```

```
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}},
{{1,2,3,4,0,0,0,0,0,0,0,0,0,0,0,0},
{2,2,3,4,0,0,0,0,0,0,0,0,0,0,0,0},
{2,3,3,4,0,0,0,0,0,0,0,0,0,0,0,0},
{3,3,4,4,0,0,0,0,0,0,0,0,0,0,0,0},
{3,4,4,5,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,5,5,0,0,0,0,0,0,0,0,0,0,0,0},
{4,5,5,5,0,0,0,0,0,0,0,0,0,0,0,0},
{5,5,5,5,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}},
{{1,2,3,4,0,0,0,0,0,0,0,0,0,0,0,0},
{2,2,3,4,0,0,0,0,0,0,0,0,0,0,0,0},
{3,3,4,5,0,0,0,0,0,0,0,0,0,0,0,0},
{4,4,5,5,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}}};

static float qy1[1]   ={0.797885};
static float qy2[2]   ={0.452780, 1.510418};
static float qy3[4]   ={0.245094, 0.756005, 1.343909, 2.151945};
static float qy4[8]   ={0.128395, 0.388048, 0.656759, 0.942340,
1.256231, 1.618046, 2.069017, 2.732589};
static float qy5[16]  ={0.065890, 0.198052, 0.331378, 0.466700,
0.604934, 0.747136, 0.894565, 1.048783,
                          1.211804, 1.386340, 1.576228,
1.787233, 2.028729, 2.317740, 2.691119, 3.260732};
static float qy6[32]  ={0.033409, 0.100278, 0.167297, 0.234567,
0.302193, 0.370283, 0.438950, 0.508314,
                          0.578503, 0.649655, 0.721922,
0.795467, 0.870476, 0.947155, 1.025736, 1.106488,
                          1.189720, 1.275794, 1.365141,
1.458276, 1.555831, 1.658588, 1.767541, 1.883977,
                          2.009610, 2.146809, 2.298980,
2.471304, 2.672272, 2.917405, 3.240434, 3.744096};
static float qy7[64]  ={0.016828, 0.050491, 0.084173, 0.117886,
0.151645, 0.185461, 0.219348, 0.253319,
                          0.287388, 0.321568, 0.355874,
0.390320, 0.424922, 0.459693, 0.494651, 0.529812,
                          0.565193, 0.600810, 0.636684,
0.672833, 0.709278, 0.746039, 0.783140, 0.820603,
                          0.858454, 0.896720, 0.935428,
0.974610, 1.014297, 1.054523, 1.095327, 1.136747,
                          1.178829, 1.221617, 1.265165,
1.309528, 1.354766, 1:400948, 1.448149, 1.496450,
                          1.545944, 1.596733, 1.648934,
1.702677, 1.758111, 1.815408, 1.874762, 1.936404,
                          2.000602, 2.067671, 2.137993,
2.212027, 2.290339, 2.373634, 2.462811, 2.559040,
                          2.663886, 2.779514, 2.909047,
3.057246, 3.231933, 3.447430, 3.734935, 4.189692};
static float qy8[128] ={0.008446, 0.025339, 0.042235, 0.059135,
0.076040, 0.092952, 0.109874, 0.126806,
                          0.143750, 0.160707, 0.177681,
0.194671, 0.211680, 0.228709, 0.245761, 0.262836,
                          0.279938, 0.297066, 0.314223,
0.331412, 0.348633, 0.365888, 0.383180, 0.400511,
                          0.417881, 0.435294, 0.452750,
0.470253, 0.487804, 0.505405, 0.523059, 0.540767,
                          0.558532, 0.576355, 0.594240,
0.612189, 0.630203, 0.648286, 0.666440, 0.684668,
                          0.702971, 0.721353, 0.739817,
0.758366, 0.777002, 0.795728, 0.814548, 0.833464,
                          0.852480, 0.871600, 0.890826,
0.910163, 0.929614, 0.949183, 0.968874, 0.988691,
                          1.008638, 1.028719, 1.048940,
1.069305, 1.089819, 1.110487, 1.131315, 1.152307,
```

```
                                1.173470, 1.194809, 1.216331,
1.238043, 1.259952, 1.282063, 1.304386, 1.326927,
                                1.349695, 1.372699, 1.395948,
1.419451, 1.443219, 1.467261, 1.491589, 1.516215,
                                1.541152, 1.566412, 1.592010,
1.617960, 1.644279, 1.670983, 1.698090, 1.725620,
                                1.753592, 1.782030, 1.810956,
1.840396, 1.870378, 1.900931, 1.932087, 1.963882,
                                1.996352, 2.029539, 2.063489,
2.098250, 2.133878, 2.170432, 2.207980, 2.246594,
                                2.286359, 2.327367, 2.369722,
2.413544, 2.458968, 2.506148, 2.555266, 2.606528,
                                2.660181, 2.716515, 2.775879,
2.838694, 2.905482, 2.976892, 3.053753, 3.137144,
                                3.228513, 3.329863, 3.444089,
3.575608, 3.731690, 3.925669, 4.186589, 4.603524};


AlloFai()
{
    printf("  Memory allocation is error.  \n");
    exit(1);
}

void   OpenFileFai()
{
    printf("Can't open file. \n");
    printf("Press any key to exit. \n");
    getch();
    exit(1);
}

char huge *ch1D(int  siz_base)
{
  char huge *a;

    if ( (a = (char huge*)farmalloc(siz_base)) == NULL)
      AlloFai();
    return(a);
}

char huge **ch2D(int  siz_base)
{
  char huge **b;
  int    i;

    if ( (b = (char huge**)farmalloc (siz_base*sizeof(char huge*)) )
== NULL)
      AlloFai();
    for (i=0 ; i<siz_base ; i++)
      b[i] = ch1D(siz_base);
    return(b);
}

char huge ***ch3D(int  ttl_base,int  siz_base)
{
  char huge ***b;
  int    i;

    if ( (b = (char huge***)farmalloc (ttl_base*sizeof(char huge**))
) == NULL)
      AlloFai();
    for (i=0 ; i<ttl_base ; i++)
      b[i] = ch2D(siz_base);
    return(b);
}

float huge  *Mark1D(int  hor)
{
  float huge *a;

    if ((a = (float huge*)farcalloc(hor,4) ) == NULL)
      AlloFai();
    return(a);
}

float huge  **Mark2D(int  ver, int  hor)
{
  float huge **b;
  int    i;

    if ((b = (float huge**)farmalloc(ver*sizeof(float huge*) ) ) ==
NULL)
      AlloFai();
    for (i=0 ; i<ver ; i++)
      b[i] = Mark1D(hor);
    return(b);
}

float huge  ***Mark3D(int size)
{
  float huge ***c;
  int    i;
    if ((c = (float huge***)farmalloc(size*sizeof(float huge**) ) ) ==
NULL)
      AlloFai();
    for (i=0 ; i<5 ; i++)
      c[i] = Mark2D(16,16);
    return(c);
}

unsigned char   *Unsigned(int sum_blk)
{
  unsigned char *a;
    if ( (a = (unsigned char *)malloc(sum_blk)) == NULL)
      AlloFai();
    return(a);
}

void   ReadFileImage(int  ttl_base,int  iterate)
{
  char    buffer;
  int     l,m,n,o,p,i,j,k,row;
  int     point2,point1,point,fix,pos,pos1,hor,ver,dummy[16];
  int     x_pos,y_pos,total;
  FILE  *fp1;
  FILE  *fp,*fp8;
  static char  fname[] = "fname.dat";
```

```
static char  fname8[] = "fname8.dat";


 if (iterate == 0)
 {
   Image   =  ch3D(ttl_base,siz_base);
   Alloc   =  ch3D(5,16);

   if ( ( blk_stat = (char *)calloc(ttl_base,2)) == NULL )
         AlloFai();

   printf("\n  FILE NAME (LINK) ... ");
   scanf("%s",fname);
   fflush(stdin);
   if ((fp1 = fopen(fname,"rb")) == NULL)
         OpenFileFai();

   for (i=0 ; i<ttl_base ; i++)
         fread(&blk_stat[i],1,1,fp1);
   fclose(fp1);

   printf("\n  FILE NAME (COMPRESS) ... ");
   scanf("%s",fname);
   fflush(stdin);
   if ((fp = fopen(fname,"rb")) == NULL)
         OpenFileFai();

   fread(&sum_blk,2,1,fp);

   for (i=0 ; i<5 ; i++)
         fread(&Normal[i],4,1,fp);

   for (i=0 ; i<5 ; i++)
         for (j=0 ; j<16 ; j++)
           for (k=0 ; k<16 ; k++)
             fread(&Alloc[i][j][k],1,1,fp);
   fclose(fp);
 }

 printf("\n  FILE NAME (Step-%d) ... ",iterate+1);
 scanf("%s",fname8);
 fflush(stdin);
 if ((fp8 = fopen(fname8,"rb")) == NULL)
   OpenFileFai();

 total  =  0;
 pos    =  -1;

 if (iterate==0)
   for (index=0 ; index<ttl_base ; index++)
         for (l=0 ; l<4 ; l++)
           for (p=0 ; p<4 ; p++)
             Image[index][l][p]=0.0;

 do
 {
   p      =  0;
   l      =  0;
   fix    =  0;
   ver    =  0;
   hor    =  0;
   row    =  SizeImage/siz_base;

   for (point=0 ; point<16 ; point++)
   {
         dummy[point] = 0;
   }

   do
   {
         pos++;
   }
   while (blk_stat[pos] == 2 || blk_stat[pos] == 3);
   point  =  pos;

   do
   {
         dummy[p]  =  pos;
           p++;
         point1 = point;
         hor = (fix == 1) ? 1 : 0;

         if (((fix == 1)&&(blk_stat[pos] == 3)) || (fix == 0))
         {
           do
           {
             hor++;
             pos++;
             dummy[p]  =  pos;
             p++;
             if ((pos-point) < row)
                   point2  =  pos;
             fix  =  1;
           }
           while ((blk_stat[pos] == 2) && (pos < sqr(row)));
           dummy[p-1]  =  0;
           p--;
         }
         else
         {
           pos  =  sqr(row);
         }

         l  +=  row;
         point1 += l;
         pos1 =  pos;
         pos = point1;
         ver++;
   }
   while ((blk_stat[pos] == 3) && (pos1 < sqr(row)));

   hor--;
   pos  =  pos1;
   switch(ver)
   {
         case 4 :  index = 0;
                   break;
         case 2 :  index = (hor == 2) ? 1 : 2;
```

```
                break;
        case 1 :  index = (hor == 1) ? 3 : 4;
                break;
}

switch(index)
{
        case 0 :  total++;
                x_pos   =  4;
                y_pos   =  4;
                break;
        case 1 :  total++;
                x_pos   =  2;
                y_pos   =  2;
                break;
        case 2 :  total++;
                x_pos   =  2;
                y_pos   =  1;
                break;
        case 3 :  total++;
                x_pos   =  1;
                y_pos   =  2;
                break;
        case 4 :  total++;
                x_pos   =  1;
                y_pos   =  1;
                hor  =  1;
                break;
}

o   = 0;
fix = 0;
for (m=0 ; m<x_pos ; m++)
{
     for (l=0 ; l<4 ; l++)
     {
      point = fix;

      for (n=0 ; n<y_pos ; n++)
      {
       for (p=0 ; p<4 ; p++)
       {
        if (mask[index][(4*m)+l][(4*n)+p] > (iterate+1))
          Image[dummy[point]][l][p] = 0;
        if (mask[index][(4*m)+l][(4*n)+p] == (iterate+1))
        {
          fread(&buffer,1,1,fp8);
          Image[dummy[point]][l][p] = buffer;
        }
        if (mask[index][(4*m)+l][(4*n)+p] < (iterate+1))
          if (mask[index][(4*m)+l][(4*n)+p] > 0)
          {
            fread(&buffer,1,1,fp8);
            Image[dummy[point]][l][p] = buffer;
          }
       }
       point++;
      }
      o++;
        }
        fix += (y_pos < x_pos)  ?  y_pos  :  x_pos;
      }
      pos = point2-1;
    }
    while (total < sum_blk);

    fclose(fp8);
}

float       qy(code, bit)
int         code;
int         bit;
{
            switch (bit)
            {
              case 1 : return(qy1[-code]);
              case 2 : return(qy2[-code]);
              case 3 : return(qy3[-code]);
              case 4 : return(qy4[-code]);
              case 5 : return(qy5[-code]);
              case 6 : return(qy6[-code]);
              case 7 : return(qy7[-code]);
              case 8 : return(qy8[-code]);
            }
}

float       Lookup(int  code, int  bit)
{
            if (code < 0) return(-qy(-code, bit));
            return(qy(code, bit));
}

void  DeQuantize()
{
  int     l,m,n,o,p,j,k,row;
  int     point2,point,fix,pos,pos1,hor,ver,dummy[16];
  int     x_pos,y_pos,total;
  float   point1;
  FILE    *fp3;
  static  char  fname[] = "fname.dat";

  printf("\n  FILE NAME (BUFFER) ... ");
  scanf("%s",fname);
  printf("\n");
  if (( fp3 = fopen(fname,"wb") ) == NULL)
    OpenFileFai();

  total  =  0;
  pos    =  -1;

  do
  {
    p      =  0;
    l      =  0;
    fix    =  0;
    ver    =  0;
    hor    =  0;
    row    =  SizeImage/siz_base;
```

```
for (point=0 ; point<16 ; point++)
{
        dummy[point] = 0;
}

do
{
        pos++;
}
while (blk_stat[pos] == 2 || blk_stat[pos] == 3);
point  =  pos;

do
{
        dummy[p]  =  pos;
          p++;
        point1  =  point;
        hor = (fix == 1) ? 1 : 0;

        if (((fix == 1)&&(blk_stat[pos] == 3)) || (fix == 0))
        {
          do
          {
            hor++;
            pos++;
            dummy[p]  =  pos;
            p++;
            if ((pos-point) < row)
                  point2  =  pos;
            fix  =  1;
          }
          while ((blk_stat[pos] == 2) && (pos < sqr(row)));
          dummy[p-1]  =  0;
          p--;
        }
        else
        {
          pos  =  sqr(row);
        }

        l  +=  row;
        point1  +=  l;
        pos1  =  pos;
        pos  =  point1;
        ver++;
}
while ((blk_stat[pos] == 3) && (pos1 < sqr(row)));

hor--;
pos  =  pos1;
switch(ver)
{
        case 4 :  index = 0;
                  break;
        case 2 :  index = (hor == 2) ? 1 : 2;
                  break;
        case 1 :  index = (hor == 1) ? 3 : 4;
                  break;
}

switch(index)
{
        case 0 :  total++;
                  x_pos   =  4;
                  y_pos   =  4;
                  break;
        case 1 :  total++;
                  x_pos   =  2;
                  y_pos   =  2;
                  break;
        case 2 :  total++;
                  x_pos   =  2;
                  y_pos   =  1;
                  break;
        case 3 :  total++;
                  x_pos   =  1;
                  y_pos   =  2;
                  break;
        case 4 :  total++;
                  x_pos   =  1;
                  y_pos   =  1;
                  hor  =  1;
                  break;
}
o  = 0;
fix = 0;
for (m=0 ; m<x_pos ; m++)
{
        for (l=0 ; l<4 ; l++)
        {
         point = fix;

         /*  dat_fl[i][0][0] += DCmean;  */
         for (n=0 ; n<y_pos ; n++)
         {
          for (p=0 ; p<4 ; p++)
          {
           point1   = Normal[index];
           if ((((4*m)+l)==0) && (((4*n)+p)==0))
           {
             point1 = Normal[index];
             Normal[index] = 100;
           }
           else
             Normal[index] = 100;
           if (Alloc[index][(4*m)+l][(4*n)+p] == 0)
             Coeff[index][(4*m)+l][(4*n)+p] = 0;
           else
             Coeff[index][(4*m)+l][(4*n)+p] =
(Normal[index]*pow(2.0,Alloc[index][(4*m)+l][(4*n)+p]-1.0))
*Lookup(Image[dummy[point]][l][p]
,Alloc[index][(4*m)+l][(4*n)+p]);
           fwrite(&Coeff[index][(4*m)+l][(4*n)+p],4,1,fp3);
           Normal[index] = point1;
```

```
            }
            point++;
          }
          o++;
        }
        fix += (y_pos < x_pos) ? y_pos : x_pos;
      }
      pos = point2-1;
    }
    while (total < sum_blk);
    fclose(fp3);
}


float COS(int i, int j, float k)
{
  float PI;
  float a;

  PI = 3.141592654;
  a = ((float)(i*PI) * (float)(2.0*j + 1) ) / (float)(2.0*k);
  return(cos(a));
}

void CalTable()
{
  int i,j;

  for (i=0 ; i<4 ; i++)
    for (j=0 ; j<4 ; j++)
    {
        Table[4][i][j] = COS(i,j,4);
    }
  for (i=0 ; i<8 ; i++)
    for (j=0 ; j<8 ; j++)
    {
        Table[1][i][j] = COS(i,j,8);
    }
  for (i=0 ; i<16 ; i++)
    for (j=0 ; j<16 ; j++)
    {
        Table[0][i][j] = COS(i,j,16);
    }
}

int clip(int x)
{
  if (x>0)
  {
    if (x>255)
        return(255);
    else
        return(x);
  }
  else
    return(0);
}

void DCT1D(int x_pos, int y_pos)
{
  int x,y,z;

  switch (y_pos)
  {
    case 4 : z = 4;  break;
    case 8 : z = 1;  break;
    case 16 : z = 0;  break;
  }

  if (x_pos < y_pos)
    x_pos = y_pos;
  if (x_pos > y_pos)
    y_pos = x_pos;

  for (x=0 ; x<y_pos ; x++)
    Vector[x] = 0.0;

  for (x=0 ; x<x_pos ; x++)
  {
    for (y=0 ; y<y_pos ; y++)
    {
        Vector[x] += Table[z][y][x]*Array[y];
    }
  }
}

void DCT2D()
{
FILE  *fp2;
static char  fname2[] = "fname2.dat";
int  m,n,u,row;
int  i,l,o,p,point2,point1,point,fix,pos,pos1,hor,ver,dummy[16];
int  x_pos,y_pos,total,ttl0,ttl1,ttl2,ttl3,ttl4;
float  Buffer[16][16],scale1,scale2;

Table  = Mark3D(3);

printf(" FILE NAME (BUFFER) ... ");
scanf("%s",fname2);
if ((fp2 = fopen(fname2,"rb")) == NULL)
  OpenFileFai();

block_1 = Unsigned(sum_blk);

CalTable();
ttl0 = 0;
ttl1 = 0;
ttl2 = 0;
ttl3 = 0;
ttl4 = 0;
total = 0;
pos = -1;
for (u=0 ; u<256 ; u++)
  block_1[u] = 0;
u = 0;

do
```

```
{
   p      =  0;
   l      =  0;
   fix    =  0;
   ver    =  0;
   hor    =  0;
   row    =  SizeImage/siz_base;

   for (point=0 ; point<16 ; point++)
   {
         dummy[point] = 0;
   }

   do
   {
         pos++;
   }
   while (blk_stat[pos] == 2 || blk_stat[pos] == 3);
   point  =  pos;

   do
   {
         dummy[p]  =  pos;
           p++;
         point1  =  point;
         hor = (fix == 1) ? 1 : 0;

         if (((fix == 1)&&(blk_stat[pos] == 3)) || (fix == 0))
         {
           do
           {
              hor++;
              pos++;
              dummy[p]  =  pos;
              p++;
              if ((pos-point) < row)
                    point2  =  pos;
              fix  =  1;
           }
           while ((blk_stat[pos] == 2) && (pos < sqr(row)));
           dummy[p-1]  =  0;
           p--;
         }
         else
         {
           pos  =  sqr(row);
         }

         l  += row;
         point1  +=  l;
         pos1 = pos;
         pos  =  point1;
         ver++;
   }
   while ((blk_stat[pos] == 3) && (pos1 < sqr(row)));

   hor--;
   pos  =  pos1;
   switch(ver)
   {
         case 4 :  index = 0;
                   break;
         case 2 :  index = (hor == 2) ? 1 : 2;
                   break;
         case 1 :  index = (hor == 1) ? 3 : 4;
                   break;
   }

   switch(index)
   {
         case 0 :  total++;
                   ttl0++;
                   x_pos  =  4;
                   y_pos  =  4;
                   break;
         case 1 :  total++;
                   ttl1++;
                   x_pos  =  2;
                   y_pos  =  2;
                   break;
         case 2 :  total++;
                   ttl2++;
                   x_pos  =  2;
                   y_pos  =  1;
                   break;
         case 3 :  total++;
                   ttl3++;
                   x_pos  =  1;
                   y_pos  =  2;
                   break;
         case 4 :  total++;
                   ttl4++;
                   x_pos  =  1;
                   y_pos  =  1;
                   hor  =  1;
                   block_1[u]  =  dummy[0];
                   u++;
                   break;
   }

   for (m=0 ; m<(4*x_pos) ; m++)
   {
       for (n=0 ; n<(4*y_pos) ; n++)
       {
         fread(&Coeff[index][m][n],4,1,fp2);
         scale1 = (n==0)  ?  0.7071 : 1;
         scale2 = (m==0)  ?  0.7071 : 1;
         Coeff[index][m][n] *= ((scale1*scale2)/16);
         Array[n] = Coeff[index][m][n];
       }
       DCT1D(4*x_pos,4*y_pos);
       for (n=0 ; n<(4*y_pos) ; n++)
         Buffer[m][n] = Vector[n];
   }

   o  = 0;
   fix = 0;
   for (m=0 ; m<y_pos ; m++)
```

```
    {
        for (l=0 ; l<4 ; l++)
        {
         point = fix;

         for (n=0 ; n<(4*x_pos) ; n++)
            Array[n] = Buffer[n][(4*m)+l];
         DCT1D(4*y_pos,4*x_pos);
         for (n=0 ; n<x_pos ; n++)
         {
          for (p=0 ; p<4 ; p++)
            Image[dummy[point]][p][l] = clip(Vector[(4*n)+p])-
128;
          point += (y_pos < x_pos)  ?  y_pos  :  x_pos;
         }
         o++;
        }
        fix++;
    }
    pos = point2-1;
  }
  while (total < sum_blk);
  fclose(fp2);
  B_Rate = 0.0;
  for (l=0 ; l<5 ; l++)
  {
    point  =  0;
    switch(l)
    {
          case 0  :  for (m=0 ; m<16 ; m++)
                        for (n=0 ; n<16 ; n++)
                          point += Alloc[l][m][n];
                     fix = ttl0*point;  break;
          case 1  :  for (m=0 ; m<8 ; m++)
                        for (n=0 ; n<8 ; n++)
                          point += Alloc[l][m][n];
                     fix = ttl1*point;  break;
          case 2  :  for (m=0 ; m<8 ; m++)
                        for (n=0 ; n<4 ; n++)
                          point += Alloc[l][m][n];
                     fix = ttl2*point;  break;
          case 3  :  for (m=0 ; m<4 ; m++)
                        for (n=0 ; n<8 ; n++)
                          point += Alloc[l][m][n];
                     fix = ttl3*point;  break;
          case 4  :  for (m=0 ; m<4 ; m++)
                        for (n=0 ; n<4 ; n++)
                          point += Alloc[l][m][n];
                     fix = ttl4*point;  break;
    }
    B_Rate  +=  fix;
  }
  fix  =  ttl4+1256;
  fix  +=  ttl3*3;   /* 1+2 */
  fix  +=  ttl2*4;   /* 1+3 */
  fix  +=  ttl1*8;   /* 1+3+2+2 */
  fix  +=  ttl0*34;  /* 1+(3*3)+(2*12) */
  B_Rate += fix;
  printf("\n Type 0 has %d blocks\n",ttl0);
  printf("\n Type 1 has %d blocks\n",ttl1);
  printf("\n Type 2 has %d blocks\n",ttl2);
  printf("\n Type 3 has %d blocks\n",ttl3);
  printf("\n Type 4 has %d blocks\n",ttl4);
}


/*---set gray register---*/
set_gray_REG ()
{
  int i,j=0;

  for (i=64;i<128;i++)
  {
    wrcolor (i,j,j,j);
    j++;
  }
}

wrcolor (int regnum,int red,int green,int blue)
{
  _BX = regnum;
  _DH = red;
  _CH = green;
  _CL = blue;
  _AX = 0x1010;
  geninterrupt (0x10);
}

/*---set graphic mode---*/
setVGA  (char mode)
{
  _AH = 0;
  _AL = mode;
  geninterrupt (0x10);
}

/*---write dot---*/
put(c,p,x,y)
int c,p,x,y;
{
  _AH = 0x0C;
  _AL = c;
  _BH = p;
  _CX = x;
  _DX = y;
  geninterrupt(0x10);
}

void display(char huge ***Image ,int x, int y,int z,int bias)
{
  int    i,j,k,row;
  row = SizeImage/siz_base;
  for (i=0;i<z;i++)
    for (j=0;j<x;j++)
        for (k=0;k<y;k++)
          put((Image[i][j][k]+127)/4 +
64,0,((i%row)*x)+k+20+bias,((i/row)*y)+j+20);
```

```
}

void WriteImage(char huge ***Image,int ttl_base)
{
  int     row,col,data,buffer,block,a,b,u;
  float   snr,mse,nmse,power;
  float   snr_1,mse_1,nmse_1,power_1;
  float   snr_2,mse_2,nmse_2,power_2;
  char    order,error;
  FILE    *fp4,*fp5,*fp6,*fp7;
  static  char   fname[] = "fname.dat";
  static  char   fname1[] = "fname1.dat";
  static  char   fname2[] = "fname2.dat";

   if (Alloc[3][0][0] == 0 && Alloc[2][0][0] == 0)
     for (a=0 ; a<ttl_base ; a++)
           block_1[a] = a;

   mse  = mse_1  = mse_2  = 0.0;
   nmse = nmse_1 = nmse_2 = 0.0;
   power = power_1 = power_2 = 0.0;
   order = 0;

   printf("\nImage File (Original) ... ");
   scanf("%s",fname);
   fflush(stdin);
   if ((fp4 = fopen(fname,"rb")) == NULL)
     OpenFileFai();

   printf("\nImage File (Result) ... ");
   scanf("%s",fname2);
   fflush(stdin);
   if ((fp7 = fopen(fname2,"wb")) == NULL)
     OpenFileFai();

   printf("\nImage File (Error) ... ");
   scanf("%s",fname1);
   fflush(stdin);
   if ((fp6 = fopen(fname1,"wb")) == NULL)
     OpenFileFai();

   setVGA(0x13);
   set_gray_REG();
   display(Image,siz_base,siz_base,ttl_base,0);

   for (a=0 ; a<SizeImage ; a++)
     for (b=0 ; b<SizeImage ; b++)
     {
           buffer = 0;
           error  = 0;
           fread(&buffer,1,1,fp4);
           row    = a;
           col    = b;
           data   = siz_base-1;
           while(col>data || row>data)
           {
             if (col > data)
               col -= siz_base;
             if (row > data)
               row -= siz_base;
           }
           block  = (sqrt((double)
ttl_base)*(a/siz_base))+(b/siz_base);

           Image[block][row][col] =
clip(Image[block][row][col]+127);
           fwrite(&Image[block][row][col],1,1,fp7); /* write result
*/
           Image[block][row][col] = Image[block][row][col]-127;
/*to normal*/
           if (block_1[order] == block)
             mse_1  += sqr(Image[block][row][col]-buffer+128);
           else
             mse_2  += sqr(Image[block][row][col]-buffer+128);

           mse   += sqr(Image[block][row][col]-buffer+128);
           error  = Image[block][row][col] - buffer;
           if (error < 0)
             error = -error;
           Image[block][row][col] = buffer-127; /*change image
to original*/

           if (block_1[order] == block)
           {
             power_1 += sqr(Image[block][row][col]);
             order++;
           }
           else
             power_2 += sqr(Image[block][row][col]);

           power  += sqr(Image[block][row][col]);
           fwrite(&error,1,1,fp6);  /* write error */
     }

   nmse   = mse/power;
   nmse_1 = mse_1/power_1;
   nmse_2 = mse_2/power_2;
   mse   /= sqr(SizeImage);
   mse_1 /= sqr(SizeImage);
   mse_2 /= sqr(SizeImage);
   snr    = 10*(log10(65025.0/mse));
   snr_1  = 10*(log10(65025.0/mse_1));
   snr_2  = 10*(log10(65025.0/mse_2));
   power  = 0.0;
   power_1 = 0.0;
   power_2 = 0.0;
   power  = mse/65025;
   power_1 = mse_1/65025;
   power_2 = mse_2/65025;
   display(Image,siz_base,siz_base,ttl_base,128); /*Show original*/
   fclose(fp6);
/* Write error image */
   printf("\nImage File (Error) ... ");
   scanf("%s",fname1);
   fflush(stdin);
   if ((fp5 = fopen(fname1,"rb")) == NULL)
     OpenFileFai();
```

```
    for (a=0 ; a<SizeImage ; a++)
      for (b=0 ; b<SizeImage ; b++)
      {
            buffer  =  0;
            fread(&buffer,1,1,fp5);
            row     =  a;
            col     =  b;
            data    =  siz_base-1;
            while(col>data || row>data)
            {
              if (col > data)
                col  -=  siz_base;
              if (row > data)
                row  -=  siz_base;
            }
            block   =  (sqrt((double)
ttl_base)*(a/siz_base))+(b/siz_base);
            Image[block][row][col] = ((buffer+127)/4); /*change
image to error*/
      }
    display(Image,siz_base,siz_base,ttl_base,128);   /*Show error*/

    gotoxy (130,10);
    puts("Enter any  key to see coder performance");
    getch();
    setVGA(3);

    printf("\n\n\nMean Square Error          : %f percent\n",mse);
    printf("Normalized to Image Energy  : %f percent\n",nmse);
    printf("Normalized to p-p Intensity : %f percent\n",power);
    printf("Signal to Noise Ratio       : %f dB\n",snr);
    printf("\n\n[1*1 block size performance]\n");
    printf("Mean Square Error          : %f percent\n",mse_1);
    printf("Normalized to Image Energy  : %f percent\n",nmse_1);
    printf("Normalized to p-p Intensity : %f percent\n",power_1);
    printf("Signal to Noise Ratio       : %f dB\n",snr_1);
    printf("\n\n[Others blocks size performance]\n");
    printf("Mean Square Error          : %f percent\n",mse_2);
    printf("Normalized to Image Energy  : %f percent\n",nmse_2);
    printf("Normalized to p-p Intensity : %f percent\n",power_2);
    printf("Signal to Noise Ratio       : %f dB\n",snr_2);
    printf("\n\nApprox. Bit Rate\t: %f
bit/pel\n",B_Rate/sqr(SizeImage));
    getch();
    free(block_1);
    fclose(fp4);
    fclose(fp5);
    fclose(fp7);
    exit(1);
}

main()
{
  int  ttl_base,iterate,respond;

  printf("\n\n\n**************************************\n");
  printf("* This is DCT Decoder using blk_stat *\n");
  printf("**************************************\n\n\n");
  printf("\n\nEnter Size Image\n");
  scanf("%d",&SizeImage);
  printf("\nBasic Block size is 4\n");
  siz_base  =  4;
  ttl_base  =  sqr(SizeImage/siz_base);
  Coeff = Mark3D(5);


  iterate   =  0;
  do
  {
    ReadFileImage(ttl_base,iterate);
    DeQuantize();
    DCT2D();

    setVGA(0x13);
    set_gray_REG();
    display(Image,siz_base,siz_base,ttl_base,128);
    puts("Enter any key to pass");
    getch();
    setVGA(3);

    do
    {
      gotoxy(1,20);
      puts("DO YOU NEED MORE DETIAL ?  (y/n)..");
      switch ( respond = getch() )
      {
        case 'Y':
        case 'y':
        case 'n':
        case 'N': break;
        default : gotoxy (1,22);
                  puts("Error charecter. ");
      }
    }
    while (!(respond == 'y' ||  respond == 'n'));
    ++iterate;
  }
  while ((iterate < 5) && (respond == 'y'));

  WriteImage(Image,ttl_base);
  setVGA(3);
}
```