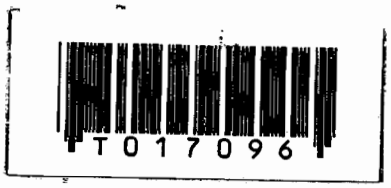
สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การลดข้อมูลภาพโดยการเข้ารหัสพื้นที่ของภาพ

IMAGE DATA COMPRESSION VIA SEGMENTED REGION CODING



สุชาติ ฉิมประดิษฐ

SUCHAT CHIMPRADID

อาจารย์ที่ปรึกษา

รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์

Assoc.Prof.Dr.FUSAK CHEEVASUVIT

เลขหมู่

เลขทะเบียน 17096

วัน, เดือน, ปี 5 ก.พ. 2535

วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์มหาบัณฑิต

สาขาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์

บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2534

ISBN 974-8157-08-3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทคัดย่อ

การลดข้อมูลภาพก็คือการลดจำนวนของข้อมูลของภาพให้มีขนาดเล็กลง แต่ยังคงรักษารายละเอียดและความคมชัดของภาพเอาไว้ให้ได้มากที่สุด เทคนิคการลดข้อมูลภาพสามารถกระทำได้โดยการเข้ารหัสพื้นที่ของภาพที่นำเสนอในวิทยานิพนธ์นี้มีคุณสมบัติเด่นคือ ที่อัตราบิทเรทสูง ๆ จะสามารถรักษาความคมชัดของบริเวณที่เป็นขอบของส่วนต่าง ๆ ในภาพไว้ได้ดีกว่า เมื่อเทียบกับวิธี Discrete Cosine Transform (DCT)

การลดข้อมูลภาพโดยการเข้ารหัสพื้นที่ของภาพนี้ ภาพที่จะนำมาเข้ารหัสจะต้องทำอิมเมจเซกเม้นเตชั่นให้ได้เป็นพื้นที่ส่วนย่อย ๆ เสียก่อน วิธีการทำอิมเมจเซกเม้นเตชั่นที่นำมาใช้นั้นอาศัยหลักการพื้นฐานทางทฤษฎีกราฟ ที่เรียกว่ารีเคอร์ซีฟช็อตเตสท์สแพนนิ่งทรี (RSST) วิธีดังกล่าวจะใช้ข้อมูลส่วนใหญ่ของภาพ (Global Data) เป็นตัวตัดสินการแบ่งแยกพื้นที่ในภาพ เมื่อได้ภาพจากการทำอิมเมจเซกเม้นเตชั่นแล้วก็จะนำไปเข้ารหัส ซึ่งรหัสที่ได้จะประกอบด้วยส่วนสำคัญสามส่วนคือ ตำแหน่งจุดเริ่มต้นของขอบ ค่ารหัสขอบ และค่าความเข้ม (gray level) ของแต่ละส่วน โดยที่ทิศทางของขอบจะเป็นการเข้ารหัสแบบลูกโซ่ของฟรีแมน (Freeman's chain code) นอกจากนี้ยังได้ปรับปรุงวิธีการเข้ารหัสขอบให้เหลือรหัสน้อยลงด้วยวิธีการใช้รหัสของความยาว (run length code) จากส่วนสำคัญทั้งสามนี้จะเป็นรายละเอียดของภาพทั้งภาพที่ผ่านการเข้ารหัส และในกรณีที่ต้องการส่งภาพไปในระยะทางไกล ๆ โดยผ่านทางช่องสัญญาณความเร็วต่ำนั้น สามารถที่จะกระทำได้โดยส่งข้อมูลที่ได้จากการเข้ารหัสนี้ไปยังปลายทาง และข้อมูลภาพนี้เมื่อทำการแปลงกลับที่ปลายทาง ก็จะได้ภาพที่มีรายละเอียดของขอบภาพที่สมบูรณ์ ซึ่งการส่งแบบนี้จะใช้เวลาน้อยกว่าการส่งข้อมูลภาพเดิมที่ไม่มีการลดข้อมูลของภาพมาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ABSTRACT

Image data compression is the technique to reduce the number of image data ,but still preserves the most detail and the sharpness of the image. Image Data Compression Via Image Segmented Coding is presented in this thesis. The advantage of our method when compares with Discrete Cosine Transform (DCT) at higher bit rate is the more sharpness of region or the edge of image objects.

Image Data Compression Via Image Segmented Coding is the method that used recursive shortest spanning tree (RSST) based on graph theory as image segmentation algorithm to decompose image into several region. The decomposed image then encoded, and within each region there are three kinds of code. Which are a list of edge start tracing coordinates, the edge description and gray level information. The edge regions are encoded by Freeman's chain code and run length of such edge. The segmented image can be transmitted over low capacity transmission channels. With this technique, a lot of transmitting time can be saved .

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ขอขอบพระคุณ รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์ เป็นอย่างสูง ที่ได้ให้การประสิทธิ์ประสาทวิชา ให้คำแนะนำปรึกษาในเรื่องต่าง ๆ แก่ผู้เขียน และขอขอบพระคุณอาจารย์อิทธิชัย อรุณศรีแสงไชย รศ.ถวิล พึ่งมา ที่ได้ช่วยเหลือ และให้คำแนะนำต่างๆ และขอขอบคุณเพื่อนๆ ที่ได้ช่วยเหลือ และเป็นกำลังใจ ในการทำวิทยานิพนธ์นี้จนสำเร็จลุล่วงได้ด้วยดี

สุชาติ ฉิมประดิษฐ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ในยุคปัจจุบันคอมพิวเตอร์(computer) ได้เข้ามามีบทบาทอย่างมากต่อชีวิตความเป็นอยู่แทบจะทุก ๆ ด้านก็ว่าได้ คอมพิวเตอร์จะถูกนำไปใช้ในสาขาวิชาชีพต่าง ๆ เช่น ด้านวิศวกรรม ด้านการแพทย์ และงานอุตสาหกรรมแขนงต่าง ๆ โดยเฉพาะงานทางด้านวิศวกรรมนั้น คอมพิวเตอร์จะถูกนำมาใช้แทบทุกสาขาไม่ว่าจะเป็นด้านการสื่อสาร งานด้านการออกแบบโครงสร้าง งานประมวลผลภาพของ remote sensing ตลอดไปถึงงานเอกสารในสำนักงานที่มีการติดต่อเชื่อมโยงเครือข่ายของข้อมูลเป็นระบบที่เรียกว่า office automation และในปัจจุบันได้เริ่มเข้าสู่ยุคของการแข่งขันทางเทคโนโลยีข่าวสาร ความสามารถของคอมพิวเตอร์ถูกวัดด้วยความเร็วในการประมวลผล และเวลาที่ใช้ในการติดต่อแลกเปลี่ยนข้อมูลข่าวสารระหว่างเครื่องและแหล่งข้อมูล

ข้อมูลข่าวสารต่างๆ ที่สำคัญนั้นนอกจากจะอยู่ในรูปของเสียง เอกสาร และสัญลักษณ์ต่างๆ แล้ว ข้อมูลอีกอย่างหนึ่งที่มีความสำคัญไม่ยิ่งหย่อนไปกว่ากันนั้นก็คือภาพ โดยที่ภาพนั้นอาจจะอยู่ในรูปของภาพถ่าย ภาพทางจอโทรทัศน์ และข้อมูลภาพอื่น ๆ ที่แสดงทางจอภาพ (monitor) ข้อมูลภาพเหล่านี้สามารถนำมาใช้ประโยชน์ได้อย่างกว้างขวาง เป็นต้นว่า ข้อมูลภาพถ่ายทางดาวเทียมได้นำมาใช้ในการพัฒนาและการอนุรักษ์ทรัพยากรธรรมชาติ สภาพแวดล้อมของโลก และการสำรวจหรือพยากรณ์ในระยะทางไกล(remote sensing)อื่น ๆ นอกจากนี้ยังมีการนำข้อมูลภาพไปใช้ในระบบฐานข้อมูลที่เป็นแฟ้มประวัติบุคคล เพื่อสามารถตรวจสอบได้ทั้งประวัติและหน้าตาของผู้เป็นเจ้าของประวัตินั้น และตัวอย่างการนำภาพมาใช้อีกสาขาหนึ่งที่กำลังได้รับความสนใจอยู่ในขณะนี้คือ ระบบโทรศัพท์ที่สามารถเห็นภาพของคู่สนทนา (video phone) ซึ่งอาจจะเข้ามาแทนที่โทรศัพท์ระบบเดิมในอีกไม่ช้าก็เป็นได้ โดยมากภาพที่นำมาใช้ในงานต่าง ๆ ที่กล่าวมาแล้วนั้นจะอยู่ในรูปของข้อมูลทางดิจิตอล การที่จะให้ภาพแต่ละภาพมีรายละเอียดหรือความคมชัดเพียงพอต่อการใช้งานนั้น จะต้องใช้หน่วยความจำเป็นจำนวนมากสำหรับการเก็บข้อมูล และในกรณีที่ระบบมีการรับส่งข้อมูลภาพด้วยแล้ว จะทำให้สิ้นเปลืองเวลาที่ใช้ไปในการรับส่งข้อมูลภาพแต่ละภาพ เนื่องจากข้อมูลมีขนาดใหญ่ จากปัญหาดังกล่าวจึงเป็นที่มาของวิทยานิพนธ์ฉบับนี้ ดังรายละเอียดในหัวข้อต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.1 วัตถุประสงค์ของวิทยานิพนธ์

เทคนิคการลดขนาดข้อมูลภาพ เป็นวิธีการที่นำมาใช้ในการลดขนาดของหน่วยความจำที่ใช้ในการเก็บภาพ และยังสามารถที่จะลดเวลาที่ต้องสูญเสียไปในการส่งรับข้อมูลภาพเหล่านี้ไปยังปลายทางที่ไกลออกไป โดยผ่านทางช่องการสื่อสารความเร็วต่ำ(low speed communication channel) อย่างเช่น การส่งภาพผ่านทางสายโทรศัพท์ซึ่งเป็นระบบที่มีความเร็วต่ำ เมื่อทำการลดข้อมูลแล้วเวลาที่ใช้ในการรับส่งข้อมูลภาพแต่ละภาพก็จะน้อยลงไปด้วย เปรียบเสมือนว่าเป็นการเพิ่มอัตราบิทเรท(bit rate) ในการส่งข้อมูลภาพให้สูงขึ้นนั่นเอง

การลดข้อมูลภาพโดยการเข้ารหัสพื้นที่ของภาพที่เสนอในวิทยานิพนธ์ฉบับนี้ สามารถนำไปประยุกต์ใช้กับการสร้างฐานข้อมูลภาพ และโดยเฉพาะการรับส่งภาพในระยะไกลโดยผ่านโครงข่ายโทรศัพท์สาธารณะ ซึ่งเมื่อพัฒนาต่อไปทางด้านฮาร์ดแวร์และซอร์ฟแวร์สนับสนุนการทำงานของระบบ สามารถที่จะเป็นต้นแบบของโทรศัพท์เห็นภาพ (video phone) หรือการประชุมระยะไกลผ่านทางสาย (video conferencing) ได้

1.2 ขอบเขตการวิจัย

ในส่วนของการวิจัยได้เสนอเทคนิคการเข้ารหัสของวิธีต่าง ๆ เพื่อที่จะสามารถลดขนาดของข้อมูลให้ได้มากที่สุด รายละเอียดการวิจัยของวิทยานิพนธ์ฉบับนี้ จะแบ่งย่อยออกเป็น 6 บท โดยที่แต่ละบทมีหัวข้อและเนื้อหาดังต่อไปนี้

บทที่ 1. เป็นบทนำ กล่าวถึงวัตถุประสงค์ และขอบเขตของวิทยานิพนธ์

บทที่ 2. เป็นวิธีการและหลักการเบื้องต้นของการลดข้อมูลภาพ เทคนิคต่าง ๆ ที่นิยมใช้ในกระบวนการลดข้อมูลภาพซึ่งมีวิธีการของ predictive coding, Transform coding โดยเฉพาะวิธีของ Discrete Cosine Transform (DCT), hybrid coding, หลักการเบื้องต้นของการลดข้อมูลภาพโดยการเข้ารหัสพื้นที่ของภาพที่จะเสนอต่อไป และรวมถึงเกณฑ์ที่ใช้ในการวัดความเหมือนจริงของภาพ

บทที่ 3. เป็นเทคนิคการกำจัดสัญญาณรบกวนที่อาจจะเกิดขึ้นได้กับข้อมูลภาพดิจิตอล โดยใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วงจรกรองมัธยฐานแบบที่สามารถปรับขนาดต่าง หน้าต่างได้อย่างอัตโนมัติ ซึ่งเป็นวิธีที่สามารถกำจัดสัญญาณรบกวนได้เป็นอย่างดี ทั้งยังรักษาความคมชัดของภาพได้ดีกว่าวิธีอื่นๆ

บทที่ 4. เป็นเทคนิคการทำเซกเม้นเตชั่นภาพ (image segmentation) ที่นำมาใช้ในกระบวนการเข้ารหัสภาพ โดยใช้ทฤษฎีกราฟ (graph theory)ที่เรียกว่า SST (shortest spanning tree) มาใช้ในการทำเซกเม้นต์ภาพ ซึ่งเป็นวิธีที่สามารถที่จะกำหนดจำนวนของเซกเม้นต์ในภาพได้ตามต้องการ รวมไปถึงการทำเซกเม้นต์ที่ใช้เทคนิคที่ปรับปรุงมาจาก SST ที่เรียกว่า RSST (recursive shortest spanning tree) เป็นการใช้ข้อมูลรวมของภาพในการทำเซกเม้นเตชั่น

บทที่ 5. เป็นวิธีการและเทคนิคในการเข้ารหัสภาพที่ได้มาจากการทำเซกเม้นเตชั่น โดยอาศัยเทคนิคที่เรียกว่า การเข้ารหัสบริเวณปิด (contour coding) เป็นส่วนๆ ไป โดยเก็บเฉพาะค่าจุดเริ่มต้น ค่ารหัสขอบ และค่าระดับสีเทาของแต่ละส่วน ต่อจากนั้นเป็นวิธีการการเข้ารหัสภาพที่สามารถลดจำนวนของจุดเริ่มต้นและจำนวนของรหัสขอบ ซึ่งเป็นวิธีที่ได้พัฒนามาจากวิธีแรก โดยการเก็บเฉพาะจุดเริ่มต้นของเซกเม้นต์ ที่ไม่เชื่อมต่อถึงกันกับเซกเม้นต์อื่นๆ และขอบจะไม่เข้ารหัสย้อนสวนทางกับส่วนที่เข้ารหัสไปแล้ว จึงเป็นวิธีที่สามารถลดจำนวนรหัสของภาพได้มากกว่าวิธีแรก

บทที่ 6. เป็นบทสรุป เปรียบเทียบผลที่ได้กับผลของการลดข้อมูลโดยวิธีของ DCT และวิจารณ์ถึงข้อดีและข้อเสียของเทคนิคการลดข้อมูลภาพโดยวิธีการเข้ารหัสพื้นที่ของภาพหรือการเข้ารหัสภาพเซกเม้นเตชั่นพร้อมทั้งเสนอแนะแนวทางการวิจัยที่สามารถที่จะพัฒนาต่อไปได้

และในส่วนสุดท้ายซึ่งเป็นภาคผนวก ได้ใส่รายละเอียดของโปรแกรมที่ใช้ในแต่ละบทเอาไว้ด้วยเพื่อความสดวกของผู้ที่จะค้นคว้าวิจัยต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
หลักการเบื้องต้นของการลดข้อมูลภาพ
(Principle of Image Data Compression)

สัญญาณภาพโดยธรรมชาติแล้วจะเป็นสัญญาณต่อเนื่อง (analog) ที่มีความกว้างของย่านความถี่ที่สูงมาก ซึ่งเมื่อเทียบกับสัญญาณเสียงแล้วความกว้างของย่านความถี่จะสูงกว่าของสัญญาณเสียงเป็น 1000 เท่าหรือมากกว่า การนำข้อมูลภาพมาใช้ในการประมวลผลในด้านต่าง ๆ ที่มีเครื่องคอมพิวเตอร์เป็นตัวประมวลผลแล้วนั้น สัญญาณภาพจะต้องแปลงให้เป็นสัญญาณที่เครื่องสามารถเข้าใจได้เสียก่อนนั้นคือต้องแปลงให้อยู่ในรูปของสัญญาณดิจิตอล ซึ่งอยู่ในลักษณะที่ไม่ต่อเนื่อง(discrete signal) โดยผ่านการสุ่มและจัดระดับ (sampled and quantised) เมื่อรายละเอียดของภาพ ๆ หนึ่ง ถูกแทนด้วยข้อมูลแบบดิจิตอล ขนาดของข้อมูลภาพจึงมีจำนวนสูงมากเพื่อที่จะสามารถเก็บรายละเอียดของภาพได้เพียงพอ ทำให้ต้องใช้ขนาดของหน่วยความจำในการเก็บข้อมูลเหล่านี้มีขนาดใหญ่ตามไปด้วย ในกรณีที่มีการรับส่งข้อมูลภาพเหล่านี้ด้วยแล้วจะต้องเสียเวลาในการรับส่งเป็นอันมาก

การลดขนาดของข้อมูลภาพ จึงถูกนำมาใช้ในการแก้ปัญหาเหล่านี้ เพื่อเป็นการประหยัดทางด้านเศรษฐกิจในส่วนของจำนวนหน่วยความจำที่ลดลง และในส่วนของเวลาที่ใช้ไปในการรับส่งข้อมูลภาพ สิ่งที่เป็นตัวกำหนดว่าข้อมูลของภาพสามารถที่จะลดลงไปได้เท่าไรนั้น ขึ้นอยู่กับงานที่ใช้ว่าต้องการรายละเอียดมากน้อยเพียงใด เมื่อทำการลดข้อมูลแล้วจึงจะสามารถนำข้อมูลเดิมกลับมา (reconstruct) ได้อย่างเหมาะสม ซึ่งงานแต่ละอย่างมีความต้องการรายละเอียดของภาพที่ไม่เท่ากัน อย่างเช่นถ้านำไปใช้ในการตรวจสอบและจดจำรูปแบบของวัตถุต่าง ๆ โดยใช้คอมพิวเตอร์แล้ว ภาพที่ใช้จะเน้นเฉพาะส่วนของขอบวัตถุในภาพนั้น ๆ ก็เพียงพอ โดยที่ไม่จำเป็นต้องมีรายละเอียดโครงสร้างภายในของภาพ ในขณะที่การลดข้อมูลภาพเพื่อการลดเวลาที่ใช้ไปในการรับส่งภาพผ่านช่องสัญญาณความเร็วต่ำอย่างเช่นระบบโครงข่ายโทรศัพท์สาธารณะนั้น มีความจำเป็นอย่างที่จะต้องเก็บรายละเอียดต่างๆ ของภาพให้ได้มากที่สุด ซึ่งเป็นจุดประสงค์หลักของวิทยานิพนธ์ฉบับนี้ หลักการพื้นฐานที่นิยมใช้ในการลดข้อมูลภาพมีอยู่สามหลักการด้วยกันคือ Predictive Coding ซึ่งเป็นการเข้ารหัสใน data domain หลักการที่สองคือ Transform Coding เป็นวิธีที่ประมวลผล (process) ในโดเมนของความถี่ ส่วนหลักการสุดท้ายเป็นการรวมเอาข้อดีต่างๆ จากสองหลักการแรกเข้าด้วยกัน เรียกว่า Hybrid coding

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1 Predictive Coding

การลดขนาดข้อมูลภาพด้วยวิธี Predictive Coding นี้ เป็นการอาศัยคุณสมบัติของข้อมูลภาพที่มักจะมีค่าซ้ำๆ กัน และเมื่อข้อมูลภาพอินพุทถูกกำหนดให้มีความเกี่ยวพันธ์กันนั้นก็คือ จุดภาพที่มีตำแหน่งอยู่ใกล้ๆ กัน มักจะมีค่าระดับแอมปลิจูดใกล้เคียงกันหรือเท่ากัน ดังนั้นจึงสามารถที่จะใช้ค่าของจุดภาพหนึ่งจุดหรือหลายๆ จุด ที่ผ่านมาในไลน์(line)นั้น ไลน์ก่อนหน้านั้น หรือในเฟรมที่ผ่านมา เป็นตัวคาดคะเนหรือแทนค่าจุดภาพจุดปัจจุบัน ซึ่งโดยธรรมชาติทางสถิติของข้อมูลภาพเราสามารถที่จะคาดคะเนค่าของข้อมูลได้ไม่ผิดพลาดมากนัก จากค่าที่ได้จากการคาดคะเนนี้เอาไปลบกับค่าจริงของจุดภาพ จะได้เป็นค่าความแตกต่างระหว่างค่าจริงกับค่าที่เราคาดคะเนเอาไว้ ซึ่งค่านี้จะมีขนาดเล็ก และค่าผลต่างนี้จะถูกนำไปเข้ารหัสเพื่อที่จะเก็บไว้ใช้ในตอนถอดรหัส พร้อมกับค่าที่เราคาดคะเนเอาไว้ ดังนั้นในการที่จะเก็บไว้ในหน่วยความจำหรือต้องการส่งก็ใช้ค่าสองค่านี้ เมื่อถึงตอนที่จะนำภาพเดิมกลับมาหรือตอนถอดรหัส จะนำเอาค่าที่เราคาดคะเนไว้ในในตอนแรกบวกกับค่าของผลต่างของจุดภาพนั้นกับค่าคาดคะเน ก็จะได้เป็นค่าของจุดภาพนั้นๆ ค่าผิดพลาดที่ได้ในตอนถอดรหัสเกิดขึ้นได้จากกรณีเดียวเท่านั้นคือ ตอนจัดระดับ (quantised) ค่าความแตกต่างของการเข้ารหัสเท่านั้น วิธีนี้เป็นวิธีที่ง่ายแก่การสร้างระบบ และสามารถลดขนาดของข้อมูลภาพให้เหลือประมาณ 1-2 bit/element [1]

2.2 Transform Coding

การลดข้อมูลภาพด้วยวิธีของ Transform Coding นี้เป็นวิธีที่ซับซ้อน และมีขั้นตอนมากกว่าวิธีการลดข้อมูลภาพโดย Predictive Coding หลักการของวิธี Transform Coding จะทำการแปลงข้อมูลอินพุทที่อยู่ในรูปของ data domain ให้อยู่ในรูปของ spectral หรือ frequency domain โดยใช้วิธีการ Transform แบบต่าง ๆ เช่น Fourier Transform แต่ในปัจจุบันนี้มีการใช้การ Transform ตัวอื่น ๆ ที่มีข้อดีและมีประสิทธิภาพในการเข้ารหัสสูง มีการคำนวณที่ง่ายกว่าอย่างเช่น คำนวณเฉพาะค่าจริงเท่านั้น การทำงานของ Transform จะเป็นการแปลงข้อมูลที่อยู่ในรูปของ spatial domain ให้อยู่ในรูปสัมประสิทธิ์ของพลังงานความถี่ โดยค่าความถี่ต่ำๆ จะมีพลังงานสูง ที่ความถี่สูงพลังงานจะลดลงไป การเข้ารหัสจึงใช้จำนวนบิทสำหรับแต่ละช่วงความถี่ไม่เท่ากัน เมื่อต้องการอัตราบิทเรทสูงๆ ค่าของพลังงานความถี่สูงจะถูกตัดทิ้งไปเป็นส่วนใหญ่ เป็นเหตุให้รายละเอียดส่วนที่เป็นขอบในภาพขาดหายไป ทำให้ภาพที่ได้เบลอ ขาดความคมชัด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การ Transform ที่นิยมนำมาใช้ในการลดข้อมูลภาพมีอยู่ด้วยกันหลายวิธี เช่น Fast Fourier Transform, Fast Walsh-Hadamard Tranform, Fast Slant Transform, Fast Discrete Cosine Transform, Fast Discrete Sine Transform เป็นต้น ซึ่งแต่ละวิธีก็มีข้อดีข้อเสียที่แตกต่างกันไป การ Transform ที่นิยมใช้กันมากได้แก่ Discrete Cosine Tranform เพราะเป็นวิธีที่สามารถคำนวณได้ง่ายเนื่องจากมีการคำนวณเฉพาะค่าจริง ไม่มีค่าจินตภาพ(imaginary)

วิธี Tranform Coding ถึงแม้ว่าจะเป็นวิธีที่ยุ่งยากแต่ก็สามารถสร้างระบบได้ด้วยอุปกรณ์ทางฮาร์ดแวร์(hardware)ดิจิตอลความเร็วสูงได้ง่าย และเป็นระบบที่ยืดหยุ่น (adaptive system) สามารถที่จะลดขนาดของข้อมูลให้อยู่ในอัตราบิทเรทประมาณ 0.5-1.0 บิทต่อจุดภาพ ซึ่งเป็นช่วงอัตราการลดที่สามารถสร้างภาพเดิมกลับมาได้สมบูรณ์เพียงพอต่อการนำไปใช้งานต่างๆ

2.3 Hybrid Coding

Hybrid Coding เป็นเทคนิคที่นำข้อดีของวิธี Predictive Coding และ Transform Coding มาใช้ร่วมกัน ทั้งนี้เพราะบางครั้งการลดขนาดข้อมูลภาพด้วย Transform Coding ไม่อาจจะให้รายละเอียดของภาพเพียงพอ ดังนั้นการเพิ่มรายละเอียดของภาพจึงต้องใช้ความสัมพันธ์ของจุดภาพที่มีอยู่ทั้งทางแนวนอนและทางแนวตั้ง โดยการใช้การ Transform แบบมิติเดียวในทิศทางแกนใดแกนหนึ่งของภาพ เช่น ใช้การ Transform กับข้อมูลภาพในทางแนวนอน ต่อจากนั้นจึงใช้ Predictive Coding กับประสิทธิ์ของการ Transform ที่ได้ เพื่อที่จะใช้ในการคาดคะเนค่าของกลุ่มสัมประสิทธิ์ที่เหมือน ๆ กัน ที่ตามมาในทางแนวนอน ในกรณีนี้ความสัมพันธ์ของข้อมูลในทางแนวตั้งจะเป็นตัวกำหนดผลที่ได้ว่าถูกต้องมากน้อยเพียงใด แต่โดยเฉลี่ยแล้วค่าสัมประสิทธิ์ที่ตามมามักจะมีค่าที่เหมือน ๆ กัน ซึ่งเหมือนกับการทำ Predictive ใน data domain นั้นเอง และในกรณีของการ Transform แบบสองมิติสามารถทำได้เร็วขึ้น อาจจะใช้การ Predictive กับค่าสัมประสิทธิ์ของการ Transform ของภาพบริเวณเดียวกันแต่เป็นเฟรมที่แตกต่างกันที่ตามมา การใช้ Hybrid Coding ในลักษณะนี้สามารถใช้สำหรับภาพเคลื่อนไหวได้ ความจริงแล้ววิธีการของ Hybrid Coding นี้สามารถเป็นไปได้อีกลักษณะหนึ่งคือ ใช้การ Predictive ในขั้นตอนแรกก่อนแล้วจึงทำการ Transform แต่ในกรณีนี้ต้องการระบบที่ซับซ้อนมากกว่า

การลดข้มูลภาพด้วยวิธีของ Hybrid Coding นี้ อาจจะพูดได้ว่ามีคุณสมบัติอยู่ระหว่างวิธีของ Predictive Coding และ Transform Coding ดังนั้นอัตราบิตเรทต่ำสุดของวิธีนี้จึง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไม่สามารถที่จะทำให้ได้ต่ำกว่าวิธีของ Transform Coding เพียงแต่สามารถสร้างได้ง่ายกว่า โดยอัตราบิทเรตของระบบจะมีค่าประมาณ 1 บิทต่อจุดภาพ ซึ่งเป็นอัตราบิทเรทที่สามารถสร้างภาพเดิมกลับมาใหม่ (reconstruction) ได้ภาพที่มีคุณภาพดีพอสมควร

2.4 เกณฑ์การวัดความเหมือนจริงของภาพ (Image Fidelity)

ในการลดข้อมูลภาพนั้น จะมีข้อมูลส่วนหนึ่งที่เกิดผิดพลาดหรือสูญเสียไป ข้อผิดพลาดที่เกิดขึ้นนี้จะมีผลในตอนที่สร้างภาพกลับคืนมา(reconstruction) และค่าความผิดพลาดนี้จะอยู่ในช่วงหนึ่งที่สามารถยอมรับได้ ดังนั้นเกณฑ์การวัดความเหมือนจริงของภาพสามารถนำมาใช้ในการวัดประสิทธิภาพของระบบได้ ตัวอย่างเกณฑ์ที่นิยมใช้ในการวัดคุณภาพของภาพคือ ค่า root-mean-square (rms) ของ error ระหว่างข้อมูลภาพอินพุทและข้อมูลภาพเอ้าท์พุท นอกจากนี้ยังมีค่า rms ของอัตราส่วนของสัญญาณต่อสัญญาณรบกวนของภาพเอ้าท์พุท (Signal-to-Noise Ratio) เมื่อกำหนดให้ข้อมูลภาพอินพุทประกอบด้วยอาเรย์ขนาด N x N ของจุดภาพ f(x,y) โดยที่ x, y มีค่าเป็น 0,1,...N-1 และแต่ละจุดภาพมีค่าของระดับสีเทาที่เป็นไปได้คือ 2^m เมื่อ m เป็นจำนวนบิทของเลขไบนารี

สำหรับทุกค่าของ x และ y ในช่วง 0,1,...N-1 ค่า error ระหว่างจุดภาพอินพุทและเอ้าท์พุทคือ

$$e(x,y) = g(x,y) - f(x,y) \qquad (2.1)$$

เมื่อ f(x,y) คือ ค่า input image ณ. จุด x, y ใด ๆ

g(x,y) คือ ค่า output image ณ. จุด x, y ใด ๆ

ค่าเฉลี่ยของ error กำลังสองของภาพ (mean square error) คือ

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

$$= \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \left[g(x,y) - f(x,y) \right]^2 \qquad (2.2)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังนั้นค่า rms ของ error จึงสามารถเขียนได้ดังนี้ คือ

$$e_{rms} = [e^2]^{1/2} \tag{2.3}$$

ค่า root mean square error เป็นค่าที่ใช้ในการวัดความแตกต่างของข้อมูลอินพุทกับข้อมูลเอ้าท์พุท แต่เมื่อพิจารณาขนาดของข้อมูลเอ้าท์พุทต่อขนาดของสัญญาณรบกวน (noise) ก็จะได้เป็นค่า Signal-to-Noise Ratio (SNR) เมื่อกำหนดให้สัญญาณภาพเอ้าท์พุทแต่ละจุดประกอบด้วยสัญญาณอินพุทบวกด้วยค่าสัญญาณรบกวน นั้นคือ

$$g(x,y) = f(x,y) + e(x,y) \tag{2.4}$$

ดังนั้นค่า mean square Signal-to-Noise ของข้อมูลภาพเอ้าท์พุท สามารถหาได้โดยค่าเฉลี่ยของสัญญาณอินพุทกำลังสอง $g^2(x,y)$ หารด้วยค่าเฉลี่ยของสัญญาณรบกวนกำลังสอง $e^2(x,y)$ ของข้อมูลภาพทั้งหมด ซึ่งสามารถเขียนได้ดังนี้

$$(SNR)_{ms} = \frac{\frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} g^2(x,y)}{\frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} e^2(x,y)} \tag{2.5}$$

ค่า rms ของ SNR จึงสามารถเขียนได้ดังต่อไปนี้

$$(SNR)_{rms} = \left[\frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} g^2(x,y) \Big/ \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} [g(x,y)-f(x,y)]^2 \right]^{1/2} \tag{2.6}$$

โดยที่เทอมส่วนของสมการข้างบนเป็นสมการของสัญญาณรบกวนที่อยู่ในรูปของผลต่างระหว่างข้อมูลอินพุทกับข้อมูลเอ้าท์พุท

จากวิธีการที่ใช้ในการวัดความเหมือนจริงของภาพที่กล่าวมาข้างบนนี้ ไม่สามารถที่จะบ่งบอกหรือใช้เป็นเกณฑ์ในการพิจารณาได้แต่เพียงอย่างเดียว ในกรณีของภาพเอ้าท์พุทที่ได้จากการประมวลผลหรือรับส่งสัญญาณ โดยมีสายตาของมนุษย์เป็นตัวรับภาพจากจอภาพอีกทีหนึ่งอย่างเช่น broadcast TV, picture phone, หรืองานต่างที่มีการใช้ image processing เป็นต้น ระบบการมองเห็นของมนุษย์จะมีลักษณะคุณสมบัติพิเศษกล่าวคือ ระบบการมองเห็นของตาจะไวต่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ความเข้มของแสงในลักษณะของ logarithmic ดังนั้น error ในบริเวณที่เป็นที่มืดของภาพจะเห็นได้ชัดกว่า error ที่อยู่ในบริเวณที่สว่าง และระบบการมองเห็นยังไวต่อการเปลี่ยนแปลงอย่างทันทีทันใดของระดับสีเท่าด้วย error ที่อยู่บนขอบหรือใกล้ ๆ ขอบของวัตถุจะมีผลต่อการมองเห็นมากกว่า error ที่อยู่ในโครงสร้างที่เป็นฉากหลังของภาพ ด้วยเหตุนี้ถึงแม้ว่าภาพสองภาพจะมีค่าของ rms error เท่ากัน แต่อาจจะปรากฏความแตกต่างของคุณภาพของการมองเห็นที่แตกต่างกันได้

2.5 Discrete Cosine Transform coding

Discrete Cosine Transform Coding เป็นวิธีการที่ใช้ในการลดข้อมูลภาพที่ได้รับความนิยมอย่างแพร่หลาย ทั้งนี้เพราะค่าสัมประสิทธิ์ในโดเมนของความถี่ที่ได้จะเป็นเทอมของค่าจริง(real time)เท่านั้น อีกทั้งยังสามารถคำนวณได้แบบรวดเร็ว(fast algorithms) และยังสามารถสร้างใช้งานจริงในลักษณะ real time โดยใช้ฮาร์ดแวร์ได้ไม่ยาก ในปัจจุบันหลักการของ Discrete Cosine Transform ยังคงมีการวิจัยกันอยู่ต่อไปเพื่อให้สามารถลดขนาดของข้อมูลให้ได้มากที่สุดพร้อมทั้งหาวิธีที่จะเพิ่มความเร็วในการคำนวณให้ได้เร็วขึ้นไปอีก

ดังนั้นในที่นี้จึงยกตัวอย่างระบบการลดข้อมูลภาพโดยใช้เทคนิคของ Discrete Cosine Transform Coding โดยมีโครงสร้างของระบบเป็นดังรูปที่ 2.1 จาก block diagram ของการ Transform coder ภาพอินพุทที่เข้ามาจะถูกแยกออกเป็นบล็อคเล็ก ๆ โดยเราสามารถกำหนดขนาดของบลอ็คได้ว่าให้เป็นเท่าไร ขนาดของบล็อคที่เหมาะสมจะเป็นตัวเพิ่มประสิทธิภาพในการรวมพลังงานของการ Trannsform ซึ่งจะทำให้ภาพที่ได้มีรายละเอียดที่ดี ความเหมาะสมของขนาดบล็อคค่าหนึ่ง ๆ จะเหมาะสมที่ค่าบิทเรตค่าหนึ่ง ๆ แต่อย่างไรก็ตามขนาดของบล็อคที่เหมาะ สมสามารถคำนวณได้ด้วยคณิตศาสตร์ที่ยุ่งยากพอ ๆ กับการ Transform เลยทีเดียว โดยส่วนมากแล้วขนาดของบล็อกจะมีค่าเป็นเลขยกกำลังของสองอย่าง เช่น 4, 8, 16 ซึ่งได้มาจาก 2^2, 2^3, 2^4 เป็นต้น หลังจากการแยกข้อมูลออกเป็นบล็อคย่อย ๆ แล้ว ก็จะทำการ Transform ไปโดยอิสระของแต่ละบล็อค โดยมีสมการของการ Transform ทั้งแบบ มิติเดียว และสองมิติ ดังต่อไปนี้เป็น

$$F(k) = \sqrt{\frac{2}{N}}\,\alpha(k)\sum_{n=0}^{N-1} f(n)\cos\left[\frac{(2n+1)\pi k}{2N}\right] \quad \text{เมื่อ} \quad k = 0,1,\ldots,N-1 \qquad (2.7)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อ

$$\alpha(0) = \sqrt{\frac{1}{2}} \quad \text{และ} \quad \alpha(k) = 1 \text{ เมื่อ } k \text{ ไม่เท่ากับ } 0 \tag{2.8}$$

F(k) เป็นผลที่ได้จากการ Transform และ f(n) เป็นข้อมูลอินพุทตัวที่ n

ส่วนสมการของการ Transform แบบสองมิติสามารถเขียนได้ดังนี้คือ

$$F(u,v) = \frac{2}{N}\alpha(u)\alpha(v)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f(m,n)\cos\left[\frac{(2m+1)u\pi}{2N}\right]\cos\left[\frac{(2n+1)v\pi}{2N}\right] \tag{2.9}$$

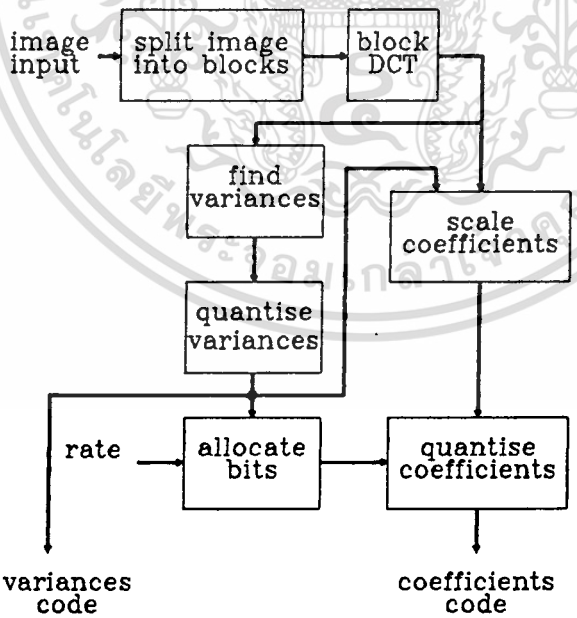
เมื่อ

u,v เป็นตัวแปรของ discrete frequency มีค่าเป็น 0,1,2,....,N-1

f(m,n) เป็นตำแหน่งของจุดภาพภายในบล็อกขนาด N x N (0,1,2,....,N-1)

F(u,v) เป็นผลจากการทำ discrete cosine Transform

$\alpha(0) = \sqrt{\frac{1}{2}}$ และ $\alpha(j) = 1$ เมื่อ j ไม่เท่ากับ 0



รูปที่ 2.1 บล็อคไดอะแกรม ตอน Forward Transform ของ DCT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลังจากผ่านการ Transform โดยใช้ Discrete Cosine Transform เพื่อเปลี่ยนข้อมูลให้อยู่ในรูปของ frequency domain พลังงานรวมส่วนใหญ่ของข้อมูลจะถูกเก็บ (pack) อยู่ในสัมประสิทธิ์เพียงไม่กี่ตัวของการ Transform จากสัมประสิทธิ์ที่ได้จะถูกนำมาหาค่า variances ของแต่ละบล็อค เพื่อที่จะนำค่า standard deviation ไปคำนวณหา bit allocate สำหรับจัดให้กับสัมประสิทธิ์แต่ละตัวในบล็อค และนอกจากนั้นยังส่งค่า standard deviation ไปยังทางด้านรับด้วยเพื่อใช้ในตอน decode ดังนั้นจึงต้องทำการจัดระดับ (quantise) ค่านี้ก่อนที่จะเก็บหรือส่งไปยังด้านรับ และค่าสัมประสิทธิ์ตัวอื่น ๆ ก็จะถูกปรับระดับ (normalise) ด้วยค่าของ variances ที่ได้หลังจากการกำหนด bit allocate เพื่อปรับค่าของสัมประสิทธิ์ให้อยู่ในช่วงของค่าบิทเรทที่กำหนด ก่อนที่จะทำการ quantise ส่วนการกำหนดค่า bit allocate ของสัมประสิทธิ์ที่ตำแหน่ง (u,v) หลังการ Transform ในแต่ละบล็อคคือ

$$b_{uv} = R + \frac{1}{2}\log_2\left[\frac{\sigma_{uv}^2}{\left[\prod_{k=0}^{n-1}\prod_{l=0}^{n-1}\sigma_{kl}^2\right]^{1/N}}\right]$$

$$= \log_2(\sigma_{uv}) + R - \frac{1}{N}\log_2\left[\prod_{k=0}^{n-1}\prod_{l=0}^{n-1}\sigma_{kl}\right]$$

$$= \log_2(\sigma_{uv}) + \beta \qquad (2.10)$$

เมื่อ σ_{uv} คือค่า standard deviation ของสัมประสิทธิ์ของการ Transform ที่ความถี่ (u,v) และ σ_{kl} เป็นค่า standard deviation ของสัมประสิทธิ์ของการ Transform ที่ตำแหน่ง kl โดย k และ l มีค่าเป็น 0 ถึง n-1 เมื่อ n คือขนาดของบล็อค และ N คือจำนวนของสัมประสิทธิ์ในแต่ละบล็อค โดยค่าคงที่ β กำหนดได้จาก

$$\beta = R - \frac{1}{N}\log_2\left[\prod_{k=0}^{n-1}\prod_{l=0}^{n-1}\sigma_{kl}\right]$$

และค่าสูงสุดของบิทที่จะกำหนดให้กับสัมประสิทธิ์แต่ละตัวจะกำหนดให้อยู่ภายใต้ค่า b_{max} $0 \leqslant b_{uv} \leqslant b_{max}$ และค่า b_{uv} เป็นค่าจำนวนเต็ม

โดยที่ $\sum_u \sum_v b_{uv} = B$ และ $B = R$ เมื่อ R คือ ค่าบิทเรทที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลังจากการคำนวน bit allocation ค่าสัมประสิทธิ์ของการ Transform แต่ละตัวในบล็อค จะถูกปรับค่า(normalise) ให้อยู่ในช่วงขนาดของของค่าไม่เกินจำนวนบิทที่กำหนด แล้วจึงทำการ quantise ค่าของสัมประสิทธิ์แต่ละตัว โดยอาศัยคุณสมบัติการกระจายพลังงานของสัมประสิทธิ์ของการ Transform ที่กล่าวไว้ว่าพลังงานส่วนใหญ่จะอยู่ที่ช่วงความถี่ต่ำ ๆ หรืออาจนำรูปแบบการกระจายความหนาแน่นแบบ Gaussian หรือ Laplacian มาใช้ก็ได้ ดังนั้นสัมประสิทธิ์แต่ละตัวจะถูกกำหนดด้วยจำนวนบิทที่ไม่เท่ากันอย่างเช่น ตัวอย่างในรูปที่ 2.2 กลุ่มสัมประสิทธิ์ที่อยู่บนมุมซ้ายบน เป็นกลุ่มของความถี่ต่ำซึ่งถือว่าสำคัญจะถูกกำหนดด้วยจำนวนบิทสูง ในขณะที่กลุ่มสัมประสิทธิ์ที่อยู่มุมขวาล่างเป็นกลุ่มของความถี่สูงซึ่งถือว่าไม่สำคัญจะถูกกำหนดด้วยจำนวนบิทต่ำ ๆ

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

รูปที่ 2.2 ตัวอย่างการกำหนดบิทให้กับสัมประสิทธิ์ในโดเมนความถี่หลังการแปลงด้วย DCT กับบล็อคขนาด 16x16 ซึ่งสามารถลดข้อมูลลงได้จาก 8 บิทต่อจุดภาพให้เหลือ 1.5 บิทต่อจุดภาพ

สำหรับขั้นตอนที่จะนำภาพกลับคืนมาหลังจากที่ได้เข้ารหัสไว้แล้ว สิ่งที่ต้องนำมาใช้ก็คือค่าของสัมประสิทธิ์ และค่าของ variances ของการ Transform โดยที่ค่าของ variances ถูกนำมาคำนวนค่า bit allocation จากค่าบิทเรทที่กำหนด แล้วจึงนำค่าที่ได้นี้ไปใช้ในการคำนวณปรับค่าของสัมประสิทธิ์ที่ได้จากการเข้ารหัสในตอนแรก เพื่อให้ได้ค่าสัมประสิทธิ์เดิมกลับมา (rescale coefficients) เพื่อจะได้ทำ Inverse Transform ต่อไป โดยมีสมการของการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Inverse Transform เป็นดังนี้

กรณี one dimensional Inverse Transform

$$f(n) = \sqrt{\frac{2}{N}} \sum_{k}^{N-1} \sigma(k)F(k)\cos\left[\frac{(2n+1)\pi k}{2N}\right] \quad \text{เมื่อ } k = 0,1,\ldots,N-1 \qquad (2.11)$$

เมื่อ f(n) เป็นผลที่ได้จากการทำ Inverse Transform กับ F(k) ซึ่งเป็นสัมประสิทธิ์จากการ Transform

กรณี two dimensional Inverse Transform

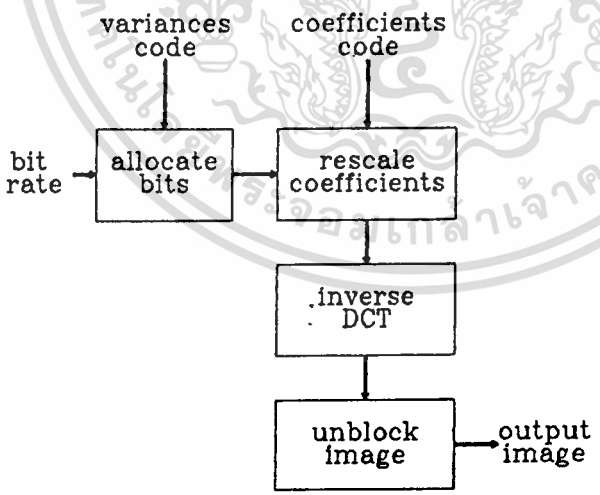
$$f(m,n) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v)F(u,v)\cos\left[\frac{(2m+1)u\pi}{2N}\right]\cos\left[\frac{(2n+1)v\pi}{2N}\right] \qquad (2.12)$$

เมื่อ

m,n เป็นตำแหน่งของจุดภาพ มีค่าตั้งเป็น 0,1,2,....,N-1

F(u,v) เป็นสัมประสิทธิ์ที่ได้จากการ Transform ภาพขนาด N x N

f(m,n) เป็นผลลัพธ์ของการทำ inverse Transform



รูปที่ 2.3 แสดงบล็อคไดอะแกรมของ DCT decoding

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถึงแม้ว่าการลดข้อมูลภาพโดยใช้หลักการของการ Transform จะเป็นที่ยอมรับและได้รับความนิยมแพร่หลายมาจนทุกวันนี้เพราะสามารถสร้างได้ง่าย ทั้งการใช้ software และการใช้ hardware อีกทั้งยังสามารถสร้างในลักษณะของ real time ได้ด้วย แต่ค่าของบิทเรทที่ใช้จะอยู่ในช่วง 1 ถึง 0.5 เพราะถ้าต่ำกว่านี้คุณภาพของภาพที่ได้จะแย่มาก คือเป็นภาพที่ขาดรายละเอียด ส่วนของขอบภาพจะเบลอไม่มีความคมชัด ทั้งนี้เนื่องจากการแปลง data domain ไปอยู่ในรูปของ frequency domain นั้น ที่ช่วงความถี่สูง ๆ ซึ่งเป็นส่วนบริเวณขอบต่าง ๆ ในภาพมักจะถูกตัดทิ้งไปเพื่อให้ขนาดของข้อมูลลดลง จึงทำให้ขอบต่างๆ ในภาพไม่คมชัด และภาพที่ได้ยังมีลักษณะเป็นบล็อค ๆ ตามขนาดของบล็อคที่ถูกแบ่งในตอน Transform

รูปที่ 2.4 แสดงผลของ DCT ที่อัตราบิทเรท 0.5 bits/pixel

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

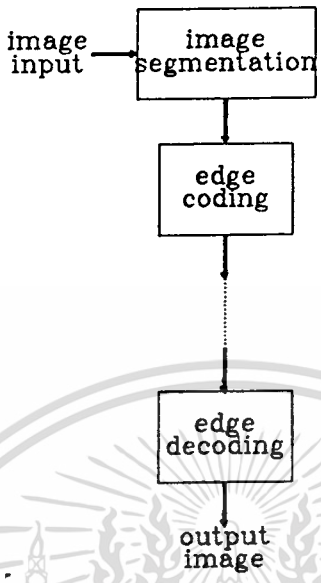
รูปที่ 2.5 แสดงผลของ DCT ที่อัตราบิทเรท 0.3 bits/pixel

2.6 การลดข้อมูลภาพโดยการเข้ารหัสพื้นที่ของภาพ

วิธีการลดข้อมูลภาพโดยการเข้ารหัสพื้นที่ของภาพหรือการเข้ารหัสภาพเซกเม้นเตชั่น มีหลักการคล้ายกับการเข้ารหัสแบบ predictive coding กล่าวคืออาศัยคุณสมบัติที่ว่าจุดภาพที่อยู่ในบริเวณใกล้เคียงกันมักจะมีลักษณะที่คล้ายกัน ซึ่งวิธีนี้สามารถที่จะลดขนาดของข้อมูลให้เหลืออัตราบิทเรทที่ต่ำกว่า 0.5 bits/pixel โดยที่ยังคงสามารถรักษาหรือเน้นความคมชัดของขอบภาพซึ่งเป็นส่วนสำคัญของภาพที่มีผลต่อการมองเห็นได้ดีกว่าการลดข้อมูลโดยการใช้ Transform coding

หลักการของการลดข้อมูลภาพโดยการเข้ารหัสภาพเซกเม้นต์ อาศัยเทคนิคของ image processing ที่เรียกกันว่า image segmentation ในขั้นตอนการแบ่งส่วนของภาพเพื่อที่จะเน้นส่วนที่สำคัญในภาพให้เด่นชัดออกมาเป็นส่วน ๆ และใช้ผลที่ได้จากการแบ่งส่วนมาใช้ในขั้นตอนของการเข้ารหัส และเทคนิคการทำเซกเม้นเตชั่นที่นำมาใช้เป็นวิธีที่ใช้พื้นฐานของทฤษฎีกราฟที่เรียกว่า shortest spanning tree ซึ่งสามารถแบ่งรายละเอียดของภาพแต่ละส่วนให้สอดคล้องกับรายละเอียดของภาพทั้งภาพได้ดี เมื่อได้ภาพหลังจากการทำเซกเม้นเตชั่นแล้วจะนำไปเข้ารหัสส่วนของภาพแต่ละส่วน โดยจะทำการเข้ารหัสและเก็บเฉพาะส่วนที่เป็นขอบ พร้อมทั้งตำแหน่งจุดเริ่มต้นและค่าระดับสีเทาของแต่ละส่วนเท่านั้น รูปที่ 2.6 เป็นบล็อคไดอะแกรมของระบบการลดข้อมูลภาพที่เสนอต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.6 แสดงบล็อคไดอะแกรม การลดข้อมูลภาพโดยการเข้ารหัสภาพเซกเม้นต์

จากรหัสที่เก็บเอาไว้เมื่อต้องการนำภาพเดิมกลับมาหรือการถอดรหัส จะใช้รหัสของขอบโดยเริ่มจากจุดเริ่มต้น สร้างขอบขึ้นมาก่อนบนอาเรย์ของภาพ แล้วจึงทำการใส่ค่าระดับความเข้มของส่วนที่ได้สร้างขอบขึ้นมาจนครบทุกส่วน รายละเอียดของการทำเซกเม้นเตชั่น การเข้ารหัสและการถอดรหัสนั้นจะกล่าวถึงในบทต่อ ๆ ไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

บทที่ 3

การกำจัดสัญญาณรบกวนด้วยวงจรกรองมัธยฐานที่ปรับขนาดได้อย่างอัตโนมัติ

(Noise Removing by Median Filter With Auto-adaptive Length)

ลักษณะการได้มาของข้อมูลภาพทางดิจิตอลมีวิธีหลักๆ ที่สำคัญคือการใช้อุปกรณ์พวกสแกนเนอร์สแกนจากภาพถ่าย และวิธีที่สองก็คือการใช้อุปกรณ์พวกดิจิไตเซอร์ที่ใช้หลักการของการเปลี่ยนสัญญาณภาพที่ได้จากกล้องวีดีโอให้เป็นข้อมูลทางดิจิตอล โดยการถ่ายจากภาพโดยตรงหรือไม่ก็ถ่ายจากภาพที่มีการบันทึกไว้ก่อนแล้ว และในที่นี้ข้อมูลภาพที่นำมาใช้เป็นข้อมูลที่ได้มาจากการดิจิไตซ์ ซึ่งภาพที่ได้นี้บางภาพอาจจะมีสัญญาณรบกวนจากภายในปะปนเข้ามาด้วย หรือไม่ก็อาจจะเป็นสัญญาณรบกวนที่เกิดขึ้นในขั้นตอนการบันทึกก่อนที่จะมาทำการดิจิไตร์ ซึ่งอาจจะมีผลเสียในการนำไปใช้ในขั้นตอนต่อๆ ไปได้ โดยเฉพาะข้อมูลภาพถ่ายดาวเทียมมักจะมีปัญหานี้เกิดขึ้นเสมอ ปกติถ้าหากมีสัญญาณรบกวนที่เรียกว่า Salt-and-Pepper noise ปะปนเข้ามาด้วยแล้วจะถูกทิ้งไปไม่มีการนำมาใช้งาน ข้อมูลที่มีสัญญาณรบกวนแบบนี้ไม่มีผู้ใช้รายใดต้องการซื้อไปใช้งาน ถึงแม้ให้ฟรีก็ไม่มีนี้ใครอยากได้ จึงทำให้ทางสถานีรับสัญญาณดาวเทียมต้องสูญเสียรายได้จากการขายภาพดังกล่าว และทำให้ผู้ใช้ต้องเสียเวลารออีกช่วงระยะเวลาหนึ่งเพื่อดาวเทียมจะได้โคจรผ่านบริเวณที่ต้องการ จากปัญหาดังกล่าว จึงได้พยายามคิดหาวิธีการที่จะกำจัดสัญญาณรบกวนเหล่านี้ และจากการศึกษาค้นคว้าได้พบว่า การกำจัดสัญญาณรบกวนด้วยวงจรกรองมัธยฐานแบบปรับขนาดได้อย่างอัตโนมัติ[3] เป็นวิธีการที่ให้ผลในการกำจัดสัญญาณรบกวนได้ดี และสามารถนำมาใช้ได้ผลดีกับภาพที่ต้องการลดขนาดข้อมูลโดยการเข้ารหัสภาพเซกเม้นเตชั่น เพราะว่าในขั้นตอนของการทำเซกเม้นต์จะไม่มีส่วนที่เป็น region ที่มีจุดภาพเพียง 1 หรือ 2 จุด ซึ่งอาจจะเกิดจากสัญญาณรบกวนหรือ noise ได้

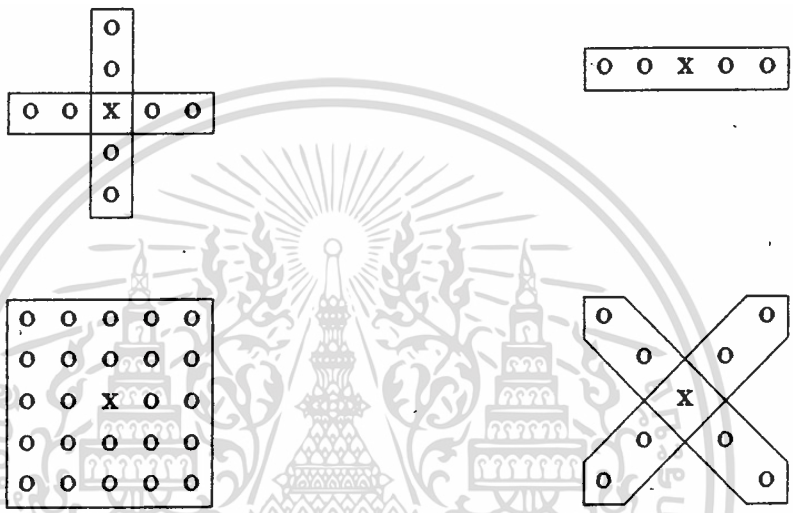
3.1 วงจรกรองมัธยฐาน (Median Filters)

วงจรกรองมัธยฐานเป็นการประมวลผลภาพแบบไม่เป็นเชิงเส้น(Non-linear Operator) โดยอาศัยข้อมูลในตารางหน้าต่างที่ได้กำหนดขึ้นในบริเวณข้อมูลอินพุท แล้วนำข้อมูลเหล่านั้นมาเรียงกันตามค่าของระดับความเข้มจากค่าต่ำสุดไปหาค่าสูงสุดตามลำดับ จากการนำข้อมูลมาเรียงกันจะมีลำดับของข้อมูลที่อยู่ตรงกลางของค่าทั้งหมด และค่านี้อาจจะเป็นค่าอินพุทตัวเดิมหรือค่าใหม่ก็ได้ ถ้าหากว่าอินพุทที่เราสนใจมีค่าแตกต่างไปจากค่าของจุดภาพที่อยู่ใกล้เคียงมาก ค่านี้จะถือเป็นสัญญาณรบกวนที่ปะปนเข้ามากับข้อมูลภาพ เมื่อค่าอินพุทตัวนี้ไม่ถูกเลือกนั้นก็หมายความว่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

17096

สัญญาณรบกวนได้ถูกกำจัดออกไป ความสามารถในการกำจัดสัญญาณรบกวน และคุณภาพความคมชัดของภาพเอ้าท์พุทที่ได้จะขึ้นอยู่กับรูปแบบและขนาดของตารางหน้าต่างที่กำหนด ซึ่งรูปแบบของตารางหน้าต่างที่นิยมนำมาใช้มีอยู่หลายแบบหลายขนาดด้วยกัน รูปที่ 3.1 เป็นตัวอย่างของตารางหน้าต่างที่นิยมใช้



รูปที่ 3.1 แบบต่าง ๆ ของตารางหน้าต่าง

วงจรกรองมัธยฐานเป็นที่รู้จักกันดีว่า สามารถกำจัดสัญญาณรบกวนแบบ Impulse ได้เป็นอย่างดีโดยที่ยังคงรักษาความคมชัดหรือขอบของภาพเอาไว้ได้[4-6] แต่อย่างไรก็ตามขนาดของตารางหน้าต่างของวงจรกรองมัธยฐาน (Window Length) จะเป็นตัวจำกัดคุณภาพของภาพที่ได้หลังจากการกรอง โดยภาพที่ได้หลังจากการกรองจะเบลอมากถ้าหากใช้ขนาดของตารางหน้าต่างใหญ่เกินไป ในกรณีตรงกันข้ามถ้าขนาดของตารางหน้าต่างมีขนาดเล็กเกินไปก็ไม่สามารถกำจัดสัญญาณรบกวนได้หมด การชดเชยปัญหาทั้งสองจึงได้มีการสร้างวงจรกรองมัธยฐานที่มีขนาดตารางหน้าต่างปรับขนาดได้อย่างอัตโนมัติตามปริมาณของสัญญาณรบกวนที่ปรากฏในตารางหน้าต่าง สำหรับสัญญาณรบกวนที่ปรากฏในข้อมูลภาพจะมีทั้งแบบพัลส์บวก(Positive Pulse) ซึ่งให้สัญญาณรบกวนเป็นจุดสีขาวหรือที่เรียกว่า Salt Noise กับสัญญาณที่เป็นพัลส์ลบ (Negative Pulse) ให้สัญญาณรบกวนเป็นจุดสีดำหรือที่เรียกว่า Pepper Noise

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 รูปแบบการจำลองสัญญาณรบกวน

สัญญาณรบกวนแบบอิมพัลส์(impulse noise) จะมีลักษณะการรบกวนเป็นจุดเล็ก ๆ แต่จะเป็นค่าที่สูงหรือต่ำมาก ๆ ซึ่งบางทีอาจจะเรียกว่า Salt-and-Pepper noise เป็นสัญญาณที่มีค่าระดับสีเทา(Gray Level) ที่แตกต่างไปจากจุดข้างเคียงมาก ๆ และเกิดในช่วงสั้น การรบกวนแบบนี้จะเห็นได้ชัดเจนมาก การจำลองการรบกวนแบบนี้ทำได้ดังต่อไปนี้

กำหนดให้

S_{ij} คือ ค่าระดับสีเทาของภาพต้นแบบที่จุด (i,j)

X_{ij} คือ ค่าระดับสีเทาของภาพที่ถูกรบกวนที่จุด (i,j)

n คือ ค่าของสัญญาณรบกวน

ซึ่งจะได้ว่า

$X_{ij} = S_{ij}$ เมื่อความน่าจะเป็นคือ 1-p

$X_{ij} = S_{ij} + n$ เมื่อความน่าจะเป็นคือ p

เมื่อ p คือความน่าจะเป็นของการถูกรบกวน (error probability) ต่อจุดภาพทั้งหมดในภาพ

n มีค่าการกระจายแบบ Uniform ในช่วงดังต่อไปนี้

$0 < n < 255-S_{ij}$ สำหรับ Salt Noise

$-S_{ij} < n < 0$ สำหรับ Pepper Noise

เนื่องจากภาพที่ใช้นั้น ในแต่ละจุดมีค่าระดับสีเทาอยู่ระหว่าง 0-255 โดยค่า 0 แสดงถึงค่าระดับสีเทาที่ดำที่สุด และค่า 255 แสดงถึงค่าระดับสีที่ขาวที่สุด ดังนั้นค่าของ Salt Noise จึงมีค่าสูงสุดไม่เกิน 255 และค่าของ Pepper Noise มีค่าต่ำสุดคือ 0 ในการทดลองจะทำการสุ่มสัญญาณรบกวนทั้งสองแบบไปพร้อมๆ กัน โดยกำหนดค่า p = 0.05 ดังรูปที่ 3.4

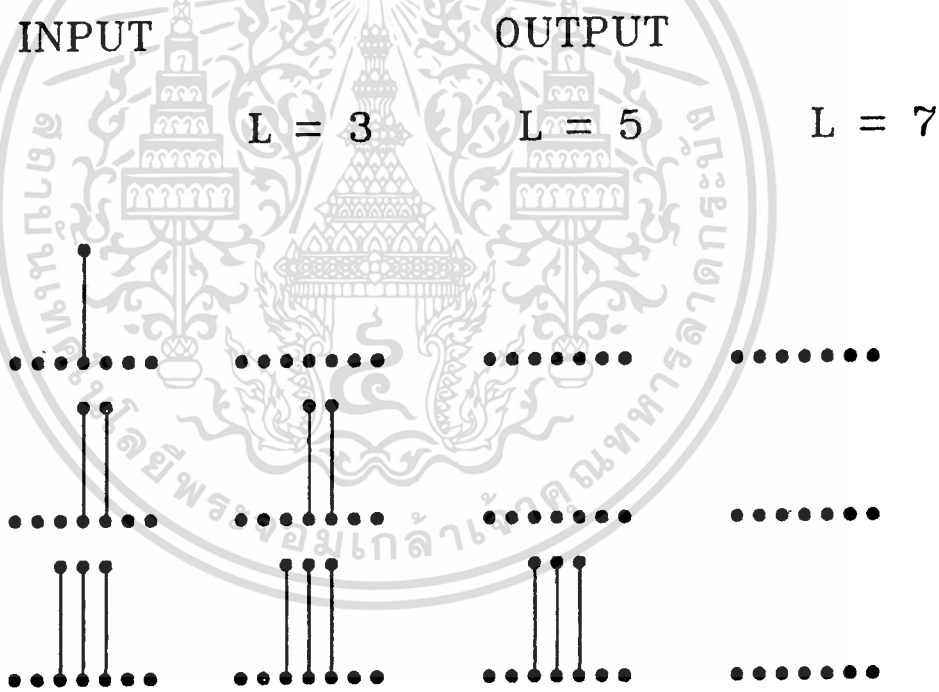
3.3 หลักการของวงจรกรองมัธยฐานแบบปรับขนาดได้อย่างอัตโนมัติ

วงจรกรองมัธยฐานเป็นเทคนิคในการประมวลผลสัญญาณแบบไม่เป็นเชิงเส้น ทำงานโดยการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กำหนดตารางหน้าต่างให้เคลื่อนผ่านลำดับของจุดภาพ และแทนค่าของจุดภาพที่อยู่ตำแหน่งกึ่งกลางของตารางหน้าต่างด้วยค่ามัธยฐานของค่าจุดภาพที่อยู่ในตารางหน้าต่างนั้น รูปแบบของตารางหน้าต่างที่ใช้อาจมีได้หลายลักษณะเช่น แบบกากบาท (Cross-Shaped) แบบจัตุรัส (Square-Shaped) เป็นต้น แต่ที่ยกตัวอย่างในบทนี้เป็นการแสดงผลที่ได้จากการใช้ตารางหน้าต่างแบบกากบาทเพียงแบบเดียว

สำหรับขบวนการของวงจรกรองมัธยฐาน ได้จากการพิจารณาคุณสมบัติของการกรองแบบมัธยฐานดังนี้ จากรูปที่ 3.2 ที่ขนาดความยาวตารางหน้าต่างเท่ากับ 3 จุดภาพ มีความสามารถในการกำจัดสัญญาณรบกวนที่มีความกว้างเท่ากับ 1 จุดภาพ ในขณะที่ใช้ความยาวตารางหน้าต่างเท่ากับ 5 จุดภาพ สามารถกำจัดสัญญาณรบกวนที่มีความกว้างเท่ากับ 2 จุดภาพ และที่ความยาวตารางหน้าต่างเท่ากับ 7 จุดภาพ มีความสามารถในการกำจัดสัญญาณรบกวนที่มีความกว้างเท่ากับ 3 จุดภาพ



รูปที่ 3.2 ความสามารถในการกำจัดสัญญาณรบกวนของการกรองแบบมัธยฐานที่มีขนาดความยาวของตารางหน้าต่าง L ขนาดต่าง ๆ กัน

เราอาศัยคุณสมบัติที่กล่าวมาข้างต้น ช่วยในการตัดสินใจเลือกใช้ขนาดความยาวของตารางหน้าต่างให้เหมาะสม โดยการตรวจสอบจากจำนวนจุดของสัญญาณรบกวนซึ่งพิจารณาได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังต่อไปนี้ วงจรกรองมัธยฐานแบบปรับขนาดได้อย่างอัตโนมัติที่ใช้กับสัญญาณรบกวนแบบ Salt Noise เมื่อตารางหน้าต่างเป็นแบบกากบาทขนาดสูงสุดของความยาวตารางหน้าต่างเท่ากับ 5 x 5

จุดในตารางหน้าต่างในทิศทางแนวนอน ซึ่งกำหนดโดย

$$x(n-2), x(n-1), x(n), x(n+1), x(n+2).$$

จุดในตารางหน้าต่างในทิศทางแนวตั้ง ซึ่งกำหนดโดย

$$y(n-2), y(n-1), y(n), y(n+1), y(n+2).$$

โดยที่ตำแหน่ง $x(n) = y(n) =$ จุดกึ่งกลางของตารางหน้าต่าง

เรากำหนดให้ setของ P และ Q เป็นผลต่างของค่าระดับสีเทาของจุดภาพ ณ. ตำแหน่งต่างๆ ในตารางหน้าต่าง เพื่อเป็นการตรวจสอบขนาดความกว้างของสัญญาณรบกวนที่อยู่ในตารางหน้าต่าง และนำค่าเหล่านี้ไปใช้ในขบวนการการตัดสินใจเพื่อเลือกขนาดของตารางหน้าต่าง

$$P_j = x(n+j-1)-x(n+j) \quad \text{เมื่อ } j = 1, 2$$
$$P_j = x(n+j+1)-x(n+j) \quad \text{เมื่อ } j = -1, -2$$
$$Q_j = y(n+j-1)-y(n+j) \quad \text{เมื่อ } j = 1, 2$$
$$Q_j = y(n+j+1)-y(n+j) \quad \text{เมื่อ } j = -1, -2$$

กำหนดให้ L_h, L_v คือ ขนาดของตารางหน้าต่างในแนวนอนและแนวตั้งตามลำดับ การพิจารณาเลือกขนาดของ L_h และ L_v ทำได้ดังนี้

I. การตรวจสอบทางแนวนอน :

ถ้า ($P_2 > T_2$ หรือ $P_{-2} > T_2$) หรือ
($P_1 > T_2$ และ $P_{-2} > T_2$) หรือ
($P_2 > T_2$ และ $P_{-1} > T_2$)
เลือก $L_h = 5$ แล้วไปตรวจสอบแนวตั้ง
ถ้า ($P_1 > T_1$ และ $P_{-1} > T_1$) หรือ
($1/2 < P_1/P_{-1} < 2$)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เลือก $L_h = 3$ แล้วไปตรวจสอบแนวตั้ง

ถ้า ($-T_1 < P_1 < T_1$ และ $-T_1 < P_{-1} < T_1$ และ
$-T_2 < P_2 < T_2$ และ $-T_2 < P_{-2} < T_2$)
เลือก $L_h = L_v = 1$ คือที่จุดกึ่งกลางของตารางหน้าต่างยังคงค่าเดิม
นอกเหนือจากนี้ให้เลือก $L_h = 1$ แล้วไปตรวจสอบทางแนวตั้งต่อไป

II. การตรวจสอบทางแนวตั้ง :

ถ้า ($Q_2 > T_2$ และ $Q_{-2} > T_2$) หรือ
($Q_1 > T_2$ และ $Q_{-2} > T_2$) หรือ
($Q_2 > T_2$ และ $Q_{-1} > T_2$)
เลือก $L_v = 5$ แล้วทำการจัดเรียงลำดับข้อมูล

ถ้า ($Q_1 > T_1$ และ $Q_{-1} > T_1$) หรือ
($1/2 < Q_1/Q_{-1} < 2$)
เลือก $L_v = 3$ แล้วทำการจัดเรียงลำดับข้อมูล

ถ้า ($-T_1 < Q_1 < T_1$ และ $-T_1 < Q_{-1} < T_1$ และ
$-T_2 < Q_2 < T_2$ และ $-T_2 < Q_{-2} < T_2$)
เลือก $L_v = L_h = 1$ คือเลือกค่าเดิมที่จุดกึ่งกลางของตารางหน้าต่าง
นอกเหนือจากนี้ เลือก $L_v = 1$ แล้วทำการจัดเรียงลำดับข้อมูล

ค่า T_1 และ T_2 เป็นค่า threshold ที่ใช้ในการทดสอบว่าค่า P และ Q จะเป็นเท่าไหร่จึงจะถือว่าเป็นสัญญาณรบกวน โดยจากการทดลองพบว่าค่าของ T_1 และ T_2 ที่ใช้มีค่าอยู่ระหว่าง 16 ถึง 48 และเป็นค่าเฉพาะภาพหนึ่งๆ เท่านั้น สำหรับในกรณีของ Pepper Noise ก็ทำในลักษณะเดียวกันกับของ Salt Noise ยกเว้นเพียงเปลี่ยนเครื่องหมายของ P และ Q ให้เป็นตรงกันข้ามแล้วทำตามขั้นตอนเดิมทุกประการ ซึ่งจากการทดลองได้ใช้การจำลองของสัญญาณรบกวนแบบผสมกันทั้งสองแบบ ดังนั้นในการกำจัดสัญญาณรบกวนนี้จึงต้องใช้ขั้นตอนของการกรองแบบมัธยฐานแบบที่ใช้กับ Salt Noise ก่อนแล้วจึงใช้วงจรกรองแบบที่ใช้กับ Pepper Noise อีกครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 ตัวอย่างและผลของวงจรกรอง

ตัวอย่างการตรวจสอบค่าของสัญญาณรบกวน โดยสมมุติให้ค่าระดับสีเทาของภาพในตารางหน้าต่างเป็นดังรูปต่อไปนี้

55
57
57 57 56 120 110
56
57

โดยตำแหน่งของจุดที่จะพิจารณาอยู่ในตำแหน่ง n = 3 เมื่อใช้ขั้นตอนของการกรองของ Salt Noise จะได้ว่า

x(1) = 57, y(1) = 55
x(2) = 57, y(2) = 57
x(3) = 56, y(3) = 56
x(4) = 120, y(4) = 56
x(5) = 110, y(5) = 57

P_1 = x(3)-x(4) = 56-120 = -64
P_2 = x(4)-x(5) = 120-110 = 10
P_{-1} = x(3)-x(2) = 56-57 = -1
P_{-2} = x(2)-x(1) = 57-57 = 0

ในทำนองเดียวกัน

Q_1 = y(3)-y(4) = 56-56 = 0
Q_2 = y(4)-y(5) = 56-57 = -1
Q_{-1} = y(3)-y(2) = 56-57 = -1
Q_{-2} = y(2)-y(1) = 57-55 = 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และโดยกำหนดให้ว่า $T_1 = 30$, $T_2 = 40$ แล้วจึงทำการตรวจสอบ

การตรวจสอบทางแนวนอน

จากเงื่อนไขแรก (คือถ้าเป็นจริงให้เลือก $L_h = 5$) ปรากฏว่าไม่เป็นจริง

จากเงื่อนไขต่อมา ปรากฏว่าไม่เป็นไปตามเงื่อนไขอีก จึงต้องตรวจสอบต่อไป

ในเงื่อนไขที่ 3 ก็ไม่เป็นจริงดังนั้นจึง เลือก $L_h = 1$

การตรวจสอบทางแนวตั้ง

จากเงื่อนไขแรกปรากฏว่าไม่เป็นจริง

จากเงื่อนไขที่ 2 ก็ไม่เป็นจริงตามเงื่อนไข จึงต้องตรวจต่อไป

และในเงื่อนไขถัดมาก็ไม่เป็นจริงอีก ดังนั้น จึงเลือก $L_v = 1$

ดังนั้นเราจึงได้ความยาวของตารางหน้าต่างทั้งในแนวนอนและแนวตั้งเท่ากันคือ 1 ซึ่งนั้นก็หมายความว่าไม่ต้องมีการจัดเรียงลำดับของข้อมูลในตารางหน้าต่าง หรือไม่มีการเปลี่ยนแปลงค่าของข้อมูลในจุดที่กำลังพิจารณา(จุดกึ่งกลางของหน้าต่าง)นั้นเอง จากตัวอย่างข้างบนจะเห็นว่าที่จุด x(4) มีค่าสูงแตกต่างจากจุดใกล้เคียงจุดอื่นๆ มาก นั้นก็หมายความว่าจุดนี้อาจจะเป็นสัญญาณรบกวนซึ่งเป็นสัญญาณรบกวนสีขาว(Salt Noise) หรืออาจจะเป็นขอบของภาพก็ได้ เพราะค่าของจุดภาพที่ตำแหน่งถัดไปคือ x(5) ก็มีค่าสูงเช่นกัน ส่วนในกรณีของสัญญาณรบกวนแบบ Pepper Noise นั้นจะมีค่าที่ต่ำกว่าค่าใกล้เคียงมากๆ เมื่อดูจากภาพก็จะเห็นเป็นจุดสีดำที่เด่นชัดมาก ในการกำจัดเราก็เลือกใช้ขั้นตอนของการกรองตามชนิดของสัญญาณรบกวน จากตัวอย่างถ้าเราใช้วงจรกรองมัธยฐานแบบธรรมดา คือทำการเรียงข้อมูลในตารางหน้าต่างที่กำหนดขนาดไว้ตายตัว ค่าของจุดที่กำลังพิจารณาอยู่จะถูกเปลี่ยนไปโดยที่จุดนั้นไม่ได้เป็นสัญญาณรบกวนเลย ซึ่งจากค่าเดิมคือ 56 เป็น 57 จึงเป็นสาเหตุทำให้ภาพที่ได้เบลอไม่คมชัด

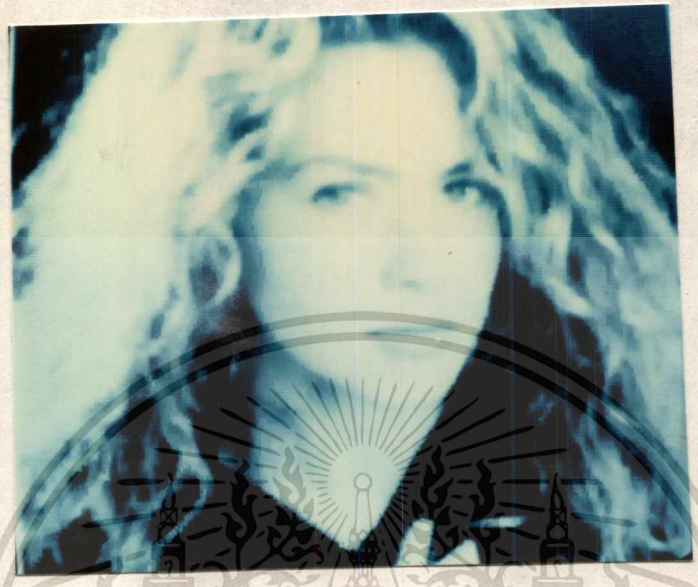
สำหรับสาเหตุที่เลือกใช้ขนาดความยาวสูงสุดของตารางหน้าต่างเท่ากับ 5 เพราะว่าเป็นขนาดที่กำลังเหมาะสมทั้งในด้านประสิทธิภาพในการกำจัดสัญญาณรบกวน และขั้นตอนในการทำงาน เพราะถ้าใช้ความยาวของตารางหน้าต่างที่ยาวมากกว่านี้ ขั้นตอนในการตรวจเช็คก็จะซับซ้อนและเสียเวลามากขึ้นในขณะที่ถ้าใช้ขนาดของตารางหน้าต่างเล็กกว่านี้ประสิทธิภาพก็จะด้อยตามไปด้วย แต่ถ้าหากว่าในภาพมีความหนาแน่นของสัญญาณรบกวนมาก ก็สามารถที่จะเพิ่มขนาดของตารางหน้าต่างได้ โดยเพิ่มขั้นตอนในการตรวจสอบเข้าไปอีก ในการทดลองได้จำลองสัญญาณรบกวนแบบอิมพัลส์สีขาวและสีดำเข้าไปในข้อมูลภาพดังรูปที่ 3.4 ส่วนในรูปที่ 3.5 3.6 เป็นผลของวงจรกรองมัธยฐานแบบตารางหน้าต่างคงที่ และแบบที่สามารถปรับขนาดได้อัตโนมัติตามลำดับ ความแตกต่างของผลที่ได้จากวงจรกรองทั้งสองสามารถแสดงให้เห็นได้ดังรูปที่ 3.7 และ 3.8 ซึ่งเป็น error ของวงจรกรองที่อยู่ในรูปค่าสมบูรณ์ของผลต่างของภาพเอาท์พุทกับภาพเดิม(รูปที่ 3.3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.3 แสดงข้อมูลภาพเดิมก่อนมีสัญญาณรบกวน

รูปที่ 3.4 เป็นภาพจากรูปที่ 3.3 ที่ได้ใส่สัญญาณรบกวนแบบอิมพัลส์เข้าไป

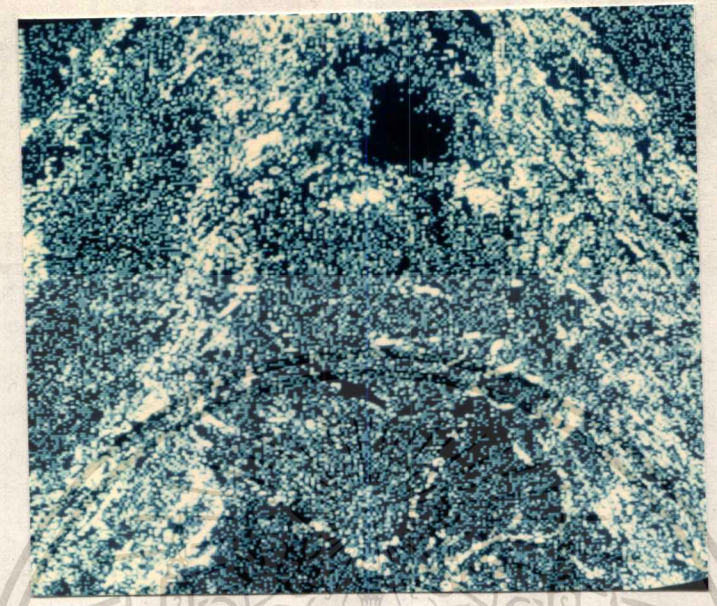
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



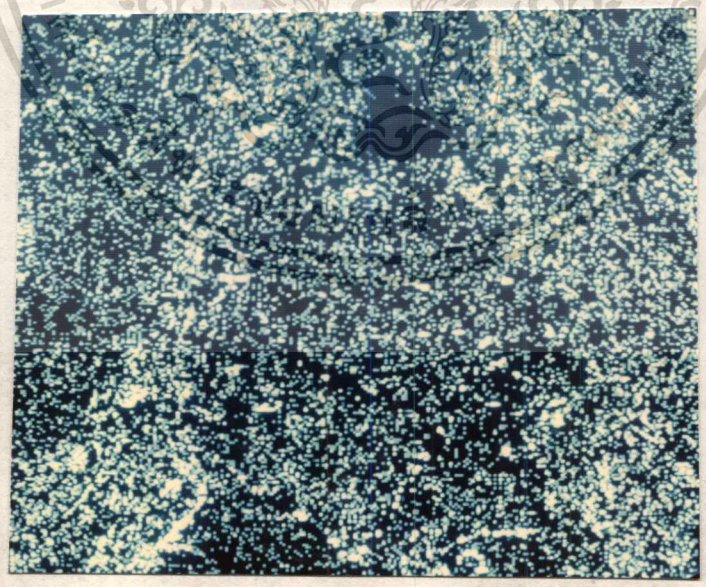
รูปที่ 3.5 เป็นผลที่ได้จากการใช้วงจรกรองมัธยฐานแบบกากบาทขนาด 5 x 5

รูปที่ 3.6 เป็นผลที่ได้จากการใช้วงจรกรองมัธยฐานแบบปรับขนาดได้อัตโนมัติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.7 แสดง Error ของวงจรกรองมัธยฐานแบบกากบาทคงที่ขนาด 5x5



รูปที่ 3.8 แสดง Error ของวงจรกรองมัธยฐานแบบปรับขนาดได้อัตโนมัติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การทำอิมเมจเซกเม้นเตชั่น
(Image Segmentation)

การนำภาพมาใช้ในงานการประมวลผลทางดิจิตอลนั้น ภาพที่นำมาใช้จะต้องอยู่ในลักษณะของข้อมูลทางดิจิตอล แต่โดยทั่วไปแล้วการเก็บบันทึกภาพจะทำกันในรูปของสัญญาณที่ต่อเนื่อง (analog) อย่างเช่นใช้กล้องทีวี (TV camera) เป็นตัวรับภาพแล้วบันทึกลงบนเทปแม่เหล็ก ดังนั้นการที่จะนำภาพเหล่านี้มาใช้กับการประมวลผลทางดิจิตอลแล้ว จึงจำเป็นต้องมีกรรมวิธีในการเปลี่ยนลักษณะของสัญญาณให้เหมาะสมเสียก่อน โดยอาศัยวงจรแปลงอนาล็อกเป็นดิจิตอล (A/D Converter) จากสัญญาณต่อเนื่องมาเป็นจุดภาพ ซึ่งจุดภาพเหล่านี้จะเป็นส่วนที่เล็กที่สุดที่รวมขึ้นมาเป็นภาพ โดยจำนวนของจุดภาพที่มีเป็นพันเป็นหมื่นจุดเรียงตัวกันตามเวลาของสัญญาณอินพุทที่เข้ามา และการแสดงภาพทางดิจิตอลส่วนมากแล้วนิยมใช้แสดงออกทางจอทีวี (TV screens) หรือไม่ก็ทางมอนิเตอร์ (monitor) โดยอยู่ในรูปของอาเรย์สองมิติ (x,y) ซึ่งจะเป็นการกำหนดจำนวนเส้นและจำนวนจุดต่อเส้นของภาพ ตัวอย่างเช่นภาพมีความละเอียดขนาด 256 เส้น และเส้นละ 256 จุด เป็นต้น โดยที่จุดภาพแต่ละจุดสามารถแสดงความแตกต่างของระดับสีเท่า (gray level) ได้เท่าไหร่นั้นขึ้นอยู่กับขนาดของจำนวนบิทที่ใช้ในการเก็บข้อมูล โดยทั่วไปแล้วมักจะกำหนดให้มีความแตกต่างของระดับสีเป็น 256 ระดับ สำหรับภาพขาวดำ ซึ่งนับว่าเป็นค่าที่เพียงพอในการแสดงภาพ เพราะตาของคนเรานั้นปกติก็ไม่สามารถที่จะแยกความแตกต่างของระดับสีเทาได้ถึง 256 ระดับ

4.1 ทำ Image Segmentation เพื่ออะไร

ลักษณะการมองของมนุษย์นั้น จะรับหรือกลั่นกรองเอาส่วนที่สำคัญของภาพที่รับเข้ามาทางสายตา แล้วส่งส่วนที่สำคัญเหล่านั้นไปยังสมองเพื่อที่จะแปรความหมายของภาพที่รับเข้ามา จุดประสงค์ของการทำอิมเมจเซกเม้นเตชั่นก็เป็นเช่นเดียวกัน กล่าวคือ การแยกหรือกลั่นกรองส่วนที่สำคัญของภาพออกมาให้เห็นเด่นชัดขึ้น โดยการรวมส่วนของภาพที่มีค่าระดับความเข้มใกล้เคียงหรือที่เท่ากันให้เป็นส่วนเดียวกัน และแยกส่วนที่มีค่าระดับความเข้มที่ต่างกันมากออกไปเป็นอีกส่วนหนึ่ง ซึ่งค่าระดับความแตกต่างที่มีค่าน้อยนั้น บางครั้งสายตามนุษย์ไม่สามารถจะแยกแยะได้ แต่เมื่อผ่านการทำเซกเม้นเตชั่นความแตกต่างนั้นจะเห็นได้ชัดขึ้น

ถึงแม้ว่าในปัจจุบันนี้ราคาของหน่วยความจำจะถูกลงมาบ้างแล้วก็ตาม การลดขนาดของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลภาพก็ยังเป็นสิ่งจำเป็นที่สำคัญ เพราะว่าภาพๆ หนึ่งไม่เพียงแต่ใช้พื้นที่หน่วยความจำที่มีขนาดใหญ่เท่านั้น แต่ในกรณีที่ต้องการส่งข้อมูลภาพผ่านไปในระบบการสื่อสารนั้นยังคงต้องใช้เวลาและความกว้างของแบนด์วิดท์สำหรับการส่งข้อมูลเป็นจำนวนมาก ภาพเซกเม้นเตชั่นจึงสามารถนำมาประยุกต์ใช้ในการแก้ปัญหาดังที่กล่าวได้เป็นอย่างดี

4.2 หลักการของ Image Segmentation

ดังที่ได้กล่าวมาแล้วว่าการทำอิมเมจเซกเม้นเตชั่น เป็นการแยกหรือกลั่นกรองส่วนที่สำคัญของภาพออกมาให้เด่นชัดขึ้น โดยจะทำการแยกส่วน (region) ของวัตถุในภาพออกเป็นส่วน ๆ ทั้งนี้เนื่องจากเนื้อของวัตถุที่เป็นบริเวณเดียวกัน จะมีลักษณะของจุดภาพที่มีคุณสมบัติพื้นฐานที่เหมือนกัน อย่างเช่น สี ความสว่างของภาพ หรือโครงร่างของภาพ จากนั้นจึงทำการแทนค่าของจุดภาพในแต่ละส่วนด้วยค่าเฉลี่ยของจุดภาพในบริเวณนั้น การทำเซกเม้นเตชั่นนั้นเราสามารถใช้หลักการต่าง ๆ ดังต่อไปนี้ได้

4.2.1 Edge Detection

การตรวจหาขอบ (Edge Detection) สามารถที่จะนำมาใช้ในขบวนการของการทำอิมเมจเซกเม้นเตชั่นได้ เพราะว่าวัตถุหรืออาณาบริเวณของวัตถุสามารถที่จะแยกออกจากสิ่งรอบข้างได้ด้วยขอบ เราสามารถที่จะตรวจสอบ (detected) ได้ด้วยความแตกต่างของระดับสีหรือโครงร่างของวัตถุในภาพ ส่วนของภาพ (region) ที่ตรวจได้จะถูกล้อมรอบด้วยขอบ อย่างไรก็ตาม การทำเซกเม้นเตชั่นด้วยการตรวจหาขอบนั้น ส่วนของภาพที่ได้จะขึ้นอยู่กับความคมชัดของขอบ ซึ่งภาพที่เบลอ ภาพที่โฟกัสไม่ดี หรือภาพที่ขอบไม่มีความคมชัด ถ้าหากนำไปทำเซกเม้นเตชั่นได้ด้วยวิธีนี้ จะให้ผลที่ไม่ดี อีกทั้งการทำเซกเม้นเตชั่นด้วยวิธีนี้เราไม่สามารถที่จะกำหนดจำนวนส่วน (number of region) ของภาพได้

4.2.2 Thresholding

วิธีการของ Thresholding นี้ อาศัยพื้นฐานของฮีสโตแกรม (histogram) ของระดับสีเทาที่กระจายอยู่ในภาพเป็นตัวแบ่งส่วนของภาพ โดยการกำหนดว่าถ้าฮีสโตแกรมแยกออกเป็นหลาย ๆ กลุ่มที่เด่นชัด กลุ่มเหล่านี้จะเป็นตัวแสดงคุณสมบัติที่แตกต่างของภาพนั้น ๆ ค่าเธร์สโฮลด์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

(threshold) จะถูกเลือกจากตำแหน่งรอยต่อระหว่าง mode ของกลุ่มข้อมูลในฮีสโตแกรม แล้วจึงนำมาใช้ในการแบ่งส่วนของภาพนั้น ซึ่งวิธีนี้ส่วนของภาพที่ได้จะขึ้นอยู่กับการกระจายของ mode ในฮีสโตแกรมของภาพ

4.2.3 Region Clustering

Region Clustering เป็นวิธีการการทำเซกเม้นเตชั่นที่อาศัยคุณสมบัติที่เหมาะสมของจุดภาพในการสร้างส่วนของภาพ โดยที่ภายในบริเวณเดียวกันจะต่อถึงกันและมีค่าระดับสีเทาใกล้เคียงกัน ในปัจจุบันนี้มีวิธีการที่ใช้หลักการอันนี้หลายวิธีด้วยกัน เช่น split and merge วิธีการนี้จะเริ่มต้นด้วยการเลือกจุดภาพขึ้นมาก่อนหนึ่งจุด ต่อจากนั้นก็จะทำการพิจารณาจุดภาพที่อยู่ใกล้เคียง (neighbours) เพื่อที่จะรวมเข้าเป็นส่วนเดียวกัน จุดภาพที่อยู่ในบริเวณใกล้เคียงทุกจุดที่อยู่ภายใต้เกณฑ์การรวมจะกลายเป็นส่วนของภาพที่เกิดขึ้นในภาพนั้น แต่ถ้าจุดภาพที่อยู่ใกล้เคียงนั้นตรวจสอบแล้วไม่อยู่ภายใต้ของเกณฑ์การรวม จุดภาพนั้นจะไม่ถูกรวมเข้าไปในส่วนนั้นของภาพ แต่จะถูกเลือกให้เป็นจุดเริ่มต้นของส่วนอื่นๆ ต่อไป หลังจากที่จุดภาพทุกจุดได้รวมตัวกันเป็นกลุ่ม (region) การที่จะทำให้ได้ส่วนของภาพสมบูรณ์ขึ้นก็สามารถที่จะเริ่มใหม่โดยการแยกแต่ละส่วนออก แล้วใช้หลักการการรวมกันของจุดภาพที่กล่าวมาแล้วอีกครั้งหนึ่ง วิธีการของ Region clustering จะพยายามหาจุดภาพเริ่มต้นในการสร้าง region เพื่อให้ได้ผลที่ดีที่สุดในการเริ่มรวมจุดภาพ วิธีการทำเซกเม้นเตชั่นที่ใช้หลักการอันนี้อีกวิธีหนึ่งคือ การทำอิมเมจเซกเม้นเตชั่นด้วยการแบ่งแยกและรวมโดยตรง (Directed Split-and-Merge)[7] วิธีการนี้อาศัยโครงสร้าง Quartic Picture Tree ของข้อมูล แต่จะมีข้อเสียในขั้นตอนของการรวมจุดภาพเข้าด้วยกัน เนื่องจากในขั้นตอนดังกล่าวจะขาดความอิสระในการรวมจุดภาพ ดังนั้นจึงได้มีผู้เสนอวิธีการใหม่คือ วิธีการแบ่งและรวบรวมที่มีการปรับปรุง(Modification of Split-and-Merge) [8] ถึงแม้ว่าวิธีการนี้จะให้ผลดีกว่าสองวิธีการที่กล่าวมาแล้วข้างต้นก็จริง แต่อย่างไรก็ตาม การทำเซกเม้นเตชั่นด้วยหลักการของ region clustering ก็ยังมีข้อจำกัดอยู่ คือไม่สามารถที่จะสร้างส่วนของภาพที่เป็นเส้นที่มีขนาดเล็กในภาพได้ดีพอ

4.3 ทำไมถึงใช้วิธีการของทฤษฎีกราฟ

เนื่องจากวิธีของทฤษฎีกราฟ เป็นวิธีที่มีการนำเอาข้อมูลส่วนใหญ่ของภาพ(global data) มาใช้เป็นเกณฑ์ในการตัดสินการแยกและการรวมของ region ด้วย ดังนั้นการทำเซกเม้นเตชั่น

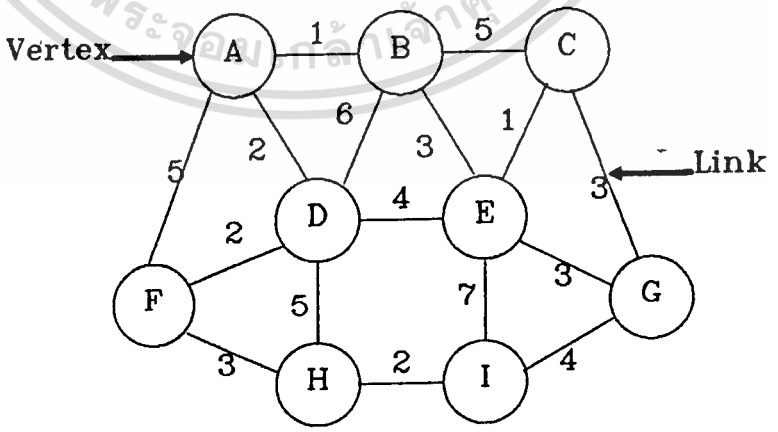
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยใช้วิธีของทฤษฎีกราฟจึงสามารถให้ภาพเซกเม้นต์ที่ได้มีความกลืนของ region ตามลักษณะโครงสร้างของภาพเดิม นอกจากนั้นยังสามารถที่จะกำหนดจำนวนของ region ได้แน่นอนตามที่ต้องการ ดังนั้นรายละเอียดของภาพซึ่งขึ้นอยู่กับจำนวนของ region ก็สามารถกำหนดได้เช่นกัน การทำเซกเม้นเตชั่นด้วยวิธีนี้จึงเหมาะในการนำไปใช้ในงาน image understanding และ image data compression จากการที่เราสามารถกำหนดจำนวน region ของพื้นที่ในภาพได้ เมื่อนำมาประยุกต์กับเทคนิคการเข้ารหัสภาพที่จะได้เสนอในวิทยานิพนธ์นี้ ทำให้เราสามารถลดข้อมูลภาพลงได้เป็นจำนวนมาก

4.4 ทฤษฎีกราฟ (Graph Theory)

ในหัวข้อนี้จะเป็นการกล่าวนำถึงความหมายของคำศัพท์ และรายละเอียดของทฤษฎีกราฟที่จำเป็นที่ต้องนำมาใช้ในการทำเซกเม้นเตชั่นภาพ ทฤษฎีกราฟเป็นการศึกษาเกี่ยวกับกราฟและการนำมาประยุกต์ใช้งานของกราฟ โดยให้ $G = (V, E)$ เป็นกราฟที่ประกอบด้วยจุดยอด (vertices) ที่ต่อเชื่อมกับจุดยอดอื่นๆ โดยตัวเชื่อม (Links) $E_{i,j}$ จะเชื่อมต่อจุดยอด V_i และ V_j เข้าด้วยกัน ในขณะที่น้ำหนัก (Weight) จุดยอดของกราฟ V_i คือ v_i และค่าน้ำหนักของตัวเชื่อม (Link Weight) $E_{i,j}$ คือ $e_{i,j}$ จุดยอดแต่ละจุดของกราฟไม่จำเป็นที่จะต้องเชื่อมต่อกับจุดยอดอื่นๆ ทุกจุด แต่ถ้ามีการเชื่อมทุกๆ จุดก็จะเป็นกราฟที่สมบูรณ์

รูปที่ 4.1 เป็นตัวอย่างของกราฟที่ประกอบด้วยจุดยอด 9 จุดคือ A, B, C, D, E, F, G, H, และ I ตามลำดับ พร้อมกันนี้ได้แสดงค่าน้ำหนักของตัวเชื่อมระหว่างจุดยอดแต่ละจุดด้วย



รูปที่ 4.1 แสดงกราฟตัวอย่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับคำศัพท์ที่สำคัญ ๆ ในทฤษฎีกราฟ ที่จำเป็นต้องนำมาใช้ในการทำเซกเม้นเตชั่นของภาพ ดังมีรายละเอียดดังต่อไปนี้

กราฟย่อย (Partial Graph) เป็นกราฟที่ประกอบด้วยจำนวนจุดยอดเท่ากับกราฟต้นแบบ (Original Graph) แต่จะมีตัวเชื่อมเป็นซับเซ็ต(subset)ของกราฟต้นแบบ ดังตัวอย่างในรูปที่ 4.2

ลูกโซ่ (Chain) เป็นจำนวนของจุดยอดที่ต่อเนื่องกัน โดยจุดยอดแต่ละจุดจะเชื่อมต่อกับจุดยอดถัด ๆ ไปด้วยตัวเชื่อมภายในกราฟ

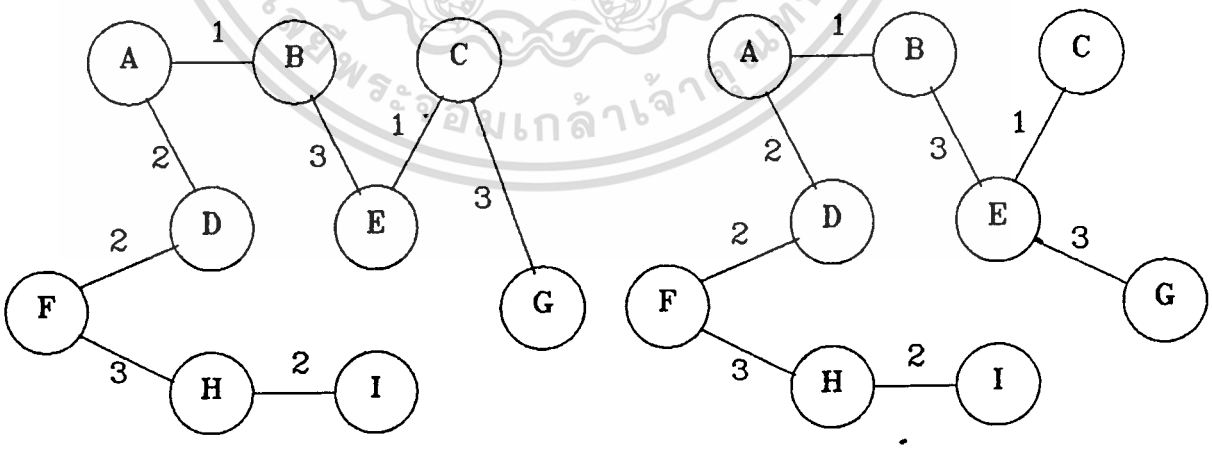
ไซเคิล (Cycle) เป็นทางเดินลูกโซ่ ที่มีจุดยอดเริ่มต้นและจุดยอดสุดท้ายเป็นจุดเดียวกัน ดังตัวอย่างของจุดยอด A, B, E, และ D ในรูปที่ 4.1

ทรี (Tree) เป็นเซ็ตของลูกโซ่ที่ต่อถึงกันในกราฟ แต่จะไม่ครบวงจรหรือไม่เป็นไซเคิล ดังนั้นทรีที่เชื่อมต่อด้วยจุดยอด N จุด จะมีตัวเชื่อมทั้งหมดเท่ากับ N-1

สแพนนิ่งทรี (Spanning Tree) คือทรีที่เป็นกราฟย่อย

ช็อตเตสท์สแพนนิ่งทรี (Shortest Spanning Tree) หรือ SST เป็นสแพนนิ่งทรีที่ให้ผลรวมของน้ำหนักตัวเชื่อมมีค่าต่ำสุดเท่าที่จะเป็นไปได้ ดังนั้นช็อตเตสท์สแพนนิ่งทรีจึงไม่จำเป็นที่จะต้องมีเพียงกรณีเดียวดังตัวอย่างในรูปที่ 4.2 แสดงลักษณะของกราฟย่อยที่เป็น SST ด้วย

ฟอเรสท์ (Forest) เป็นเซ็ตของทรีที่อยู่ในกราฟ



รูปที่ 4.2 แสดงกราฟย่อยที่เป็นช็อตเตสท์สแพนนิ่งทรี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากคำศัพท์ต่าง ๆ ที่กล่าวมา จะถูกนำมาใช้ในขั้นตอนของการแปลงข้อมูลภาพให้อยู่ในรูปของกราฟเพื่อการหาช็อตเตสท์สแพนนิ่งทรีของกราฟ สำหรับการทำเซกเม้นเตชั่น

4.4.1 การแปลงข้อมูลภาพไปเป็นกราฟ

การใช้ทฤษฎีกราฟในการวิเคราะห์ข้อมูลภาพหรือการทำเซกเม้นเตชั่นนั้น ข้อมูลเดิมของภาพ (original image) จะถูกเปลี่ยนให้อยู่ในรูปของกราฟเสียก่อน โดยการแปลงจุดภาพแต่ละจุดให้เป็นจุดยอดของกราฟ และน้ำหนักของจุดยอดบนกราฟขึ้นอยู่กับระดับความเข้มของจุดภาพ

ดังนั้นถ้าความเข้มหรือระดับสีเทาของของจุดภาพที่ตำแหน่ง (x,y) คือ f(x,y) แล้วน้ำหนักจุดยอดของกราฟก็คือ

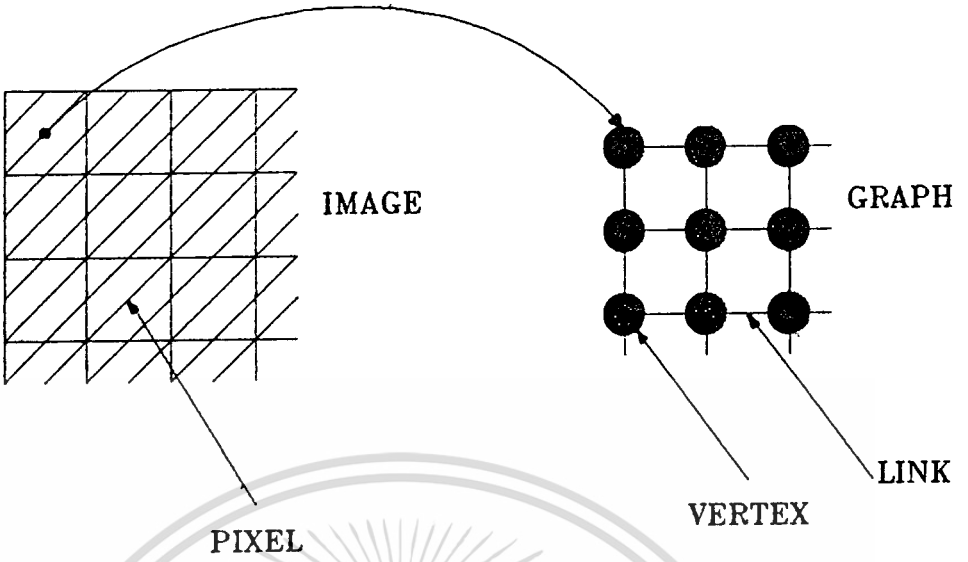
$$v_i = f(x,y) \qquad (4.1)$$

โดยที่ x, y ถูกแปลง(map) ไปเป็น i ในลักษณะ one-to-one mapping หลังจากนั้นทำการกำหนดค่าน้ำหนักตัวเชื่อมของจุดยอดต่าง ๆ ซึ่งสามารถที่จะกำหนดได้หลายแบบ แต่ในที่นี้จะใช้เป็นค่าสมบูรณ์ (absolute value) ของค่าความแตกต่างระหว่างจุดยอดของกราฟที่อยู่ข้างเคียง ซึ่งเป็นการวัดความเหมือนหรือความใกล้เคียง (similarity) ของระดับความเข้มระหว่างจุดภาพที่อยู่ใกล้ ๆ กันนั้นเอง จะได้ว่า

$$e_{i,j} = \left| v_i - v_j \right| \qquad (4.2)$$

จุดยอดของกราฟแต่ละจุดสามารถที่จะเชื่อมกับจุดยอดอื่น ๆ ภายในกราฟ แต่ที่มีประโยชน์และเพื่อลดความยุ่งยาก จะใช้การเชื่อมต่อของจุดยอดเฉพาะจุดยอดที่อยู่ใกล้กันที่สุดเท่านั้น โดยที่สามารถเชื่อมกับจุดใกล้เคียงโดยรอบได้แบบ 4 ทิศทาง และ 8 ทิศทาง แต่ในที่นี้การทำเซกเม้นเตชั่นเราจะใช้การเชื่อมต่อแบบ 4 ทิศทาง เพื่อความสดวกและเป็นการลดขนาดของข้อมูลในตอนประมวลผล(process)จากรูปที่ 4.3 เป็นตัวอย่างการแปลงข้อมูลภาพเดิมให้เป็นกราฟสำหรับนำไปดำเนินการในขั้นต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.3 แสดงการแปลงข้อมูลภาพไปเป็นกราฟ

4.4.2 การหาซ็อตเตสท์สแพนนิ่งทรีของกราฟ

วิธีการในการคำนวณหาซ็อตเตสท์สแพนนิ่งทรีของกราฟ มีรูปแบบการคำนวนอยู่หลายวิธี เมื่อภาพถูกแปลงให้อยู่ในรูปของกราฟที่มีความสัมพันธ์แบบ 4 ทิศทาง วิธีการของ Kruskal's Algorithm [10] ถูกนำมาใช้ เวลาในการคำนวนของวิธีนี้จะเป็นสัดส่วนกับจำนวนของตัวเชื่อม คูณด้วยค่าล็อก (log) ของจุดยอด หรือเท่ากับ E.log V เมื่อ E เป็นจำนวนของตัวเชื่อม และ V เป็นจำนวนจุดยอดในกราฟ วิธีการนี้จะสร้างซ็อตเตสท์สแพนนิ่งทรีโดยการเชื่อมทรีของฟอเรสท์เข้าด้วยกันด้วยตัวเชื่อมที่มีค่าน้ำหนักต่ำที่สุด เริ่มต้นจากจุดยอดทุก ๆ จุด ทำการหาค่าตัวเชื่อมที่มีค่าต่ำสุด แล้วจึงทำการสร้างทรีขึ้นมาจากจุดยอดเหล่านี้ โดยค่อย ๆ เชื่อมจุดยอดของกราฟที่มีค่าน้ำหนักของตัวเชื่อมน้อยที่สุดขึ้นเป็นทรีก่อน แล้วจึงเชื่อมทรีเหล่านี้เข้าด้วยกันเป็นซ็อตเตสท์สแพนนิ่งทรีด้วยตัวเชื่อมที่มีค่าน้ำหนักต่ำสุดเช่นกัน จนกระทั่งจุดยอดทุกจุดบนกราฟกลายเป็นสแพนนิ่งทรีของกราฟ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รายละเอียดของวิธีการหาช็อตเตสท์สแพนนิ่งทรีมีดังต่อไปนี้

(1) จัดเรียงลำดับค่าน้ำหนักตัวเชื่อมของกราฟจากค่าต่ำไปหาค่าสูง

(2) เริ่มจากฟอเรสท์ของกราฟที่ยังไม่มีตัวเชื่อม

(3) กระทำซ้ำจากนี้ไป

(4) หาตัวเชื่อมที่มีค่าน้ำหนักต่ำสุดตัวต่อไป

(5) ถ้าตัวเชื่อม ๆ ต่อกันแล้วไม่ทำให้เกิดเป็นไซเคิล

(6) รวมตัวเชื่อมตัวนี้เข้าไปในฟอเรสท์

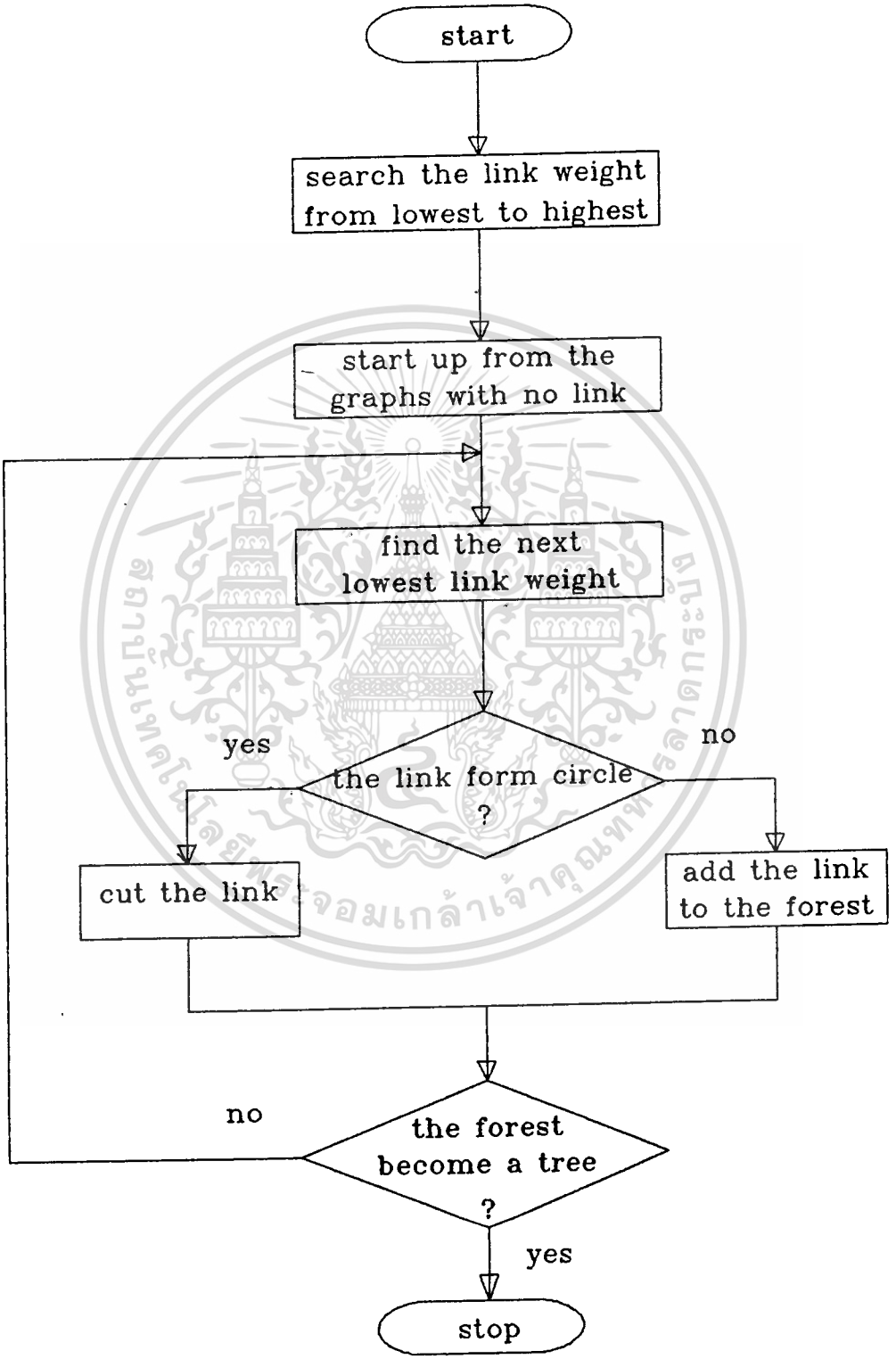
(7) ถ้าทำให้เกิดไซเคิล

(8) ตัดตัวเชื่อมนี้ทิ้งไป

(9) จนกระทั่งฟอเรสท์กลายเป็น สแพนนิ่งทรีของกราฟ

จากลำดับหัวข้อทั้ง 9 สามารถที่จะเขียนเป็นโฟลว์ชาร์ตของการหาช็อตเตสท์สแพนนิ่งทรีของกราฟได้ดังรูปที่ 4.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.4 แสดงโฟลว์ชาร์ต การหา SST ของกราฟ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

4.5 การแบ่งส่วนภาพจากสแพนนิ่งทรีของกราฟ (Segmentation from Spanning Tree)

เมื่อการทำอิมเมจเซกเม้นเตชั่นถึงการแบ่งภาพออกเป็น region ย่อย ๆ ที่ไม่ซ้อนทับกัน และทุก ๆ region ยังคงเป็นส่วนประกอบของภาพอยู่เช่นเดิม ดังนั้นสแพนนิ่งทรีซึ่งเป็นส่วนของกราฟที่ได้มาจากการแปลงข้อมูลภาพ จึงสามารถที่จะนำมาใช้ในการแบ่งส่วนของภาพได้ โดยที่เซกเม้นต์แต่ละเซกเม้นต์จะได้มาจากการตัดตัวเชื่อมของสแพนนิ่งทรี ฟอเรสท์ของสแพนนิ่งทรีที่โดนตัดจะแทนแต่ละส่วนของภาพ โดยถ้า T เป็นทรีในฟอเรสท์ ดังนั้น $P(T)_i$ สามารถกำหนดได้ดังนี้

$$P(T)_i = \begin{cases} 1 & \text{ถ้า } V_i \text{ เป็นสมาชิกของ } T \\ 0 & \text{เมื่อเป็นอย่างอื่น} \end{cases} \qquad (4.3)$$

ถ้ากำหนดให้ $M_{x,y}$ เป็นการแปลง (mapping) จาก (x,y) ไปเป็น T ดังนั้น $P(T)_i$ ของกราฟสามารถแปลงกลับไปเป็นภาพได้ โดยที่ทุก ๆ ตัวของ $P(T)_i$ จะเป็นตัวกำหนดขอบเขตและจำนวนของจุดภาพในแต่ละเซกเม้นต์ของภาพ และแต่ละเซกเม้นต์จะมีค่าระดับของความเข้มเป็นค่า ๆ หนึ่ง เมื่อกำหนดให้ p(T) เป็นค่าความเข้มของเซกเม้นต์นั้นๆ จะได้ว่า

$$p(T) = \frac{\sum_i P(T)_i . V_i}{\sum_i P(T)_i} \quad \text{สำหรับทุกค่าของ } i \qquad (4.4)$$

ในที่นี้ p(T) จึงหมายถึงค่าเฉลี่ยน้ำหนักของจุดยอดของทรี ที่ใช้เป็นค่าความเข้มของเซกเม้นต์ในภาพ ส่วนการแปลงจากภาพเดิมไปเป็นภาพเซกเม้นเตชั่น $S_{x,y}$ นั้น สามารถที่จะเขียนได้ดังนี้

$$S_{x,y} = p(M_{x,y}) \qquad (4.5)$$

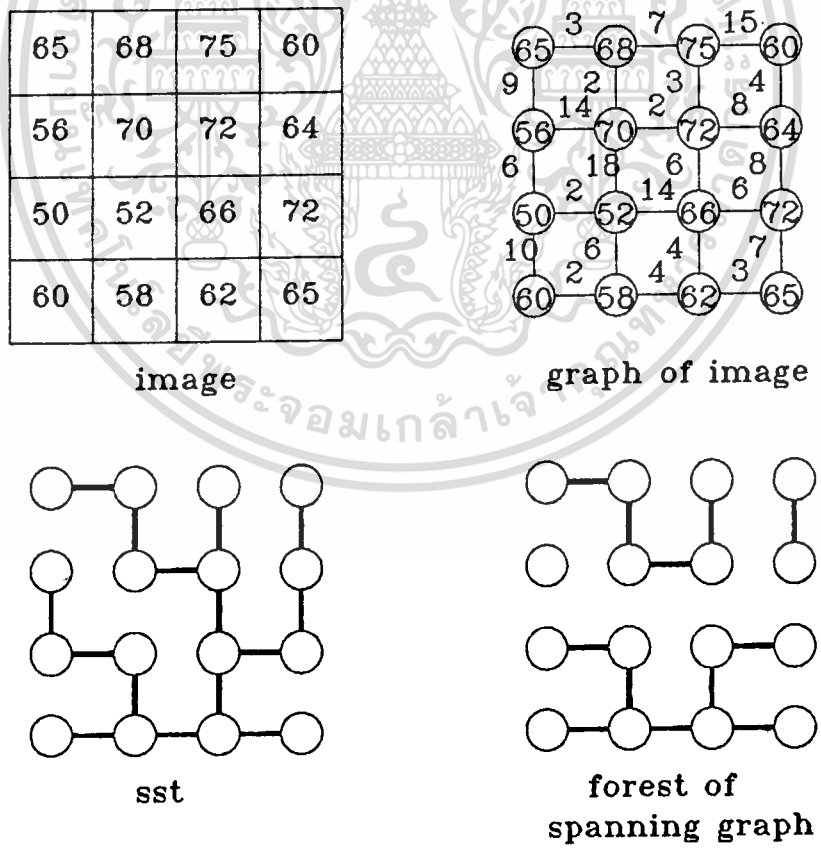
จากกราฟที่เราได้ทำการคำนวนหา SST ไว้แล้ว สามารถนำมาใช้ในการทำเซกเม้นเตชั่นโดยการตัด SST ของกราฟตรงตำแหน่งที่มีค่าของตัวเชื่อมที่มีค่าสูงสุด ทั้งนี้เพื่อให้แน่ใจว่าแต่ละ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เซกเม้นต์ของภาพที่แยกกันนั้นมีค่าความแตกต่างจากบริเวณจุดภาพที่ใกล้เคียงมากที่สุด และยังเป็นตัวยืนยันได้ว่าจุดภาพทุก ๆ จุด ที่อยู่ในเซกเม้นต์เดียวกัน จะมีคุณสมบัติใกล้เคียงกับจุดภาพที่อยู่ใกล้ ๆ กันในของแต่ละเซกเม้นต์มากกว่าจุดภาพอื่นๆ ที่อยู่ต่างเซกเม้นต์ออกไป และเซกเม้นต์ต่อ ๆ ไปสามารถหาได้โดยการตัดตัวเชื่อมของ SST ตัวต่อไป วิธีการในการทำเซกเม้นเตชั่นจาก SST สามารถสรุปได้ดังต่อไปนี้

1. ทำการแปลงข้อมูลภาพให้อยู่ในรูปของกราฟ
2. ทำการหา SST ของกราฟ
3. ทำการตัด SST ของกราฟเป็นจำนวน N-1 ครั้ง จะทำให้ได้ภาพที่ประกอบด้วย N เซกเม้นต์
4. ทำการแปลงกราฟที่ถูกตัดแล้วซึ่งเรียกว่า ฟอเรสท์ กลับเป็นภาพเซกเม้นต์

รูปที่ 4.5 เป็นตัวอย่างข้อมูลภาพขนาด 4x4 จุดภาพนำมาแปลงเป็นกราฟแล้วหา SST ของกราฟ ต่อจากนั้นจึงนำมาทำเซกเม้นเตชั่นดังรูปที่ 4.6 ตามขั้นตอนที่กล่าวมาแล้วข้างต้น



รูปที่ 4.5 แสดง SST ของกราฟและการได้มาของภาพเซกเม้นเตชั่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

70	70	70	62
56	70	70	62
60.6	60.6	60.6	60.6
60.6	60.6	60.6	60.6

segment image

รูปที่ 4.6 แสดงภาพเซกเม้นต์ที่ได้จากข้อมูลในรูปที่ 4.5

4.6 การแบ่งส่วนภาพจากรีเคอร์ซีฟช็อตเตสท์สแพนนิ่งทรี (RSST)

เนื่องจากขั้นตอนในการสร้าง SST ของกราฟนั้น ได้จากการจัดเรียงค่าลิ้งเวทจากค่าต่ำสุดไปยังค่าสูงสุด ในการจัดเรียงค่าลิ้งเวทแต่ละครั้งหมายถึงการรวมเอา region ทั้งสองที่ถูกเชื่อมต่อด้วยลิ้งเวทนั้นเข้าด้วยกัน ภายหลังการรวม region แล้วลิ้งเวทอื่น ๆ ที่เชื่อมต่อกับ region นั้น มิได้มีการปรับเปลี่ยนค่าให้สอดคล้องกับค่าเฉลี่ยของ region ขบวนการดังกล่าวก่อให้เกิดข้อเสียสองประการกล่าวคือ ประการแรก ถ้าหากเกิดสัญญาณรบกวนในภาพก่อนทำการเซกเม้นเตชั่น จะทำให้จุดภาพที่เป็นสัญญาณรบกวนถูกแยกออกมาเป็นเซกเม้นต์เล็ก ๆ ซึ่งจะมีผลเสียโดยตรงต่อการลดข้อมูลภาพ ประการสุดท้ายจะเกิดปัญหาเนื่องจากบางครั้งจุดภาพขาวกับจุดภาพดำที่แตกต่างกันมาก อาจถูกนำมารวมอยู่ในเซกเม้นท์เดียวกัน ถ้าหากจุดภาพทั้งสองดังกล่าวถูกเชื่อมต่อด้วยกลุ่มจุดภาพที่มีค่าระดับความเข้มสว่างเรียงไล่กันไป ดั้งนั้นค่าความแตกต่างของระดับความเข้มสว่างของจุดภาพที่ใกล้กันจะมีค่าแตกต่างกันน้อย ทำให้ค่าลิ้งเวทที่ได้จากกลุ่มจุดภาพดังกล่าวมีค่าต่ำ ๆ จึงมีผลทำให้ต้องรวมจุดภาพขาวกับจุดภาพดำอยู่ใน region เดียวกัน

ปัญหาต่าง ๆ ของวิธี SST นั้นเกิดจากค่าเวทที่ใช้ กล่าวคือไม่ได้มีการปรับเปลี่ยนค่าเวทและค่าลิ้งเวทให้สอดคล้องกับค่าเฉลี่ยของ region ซึ่งเป็น global information ดังนั้นในการแก้ปัญหาของ SST สามารถทำได้โดยการใช้วิธีของ Recursive Shortest Spanning Tree (RSST) ซึ่งใน RSST นี้จะได้ทำการปรับเปลี่ยนค่าลิ้งเวทต่าง ๆ ที่เชื่อมต่อกับ region ผลลัพธ์ให้สอดคล้องกับค่าเฉลี่ยของ region (ซึ่งถือเป็นเวทของ region นั้นเอง) ที่ได้จากการกำจัดค่าลิ้งเวทต่ำสุด อันเป็นการรวมจุดภาพเข้ามาเป็นส่วนหนึ่งของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

region นั้นเอง ดังนั้นในการรวมจุดภาพแต่ละครั้งจะมีการปรับปรุงค่าลิ้งเวทตามค่าเฉลี่ยของ region ใหม่ตลอดเวลา วิธีการแบบนี้จึงถูกเรียกว่า รีเคอร์ซีฟช็อตเตสท์เพนนิ่งทรี (Recursive Stortest Spanning Tree)

เนื่องจากทุก ๆ จุดภาพในแต่ละ region จะมีค่าระดับสีเทาเท่ากัน ซึ่งอาจกล่าวได้ว่าจุดยอดของกราฟจุดหนึ่งสามารถใช้แสดงแทนตลอดทั้ง region ถ้าหากตัด SST ออกเป็นฟอเรสท์ (เพื่อแบ่งเป็น region ต่าง ๆ) แต่ละทรีจะกลายเป็นจุดยอดจุดหนึ่ง ดังนั้นในการแปลงจากกราฟกลับไปเป็นภาพจะเป็นการแปลงแบบหนึ่งจุดไปยังหลายจุด และแต่ละ partition $P(i)_{x,y}$ ที่ถูกกำหนดโดยจุดยอด V_i ของกราฟจะเป็นไปตามสมการ (4.6)

$$P(i)_{x,y} = \begin{cases} 1 \text{ ถ้า } V_i \text{ map ไปเป็น } (x,y) \\ 0 \text{ เมื่อเป็นอย่างอื่น} \end{cases} \qquad (4.6)$$

และค่าเวทของจุดยอดของกราฟสามารถกำหนดได้จาก

$$v_i = \frac{\sum_{x,y} P(i)_{x,y} . f(x,y)}{\sum_{x,y} P(i)_{x,y}} \quad \text{สำหรับทุกๆ ค่าของ } x \text{ และ } y \qquad (4.7)$$

$f(x,y)$ คือค่าระดับสีเทาของจุดภาพ ณ ตำแหน่ง x,y

การแปลงข้อมูลภาพไปเป็นกราฟในตอนเริ่มแรกนั้นเหมือนกับวิธีแรกที่กล่าวมาแล้ว แต่ในขั้นตอนต่อ ๆ ไปโครงสร้างของกราฟจะถูกทำให้เปลี่ยนแปลงตลอดเมื่อมีการคำนวณซ้ำ ๆ (recursion) ในตอนเริ่มต้นจุดภาพแต่ละจุดจะถูกพิจารณาเป็นส่วนของภาพที่แยกกันเป็นคนละ region จากนั้นจะทำการรวมเอาสอง region เข้าด้วยกัน การรวมจุดยอดจะเริ่มจาก จุดยอด V_i และ V_j ที่มีค่าความแตกต่างน้อยที่สุด จะถูกรวมเข้าด้วยกันได้เป็นจุดยอดใหม่ V_k ซึ่งเป็นตัวกำหนดค่าเฉลี่ยของ region ใหม่นั้นเอง โดยที่

$$v_k = \frac{\sum_{x,y} (P(i)_{x,y} . f(x,y) + P(j)_{x,y} . f(x,y))}{\sum_{x,y} (P(i)_{x,y} + P(j)_{x,y})} \quad \text{สำหรับทุกค่าของ } x,y \qquad (4.8)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และ region ใหม่ที่ได้คือ $P(k)_{x,y}$ ซึ่งเป็นผลจากการรวม region ทั้งสองเข้าด้วยกัน นั้นคือ

$$P(k)_{x,y} = P(i)_{x,y} + P(j)_{x,y} \tag{4.9}$$

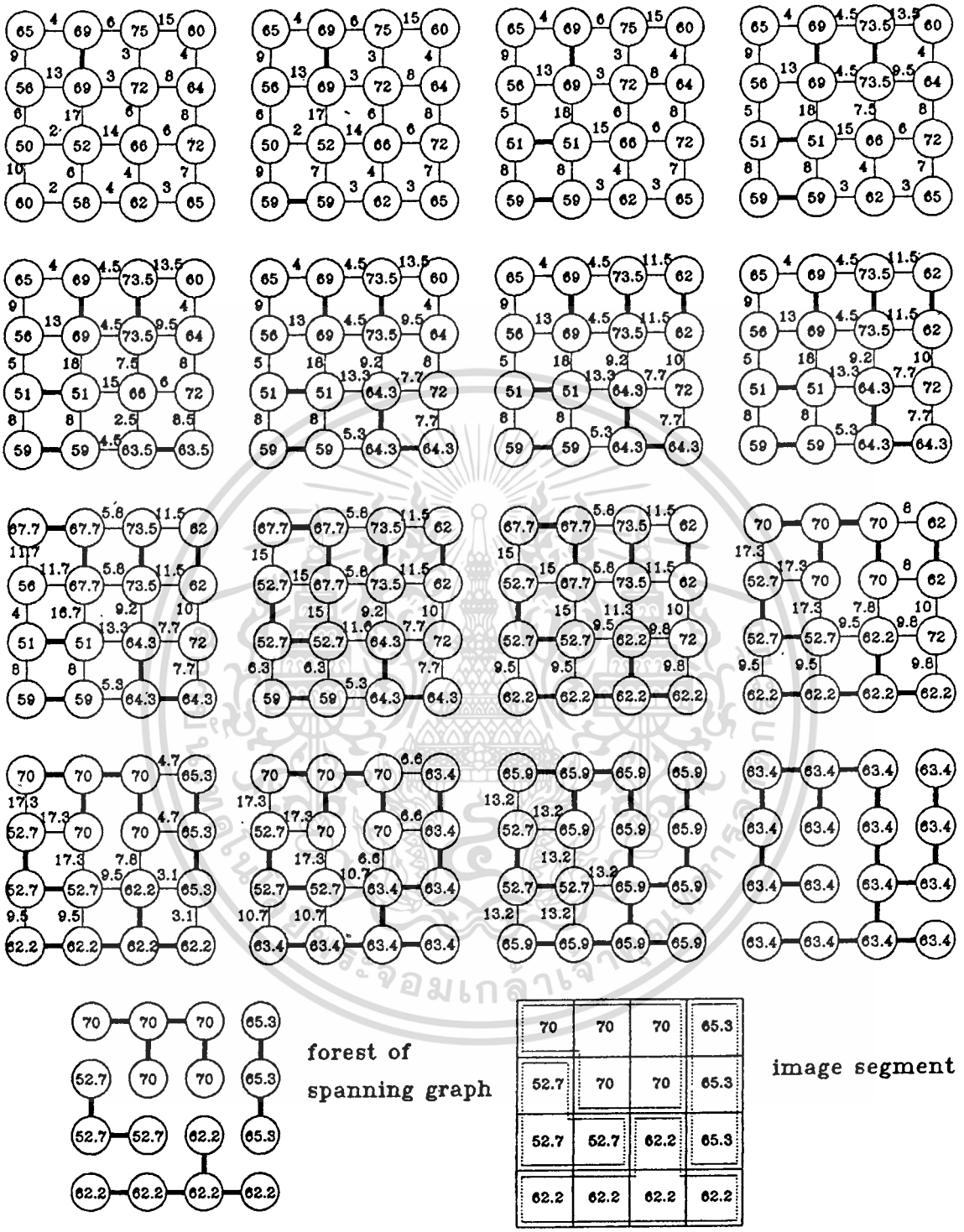
ลิ้งที่เชื่อมต่อจุดยอด V_i และ V_j จะถูกตัดออกและเก็บเอาไว้เพื่อนำมาใช้ในขั้นตอนการแปลงกราฟกลับเป็นภาพ ลิ้งอื่นทุก ๆ ตัวที่มีจุดยอดใกล้เคียงกับจุดยอด V_i และ V_j จะถูกคำนวนค่าของลิ้งเวทใหม่

ขั้นตอนและวิธีการของการทำเซกเม้นเตชั่นโดยใช้รีเคอร์ซีฟช็อตเตสท์สแพนนิ่งทรีของกราฟมีดังต่อไปนี้

1. แปลงข้อมูลภาพไปอยู่ในรูปจุดยอดของกราฟ
2. ในขณะที่มีจุดยอดของกราฟมากกว่าหนึ่งจุด
   3. หาค่าเวทของลิ้งที่มีค่าต่ำสุดตัวต่อไป
   4. เก็บลิ้งของตำแหน่งนั้นไว้
   5. รวมจุดยอดทั้งสองที่ต่อด้วยลิ้งตัวนี้
   6. คำนวณค่าเวทของจุดยอดใหม่และของลิ้งต่างๆ ที่เชื่อมต่อกับจุดยอดอันนี้
7. สร้างสแพนนิ่งทรีจากลิ้งที่ได้เก็บเอาไว้แล้ว
8. ทำการตัดสแพนนิ่งทรีของกราฟเป็นจำนวนเท่ากับ region ของภาพที่ต้องการลบหนึ่ง
9. แปลงกราฟกลับเป็นภาพเซกเม้นต์

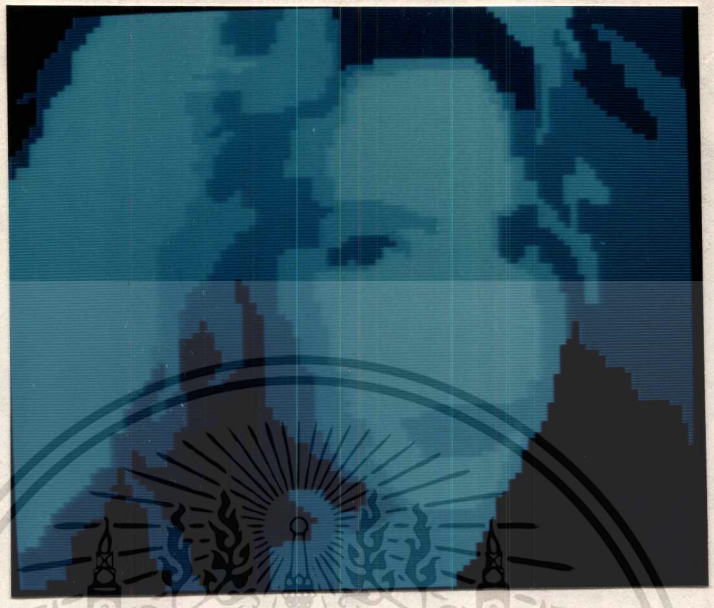
ลำดับขั้นตอนในการหา RSST ของกราฟ การตัด RSST ของกราฟ และการแปลงฟอเรสท์ของกราฟไปเป็นภาพเซกเม้นต์สามารถแสดงได้ดังรูปที่ 4.7 ส่วนรูปที่ 4.8 และรูปที่ 4.9 เป็นผลของการทำเซกเม้นต์จากข้อมูลภาพขนาด 200x200 โดยใช้วิธี RSST ของกราฟแบ่งจำนวนเซกเม้นต์เป็น 20 และ 100 เซกเม้นต์ตามลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.7 แสดงขั้นตอนการหา RSST ของกราฟ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.8 แสดงภาพเซกเม้นเตชั่นขนาด 20 เซกเม้นต์โดยใช้วิธี RSST

รูปที่ 4.9 แสดงภาพเซกเม้นเตชั่นขนาด 100 เซกเม้นต์โดยใช้วิธี RSST

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

การลดข้อมูลภาพโดยการเข้ารหัสพื้นที่ของภาพ

(Image Data Compression Via Image Segmented Coding)

การลดข้อมูลภาพเป็นวิธีการที่ถูกนำมาใช้กับข้อมูลภาพดิจิตอลอย่างกว้างขวาง นับแต่อดีต เนื่องจากข้อมูลภาพในระบบดิจิตอลนั้น แต่ละภาพจะมีขนาดใหญ่มาก ตัวอย่างเช่น ภาพขนาด 256x256 จุดภาพ ถ้าหากแต่ละจุดภาพให้ความแตกต่างของระดับสีเทา 256 ระดับ นั้นก็หมายความว่า แต่ละจุดภาพต้องใช้หน่วยความจำถึง 65536 ไบท์ หรือ 64 กิโลไบท์ ขนาดของข้อมูลภาพดังกล่าวไม่เพียงจะก่อให้เกิดปัญหาทางด้านหน่วยความจำเท่านั้น แต่จะยังก่อให้เกิดปัญหาเรื่องแบนด์วิดท์และเวลาที่ใช้ในการส่งข้อมูลภาพผ่านไปในระบบการสื่อสารด้วย ปัญหาหลังนี้ได้รับความสนใจอย่างกว้างขวางและถูกให้ความสำคัญสูงมาก ทั้งนี้เพราะระบบสื่อสารที่เราใช้กันอยู่มี capacity จำกัด ดังนั้นถ้าหากเราสามารถลดเวลาที่ใช้ในการรับส่งข้อมูลลงได้ก็เปรียบเสมือนเป็นการเพิ่ม capacity ให้กับระบบสื่อสารโดยตรง แนวทางในการแก้ปัญหาก็คือการทำการลดขนาดของข้อมูลภาพดิจิตอลลง แต่ยังคงรักษาคุณภาพของภาพทางด้านรับให้อยู่ในระดับความพอใจระดับหนึ่ง ในบทนี้จึงเป็นการเสนอวิธีการลดข้อมูลภาพดิจิตอลด้วยการเข้ารหัสภาพเซกเม้นเตชั่นซึ่งได้จากการแบ่งภาพออกเป็นเซกเม้นต์หรือ region จำนวนหนึ่ง โดยแต่ละเซกเม้นต์จะประกอบไปด้วยจุดภาพที่มีคุณลักษณะเหมือน ๆ กัน รอยต่อระหว่างเซกเม้นต์จะเป็นขอบต่าง ๆ ในภาพนั้นเอง

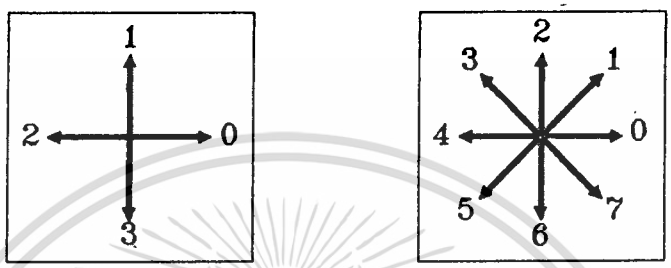
5.1 การเข้ารหัสภาพเซกเม้นเตชั่น (segmentation image coding)

ส่วนสำคัญของภาพเซกเม้นเตชั่นที่นำมาใช้ในการลดข้อมูลนั้นก็คือ ส่วนที่เป็นขอบของแต่ละเซกเม้นต์ โดยที่แต่ละเซกเม้นต์จะประกอบไปด้วยจุดภาพจำนวนหนึ่งและเมื่อรวมจุดภาพทั้งหมดของทุก ๆ เซกเม้นต์ก็จะมีค่าเท่ากับจุดภาพทั้งหมดของภาพเดิม ก่อนที่จะกล่าวถึงรายละเอียดของวิธีการเข้ารหัสภาพเซกเม้นต์ จะขอกล่าวถึงหลักการและรายละเอียดที่เกี่ยวข้องในการหาขอบของเซกเม้นต์ก่อน ทั้งนี้เนื่องจากการเรียงตัวของจุดภาพที่บริเวณขอบของเซกเม้นต์สามารถที่จะพิจารณาถึงลักษณะการเชื่อมต่อ (connectivity) ของจุดภาพได้เป็นสองแบบ ดังนั้นการติดตามหาขอบเซกเม้นต์ จึงสามารถกำหนดได้ 2 รูปแบบดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แบบที่ 1 เป็นแบบที่มี 4-Connectivity ซึ่งเราจะพิจารณาในรูปของทิศทางได้ 4 ทิศทาง สามารถแทนค่าตัวเลขให้แก่รหัสทิศทางได้เป็น 0,1,2 และ 3 ตามลำดับ ดังรูปที่ 5.1

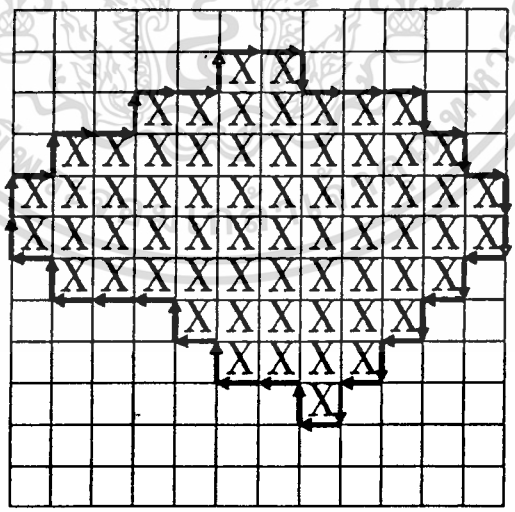
แบบที่ 2 เป็นแบบที่มี 8-Connectivity ซึ่งเราจะพิจารณาในรูปของทิศทางได้ 8 ทิศทาง สามารถแทนค่าตัวเลขให้แก่รหัสทิศทางได้เป็น 0,1,2,3,4,5,6และ 7 ตามลำดับ ดังรูปที่ 5.1



4 8

รูปที่ 5.1 แสดงทิศทางและค่ารหัสแบบ 4 ทิศทางและ 8 ทิศทาง

รูปแบบของการหาขอบทั้งแบบ 4 ทิศทาง และ 8 ทิศทาง เมื่อนำไปใช้ในการหาขอบของเซกเม้นต์จะให้จำนวนโค้ดที่แตกต่างกันกล่าวคือ แบบ 4 ทิศทางจะมีจำนวนรหัสมากกว่า ดังตัวอย่างในรูปที่ 5.2 และ 5.3

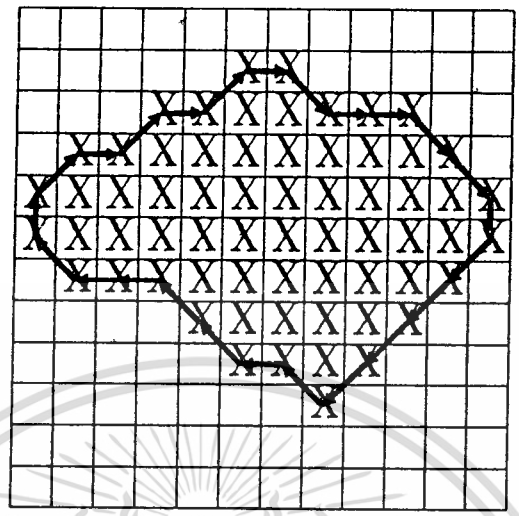


รูปที่ 5.2 แสดงตัวอย่างรหัสลูกโซ่ของเซกเม้นต์แบบ 4 ทิศทาง

แสดงการใช้รหัสแบบ 4 ทิศทาง ในการหาขอบของเซกเม้นต์ ซึ่งจะได้ค่ารหัสลูกโซ่เป็นดังนี้

00300030303323232323212212122212110100 1001

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.3 แสดงตัวอย่างรหัสลูกโซ่ของเชกเม้นต์แบบ 8 ทิศทาง

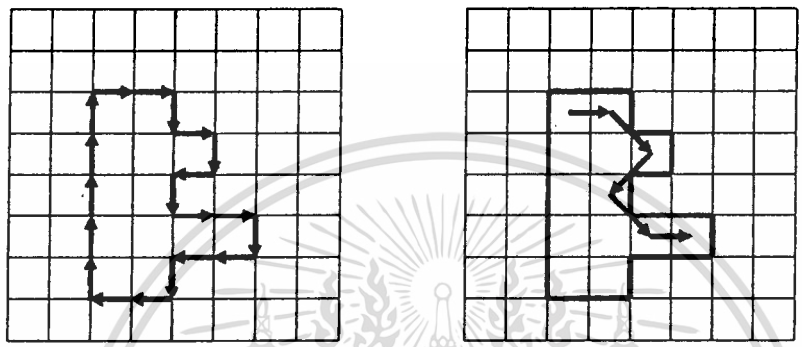
แสดงการใช้รหัสแบบ 8 ทิศทาง ในการหาขอบของเชกเม้นต์ ซึ่งจะได้ค่ารหัสลูกโซ่เป็นดังนี้

0700776555534334432101O1

จากรูปที่ 5.2 และ 5.3 จะเห็นว่าจำนวนรหัสของการตามขอบแบบ 8 ทิศทาง จะมีจำนวนน้อยกว่าการใช้รหัสแบบ 4 ทิศทาง และถึงแม้ว่าค่าของรหัสแบบ 8 ทิศทาง สามารถแทนด้วยค่าทางดิจิตอลขนาด 3 บิท ในขณะที่แบบ 4 ทิศทาง แทนด้วยค่าทางดิจิตอลขนาด 2 บิท แต่เมื่อรวมรหัสทั้งหมดการติดตามขอบแบบ 8 ทิศทาง ก็ยังมีจำนวนรหัสน้อยกว่าแบบ 4 ทิศทาง แต่ในวิทยานิพนธ์นี้เลือกใช้การต่อเนื่องของการหาขอบแบบ 4 ทิศทางในการเข้ารหัสขอบของเชกเม้นต์ เนื่องจากว่าภาพที่ได้จากการทำเชกเม้นเตชั่นบางภาพอาจจะให้ภาพที่มีลักษณะที่ไม่สามารถใช้การต่อเนื่องแบบ 8 ทิศทางได้ ดังตัวอย่างในรูปที่ 5.4 จากรูปจะเห็นว่าในกรณีที่ภาพมีส่วนที่ยื่นออกมามีลักษณะเป็นจุดภาพที่มีความหนาเพียง 1 จุดภาพ การใช้การต่อเนื่องแบบ 8 ทิศทางไม่สามารถที่จะเข้ารหัสให้ครบรอบ(Loop)ได้ ในขณะที่รหัสแบบ 4 ทิศทาง สามารถทำได้ และเมื่อรหัสลูกโซ่ของการติดตามขอบแบบ 4 ทิศทาง หนึ่งตัวสามารถใช้หน่วยความจำในการเก็บแค่เพียง 2 บิทดังนั้นหน่วยความจำขนาดหนึ่งไบท์จึงสามารถเก็บค่าของรหัสขอบได้ 4 ตัว ส่วนที่สองของรหัสคือค่าจุดเริ่มต้นของขอบที่เริ่มเข้ารหัส ซึ่งค่านี้จะนำไปใช้เป็นตัวกำหนดจุดเริ่มต้นของขอบอีกทีหนึ่งในขั้นตอนของการถอดรหัส โดยที่จุดเริ่มต้นนี้จะเก็บเป็นค่าตำแหน่งของจุดภาพว่าเป็นตำแหน่งที่เท่าไหร่ในอาเรย์ของภาพ ค่าตำแหน่งนี้จะใช้หน่วยความจำในการเก็บขนาด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2 ไบท์ และส่วนสุดท้ายของรหัสคือค่าของระดับสีเทา(ระดับความเข้ม)ของแต่ละเชกเม้นต์ ซึ่งมีค่าที่เป็นไปได้ 256 ระดับ ซึ่งจะใช้หน่วยความจำเพียงขนาดหนึ่งไบท์สำหรับแต่ละเชกเม้นต์



รูปที่ 5.4 แสดงข้อจำกัดของการใช้รหัสแบบ 8 ทิศทาง

เมื่อเราเลือกรูปแบบการติดตามขอบเป็นแบบ 4 ทิศทาง ต่อไปก็มาถึงขั้นตอนในการตัดสินใจของการตรวจสอบหาขอบของแต่ละเชกเม้นต์ เนื่องจากภาพหนึ่งๆ มีหลายเชกเม้นต์ โดยแต่ละเชกเม้นต์ก็จะมีค่าระดับสีเทาค่าหนึ่ง บางครั้งก็อาจจะมีหลายเชกเม้นต์ที่มีค่าระดับสีเทาเท่ากัน แต่เชกเม้นต์เหล่านี้จะไม่มีโอกาสอยู่ติดกัน ทั้งนี้เพราะถ้าอยู่ติดกันก็จะถูกรวมเข้าเป็นเชกเม้นต์เดียวกันตั้งแต่ในขั้นตอนของการทำเชกเม้นเตชั่นภาพ ขั้นตอนในการตรวจสอบหาขอบเชกเม้นต์เมื่อใช้การต่อเนื่องแบบ 4 ทิศทาง จะมีการพิจารณาได้เป็น 4 กรณี

โดยถ้ากำหนดให้

g คือค่าของระดับสีเทาของเชกเม้นต์ที่กำลังหาขอบ

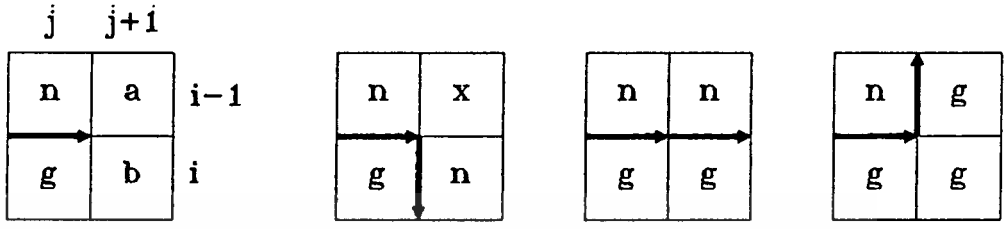
n คือค่าของระดับสีเทาของเชกเม้นต์อื่น ๆ

x คือค่าที่เราไม่สนใจว่าเป็นค่าอะไร

a,b คือค่าของระดับสีเทาที่เรายังไม่ทราบค่าและจะต้องตรวจสอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

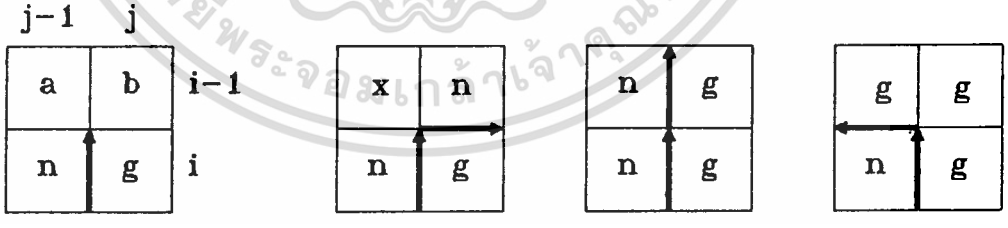
กรณีที่1 เป็นกรณีที่ค่ารหัสก่อนหน้ามีค่าเป็น " 0 " และตำแหน่งนั้นเป็นตำแหน่งที่ (i,j) กรณีที่เป็นไปได้ของรหัสตัวต่อไปคือ



รูปที่ 5.5 แสดงเงื่อนไขการหาขอบกรณีที่ 1

1. ตรวจสอบค่าที่ตำแหน่ง (i,j+1) ถ้าไม่เท่ากับ g รหัสตัวต่อไปคือ " 3 " ถ้าเท่ากับ g ไปตรวจสอบข้อ 2.
2. ตรวจสอบค่าที่ตำแหน่ง (i-1,j+1) ถ้าไม่เท่ากับ g รหัสตัวต่อไปคือ " 0 " ถ้าเท่ากับ g รหัสตัวต่อไปคือ " 1 "

กรณีที่2 เป็นกรณีที่ค่ารหัสก่อนหน้ามีค่าเป็น " 1 " และตำแหน่งนั้นเป็นตำแหน่งที่ (i,j) กรณีที่เป็นไปได้ของรหัสตัวต่อไปคือ



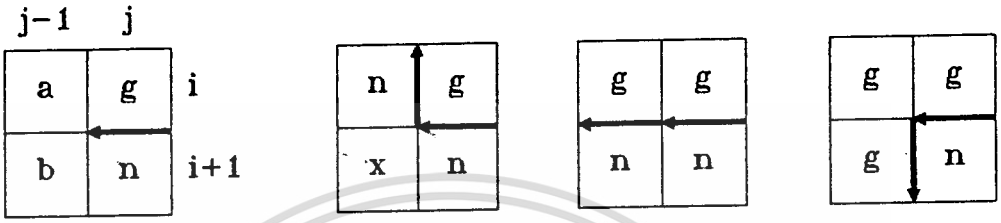
รูปที่ 5.6 แสดงเงื่อนไขการหาขอบกรณีที่ 2

1.ตรวจสอบค่าที่ตำแหน่ง (i-1,j) ถ้ามีค่าไม่เท่ากับ g รหัสตัวต่อไปก็คือ " 0 " ถ้ามีค่าเท่ากับ g ให้ไปตรวจสอบข้อ 2.
2. ตรวจสอบค่าที่ตำแหน่ง (i-1,j-1) ถ้าไม่เท่ากับ g รหัสตัวต่อไปคือ " 1 " ถ้าเท่ากับ g

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รหัสตัวต่อไปคือ " 2 "

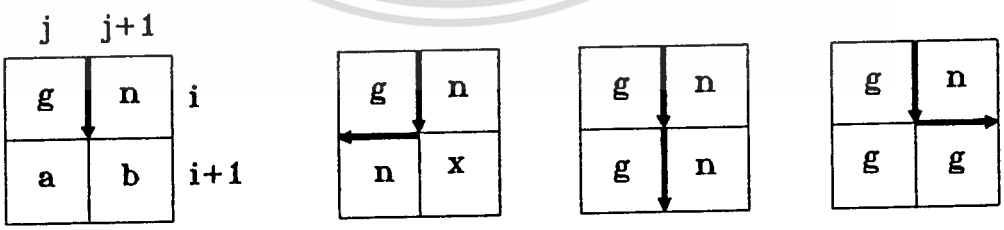
กรณีที่3 เป็นกรณีที่ค่ารหัสก่อนหน้ามีค่าเป็น " 2 " และตำแหน่งนั้นเป็นตำแหน่งที่ (i,j) กรณีที่เป็นไปได้ของรหัสตัวต่อไปคือ



รูปที่ 5.7 แสดงเงื่อนไขการหาขอบกรณีที่ 3

1. ตรวจสอบค่าที่ตำแหน่ง (i,j-1) ถ้าไม่เท่ากับ g รหัสตัวต่อไปคือ " 1 " ถ้าเท่ากับ g ไปตรวจสอบข้อ 2.
2. ตรวจสอบค่าที่ตำแหน่ง (i+1,j-1) ถ้าไม่เท่ากับ g รหัสตัวต่อไปคือ " 2 " ถ้าเท่ากับ g รหัสตัวต่อไปคือ " 3 "

กรณีที่4 เป็นกรณีที่ค่ารหัสก่อนหน้ามีค่าเป็น " 3 " และตำแหน่งนั้นเป็นตำแหน่งที่ (i,j) กรณีที่เป็นไปได้ของรหัสตัวต่อไปคือ



รูปที่ 5.8 แสดงเงื่อนไขการหาขอบกรณีที่ 4

1. ตรวจสอบค่าที่ตำแหน่ง (i+1,j) ถ้าไม่เท่ากับ g รหัสตัวต่อไปคือ " 2 " ถ้าเท่ากับ g

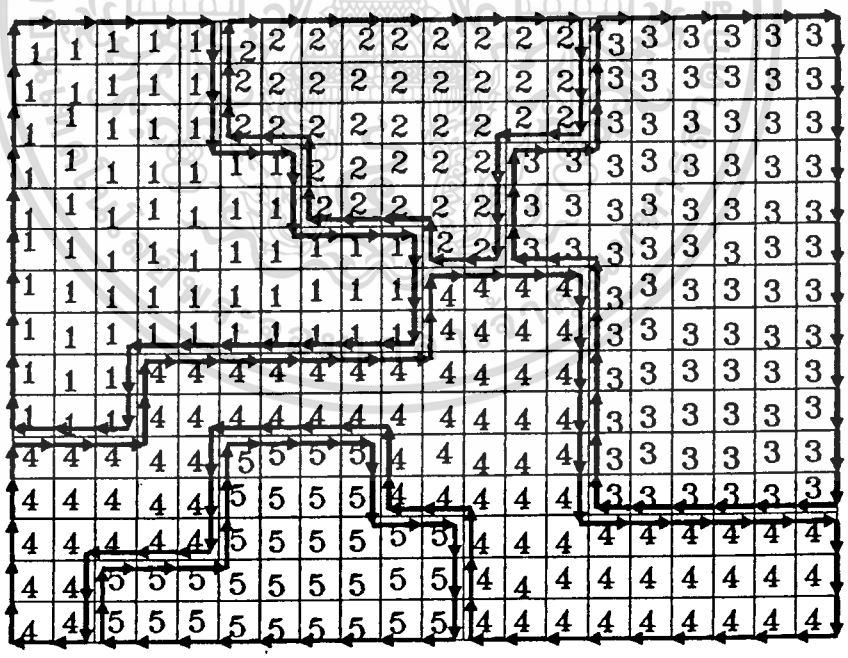
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไปตรวจสอบข้อ 2.

2. ตรวจสอบค่าที่ตำแหน่ง (i+1,j+1) ถ้าไม่เท่ากับ g รหัสตัวต่อไปคือ " 3 " ถ้าเท่ากับ g รหัสตัวต่อไปคือ " 0 "

5.1.1 การใช้เทคนิค CONTOUR CODING ในการเข้ารหัสขอบ

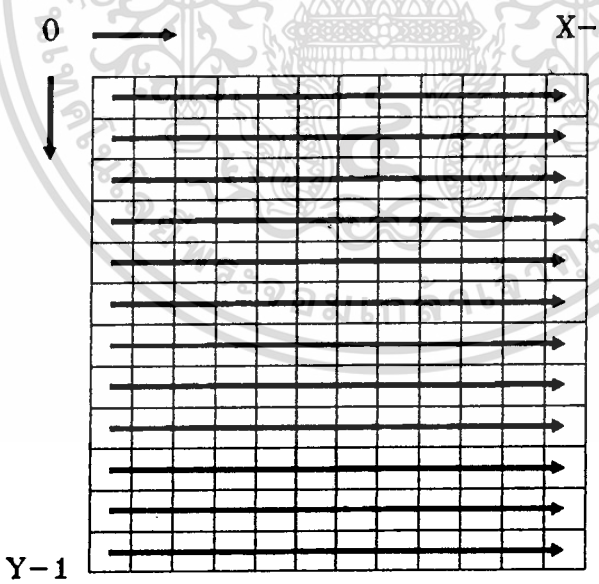
เทคนิค Contour Coding เป็นวิธีการเข้ารหัสขอบของบริเวณจุดภาพที่ค่าระดับสีเทามีค่าเป็นระดับเดียวกัน การเข้ารหัสจะทำการตรวจสอบส่วนขอบของบริเวณหรือ contour ของเซกเม้นต์ แล้วเปลี่ยนค่าเป็นรหัสลูกโซ่ตามทิศทางของขอบ การเข้ารหัสจะเริ่มต้นจากจุดบนสุดของเซกเม้นต์ แล้วเคลื่อนที่ไปในทิศทางตามเข็มนาฬิกา จนกว่าจะมาบรรจบพบกับจุดเริ่มต้นใหม่จากจุดภาพที่อยู่ภายในของเซกเม้นต์ ซึ่งถูกเข้ารหัสไว้แล้วนั้น จะถูกกำหนดค่าระดับสีเทาเอาไว้ด้วย โดยค่าระดับสีเทาจะมีค่าอยู่ระหว่าง 0-255 ระดับ ตัวอย่างของการเข้ารหัสของ contour coding ดังแสดงในรูปที่ 5.9



รูปที่ 5.9 แสดงตัวอย่างการเข้ารหัสขอบของเซกเม้นต์

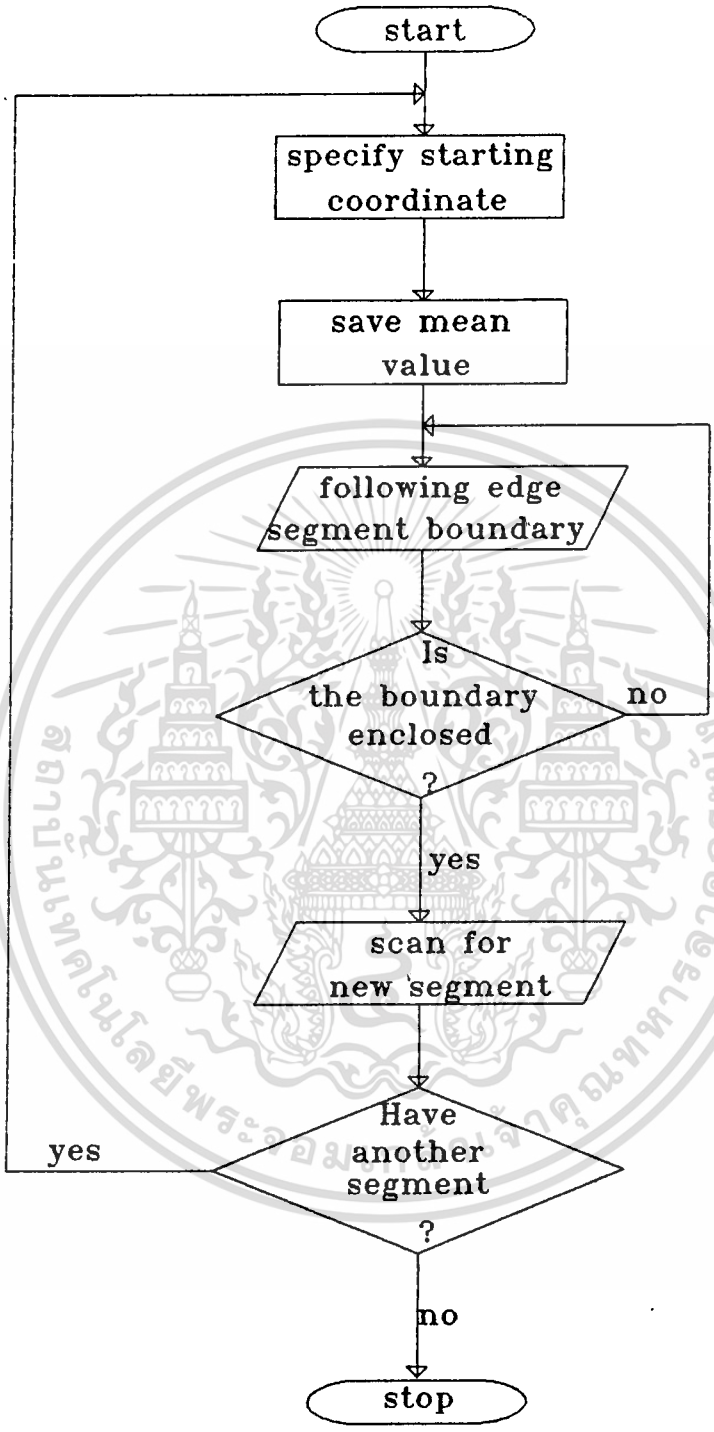
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การนำเทคนิคของ contour coding มาใช้ในการเข้ารหัสภาพเซกเม้นต์นั้น มีขบวนการคือทำการโหลดภาพลงบนอาเรย์ของบัฟเฟอร์ภาพ (image buffer) ที่มีขนาดเท่ากับ X x Y เมื่อ X เป็นจำนวนจุดภาพในหนึ่งสแกนไลน์ (scan line) และ Y เป็นจำนวนของสแกนไลน์ จากนั้นทำการสแกนหาจุดเริ่มต้นของเซกเม้นต์ โดยที่จุดเริ่มต้นของเซกเม้นต์แรกจะอยู่ที่ตำแหน่ง X = 0 และ Y = 0 เมื่อได้จุดเริ่มต้นจึงเริ่มหาขอบของเซกเม้นต์นั้นตามทิศทางของจุดภาพที่เป็นขอบตามรูปแบบการติดตามขอบ (edge following) ทั้ง 4 กรณีที่กล่าวมาแล้ว โดยที่ค่าของขอบจะอยู่ในลักษณะของรหัสลูกโซ่ การเข้ารหัสของขอบจะดำเนินการไปเรื่อย ๆ จนกระทั่งพบว่าจุดภาพสุดท้ายของขอบเป็นจุดเดียวกับจุดเริ่มต้น ค่ารหัสของขอบสำหรับเซกเม้นต์นี้จึงสมบูรณ์ แต่ในหนึ่งเซกเม้นต์จะต้องทำการเก็บค่าตำแหน่งเริ่มต้นของขอบและค่าระดับสีเทาด้วย เมื่อการเข้ารหัสของเซกเม้นต์แรกเสร็จ จะทำเครื่องหมายหรือมาร์คจุดภาพของเซกเม้นต์นี้เอาไว้ เพื่อกันการกลับมาเข้ารหัสซ้ำของเซกเม้นต์เดิม ต่อจากนั้นก็ทำการสแกนหาจุดเริ่มต้นของเซกเม้นต์ต่อ ๆ ไป โดยสแกนจากซ้ายไปทางขวาและจากบนลงล่าง แล้วจึงเข้ารหัสในทำนองเดียวกันตลอดจนครบทุกเซกเม้นต์ที่มีอยู่ในภาพนั้น รูปที่ 5.10 เป็นลักษณะการสแกนหาจุดเริ่มต้นของเซกเม้นต์ ส่วนรูปที่ 5.11 เป็นโฟลชาร์ทของการเข้ารหัสภาพเซกเม้นต์



รูปที่ 5.10 แสดงลักษณะการสแกนหาจุดเริ่มต้นของเซกเม้นต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.11 แสดงโฟลว์ชาร์ตการเข้ารหัสภาพเซกเม้นเตชั่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.2 การถอดรหัสภาพ (image decoding)

การสร้างภาพขึ้นใหม่ (reconstruction) ในขั้นตอนของการถอดรหัส (decode) นั้น จะนำเอาค่าจากข้อมูลทั้งสามส่วนคือ รหัสของขอบที่อยู่ในรูปของรหัสลูกโซ่ จุดเริ่มต้นของขอบ และค่าระดับความเข้มของเซกเม้นต์ที่ได้ในขั้นตอนการเข้ารหัสมาใช้ โดยนำค่าเหล่านี้มาสร้างเป็นภาพลงบนบัฟเฟอร์ที่เตรียมเอาไว้สำหรับการสร้างภาพขึ้นมาใหม่ สำหรับขั้นตอนในการสร้างภาพใหม่มีดังต่อไปนี้

1. นำเอาตำแหน่งจุดเริ่มต้นของเซกเม้นต์ โดยเริ่มจากเซกเม้นต์แรกคือค่า $x = 0$ และ $y = 0$ แล้วจึงใส่ค่าระดับความเข้มจุดภาพลงบนตำแหน่งนี้ของบัฟเฟอร์
2. นำเอาค่ารหัสขอบของเซกเม้นต์มาถอดรหัสให้เป็นค่าของตำแหน่งทิศทาง โดยที่ค่าของรหัสและทิศทางเป็นดังนี้

00	ไปทางขวา
01	ขึ้นข้างบน
10	ไปทางซ้าย
11	ลงข้างล่าง

3. จากค่าของตำแหน่งทิศทางในข้อ 2. นำมาสร้างเป็นขอบของเซกเม้นต์ลงบนบัฟเฟอร์ภาพ (image buffer) โดยใส่ขอบลงไปในตำแหน่งที่ต่อจากจุดเริ่มต้นตามทิศทางที่ได้จากรหัสเรื่อยไปจนกระทั่งครบรอบเซกเม้นต์ที่ตำแหน่งจุดเริ่มต้น ค่าระดับความเข้มหรือระดับสีเทาของขอบที่ใส่ลงไปบนบัฟเฟอร์ จะเป็นค่าเดียวกันกับค่าที่ใส่บนจุดเริ่มต้น ซึ่งเป็นค่าเฉลี่ยของเซกเม้นต์นั้นเอง
4. เมื่อเซกเม้นต์ทั้งเซกเม้นต์มีขอบครบสมบูรณ์ตามขั้นตอนในข้อ 3. ต่อไปเป็นการให้ค่าระดับสีเทากับทุก ๆ จุดภาพที่อยู่ภายในขอบนั้น โดยค่าระดับสีเทาเป็นค่าเดียวกันกับค่าระดับสีเทาที่ขอบของเซกเม้นต์ ในที่สุดก็จะได้เซกเม้นต์ที่สมบูรณ์เหมือนกับเซกเม้นต์ในภาพเดิม
5. ทำซ้ำขั้นตอนในข้อ 1.ถึง 4. จนกระทั่งครบทุกเซกเม้นต์ที่มีในภาพ เมื่อครบทุกเซกเม้นต์ก็จะได้ภาพเดิมที่สมบูรณ์กลับมา

เมื่อทำการถอดรหัสหรือการสร้างภาพขึ้นมาใหม่ได้เสร็จสมบูรณ์ทั้งภาพ ข้อมูลภาพที่ได้ซึ่งเป็นเอาท์พุทก็จะเก็บอยู่ในบัฟเฟอร์ และพร้อมที่จะนำไปใช้งานได้ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1.3 ผลการทดลอง

จากการทดลองโดยใช้ภาพในรูปที่ 5.12, 5.13, และ 5.14 ซึ่งเป็นภาพที่ใช้ในการทดลองที่มีขนาด 200x200 จุด โดยแต่ละจุดใช้หน่วยความจำขนาด 8 บิท ดังนั้นภาพ ๆ หนึ่งจึงใช้หน่วยความจำขนาด 40000 ไบท์ภาพทั้ง 3 จะถูกนำมาทำเซกเม้นเตชั่นเพื่อใช้ในการลดข้อมูล โดยภาพที่ 12(a), 13(a), และ 14(a) เป็นภาพต้นแบบ ส่วนรูปที่ 12(b), 13(b), 14(b) เป็นภาพจาก (a) นำมาทำเซกเม้นต์จำนวน 20 เซกเม้นต์ ส่วนรูปที่ 12(c), 13(c), 14(c) เป็นภาพจาก (a) นำมาทำเซกเม้นต์จำนวน 50 เซกเม้นต์ และรูปที่ 12(d), 13(d), 14(d) เป็นภาพจาก (a) นำมาทำเซกเม้นต์จำนวน 100 เซกเม้นต์ ผลการลดขนาดของข้อมูลของแต่ละภาพสามารถสรุปได้ดังตารางที่ 1, 2, และ 3 ตามลำดับ โดยที่อัตราบิทเรทของการลดขนาดของข้อมูลสามารถคำนวณได้ดังนี้

$$\frac{\text{จำนวนของข้อมูลรหัสภาพเซกเม้นต์ (bytes) X 8}}{\text{จำนวนจุดภาพทั้งหมดของภาพต้นแบบ}} = \text{จำนวนบิทต่อจุดภาพ(bits/pixel)}$$

รูปที่	5.12(a)	5.12(b)	5.12(c)	5.12(d)
จำนวน เซกเม้นต์	ต้นแบบ	20	50	100
จำนวน ข้อมูล	40000 Bytes	1892 Bytes	2904 Bytes	4123 Bytes
bits/pixel	8	0.378	0.580	0.824

ตารางที่ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่	5.13(a)	5.13(b)	5.13(c)	5.13(d)
จำนวน เซกเม้นต์	ต้นแบบ	20	50	100
จำนวน ข้อมูล	40000 Bytes	2103 Bytes	3270 Bytes	4168 Bytes
bits/pixel	8	0.420	0.654	0.837

ตารางที่ 2

รูปที่	5.14(a)	5.14(b)	5.14(c)	5.14(d)
จำนวน เซกเม้นต์	ต้นแบบ	20	50	100
จำนวน ข้อมูล	40000 Bytes	2012 Bytes	3130 Bytes	4319 Bytes
bits/pixel	8	0.402	0.626	0.864

ตารางที่ 3

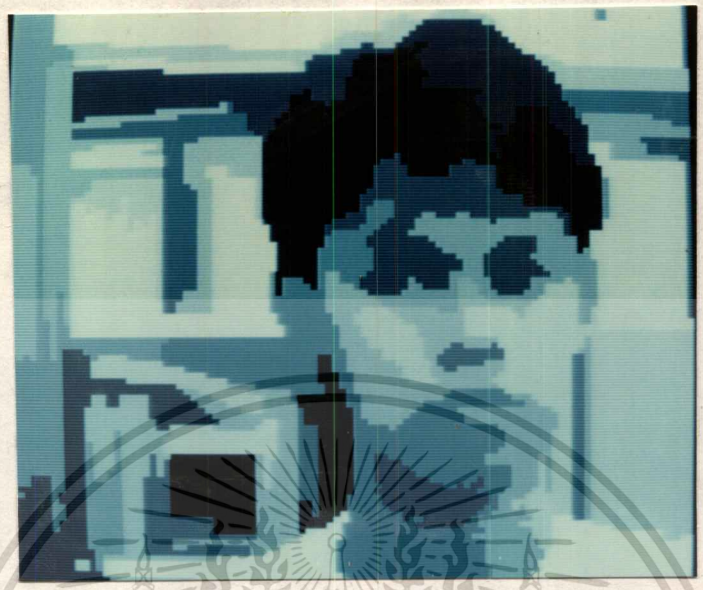
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(a) ภาพต้นแบบ

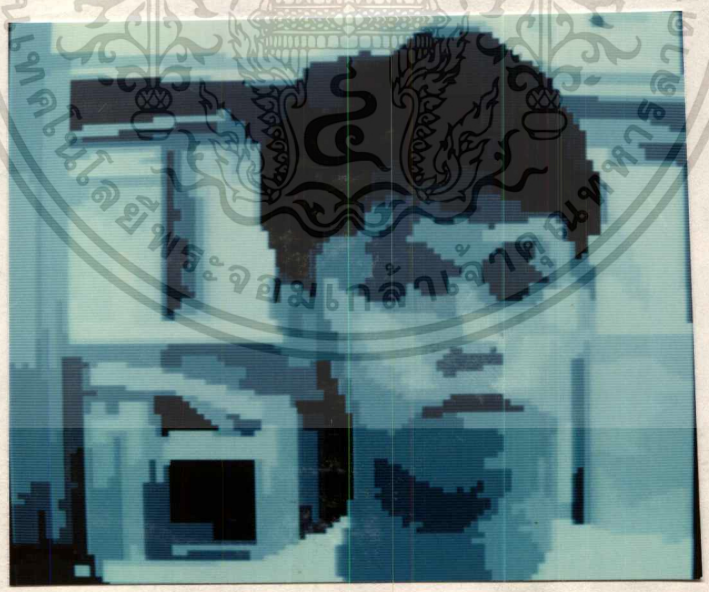
(b) ภาพที่แบ่งเป็น 20 เซกเม้นต์

รูปที่ 5.12 แสดงภาพทดสอบ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(c) ภาพที่แบ่งเป็น 50 เซกเม้นต์



(d) ภาพที่แบ่งเป็น 100 เซกเม้นต์

รูปที่ 5.12 (ต่อ) แสดงภาพทดสอบ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(a) ภาพต้นแบบ

(b) ภาพที่แบ่งเป็น 20 เซกเม้นต์

รูปที่ 5.13 แสดงภาพทดสอบ 2

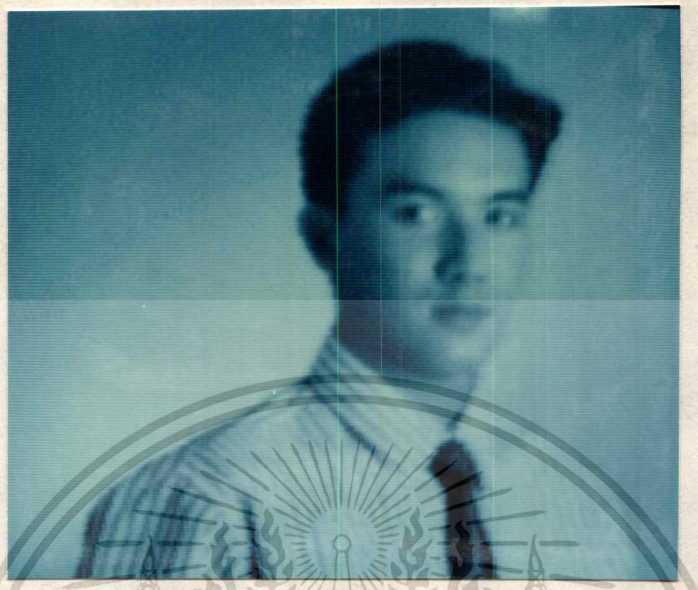
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(c) ภาพที่แบ่งเป็น 50 เซกเม้นต์

(d) ภาพที่แบ่งเป็น 100 เซกเม้นต์

รูปที่ 5.13 (ต่อ) แสดงภาพทดสอบ 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(a) ภาพต้นแบบ

(b) ภาพที่แบ่งเป็น 20 เซกเม้นต์

รูปที่ 5.14 แสดงภาพทดสอบ 3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(c) ภาพที่แบ่งเป็น 50 เซกเม้นต์

(d) ภาพที่แบ่งเป็น 100 เซกเม้นต์

รูปที่ 5.14 (ต่อ)แสดงภาพทดสอบ 3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2 การปรับปรุงเทคนิคการเข้ารหัสขอบเพื่อลดขนาดของข้อมูล

จากวิธีการเข้ารหัสภาพเซกเม้นต์ดังที่กล่าวมาแล้วในหัวข้อ 5.1 นั้นการเข้ารหัสจะทำในลักษณะที่แยกเซกเม้นต์ออกเป็นอิสระต่อกัน โดยที่แต่ละเซกเม้นต์จะมีจุดเริ่มต้นของตัวเอง และการเข้ารหัสของขอบที่ชิดกันจะมีการซ้ำซ้อนในลักษณะส่วนทาง (back trace) ซึ่งกันและกัน กล่าวคือส่วนที่อยู่ชิดติดกันจะทำการเก็บรหัสทั้งสองด้าน จะเห็นว่าเป็นการสิ้นเปลืองหรือเพิ่มจำนวนของรหัส ดังนั้นน่าจะมีวิธีการที่สามารถจะเข้ารหัสขอบของเซกเม้นต์ได้โดยไม่มีการเก็บซ้ำขอบส่วนที่ผ่านการเก็บรหัสไว้ก่อนหน้านั้นแล้ว ซึ่งจะทำให้สามารถลดจำนวนของรหัสลงไปได้อีก ในหัวข้อนี้ จึงเป็นวิธีการในการเข้ารหัสขอบของเซกเม้นต์ที่ไม่มีการเก็บขอบของแต่ละส่วนซ้ำกัน พร้อมทั้งยังสามารถลดจำนวนของจุดเริ่มต้นได้ด้วย กล่าวคือเราจะไม่เก็บทุก ๆ จุดเริ่มต้น ของเซกเม้นต์ในภาพ แต่จะเก็บเฉพาะจุดเริ่มต้นของเส้นขอบที่ไม่มีส่วนเชื่อมกับส่วนอื่น นั้นคือเส้นขอบ (line diagram) ของเซกเม้นต์ในแต่ละภาพที่มีส่วนที่ต่อถึงกันหรือเป็นขอบร่วม โดยกลุ่มที่ต่อถึงกันแต่ละกลุ่มจะมีจุดเริ่มต้นเพียงจุดเดียวเท่านั้นแทนที่จะเป็นทุก ๆ เซกเม้นต์ ส่วนมากแล้วภาพหนึ่ง ๆ จะมีขอบส่วนที่แยกออกไปเป็นกลุ่มย่อยหรือที่เรียกว่าเกาะ (island) ไม่กี่กลุ่ม ดังนั้นการเข้ารหัสด้วยวิธีนี้ จึงสามารถลดขนาดจำนวนของรหัสรวมที่ใช้ในการเก็บภาพได้มากกว่าวิธีแรก

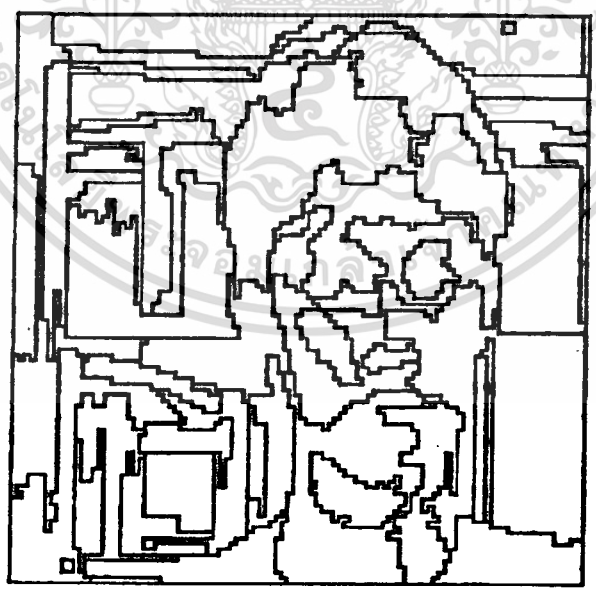
ขั้นตอนในการเข้ารหัสภาพเซกเม้นต์ แบบการเก็บรหัสขอบไม่ซ้ำกับขอบที่ได้เข้ารหัสไปแล้ว (non-back trace) นั้น จากคุณสมบัติของภาพที่ได้จากการทำเซกเม้นเตชั่นนั้นแต่ละเซกเม้นต์ของภาพจะมีค่าระดับความเข้มที่แตกต่างกัน ถึงจะมีบางเซกเม้นต์ที่มีค่าเท่ากันแต่ก็ไม่มีโอกาสอยู่ใกล้หรือชิดกัน ดังนั้นจึงเป็นการง่ายสำหรับการหาเส้นที่แบ่งขอบของเซกเม้นต์แต่ละเซกเม้นต์ในภาพให้ออกจากกันก่อน โดยทำการเปรียบเทียบระหว่างจุดภาพที่อยู่ใกล้เคียงกันจากซ้ายไปขวาและจากบนลงล่าง เมื่อเป็นค่าเดียวกันหรือเท่ากันก็แสดงว่าไม่ใช่เป็นขอบแต่เป็นส่วนเดียวกัน แต่ถ้าค่าจากการเปรียบเทียบต่างกันก็แสดงว่าจุดภาพสองจุดนั้น เป็นจุดภาพที่อยู่ต่างเซกเม้นต์กันหรือต่างส่วนออกไป จึงกำหนดให้เป็นขอบของเซกเม้นต์ทั้งสองที่แตกต่างกัน ในการหาขอบของเซกเม้นต์จะกระทำบนบัฟเฟอร์อาเรย์ที่มีขนาดใหญ่กว่าอาเรย์ของภาพ นั้นคือ ถ้าภาพมีขนาด X x Y บัฟเฟอร์อาเรย์จะมีขนาด (X+1)(Y+1) ที่เป็นเช่นนี้เพราะว่าเส้นขอบของเซกเม้นต์ไม่ได้อยู่ตรงกลางของจุดภาพแต่จะอยู่ที่ขอบของจุดภาพ โดยจุดภาพที่มุมทั้งสี่ไม่มีโอกาสที่จะเป็นค่าเริ่มต้นขอบของเซกเม้นต์ได้เลย

รูปที่ 5.15 เป็นตัวอย่างขอบของเซกเม้นต์ ที่ได้มาจากข้อมูลตัวอย่างในรูปที่ 5.9 ส่วนรูปที่ 5.16 เป็นขอบของเซกเม้นต์ที่ได้จากภาพขนาด 200 x 200 และมีจำเซกเม้นต์เท่ากับ 50

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

					X								X					X
					X								X					X
					X								X					X
					X	X	X					X	X	X				X
							X					X						X
							X	X	X	X		X						X
										X	X	X	X	X				X
										X				X				X
			X	X	X	X	X	X	X	X				X				X
			X											X				X
X	X	X	X		X	X	X	X	X					X				X
					X				X					X				X
					X				X	X	X			X	X	X	X	X
		X	X	X	X						X							X
		X									X							X
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

รูปที่ 5.15 แสดงขอบของเซกเม้นต์ที่ได้จากภาพ 5.9



รูปที่ 5.16 เป็นขอบของภาพเซกเม้นเตชั่นที่มีจำนวนเซกเม้นต์เท่ากับ 50

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เส้นแบ่งขอบเซกเม้นต์ซึ่งต่อไปจะเรียกว่าไลน์ไดอะแกรม (line diagram) ซึ่งมีลักษณะเป็นเส้นที่มีขนาดความหนาเพียงหนึ่งจุดภาพเท่านั้น และจากไลน์ไดอะแกรมที่ได้จากขั้นตอนแรกนั้นสามารถที่จะนำมาเข้ารหัสได้โดยง่าย เมื่อกำหนดให้ทิศทางของไลน์ไดอะแกรมเป็นแบบ 4 ทิศทาง นั้นคือ ซ้าย ขวา บน และล่าง ตามลำดับ ดังนั้นส่วนต่างของรหัสจะประกอบไปด้วย

1. ความยาวของไลน์ไดอะแกรมที่เป็นเส้นตรง (run length)
2. ทิศทางของไลน์ไดอะแกรม (sequence of run length) ซึ่งสามารถกำหนดให้เป็นได้ 2 กรณีสำหรับการต่อเนื่องแบบ 4 ทิศทาง นั้นคือซ้ายกับขวาดังรูป

รูปที่ 5.17 แสดงทิศทางที่เป็นไปได้ของ sequence of run length

3. รหัสของจุดแยกซึ่งมีอยู่ด้วยกัน 4 กรณี ซึ่งจุดแยกที่เกิดขึ้นนี้เนื่องจากเซกเม้นต์ในภาพ 3 หรือ 4 เซกเม้นต์อยู่ติดกัน ลักษณะของจุดแยกแสดงในรูปที่ 5.24

กรณีที่ 1 จุดแยก ขวา - ตรงไป (Right-Forward) ให้รหัสเป็น 0
กรณีที่ 2 จุดแยก ตรงไป - ซ้าย (Forward-Left) ให้รหัสเป็น 1
กรณีที่ 3 จุดแยก ขวา - ซ้าย (Right-Left) ให้รหัสเป็น 2
กรณีที่ 4 จุดแยก ขวา-ตรงไป-ซ้าย (Right-Forward-Left) ให้รหัสเป็น 3

จากจุดแยกที่เป็นไปได้ทั้ง 4 นี้ ทำให้ต้องใช้รหัสจำนวน 2 บิทในการจำแนกทิศทางดังกล่าว

เมื่อกำหนดรหัสของแต่ละส่วนสำหรับไลน์ไดอะแกรมที่เป็นไปได้แล้ว ขั้นตอนต่อไปจึงเป็นการเข้ารหัสไลน์ไดอะแกรมของภาพเซกเม้นต์ การเข้ารหัสเริ่มโดยการสแกนจากทางซ้ายมือไปขวามือ และจากบนลงล่างเพื่อหาจุดเริ่มต้นของไลน์ไดอะแกรม เมื่อพบจุดเริ่มต้นจะเก็บตำแหน่งของจุดนี้เป็นรหัสตัวหนึ่ง ต่อจากนั้นจึงทำการเข้ารหัสของไลน์ไดอะแกรม โดยรหัสจะเก็บเป็นความยาวของเส้นตรง แล้วจึงตามด้วยค่าทิศทางการเลี้ยวของเส้นตรง ซึ่งมีสองกรณีก็คือจากทิศทางปัจจุบันแล้วเลี้ยวซ้ายกับเลี้ยวขวา รหัสอันนี้ใช้ข้อมูลเพียงบิทเดียวเท่านั้น โดยค่า "0" ให้เป็นการเลี้ยวไปทางขวา ส่วนค่า "1" ให้เป็นการเลี้ยวไปทางซ้าย และเมื่อเข้ารหัสไปเจอกับจุดแยกที่ไม่ใช่จุดเลี้ยวธรรมดาก็จะมีรหัสของจุดแยกที่แตกต่างกันไปตามกรณีของการแยก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

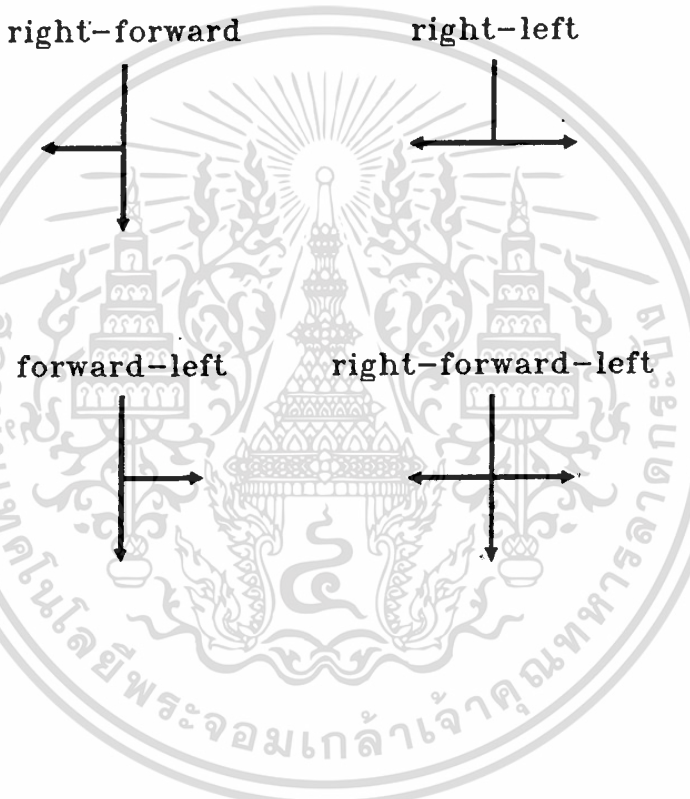
ซึ่งมี 4 กรณีดังรูปที่ 5.18 ดังนั้นจึงใช้ข้อมูลจำนวน 2 บิทเพื่อแทนค่าของรหัสทั้ง 4 คือ

00 แทนการแยก ขวา - ตรงไป

01 แทนการแยก ตรงไป - ซ้าย

10 แทนการแยก ซ้าย - ขวา

11 แทนการแยก ขวา - ตรงไป - ซ้าย



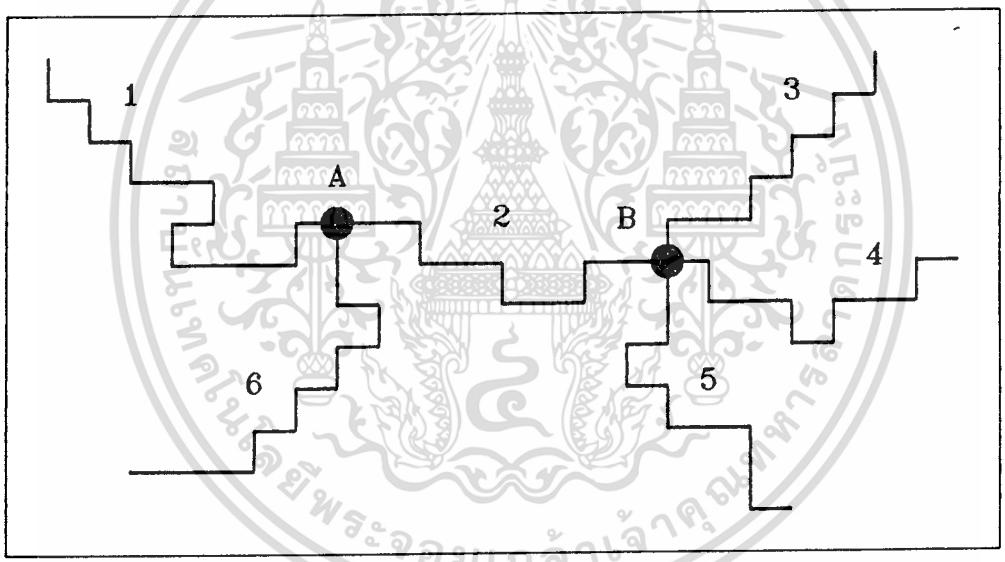
รูปที่ 5.18 แสดงกรณีของจุดแยกที่เป็นไปได้

เมื่อเป็นจุดแยกส่วนต่อไปของไลน์ไดอะแกรมที่จะทำการเข้ารหัส จะเป็นเส้นที่อยู่ทางซ้ายสุดของจุดแยก ส่วนเส้นที่เหลือจะเก็บลงสแต็ค (stack) โดยเก็บเส้นที่อยู่ทางขวาสุดของจุดแยกก่อนแล้วจึงไล่มาตามลำดับ ส่วนเส้นที่กำลังเข้ารหัสอยู่จะสิ้นสุดเมื่อเส้นนั้นไปเจอขอบของภาพหรือจุดแยกจุดต่อไป หรือกลับมาที่จุดแยกเดิมในกรณีที่เส้นนั้นเป็นวง (loop) เมื่อเจอขอบภาพไลน์ไดอะแกรมเส้นต่อไปก็จะถูกอ่านออกมาจากสแต็ค ค่าของเส้นที่อ่านออกมานี้จะเป็นเส้นที่อยู่ทางซ้ายสุดที่อยู่ในสแต็ค เพราะการเก็บของสแต็คจะมีลักษณะของการเก็บแบบเข้าก่อนออกทีหลัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(first in last out) ดังนั้นเมื่อเราเก็บเส้นขวาสุดก่อน ในตอนที่อ่านออกมาจะเป็นเส้นซ้ายสุด การเข้ารหัสจะดำเนินการไปเรื่อย ๆ ถ้าหากเจอจุดแยกก็จะเก็บลงสแต็ค เมื่อสิ้นสุดการเข้ารหัสแต่ละเส้นก็จะอ่านออกมาใหม่จากสแต็คจนกว่าค่าในสแต็คจะเป็นศูนย์ จากนั้นจึงทำการสแกนหาค่าจุดเริ่มต้นจุดต่อไป เมื่อเจอจุดเริ่มต้นของไลน์ไดอะแกรมเส้นใหม่ก็จะทำการเข้ารหัสในทำนองเดียวกัน การสแกนหาจุดเริ่มต้นของไลน์ไดอะแกรมจะไม่มีโอกาสซ้ำเส้นเดิมเพราะว่าไลน์ไดอะแกรมที่ผ่านการเข้ารหัสแล้วจะถูกทำเครื่องหมายเอาไว้หรือไม่ก็ทำลายไปเสีย เพื่อป้องกันการสแกนกลับมาเจอซ้ำ การเข้ารหัสจะสิ้นสุดลงเมื่อการสแกนหาจุดเริ่มต้นของไลน์ไดอะแกรมในภาพแล้วไม่มีส่วนใดเหลืออยู่อีก

รูปที่ 5.19 เป็นตัวอย่างลำดับการเข้ารหัสของเส้นไดอะแกรมที่มี จุดแยก 2 จุด และมีจำนวนเส้นไดอะแกรมทั้งหมด 6 เส้น



รูปที่ 5.19 แสดงตัวอย่างลำดับการเข้ารหัสของไลน์ไดอะแกรม

จากรูปที่ 5.19 ในการเข้ารหัสจะเริ่มทำการสแกนเพื่อหาจุดเริ่มต้นของไลน์ไดอะแกรม โดยสแกนจากซ้ายไปทางขวา ซึ่งจะพบจุดเริ่มต้นเป็นส่วนที่ 1 ของไลน์ไดอะแกรม แล้วจึงทำการเข้ารหัสของส่วนนี้ โดยเก็บเป็นค่าความยาวและทิศทางของแต่ละส่วนของเส้น จนกระทั่งถึงจุดแยก A จึงตรวจสอบว่าเป็นจุดแยกแบบใด ซึ่งในรูปจุด A จะเป็นกรณีที่ 1 คือ ขวาและตรงไป ตำแหน่งจุดแยกนี้แทนด้วยรหัส 00 แล้วจึงทำการเก็บส่วนที่ 6 ซึ่งเป็นส่วนขวาสุดของจุดแยกลงสแต็ค ต่อจากนั้นก็ทำการเข้ารหัสส่วนที่ 2 ต่อไป จนกระทั่งถึงจุดแยก B เมื่อตรวจสอบแล้วได้

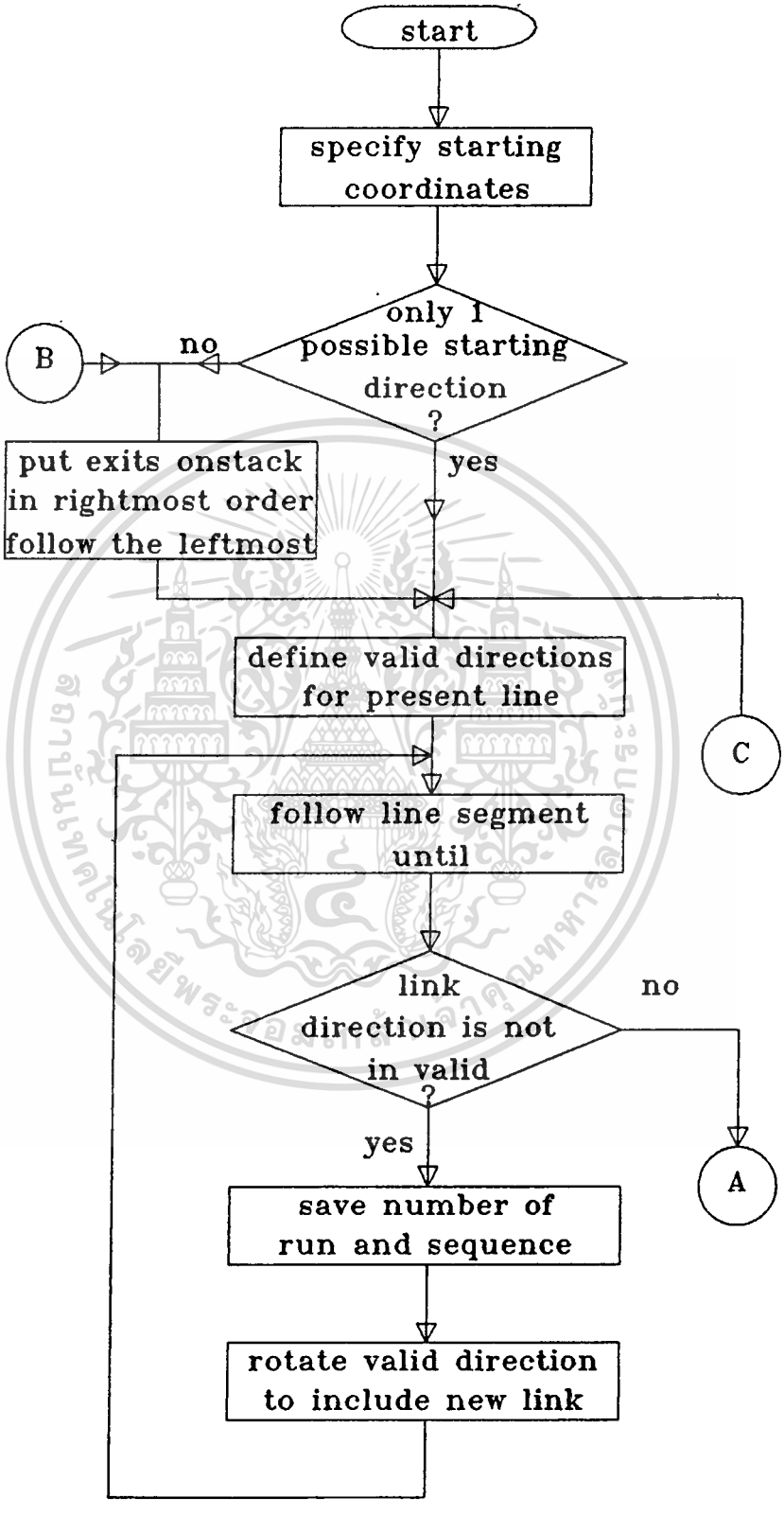
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นจุดแยกในกรณีที่ 4 คือแยก ซ้าย - ตรงไป - ขวา จึงแทนด้วยรหัส 11 จากนั้นทำการเก็บส่วนไลน์ไดอะแกรมที่แยกออกจากจุด B ลงสแตคโดยเริ่มจากส่วนที่ 5 ซึ่งเป็นขวาสุด แล้วตามด้วยส่วนที่ 4 การเข้ารหัสส่วนที่ 3 จะเป็นขั้นตอนสุด ขั้นต่อไปทำการอ่านค่าจากสแต็คออกมาหนึ่งตัว ซึ่งจะเป็นส่วนที่ 4 เมื่อเข้ารหัสส่วนที่ 4 จนเสร็จ ค่าใหม่จากสแต็คถูกอ่านออกมาอีกคือส่วนที่ 5 เมื่อส่วนที่ 5 เข้ารหัสจนเสร็จ ค่าใหม่จากสแต็คก็ถูกอ่านออกมาอีก ค่าใหม่นี้จะเป็นส่วนที่ 6 ซึ่งเป็นส่วนสุดท้าย เมื่อเข้ารหัสส่วนนี้เรียบร้อยก็จะทำการตรวจสอบดูว่าสแตคยังเหลือส่วนอื่น ๆ ที่เก็บอยู่อีกหรือไม่ ถ้ามีก็เอามาเข้ารหัสต่อไป แต่ในรูปตัวอย่างไม่มีส่วนอื่นอีกต่อไป ดังนั้นจึงทำการสแกนหาจุดเริ่มต้นในส่วนต่อไป แต่ในตัวอย่างนี้มีอยู่ส่วนเดียว และทุกส่วนได้ทำการเข้ารหัสหมดแล้วจึงเป็นการสิ้นสุดการเข้ารหัสขอบของภาพเซกเม้นต์

ส่วนค่าระดับสีเทาของแต่ละเซกเม้นต์ นั้นสามารถกำหนดได้โดยข้อมูลขนาด 8 บิท เพราะความเข้มของข้อมูลภาพอยู่ในช่วง 0 - 255 การกำหนดค่าระดับสีเทาแต่ละเซกเม้นต์ จะเริ่มจากเซกเม้นต์แรกที่อยู่บนสุดด้านซ้ายของภาพ และเซกเม้นต์ต่อ ๆ มาจะเรียงจากซ้ายไปขวาจากบนลงล่างตามลำดับ โดยเมื่อสแกนพบเซกเม้นต์ใหม่ก็จะเก็บค่าระดับสีเทาของเซกเม้นต์นั้นเอาไว้ แล้วจึงทำเครื่องหมาย (mark) เซกเม้นต์นั้นเอาไว้เพื่อไม่ให้กลับมาซ้ำเดิมอีก ดังนั้นจำนวนของเซกเม้นต์มีเท่าไหร่จำนวนของรหัสค่าระดับสีเทาของเซกเม้นต์จะมีเท่านั้นเช่น ถ้าหากภาพมีจำนวนเซกเม้นต์ทั้งหมด 100 เซกเม้นต์ จำนวนรหัสที่ใช้เก็บค่าระดับสีเทาของภาพก็จะเป็น 100 ไบท์ด้วย

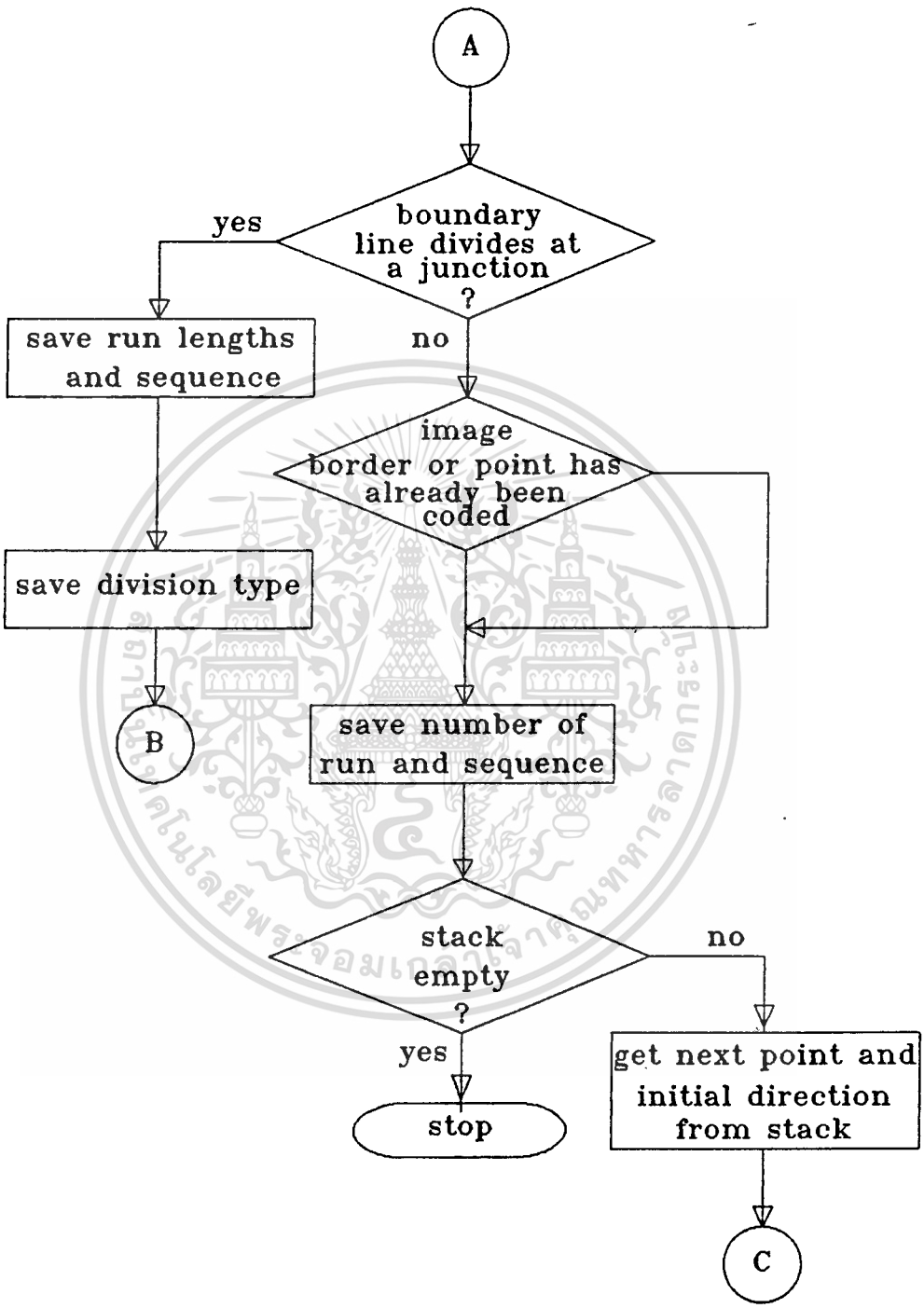
รูปที่ 5.20 แสดงโฟลว์ชาร์ทของขั้นตอนการเข้ารหัสภาพเซกเม้นเตชั่นแบบ non-back trace

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.20 แสดงโฟลว์ชาร์ทการเข้ารหัสขอบแบบ non-back trace

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.20 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2.1 การถอดรหัสภาพและผลการทดลอง

การถอดรหัสภาพจะทำบนอาเรย์ที่เป็นบัฟเฟอร์ โดยอาเรย์จะมีขนาดเท่ากับขนาดของภาพ และรหัสของขอบต่างๆ ซึ่งเป็นข้อมูลที่จะนำมาใช้สร้างภาพเดิมกลับคืนมานั้น ในขั้นแรกจะนำเอารหัสมาเปลี่ยนเป็นค่าความยาวและทิศทางของไลน์ไดอะแกรมขอบเซกเม้นต์ แล้วจึงเขียนลงบนบัฟเฟอร์ เมื่อสร้างเป็นไลน์ไดอะแกรมของเซกเม้นต์เสร็จสมบูรณ์ทั้งภาพ ภาพที่ได้ในตอนนี้จะยังไม่สมบูรณ์เพราะยังไม่ได้ใส่ค่าระดับสีเทาให้กับแต่ละเซกเม้นต์ ดังนั้นขั้นตอนต่อไปจึงเป็นการใส่ค่าระดับสีเทาให้กับแต่ละเซกเม้นต์ จากรหัสของระดับสีเทาซึ่งในตอนเข้ารหัสนั้นเก็บจากเซกเม้นต์บนสุดทางซ้ายเรียงไปทางขวาและจากบนลงล่าง ดังนั้นในขั้นตอนของการสร้างภาพใหม่จึงทำในลักษณะเดียวกัน คือทำการใส่ค่าระดับสีเทาจากเซกเม้นต์ด้านซ้ายบนสุดไล่ไปทางขวาและจากบนลงล่าง จนครบทุกเซกเม้นต์ ภาพที่ได้ก็จะสมบูรณ์สามารถนำไปใช้งานได้

ภาพที่ได้จากการสร้างภาพใหม่จากขั้นตอนการลดข้อมูลนั้น จะมีลักษณะเหมือนกับภาพก่อนการเข้ารหัสทุกประการ ผลที่ได้จากการลดขนาดข้อมูลโดยการเข้ารหัสแบบไม่เก็บซ้ำขอบเดิมนี้จะช่วยลดขนาดข้อมูลลงไปอีก ดังจะเห็นได้จากตารางที่ 4, 5, และ 6 เมื่อเปรียบเทียบกับตารางที่ 1, 2, และ 3 ตามลำดับ

รูปที่	5.12(a)	5.12(b)	5.12(c)	5.12(d)
จำนวนเซกเม้นต์	ต้นแบบ	20	50	100
จำนวนข้อมูล	40000 Bytes	668 Bytes	1163 Bytes	1691 Bytes
bits/pixel	8	0.134	0.233	0.358

ตารางที่ 4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่	5.13(a)	5.13(b)	5.13(c)	5.13(d)
จำนวน เซกเม้นต์	ต้นแบบ	20	50	100
จำนวน ข้อมูล	40000 Bytes	994 Bytes	1676 Bytes	2186 Bytes
bits/pixel	8	0.203	0.345	0.457

ตารางที่ 5

รูปที่	5.14(a)	5.14(b)	5.14(c)	5.14(d)
จำนวน เซกเม้นต์	ต้นแบบ	20	50	100
จำนวน ข้อมูล	40000 Bytes	1067 Bytes	1762 Bytes	2551 Bytes
bits/pixel	8	0.217	0.362	0.530

ตารางที่ 6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

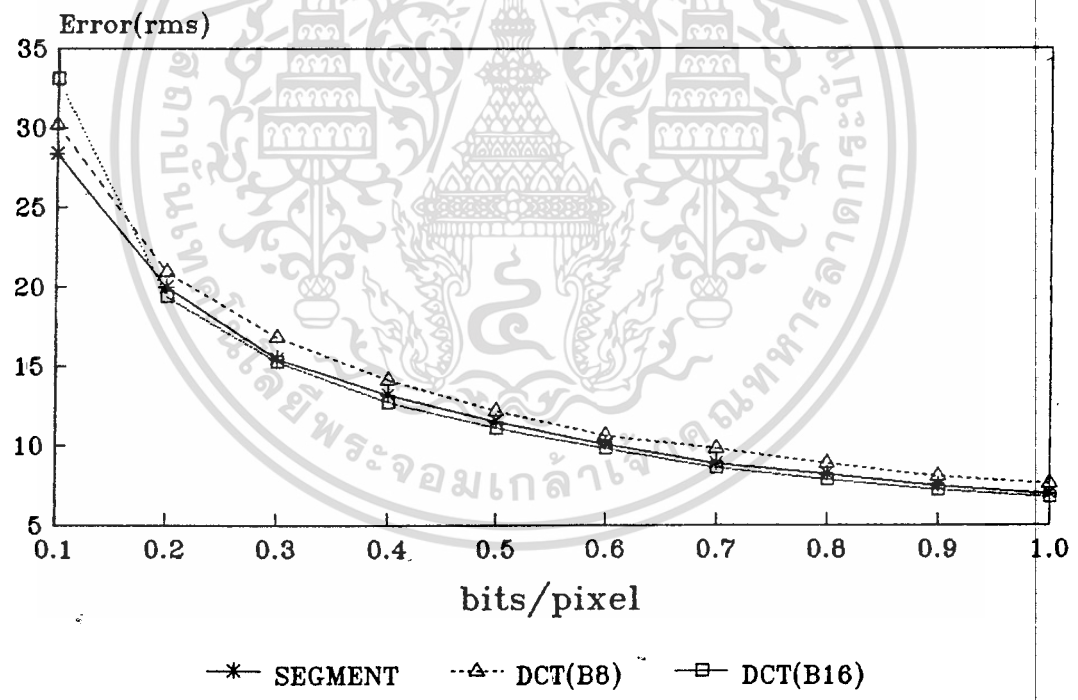
บทที่ 6
สรุป

6.1 สรุปผลการวิจัย

การลดข้อมูลภาพโดยการเข้ารหัสพื้นที่ของภาพ เป็นการลดข้อมูลโดยการนำภาพที่ผ่านการทำเซกเม้นเตชั่นมาเข้ารหัสพื้นที่ของแต่ละเซกเม้นต์ ดังนั้นภาพที่ได้หลังจากการถอดรหัสจะยังคงเป็นภาพที่มีความคมชัดของขอบอยู่เช่นเดิม โดยความคมชัดและคุณภาพของภาพที่ได้จะขึ้นอยู่กับจำนวนเซกเม้นต์ที่ถูกแบ่งภายในแต่ละภาพ รวมถึงวิธีการที่นำมาใช้ในการทำเซกเม้นเตชั่น ในวิทยานิพนธ์นี้ได้เลือกวิธีของทฤษฎีกราฟที่เรียกว่า Recursive shortest spanning tree (RSST) มาใช้ในการทำอิมเมจเซกเม้นเตชั่น ซึ่งเป็นวิธีที่ให้ภาพเอาท์พุตเซกเม้นต์ที่มีลักษณะการแบ่งรายละเอียดในภาพได้ใกล้เคียงข้อมูลภาพเดิมมากที่สุด ทั้งที่มีจำนวนเซกเม้นต์น้อย ๆ เมื่อเทียบกับวิธีอื่นๆ ทั้งนี้เพราะมีการใช้ข้อมูลส่วนใหญ่ของภาพในการตัดสินการแบ่งแยกแต่ละส่วนของภาพออกจากกัน และยังสามารถที่จะกำหนดจำนวนของเซกเม้นต์ที่ต้องการจะแบ่งได้แน่นอนตามที่กำหนด ซึ่งผลที่ได้นี้เมื่อนำมาใช้ในการเข้ารหัสเพื่อลดขนาดของข้อมูลภาพสามารถที่จะทำให้จำนวนข้อมูลที่ได้หลังการเข้ารหัสมีค่าต่ำๆ ได้ดี เพราะจำนวนของข้อมูลที่ได้ขึ้นอยู่กับจำนวนของเซกเม้นต์ภายในภาพ วิธีการที่นำมาใช้ในการเข้ารหัสขอบของเซกเม้นต์ได้ใช้รหัสลูกโซ่ (chain code) และรหัสความยาว (run length code) โดยนำมาใช้กับวิธีแบบ contour และแบบ non-back trace ตามลำดับ โดยวิธีของ contour coding นั้นจะเข้ารหัสขอบของเซกเม้นต์แยกอิสระออกไปแต่ละเซกเม้นต์ ซึ่งเซกเม้นต์หนึ่งๆ จะมีจุดเริ่มต้นและจุดสิ้นสุดเป็นของตัวเอง ส่วนวิธีการเข้ารหัสแบบ non-back trace coding นั้นจะทำการหาส่วนที่เป็นขอบของเซกเม้นต์ในภาพออกมาก่อนโดยเส้นที่ได้จะเป็นเส้นที่แบ่งแต่ละเซกเม้นต์ที่อยู่ติดกันออกจากกันและมีความหนาเพียงหนึ่งจุดภาพเท่านั้น ดังนั้นเมื่อทำการเข้ารหัสเส้นแบ่งขอบจึงไม่มีการเข้ารหัสส่วนทางกันเหมือนกับวิธีแรก จำนวนของรหัสที่ได้ของการเข้ารหัสแบบ non-back trace จึงมีจำนวนน้อยกว่าวิธีแรกแต่จะใช้เวลามากกว่า เนื่องจากต้องหาขอบของเซกเม้นต์ก่อน และในตอนที่เข้ารหัสยังมีขั้นตอนที่ยุ่งยากกว่า เพราะต้องมีการเก็บส่วนแยกของเส้นขอบลงสแต็คก่อน เมื่อการเข้ารหัสมาเจอจุดแยกของขอบ แต่อย่างไรก็ตามถึงแม้จะเสียเวลาไปบ้างผลที่ได้นั้นสามารถที่จะลดขนาดของข้อมูลได้มากกว่าวิธีแรกอยู่มากคือ จากผลที่ได้เมื่อเปรียบกันแล้ววิธีการเข้ารหัสขอบแบบ non-back trace จะได้ค่ารหัสน้อยกว่าแบบ contour ประมาณ 50 เปอร์เซนท์ ดังนั้นในการที่จะนำวิธีการลดข้อมูลภาพด้วยการเข้ารหัสพื้นที่ของภาพไปใช้งานต่างๆ โดยใช้วิธีของการเข้ารหัสขอบแบบ non-back trace จะสามารถลดจำนวนข้อมูลได้มากกว่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อัตราบิทเรทของการลดข้อมูลที่ได้หลังจากการเข้ารหัสตามวิธีที่เสนอนี้ จะขึ้นอยู่กับจำนวนของเซกเม้นต์ที่ถูกแบ่งภายในภาพว่ามากน้อยเพียงใด ถ้าเซกเม้นต์มีจำนวนมากขึ้นจำนวนบิทต่อจุดภาพก็จะเพิ่มมากขึ้น นั้นก็หมายความว่าความคมชัดของภาพก็เพิ่มมากขึ้นด้วย เมื่อเปรียบเทียบกับวิธีการลดข้อมูลของ Discrete Cosine Transform (DCT) ซึ่งเป็นวิธีที่ใช้กันอย่างแพร่หลาย จะเห็นได้ว่า ที่อัตราบิทเรทสูงๆ คือประมาณ 0.5 บิทต่อจุดภาพขึ้นไป ผลที่ได้จากวิธีที่เสนอนี้มีความคมชัดมากกว่าวิธีของ DCT โดยเฉพาะส่วนเป็นขอบของภาพจะไม่เบลอ และเมื่อคำนวณค่าที่ผิดไปจากภาพเดิมเปรียบเทียบกัน โดยใช้ค่า Root Mean Square Error ตามสมการที่ (2.3) ผลที่ได้จากการลดข้อมูลโดยการเข้ารหัสภาพเซกเม้นต์จะมี Error น้อยกว่าวิธีของ DCT ดังตัวอย่างในรูปที่ 6.2 เป็นภาพที่อัตราบิทเรท 0.2 บิทต่อจุดภาพที่ได้จากผลของ DCT และ ผลจากการเข้ารหัสภาพเซกเม้นต์เปรียบเทียบกัน ส่วนรูปที่ 6.1 เป็นกราฟเปรียบเทียบ error ของ DCT และ error ของการเข้ารหัสภาพเซกเม้นต์



รูปที่ 6.1 เป็นกราฟแสดง error ของ DCT และ error ของการเข้ารหัสภาพเซกเม้นต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 6.2 แสดงผลของ DCT และ การเข้ารหัสภาพเซกเม้นต์ ที่อัตราบิทเรท 0.2 บิทต่อจุดภาพ

6.2 ปัญหาที่เกิดขึ้นและข้อเสนอแนะ

การลดข้อมูลภาพโดยการเข้ารหัสพื้นที่ของภาพเซกเม้นเตชั่นนั้น ขั้นตอนแรกต้องทำอิมเมจเซกเม้นเตชั่นก่อน ดังนั้นวิธีที่นำมาใช้จึงเป็นส่วนสำคัญที่บ่งบอกถึงประสิทธิภาพของระบบ อย่างเช่นวิธี RSST ที่นำมาใช้เป็นวิธีที่ต้องใช้เวลาในการคำนวณมาก เพราะมีการบวกซ้ำๆ กันของจุดยอดในกราฟเพื่อที่จะหาตัวเชื่อมตัวใหม่ที่มีค่าต่ำสุดที่สัมพันธ์กับส่วนเดิม และการทำงานบนเครื่องไมโครคอมพิวเตอร์ที่มีหน่วยความจำหลักเพียง 640 Kbytes จะมีข้อจำกัดของขนาดภาพที่นำมาทำเซกเม้นเตชั่น เนื่องจากจะต้องจองตัวแปรสำหรับค่าของจุดบนกราฟ ค่าของตัวเชื่อมระหว่างจุดยอดพร้อมทั้งลำดับตำแหน่งต่างๆ ของกราฟ เพื่อเพิ่มความเร็วในการคำนวณหาสแพนนิ่งทรีของกราฟ และในการคำนวณยังต้องใช้ข้อมูลแบบ floating point ด้วย แต่ถ้าหากสามารถที่จะเขียนโปรแกรมให้สามารถทำงานบนโปรเทคโหมดของ CPU 80286/80386 ได้ก็จะสามารถเพิ่มหน่วยความจำเข้าไปได้อีก ทำให้ข้อจำกัดในเรื่องขนาดของภาพที่นำมาใช้ก็จะหมดไป

ในส่วนของการที่จะพัฒนาต่อๆ ไป นั้นสามารถทำได้คือ หาวิธีการในการเพิ่มความเร็วของทำอิมเมจเซกเม้นเตชั่น อย่างเช่นแบ่งภาพออกเป็นส่วนๆ แล้วจึงนำไปทำเซกเม้นต์พร้อม ๆ กัน ที่เรียกว่าการประมวลผลแบบขนาน ในส่วนของการเข้ารหัสขอบสามารถที่จะปรับปรุงวิธีการที่นำมาใช้ให้สามารถลดขนาดของข้อมูลลงไปได้อีกอย่างเช่น ใช้วิธีของ Digital Line Segment Coding [12] Predictive Run Coding[13]. เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

[1] R. J. CLARKE, Transform Coding of Image, ACADEMIC PRESS Ins. London ,1975.

[2] N.S.Jayant and Perter Noll, "Digital coding of waveforms", Prentice-Hall, New Jersey, 1984.

[3] Ho. MingLin and Allan N. Willson, " Median Filters with Adaptive Length. ", IEEE Trans. on Circuit and System Vol. CAS-35, No. 6, pp. 675-690, June 1988.

[4] B.I.Justusson, " Median Filtering : Statistical property", in Two-Dimensional Digital Signal Processing Vol.II, T.S. Huang, Ed. New York: Spring Veriag, pp. 161-196, 1981.

[5] S.G.Tyan, " Median Filtering : Deterministic property ", in Two-Dimension Digital signal Processing Vol.II, T.S. Huang, Ed. New York : Spring-Veriag, pp. 197-217, 1981.

[6] N.C.Gallagher, Jr. and G.L. Wise, "A theoritical analysis of the property of median filters ", IEEE Trans. on Acoust., Speech, Signal Processing, Vol. ASSP-29, pp. 1136-1141, December 1981.

[7] S.L. Horowitz and Pavlidis, "Picture segmentation by a directed split-and-merge procedure", Proc. 2nd International Joint Conference on Pattern Recognition, Copenhagen, Aug. 13-15, pp. 424-433, 1987.

[8] Y. Pramotepipop and F. Cheevasuvit," Modification of split-and-merge algorithm for image segmentation", The 9th Asian Conference on Remote Sensing, Bangkok, Thailand, pp. Q-26-1 - Q-26-6, Nov. 23-29, 1988.

[9] O.J.Morris and M.de J. Lee, "Graph theory for image analysis : an approach based on the shortest spanning tree", IEE Proceedings, Vol. 133 , Pt.F, No.2, pp. 146-152, April 1986.

[10] Robert Sedgewick, " Algorithms:Graph Algorithms", ADDISON-WESLEY, New York, 1988.

[11] Robert Cunningham, " Segmenting Binary Image ", in Robotics Age, Hayden Book Company, New Jersey, 1983.

[12] B.B. Chaudhuri, M.K. Kundu, " Digital line segment coding: A New efficient contour coding scheme", IEE Proceedings, Vol.131, Pt.E, No.4, pp. 143-147, July 1984

[13] M.J. Biggar, " Thin Line Codeing Techniques ", International Conference on Digital Signal Processing Florence, September 7-10, pp. 101-107, 1987.

[14] TORU KANEKO and MASASHI OKUDAIRA, " Encoding of Arbitrary Curves Based on the Chain Code Representation ", IEEE Trans. on Communications, VOL. COM-33, No. 7, pp. 697-707, JULY 1985.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก. ผลงานวิจัยที่ได้รับการตีพิมพ์

[1] ศักดิ์ เสกชุนทด, สุชาติ ฉิมประดิษฐ์, รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์, "การวิเคราะห์ข้อมูลภาพถ่ายดาวเทียมด้วยเทคนิคการวิเคราะห์หลายตัวแปร", การประชุมทางวิชาการ สถิติประยุกต์ ครั้งที่ 8, สถาบันบัณฑิตพัฒนบริหารศาสตร์, หน้า 460-474, 24-25 พฤษภาคม 2533.

[2] ศักดิ์ เสกชุนทด, สุชาติ ฉิมประดิษฐ์, รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์, "การกำจัดสัญญาณรบกวนด้วยวงจรกรองมัธยฐานที่ปรับขนาดได้อย่างอัตโนมัติ", การประชุมทางวิชาการ สถิติประยุกต์ ครั้งที่ 8, สถาบันบัณฑิตพัฒนบริหารศาสตร์, หน้า 365-374, 24-25 พฤษภาคม 2533.

[3] สุชาติ ฉิมประดิษฐ์, ยอดเยี่ยม ปราโมทย์ภิภพ, รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์," การแบ่งส่วนภาพโดยอาศัยพื้นฐานทฤษฎีกราฟ", การประชุมทางวิชาการ วิศวกรรมไฟฟ้า ครั้งที่ 13, ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่, หน้า 690-699 8-9 พฤศจิกายน 2533.

[4] สุชาติ ฉิมประดิษฐ์, วิทยา ทิพย์สุวรรณพร, รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์, "การลดข้อมูลภาพด้วยวิธีการเข้ารหัสภาพเซกเม้นเตชั่น", การประชุมทางวิชาการวิศวกรรมไฟฟ้า ประจำปี 2534, วิศวกรรมสถานแห่งประเทศไทย ในพระบรมราชูปถัมภ์ และ สมาคมสถาบันวิศวกรไฟฟ้าและอิเล็กทรอนิกส์แห่งประเทศไทย, หน้า 43-53, 23-26 พฤษภาคม 2534.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*-------------------------------------------------------------------*/
/*                  Body of rsst program.                            */
/*                                                                   */
/* This is common to all size and cost function, which may be        */
/* altered by changing the last two  "include" files.                */
/* rsstsmall.h contains ndefinitions necessary for small images      */
/* rsstmed.h  contains ndefinitions necessary for medium images      */
/* rsstbig.h  contains ndefinitions necessary for large  images      */
/* rsstcost.ad absolute difference, mean intensitys.                 */
/*-------------------------------------------------------------------*/

    #include <stdio.h>
    #include <math.h>
    #include <alloc.h>
    #include <time.h>
    #include "rsstsmal.h"
    #include "rsstcost.ad"

    #define  sqr(A) ((A)*(A))

/*---------------------------------------*/
/*          Grobal Variable              */
/*---------------------------------------*/

    struct   LinkElement{
    TYPEREGIONACCESS   r1,r2;
    float              Weight;
    }   huge *LinkArray;
    struct   RegionRecord{
    float              mean;
    TYPEREGIONACCESS   NumPixels;
    TYPELINKACCESS     FirstLinkIndex;
    }    huge *RegionArray;
    struct    TreeEntry{
    TYPEREGIONACCESS   Vertex1;
    char               LinkDir;
    }      TreeRecord;
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
struct    ListEntry{
TYPELINKACCESS      LinkArrayPtr, NextListEntry;
}  huge *LinkList;
TYPELINKACCESS    *Heap,huge *Inv;
unsigned          NumOnHeap, PixelsInImage, RawLinkCount;
unsigned short    Xsize, Ysize;
TYPEREGIONACCESS  *AlreadyLinked, LinkedValue = 0;
char              progname[STRINGSIZE];
TYPEREGIONACCESS  LinksOutput;
FILE              *Treefile;

/*-------------------------------------*/
/*            AllocateFailure          */
/*-------------------------------------*/

void AllocateFailure()
     {
     fprintf(stderr,"%s: failure in memory allocation",progname);
     exit(1);
}

/*-------------------------------------*/
/*            Allocate Memory          */
/*-------------------------------------*/

void AllocateMemory()
{

     long lk,la,ra,in,he,al;

      lk = (4L*PixelsInImage - 2L*Xsize - 2L*Ysize + 1L) *
           sizeof(struct ListEntry);
      la = RawLinkCount  * sizeof(struct LinkElement);
      ra = PixelsInImage * sizeof(struct RegionRecord);
      in = RawLinkCount  * sizeof(TYPELINKACCESS);
      he = (2L * PixelsInImage - Xsize - Ysize + 1L)   *
           sizeof(TYPELINKACCESS);
      al = (long)PixelsInImage * (long)sizeof(TYPEREGIONACCESS);
     if((LinkList = (struct ListEntry*)farmalloc(lk)) == NULL)
     AllocateFailure();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    if((LinkArray = (struct LinkElement*)farmalloc(la)) == NULL)
    AllocateFailure();
    if((RegionArray = (struct RegionRecord*)farmalloc(ra)) == NULL)
    AllocateFailure();
    if((Inv = (TYPELINKACCESS*)farmalloc(in)) == NULL)
    AllocateFailure();
    if((Heap = (TYPELINKACCESS*)farmalloc(he)) == NULL)
    AllocateFailure();
    if((AlreadyLinked = (TYPEREGIONACCESS*)farmalloc(al)) == NULL)
    AllocateFailure();
}

/*---------------------------------------*/
/*    Initial Variable and Constant      */
/*---------------------------------------*/

void Initialise(argc,argv)
    int     argc;
    char    *argv[];
    {
    char      ImageName[STRINGSIZE], TreeName[STRINGSIZE];
    FILE      *Infile;
    struct    RegionRecord    huge *p;
    int       i, BytesPerPixel,j;
    short     shortbuffer;

    strcpy(progname,argv[0]);
    switch(argc)
    {
    case  1: printf("Input image filename: ");
    scanf("%s", ImageName);
    case  2: printf("Output Tree filename: ");
    scanf("%s", TreeName);
    case  3: printf("Image x size : ");
    scanf("%d",&Xsize);
    case  4: printf("Image y size : ");
    scanf("%d",&Ysize);
    case  5: break;
    default:fprintf(stderr,"%s: wrong number of arguments\n",progname);
    }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    switch(argc)
    {
      case 5: sscanf(argv[4], "%d",&Ysize);
      case 4: sscanf(argv[3], "%d",&Xsize);
      case 3: strcpy(TreeName,argv[2]);
      case 2: strcpy(ImageName,argv[1]);
      default: break;
    }
    if((Infile = fopen(ImageName,"rb")) == NULL)
    {
    fprintf(stderr,"%s: unable to open file %s\n",
    progname, ImageName);
    exit(1);
    }
    PixelsInImage = Xsize * Ysize;
    RawLinkCount =  2*PixelsInImage - 1;
    AllocateMemory();
    i = 0;
    for(p = RegionArray; p<RegionArray+PixelsInImage; p++)
    {
    fread((char*)&shortbuffer,1,1,Infile);
    p->mean = (float)shortbuffer;
    p->NumPixels = 1;
    }
    fclose(Infile);
    if((Treefile = fopen(TreeName,"wb")) == NULL)
    {
    fprintf(stderr,"%s: unable to open file %s\n",
    progname, ImageName);
    exit(1);
  }
    for(i=0; i<PixelsInImage; i++) AlreadyLinked[i] = 0;
    }

/*---------------------------------------*/
/*    Map Image array to Graph array     */
/*---------------------------------------*/

void SetInitialLinks()
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    register TYPELINKACCESSLinkIndex,LastLinkIndex;
    register structLinkElement *LinkElementPtr;
    register TYPELINKACCESSLinkPtr;
    register unsigned intTwoI;
    TYPELINKACCESSi,CountLinks,LinkEntry;

    LinkEntry = 0;
    for(i=0; i<PixelsInImage; i++)
    {
    TwoI = i + i;
    CountLinks = 0;
    if(i % Xsize)
    {
    LinkIndex = ++LinkEntry;
    RegionArray[i].FirstLinkIndex = LinkIndex;
    LinkList[LinkIndex].LinkArrayPtr = TwoI - 1;
    LastLinkIndex = LinkIndex;
    CountLinks++;
   }
  if(i >= Xsize)
   {
    LinkIndex = ++LinkEntry;
    if(CountLinks) LinkList[LastLinkIndex].NextListEntry = LinkIndex;
    else RegionArray[i].FirstLinkIndex = LinkIndex;
    LinkList[LinkIndex].LinkArrayPtr = TwoI - 2*Xsize + 2;
    LastLinkIndex = LinkIndex;
    CountLinks++;
   }
  if((i+1) % Xsize)
   {
    LinkIndex = ++LinkEntry;
    if(CountLinks) LinkList[LastLinkIndex].NextListEntry = LinkIndex;
    else RegionArray[i].FirstLinkIndex = LinkIndex;
    LinkPtr = LinkList[LinkIndex].LinkArrayPtr = TwoI + 1;
    LinkElementPtr = &LinkArray[LinkPtr];
    LinkElementPtr->r1 = i;
    LinkElementPtr->r2 = i+1;
    LinkElementPtr->Weight = FindInitialWeight;
    LastLinkIndex = LinkIndex;
    CountLinks++;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }
   else if(i != PixelsInImage-1) LinkArray[TwoI+1].Weight = -1.0;
   if(i < PixelsInImage - Xsize)
    {
     LinkIndex = ++LinkEntry;
     LinkList[LastLinkIndex].NextListEntry = LinkIndex;
     LinkPtr = LinkList[LinkIndex].LinkArrayPtr = TwoI + 2;
     LinkElementPtr = &LinkArray[LinkPtr];
     LinkElementPtr->r1 = i;
     LinkElementPtr->r2 = i + Xsize;
     LinkElementPtr->Weight = FindInitialWeight;
     LastLinkIndex = LinkIndex;
     CountLinks++;
    }
   else if(i != PixelsInImage-1) LinkArray[TwoI+2].Weight = -1.0;
   LinkList[LastLinkIndex].NextListEntry = 0;
 }
}

/*-----------------------------------------*/
/*-----------------------------------------*/

void PQDownHeap(k)
     register TYPELINKACCESSk;
     {
     register unsigned intj,HalfNumOnHeap;
     register unsigned intHeapOfj,HeapOfjPlus1;
     unsigned intv;
     register floatvvalue;

     v = Heap[k];
     vvalue = LinkArray[v].Weight;
     HalfNumOnHeap = NumOnHeap/2;
     while (k <= HalfNumOnHeap)
     {
     j = k + k;
     HeapOfj = Heap[j];
     HeapOfjPlus1 = Heap[j+1];
     if(j<NumOnHeap)
      if(LinkArray[HeapOfj].Weight > LinkArray[HeapOfjPlus1].Weight)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        {
        j++;
        HeapOfj = HeapOfjPlus1;
        }
             if(vvalue <= LinkArray[HeapOfj].Weight ) break;
             Heap[k] = HeapOfj;
             Inv[HeapOfj] = k;
             k = j;
        }
        Heap[k] = v;
        Inv[v]  = k;
        }

/*-----------------------------------------*/
/*      Construct and sort the heap        */
/*-----------------------------------------*/

void PQConstruct()
     {
     register unsigned intk;

     NumOnHeap = 0;
     Heap[0] = 0;
     LinkArray[0].Weight = -1.0;
     for(k=1; k<RawLinkCount; k++)
     if(LinkArray[k].Weight >= 0.0)
     {
     Heap[++NumOnHeap] = k;
     Inv[k] = NumOnHeap;
    }
     for(k = NumOnHeap/2; k>=1;k--)
       PQDownHeap(k);
     }

/*-----------------------------------------*/
/*-----------------------------------------*/

void PQUpHeap(k)
     register unsigned int k;
     {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการ
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    register unsigned intv,Halfk,HalfHeapOfk;
    register   floatLinkvWeight;

    v = Heap[k];
    Halfk = k/2;
    HalfHeapOfk = Heap[Halfk];
    LinkvWeight = LinkArray[v].Weight;
    while(LinkArray[HalfHeapOfk].Weight > LinkvWeight)
    {
     Heap[k] = HalfHeapOfk;
     Inv[HalfHeapOfk] = k;
     k = Halfk;
     Halfk = k/2;
     HalfHeapOfk = Heap[Halfk];
    }
    Heap[k] = v;
    Inv[v]  = k;
}

/*--------------------------------------*/
/*        Remove duplicate link         */
/*--------------------------------------*/

unsigned int PQRemove()
    {
    register intLinkNumber;

    LinkNumber = Heap[1];
    Heap[1] = Heap[NumOnHeap];
    if(--NumOnHeap) PQDownHeap(1);
    return(LinkNumber);
    }

/*--------------------------------------*/
/*--------------------------------------*/

void PQChange(LinkID)
    register unsigned int LinkID;
    {
    register unsigned int k;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการ
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใ

```
        k = 0;
        if(LinkArray[Heap[k/2]].Weight > LinkArray[Heap[k]].Weight)
        PQUpHeap(k);
        else PQDownHeap(k);
        }

/*---------------------------------------*/
/*---------------------------------------*/

void PQRid(LinkID)
        register unsigned int LinkID;
        {
        register unsigned int k;

        k = Inv[LinkID];
        Inv[Heap[k]] = k;
        if(k == NumOnHeap) NumOnHeap--;
        else
    {
        Heap[k] = Heap[NumOnHeap--];
        PQDownHeap(k);
    }
}

/*---------------------------------------*/
/*          Save out put link            */
/*---------------------------------------*/

void OutputLink(LinkNumber)
        register unsigned int LinkNumber;
        {
        TreeRecord.Vertex1 = (LinkNumber-1)/2;
        if(LinkNumber % 2) TreeRecord.LinkDir = 0;
         else TreeRecord.LinkDir = 1;
        fwrite(&TreeRecord,sizeof(struct TreeEntry),1,Treefile);
        }

/*---------------------------------------*/
/*---------------------------------------*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void ReviseOtherRegionList(RegionNumber,IndexToRemove)
     register unsigned int  RegionNumber,IndexToRemove;
     {
     register unsigned int  StepIndex,LastStepIndex;
     register struct ListEntry       huge *LinkListPtr;
     char FirstLink;
     FirstLink = 1;
     StepIndex = RegionArray[RegionNumber].FirstLinkIndex;
     LinkListPtr = &LinkList[StepIndex];
     while(LinkListPtr->LinkArrayPtr != IndexToRemove)
    {
      FirstLink = 0;
      LastStepIndex = StepIndex;
      StepIndex = LinkListPtr->NextListEntry;
      LinkListPtr = &LinkList[StepIndex];
    }
  if(FirstLink)
  RegionArray[RegionNumber].FirstLinkIndex = LinkListPtr->NextListEntry;
  else
  LinkList[LastStepIndex].NextListEntry = LinkListPtr->NextListEntry;
}
/*-----------------------------------------*
 *-----------------------------------------*/

void MergeRegions(LinkNumber)
     unsigned int LinkNumber;
     {
     #define mean1       RegionArray[LinkArray[LinkNumber].r1].mean
  #define mean2       RegionArray[LinkArray[LinkNumber].r2].mean
  #define Num1        RegionArray[LinkArray[LinkNumber].r1].NumPixels
  #define Num2        RegionArray[LinkArray[LinkNumber].r2].NumPixels
  #define LinkIndex1  RegionArray[LinkArray[LinkNumber].r1].FirstLinkIndex
  #define LinkIndex2  RegionArray[LinkArray[LinkNumber].r2].FirstLinkIndex

     register unsigned int LLINextListEntry,LLILinkArrayPtr,LinkIndex,
                      LastLinkIndex;
     unsigned int          TotalPixels,CountLinks,OtherRegion;
     char FinishedList,RegionBIsFirst;
     struct ListEntry  *LinkListIndex;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  TotalPixels = Num1 + Num2;
  mean1 = (mean1*Num1 + mean2*Num2)/TotalPixels;
  Num1 = TotalPixels;
  LinkedValue++;
  CountLinks = 0;
  FinishedList = 0;
  LinkIndex = LinkIndex1;
  LinkListIndex = &LinkList[LinkIndex];
  LLINextListEntry = LinkListIndex->NextListEntry;
  LLILinkArrayPtr = LinkListIndex->LinkArrayPtr;
  while(!FinishedList)
  {
 if((LLINextListEntry == 0) && (LLILinkArrayPtr == LinkNumber))
   {
       if(CountLinks) LinkList[LastLinkIndex].NextListEntry =
       LinkIndex2;
       else  LinkIndex1 = LinkIndex2;
       LinkIndex = LinkIndex2;
       LinkListIndex = &LinkList[LinkIndex];
       LLINextListEntry = LinkListIndex->NextListEntry;
       LLILinkArrayPtr = LinkListIndex->LinkArrayPtr;
       FinishedList = 1;
   }
else if(LLILinkArrayPtr == LinkNumber)
  {
    LinkIndex = LLINextListEntry;
    LinkListIndex = &LinkList[LinkIndex];
    LLINextListEntry = LinkListIndex->NextListEntry;
    LLILinkArrayPtr = LinkListIndex->LinkArrayPtr;
    if(CountLinks)LinkList[LastLinkIndex].NextListEntry = LinkIndex;
    else LinkIndex1 = LinkIndex;
  }
else
  {
    LinkArray[LLILinkArrayPtr].Weight = FindWeight(LLILinkArrayPtr);
    PQChange(LLILinkArrayPtr);
    if(LinkArray[LinkNumber].r1 == LinkArray[LLILinkArrayPtr].r1)
       OtherRegion = LinkArray[LLILinkArrayPtr].r2;
    else
       OtherRegion = LinkArray[LLILinkArrayPtr].r1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    AlreadyLinked[OtherRegion] = LinkedValue;
    CountLinks++;
    if(LLINextListEntry == 0)
      {
         LastLinkIndex = LinkIndex;
         LinkListIndex->NextListEntry = LinkIndex2;
         LinkIndex = LinkIndex2;
         LinkListIndex = &LinkList[LinkIndex];
         LLINextListEntry = LinkListIndex->NextListEntry;
         LLILinkArrayPtr = LinkListIndex->LinkArrayPtr;
         FinishedList = 1;
      }
    else
      {
         LastLinkIndex = LinkIndex;
         LinkIndex = LLINextListEntry;
         LinkListIndex = &LinkList[LinkIndex];
         LLINextListEntry = LinkListIndex->NextListEntry;
         LLILinkArrayPtr = LinkListIndex->LinkArrayPtr;
      }
    }
}

  FinishedList = 0;
  while(!FinishedList)
  {
 if((LLINextListEntry == 0)&&(LLILinkArrayPtr == LinkNumber))
  {
    LinkList[LastLinkIndex].NextListEntry = 0;
    FinishedList = 1;
  }
   else
    if(LLILinkArrayPtr == LinkNumber)
       {
       LinkIndex = LLINextListEntry;
       LinkListIndex = &LinkList[LinkIndex];
       LLINextListEntry = LinkListIndex->NextListEntry;
       LLILinkArrayPtr = LinkListIndex->LinkArrayPtr;
       if(CountLinks) LinkList[LastLinkIndex].NextListEntry =
            LinkIndex;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        else LinkIndex1 = LinkIndex;
        }
     else
        {
        if(LinkArray[LinkNumber].r2==LinkArray[LLILinkArrayPtr].r1)
        {
        OtherRegion = LinkArray[LLILinkArrayPtr].r2;
        RegionBIsFirst = 1;
        }
        else
        {
        OtherRegion = LinkArray[LLILinkArrayPtr].r1;
        RegionBIsFirst = 0;
        }
   if(AlreadyLinked[OtherRegion] == LinkedValue)
        {
        PQRid(LLILinkArrayPtr);
        ReviseOtherRegionList(OtherRegion,LLILinkArrayPtr);
        if(LLINextListEntry == 0)
          {
            LinkList[LastLinkIndex].NextListEntry = 0;
            FinishedList = 1;
          }
        else
        {
        LinkIndex = LLINextListEntry;
        LinkListIndex = &LinkList[LinkIndex];
        LLINextListEntry = LinkListIndex->NextListEntry;
        LLILinkArrayPtr = LinkListIndex->LinkArrayPtr;
        LinkList[LastLinkIndex].NextListEntry = LinkIndex;
        }
   }
else
{
if(RegionBIsFirst)
LinkArray[LLILinkArrayPtr].r1  = LinkArray[LinkNumber].r1;
else
 LinkArray[LLILinkArrayPtr].r2 = LinkArray[LinkNumber].r1;
 LinkArray[LLILinkArrayPtr].Weight = FindWeight(LLILinkArrayPtr);
 PQChange(LLILinkArrayPtr);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        CountLinks++;
        if(LLINextListEntry == 0) FinishedList = 1;
        else
        {
         LastLinkIndex = LinkIndex;
         LinkIndex = LLINextListEntry;
         LinkListIndex = &LinkList[LinkIndex];
         LLINextListEntry = LinkListIndex->NextListEntry;
         LLILinkArrayPtr = LinkListIndex->LinkArrayPtr;
         }
        }
     }
   }
}

/*-----------------------------------------*/
/*          Do Kruskal Algorithm           */
/*-----------------------------------------*/

void DoKruskalOnLowestWeights()
     {
     unsigned intSmallestLink;

     LinksOutput = 0;
     while(LinksOutput < PixelsInImage - 1)
     {

          SmallestLink = PQRemove();
          LinksOutput++;
          OutputLink(SmallestLink);
          MergeRegions(SmallestLink);
     }
          fclose(Treefile);
     }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*-----------------------------------------*/
/*                  Main                   */
/*-----------------------------------------*/

main(argc,argv)
     int  argc;
     char *argv[];
     {
     int i;

          Initialise(argc,argv);
          SetInitialLinks();
          PQConstruct();
          printf("<< Kruskal computing >> ");
          DoKruskalOnLowestWeights();

     }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*--------------------------------------*/
/*  Programe to generate segmented image */
/*  frome the original image and a      */
/*  spanning tree                       */
/*--------------------------------------*/

     #include <stdio.h>
     #include <alloc.h>

     #define STRSIZE  50

     char   progname[STRSIZE];
     char   ImageFile[STRSIZE],TreeFile[STRSIZE] ,SegFile[STRSIZE];
     unsigned char  *ImageArray;
     unsigned char  *LinkArray,*MarkerArray;
     long int       *PixelStack;
     long int        IncImageIndex[9];
     long int        RequiredRegions;
     int        XSize,YSize;
     int        MaxInt = -32000, MinInt = 32000;
     long int        TotalPixels;
     FILE            *InFile;

/*--------------------------------------*/
/*--------------------------------------*/

void AllocFailure()
     {
     fprintf(stderr,"%s: Failure to allocate memory\n",progname);
     exit(1);
     }

/*--------------------------------------*/
/*--------------------------------------*/

void GetSpace()
     {
     int   i;

     if((ImageArray=(short*)malloc(TotalPixels*sizeof(short))) == NULL)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        AllocFailure();
    if((LinkArray = malloc(TotalPixels*sizeof(char))) == NULL)
        AllocFailure();
    if((MarkerArray = malloc(TotalPixels*sizeof(char))) == NULL)
        AllocFailure();
    if((PixelStack = (int*)malloc(TotalPixels*sizeof(int))) == NULL)
        AllocFailure();
    for(i=0; i<TotalPixels; i++)
    {
        LinkArray[i] = 0;
        MarkerArray[i] = 0;
    }
}

/*-------------------------------------*/
/*-------------------------------------*/

void GetInputs(argc,argv)
    int argc;
    char *argv[];
    {
    long i,j;
    char buff;

    strcpy(progname,argv[0]);


    switch(argc)
        {
            case 1: printf("Original image filename: ");
                    scanf("%s",ImageFile);
            case 2: printf("X size: ");
                    scanf("%d",&XSize);
            case 3: printf("Y size: ");
                    scanf("%d",&YSize);
            case 4: printf("Tree filename: ");
                    scanf("%s",TreeFile);
            case 5:  printf("Output filename: ");
                    scanf("%s",SegFile);
            case 6: printf("Required no. of regions: ");
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    scanf("%d",&RequiredRegions);
                case 7: break;
                default: fprintf(stderr,"%s: wrong number of arquments\n",
            progname);
             }

        switch(argc)
             {
             case 7:  sscanf(argv[6], "%d",&RequiredRegions);
             case 6:  strcpy(SegFile,argv[5]);
             case 5:  strcpy(TreeFile,argv[4]);
             case 4:  sscanf(argv[3],"%d",&YSize);
             case 3:  sscanf(argv[2],"%d",&XSize);
             case 2:  strcpy(ImageFile, argv[1]);
             default: break;
             }


        if((InFile = fopen(ImageFile,"rb")) == NULL)
        {
        fprintf(stderr,"%s: cannot open file %s\n",progname,ImageFile);
        exit(1);
        }

        TotalPixels = XSize * YSize;
        GetSpace();


    /**  Read Image File **/

           for(i=0; i<TotalPixels; i++)
            {
             fread((char*)&buff,1,1,InFile);
             ImageArray[i] = buff;
            }
        IncImageIndex[1] = -XSize;
        IncImageIndex[2] = 1L;
        IncImageIndex[4] = XSize;
        IncImageIndex[8] = -1L;
        }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*-------------------------------------*/
/*-------------------------------------*/

void ReadLinks(TreeFileName,Regions)
     char *TreeFileName;
     long  Regions;
     {
     register int i,NumLinks;
     FILE            *InTreeFile;
     structSmallTreeEntry
          {
          unsigned int Vertext;
          char     LinkDir;
          } SmallBuffer;

     if((InTreeFile = fopen(TreeFileName,"rb")) == NULL)
     {
     fprintf(stderr,"%s: cannot open file %s\n",progname,TreeFileName);
     exit(1);
     }
     NumLinks = TotalPixels - Regions;
     for(i=0; i<NumLinks; i++)
    {
     if(fread((char*)&SmallBuffer,sizeof(SmallTreeEntry),1,InTreeFile) !=
      {
       fprintf(stderr,"%s: error in reading file %s\n",
              progname,TreeFileName);
       exit(1);
      }
    if(SmallBuffer.LinkDir)
      {
       LinkArray[SmallBuffer.Vertext] |= 4;
       LinkArray[SmallBuffer.Vertext + XSize] |= 1;
      }
    else
      {
       LinkArray[SmallBuffer.Vertext] |= 2;
       LinkArray[SmallBuffer.Vertext + 1] |= 8;
      }
 }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    fclose(TreeFileName);
    }


/*-------------------------------------*/
/*-------------------------------------*/

int FindMean(Index)

    register int intIndex;
    {
    register int CheckPixel,StackPointer = 0,TotalIntensity,NumInRegion;
    register char LookAround;

    MarkerArray[Index] = 1;
    TotalIntensity = (int)ImageArray[Index];
    NumInRegion = 1;

    do
    {
    for(LookAround=1; LookAround<16; LookAround<<=1)
    {
     if(LinkArray[Index] & LookAround)
         {
         CheckPixel = Index + IncImageIndex[LookAround];
         if(!MarkerArray[CheckPixel])
         {
              MarkerArray[CheckPixel] = 1;
              TotalIntensity += (int)ImageArray[CheckPixel];
               NumInRegion++;
              PixelStack[++StackPointer] = CheckPixel;
              }
         }
    }
         Index = PixelStack[StackPointer--];
    } while (StackPointer >= 0);
    return((short)((TotalIntensity+NumInRegion/2)/NumInRegion));
    }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*-------------------------------------*/
/*-------------------------------------*/

void SetMean(Index,MeanValue)
     register int int Index;
     register int MeanValue;
     {
     register  int CheckPixel,StackPointer = 0;
     register char LookAround;

     MarkerArray[Index] = 2;
     ImageArray[Index] = MeanValue;
     do
     {
     for(LookAround=1; LookAround<16; LookAround<<=1)
     if(LinkArray[Index] & LookAround)
     {
      CheckPixel = Index + IncImageIndex[LookAround];
      if(MarkerArray[CheckPixel]==1)
          {
          MarkerArray[CheckPixel] = 2;
          ImageArray[CheckPixel] = MeanValue;
          PixelStack[++StackPointer] = CheckPixel;
          }
     }
     Index = PixelStack[StackPointer--];
     } while (StackPointer >= 0);
     }

/*-------------------------------------*/
/*-------------------------------------*/

void  GetSegmentation()
     {
     register  int Pixel;
     int       MeanIntensity;

     for(Pixel=0; Pixel<TotalPixels; Pixel++)
     if(!MarkerArray[Pixel])
     {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      MeanIntensity = FindMean(Pixel);
      SetMean(Pixel,MeanIntensity);
      if(MaxInt<MeanIntensity) MaxInt = MeanIntensity;
      if(MinInt>MeanIntensity) MinInt = MeanIntensity;
     }
}

/*-------------------------------------*/
/*-------------------------------------*/

void  StoreImage(FileName,Image)
      char  *FileName;
      int    Image[];
      {
      FILE   *OutFile;
      int  ErrNo,x,y,i,j;

      if((OutFile = fopen(FileName,"wb")) == NULL)
      {
      fprintf(stderr,"%s: cannot open file %s\n",progname,FileName);
      exit(1);
      }
      if((x = fwrite((char*)Image,1,XSize*YSize,OutFile))==NULL)
      printf(" Error \n");

/*-------------------------------------*/
/*           Segmented Main            */
/*-------------------------------------*/

main(argc,argv)
      intargc;
      charargv[];
      {
      clrscr();
      GetInputs(argc,argv);
      ReadLinks(TreeFile,RequiredRegions);
      GetSegmentation();
      StoreImage(SegFile,ImageArray);
      return(0);
      }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*--------------------------------------*/
/* Program Segmented Image Contour Line  */
/*              Coding                   */
/*--------------------------------------*/

     #include <stdio.h>
     #include <alloc.h>
     #include <stdlib.h>
     #include <dir.h>
     #include <io.h>
     #include <math.h>

     struct edge-code     *make-edge();
     struct region-code   *mark-region();

     struct   region-code {
          char      ave;
          unsigned  pix-start;
          struct region-code *next;
          };
     struct   edge-code {
          char edging;
          struct edge-code  *next;
          };
     unsigned   *index,maxindex=0;
     struct     region-code  *region,  *last-region;
     struct     edge-code    *edge,    *last-edge;
     unsigned   int xsi,ysi,count;
     unsigned   int w,linkcount;
     unsigned   char code;
     FILE       *out;
     char       out-file[30],in-file[30];


void initial(argc,argv)
     int   argc;
     char  *argv[];
     {
     int i;
        clrscr();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
     region =  last-region = NULL;
     edge   =  last-edge   = NULL;
     switch(argc) {
      case 1: printf(" Input Image Segment File : ");
      scanf("%s",in-file);
      case 2: printf(" Image Size X = ");
      scanf("%d",&xsi);
      case 3: printf(" Image Size Y = ");
      scanf("%d",&ysi);
      case 4: break;
     }
     switch(argc) {
           case 4: sscanf(argv[3],"%d",&ysi);
          case 3: sscanf(argv[2],"%d",&xsi);
          case 2: strcpy(in-file,argv[1]);
          default: break;
     }
     if((index = (int*)farmalloc(6000L * (long)sizeof(int)))== NULL)
     allo-err(1);
     count =  w = code = 0;
     }
/*-------------------------------------*/
/*      allocate memory                */
/*-------------------------------------*/

void allo-err(d)
     int d;
     {
     printf("memory allocate failure..(%d).. \n",d);
     exit(1);
     }

/*-------allocate memory for  input image-------*/
char *space(image)
     unsigned char *image;
     {
     if((image = (char*)malloc(xsi*ysi))==NULL)
          allo-err(2);
          return(image);
     }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*------------------- free memory-----------------*/

void freem()
     {
     free(image);
     free(mark);
     farfree(index);
     }
/*--------------- read-image ------------*/
void read-image(image)
     unsigned char  *image;
     {
     FILE *infile;
     long f-size,temp,temp1,k;
     int xi,yi;

      f-size = (long)xsi*(long)ysi;
     if ((infile = fopen(in-file,"rb")) == NULL) {
          printf(" Open file error..\n");
          exit(1);
     }
     k = 0;
     temp = 32768;
     if(f-size > 32768)
     do {
       fread(image+k,1,temp,infile);
       temp1 = f-size-temp;
       f-size = temp1;
       k += temp;
       if(temp1<temp) temp = temp1;
    } while (!(temp1 <= 0));
    else
         fread(image+k,1,f-size,infile);
      fclose(infile);
      }

 /*------save output code------------*/
 void save-data()
      {
      FILE *outf;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
char name[35];
int i,j;
struct   region-code *p;
struct   edge-code   *q;

clrscr();
printf(" Enter output file name.. ");
scanf("%s",name);
if((outf = fopen(name,"wb"))==NULL){
   printf("Cann't open file %s \n",name);
   exit(1);
}
fwrite(&count,2,1,outf);
p = region;
do {
    fwrite(p,3,1,outf);
    p = p->next;
}while(p);
q = edge;
do {
    fwrite(q,1,1,outf);
    q = q->next;
}while(q);
fclose(outf);
}
/*-------------------------------------*/
/*         display output image        */
/*-------------------------------------*/

/*-----black and white monitor display-----*/
void bw-dis(buff)
unsigned char huge *buff;
{
 long i,j;
   outportb(0x300,0x61);
for(i=0;i<ysi;i++)
 for(j=0;j<xsi;j++)
  pokeb(0xd000,(i*256)+j,buff[(long)xsi*i + j]);
   outportb(0x300,0x81);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*-----VGA moniter display-----*/
display (x,y,xposi,yposi,image)
int y,x,xposi,yposi;
unsigned char   *image;
{
 long i,j,f;

 for (i=0;i<y;i++)
   for (j=0;j<x;j++){
f = i*(long)x+j;
put (image[f]/4+64,0,(int)j+xposi,(int)i+yposi);
   }
}

/*-------------------------------------*/
void blank-display (x,y,xpo,ypo,val)
int y,x,xpo,ypo,val;
{
 long i,j,f;

 for (i=0;i<y;i++)
   for (j=0;j<x;j++){
f = i*(long)x+j;
put (val,0,(int)j+xpo,(int)i+ypo);
   }
}

/*-----set gray register-----*/
void set-gray-REG ()
{
int i,j=0;
for (i=64;i<128;i++)
 {
  wrcolor (i,j,j,j);
  j++;
 }
}
/*-------------------------------------*/
void wrcolor (int regnum,int red,int green,int blue)
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 -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);
}

/*-------------------------------------*/
/*        Read input image  code       */
/*-------------------------------------*/

void input-segment()
{
   FILE *infile;
   struct ffblk file;
   int i,j,byte,true;
   char fi-name[50],c;

   do {
true = 0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
printf(" Enter input code file name ..");
scanf("%s",fi-name);
infile = fopen(fi-name,"rb");
if(infile==NULL) {
   printf(" File %s not found !\n",fi-name);
   printf(" Again press any key or Exit press ESC \n");
   c = getch();
   if(c == 0x1b)
      exit(1);
   else
      true = 0;
}
else  true = 1;
   } while (!true);
   if(findfirst(fi-name,&file,0xff)==0)
   byte = (int)file.ff-fsize;
   printf(" %d byte \n",byte);
   fread(&count,2,1,infile);
   mean-code  = (struct region-code*)malloc
              (count*sizeof(struct region-code));
   edge-code  = (char*)malloc(byte-(count*3)-2);
   fread(mean-code,sizeof(struct region-code),count,infile);
   fread(edge-code,1,(byte-(count*3)-2),infile);
   fclose(infile);
}

/*-------------------------------------*/
/*        Contour chain coding         */
/*-------------------------------------*/

/*-----scan image array------*/
void chain-code()
{
 unsigned char  *image,  *mark;
 unsigned long i,j;

 image = space(image);
 mark  = space(mark);
 for(i=0; i<xsi*ysi; i++){ mark[i] = 120; image[i]=0;}
 setVGA (0x13);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
set-gray-REG ();
read-image(image);
display (xsi,ysi,60,0,image);
setVGA (0x3);
i = 0;
trace(i,image,mark);
for(i=0; i<ysi; i++)
 for(j=0; j<xsi; j++)
     if(mark[(long)xsi*i + j] != 100)
        trace((long)xsi*i + j,image,mark);
save-data();
free(image);
free(mark);
}

/*----trace border-----*/
void trace(xy,image,mark)
unsigned long xy;     /* start coordinate */
unsigned char  *image,  *mark;
{
  long xy0,maxpix,stackpointer=0,i,j;
  char mean,m = 0;

  maxpix = xsize*ysize;
  count++;
  xy0  = xy;
  mean = image[xy];

  last-region  = mark-region(xy,mean);
 do {
   cou++;
   switch(m) {
    case 0: if(!((xy+1)%xsi)) {     /* right edge */
             m=3;
             str-link(3);
            }
            else
            if( xy<xsi && image[xy+1] == mean) { /* top edge and next */
              m=0;                               /* pixel is same region*/
              str-link(0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            xy = xy+1;
          }
         else
          if(image[xy+1] != mean) { /* not top edge and next */
             m=3;                   /* pixel isn't the same region */
             str-link(3);
          }
          else
           if(image[xy-xsi+1] == mean) {
              m=1;
              str-link(1);
              xy = xy-xsi+1;
           }
           else {
              m=0;
              str-link(0);
              xy = xy+1;
           }
        break;
case 1: if(xy<xsi) {     /* top edge */
        m=0;
        str-link(0);
        }
        else
         if(!(xy%xsi) && image[xy-xsi] == mean) {
            m=1;
            str-link(1);
            xy = xy-xsi;
           }
        else
        if(image[xy-xsi] != mean) {
         m=0;
         str-link(0);
         }
         else
        if(image[xy-xsi-1] == mean) {
           m=2;
           str-link(2);
           xy = xy-xsi-1;
        }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        else {
          m=1;
          str-link(1);
          xy = xy - xsi;
        }
        break;
case 2: if(!(xy%xsi)) {        /* left edge */
         m=1;
         str-link(1);
         }
       else
       if(xy>=maxpix-xsi && image[xy-1] == mean) { /* bottom edge */
         m=2;
         str-link(2);
         xy = xy-1;
       }
       else
       if(image[xy-1] != mean) {
         m=1;
         str-link(1);
       }
       else
       if(image[xy+xsi-1] == mean) {
           m=3;
           str-link(3);
           xy = xy+xsi-1;
       }
       else {
           m=2;
           str-link(2);
           xy = xy-1;
       }
        break;
case 3: if(xy>maxpix-xsi) {     /* bottom edge */
           m=2;
           str-link(2);
       }
       else
       if(!((xy+1)%xsi) && image[xy+xsi] == mean) {
           m=3;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                str-link(3);
                xy = xy+xsi;
            }
            else
            if(image[xy+xsi]!=mean) {
                m=2;
                str-link(2);
            }
            else
            if(image[xy+xsi+1]==mean) {
                m=0;
                str-link(0);
                xy = xy + xsi +1;
            }
            else {
                m = 3;
                str-link(3);
                xy = xy + xsi;
            }
            break;
            }
            cou++;
    } while (!(m==0 && xy==xy0));

cou = 0;
mark[xy]  = 100;
do {
   if(xy>=xsi && image[xy-xsi]==mean)  /* not top edge */
      if(mark[xy-xsi]!=100)
         index[++stackpointer] = xy-xsi;
   if((xy+1)%xsi && image[xy+1]==mean) /* not right edge */
      if(mark[xy+1]!=100)
         index[++stackpointer] = xy + 1;
   if(xy%xsi && (image[xy-1]==mean))   /* not left edge */
      if(mark[xy-1]!=100)
         index[++stackpointer] = xy - 1;

   if(xy<maxpix-xsi && image[xy+xsi]==mean) /* not bottom edge */
      if(mark[xy+xsi]!=100)
        index[++stackpointer] = xy + xsi;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    xy = index[stackpointer--];
    mark[xy]   = 100;
  } while(stackpointer >= 0);
}

/*-------------------------------------*/
/*         create link-list            */
/*-------------------------------------*/
void  str-link(m)
 char m;
 {

  switch(m){
   case 0: code = code |  0x00;
   if(linkcount<3)   code = code << 2;
   break;
   case 1: code = code |  0x01;
   if(linkcount<3)   code = code << 2;
   break;
   case 2: code = code |  0x02;
   if(linkcount<3)   code = code << 2;
   break;
   case 3: code = code |  0x03;
   if(linkcount<3)   code = code << 2;
   break;
   };
  linkcount += 1;
  if(linkcount==4)
    {
      w += 1;
      linkcount = 0;
      last-edge = make-edge(code);
      code = 0;
    }
 }


/*-------------------------------------*/
struct region-code  far *mark-region(xyo,m)
unsigned long xyo;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
unsigned char m;            /* make list for gray and start of a region */
{
 struct region-code *newptr;

if((newptr = (struct region-code*)malloc(sizeof(struct region-code)))==NULL)
    {
       printf("internal memory for coordinate not enough..\n");
       exit(1);
    }
 if(region==NULL)  region = newptr;
 if(last-region!=NULL) last-region->next = newptr;
   newptr->ave         = (char)m;
   newptr->pix-start   = (unsigned)xyo;
   newptr->next        = NULL;
   return(newptr);
}

/*-------------------------------------*/
struct edge-code far *make-edge(e)     /*  edge code storring */
char e;
{
   struct  edge-code far *newptr;

   if((newptr = (struct edge-code*)malloc(sizeof(struct edge-code)))==NULL)
      {
       printf("internal memory for edge not enough..\n");
       exit(1);
      }
   if(edge==NULL) edge = newptr;
   if(last-edge != NULL) last-edge->next = newptr;
   newptr->edging = e;
   newptr->next   = NULL;
   return(newptr);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*-------------------------------------*/
/*              image decode           */
/*-------------------------------------*/
void decode()
{
unsigned char *image, *mark,mean;
int           data[4],bit,d,l,m,m1,k;
long numpix,i,xy,xy0,stackpointer=0;
struct edge-code *restor();

 image = space(image);
 mark  = space(mark);
 for(i=0; i<xsi*ysi; i++){ mark[i] = 0; image[i]=0;}
 numpix = (long)xsi*(long)ysi;
 d   = 0;
 bit = 0;
 edge =  restor(edge,data);
 for(l=0; l<count; l++)
 {
   xy = xy0 =  region->pix-start;
        mean =  region->ave;
     region =  region->next;
     m1   = 0;

do {
      m = data[bit];
      switch(m){
       case 0x00: if(m1==0x00) {
     xy = xy+1;
     if(xy>=xsi)
     image[xy-xsi]=1;
   }
  else
    if(m1==0x40) {
      if(xy>=xsi)
       image[xy-xsi]=1;
     }
    else {
      xy = xy + 1;
      image[xy+1]=1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        xy = xy + xsi;
       }
   break;
        case 0x40: if(m1==0x00) {
       xy = xy-xsi;
       image[xy]=1;
       xy = xy+1;
      }
   else
     if(m1==0x40) {
        xy = xy-xsi;
        if(xy%xsi)
image[xy-1]=1;
       }
     else {
        if(xy%xsi)
 image[xy-1]=1;
       }
   break;
        case 0x80: if(m1==0x40)  {
      xy = xy-1;
      image[xy]=1;
      xy = xy-xsi;
     }
   else
     if(m1==0x80)  {
        xy = xy-1;
        if(xy<numpix-xsi)
 image[xy+xsi]=1;
       }
      else  {
if(xy<numpix-xsi)
 image[xy+xsi]=1;
       }
   break;
        case 0xc0: if(m1==0x00) {
       if((xy+1)%xsi)
 image[xy+1]=1;
      }
   else
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    if(m1==0x80) {
       xy = xy+xsi;
       image[xy]=1;
       xy = xy-1;
      }
    else  {
       xy = xy+xsi;
       if((xy+1)%xsi)
image[xy+1]=1;
      }
  break;
}
    m1 = m;
    bit += 1;
    if(bit == 4){
     bit = 0;
     d  += 1;
     edge = restor(edge,data);
     }
  } while(!(xy==xy0 && m==0));

  mark[xy] = mean;
  do{
   if(xy>=xsi && image[xy-xsi] != 1) {                    /* not top */
      index[++stackpointer] = xy - xsi;
   }
   if((xy+1)%xsi && image[xy+1] != 1) {                   /* not right */
      index[++stackpointer] = xy + 1;
   }
   if(xy<numpix-xsi && image[xy+xsi] != 1) {              /* not bottom */
      index[++stackpointer] = xy + xsi;
      }
      if(xy%xsi && image[xy-1] != 1) {                    /* not left */
         index[++stackpointer] = xy-1;
         }
         xy = index[stackpointer--];
         mark[xy] = mean;
         image[xy] = 1;
    }while(stackpointer >= 0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    xy = xy0;
    stackpointer = 0;
    image[xy] = 0;
    do{
    if(xy>=xsi && image[xy-xsi] != 0) {                    /* not top */
      index[++stackpointer] = xy - xsi;
   }
   if((xy+1)%xsi && image[xy+1] != 0) {                    /* not right */
      index[++stackpointer] = xy + 1;
   }
   if(xy<numpix-xsi && image[xy+xsi] != 0) {               /* not bottom */
      index[++stackpointer] = xy + xsi;
    }
     if(xy%xsi && image[xy-1] != 0) {                      /* not left */
           index[++stackpointer] = xy-1;
    }
    xy = index[stackpointer--];
    image[xy] = 0;
    }while(stackpointer >= 0);
}
    setVGA (0x13);
    set-gray-REG ();
    display (xsi,ysi,60,0,mark);
    getch();
}
/*-------------------------------------*/
    struct edge-code *restor(b,data)
    struct edge-code *b;
    int data[];
    {
    int k,d;

    d = b->edging;
    for(k=0;k<4;k++)
    {
         if(k==0)
         data[k] = d & 0xc0;
       else
         data[k] = (d<<=2) & 0xc0;
    }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
     edge = b->next;
      return(edge);
}

/*-------------------------------------*/
/*           Main Program              */
/*-------------------------------------*/

void  main(argc,argv)
     int   argc;
     char  *argv[];
     {
      int i,j,k;

     initial(argc,argv);
     chain-code();
     decode();
     }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/************************************/
/*              PROGRAM             */
/*    NON-BACK TRACE LINE CODING    */
/************************************/

#include <stdio.h>
#include <alloc.h>
#include <dos.h>

struct  st {
unsigned char x;
unsigned char y;
unsigned char m;
} *stack;

unsigned char **mark,**image1;
unsigned char *code;
int  xsize,ysize,count=0,code-num,segment;
unsigned char *image;

void AlloFai()
{
     printf(" failure to allocate memory \n");
     exit(1);
}

unsigned char *mark1D()
     {
     unsigned char *a;
     if((a=(char*)calloc(xsize+1,1))==NULL) AlloFai();
      return(a);
}

unsigned char *index1D()
     {
     struct st *a;
     if((a=(struct st*)malloc(1000*sizeof(struct st)))==NULL) AlloFai();
      return(a);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
unsigned char *image1D()
     {
     unsigned char *a;
     if((a=(char*)malloc(xsize*ysize))==NULL) AlloFai();
      return(a);
}

unsigned char **mark2D()
     {
     unsigned char **b;
     unsigned i;
     if((b=(char**)malloc((ysize+1)*sizeof(char*)))==NULL) AlloFai();
     for(i=0; i<ysize+1; i++) b[i] = mark1D();
      return(b);
}

getspace()
     {
     mark  = mark2D();
     stack = index1D();
      image1= mark2D();
}

/************************************/
/************************************/
readJimage()
     {
     FILE *inf;
     char name[50];
     long  i,j;
     int  x,y;
     printf("Enter input image file  <..");
     scanf("%s",name);
     if((inf = fopen(name,"rb"))==NULL){
     printf("Open file fail !\n");
     exit(1);
     }
     for(i=0;i<ysize;i++)
          fread(image1[i],1,xsize,inf);
      fclose(inf);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*********************************/
/*********************************/
writeJimage()
     {
     FILE *outf;
     char name[50];
     long i,j;

     printf("Enter output file name  <..");
     scanf("%s",name);
     if((outf = fopen(name,"wb"))==NULL){
     printf("Open file fail !\n");
     exit(1);
     }
     j = (long)xsize*(long)ysize;
     if(j>32768){
     i = 32768;
     fwrite(image,1,i,outf);
     }
     else{
     fwrite(image,1,j,outf);
     i = 0;
     }
     if(i!=0){
     j = j -i;
     fwrite(image+32768,1,j,outf);
     }
     fclose(outf);
 }

/***********************************/
/*       Find Line Diagram         */
/***********************************/

scanJimage()
     {
     int x,y;

     for(y=0; y<ysize; y++){
      for(x=0; x<xsize; x++) {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
   if((image1[y][x]-image1[y][x+1])!=0)
      mark[y][x+1] = 1;
   if((image1[y][x]-image1[y+1][x])!=0)
      mark[y+1][x] = 1;
  if((image1[y][x]-image1[y][x+1])!=0 && (image1[y][x]-image1[y+1][x])!=0)
      mark[y+1][x+1] = 1;
     }
   }
}

/**********************************/
/*     Specifi Edge Direction     */
/**********************************/

unsigned direction(unsigned y,unsigned x,unsigned m)
     {
      unsigned i,j;

     switch(m) {
     case 0:if(mark[y-1][x]==1 && mark[y][x+1]==1 && mark[y+1][x]==1)
                                                {m = 4;mark[y][x]=0;}
        else if(mark[y-1][x]==1 && mark[y][x+1]==1)
                                                {m = 5;mark[y][x]=0;}
           else if(mark[y][x+1]==1 && mark[y+1][x]==1)
                                                {m = 6;mark[y][x]=0;}
              else if(mark[y-1][x]==1 && mark[y+1][x]==1)
                                                {m = 7;mark[y][x]=0;}
                 else if(mark[y-1][x]==1)
                                                {m = 1;mark[y][x]=0;}
                    else if(mark[y+1][x]==1)
                                                {m = 3;mark[y][x]=0;}
                       else                      m = 8;
       break;
       case 1:if(mark[y][x-1]==1 && mark[y-1][x]==1 && mark[y][x+1]==1)
                                                {m = 4;mark[y][x]=0;}
          else if(mark[y][x-1]==1 && mark[y-1][x]==1)
                                                {m = 5;mark[y][x]=0;}
             else if(mark[y-1][x]==1 && mark[y][x+1]==1)
                                                {m = 6;mark[y][x]=0;}
                else if(mark[y][x-1]==1 && mark[y][x+1]==1)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                                                {m = 7;mark[y][x]=0;}
                        else if(mark[y][x-1]==1)
                                                {m = 2;mark[y][x]=0;}
                            else if(mark[y][x+1]==1)
                                                {m = 0;mark[y][x]=0;}
                                else            m = 8;
    break;
    case 2:if(mark[y+1][x]==1 && mark[y][x-1]==1 && mark[y-1][x]==1)
                                                {m = 4;mark[y][x]=0;}
       else if(mark[y+1][x]==1 && mark[y][x-1]==1)
                                                {m = 5;mark[y][x]=0;}
            else if(mark[y][x-1]==1 && mark[y-1][x]==1)
                                                {m = 6;mark[y][x]=0;}
                else if(mark[y+1][x]==1 && mark[y-1][x]==1)
                                                {m = 7;mark[y][x]=0;}
                    else if(mark[y+1][x]==1)
                                                {m = 3;mark[y][x]=0;}
                        else if(mark[y-1][x]==1)
                                                {m = 1;mark[y][x]=0;}
                            else                m = 8;
    break;
    case 3:if(mark[y][x+1]==1 && mark[y+1][x]==1 && mark[y][x-1]==1)
                                                {m = 4;mark[y][x]=0;}
         else if(mark[y][x+1]==1 && mark[y+1][x]==1)
                                                {m = 5;mark[y][x]=0;}
              else if(mark[y+1][x]==1 && mark[y][x-1]==1)
                                                {m = 6;mark[y][x]=0;}
                  else if(mark[y][x+1]==1 && mark[y][x-1]==1)
                                                {m = 7;mark[y][x]=0;}
                      else if(mark[y][x+1]==1)
                                                {m = 0;mark[y][x]=0;}
                          else if(mark[y][x-1]==1)
                                                {m = 2;mark[y][x]=0;}
                              else              m = 8;
    break;
    }
    return(m);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/************************************/
/*       trace line diagram         */
/************************************/

trace(unsigned char y,unsigned char x)
     {
     int  start,stackpointer = 0;
     unsigned char m1,m;
     unsigned dir,length=0;
     unsigned char byte;

     if(count==0){
     code[0] = x;
     code[1] = y;
     count = 2;
     }
     else {
     code[count++]= 0;
     code[count++]= 0;
     code[count++]= x;
     code[count++]= y;
     }

     if(mark[y][x+1]==1 && mark[y+1][x]==1) {
     stackpointer += 1;
     stack[stackpointer].x = x+1;
     stack[stackpointer].y = y;
     stack[stackpointer].m = 0;
     m    = 3;
     dir = 1;
     }
     else if(mark[y][x+1]==1){ m    = 0; dir = 0; }
          else { m    = 3; dir = 1; }


     do {
     switch(m)  {
     case 0: while(mark[y-1][x]!=1 && mark[y][x]==1 && mark[y+1][x]!=1)
     {
          length += 1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    mark[y][x]=0;
    put(0,0,x+50,y);delay(10);
    if(x<xsize-1)
     x += 1;
  }
 break;
 case 1: while(mark[y][x-1]!=1 && mark[y][x]==1 && mark[y][x+1]!=1)
  {
    length += 1;
    mark[y][x]=0;
    put(0,0,x+50,y);delay(10);
    if(y>0)
     y -= 1;
  }
 break;
 case 2: while(mark[y+1][x]!=1 && mark[y][x]==1 && mark[y-1][x]!=1)
  {
    length += 1;
    mark[y][x]=0;
    put(0,0,x+50,y);delay(1);
    if(x>0)
      x -= 1;
  }
 break;
 case 3: while(mark[y][x+1]!=1 && mark[y][x]==1 && mark[y][x-1]!=1)
  {
    length += 1;
    mark[y][x]=0;
    put(0,0,x+50,y);delay(1);
    if(y<ysize-1)
      y += 1;
  }
 break;
 }

      length += 1;
    m1      = direction(y,x,m);     /* check direction and junction */
    if(m1<4) {                      /* turn left and right case     */
    if(length>126) byte = 127;
    else byte = (char)length;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  if(dir) byte = byte | 0x80;
  else byte = byte & 0x7f;
  code[count++] = byte;
          length = 0;
        m      = m1;
  switch(m1) {
  case 0: if(m==1) dir = 1;
  else dir = 0;
  if(x<xsize-1) x++;
  break;
  case 1: if(m==0) dir = 0;
  else dir = 1;
  if(y>0) y--;
  break;
  case 2: if(m==1) dir = 0;
  else dir = 1;
  if(x>0) x--;
  break;
  case 3: if(m==0) dir = 1;
  else dir = 0;
  if(y<ysize-1) y++;
  break;
 }
}
else if(m1<8) {
     switch(m1) {
     case 4: if(m==0) {
             if(length>126) byte = 127;
             else byte = (char)length;
             if(dir) byte = byte | 0x80;
             else byte = byte & 0x7f;
             code[count++] = byte;
             length = 0;
             code[count++] = 0x80;
             stackpointer += 1;
             stack[stackpointer].x = x;
             stack[stackpointer].y = y+1;
             stack[stackpointer].m = 3;
             stackpointer += 1;
             stack[stackpointer].x = x+1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            stack[stackpointer].y = y;
            stack[stackpointer].m = 0;
            dir =  1;
            m   =    1;
            if(y>0) y--;
    }
    else if(m==1) {
          if(length>126) byte = 127;
          else byte = (char)length;
          if(dir) byte = byte | 0x80;
          else byte = byte & 0x7f;
          code[count++] = byte;
          length = 0;
          code[count++]  = 0x80;
          stackpointer += 1;
          stack[stackpointer].x = x+1;
          stack[stackpointer].y = y;
          stack[stackpointer].m = 0;
          stackpointer += 1;
          stack[stackpointer].x = x;
          stack[stackpointer].y = y-1;
          stack[stackpointer].m = 1;
          dir =  1;
          m   =    2;
          if(x>0) x--;
}
else if(m==2) {
          if(length>126) byte = 127;
          else byte = (char)length;
          if(dir) byte = byte | 0x80;
          else byte = byte & 0x7f;
          code[count++] = byte;
          length = 0;
          code[count++]          = 0x80;
          stackpointer += 1;
          stack[stackpointer].x = x;
          stack[stackpointer].y = y+1;
          stack[stackpointer].m = 1;
          stackpointer += 1;
          stack[stackpointer].x = x-1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        stack[stackpointer].y = y;
        stack[stackpointer].m = 2;
        dir =    1;
        m   =    3;
        if(y<ysize-1) y++;
    }
      else {
           if(length>126) byte = 127;
           else byte = (char)length;
           if(dir) byte = byte | 0x80;
           else byte = byte & 0x7f;
           code[count++] = byte;
           length = 0;
           code[count++] = 0x80;
           stackpointer += 1;
           stack[stackpointer].x = x-1;
           stack[stackpointer].y = y;
           stack[stackpointer].m = 2;
           stackpointer += 1;
           stack[stackpointer].x = x;
           stack[stackpointer].y = y+1;
           stack[stackpointer].m = 3;
           dir =  1;
           m   =  0;
           if(x<xsize-1) x++;
      }
    break;
    case 5: if(m==0) {
           if(length>126) byte = 127;
           else byte = (char)length;
           if(dir) byte = byte | 0x80;
           else byte = byte & 0x7f;
           code[count++] = byte;
           length = 0;
           code[count++] = 0x00;
           stackpointer += 1;
           stack[stackpointer].x = x+1;
           stack[stackpointer].y = y;
           stack[stackpointer].m = 0;
           dir  =  0; m    =  1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        if(y>0) y--;
    }
        else if(m==1) {
        if(length>126) byte = 127;
        else byte = (char)length;
        if(dir) byte = byte | 0x80;
        else byte = byte & 0x7f;
        code[count++] = byte;
        length = 0;
        code[count++] = 0x00;
        stackpointer += 1;
        stack[stackpointer].x = x;
        stack[stackpointer].y = y-1;
        stack[stackpointer].m = 1;
        dir  =  0; m    =  2;
        if(x>0) x--;
}
else if(m==2) {
        if(length>126) byte = 127;
        else byte = (char)length;
        if(dir) byte = byte | 0x80;
        else byte = byte & 0x7f;
        code[count++] = byte;
        length = 0;
        code[count++] = 0x00;
        stackpointer += 1;
        stack[stackpointer].x = x-1;
        stack[stackpointer].y = y;
        stack[stackpointer].m = 2;
        dir  =  0; m    =  3;
        if(y<ysize-1) y++;
}
else {
        if(length>126) byte = 127;
        else byte = (char)length;
        if(dir) byte = byte | 0x80;
        else byte = byte & 0x7f;
        code[count++] = byte;
        length = 0;
        code[count++]  = 0x00;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

```
        stackpointer += 1;
        stack[stackpointer].x = x;
        stack[stackpointer].y = y+1;
        stack[stackpointer].m = 3;
        dir  =  0;  m    =  0;
        if(x<xsize-1) x++;
    }
    break;
    case 6: if(m==0) {
            if(length>126) byte = 127;
            else byte = (char)length;
            if(dir) byte = byte | 0x80;
            else byte = byte & 0x7f;
            code[count++] = byte;
            length = 0;
            code[count++]  = 0x00;
            stackpointer += 1;
            stack[stackpointer].x = x;
            stack[stackpointer].y = y+1;
            stack[stackpointer].m = 3;
            dir  =  1; m    =  0;
            if(x<xsize-1) x++;
        }
        else if(m==1) {
            if(length>126) byte = 127;
            else byte = (char)length;
            if(dir) byte = byte | 0x80;
            else byte = byte & 0x7f;
            code[count++] = byte;
            length = 0;
            code[count++] = 0x00;
            stackpointer += 1;
            stack[stackpointer].x = x+1;
            stack[stackpointer].y = y;
            stack[stackpointer].m = 0;
            dir  =  1; m    =  1;
            if(y>0) y--;
        }
            else if(m==2) {
                if(length>126) byte = 127;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                else byte = (char)length;
                if(dir) byte = byte | 0x80;
                else byte = byte & 0x7f;
                code[count++] = byte;
                length = 0;
                code[count++] = 0x00;
                stackpointer += 1;
                stack[stackpointer].x = x;
                stack[stackpointer].y = y-1;
                stack[stackpointer].m = 1;
                dir  =  1; m    =  2;
                if(x>0) x--;
        }
        else {
                if(length>126) byte = 127;
                else byte = (char)length;
                if(dir) byte = byte | 0x80;
                else byte = byte & 0x7f;
                code[count++] = byte;
                length = 0;
                code[count++]  = 0x00;
                stackpointer += 1;
                stack[stackpointer].x = x-1;
                stack[stackpointer].y = y;
                stack[stackpointer].m = 2;
                dir  =  1;  m    =  3;
                if(y<ysize-1) y++;
        }
break;
case 7: if(m==0) {
                if(length>126) byte = 127;
                else byte =  (char)length;
                if(dir) byte = byte | 0x80;
                else byte = byte & 0x7f;
                code[count++] = byte;
                length = 0;
                code[count++]  = 0x80;
                stackpointer += 1;
                stack[stackpointer].x = x;
                stack[stackpointer].y = y+1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        stack[stackpointer].m = 3;
        dir  =  0; m    =  1;
        if(y>0) y--;
    }
    .else if(m==1) {
        if(length>126) byte = 127;
        else byte = (char)length;
        if(dir) byte = byte | 0x80;
        else byte = byte & 0x7f;
        code[count++] = byte;
        length = 0;
        code[count++]  = 0x80;
        stackpointer += 1;
        stack[stackpointer].x = x+1;
        stack[stackpointer].y = y;
        stack[stackpointer].m = 0;
        dir  =  0; m    =  2;
        if(x>0) x--;
    }
        else if(m==2) {
        if(length>126) byte = 127;
        else byte = (char)length;
        if(dir) byte = byte | 0x80;
        else byte = byte & 0x7f;
        code[count++] = byte;
        length = 0;
        code[count++]  = 0x80;
        stackpointer += 1;
        stack[stackpointer].x = x;
        stack[stackpointer].y = y-1;
        stack[stackpointer].m = 1;
        dir  =  0; m    =  3;
        if(y<ysize-1) y++;
    }
        else {
            if(length>126) byte = 127;
            else byte = (char)length;
            if(dir) byte = byte | 0x80;
            else byte = byte & 0x7f;
            code[count++] = byte;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    length = 0;
                    code[count++]  = 0x80;
                    stackpointer += 1;
                    stack[stackpointer].x = x-1;
                    stack[stackpointer].y = y;
                    stack[stackpointer].m = 2;
                    dir  =  0;  m    =  0;
                    if(x<xsize-1) x++;
                }
      break;
      }
         }
         else {
         if(length>126) byte = 127;
         else byte = (char)length;
         if(dir) byte = byte | 0x80;
         else byte = byte & 0x7f;
         code[count++] = byte;
         length = 0;

         x = stack[stackpointer].x;
         y = stack[stackpointer].y;
         m = stack[stackpointer].m;
         stackpointer -= 1;
      }
      } while (stackpointer>= 0);
}

/*************************************/
/*          Edge Coding              */
/*************************************/
      edge]code()
      {
      unsigned x,y;

      for(y=0; y<ysize; y++)
        for(x=0; x<xsize; x++)
           if(mark[y][x] == 1)
               trace(y,x);
      }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

```
/*************************************/
/*              MAIN                 */
/*************************************/

main()
  {

     int i,j,r,p,t;


     clrscr();
     xsize = 200;
     ysize = 200;
     getspace();
     readJimage();
     scanJimage();
     edgeJcode();
  }
```

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้