การกำหนดดัชนีและการเรียกภาพนิ่งโดยใช้สีของภาพ

IMAGE INDEXING AND RETRIEVAL BASED ON COLOR

โดย

นางสาวพรรณทิพย์ รงค์เหลืองอร่าม

นางสาวพัชรี รอดพ้น

อาจารย์ที่ปรึกษา

ดร.ชุติเมษฎ์ ศรีนิลทา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

การกำหนดดัชนีและการเรียกภาพนิ่งโดยใช้สีของภาพ
IMAGE INDEXING AND RETRIEVAL BASED ON COLOR

โดย

นางสาวพรรณทิพย์ รงศ์เหลืองอร่าม

นางสาวพัชรี รอดพ้น

อาจารย์ที่ปรึกษา

ดร.ชุติเมษฏ์ ศรีนิลทา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

รายงานวิชา Project I ภาคเรียนที่ 1 ปีการศึกษา 2542

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การกำหนดดัชนีและการเรียกภาพนิ่งโดยใช้สีของภาพ

Image indexing and retrieval based on color

ผู้จัดทำ

1. นส.พรรณทิพย์ รงค์เหลืองอร่าม รหัสนักศึกษา 39014355
2. นส.พัชรี รอดพ้น รหัสนักศึกษา 39014362

ดร.ชุติเมษฏ์ ศรีนิลทา

(อาจารย์ที่ปรึกษา)

การกำหนดดัชนีและการเรียกภาพนิ่งโดยใช้สีของภาพ

นางสาวพรรณทิพย์ รงศ์เหลืองอร่าม 39014355
นางสาวพัชรี รอดพ้น 39014362
ดร. ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการนำเสนอวิธีการกำหนดดัชนีโดยใช้สีของภาพเพื่อนำไปใช้ประโยชน์ในการค้นหาภาพจากฐานข้อมูล ในโครงงานจะเป็นการทำงานกับไฟล์ของภาพประเภทบิทแมพ (bitmap file) ซึ่งเก็บพิกเซลต่างๆของภาพในรูปแบบของ RGB Color space เพื่อลดความซับซ้อนของมิติสีเราจึงทำการแปลงภาพให้อยู่ในรูปแบบของ HSV Color space และลดระดับสีให้เหลือ 166 ระดับสี จากนั้นจึงทำการคำนวณฮิสโตแกรมสี (Color Histogram) ในภาพนั้นและทำการเก็บลงแฟ้มข้อมูล เพื่อใช้เป็นดัชนีของภาพในการเปรียบเทียบสำหรับการค้นหาภาพนิ่งที่ต้องการต่อไป ในการเรียกภาพนิ่งจะใช้วิธีฮิสโตแกรมอินเตอร์เซกชัน (Histogram Intersection) หาค่าความเหมือนระหว่างคู่ของภาพเป้าหมายและภาพโมเดลแต่ละภาพ สุดท้ายจึงนำภาพโมเดลที่มีความเหมือนกับภาพเป้าหมายมากที่สุดมาแสดงผล

โครงงานนี้ได้พัฒนาเพื่อทดสอบประสิทธิภาพของวิธีฮิสโตแกรมอินเตอร์เซกชันและหาค่าความเหมือนที่เหมาะสม จากนั้นนำค่าความเหมือนที่ได้มาทดลองใช้กับแฟ้มข้อมูลขนาด 67 ภาพ พบว่าวิธีนี้ใช้ได้ผลดีกับภาพที่มีการเปลี่ยนมุมมองและภาพที่มีกลุ่มสีคล้ายๆกัน

Image indexing and retrieval based on color

Pantip Rongluengaram

Patcharee Rodpon

Dr. Chutimet Srinilta Advisor

ABSTRACT

This thesis experiments color-based image indexing and retrieval schemes. Indexing scheme works with bitmap images whose color points are in RGB color space. In order to reduce the complexity, color points are transformed to HSV and then quantized to 166-HSV color spaces. After that, the color histogram of the image is computed and stored in the directory as an index of that image. A technique called histogram intersection is used in the retrieval. Histogram of the target image is computed and then compared to determine the similarity of the color components. Images with high degree of similarity are then retrieved.

This project is developed to test the performance of color histogram intersection scheme and finding the appropriate similarity value. Besides that we tested this similarity value on 67 images in a library. The proof of our results reveals a good response with the images that have difference angle of views and the images that have a likely color.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และความร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษาของโครงงานที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

พี่นัส มนัส ที่คอยให้คำปรึกษาในการเขียนโปรแกรม VC++

พี่อุ้ม สรวิชญ์ ที่คอยกระตุ้นให้ขยันทำโครงงาน และพยายามให้คำแนะนำเกี่ยวกับ image processing

อาจารย์ทุกๆท่านที่ช่วยให้คำแนะนำต่างๆและให้ความสนใจกับโครงงานของเราเป็นอย่างดี

เพื่อนๆที่คอยให้กำลังใจในการทำโครงงานและให้คำแนะนำเกี่ยวกับ interface ของโปรแกรม รวมทั้งช่วยกันจำลองตัวเองเป็นผู้ใช้เพื่อทดสอบโปรแกรม

คู่โปรเจ็คที่แสนดี (ต้องขอบคุณในความร่วมมือเป็นอย่างดี ทำให้โครงงานผ่านลุล่วงไปได้)

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจเอาใจใส่เสมอมา ในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

พรรณทิพย์ รงศ์เหลืองอร่าม

พัชรี รอดพ้น

สารบัญ

สารบัญภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันนี้มีหลายวิธีที่ใช้ในการค้นหาข้อมูลในฐานข้อมูล การให้ดัชนี (index) กับข้อมูลมีผลเป็นอย่างมากต่อประสิทธิภาพของการค้นหา เนื่องจากภาพมีความหมายในตัวและเป็นการยากที่จะให้คำจำกัดความเป็นคำพูด ดังนั้น วิธีการค้นหาโดยใช้คำเป็นดัชนีเหมือนกับที่ใช้ในฐานข้อมูลที่เก็บข้อมูลเป็นตัวอักษรจึงไม่เหมาะสมกับฐานข้อมูลที่เก็บภาพ เพื่อให้การกำหนดดัชนีของภาพเป็นไปอย่างมีประสิทธิภาพ จึงได้มีการนำเอาลักษณะเฉพาะ (feature) ของภาพเป็นตัวกำหนดดัชนี เช่น สี, โครงร่าง (texture), รูปร่าง (shape), การเคลื่อนไหว (motion) และข้อมูลของภาพ

การเติบโตของภาพดิจิตอล ทั้งในรูปแบบของ ภาพทางโทรทัศน์ ภาพถ่ายดิจิตอล ภาพกราฟฟิก ภาพอนิเมชัน และวีดีโอ ทำให้มีการเก็บรวบรวมภาพและข้อมูลของภาพลงในฐานข้อมูลเพื่อที่จะสามารถค้นหาภาพและข้อมูลของภาพได้อย่างสะดวกและรวดเร็ว เช่น การค้นหาภาพและข้อมูลภาพจากอินเตอร์เน็ต, video-on-demand, ภาพในคลังข้อมูล (stock photography distribution) และอื่นๆ ซึ่งข้อมูลเหล่านี้อาจไม่เหมาะสมกับวิธีการกำหนดดัชนีโดยใช้ตัวอักษรเนื่องจากไม่สามารถระบุความหมายของภาพได้อย่างครบถ้วน

จะเห็นได้ว่า การกำหนดดัชนีให้กับภาพโดยใช้ลักษณะเฉพาะของภาพจึงเป็นวิธีที่เหมาะสมกว่า ในโครงงานนี้ได้นำเอาสีของภาพซึ่งเป็นลักษณะเฉพาะของภาพอย่างหนึ่งเป็นดัชนี เนื่องจากเป็นลักษณะหลักที่เห็นได้ชัดเจน ง่ายต่อการเข้าใจ เทคนิคของการใช้สีเป็นตัวดัชนีนี้แบ่งออกเป็น 2 วิธีคือ การใช้ global color เป็นดัชนี และการใช้ local หรือ region color เป็นดัชนี ความแตกต่างของวิธีทั้งสองนี้คือ ในการใช้ดัชนีแบบ global นั้น จะใช้สีทั้งหมดที่ปรากฏในภาพในการเปรียบเทียบ ซึ่งจะมีประโยชน์สำหรับการค้นหาภาพที่ไม่เจาะจงตำแหน่งใดๆในภาพ เช่น ภาพป่าไม้ หรือ สนามฟุตบอล ในขณะที่การใช้ดัชนีแบบ local จะเป็นการเปรียบเทียบเฉพาะขอบเขตที่สนใจในภาพนั้นและใช้ประโยชน์สำหรับค้นหาส่วนของภาพ (ตำแหน่ง) ที่เราสนใจ เช่น ภาพดวงอาทิตย์ขึ้นจากขอบฟ้า ซึ่งดวงอาทิตย์อยู่ทางด้านซ้ายของรูป แต่วิธีการใช้ดัชนีแบบ local color นั้นมีความซับซ้อนกว่า แบบ global color

โครงงานที่ได้จัดทำนี้เลือกใช้วิธี global color โดยการทำฮิสโตแกรมของสี (color histogram) ทั่วทั้งภาพ เนื่องจากว่าเป็นวิธีที่เข้าใจง่ายและสามารถนำไปพัฒนาร่วมกับเทคนิคอื่นๆเพื่อปรับปรุงประสิทธิภาพให้ดีขึ้นได้ ฮิสโตแกรมที่ได้จากการทำฮิสโตแกรมของสีจะมีความใกล้เคียงกันแม้ว่าวัตถุในภาพจะมีการเปลี่ยนมุมมอง (angle of view), การหมุน, การเปลี่ยนแปลงสเกลของภาพ หรือภาพที่มีบางส่วนขาดหายไป ซึ่งภาพเหล่านี้พบเห็นโดยทั่วไปในชีวิตประจำวัน ภาพที่ใช้ในโครงงานนี้จะเป็นภาพบิทแมพไฟล์ เนื่องจากว่าเป็นภาพที่มีรูปแบบพื้นฐานและสะดวกต่อการนำมาใช้และคำนวณ

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 ศึกษาความเป็นไปได้ของโครงงานในการนำเอาสีของภาพเป็นดัชนี ศึกษาถึงความสามารถและประสิทธิภาพทางด้านการค้นหาของวิธีการใช้สีเป็นดัชนีให้กับภาพ

1.2.2 รวบรวมแนวคิดและพัฒนาวิธีการค้นหาภาพโดยใช้สีเป็นดัชนี ศึกษาข้อจำกัดของแนวคิดและการแก้ไขเพื่อให้การกำหนดดัชนีโดยใช้สีของภาพมีประสิทธิภาพมากที่สุด

1.2.3 ศึกษาถึงแนวทางในการพัฒนาโดยใช้เทคนิคอื่นๆร่วมกับการใช้สีเป็นดัชนี เพื่อลดข้อเสียเปรียบของวิธีการใช้สีเป็นดัชนี

1.3 งานวิจัยอื่นที่เกี่ยวข้อง

จากการศึกษางานวิจัยเกี่ยวกับการค้นหาภาพด้วยเทคนิคต่างๆ ได้สรุปส่วนที่น่าสนใจไว้ดังนี้

1 Capacity of Color Histogram Indexing ให้ทุกภาพมีจำนวนพิกเซลรวมเท่ากัน สร้างฮิสโตแกรมสีของภาพโดยเก็บอยู่ในรูปของเวกเตอร์ หาค่าความเหมือนของภาพโดยดูจากระยะห่าง (distance) ระหว่างฮิสโตแกรมของภาพ

ผลที่ได้ 1. ถ้าภาพมีสีพื้นหลังที่เหมือนกันถึงแม้จะเป็นภาพที่ต่างกัน ก็จะทำให้ค่าระยะห่างที่ได้ต่ำมาก

2. การกำหนดดัชนีโดยใช้ฮิสโตแกรมของสีจะได้ผลดีก็ต่อเมื่อมีการกระจาย (sparse) ของฮิสโตแกรม
3. ค่าความจุ (capacity) จะบอกได้เพียงขนาดของฮิสโตแกรมของสีที่จะเก็บได้เท่านั้น แต่ไม่ได้เป็นตัววัดถึงความสำเร็จของวิธีการในการกำหนดดัชนี
4. สามารถใช้กับภาพที่เปลี่ยนมุมมองหรือความละเอียดของภาพ (Resolution) ได้
5. จะเกิดความผิดพลาดมากเมื่อภาพโดนรบกวนเนื่องมาจากเซนเซอร์(ทำให้พิกเซลสีเปลี่ยนสี)หรือมีความสว่างเปลี่ยนไป
6. ไม่มีผลกระทบกับภาพที่ไม่ชัดที่ถ่ายได้จากกล้องดิจิตอล
7. วัตถุที่สนใจในภาพสามารถเปลี่ยนตำแหน่งได้ เนื่องจากฮิสโตแกรมไม่ขึ้นกับตำแหน่งของภาพ

2 Color Clustering Techniques for Color-Content-Based Image Retrieval from Image Database ลดความซับซ้อนของฮิสโตแกรมของภาพ โดยใช้เทคนิคของการทำ color clustering แบบต่างๆ

ผลที่ได้ 1. วิธี equalize quantization เป็นวิธีที่ง่ายแต่มี color loss สูง

2. วิธี hierachical clustering เป็นวิธีที่มี color loss ต่ำที่สุดแต่ใช้เวลาในการทำงานนานมาก
3.. วิธี color naming system (CNS) เป็นวิธีที่ยาก แต่ใช้เวลาในการทำงานน้อยที่สุด

คำนวณดัชนีและเปรียบเทียบดัชนีนั้นกับดัชนีที่อยู่ในแฟ้มข้อมูล ภาพที่มีดัชนีเหมือนกันหรือใกล้เคียงกันมากที่สุดจะถูกดึงออกมาพร้อมกับข้อมูลของภาพจากแฟ้มข้อมูลเพื่อแสดงผล

งานในโครงงานนี้แบ่งออกเป็น 2 ส่วนคือ

1.4.1 การเก็บรูปภาพและดัชนีของภาพลงแฟ้มข้อมูล โดยรับภาพที่เป็น RGB color space ซึ่งต่อไปจะเรียกว่า ภาพโมเดล มาทำการกำหนดขอบเขตของภาพที่ต้องการเก็บ ส่วนของภาพที่ถูกกำหนดของเขตนี้จะถูกทำการแปลงสีและลดระดับสี เพื่อให้อยู่ในรูปของ 166 สี HSV color space จากนั้นทำการคำนวณหาฮิสโตแกรมของภาพ โดยแยกทำเป็น 4 ฮิสโตแกรมคือ ฮิสโตแกรมของระดับสี, ฮิสโตแกรมของความเข้มของสี, ฮิสโตแกรมของความสว่าง และฮิสโตแกรมของสีเทา เพื่อใช้เป็นดัชนีของภาพในรูปแบบของไฟล์ข้อมูลแล้วเก็บลงแฟ้มข้อมูล

1.4.2 การค้นหาและดึงภาพออกจากแฟ้มข้อมูล เมื่อผู้ใช้ต้องการหาข้อมูลของภาพเป้าหมาย จะต้องทำการกำหนดขอบเขตของภาพเป้าหมายก่อน ส่วนของภาพที่ได้กำหนดไว้จะถูกทำการแปลงสีและลดระดับสีให้อยู่ในรูปของ 166 สี HSV color space และทำการคำนวณค่าฮิสโตแกรมทั้ง 4 ประเภท ด้วยวิธีเดียวกันกับการเก็บภาพและดัชนีของภาพลงแฟ้มข้อมูล ฮิสโตแกรมที่คำนวณได้นี้จะนำไปเปรียบเทียบกับฮิสโตแกรมที่เก็บอยู่ในแฟ้มข้อมูลด้วยวิธีฮิสโตแกรมอินเตอร์เซกชันซึ่งวิธีการนี้จะทำการหาค่าความเหมือน (matching value) ระหว่างคู่ของภาพเป้าหมายและภาพโมเดล โดยถ้าค่าความเหมือนมีค่าเข้าใกล้ 1 มากเท่าไรก็แสดงว่า ภาพเป้าหมายมีความคล้ายคลึงกับภาพโมเดลนั้นๆ มากเท่านั้น

ในโครงงานนี้ยังถือว่าเป็นโครงการที่ทดลองสร้าง เพื่อศึกษาความเป็นไปได้ในการใช้งาน ดังนั้นจึงมีข้อจำกัดของรูปแบบบางประการ เช่น สีพื้นหลังของภาพโมเดลแต่ละภาพอาจจะใช้การเทสีเป็นสีเดียวเพื่อใช้ในการทดลองการค้นหาภาพและข้อมูลเพื่อหาข้อจำกัดอื่นๆของการใช้สีเป็นดัชนี

1.5 วิธีการดำเนินงาน

โครงงานนี้เริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับโครงงาน ซึ่งก็มีเรื่องหลัก ๆ อยู่ด้วยกัน 4 เรื่อง คือ

1. color space และคุณสมบัติต่างๆของสี
2. การแปลงสี การลดระดับสี และการกำหนดขอบเขตของภาพ
3. การทำฮิสโตแกรมของสี
4. การทำฮิสโตแกรมอินเตอร์เซกชัน

ซึ่งมีรายละเอียดดังในบทที่ 2, 3, 4 และ 5 ตามลำดับจากนั้นก็จะนำเอาความรู้ที่ได้ศึกษาทั้งหมดมาออกแบบโปรแกรมค้นหาภาพและข้อมูลของภาพ ซึ่งมีรายละเอียดในบทที่ 6

จากนั้นก็จะเริ่มเข้าสู่ขั้นตอนของการพัฒนาโปรแกรม โดยบทที่ 7 จะเป็นการทดลองโปรแกรมและทำการบันทึกผลการทดลองเพื่อหาค่าความเหมือนต่ำสุดที่เหมาะสมกับโปรแกรมรวมถึงปัญหาที่พบและการแก้ไข และบทที่ 8 ซึ่งเป็นบทสุดท้ายก็จะเป็นการสรุปการทำงาน ผลที่ได้รับจากโครงงานชิ้นนี้ ข้อจำกัดของชิ้นงานที่พบ แนวทางในการพัฒนางานโครงงานนี้เพิ่มเติม และแนวทางในการประยุกต์ใช้งาน

บทที่ 2
Color Space

2.1 บทนำ

โครงงานที่ได้จัดทำนี้เลือกใช้วิธีฮิสโตแกรมของสีโดยภาพจะต้องผ่านการแปลงสีและการลดระดับของสี ดังที่ได้กล่าวมาแล้วในบทที่ 1 ซึ่งใช้องค์ประกอบของสีเป็นดัชนีให้กับภาพ จึงจำเป็นต้องรู้รายละเอียดลักษณะเฉพาะของสีเสียก่อน จุดประสงค์ของบทนี้จะกล่าวถึงคุณสมบัติของสีและ color space ที่มีในระบบสีที่เหมาะสมกับวิธีฮิสโตแกรมของสีและ color space ที่เลือกใช้ในโครงงาน

2.2 คุณสมบัติของ color space

โดยทั่วไปแล้วภาพดิจิตอลจะอยู่ในรูปแบบของ RGB color space ซึ่งแทนสีทั้งหมดใน color space ด้วยเวกเตอร์ 3 มิติ เราจึงใช้ color space นี้เป็นจุดเริ่มต้นของลักษณะเฉพาะของภาพโดยใช้สี (color feature) จากนั้นค่อยทำการแปลงสีและลดระดับของสีให้มีคุณสมบัติที่เหมาะสมต่อการทำฮิสโตแกรม คุณสมบัติดังกล่าวคือ

2.2.1 Uniformity เป็นคุณสมบัติที่ใช้ในการเปรียบเทียบความใกล้เคียงของเมตริกสีโดย color space ที่มีคุณสมบัตินี้จะต้องคำนวณได้ง่าย อย่างเช่น การหาความใกล้เคียงของเมตริกสีโดยใช้ฟังก์ชันที่ไม่ขึ้นกับตำแหน่งใน color space นั้น การทำการแปลงสีจะเป็นวิธีที่ทำให้ color space มีคุณสมบัตินี้

2.2.2 Completeness เป็นคุณสมบัติสำคัญที่ color space จะต้องมีสีที่เราสามารถแยกแยะได้อย่างชัดเจน สามารถทำได้โดยการทำการลดระดับของสี

2.2.3 Compactness เป็นคุณสมบัติที่สีใน color space จะต้องแตกต่างจากสีอื่นๆ ใน color space เดียวกัน คุณสมบัตินี้จะเป็นตัวกำหนดมิติของภาพเพื่อให้แน่ใจว่า เมื่อผ่านการ map จาก RGB color space แล้วจะไม่มีสี 2 สีที่เหมือนกัน การลดระดับของสีจะทำให้เกิดคุณสมบัตินี้

2.2.4 Naturalness เป็นคุณสมบัติของสีที่ผู้ใช้สามารถเข้าใจได้ง่าย ทำให้ผู้ใช้สามารถทำการดึงข้อมูลจากฐานข้อมูลโดยใช้สีเป็นดัชนีได้ ซึ่งสีดังกล่าวจะถูกแยกแยะออกเป็น 3 ส่วน คือ ความสว่าง (brightness), ระดับสี (hue) และความเข้มของสี (saturation)

- ความสว่าง คือคุณสมบัติที่มีผลกระทบโดยตรงต่อสีในด้านของแสง เช่น การเปลี่ยนแปลงระหว่างความสว่างถึงความมืด (bright to dim)
- ระดับสี คือคุณสมบัติที่มีผลกระทบต่อสีที่จะทำให้สีเหมือนกับ สีแดง สีเหลือง สีเขียว สีน้ำเงิน ฯลฯ
- ความเข้มของสี คือคุณสมบัติที่มีผลกระทบต่อสีหนึ่งๆของระดับสี โดยจะพิจารณาถึงปริมาณของสี (colored light) โดยไม่สนใจถึงความสว่าง

เทคนิคของการแปลงสีจะต้องได้ color space ตรงตามคุณสมบัตินี้ อย่างไรก็ตามอาจขึ้นอยู่กับการนำไปใช้งานว่าต้องการคุณสมบัติของ color space ใดบ้าง เช่น การส่งภาพผ่านดาวเทียมจำเป็นจะต้องทำการแปลงสีให้มีคุณสมบัติเพียง uniformity, completeness และ compactness ก็เพียงพอ

ตารางที่ 2-1 แสดงการเปรียบเทียบการแปลงระดับของสีจาก RGB color space ไปเป็น color space ต่างๆ ในส่วนของโครงงานนี้เลือกใช้การแปลงสีจาก RGB color space ไปเป็น 166 สี HSV color space (เหตุผลที่เลือกจะกล่าวในบทที่ 3 เรื่องการแปลงสีและการลดระดับของสี) จากตารางมีเพียง 166 สี HSV color space เท่านั้นที่มีคุณสมบัติของ uniformity, compactness, completeness และ naturalness ครบถ้วน สำหรับ color space อื่นๆจะมีเพียงบางคุณสมบัติเท่านั้น

คุณสมบัติ	RGB	OPP/YIQ/YUV/YcrCb	Munsell	CIE	166 สี HSV
Uniformity	No	No	No	Yes	Yes
Compactness	No	No	Yes	Possible	Yes
Completeness	Yes	Yes	Yes	Yes	Yes
Naturalness	No	No	Yes	Yes	Yes

ตารางที่ 2-1 การเปรียบเทียบคุณสมบัติต่างๆของแต่ละ color space

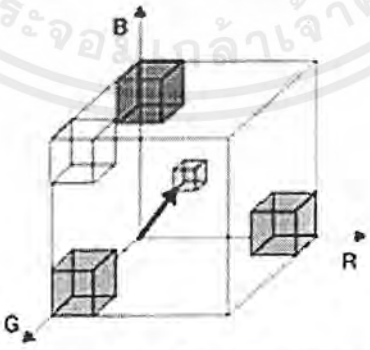
L = Linear

NL = Non-linear

2.3 Color Space

ในหัวข้อนี้จะกล่าวถึง color space ที่ใช้ในโครงงานเท่านั้น ซึ่งก็คือ RGB color space และ HSV color space

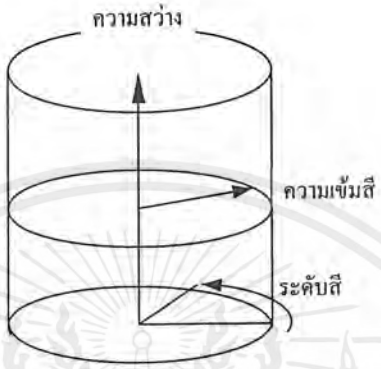
2.3.1 RGB color space ภาพสีที่อยู่ในรูปแบบของ RGB จะเป็นรูปแบบพื้นฐานของภาพที่มีการแสดงบนจอคอมพิวเตอร์โดยใช้ 1 พิกเซลของภาพประกอบด้วยองค์ประกอบสี 3 ค่า (channel) คือสีแดง, สีเขียว และสีน้ำเงิน โดยแสดงองค์ประกอบสีละ n บิต ดังนั้นจะสามารถแสดงเฉดสีได้องค์ประกอบสีละ 2^n เฉดสี ที่แตกต่างกัน รูปที่ 2.1 แสดงความสัมพันธ์ของสีแดง สีเขียว และสีน้ำเงินในหนึ่งพิกเซล



รูปที่ 2.1 แสดงความสัมพันธ์ของสีใน RGB Color space

ในระบบนี้แกน R จะแสดงปริมาณของสีแดงที่มีในพิกเซลนั้น แกน G จะแสดงปริมาณของสีเขียวที่มีในพิกเซลนั้น และแกน B จะแสดงปริมาณของสีน้ำเงินที่มีในพิกเซลนั้น เนื่องจากมีการเก็บสีแยกเป็นคนละค่าของสีกันทำให้เมื่อมีการคำนวณแยกทีละค่าของสีแล้วจะเกิดการ color shift สีของภาพที่ได้ก็จะเกิดการเพี้ยนไป

2.3.2 HSV Color space การกำหนดภาพสีในรูปแบบของ HSV (hue, saturation, value) HSI (hue, saturation, intensity) หรือ HLS (hue, lightness, saturation) ทั้งสามชื่อนี้เป็นการกล่าวถึง color space ในรูปแบบเดียวกันแต่ได้มีการเรียกชื่อต่างกันไป โดยความสัมพันธ์ที่แสดงโดยใช้ color space นี้จะเป็นเหมือนกับหลักการของจิตรกรคือการใช้ สี เงา และความเข้มของสี ดังรูปที่ 2.2 แสดงความสัมพันธ์ของระดับสี, ความเข้มสี และความสว่าง



รูปที่ 2.2 แสดงความสัมพันธ์ของระดับสี, ความเข้มสี และความสว่าง บน HSV color space

จากรูปค่าในแนวแกนตั้งของรูปทรงกระบอกจะแสดงระดับของความสว่าง ระยะห่างจากแกนกลางจะแสดงถึงระดับของความเข้มสีและองศาที่แสดงในแนวระนาบจะเป็นระดับสีที่ปรากฏใน 1 พิกเซล

ในระบบนี้ระดับสีคือสีที่อธิบายโดยใช้ความยาวคลื่นของสี ยกตัวอย่างเช่น ความแตกต่างระหว่างสีเหลืองและสีแดง ความเข้มสีเป็นปริมาณของสีที่ปรากฏ ยกตัวอย่างเช่น ความแตกต่างระหว่างสีแดงและสีชมพู แกนที่สามเรียกว่า ความสว่าง (lightness), ความเข้ม (intensity) หรือ ค่าของสี (value) เป็นปริมาณของแสง เช่น ความแตกต่างระหว่างสีแดงเข้ม (dark red) และสีแดงสว่าง (light red) หรือสีเทาเข้ม (dark grey) และสีเทาอ่อน (light grey)

ระบบสี HSV มีความเหมาะสมในการนำมาใช้ในโครงงานนี้ เนื่องจากสีที่ปรากฏในภาพเมื่อเราใช้ระดับสีในการแสดงสีจะเป็นการตอบสนองต่อการมองเห็นของมนุษย์โดยไม่มีผลกระทบของแสงเงาและการทำ image processing โดยใช้อัลกอริทึม spatial smoothing หรือ median filtering ในการลดการรบกวน (noise) ต่างๆในภาพที่อยู่ในรูปแบบของ RGB color space โดยแยกทำทีละค่าของสีจะมีผลทำให้เกิดการผิดเพี้ยนของสี แต่ถ้าใช้อัลกอริทึมเหล่านี้กับรูปภาพที่เป็น HSV color space จะไม่เกิดการผิดเพี้ยนของสี แต่เนื่องจากการทำงานอุปกรณ์แสดงภาพส่วนมากมักจะสนับสนุนการแสดงภาพในรูปแบบของ RGB color space เราจึงต้องใช้การคำนวณในการแปลงสีภาพที่อยู่ในรูปแบบของ RGB color space ให้มาอยู่ในรูปแบบของ HSV color space และเมื่อเราทำงานกับภาพนั้นเสร็จแล้วจะต้องทำการเปลี่ยนกลับให้มาอยู่ในรูปแบบของ RGB color space ดังเดิมเพื่อที่จะได้สามารถแสดงภาพบนหน้าจอได้

บทที่ 3
การแปลงสี การลดระดับของสี

3.1 บทนำ

จุดประสงค์ของการทำการแปลงสี คือการทำให้ภาพอยู่ใน color space ที่มีคุณสมบัติเหมาะสมต่องานที่จะนำไปใช้ ในโครงงานนี้เลือกใช้การแปลงสีให้อยู่ในรูป 166 สี HSV color space เนื่องจากว่าโครงงานที่จะจัดทำนี้ ต้องการภาพที่มีคุณสมบัติในการแยกสี ความสว่าง และความเข้มของสี ออกจากกันได้ มีสมการการแปลงสีที่สามารถเข้าใจได้ง่าย และทำการลดระดับของสีเพื่อกำหนดให้จำนวนสีมีขอบเขตที่จำกัด คือสามารถนับจำนวนสีเพื่อทำฮิสโตแกรมของสีได้

3.2 การทำการแปลงสีและการลดระดับของสีบน Color Space
(Color space Transformation and Quantization)

การแปลงสี (Color transformation, Tc) คือการแปลงสีจาก color space หนึ่งให้เป็นอีก color space หนึ่ง ปกติแล้วการแปลงสีอย่างง่ายจะเป็นการแปลงสีแบบเชิงเส้น (linear transformation) เช่น การแปลงสีจาก RGB color space ไปเป็น YIQ color space YcrCb color space หรือ OPP color space แต่ก็มีการแปลงสีที่ไม่เป็นแบบเชิงเส้น เช่น การแปลงสีจาก RGB color space ไปเป็น HSV color space, CIE 1976 (L*a*b*) และ CIE 1976 (L*u*v*) ซึ่งในโครงงานนี้จะเลือกใช้การแปลงสีแบบไม่เป็นเชิงเส้น เนื่องจากเราต้องการใช้ HSV color space รูปที่ 3.1 แสดงการแปลงสีจาก RGB color space ไปเป็น HSV color space

นิยาม ให้ $\hat{v}_c = (r,g,b)$ เป็น color point บน RGB color space

$\hat{w}_c = (h,s,v)$ เป็น color point บน HSV color space ซึ่งได้จากการแปลงสี

ซึ่ง $\hat{w}_c = Tc * \hat{v}_c$ สำหรับ r,g,b $\in [0\ldots1]$

เมื่อผ่าน Tc จะได้ h,s,v $\in [0\ldots1]$ โดยมีสมการการแปลงสี ดังสมการที่ 1-3

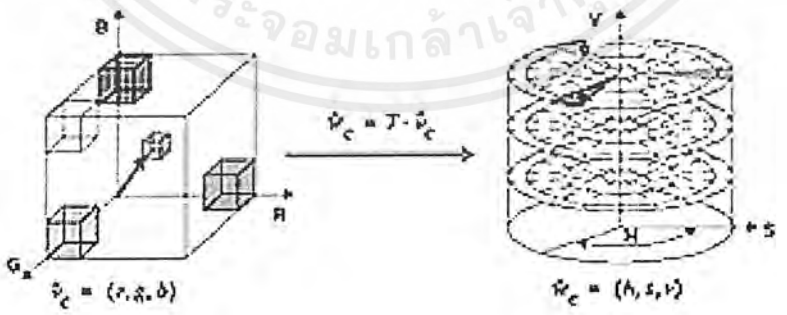
$$v = \max(r,g,b) \qquad (1)$$

$$s = \frac{v - \min(r,g,b)}{v} \qquad (2)$$

$$6h = \begin{cases} 5+b' & \text{ถ้า } r = \max(r,g,b) \text{ และ } g = \min(r,g,b) \\ 1-g' & \text{ถ้า } r = \max(r,g,b) \text{ และ } g \neq \min(r,g,b) \\ 1+r' & \text{ถ้า } g = \max(r,g,b) \text{ และ } b = \min(r,g,b) \\ 3-b & \text{ถ้า } g = \max(r,g,b) \text{ และ } b \neq \min(r,g,b) \\ 3+g' & \text{ถ้า } b = \max(r,g,b) \text{ และ } r = \min(r,g,b) \\ 5-r' & \text{ถ้าไม่ตรงกับกรณีที่กล่าวมาข้างต้น} \end{cases} \qquad (3)$$

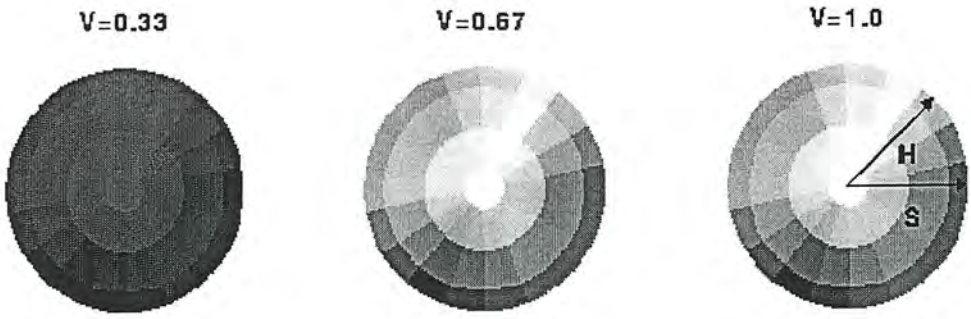
โดย $r' = \dfrac{v-r}{v-\min(r,g,b)}$, $g' = \dfrac{v-g}{v-\min(r,g,b)}$, $b' = \dfrac{v-b}{v-\min(r,g,b)}$

การลดระดับของสี (Color quantization, Qc) จะทำการลดจำนวนสีบน HSV color space ให้เหลือ 166 สี จากสมการการแปลงสีที่กล่าวมาข้างต้น จะได้ HSV color space เป็นรูปทรงกระบอก แสดงได้ดังรูปที่ 3.1ข โดยแกนแนวตั้ง (V) จะแทนความสว่างซึ่งก็คือค่าความมืดถึงความสว่าง (blackness to whiteness) แกนแนวนอน (S) จะแทนความเข้มสีซึ่งก็คือปริมาณของสี และมุมรอบแกนบนระนาบแนวนอน (H) แทนระดับสีซึ่งก็คือโทนสีที่ประกอบไปด้วย สีแดง สีเขียวและสีน้ำเงิน อย่างละ 120 องศา เมื่อทำการลดระดับของสี ที่วงระดับสีจะแบ่งทุกๆ 20 องศาในแต่ละสี จะได้ค่าระดับสีสีละ 6 ระดับ ซึ่งเพียงพอที่จะแทนได้ทั้งกลุ่มสีแดง (red), เขียว (green), น้ำเงิน (blue), เหลือง (yellow), แดงม่วง (magenta) และเหลืองน้ำเงิน (cyan) ในส่วนของความเข้มสีและความสว่างจะถูกลดระดับ โดยแบ่งเป็น 3 ระดับ จากการลดระดับจะได้สีที่แตกต่างกัน 166 สีบน HSV color space (ระดับสี 18 ระดับ * ความเข้มสี 3 ระดับ * ความสว่าง 3 ระดับ + สีเทาอีก 4 ระดับ = 166 ระดับ) ดังแสดงในรูปที่ 3.2 และ รูปที่ 3.3 แสดงตัวอย่างของการแปลงสีและลดระดับของสี โดยรูปที่ 3.3ข เกิดจากการแปลงสีและการลดระดับของสีของรูปที่ 3.3 ก (ด้วยวิธีที่ได้บรรยายมา)

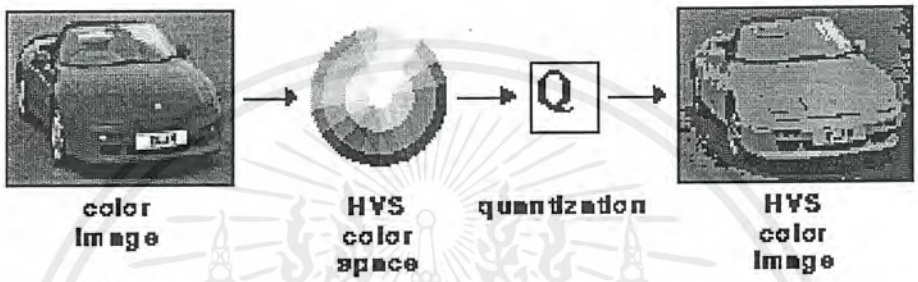


ก) ข)

รูปที่ 3.1 แสดงการแปลงสีจาก RGB color space ไปเป็น HSV color space



รูปที่ 3.2 แสดง 166 สี HSV color space ที่ได้จากการลดระดับของสีบน HSV color space



ก) ข)

รูปที่ 3.3 แสดงการแปลงสีและการลดระดับของสีภาพรถสีแดง

บทที่ 4
ฮิสโตแกรมของสี

4.1 บทนำ

จากบทที่ 3 การแปลงสีและการลดระดับของสีจะให้ค่าของสีที่สามารถนับจำนวนค่าของสีได้ ในการคำนวณและเก็บค่าของสีซึ่งแสดงถึงลักษณะต่างๆของภาพ การเก็บสีที่ปรากฏในภาพใช้วิธีการนับจำนวนของแต่ละสีที่ต่างกันตาม color space ที่ปรากฏในภาพและเก็บไว้โดยสร้างเป็นฮิสโตแกรมขึ้นมา ซึ่งฮิสโตแกรมนี้จะเป็นดัชนีเพื่อใช้ในการค้นหาภาพ

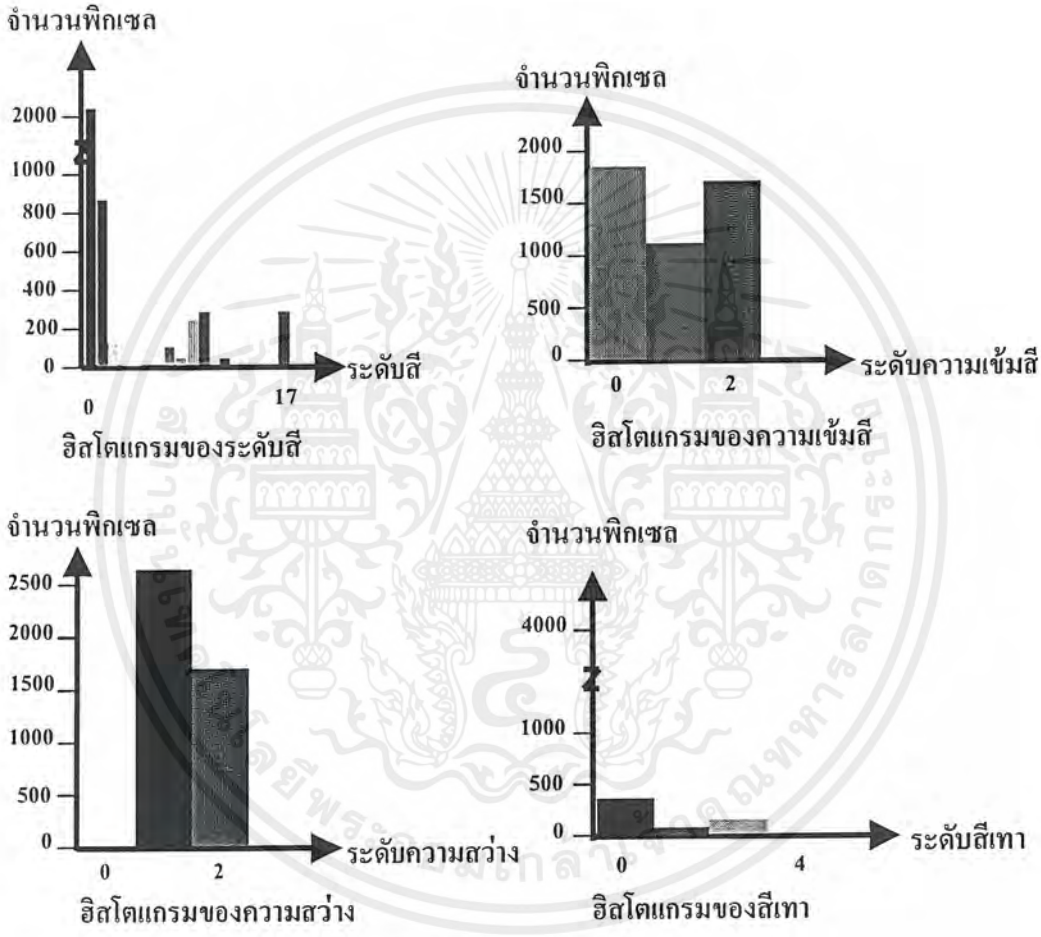
4.2 ฮิสโตแกรมของสี

ฮิสโตแกรมของสีประกอบไปด้วยระดับต่างๆ ของสี ซึ่งระดับต่างๆนี้จะเป็นจำนวนของสีในแต่ละระดับที่ปรากฏในภาพ ฮิสโตแกรมของสีจะมีความทนทานต่อการเปลี่ยนมุมมองของภาพ เนื่องจากการเปลี่ยนมุมมองจะทำให้จำนวนพิกเซลรวมของสีไม่เกิดการเปลี่ยนแปลงมากนัก แต่ถ้าเป็นรูปร่างของภาพเมื่อมีการเปลี่ยนมุมมองแล้วจะเกิดการเปลี่ยนแปลงไปมาก ฮิสโตแกรมจะเป็นเสมือนฟังก์ชันในการกำหนดความน่าจะเป็นของสีที่ปรากฏในภาพ ซึ่งค่าของสีเดียวกันจะต้องอยู่ในระดับของสีเดียวกัน ฮิสโตแกรมของสีไม่เหมาะที่จะใช้ในการทำการจดจำ (recognition) เนื่องจากการจะระบุว่าภาพใดเหมือนกันนั้นจะต้องดูที่กลุ่มของสีที่ปรากฏในรูปว่าอยู่ในตำแหน่งเดียวกันหรือไม่ แต่ฮิสโตแกรมของสีไม่สามารถบอกได้ว่าระดับสีในแต่ละระดับอยู่ในตำแหน่งใดของภาพ และจำนวนสีอาจเกิดการผิดเพี้ยนจากการรบกวนต่างๆ ได้ เช่น สีที่ได้จากเซ็นเซอร์ ซึ่งต่างจากวิธี color set ซึ่งเป็นการระบุกลุ่มสีในตำแหน่งต่างๆ ของภาพ อย่างไรก็ตาม ฮิสโตแกรมกลับได้รับความนิยมในการเป็นดัชนีของภาพ เพราะส่วนใหญ่แล้วพบว่า การที่สีที่ปรากฏในภาพมีความมัว เนื่องจากปัจจัยแวดล้อมต่างๆทำให้กลุ่มของสีที่ปรากฏในภาพเปลี่ยนตำแหน่งไป แต่ผลกระทบที่มีต่อฮิสโตแกรมของสีจะมีเพียงเล็กน้อยเท่านั้น ซึ่งทำให้การค้นหาภาพยังคงมีประสิทธิภาพที่ดีได้

4.3 การสร้างฮิสโตแกรมของสี

ในโครงงานที่ได้จัดทำนี้ การสร้างฮิสโตแกรมของสีโดยเก็บสีที่ปรากฏในภาพที่ผ่านการแปลงสีและลดระดับสีแล้ว ทำโดยใช้วิธีการนับจำนวนพิกเซลของแต่ละสีที่ต่างกันที่ปรากฏในภาพและเก็บไว้โดยสร้างเป็นฮิสโตแกรมของแต่ละภาพขึ้นมา ฮิสโตแกรมนี้จะเก็บอยู่ในรูปของไฟล์ข้อมูลพร้อมกับข้อมูลของภาพนั้นๆ เราจะทำการแยกเก็บฮิสโตแกรมของภาพออกเป็น 4 ฮิสโตแกรมโดยแบ่งตามแกนใน color space คือฮิสโตแกรมของระดับสี ฮิสโตแกรมของความเข้มสี ฮิสโตแกรมของความสว่าง และฮิสโตแกรมของสีเทา ดังตัวอย่าง รูปที่ 4.1 แสดงภาพที่อยู่ใน RGB color space ซึ่งเมื่อผ่านการแปลงสี การลดระดับของสีแล้ว สามารถสร้างฮิสโตแกรมจากภาพได้ 4 ฮิสโตแกรม ดังแสดงในรูปที่ 4.2

รูปที่ 4.1 แสดงภาพรถสีแดงที่อยู่ใน RGB color space



รูปที่ 4.2 แสดงฮิสโตแกรมของสีของภาพรถสีแดง

โดยแกนในแนวตั้งของกราฟแสดงจำนวนพิกเซลของแต่ละระดับที่ปรากฏในฮิสโตแกรม และแกนในแนวนอนของกราฟจะแสดงค่าระดับต่างๆของฮิสโตแกรม เช่น ในกราฟฮิสโตแกรมของระดับสี จะเห็นได้ว่าที่ระดับ 0 จะมีจำนวนพิกเซลมากที่สุด คือมากกว่า 2000 พิกเซล เนื่องจากภาพของรถคันนี้เป็นสีแดง ซึ่งมีค่าระดับสีส่วนใหญ่อยู่ที่ระดับ 0 จึงทำให้ระดับนี้มีจำนวนพิกเซลมาก

4.4 ข้อจำกัดของฮิสโตแกรมของสี

ข้อจำกัดอย่างหนึ่งของฮิสโตแกรมของสีคือ จำนวนพิกเซลในแต่ละฮิสโตแกรมแปรผันตามจำนวนพิกเซลที่มีอยู่ทั้งหมดในภาพ เนื่องจากขนาดของฮิสโตแกรมของระดับสีและขนาดของฮิสโตแกรมของสีเทารวมกันจะมีจำนวนเท่ากับพิกเซลรวมทั้งหมดในภาพ ดังนั้นฮิสโตแกรมของภาพที่มีขนาดต่างกันจะทำให้ประสิทธิภาพในการค้นหาลดลง เนื่องจากโครงงานนี้ใช้วิธีการค้นหาภาพโดยใช้วิธีฮิสโตแกรมอินเตอร์เซกชัน ซึ่งวิธีนี้เป็นการวัดการซ้อนทับกันของคู่ฮิสโตแกรมในแต่ละระดับ ดังจะกล่าวรายละเอียดในบทต่อไป อย่างไรก็ตาม ฮิสโตแกรมของภาพเดียวกันที่มีขนาดของภาพต่างกันจะทำให้การคำนวณการซ้อนทับของภาพผิดพลาดมากขึ้นและประสิทธิภาพในการค้นหาก็จะลดลง

บทที่ 5
ฮิสโตแกรมอินเตอร์เซกชัน

5.1 บทนำ

ในการค้นหาภาพและข้อมูลโดยใช้ฮิสโตแกรมของภาพเป็นดัชนีนั้น จำเป็นต้องใช้วิธีการค้นหาที่เหมาะสมกับโครงสร้างดัชนีด้วย ซึ่งวิธีฮิสโตแกรมอินเตอร์เซกชันเป็นวิธีที่ได้ออกแบบมาเพื่อใช้กับฮิสโตแกรมของสีโดยเฉพาะ จึงคิดว่าวิธีนี้เป็นวิธีที่เหมาะสมในการนำมาใช้ในโครงงาน ทั้งนี้ฮิสโตแกรมอินเตอร์เซกชันยังสามารถกำหนดวัตถุในภาพที่สนใจได้ด้วย ซึ่งเป็นข้อได้เปรียบหลักของฮิสโตแกรมอินเตอร์เซกชัน

5.2 ฮิสโตแกรมอินเตอร์เซกชัน

ในการค้นหาภาพ เราใช้วิธีดูความคล้ายกันของคู่ฮิสโตแกรมระหว่างฮิสโตแกรมของภาพเป้าหมายและฮิสโตแกรมของภาพโมเดล วิธีนี้เรียกว่าฮิสโตแกรมอินเตอร์เซกชัน วิธีนี้จะเป็นตัววัดว่ามีจำนวนพิกเซลของภาพโมเดลเท่าไรที่พบในภาพเป้าหมาย ซึ่งเป็นวิธีที่เหมาะสมกับภาพที่ใช้ฮิสโตแกรมเป็นดัชนี

5.3 การทำฮิสโตแกรมอินเตอร์เซกชัน

เนื่องจากวิธีฮิสโตแกรมอินเตอร์เซกชันเป็นวิธีที่ใช้ในการวัดการซ้อนทับกันระหว่างฮิสโตแกรมที่เราพิจารณา จึงทำการแยกพิจารณาทีละคู่ฮิสโตแกรมคือคู่ฮิสโตแกรมของระดับสี, คู่ฮิสโตแกรมของความเข้มของสี, คู่ฮิสโตแกรมของสีเทา และคู่ฮิสโตแกรมของความสว่าง

นิยาม กำหนดให้คู่ของฮิสโตแกรม I และ M โดยที่ I เป็นฮิสโตแกรมของภาพเป้าหมาย และ M เป็นฮิสโตแกรมของภาพโมเดล ที่เรานำมาพิจารณา เราจะสามารถหาค่าอินเตอร์เซกชั่นระหว่าง 2 ฮิสโตแกรมนี้ได้จากสมการที่ 4

$$\sum_{j=1}^{n} \min(Ij, Mj) \qquad (4)$$

โดยที่ n คือ จำนวนระดับทั้งหมดของฮิสโตแกรมที่สนใจ

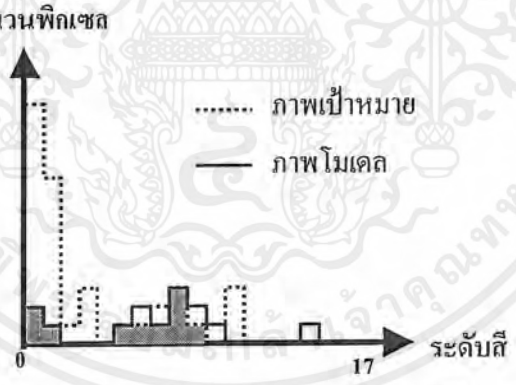
Ij คือ จำนวนพิกเซลของภาพเป้าหมายในระดับที่ j

Mj คือ จำนวนพิกเซลของภาพโมเดลในระดับที่ j

ผลที่ได้จากสมการนี้จะเป็นจำนวนรวมของพิกเซลของฮิสโตแกรมทั้งสองที่มีค่าเดียวกัน จากนั้นนำจำนวนรวมของพิกเซลของภาพโมเดลที่พิจารณามาหารกับค่าอินเตอร์เซกชัน ที่ได้จากสมการที่ 4 เพื่อให้ได้ค่าความเหมือน ดังสมการที่ 5

$$H(I,M) = \frac{\sum_{j=1}^{n} \min(Ij, Mj)}{\sum_{j=1}^{n} Mj} \qquad (5)$$

ค่าความเหมือนที่ได้จากสมการจะมีค่าอยู่ในช่วงระหว่าง 0 – 1 ค่าความเหมือนของภาพที่ใกล้เคียงกับ 1 มากที่สุดจะเป็นตัวชี้ว่าภาพคู่ใดมีความเหมือนมากที่สุด นอกจากนี้ เรายังสามารถหาค่าความเหมือนจากคู่ฮิสโตแกรมของระดับสี $H_h(I,M)$, ค่าความเหมือนจากคู่ฮิสโตแกรมของความเข้มของสี $H_s(I,M)$, ค่าความเหมือนจากคู่ฮิสโตแกรมของสีเทา $H_g(I,M)$ และค่าความเหมือนจากคู่ฮิสโตแกรมของความสว่าง $H_v(I,M)$ ได้ในลักษณะเดียวกันกับ H(I,M) ดังแสดงในรูปที่ 5.1 เป็นตัวอย่างการซ้อนทับกันของคู่ฮิสโตแกรม โดยเส้นประแสดงถึงฮิสโตแกรมของภาพเป้าหมาย เส้นทึบแสดงถึงฮิสโตแกรมของภาพโมเดล ส่วนที่ซ้อนทับกันและระบายสีทึบไว้แสดงส่วนที่เป็นฮิสโตแกรมอินเตอร์เซกชันของกราฟ ซึ่งจะเท่ากับจำนวนพิกเซลสูงสุดในแต่ละระดับที่ฮิสโตแกรมทั้งสองซ้อนทับกัน



รูปที่ 5.1 ฮิสโตแกรมอินเตอร์เซกชันของภาพเป้าหมายกับภาพโมเดล

5.4 ข้อจำกัดของการใช้ฮิสโตแกรมอินเตอร์เซกชัน

ค่าความเหมือนที่ได้อาจไม่ถูกต้องถ้า

1. พิกเซลที่เป็นพื้นหลังของภาพเป้าหมายมีสีเดียวกับพิกเซลของภาพโมเดล
2. จำนวนพิกเซลรวมของภาพเป้าหมายมีขนาดน้อยกว่าจำนวนพิกเซลรวมของภาพโมเดล

5.5 แนวทางในการลดข้อจำกัดของวิธีฮิสโตแกรมอินเตอร์เซกชัน

1. ตีกรอบภาพเป้าหมายเพื่อลดพื้นหลังที่ไม่ต้องการ

2. ทำการแปลงอัตราส่วนของภาพโมเดลให้มีขนาดใกล้เคียงกับภาพเป้าหมายให้มากที่สุดก่อนที่จะทำการเปรียบเทียบ โดยใช้วิธีทำฮิสโตแกรมสีให้เป็นมาตรฐาน (Normalization) ซึ่งทำได้โดยการหาอัตราส่วนของภาพเป้าหมายกับภาพโมเดล และนำอัตราส่วนนั้นไปคูณกับจำนวนสีในแต่ละระดับในฮิสโตแกรมของสีที่มีขนาดเล็กกว่า เพื่อให้จำนวนของสีในแต่ละระดับเพิ่มขึ้นจนใกล้เคียงกับฮิสโตแกรมที่ใหญ่กว่า ดังแสดงในรูปที่ 5.2

ภาพเป้าหมาย
ขนาด 100*100 พิกเซล

ภาพโมเดล
ขนาด 200*200 พิกเซล

รูปที่ 5.2 แสดงภาพเป้าหมายและภาพโมเดลที่ขนาดต่างกัน

จากรูปที่ 5.2 ภาพเป้าหมายขนาด 100 * 100 พิกเซลจะมีจำนวนพิกเซลในรูปเท่ากับ 10,000 พิกเซล และภาพโมเดลขนาด 200*200 พิกเซลจะมีจำนวนพิกเซลในภาพเท่ากับ 40,000 พิเซล ดังนั้นอัตราส่วนของภาพเป้าหมายต่อภาพโมเดลคือ 4 ก็เอา 4 ไปคูณกับจำนวนค่าของสีในแต่ละระดับของฮิสโตแกรมสีของภาพเป้าหมาย จะได้ภาพเป้าหมายที่มีจำนวนรวมของสีในแต่ละระดับในฮิสโตแกรมสีเท่ากับจำนวนรวมของสีในแต่ละระดับในฮิสโตแกรมสีของภาพโมเดล

5.6 ข้อได้เปรียบของการเปรียบเทียบดัชนีโดยวิธีฮิสโตแกรมอินเตอร์เซกชัน

1. ถึงแม้ภาพเป้าหมายจะเปลี่ยนมุมมองก็ยังจะสามารถรู้ได้ว่าเป็นภาพเดียวกัน เนื่องจากฮิสโตแกรมซึ่งสร้างจากองค์ประกอบสีจะยังมีค่าใกล้เคียงกันในต่างมุมมอง
2. ช่วยแก้ปัญหาในการแยกภาพเป้าหมายกับพื้นหลังของภาพ เนื่องจากเราจะใช้ฮิสโตแกรมของภาพเป้าหมายเป็นหลัก ดังนั้นถึงแม้ภาพเป้าหมายจะมีจำนวนพิกเซลรวมของแต่ละสีที่มากกว่าภาพโมเดล ก็จะยังได้ค่าความเหมือนที่ใกล้เคียงกับ 1

บทที่ 6
แนวคิดและการออกแบบ

6.1 บทนำ

ในการออกแบบตัวโปรแกรมค้นหาภาพในแฟ้มข้อมูล จะต้องกำหนดให้โปรแกรมสามารถทำงานได้ตรงตามความต้องการดังนี้คือ โปรแกรมจะต้องสามารถทำการรับภาพเป้าหมายจากผู้ใช้ ค้นหาภาพในแฟ้มข้อมูล และแสดงภาพที่ใกล้เคียงกับภาพเป้าหมายรวมทั้งข้อมูลของภาพนั้น นอกจากนี้ผู้ใช้ยังสามารถเพิ่มภาพและข้อมูลที่ต้องการลงในแฟ้มข้อมูลได้ เมื่อกำหนดการทำงานของโปรแกรมได้แล้ว จึงระบุการทำงานในส่วนหลักของโปรแกรมเพื่อใช้เป็นโครงร่างในการพัฒนารายละเอียดของโปรแกรม ในบทนี้ จะกล่าวถึงการพัฒนาโปรแกรมโดยจะกล่าวถึงการทำงานของฟังก์ชันหลักก่อนเพื่อให้เข้าใจการทำงานรวมของโปรแกรม จากนั้นจึงจะอธิบายรายละเอียดของฟังก์ชันและส่วนต่างๆในโปรแกรม รวมทั้งฟังก์ชันย่อยที่เป็นองค์ประกอบของโปรแกรม

6.2 การพัฒนาโปรแกรม

ในโครงงานนี้เลือกใช้ภาษา visual C++ เวอร์ชัน 6.0 ในการพัฒนาโปรแกรม เนื่องจากมีไลบรารี่ให้เลือกใช้เกี่ยวกับการจัดการภาพ ทำให้สะดวกในการใช้งาน โปรแกรมที่พัฒนาขึ้นมีฟังก์ชันการทำงานดังนี้

1. รับภาพจากผู้ใช้ คำนวณขอบเขตของภาพเป้าหมาย
2. แปลงสีและลดระดับสี โดยทำการแปลงสีและลดระดับสีทีละพิกเซล พิกเซลกลุ่มสีเทาจะถูกแยกออกมาก่อน เนื่องจากนิยามการแปลงสีไม่สามารถแปลงพิกเซลสีที่เป็นสีเทา(ซึ่ง R, G, B จะมีค่าเท่ากัน)ได้ พิกเซลของสีเทาจะไม่ต้องทำการแปลงสีและข้ามไปทำการลดระดับสีได้เลย พิกเซลที่ไม่ใช่สีเทาจะถูกแปลงสีตามนิยามการแปลงสีที่ได้กล่าวไปแล้วในหัวข้อทฤษฎีและหลักการ ภาพที่ได้จะอยู่ในรูป HSV color space ที่มีค่าอยู่ระหว่าง 0-1 แล้วทำการลดระดับสีจาก HSV color space ให้เป็น 166 สี HSV color space ซึ่งทำได้ดังนี้โดย

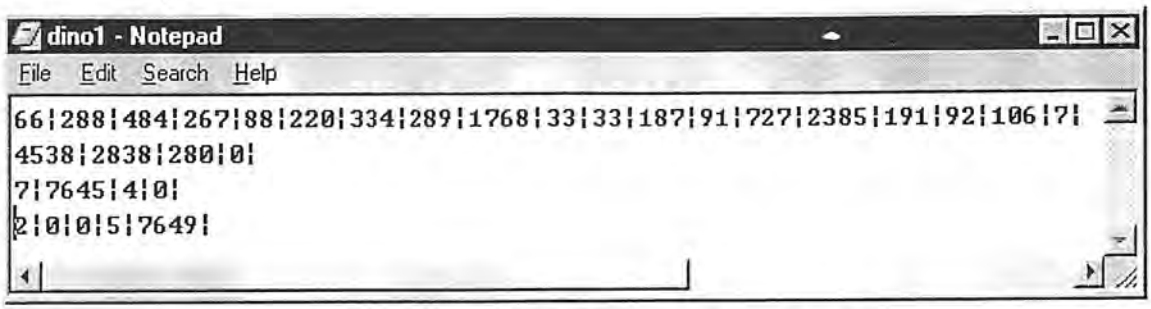
หากเป็นพิกเซลสีเทาจะถูกแบ่งออกเป็น 4 ระดับดังนี้ ตั้งแต่ 0 – 63.75 เป็นระดับที่ 0, มากกว่า 63.75 – 127.5 เป็นระดับที่ 1, มากกว่า 127.5 – 191.25 เป็นระดับที่ 2 และ มากกว่า 191.25 – 255 เป็นระดับที่ 3 และกำหนดให้มีระดับที่ 4 เก็บจำนวนพิกเซลที่ไม่ใช่สีเทา ดังนั้นระดับที่ 4 นี้จึงไม่ใช่ค่าของพิกเซลที่ได้จากการแบ่งระดับสี และกำหนดให้ค่าของความเข้มของสีเป็นระดับที่ 0 เสมอ ค่าของระดับสี ซึ่งไม่สามารถระบุสีได้(ความเข้มสีที่เท่ากับ 0 และค่าของระดับสีที่ไม่สามารถระบุได้นี้เป็นคุณสมบัติของสีเทาใน HSV color space)จะถูกกำหนดให้เป็นระดับที่ 18 (ระดับที่ 18 เป็นการเก็บจำนวนพิกเซลที่ไม่สามารถระบุสีได้ ไม่ใช่ระดับสี) ค่าของความสว่าง สามารถคำนวณตามนิยามการแปลงสีได้ โดยแบ่งได้

เป็น 3 ระดับคือ ตั้งแต่ 0 - 85 เป็นระดับที่ 0, มากกว่า 85 – 170 เป็นระดับที่ 1 และ มากกว่า 170 – 255 เป็นระดับที่ 2

หากว่าไม่ใช่พิกเซลของสีเทาจะสามารถแปลงค่าของระดับสี ให้เป็น 18 ระดับคือ ตั้งแต่ 0 - 0.055555 เป็นระดับที่ 0, มากกว่าหรือเท่ากับ 0.055555 - 0.111111 เป็นระดับที่ 1, มากกว่ามากกว่าหรือเท่ากับ 0.111111 – 0.166666 เป็นระดับที่ 2, มากกว่ามากกว่าหรือเท่ากับ 0.166666 – 0.222222 เป็นระดับที่ 3, มากกว่ามากกว่าหรือเท่ากับ 0.222222 – 0.277777 เป็นระดับที่ 4, มากกว่ามากกว่าหรือเท่ากับ 0.277777 – 0.333333 เป็นระดับที่ 5, มากกว่ามากกว่าหรือเท่ากับ 0.333333 – 0.388888 เป็นระดับที่ 6, มากกว่ามากกว่าหรือเท่ากับ 0.388888 – 0.444444 เป็นระดับที่ 7, มากกว่ามากกว่าหรือเท่ากับ 0.444444 – 0.5 เป็นระดับที่ 8, มากกว่ามากกว่าหรือเท่ากับ 0.5 – 0.555555 เป็นระดับที่ 9, มากกว่ามากกว่าหรือเท่ากับ 0.55555 – 0.611111 เป็นระดับที่ 10, มากกว่ามากกว่าหรือเท่ากับ 0.611111 – 0.666666 เป็นระดับที่ 11, มากกว่ามากกว่าหรือเท่ากับ 0.666666 – 0.722222 เป็นระดับที่ 12 ,มากกว่ามากกว่าหรือเท่ากับ 0.722222 – 0.777777 เป็นระดับที่ 13 ,มากกว่ามากกว่าหรือเท่ากับ 0.777777 – 0.833333 เป็นระดับที่ 14, มากกว่ามากกว่าหรือเท่ากับ 0.833333 – 0.888888 เป็นระดับที่ 15, มากกว่ามากกว่าหรือเท่ากับ 0.888888 – 0.944444 เป็นระดับที่ 16, มากกว่ามากกว่าหรือเท่ากับ 0.944444 - น้อยกว่า1.0 เป็นระดับที่ 17 (ส่วนค่าของสีที่ 1 จะเป็นระดับที่ 0) และจะมีระดับที่ 18 เป็นระดับที่ได้จากสีเทาซึ่งไม่สามารถคำนวณค่าระดับสีได้ จะถูกเก็บลงในระดับที่ 18 นี้ ในส่วนของค่าความเข้มของสีและส่วนของความสว่าง จะแบ่งได้เป็น 3 ระดับ คือ ตั้งแต่ 0 – 0.333333 เป็นระดับที่ 0 ,มากกว่า 0.333333 – 0.666666 เป็นระดับที่ 1 ,มากกว่า 0.666666 - 1.0 เป็นระดับที่ 2

3. สร้างและเก็บฮิสโตแกรมลงแฟ้มข้อมูล โดยทำการนับค่าของระดับสี ความเข้มสี ความสว่าง และสีเทาที่ผ่านการลดระดับของสีแล้วจนกระทั่งครบทุกพิกเซลในภาพ และเก็บลงไฟล์ฮิสโตแกรมในรูปของไฟล์ข้อมูล การเก็บฮิสโตแกรมจะแยกเก็บเป็นระดับสี ระดับของความเข้มสี ระดับความสว่างและระดับของสีเทา รูปที่ 6.1 แสดงภาพไดโนเสาร์สีม่วงที่จะทำการคำนวณฮิสโตแกรมของสี และรูปที่ 6.2 แสดงไฟล์ข้อมูลฮิสโตแกรมทั้งสี่ของภาพไดโนเสาร์ในรูปที่ 6.1

รูปที่6.1 แสดงภาพไดโนเสาร์



รูปที่ 6.2 แสดงข้อมูลฮิสโตแกรมทั้งสี่ของภาพไดโนเสาร์ ในรูป 6.1

จากรูปที่ 6.2 แต่ละระดับของฮิสโตแกรมจะคั่นด้วยเครื่องหมาย " | " เพื่อให้อ่านได้ง่าย ตัวเลขบรรทัดแรกแสดงจำนวนพิกเซลของระดับสีในแต่ละระดับ 18 ระดับ ซึ่งเทียบได้กับระดับที่ 0-17 ดังที่แสดงในกราฟฮิสโตแกรมของระดับสีในรูปที่ 4.2 และระดับสุดท้ายของระดับสีในไฟล์ข้อมูลฮิสโตแกรมแสดงจำนวนของพิกเซลที่ไม่สามารถคำนวณได้จากนิยามการแปลงสี บรรทัดที่สองแสดงจำนวนพิกเซลของความเข้มสีในแต่ละระดับรวมทั้งสิ้น 4 ระดับ ซึ่ง 3 ระดับแรกเทียบได้กับระดับที่ 0-2 ดังที่แสดงในกราฟฮิสโตแกรมของความเข้มสีในรูปที่ 4.2 (ระดับที่ 4 มีไว้แสดงความผิดพลาดของการคำนวณฮิสโตแกรม ซึ่งปกติจะมีค่าเป็น 0 แต่ถ้ามีความผิดพลาดเกิดขึ้นจากการคำนวณจะมีค่ามากกว่า 0) เช่นเดียวกับบรรทัดที่ 3 แสดงจำนวนพิกเซลของความสว่างในแต่ละระดับรวมทั้งสิ้น 4 ระดับ บรรทัดสุดท้ายแสดงระดับของสีเทา 5 ระดับ ซึ่ง 4 ระดับแรกเทียบได้กับระดับที่ 0-3 ดังที่แสดงในกราฟฮิสโตแกรมของสีเทาในรูปที่ 4.2 โดยระดับที่ 5 แสดงจำนวนของพิกเซลที่สามารถคำนวณด้วยนิยามการแปลงสีได้ ซึ่งจำนวนพิกเซลรวมทั้งหมดของทุกระดับในแต่ละฮิสโตแกรมจะมีค่าเท่ากันเสมอคือเท่ากับจำนวนพิกเซลรวมทั้งหมดในภาพ ส่วนแฟ้มข้อมูลจะเป็นไดเรกทอรี่ที่เก็บรูปภาพและไฟล์ข้อมูลฮิสโตแกรม

4. ค้นหาภาพในแฟ้มข้อมูล ในแฟ้มข้อมูลจะมีภาพและฮิสโตแกรมที่ได้จากการคำนวณฮิสโตแกรมโดยผ่านการแปลงสีและการลดระดับสีด้วยวิธีเดียวกันนี้ การค้นหาจะทำการคำนวณค่าฮิสโตแกรมเฉพาะส่วนที่ผู้ใช้เลือกจากการตีกรอบส่วนของภาพที่ต้องการ จากนั้นจะทำการดึงไฟล์ข้อมูลฮิสโตแกรมในแฟ้มข้อมูลทีละไฟล์ ฮิสโตแกรมที่ได้จะถูกนำไปเปรียบเทียบกับฮิสโตแกรมที่ดึงมาจากแฟ้มข้อมูลโดยวิธีฮิสโตแกรมอินเตอร์เซกชัน ซึ่งคำนวณได้จากนิยามของฮิสโตแกรมอินเตอร์เซกชันที่ได้กล่าวไปแล้วในหัวข้อทฤษฎีและหลักการ ค่าที่ได้จะเป็นค่าความเหมือน (matching value) ซึ่งได้คำนวณค่าความเหมือนจากฮิสโตแกรมทั้งหมดและเฉพาะฮิสโตแกรมของสีและเก็บค่าความเหมือนนี้ในเรคคอร์ดหนึ่งของลิสต์ เรคคอร์ดในลิตส์นี้ประกอบด้วย ชื่อไฟล์รูปภาพ, ค่าความเหมือน, ค่าความเหมือนของสี, ขนาดของภาพ, ข้อมูลของภาพ และพอยเตอร์ที่ชี้ไปยังเรคคอร์ดอื่นๆในลิสต์ เรคคอร์ดในลิสต์จะเรียงลำดับตามค่าความเหมือนของสีที่มากที่สุดไปยังน้อยสุด เราจะเลือกภาพโมเดลที่มีความเหมือนกับภาพเป้าหมาย โดยดูจากค่าความเหมือนของระดับสีที่ใกล้เคียงกับ 1 มากที่สุด เพราะเป็นค่าความเหมือนที่ไม่ได้นำความสว่างเข้ามาคำนวณด้วย จะทำให้ค้นหาภาพที่มีความสว่างต่างกันได้ ภาพที่แสดงผลจะเป็นภาพที่มีค่าความเหมือนของระดับสีมากกว่าค่าความเหมือนต่ำสุดที่กำหนด และจะแสดงผลภาพที่มีค่าความเหมือนมากที่สุดทีละ 5 ภาพ ค่าความเหมือนต่ำสุดนี้ได้จากการทดสอบโปรแกรมเพื่อหาค่าที่เหมาะสมที่สุด

5. ฟังก์ชันอื่นๆ เช่นการตีกรอบภาพเป้าหมาย เป็นการกำหนดขอบเขตให้กับวัตถุในภาพที่สนใจ และจะนำเอาพิกเซลสีที่อยู่ในกรอบนี้ไปคำนวณหาฮิสโตแกรมของภาพ การเลือกสีกรอบเพื่อไม่ให้สีของภาพกลืนไปกับสีของกรอบ ซึ่งจะทำให้ผู้ใช้กำหนดกรอบของภาพได้ง่ายขึ้น การดึงข้อมูลของภาพเพื่อแสดงผล โดยข้อมูลของภาพจะเก็บรวมอยู่ในแฟ้มข้อมูลด้วย และการกำหนดค่าความเหมือนต่ำสุดได้เอง ซึ่งจะทำให้ค้นหาภาพได้กว้างขึ้น

จากขั้นตอนทั้งหมดสามารถสรุปการทำงานได้ดัง flow chart แสดงการเก็บภาพลงแฟ้มข้อมูล และ flow chart แสดงการค้นหาและการเรียกภาพนิ่ง ในส่วนของภาคผนวก ค.

บทที่ 7
การทดลองโปรแกรม

7.1 บทนำ

ในบทนี้จะกล่าวถึงการออกแบบการทดลองเพื่อทดสอบประสิทธิภาพของโปรแกรมในการค้นหาภาพ โดยกำหนดชุดการทดลองต่างๆขึ้นมาชุดละ 20 ภาพ แล้วทดลองให้โปรแกรมค้นหาภาพในแต่ละชุดการทดลองที่ค่าความเหมือนต่างๆกัน จากนั้นเก็บจำนวนภาพทั้งหมดที่เรียกได้และจำนวนภาพที่เรียกได้ถูกต้อง เพื่อวิเคราะห์ความสามารถของโปรแกรมในแต่ละชุดการทดลอง และคำนวณค่าความเหมือนต่ำสุดซึ่งจะนำไปใช้เป็นค่าความเหมือนเริ่มต้นให้กับโปรแกรม จากนั้นจะทดลองให้โปรแกรมค้นหาภาพในแฟ้มข้อมูลที่มีจำนวนภาพทั้งหมด 67 ภาพ ที่ค่าความเหมือนเริ่มต้นและเก็บจำนวนภาพทั้งหมดที่เรียกได้และจำนวนภาพที่เรียกได้ถูกต้อง เพื่อวิเคราะห์ถึงผลกระทบของประสิทธิภาพของโปรแกรมเมื่อแฟ้มข้อมูลมีขนาดใหญ่ขึ้น

7.2 การทดลองการค้นหาภาพ

ภาพที่ใช้ในการทดสอบประกอบไปด้วยภาพต่างๆที่ได้มาจากการถ่ายด้วยกล้องดิจิตอลและอินเตอร์เน็ต ซึ่งเป็นภาพที่มีทั้งความสว่างต่างกัน มุมมองในภาพต่างกัน ขนาดของภาพต่างกัน ฯลฯ ดังนั้นจึงได้ออกแบบการทดลองเพื่อทดสอบโปรแกรมให้ค้นหาภาพดังกล่าวได้โดยใช้ความสามารถของวิธีที่เลือกใช้ในโครงงานนี้ ในการทดลองแต่ละชุดจะกำหนดให้ค่าความเหมือนต่ำสุดที่ใช้ในการค้นหามีค่าต่างๆกัน โดยให้มีค่าเป็น 60 , 70 , 80 และ 90 ตามลำดับ เนื่องจากเมื่อกำหนดค่าความเหมือนต่ำสุดน้อยกว่า 60 จะทำให้พบภาพเกือบทั้งหมดในแฟ้มข้อมูล การทดลองโดยกำหนดให้มีค่าความเหมือนต่างๆกันจะช่วยให้เห็นแนวโน้มของการค้นหาว่า สามารถลดภาพที่ไม่ต้องการได้มากน้อยแค่ไหนและยังคงพบภาพที่ต้องการได้ทั้งหมดหรือไม่ จากนั้นจะเลือกค่าความเหมือนที่เหมาะสมในแต่ละกรณีทดลอง มาคำนวณหาค่าความเหมือนต่ำสุดที่เหมาะสมจะใช้เป็นค่าเริ่มต้นของโปรแกรม และเก็บจำนวนภาพทั้งหมดที่เรียกได้และภาพที่เรียกได้ถูกต้อง มาวิเคราะห์ถึงประสิทธิภาพในการค้นหาโดยรวมของโปรแกรมเมื่อใช้ค่าความเหมือนต่ำสุดนี้ในการค้นหา

การทดลองโปรแกรม กำหนดให้มีภาพในแฟ้มข้อมูลที่ต่างกันเป็นชุด ชุดละ 20 ภาพโดยทดลองที่ค่าความเหมือนเท่ากับ 60 การทดลองโปรแกรมแบ่งเป็น 5 ชุดดังนี้

7.2.1 ทดลองการค้นหาภาพเมื่อภาพที่ต้องการมีความสว่างต่างจากภาพเป้าหมาย ซึ่งเป็นความสามารถของการแปลงสีและการลดระดับสี ที่จะแยกค่าของสีออกจากความเข้มและความสว่าง การทดลองจะกำหนดให้มีกลุ่มของภาพที่คล้ายกันในแฟ้มข้อมูล แต่ละกลุ่มมีภาพที่มีความสว่างต่างกัน และให้โปรแกรมรับภาพเป้าหมายที่มีความสว่างต่างกับภาพในแฟ้มข้อมูล โปรแกรมจะต้องค้นหาภาพที่คล้ายกับภาพเป้าหมายได้

การทดลองชุดที่ 1

จุดประสงค์ ทดสอบการค้นหาภาพเมื่อความสว่างต่างกัน

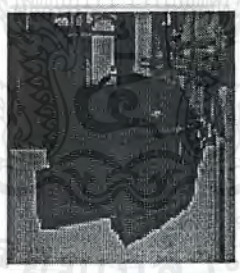
ภาพเป้าหมาย ภาพเสื้อเชิ้ตสีกรมท่าที่อยู่ในความมืด

กลุ่มของภาพในแฟ้มข้อมูล

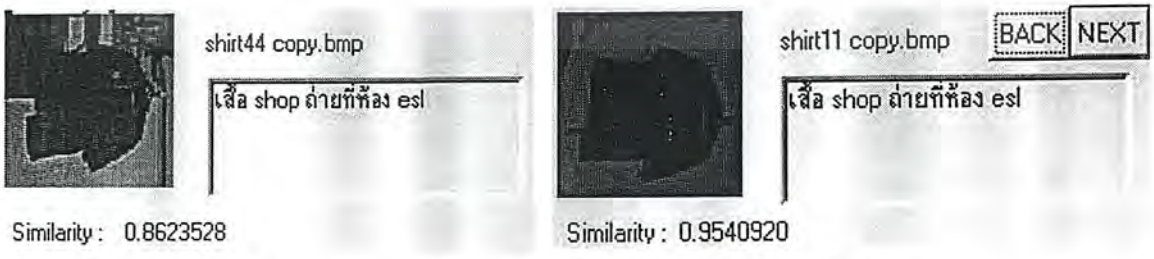
1. ภาพเสื้อเชิ้ตสีกรมท่า 4 ภาพที่มีความสว่างต่างกัน
2. ภาพโดราเอมอน 4 ภาพที่มีความสว่างต่างกัน
3. ภาพการ์ตูนส์ลามู 7 ภาพที่มีความสว่างต่างกัน
4. ภาพโปเกมอน 5 ภาพที่มีความสว่างต่างกัน

เกณฑ์ สามารถค้นหาภาพเสื้อเชิ้ตสีกรมท่าที่มีความสว่างต่างกัน ได้จำนวนใกล้เคียงกับ 4 ภาพมากที่สุด

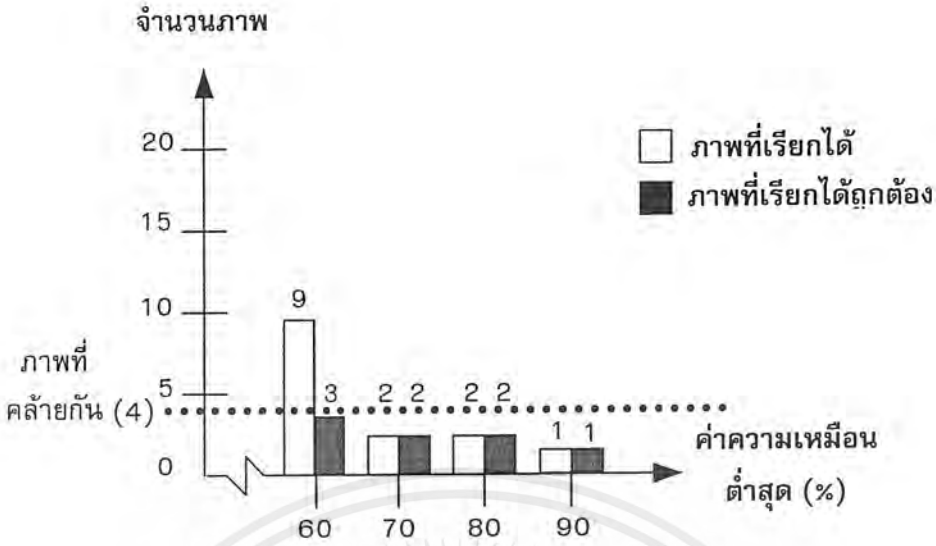
ผลการทดลอง โปรแกรมสามารถหาภาพเสื้อที่มีความสว่างต่างกันที่อยู่ในแฟ้มข้อมูลและแสดงผล 2 อันดับแรกได้ถูกต้อง แสดงว่าโปรแกรมสามารถค้นหาภาพที่มีความสว่างแตกต่างกันได้ รูปที่ 7.1ก เป็นภาพเป้าหมายที่รับจากผู้ใช้ และรูปที่ 7.1ข เป็นภาพที่ได้จากการแสดงผลการค้นหาในแฟ้มข้อมูล 2 ภาพ



รูปที่ 7.1ก แสดงภาพเป้าหมายที่ใช้ในการค้นหา



รูปที่ 7.1ข แสดงตัวอย่างภาพเสื้อที่โปรแกรมค้นหาได้จากแฟ้มข้อมูล



รูปที่ 7.2 แสดงกราฟแท่งเปรียบเทียบการค้นหาภาพที่มีความสว่างต่างกัน ของโปรแกรมที่ค่าความเหมือนต่างๆ

รูปที่ 7.2 เป็นกราฟแท่งเปรียบเทียบจำนวนภาพที่โปรแกรมค้นหาได้และจำนวนภาพที่ต้องการที่โปรแกรมค้นหาได้ที่ค่าความเหมือนต่ำสุดต่างๆ โดยแกนแนวตั้งแสดงจำนวนภาพในแฟ้มข้อมูล และแกนแนวนอนแสดงค่าความเหมือนต่ำสุดเป็นเปอร์เซ็นต์ เส้นประแสดงจำนวนภาพที่ต้องการทั้งหมดที่อยู่ในแฟ้มข้อมูล กราฟแท่งสีขาวแสดงภาพที่โปรแกรมค้นหาได้ทั้งหมดที่คู่ของฮิสโตแกรมของภาพเป้าหมายและภาพโมเดลมีค่าความเหมือนมากกว่าที่กำหนดไว้ และกราฟแท่งสีดำแสดงภาพที่ต้องการที่โปรแกรมค้นหาได้ที่คู่ของฮิสโตแกรมของภาพเป้าหมายและภาพโมเดลมีค่าความเหมือนมากกว่าที่กำหนดไว้ เมื่อทำการกำหนดค่าความเหมือนให้มีค่าต่างๆกัน พบว่าที่ค่าความเหมือนระดับ 0.7 หรือ 70 เปอร์เซ็นต์ และระดับ 0.8 หรือ 80 เปอร์เซ็นต์ โปรแกรมสามารถค้นหาภาพที่มีความสว่างต่างกันได้ดีที่สุด

เมื่อภาพมีความสว่างเปลี่ยนไป พิกเซลสีที่กระจายในภาพจะเปลี่ยนไปด้วยเล็กน้อย เมื่อสร้างเป็นฮิสโตแกรม สีที่เปลี่ยนไปในภาพจะตกอยู่ในระดับสีอื่นที่ใกล้เคียงกับระดับสีเดิม ทำให้คำนวณค่าความเหมือนน้อย ดังจะเห็นได้จากกราฟว่า เมื่อกำหนดค่าความเหมือนต่ำสุดที่ 60 จะพบภาพที่เกี่ยวข้องได้ใกล้เคียงกับภาพที่ต้องการทั้งหมดในแฟ้มข้อมูล

7.2.2 ทดลองการค้นหาภาพเมื่อภาพที่ต้องการมีมุมมองภาพต่างกัน ซึ่งเป็นความสามารถของวิธีฮิสโตแกรมอินเตอร์เซกชัน ที่จะจดจำวัตถุที่มีการเปลี่ยนมุมมอง (ที่มีพื้นหลังเหมือนกัน)ได้ การทดลองจะกำหนดให้มีกลุ่มของภาพที่คล้ายกันในแฟ้มข้อมูล แต่ละกลุ่มมีภาพที่มีมุมมองภาพต่างกัน และให้โปรแกรมรับภาพเป้าหมายที่มีมุมมองภาพต่างกับภาพในแฟ้มข้อมูล โปรแกรมจะต้องค้นหาภาพที่คล้ายกับภาพเป้าหมายได้

การทดลองชุดที่ 2

จุดประสงค์ ทดสอบการค้นหาภาพเมื่อมีมุมมองภาพต่างกัน

ภาพเป้าหมาย ภาพไดโนเสาร์สีม่วง

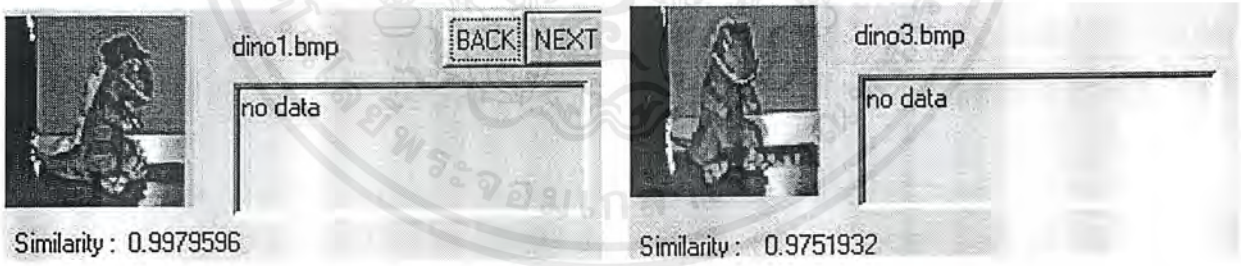
กลุ่มของภาพในแฟ้มข้อมูล

1. ภาพไดโนเสาร์ 8 ภาพที่มีมุมมองภาพต่างกัน
2. ภาพโดราเอมอน 4 ภาพที่มีมุมมองภาพต่างกัน
3. ภาพโปเกมอน 5 ภาพที่มีมุมมองภาพต่างกัน
4. ภาพนกทวิตตี้ 4 ภาพที่มีมุมมองภาพต่างกัน

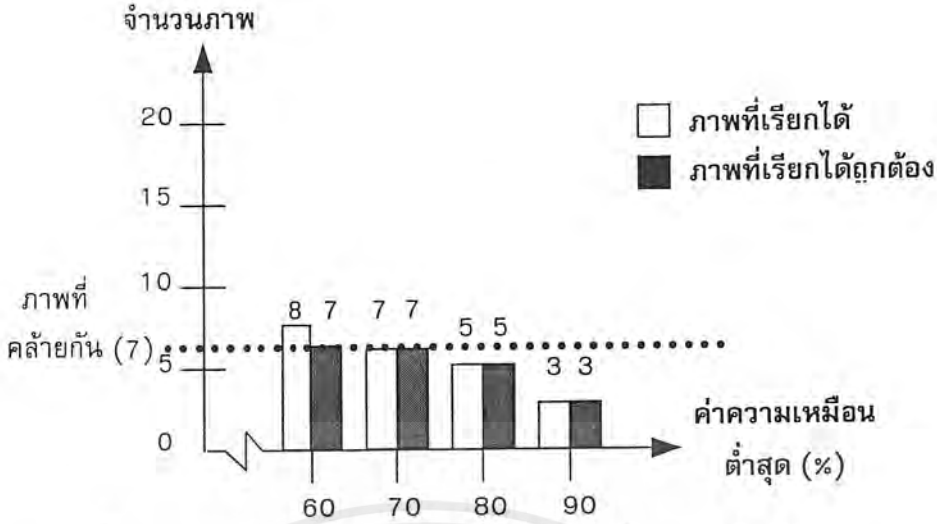
เกณฑ์ สามารถค้นหาภาพไดโนเสาร์สีม่วงในท่าทางต่างๆได้ใกล้เคียง 8 ภาพมากที่สุด

ผลการทดลอง โปรแกรมสามารถหาภาพไดโนเสาร์ที่อยู่ในแฟ้มข้อมูลและแสดงผล 7 อันดับแรกได้ถูกต้อง แสดงว่าโปรแกรมสามารถค้นหาภาพที่มีมุมมองแตกต่างกันได้ รูปที่ 7.3ก เป็นภาพเป้าหมายที่รับจากผู้ใช้ และรูปที่ 7.3ข เป็นภาพที่ได้จากการแสดงผลการค้นหาในแฟ้มข้อมูล 2 ภาพ

รูปที่ 7.3ก แสดงภาพไดโนเสาร์ที่ผู้ใช้ต้องการค้นหา



รูปที่ 7.3ข แสดงภาพไดโนเสาร์ที่โปรแกรมค้นหาได้จากแฟ้มข้อมูล



รูปที่ 7.4 แสดงกราฟแท่งเปรียบเทียบการค้นหาภาพที่มีมุมมองต่างกัน ของโปรแกรมที่ค่าความเหมือนต่างๆ

จากรูปที่ 7.4 เมื่อทำการกำหนดค่าความเหมือนให้มีค่าต่างๆกัน พบว่าที่ค่าความเหมือนระดับ 70 เปอร์เซ็นต์ โปรแกรมสามารถค้นหาภาพที่มีมุมมองต่างกันได้ดีที่สุด

เนื่องจากการใช้วิธีตีกรอบเพื่อกำหนดขอบเขตของภาพที่ต้องการค้นหา ดังนั้นส่วนของพื้นหลังจะถูกนำมาคำนวณฮิสโตแกรมด้วย เมื่อวัตถุในภาพเปลี่ยนมุมมอง ระดับสีที่เป็นสีของวัตถุในภาพยังคงเป็นระดับสีเดิมและมีส่วนของพื้นหลังที่อยู่ในระดับสีเดิมด้วย การเปลี่ยนมุมมองภาพทำให้ทำให้จำนวนสีในแต่ละระดับเปลี่ยนไป จึงทำให้ค่าความเหมือนที่ได้ไม่สูงมากนัก

7.2.3 ทดลองการค้นหาภาพเมื่อภาพที่ต้องการมีขนาดของวัตถุที่สนใจในภาพต่างกัน ซึ่งเป็นความสามารถของการ Normalize ขนาดของภาพเป้าหมายและภาพโมเดล เนื่องจากวิธีฮิสโตแกรมอินเตอร์เซกชันสามารถค้นหาภาพโมเดลที่มีขนาดเล็กกว่าภาพเป้าหมายเล็กน้อยได้เท่านั้น การทำ Normalize เพื่อปรับขนาดของภาพเป้าหมายและภาพโมเดลให้เท่ากัน ทำให้โปรแกรมสามารถค้นหาภาพเดียวกันที่มีขนาดต่างกันโดยไม่ขึ้นกับขนาดของภาพเป้าหมายอีก การทดสอบจะกำหนดให้มีกลุ่มของภาพที่คล้ายกันในแฟ้มข้อมูล แต่ละกลุ่มมีภาพที่มีขนาดของวัตถุที่สนใจในภาพต่างกัน และให้โปรแกรมรับภาพเป้าหมายที่มีขนาดของวัตถุที่สนใจในภาพ ต่างกับภาพในแฟ้มข้อมูล โปรแกรมจะต้องค้นหาภาพที่คล้ายกับภาพเป้าหมายได้

<u>การทดลองที่ 3</u>

<u>จุดประสงค์</u> ทดสอบการค้นหาภาพที่มีขนาดของวัตถุที่สนใจต่างกัน

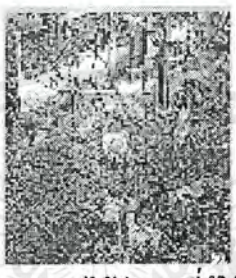
<u>ภาพเป้าหมาย</u> ภาพดอกไม้สีชมพู

กลุ่มของภาพในแฟ้มข้อมูล

1. ภาพดอกไม้สีชมพู 7 ภาพที่มีขนาดต่างกัน
2. ภาพโปเกมอน 11 ภาพที่มีขนาดต่างกัน
3. ภาพนกทวิตตี้ 2 ภาพที่มีขนาดต่างกัน

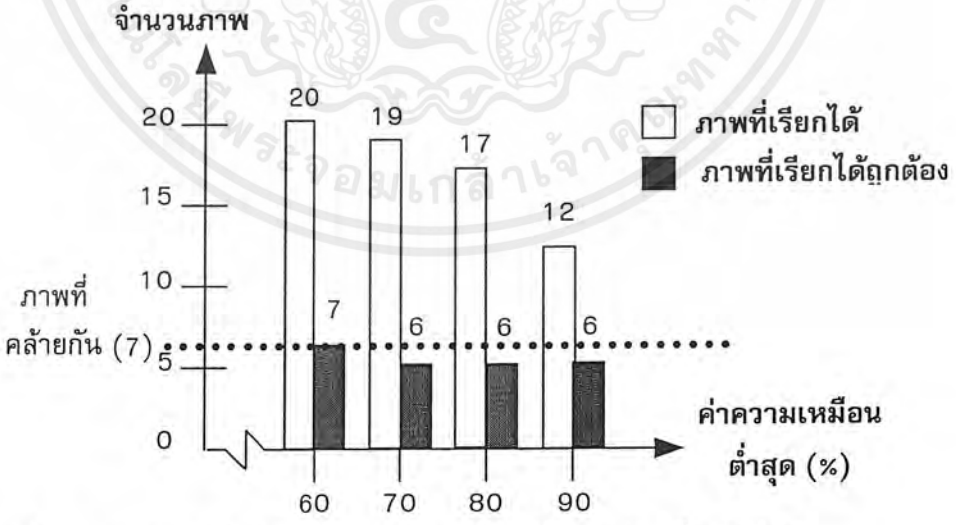
เกณฑ์ สามารถค้นหาภาพดอกไม้สีชมพูขนาดต่างๆกันได้ใกล้เคียง 7 ภาพ

ผลการทดลอง โปรแกรมสามารถหาภาพดอกไม้ที่อยู่ในแฟ้มข้อมูลและแสดงผลภาพ 4 อันดับแรกได้ถูกต้อง แสดงว่าโปรแกรมสามารถค้นหาภาพที่มีขนาดของวัตถุที่สนใจแตกต่างกันได้ รูปที่ 7.5ก เป็นภาพเป้าหมายที่รับจากผู้ใช้ และรูปที่ 7.5ข เป็นภาพที่ได้จากการแสดงผลการค้นหาในแฟ้มข้อมูล 2 ภาพ



รูปที่ 7.5ก แสดงภาพดอกไม้สีชมพูที่ผู้ใช้ต้องการค้นหา

รูปที 7.5ข แสดงภาพดอกไม้สีชมพูที่โปรแกรมค้นหาได้จากแฟ้มข้อมูล



รูปที่ 7.6 แสดงกราฟแท่งเปรียบเทียบการค้นหาภาพที่มีขนาดวัตถุที่สนใจต่างกัน ของโปรแกรมที่ค่าความเหมือนต่างๆ

จากรูปที่ 7.6 เมื่อทำการกำหนดค่าความเหมือนให้มีค่าต่างๆกัน พบว่าที่ค่าความเหมือนระดับ 0.9 หรือ 90 เปอร์เซ็นต์ โปรแกรมสามารถค้นหาภาพที่มีขนาดวัตถุที่สนใจต่างกันได้ดีที่สุด

เนื่องจากการ Normalize ที่ใช้ในโปรแกรมสามารถคำนวณได้เฉพาะภาพที่เหมือนกันที่มีขนาดต่างกันเท่านั้น ทำให้ภาพที่ไม่เกี่ยวข้องที่ผ่านการ Normalize มีจำนวนสีในแต่ละระดับเท่ากับจำนวนสีในแต่ละระดับของภาพเป้าหมาย ค่าความเหมือนที่คำนวณได้จะสูงและโปรแกรมจะเข้าใจว่าเป็นภาพที่คล้ายกัน ทำให้โปรแกรมแสดงภาพที่ไม่เกี่ยวข้องสูงมากแม้ว่าจะกำหนดให้ค่าความเหมือนมีค่าสูงแล้วก็ตาม

7.2.4 ทดลองการค้นหาภาพเมื่อภาพที่ต้องการมีสีของพื้นหลังต่างกัน ซึ่งเป็นความสามารถของวิธีฮิสโตแกรมอินเตอร์เซกชัน ที่จะแยกสีระดับสีที่ต่างกันออกไป และคำนวณเฉพาะสีที่อยู่ในระดับเดียวกันในฮิสโตแกรม ดังนั้นเมื่อภาพที่คล้ายกันแต่มีสีของพื้นหลังต่างกัน สีของพื้นหลังที่ต่างกันก็จะอยู่ในระดับของสีที่ต่างกัน และไม่ถูกนำมาคิดคำนวณค่าความเหมือน การทดสอบจะกำหนดให้มีกลุ่มของภาพที่คล้ายกันในแฟ้มข้อมูล แต่ละกลุ่มจะมีภาพที่มีสีของพื้นหลังต่างกัน และให้โปรแกรมรับภาพเป้าหมายที่มีสีของพื้นหลัง ต่างกับภาพในแฟ้มข้อมูล โปรแกรมจะต้องค้นหาภาพที่คล้ายกับภาพเป้าหมายได้

การทดลองที่ 4

จุดประสงค์ ทดสอบการค้นหาภาพที่มีพื้นหลังต่างกัน

ภาพเป้าหมาย ภาพชาร์ลี บราวน์

กลุ่มของภาพในแฟ้มข้อมูล

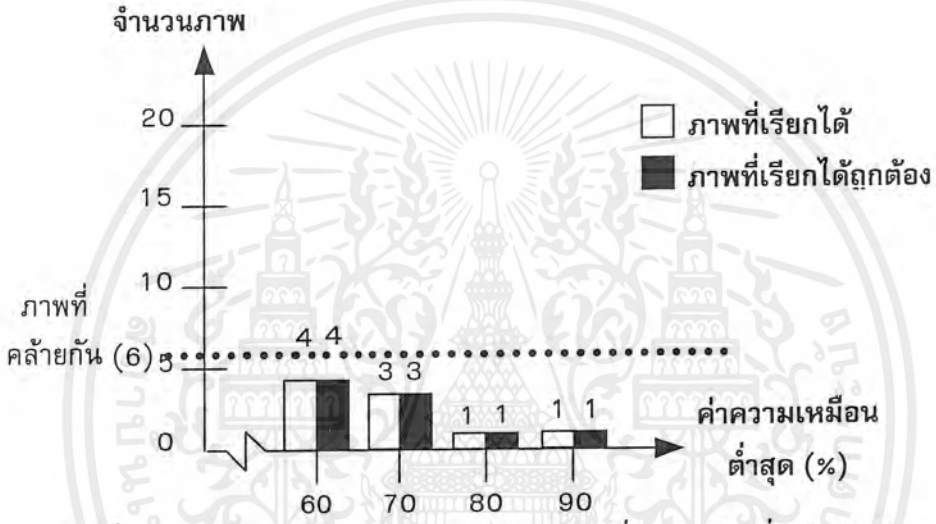
1. ภาพชาร์ลีบราวน์ 6 ภาพที่มีสีของพื้นหลังต่างกัน
2. ภาพโดราเอมอน 6 ภาพที่มีสีของพื้นหลังต่างกัน
3. ภาพนกทวิตตี้ 8 ภาพที่มีสีของพื้นหลังต่างกัน

เกณฑ์ สามารถค้นหาภาพชาร์ลี บราวน์ได้ใกล้เคียง 6 ภาพ

ผลการทดลอง โปรแกรมสามารถหาภาพชาร์ลี บราวน์ที่อยู่ในแฟ้มข้อมูลและแสดงผลภาพ 4 อันดับแรกได้ถูกต้อง แสดงว่าโปรแกรมสามารถค้นหาภาพที่มีพื้นหลังแตกต่างกันได้ รูปที่ 7.7ก เป็นภาพเป้าหมายที่รับจากผู้ใช้ และรูปที่ 7.7ข เป็นภาพที่ได้จากการแสดงผลการค้นหาในแฟ้มข้อมูล 2 ภาพ

รูปที่ 7.7ก แสดงภาพ ชาร์ลี บราวน์ ที่ผู้ใช้ต้องการค้นหา

รูปที่ 7.7ข แสดงภาพชาร์ลี บราวน์ ที่โปรแกรมค้นหาได้จากแฟ้มข้อมูล



รูปที่ 7.8 แสดงกราฟแท่งเปรียบเทียบการค้นหาภาพที่มีขนาดวัตถุที่สนใจต่างกัน ของโปรแกรมที่ค่าความเหมือนต่างๆ

รูปที่ 7.8 เมื่อทำการกำหนดค่าความเหมือนให้มีค่าต่างๆกัน พบว่าที่ค่าความเหมือนระดับ 0.6 หรือ 60 เปอร์เซ็นต์ โปรแกรมสามารถค้นหาภาพที่มีสีของพื้นหลังต่างกันได้ดีที่สุด

เนื่องจากการหาค่าความเหมือน(จากวิธีฮิสโตแกรมอินเตอร์เซกชัน)จะนำเฉพาะสีที่อยู่ในระดับเดียวกันไปคำนวณ ทำให้จำนวนสีของพื้นหลังที่ต่างกัน(จึงอยู่ต่างระดับสีกัน)ไม่ได้นำมาคำนวณด้วย(แต่ตัวหารคือจำนวนพิกเซลรวมของภาพโมเดลยังคงเท่าเดิม) ค่าความเหมือนที่ได้จึงมีค่าน้อย ในทางกลับกันภาพที่ไม่เกี่ยวข้องที่มีสีของพื้นหลังต่างกัน จะทำให้ค่าความเหมือนที่คำนวณได้มีค่าต่ำ ซึ่งเป็นเหตุให้โปรแกรมสามารถเรียกภาพที่ต้องการได้โดยไม่มีภาพที่ไม่เกี่ยวข้องปะปนมาด้วย

7.2.5 ทดลองการค้นหาภาพเมื่อภาพที่ต้องการมีกลุ่มสีของภาพคล้ายกัน ซึ่งเป็นความสามารถของการใช้สีในการค้นหา เนื่องจากการใช้สีในการค้นหาจะต้องพบภาพที่มีกลุ่มสีโดยรวมในภาพใกล้เคียงกับภาพเป้าหมาย ทั้งนี้เพื่อพิสูจน์ว่า การค้นหาภาพโดยใช้สีของภาพเป็นดัชนีสามารถค้นหาภาพที่มีสีโดยรวมของภาพใกล้เคียงกันได้จริง

การทดลองชุดที่ 5

จุดประสงค์ ทดสอบการค้นหาภาพที่มีลักษณะสีใกล้เคียงกันได้

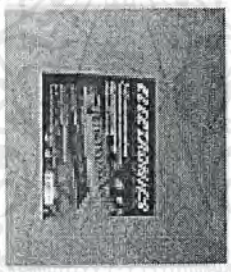
ภาพเป้าหมาย ภาพหนังสือสีส้ม

กลุ่มของภาพในแฟ้มข้อมูล

1. ภาพหนังสือสีส้ม 5 ภาพ
2. ภาพไดโนเสาร์สีม่วง 7 ภาพ
3. ภาพโดราเอมอน 4 ภาพ
4. ภาพโดรามี่ 4 ภาพ

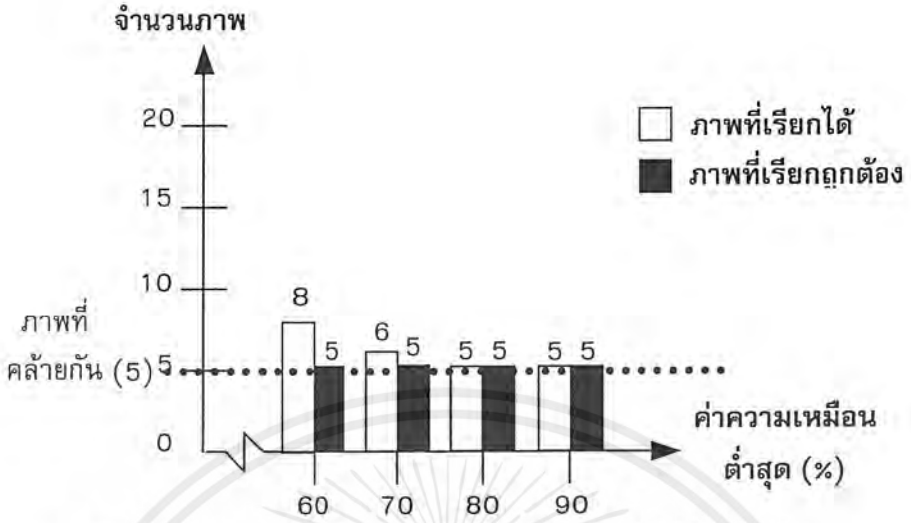
เกณฑ์ สามารถค้นหาภาพที่มีลักษณะสีใกล้เคียงกันได้ใกล้เคียง 5 ภาพ

ผลการทดลอง โปรแกรมสามารถหาภาพหนังสือที่อยู่ในกลุ่มสีส้มได้ 4 ภาพและแสดงผลภาพหนังสือสีส้ม 4 อันดับแรกได้ถูกต้อง แสดงว่าโปรแกรมสามารถค้นหาภาพที่มีสีใกล้เคียงกันได้ รูปที่ 7.9ก แสดงภาพหนังสือสีส้มที่ผู้ใช้ต้องการค้นหา และรูปที่ 7.9ข แสดงตัวอย่างภาพสองอันดับแรกที่ได้จากการค้นหาในแฟ้มข้อมูล



รูปที่ 7.9ก ภาพหนังสือสีส้มที่ผู้ใช้ต้องการค้นหา

รูปที่ 7.9ข แสดงภาพหนังสือที่โปรแกรมค้นหาได้จากแฟ้มข้อมูล



รูปที่ 7.10 แสดงกราฟแท่งเปรียบเทียบการค้นหาภาพที่มีสีใกล้เคียงกัน ของโปรแกรมที่ค่าความเหมือนต่างๆ

จากรูปที่ 7.10 เมื่อทำการกำหนดค่าความเหมือนให้มีค่าต่างๆกัน พบว่าที่ค่าความเหมือนระดับ 0.6 หรือ 60 เปอร์เซ็นต์ โปรแกรมสามารถค้นหาภาพที่มีสีใกล้เคียงกันได้ดีที่สุด

ทั้งนี้เนื่องจากโปรแกรมใช้สีของภาพมาสร้างเป็นฮิสโตแกรม ดังนั้นภาพที่มีสีใกล้เคียงกันจะต้องมีฮิสโตแกรมที่มีสีในแต่ละระดับเป็นจำนวนที่ใกล้เคียงกัน ภาพที่โปรแกรมค้นหาได้จะเป็นภาพที่มีสีที่พบได้ในภาพเป้าหมายซึ่งอาจจะเป็นภาพที่ต้องการ หรือภาพที่ไม่ต้องการซึ่งมีจำนวนรวมของสีเท่ากันกับจำนวนรวมของสีในภาพเป้าหมายก็ได้ การทดลองนี้เพื่อพิสูจน์ว่าการใช้สีเป็นดัชนีเพื่อค้นหาภาพสามารถค้นหาภาพที่ต้องการได้จริง

7.3 สรุปการทดลอง

โปรแกรมสามารถค้นหาภาพที่มีมุมมองของภาพต่างกันและภาพที่มีสีใกล้เคียงกันได้ดีที่สุด โดยจะสังเกตได้ว่า แม้ว่าจะตั้งค่าความเหมือนให้มีค่าต่ำก็ยังคงพบภาพที่ต้องการได้มากโดยที่มีภาพที่ไม่เกี่ยวข้องรวมมาด้วยเพียงเล็กน้อยเท่านั้น การค้นหาภาพที่มีความสว่างต่างกันพบว่าเมื่อกำหนดค่าความเหมือนให้ต่ำเกินไป (น้อยกว่า 60 เปอร์เซ็นต์) จะพบภาพที่ไม่เกี่ยวข้องเพิ่มขึ้นมาก ในทางกลับกัน การค้นหาภาพที่มีสีของพื้นหลังต่างกัน โปรแกรมสามารถค้นหาภาพที่คล้ายกับภาพเป้าหมายได้แต่ต้องใช้ค่าความเหมือนค่อนข้างต่ำจึงจะพบ และการค้นหาจะผิดพลาดเพิ่มขึ้น เมื่อสีของพื้นหลังของภาพปรากฏอยู่ในวัตถุในภาพที่สนใจ ส่วนการค้นหาภาพที่มีขนาดของวัตถุที่สนใจต่างกันทำได้ไม่ดีนัก เพราะยังคงมีภาพที่ไม่เกี่ยวข้องปรากฏในการค้นหา ซึ่งมีสาเหตุมาจากภาพในแฟ้มข้อมูลที่มีสีของภาพคล้ายกับภาพเป้าหมายและประสิทธิภาพของการ Normalize ที่ยังไม่ดีนัก

7.4 การคำนวณหาค่าความเหมือนต่ำสุดที่จะนำมาใช้เป็นค่าความเหมือนเริ่มต้นของโปรแกรม

ค่าความเหมือนต่ำสุด คือ ค่าความเหมือนที่น้อยที่สุดของคู่ฮิสโตแกรมที่ยอมรับว่าภาพโมเดลและภาพเป้าหมายของคู่ฮิสโตแกรมนั้นคล้ายกัน การคำนวณหาค่าความเหมือนต่ำสุดนี้ทำได้โดยการทดสอบโปรแกรมการค้นหาภาพหลายๆแบบเพื่อดูว่า ค่าความเหมือนของคู่ฮิสโตแกรมใดที่ยังคงให้ภาพโมเดลที่คล้ายกับภาพเป้าหมาย

จากการทดลองโปรแกรมให้ค้นหาภาพ สามารถสรุปผลการทดลองเพื่อนำมาหาค่าความเหมือนต่ำสุด โดยนำค่าความเหมือนที่เหมาะสมในแต่ละกรณีทดลอง มาคำนวณหาค่าเฉลี่ยค่าความเหมือนของโปรแกรม แสดงดังตารางที่ 7-1 ดังนี้

ประเภทของภาพ	ค่าความเหมือนที่เหมาะสม	จำนวนภาพที่คล้ายในแฟ้มข้อมูล	จำนวนภาพที่ถูก	จำนวนภาพที่ผิด
1. ภาพที่มีความสว่างต่างกัน	70 , 80	4	2	0
2. ภาพที่เปลี่ยนมุมมอง	70	7	7	0
3. ภาพที่มีขนาดของวัตถุที่สนใจต่างกัน	90	7	3	0
4. ภาพที่มีสีของพื้นหลังต่างกัน	60	6	4	0
5. ภาพที่มีสีใกล้เคียงกัน	80 , 90	5	5	0

ตารางที่ 7-1 แสดงผลการทดสอบโปรแกรมเพื่อหาค่าความเหมือนต่ำสุด

ค่าความเหมือนที่เหมาะสมหมายถึง ค่าความเหมือนที่ทำให้การค้นหาภาพมีประสิทธิภาพโดยรวมดีที่สุด คือสามารถค้นหาภาพที่ต้องการในแฟ้มข้อมูลได้โดยมีภาพที่ไม่ต้องการไม่มากนักในแต่ละกรณีทดลอง จำนวนภาพที่คล้ายในแฟ้มข้อมูลหมายถึง จำนวนภาพที่ต้องการทั้งหมดที่มีอยู่ในแฟ้มข้อมูล จำนวนภาพที่ถูกหมายถึง จำนวนภาพที่ต้องการที่โปรแกรมสามารถค้นหาได้ จำนวนภาพที่ผิดหมายถึง จำนวนภาพที่ไม่ต้องการที่โปรแกรมค้นหาได้

จากตารางที่ 7-1 เราจึงกำหนดให้ค่าความเหมือนต่ำสุด เป็นค่าเฉลี่ยของค่าความเหมือนที่เหมาะสม ค่าความเหมือนต่ำสุดที่ได้มีค่าเท่ากับ 77 เปอร์เซ็นต์ ซึ่งจะเป็นค่าเริ่มต้นของค่าความเหมือนของโปรแกรม ทั้งนี้ผู้ใช้สามารถปรับค่าของค่าความเหมือนของโปรแกรมนี้ ให้น้อยลงเพื่อให้โปรแกรมค้นหาภาพที่มีค่าความเหมือนได้กว้างขึ้น

7.5 การค้นหาภาพเมื่อมีแฟ้มข้อมูลเพิ่มมากขึ้น

ในความเป็นจริงภาพที่มีอยู่ในแฟ้มข้อมูลจะแตกต่างกัน ทั้งทางด้านมุมมองภาพ ความสว่างของภาพ สีของพื้นหลัง ฯลฯ ดังนั้นจึงได้ทำการทดลองโปรแกรมให้ค้นหาภาพในแฟ้มข้อมูลที่ได้จากอินเตอร์เน็ตและกล้องดิจิตอลรวมทั้งหมด 67 ภาพ และทำการค้นหาภาพต่างๆในแฟ้มข้อมูลนี้ โดยใช้ค่า

ความเหมือนต่ำสุดที่ได้จากการทดลองที่ได้กล่าวไปแล้วคือ 77 เปอร์เซ็นต์ ได้ผลการค้นหาแสดงดังตารางที่ 7-2

ตัวอย่างภาพ	จำนวนภาพที่คล้าย	จำนวนภาพที่ถูก	จำนวนภาพที่ผิด	ความสามารถในการเรียกภาพ (%)
1. ภาพเสื้อเชิ้ต	4	2	0	50
2. ภาพไดโนเสาร์	7	5	1	71
3. ภาพดอกไม้	7	7	48	100
4. ภาพชาร์ลี บราวน์	6	3	3	50
5. ภาพหนังสือสีส้ม	4	4	36	100

ตารางที่ 7-2 แสดงสรุปผลการทดลองโปรแกรมค้นหาภาพต่างๆในแฟ้มข้อมูล 67 ภาพ

จากตารางที่ 7-2 พบว่าเมื่อภาพในแฟ้มข้อมูลเพิ่มขึ้นจะทำให้โปรแกรมค้นหาภาพที่ไม่ต้องการเพิ่มมากขึ้น ดังในกรณีของการค้นหาภาพดอกไม้ และภาพหนังสือสีส้ม เนื่องจากเมื่อภาพในแฟ้มข้อมูลเพิ่มมากขึ้น โอกาสที่จะพบภาพที่มีสีใกล้เคียงกันที่ไม่ต้องการมากก็ขึ้นด้วย ทั้งนี้ความสามารถในการเรียกภาพของโปรแกรมโดยรวมยังคงดีอยู่ จะเห็นได้ว่า การค้นหาภาพบางภาพ แม้ว่าจะพบภาพที่ไม่ต้องการมากแต่ก็พบภาพที่ต้องการทั้งหมดด้วย

7.6 ปัญหาที่พบและการแก้ไขในโครงงาน

7.6.1 สีเทาใน RGB color space ไม่สามารถแปลงสีด้วยนิยามการแปลงสีได้ ทำให้การลดระดับของสีให้เป็น 166 สีนั้นทำได้ยาก เนื่องจากไม่สามารถระบุให้สีเทาอยู่ในกลุ่มใดๆ ของระดับสี จึงทำการเพิ่มระดับสีจาก 18 ระดับให้เป็น 19 ระดับ โดยระดับที่ 19 นี้ใช้บอกถึงพิกเซลที่เป็นสีเทา และทำเช่นเดียวกับการแบ่งระดับสีของสีเทา ส่วนระดับของความเข้มสีและระดับของความสว่าง ได้ทำการเพิ่มอีกอย่างละ 1 ระดับเพื่อใช้ตรวจสอบการผิดพลาดที่เกิดจากการคำนวณฮิสโตแกรม (ดังที่ได้กล่าวไปแล้วในหัวข้อวิธีการทำงานเรื่องการลดระดับของสี) ดังนั้นฮิสโตแกรมที่ใช้ในโครงงานจึงมี 32 ระดับ (19+4+4+5= 32 ระดับ)

7.6.2 ขนาดของวัตถุที่สนใจในภาพเป้าหมายมีขนาดเล็กกว่าภาพโมเดลในแฟ้มข้อมูล ทำให้วิธีฮิสโตแกรมอินเตอร์เซกชันไม่สามารถคำนวณค่าความเหมือนที่ถูกต้องได้ จึงได้แก้ไขด้วยการคำนวณให้ขนาดของภาพเป้าหมายใกล้เคียงกับขนาดของภาพโมเดลก่อนที่จะทำฮิสโตแกรมอินเตอร์เซกชัน

7.6.3 การไม่คุ้นเคยกับโปรแกรมที่ใช้ในการทำโครงงาน โครงงานนี้ใช้โปรแกรม visual C++ แม้ว่าจะมีไลบรารี่ให้เลือกใช้อยู่มาก แต่ก็เป็นยากที่จะรู้ว่าในแต่ละไลบรารี่มีฟังก์ชันอะไรให้เลือกใช้ได้บ้าง ฟังก์ชันใดเหมาะสมกับงานที่ต้องการทำ ทำให้โค้ดของโปรแกรมซับซ้อน ซึ่งได้แก้ปัญหาโดยการค้นหาฟังก์ชันจาก MSDN library visual studio 6 และทดลองใช้ฟังก์ชันหลายๆอย่าง จนพบฟังก์ชันที่เหมาะสม และมีการคำอธิบาย (comment) ส่วนของโปรแกรมออกเป็นส่วนๆ เพื่อให้เข้าใจการทำงานได้ง่าย

7.6.4 การกำหนดให้ค่าความเหมือนคำนวณจากระดับสี หรือ รวมทั้งระดับสี ความเข้มข้องสี ความสว่าง และสีเทา ฯลฯ ถ้าหากว่าให้น้ำหนักกับระดับสีมาก ก็จะทำให้โปรแกรมค้นหาภาพที่มีความสว่างต่างกันได้ หากว่าให้น้ำหนักกับทุกประเภทเท่ากันจะค้นหาภาพขนาดต่างๆกันได้ เหมาะสมกับภาพเดียวกันที่ถ่ายเวลาใกล้กัน จะให้ภาพที่มีความสว่างใกล้เคียงกัน การค้นหาค่าความเหมือนจะสูงขึ้นแต่โอกาสที่จะพบภาพที่แตกต่างกันก็จะสูงขึ้นด้วย เนื่องจากโครงงานนี้จะเน้นที่ความสามารถในการค้นหาภาพที่มีความสว่างแตกต่างกัน จึงเลือกการคำนวณหาค่าความเหมือนด้วยระดับสีเท่านั้น

บทที่ 8
สรุปและวิจารณ์

8.1 บทนำ

ในโครงงานนี้ได้ทำการศึกษาการค้นหาภาพโดยใช้สีเป็นดัชนี โดยเริ่มจากการแปลงสีและการลดระดับสี แล้วสร้างฮิสโตแกรมของสีเป็นดัชนี และค้นหาภาพด้วยวิธีฮิสโตแกรมอินเตอร์เซกชัน จากนั้นทดลองสร้างโปรแกรมค้นหาภาพและข้อมูลของภาพ บันทึกผลการทดลองเพื่อทดสอบว่าวิธีการค้นหาภาพโดยใช้ฮิสโตแกรมของสีเป็นดัชนีสามารถค้นหาภาพได้ในระดับใด พบปัญหาและมีข้อจำกัดใดบ้าง สามารถทำการแก้ไขได้อย่างไร และแนวทางในการพัฒนาเพื่อให้การค้นหาภาพและข้อมูลของภาพมีประสิทธิภาพเพิ่มขึ้น ทั้งนี้เพื่อให้ผู้ที่จะมาพัฒนาต่อสามารถทำความเข้าใจพื้นฐานหลักการของ color histogram การค้นหาภาพด้วยวิธีฮิสโตแกรมอินเตอร์เซกชันและสามารถหาแนวทางเพื่อนำไปพัฒนาต่อไปได้

8.2 สรุปและวิจารณ์

โครงงานนี้ ได้ทำการศึกษาการกำหนดดัชนีโดยใช้สีของภาพ และพัฒนาโปรแกรมตามที่ศึกษาไว้ โดยเริ่มจากการแปลงสีและการลดระดับสี เพื่อแยกค่าสีออกจากความสว่างในภาพและลดจำนวนค่าสีให้น้อยลงเพื่อสร้างเป็นฮิสโตแกรมของสีและกำหนดให้ฮิสโตแกรมของสีเป็นดัชนี จากนั้นพัฒนาให้โปรแกรมค้นหาภาพโดยวิธีฮิสโตแกรมอินเตอร์เซกชัน จนได้โปรแกรมที่สามารถค้นหาภาพและข้อมูลของภาพที่ต้องการได้ แล้วทำการทดลองโปรแกรมให้ค้นหาภาพในแต่ละกรณีต่างๆดังต่อไปนี้คือ ภาพที่มีความสว่างต่างกัน ภาพที่มีมุมมองภาพต่างกัน ภาพที่มีขนาดของวัตถุที่สนใจต่างกัน ภาพที่มีสีของพื้นหลังภาพต่างกันและภาพที่มีสีใกล้เคียงกัน จากผลการทดลองปรากฏว่าโปรแกรมสามารถค้นหาภาพที่ต้องการในแฟ้มข้อมูลได้ตามทฤษฎีของ color histogram และฮิสโตแกรมอินเตอร์เซกชัน โดยโปรแกรมสามารถค้นหาภาพที่ต้องการได้เกือบทั้งหมด เมื่อกำหนดให้ค่าความเหมือนต่ำสุดมากกว่า 60 การค้นหาภาพบางภาพอาจพบภาพที่ไม่เกี่ยวข้องมาก ซึ่งมีสาเหตุมาจากภาพที่ไม่เกี่ยวข้องส่วนใหญ่ในแฟ้มข้อมูลมีการกระจายของสีมาก แต่มีจำนวนพิกเซลรวมของสีในแต่ละระดับใกล้เคียงกับจำนวนพิกเซลรวมของสีในแต่ละระดับของภาพเป้าหมาย ดังนั้นโปรแกรมนี้จึงเหมาะสมกับแฟ้มข้อมูลที่สามารถแยกกลุ่มของภาพด้วยสีได้ เช่นกลุ่มของดอกไม้สีชมพู กลุ่มของดอกไม้สีส้ม กลุ่มของดอกไม้สีม่วง เป็นต้น และหากพัฒนาโปรแกรมให้สามารถค้นหาภาพได้นอกเหนือจากการใช้สีเป็นดัชนีก็จะทำให้ พบภาพที่ไม่เกี่ยวข้องน้อยลงด้วย

ในโครงงานนี้ยังคงมีข้อจำกัดอยู่ที่

1. ขนาดของแฟ้มข้อมูล แม้ว่าจะทำการลดระดับสีของภาพเพื่อให้จำนวนครั้งในการเปรียบเทียบในแต่ละระดับลดลงแล้ว หากแฟ้มข้อมูลมีขนาดใหญ่ขึ้นมาก จะทำให้ประสิทธิภาพในการค้นหาน้อยลง

และทำให้เวลาในการค้นหาเพิ่มมากขึ้น โดยเฉพาะถ้าฮิสโตแกรมมีขนาดใหญ่ เพราะโปรแกรมจะต้องทำการเปรียบเทียบไฟล์ฮิสโตแกรมในแฟ้มข้อมูลทั้งหมด ทั้งนี้อาจหาอัลกอริทึมในการค้นหาข้อมูลที่เหมาะสมมาพัฒนาต่อเพื่อเพิ่มความเร็วในการค้นหาได้

2. ปัญหาของภาพโมเดลในแฟ้มข้อมูลไม่เท่ากับภาพเป้าหมาย ถ้าหากว่าภาพเป้าหมายมีขนาดใหญ่กว่าภาพโมเดลมากจะทำให้ฮิสโตแกรมของภาพเป้าหมายครอบคลุมฮิสโตแกรมของภาพโมเดลซึ่งมีขนาดเล็กทั้งหมด ทำให้ค่าความเหมือนสูงและเกิดความผิดพลาดในการค้นหา แต่ถ้าหากว่าภาพเป้าหมายมีขนาดเล็กกว่าภาพโมเดลมากจะให้ค่าความเหมือนน้อยลง และทำให้ภาพอื่นๆ มีค่าความเหมือนมากกว่าโปรแกรมก็จะค้นหาผิดพลาด แม้ว่าจะทำการแปลงให้จำนวนพิกเซลรวมของภาพเป้าหมายมีจำนวนเท่ากับพิกเซลรวมของภาพโมเดลแล้วก็ตาม แต่กลับทำให้ภาพที่ต่างกันที่มีขนาดเล็ก ซึ่งแต่เดิมมีจำนวนของสีน้อยอยู่แล้วกลับเพิ่มมากขึ้นจนทำให้จำนวนพิกเซลของสีต่างๆเท่ากันกับจำนวนสีในแต่ละระดับของฮิสโตแกรมของภาพโมเดล ทำให้ค่าความเหมือนสูงขึ้นและมีภาพที่ไม่เกี่ยวข้องปรากฏในการค้นหาด้วย

3. แม้ว่าจะทำการคำนวณขนาดของภาพโมเดลให้ใกล้เคียงกับภาพเป้าหมายแล้วก่อนทำฮิสโตแกรมอินเตอร์เซกชัน แต่ยังคงมีภาพที่ไม่เกี่ยวข้องแสดงผลออกมาด้วย เนื่องมาจากการใช้แค่สีของภาพในการค้นหา รูปที่ต่างกันหากมีขนาดของภาพเท่ากันและมีกลุ่มสีและจำนวนของระดับสีที่ใกล้เคียงกันมาก โปรแกรมก็จะเข้าใจว่าเป็นรูปที่ใกล้เคียงกัน

ถึงแม้ว่าโครงงานนี้จะสามารถทำการค้นหาภาพได้ตามทฤษฎีของ color histogram แต่ก็ยังมีประสิทธิภาพไม่ดีนัก ทั้งนี้ขึ้นอยู่กับข้อจำกัดของตัว color histogram เองซึ่งไม่สามารถทำค้นหาภาพเช่น ตำแหน่งของกลุ่มสีที่ปรากฏในภาพได้ ทำให้การค้นหาภาพยังคงพบภาพที่ไม่เกี่ยวข้องมาก อย่างไรก็ดีวิธี color histogram ก็ยังสามารถค้นหาภาพที่มีกลุ่มสีใกล้เคียงกับสีของภาพเป้าหมาย ทำให้การค้นหาภาพกว้างขึ้นและมีโอกาสให้ผู้ใช้เลือกภาพที่ต้องการได้มากขึ้นด้วยทั้งยังทำความเข้าใจและทำได้ง่าย วิธี color histogram สามารถประยุกต์ใช้งานร่วมกับวิธีอื่นได้ง่าย เมื่อพัฒนาวิธีอื่นๆ ร่วมกับวิธี color histogram ก็สามารถลดภาพที่ไม่เกี่ยวข้องได้ทำให้โปรแกรมสามารถค้นหาภาพที่เกี่ยวข้องได้เพิ่มขึ้น ดังนั้นวิธี color histogram นับเป็นจุดเริ่มต้นที่ดีของการพัฒนาโปรแกรมการค้นหาภาพและข้อมูลของภาพ

8.3 แนวทางในการพัฒนาต่อ

ดังที่ได้กล่าวมาแล้วว่า วิธีที่ใช้สีอย่างเดียวเป็นดัชนียังคงมีข้อจำกัดอยู่มาก แต่ก็เป็นคุณลักษณะหนึ่งที่สำคัญอย่างมาก นักวิจัยหลายท่านได้ทำการทดลองวิธีต่างๆที่ใช้สีเป็นดัชนี รวมกับการเอาคุณลักษณะอื่นๆ เช่น โครงร่าง (texture) ในภาพและรูปร่าง (shape) ร่วมในการคำนวณกับวิธี color histgram เพื่อให้การค้นหาภาพมีประสิทธิภาพมากขึ้น ซึ่งเรียกวิธีการรวมเอาคุณลักษณะต่างๆนี้ว่าวิธีฮิสโตแกรมผสม (Joint histogram) ในส่วนของโครงงานนี้ สามารถพัฒนาให้ประสิทธิภาพดีขึ้นได้โดยการทำวิธี ฮิสโตแกรมผสมก็ได้ หรือใช้โครงร่างของภาพ ซึ่งวิธีนี้จะกำหนดโครงร่างของภาพโดยอัตโนมัติ ทำให้ความผิดพลาดที่เกิดจากพื้นหลังน้อยลง เช่นเดียวกับการหาขอบของรูป (edge detection) ซึ่งจะค้นหาวัตถุที่สนใจในภาพได้อัตโนมัติ ทั้งยังสามารถพัฒนาฮิสโตแกรมอินเตอร์เซกชันให้เป็นฮิสโตแกรมแบคโปรเจ็คชัน (histogram back-projection)ได้ ฮิสโตแกรมแบคโปรเจ็คชันสามารถระบุตำแหน่งของสีในภาพได้ แม้ว่า

ภาพจะมีขนาดต่างกัน ซึ่งการนำวิธีต่างๆเหล่านี้มาพัฒนาร่วมกับตัวโครงงานชิ้นนี้ จะทำให้การค้นหาภาพด้วยสีถูกต้องมากยิ่งขึ้น

8.4 แนวทางในการประยุกต์ใช้งาน

เนื่องจากในปัจจุบันมีภาพดิจิตอลจำนวนมากซึ่งอยู่ในฐานข้อมูลขนาดใหญ่เช่น เวิร์ล ไวด์ เวบ, vedio-on demand หรือการเก็บภาพเพื่อใช้ในงานเฉพาะทางต่างๆ เช่นอัลบั้มภาพดิจิตอล รวมไปถึงงานทางโรบอทวิชั่น เพื่อให้โรบอทจดจำภาพเพื่อใช้ในการทำงานในโรงงาน การค้นหาภาพจึงจำเป็นที่จะต้องพัฒนาเพื่อให้มีความสามารถเหมาะสมกับงานที่ต้องการ ในโครงงานที่ได้จัดทำนี้สามารถนำไปใช้ในการค้นหาภาพในอัลบั้มภาพดิจิตอล ใช้ค้นหาข้อมูลของภาพในเวิร์ล ไวด์ เวบ รวมถึงโรบอทวิชั่นได้ หากได้พัฒนาตามแนวทางในการพัฒนาต่อที่ได้กล่าวไว้ในหัวข้อที่ 8.3

ภาคผนวก

ภาคผนวก ก.

Color Space

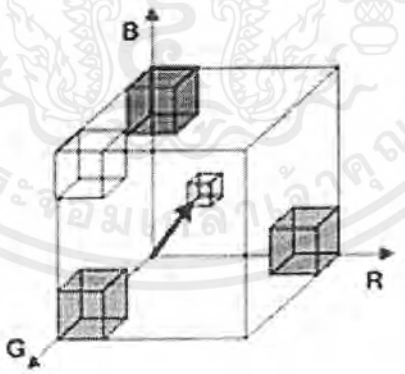
Color space คือการแสดงพิกเซลของสีในมิติ (dimension) ต่างๆ โดยหนึ่งพิกเซลจะประกอบด้วย 3 มิติ ซึ่งจะมีหลักการในการกำหนด color space ที่ต่างกัน เช่น

1. กำหนดตามแม่แสงของสี คือ สีแดง สีเขียว และสีน้ำเงิน (RGB) ซึ่งจะเหมาะสมกับการแสดงภาพบนหน้าจอคอมพิวเตอร์ซึ่งใช้การผสมกันของแสงสีต่างๆ ในการแสดงภาพ
2. กำหนดตามลักษณะการมองเห็นได้แก่ สี ความเข้มของสี ความเข้มของแสง (HSV) ซึ่งจะเหมาะสมกับการแยกแยะวัตถุตามสีที่ปรากฏ

ดังนั้นการเลือกใช้ color space จะขึ้นอยู่กับการนำไปใช้งานของเรา ว่างานประเภทนั้นเหมาะสมกับ color space แบบใด

RGB Color space

การกำหนดภาพสีในรูปแบบของ RGB จะเป็นรูปแบบพื้นฐานของภาพที่มีการแสดงบนจอคอมพิวเตอร์โดยใช้ 1 พิกเซลของภาพจะประกอบด้วยสี 3 ค่า คือสีแดง, สีเขียว และสีน้ำเงิน สีละ n บิต สี 1 ค่าของสีจะสามารถแสดงเฉดสีได้ทั้งหมด 2^n เฉดสีที่แตกต่างกัน รูปที่ 1ก. แสดงความสัมพันธ์ของสีแดง สีเขียว และสีน้ำเงินต่อหนึ่งพิกเซล

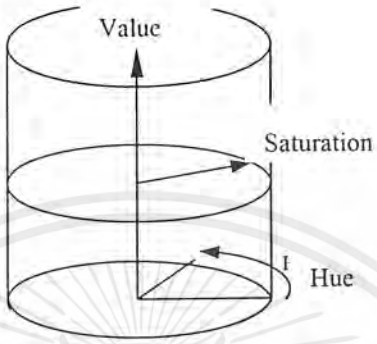


รูปที่ 1ก. ความสัมพันธ์ของสีใน RGB Color space

ในระบบนี้แกน R จะแสดงปริมาณของสีแดงที่มีในพิกเซลนั้น แกน G จะแสดงปริมาณของสีเขียวที่มีในพิกเซล นั้นและแกน B จะแสดงปริมาณของสีน้ำเงินที่มีในพิกเซลนั้น เนื่องจากมีการเก็บสีแยกเป็นคนละค่าของสีกันจึงทำให้เมื่อมีการคำนวณแยกทีละค่าของสีแล้วจะทำให้เกิดการ color shift สีของภาพที่ได้ก็จะเกิดการเพี้ยนไป

HSV Color space

การกำหนดภาพสีในรูปแบบของ HSV (hue, saturation, value) โดยความสัมพันธ์ที่แสดงโดยใช้ color space นี้จะเป็นเหมือนกับหลักการของจิตรกรคือการใช้ สี เงา และความเข้มของสี ดังรูปที่ 2ก. แสดงความสัมพันธ์ของระดับสี, ความเข้มของสี และ ความสว่าง



รูปที่ 2ก. ความสัมพันธ์ของ hue, saturation และ value บน HSV color space

จากรูปค่าในแนวแกนตั้งของรูปทรงกระบอกจะแสดงระดับของความสว่าง ระยะห่างจากแกนกลางจะแสดงถึงระดับของความเข้มของสีและองศาที่แสดงในแนวระนาบจะเป็นระดับของระดับสี ที่ปรากฏใน 1 พิกเซล

ในระบบนี้ระดับสี (hue) คือสีที่อธิบายโดยใช้ความยาวคลื่นของสี ยกตัวอย่างเช่น ความแตกต่างระหว่างสีเหลืองและสีแดง ความเข้มของสี (saturation) เป็นปริมาณของสีที่ปรากฏ ยกตัวอย่างเช่น ความแตกต่างระหว่างสีแดงและสีชมพู แกนที่สามเรียกว่า ความสว่าง (lightness) ความเข้ม (intensity) หรือ ค่าของสี (value) เป็นปริมาณของแสง เช่นความแตกต่างระหว่างสีแดงเข้ม (dark red) และสีแดงสว่าง (light red) หรือสีเทาเข้ม (dark gray) และสีเทาอ่อน (light gray)

ระบบสี HSV มีความเหมาะสมในการนำมาใช้ในโครงงานนี้ เนื่องจากในการแสดงสีที่ปรากฏในภาพเมื่อเราใช้ระดับสี ในการแสดงสีจะเป็นการตอบสนองต่อการมองเห็นของมนุษย์โดยไม่มีผลกระทบของแสงเงา และการทำ image processing โดยใช้อัลกอริทึม spatial smoothing หรือ median filtering ในการลดการรบกวน (noise) ต่างๆ ในรูปภาพที่อยู่ในรูปแบบของ RGB โดยแยกทำทีละค่าของสี จะมีผลทำให้เกิดการผิดเพี้ยนของสี แต่ถ้าใช้อัลกอริทึมเหล่านี้กับรูปภาพที่เป็น HSV จะไม่เกิดการผิดเพี้ยนของสี แต่เนื่องจากการทำงานอุปกรณ์แสดงภาพส่วนมากมักจะสนับสนุนการแสดงภาพในรูปแบบของ RGB เราจึงต้องใช้การคำนวณในการแปลงสีภาพที่อยู่ในรูปแบบของ RGB ให้มาอยู่ในรูปแบบของ HSV และเมื่อเราทำงานกับภาพนั้นเสร็จแล้วจะต้องทำการเปลี่ยนกลับให้มาอยู่ในรูปแบบของ RGB ดังเดิมเพื่อที่จะได้สามารถแสดงภาพบนหน้าจอได้

Normalization

เราสามารถทำการ normalize สีใน HSV color space ซึ่งประกอบไปด้วยสีหลัก คือ สีแดง สีเขียว และสีน้ำเงิน ในรูปของสมการที่ 1-3

$$r = \frac{R}{R+G+B} \qquad (1)$$

$$g = \frac{G}{R+G+B} \qquad (2)$$

$$b = \frac{B}{R+G+B} \qquad (3)$$

โดยเราจะสมมติว่า R,G,B เหล่านี้ถูก normalize ให้เป็น r,g,b ที่มีค่าอยู่ระหว่าง [0...1]

ภาคผนวก ข.

DIB Format

Device-independent bitmaps (DIBs) เป็นภาพกราฟฟิกที่สามารถนำไปแสดงบนอุปกรณ์แสดงผลที่ต่างกันได้ เนื่องจากจะมีตารางสีสำหรับใช้ในการแสดงผลเก็บไว้ ซึ่งไฟล์ประเภทบิทแมพก็เป็น DIBs เช่นกัน ภาพ DIB นี้จะประกอบด้วยโครงสร้างหลายประเภทดังต่อไปนี้

โครงสร้างของ BITMAPFILEHEADER

จะอยู่ในส่วนต้นของไฟล์ซึ่งมีการกำหนดโดย Windows ดังนี้

```
typedef struct tagBITMAPFILEHEADER
{       WORD bfType;
        DWORD bfSize;
        WORD bfReserved1;
        WORD bfReserved2;
        DWORD bfOffBits;
} BITMAPFILEHEADER;
```

สามารถอธิบายสมาชิกต่างๆ ของโครงสร้างนี้ได้ดังตารางที่ 1ข.

ตัวแปร	คำอธิบาย
BfType	ระบุประเภทของไฟล์โดยมีค่าเป็น ASCII คือ BM (4D42 ฐานสิบหก)
BfSize	ขนาดของไฟล์ในหน่วยไบต์
BfReserve1	มีค่าเป็น 0 เสมอ
BfReserve2	มีค่าเป็น 0 เสมอ
BfOffBits	จำนวนไบต์ตั้งแต่เริ่มต้นไฟล์จนถึง bitmap data

ตารางที่ 1ข. แสดงสมาชิกต่างๆ ของ BITMAPFILEHEADER

โครงสร้างของ BITMAPINFO

เป็นส่วนที่อยู่ถัดจาก BITMAPINFOHEADER ซึ่งมีการกำหนดโดย Windows ดังนี้

```
typedef struct tagBITMAPINFO
{
        BITMAPINFOHEADER bmiHeader;
        RGBQUAD bmiColors[1];
} BITMAPINFO;
```

โครงสร้างของ BITMAPINFOHEADER

จะมีการระบุโดย Windows เช่นกัน โดยมีโครงสร้างดังนี้

```
typedef struct tagBITMAPINFOHEADER
{
        DWORD biSize;
        DWORD biWidth;
        DWORD biHeight;
        WORD biPlanes;
        WORD biBitCount;
        DWORD biCompression;
        DWORD biSizeImage;
        DWORD biXPelsPerMeter;
        DWORD biYPelsPerMeter;
        DWORD biClrUsed;
        DWORD biClrImportant;
} BITMAPINFOHEADER;
```

สามารถอธิบายสมาชิกต่างๆ ของโครงสร้างนี้ได้ดังตารางที่ 2ข.

ตัวแปร	คำอธิบาย
biSize	เก็บขนาดของโครงสร้าง BITMAPINFOHEADER ซึ่งจะมีขนาดเท่ากับ 40 ไบต์
biWidth	ความกว้างของภาพในหน่วยพิกเซล
biHeight	ความยาวของภาพในหน่วยพิกเซล
biPlanes	มีค่าเป็น 1 เสมอ
biBitCount	จำนวนบิทต่อพิกเซลของภาพ
biCompression	ประเภทของการบีบอัดข้อมูลที่ใช้ในภาพ 0 = ไม่มีการบีบอัดข้อมูล, 1 = RLE-8 compression, 2 = RLE-4 compression
biSizeImage	ขนาดของภาพในหน่วยไบต์ จะใช้เมื่อภาพมีการบีบอัดข้อมูลเท่านั้น
biXPelsPerMeter	จำนวนพิกเซลในแนวนอนต่อเมตรของอุปกรณ์แสดงภาพ มักจะมีค่าเป็น 0 เสมอ
biYPelsPerMeter	จำนวนพิกเซลในแนวตั้งต่อเมตรของอุปกรณ์แสดงภาพ มักจะมีค่าเป็น 0 เสมอ
biClrUsed	จำนวนสีทั้งหมดในภาพ
biClrImportant	จำนวนของสีที่สำคัญ มักจะมีค่าเป็น 0 เสมอ

ตารางที่2ข. แสดงสมาชิกต่างๆ ของ BITMAPINFOHEADER

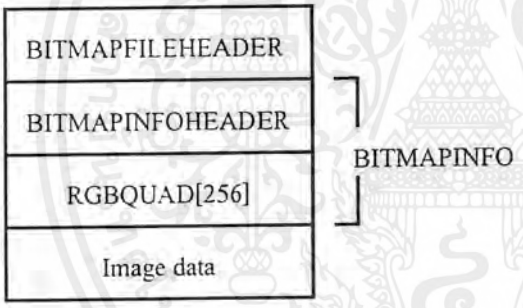
โครงสร้างของ RGBQUAD

RGBQUAD เป็นข้อมูลสุดท้ายที่ถูกระบุโดย Windows โดยมีโครงสร้างดังนี้

```
typedef struct tagRGBQUAD
{
    BYTE rgbBlue;
    BYTE rgbGreen;
    BYTE rgbRed;
    BYTE rgbReserved;
} RGBQUAD;
```

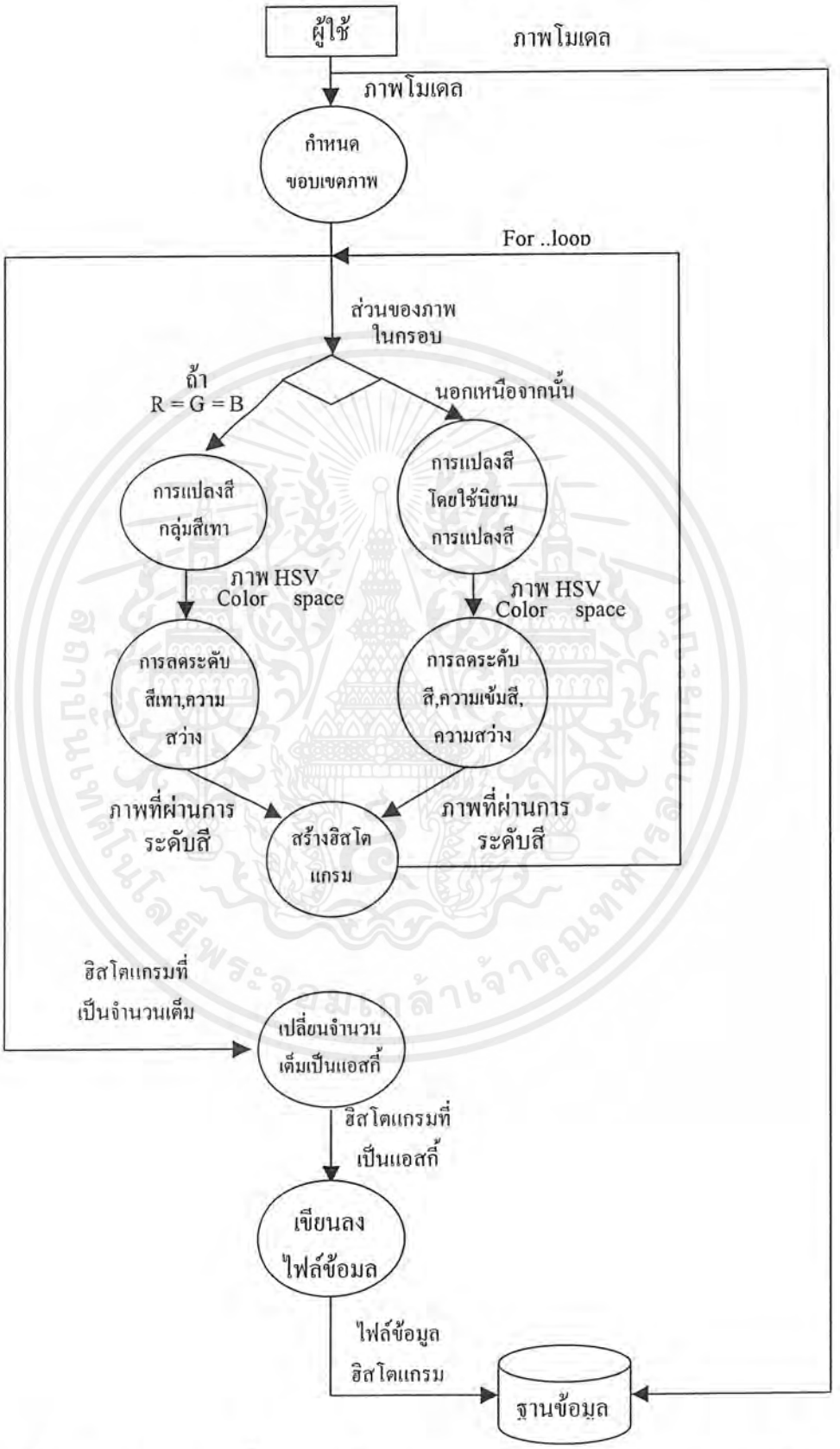
โครงสร้างนี้จะระบุถึงความเข้มของสีแดง,เขียวและน้ำเงิน แต่ละสีในภาพจะแสดงโดยโครงสร้างของ RGBQUAD นั่นคือ ภาพ 16 สี (4 บิท) จะมีตารางสีซึ่งประกอบด้วย16 RGBQUAD structures ในขณะที่ภาพ 256 สี (8 บิท) จะมีตารางสีซึ่งประกอบด้วย 256 RGBQUAD structures ภาพที่นอกเหนือจาก 24 บิท ขึ้นไปจะไม่มีตารางสี

สรุป layout ของ DIB file

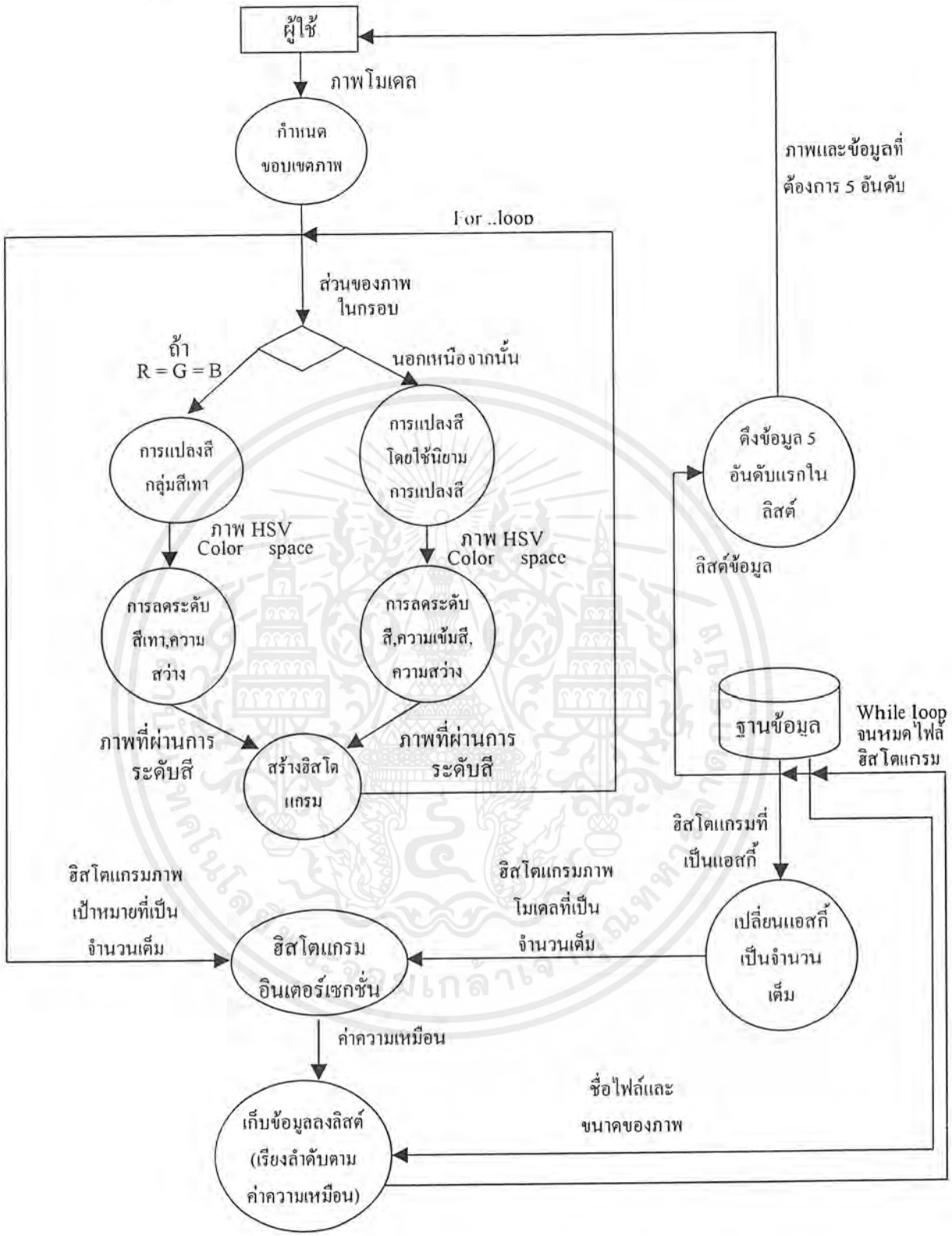


ภาคผนวก ค.

Flow Chart



รูปที่ 1ค Flow chart แสดงขั้นตอนการทำงานของการเก็บภาพและฮิสโตแกรมลงฐานข้อมูล



รูปที่ 2ค Flow chart แสดงขั้นตอนการทำงานของการค้นหาและเรียงภาพนิ่งจากแฟ้มข้อมูล

ภาคผนวก ง
การทำงานของโปรแกรมในส่วนของการติดต่อแฟ้มข้อมูล

1. การกำหนดตัวแปรที่ใช้ในการเก็บ path

คำอธิบาย

กำหนด global variable เพื่อใช้เก็บ path ของโปรแกรมเมื่อเริ่มการทำงาน

```
// DibLook.cpp
CDibLookApp NEAR theApp;
CString path;
```

โดยตัวแปรนี้จะมีการเก็บค่า path เมื่อโปรแกรมทำการเรียกฟังก์ชัน InitInstance ของ Class DibLookApp เพียงครั้งเดียวเท่านั้น เพื่อที่เราจะได้เก็บ path ของ application ไว้ใช้ในการกำหนด path ในการเรียกและเก็บภาพในแฟ้มข้อมูลได้

2. การกำหนด path ที่ใช้ในการอ่านและบันทึกไฟล์

คำอธิบาย (เป็นส่วนหนึ่งของฟังก์ชัน InitInstace( ))

```
// DibLook.cpp
BOOL CDibLookApp::InitInstance()
{
        ...
        char p[MAX_PATH];
        ::GetCurrentDirectory(MAX_PATH,p);
        path = p;
        return TRUE:
        ...
}
```

ตัวแปร p จะเป็นอะเรย์ของ character ซึ่งมีขนาดของอะเรย์เท่ากับจำนวนตัวอักษรของ path สูงสุดที่เก็บได้ โดยเมื่อเรียกฟังก์ชัน GetCurrentDirectory( ) แล้วจะได้ path ของ application ซึ่งกำลังทำงานนั้นเก็บไว้อยู่ในตัวแปร p และเราจะเก็บ path ที่ได้นี้ไว้ในตัวแปร path เพื่อที่จะสามารถนำมาใช้งานได้ต่อไป

3. การเรียกเก็บไฟล์ลงในแฟ้มข้อมูล

คำอธิบาย (เป็นส่วนหนึ่งของฟังก์ชัน Rgbvalue( ))

```
void CDibView::Rgbvalue()
```

```
{
    ...
    if (m_database)
    {
        // get save path from application
        extern    CString path;
        CString name = path;
        name.Replace("\\","/");
        name += "/hisDB/";

        // get picture name for save
        name += pDoc->GetTitle();

        // save picture and histogram and return file's buffer
        ::SaveFile(hDIB,name,target);
    }
    ...
}
```

การเก็บไฟล์ลงในแฟ้มข้อมูลจะทำเมื่อเช็คค่าของตัวแปร m_database แล้วเป็นจริง โดยในการกำหนด path ของแฟ้มที่ใช้เก็บข้อมูลจะทำโดย extern ค่าของตัวแปร path ซึ่งเป็นตัวแปรแบบโกลบอลเข้ามาและเก็บค่าของ path ไว้ในตัวแปร name ซึ่งเราจะต้องแทนที่เครื่องหมาย "\" ในค่าของ path "/" (ในโค้ดต้องใช้เครื่องหมาย "//" เนื่องจากถ้าใช้ "/" จะหมายถึงการขึ้นบรรทัดใหม่ (new line)) เพราะคำสั่งที่เราใช้ในการ save file จะต้องแสดง path ของไฟล์โดยใช้เครื่องหมาย "/" จากนั้นเราจะนำมารวมกับ "/hisDB/" และชื่อของของไฟล์ที่จะนำไปเก็บโดยชื่อไฟล์นั้นได้มาโดยการเรียกฟังก์ชัน GetTitle( ) ซึ่งเป็นฟังก์ชันของคลาส document ที่จะคืนค่าของชื่อไฟล์ปัจจุบันที่เปิดอยู่กลับมา เมื่อได้ชื่อไฟล์พร้อมทั้ง path ของไฟล์ครบแล้วจึงเรียกฟังก์ชัน SaveFile( ) เพื่อเก็บไฟล์ลงในแฟ้มข้อมูล

4. การจัดการเก็บไฟล์ลงแฟ้มข้อมูล และการสร้างไฟล์ข้อมูลฮิสโตแกรม

ชื่อฟังก์ชัน SaveFile( )

คำอธิบาย

การเรียกฟังก์ชันนี้จะต้องส่งตัวแปร hDIB ซึ่งเก็บ handle ของภาพ, ตัวแปร name ซึ่งเก็บชื่อไฟล์และ path ของไฟล์ พร้อมกับตัวแปร target ซึ่งเก็บค่าฮิสโตแกรมของภาพมา

```
//DibOper.cpp
void SaveFile(HDIB hDIB, CString name, CString target)
```

ก่อนที่จะเก็บไฟล์นั้นต้องเช็คก่อนว่าตัวแปร name ที่ส่งมาได้รวมประเภทของไฟล์ไว้ด้วยหรือไม่ โดยใช้ฟังก์ชัน find( ) ซึ่งจะคืนค่ากลับมาเป็น 0 ถ้าสามารถหาคำที่กำหนดให้เจอ และเป็น –1 เมื่อไม่สามารถหาคำที่กำหนดได้ ดังนั้นหากตัวแปร name ยังไม่รวมสกุลของภาพก็ให้เพิ่มประเภทของภาพให้เป็น ".bmp" ลงไป

```
{
    // if name not include filetype add it
        if (name.Find(".bmp",0) == -1)
            name += ".bmp";
```

เมื่อกำหนดไฟล์เสร็จแล้วจึงสร้างตัวแปร fileBMP เป็นตัวแปรในคลาสไฟล์เพื่อใช้ในการสร้างและบันทึกไฟล์ภาพโดยใช้ modeCreate และ modeWrite เป็นการสั่งให้สร้างไฟล์และสามารถบันทึกลงไปได้

```
// create fileSave to save picture
CFile fileBMP(name,CFile::modeCreate|CFile::modeWrite);
```

การบันทึกภาพลงในแฟ้มข้อมูลจะทำโดยเรียกฟังก์ชัน SaveDIB( ) และเมื่อทำงานกับไฟล์นั้นเสร็จแล้วต้องใช้ฟังก์ชัน Close( ) เพื่อปิดไฟล์

```
// save picture to DB (hisDB folder)
::SaveDIB(hDIB, fileBMP);
fileBMP.Close();
```

ต่อมาในการบันทึกฮิสโตแกรมจะใช้ชื่อเดียวกับชื่อของไฟล์ภาพ จึงใช้ฟังก์ชัน Replace( ) เปลี่ยนสกุลให้เป็นประเภท ".his" จากนั้นจึงสร้างตัวแปร fileHis ซึ่งเป็นตัวแปรในคลาสไฟล์เพื่อทำการสร้างและบันทึกไฟล์ฮิสโตแกรมโดยใช้ modeCreate และ modeWrite เช่นเดียวกัน

```
// change file type to save histogram
name.Replace(".bmp",".his");
// create file to write hisFile
CFile fileHis(name,CFile::modeCreate|CFile::modeWrite);
```

การบันทึกไฟล์ของฮิสโตแกรมจะต้องเก็บค่าของฮิสโตแกรมที่จะเขียนลงไปในไฟล์ลงในตัวแปร fileBuff ซึ่งจะทำหน้าที่เหมือนกับเป็นบัฟเฟอร์เก็บข้อมูลก่อน จึงจะนำไฟล์นั้นบันทึกลงไปโดยใช้ฟังก์ชัน Write( ) ซึ่งส่งพอยน์เตอร์ชี้บัฟเฟอร์ที่เก็บข้อมูลและความยาวของข้อมูลที่ทำการเขียนโดยหาความยาวได้โดยใช้ฟังก์ชัน GetLength( ) ซึ่งจะคืนค่าความยาวของบัฟเฟอร์กลับมา เมื่อเขียนเสร็จก็ใช้ฟังก์ชัน Close( ) ปิดไฟล์ที่ทำการเขียนนั้น

```
// copy data to buffer
CString fileBuff = target;
// write from buffer to file
fileHis.Write((LPCSTR)fileBuff,fileBuff.GetLength());
fileHis.Close();
```

ภาคผนวก จ
การทำงานของโปรแกรมในส่วนของการจัดการภาพ

1. การเปิดไฟล์ภาพ

ชื่อฟังก์ชัน OnOpenDocument( )

คำอธิบาย

ฟังก์ชันที่ใช้ในการเปิดไฟล์เขียนโดยสืบมาจากฟังก์ชันเดิม (override function) ซึ่งจะส่งค่าพอยน์เตอร์ที่ชี้ไปยังไฟล์นั้นกลับมา

```
// DibDoc.h
virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);

// DibDoc.cpp
CDibDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
        ...
        BeginWaitCursor();
        // replace calls to Serialize with ReadDIBFile function
        TRY
        {
                m_hDIB = ::ReadDIBFile(file);
        }
        CATCH (CFileException, eLoad)
        {
                file.Abort(); // will not throw an exception
                EndWaitCursor();
                ReportSaveLoadException(lpszPathName, eLoad,
                        FALSE, AFX_IDP_FAILED_TO_OPEN_DOC);
                m_hDIB = NULL;
                return FALSE;
        }
        END_CATCH

        InitDIBData():
```

```
    EndWaitCursor();
    if (m_hDIB == NULL)
    {
        // may not be DIB format
        CString strMsg;
        strMsg.LoadString(IDS_CANNOT_LOAD_DIB);
        MessageBox(NULL, strMsg, NULL, MB_ICONINFORMATION | MB_OK);
        return FALSE;
    }

    SetPathName(lpszPathName);
    return TRUE;
}
```

ถ้าการอ่านไฟล์สำเร็จ (การทำงานใน TRY) ฟังก์ชันจะไม่ทำในส่วนของ CATCH ข้ามมาทำงานที่ฟังก์ชัน InitDIBData( ) และเช็คค่าของ m_hDIB ซึ่งเป็น handle ที่ใช้ในการจัดการกับภาพ ถ้า m_hDIB เป็น null จะแสดงกล่องข้อความแจ้งว่าไม่สามารถเปิดภาพนั้นได้ ดังรูปที่ 1จ. และคืนค่าออกจากฟังก์ชันเป็นเท็จ ถ้า m_hDIB ไม่เป็น null จะ set path ของไฟล์ที่เปิด เมื่อเปิดครั้งต่อไปจะเปิดตาม path ที่เก็บไว้ จากนั้นจะออกจากฟังก์ชันโดยคืนค่าเป็นจริง ถ้าการอ่านไฟล์มี exception เกิดขึ้น ฟังก์ชันจะเข้ามาทำงานในส่วน CATCH (CFileException, eLoad) เคลียร์ค่าให้ m_hDIB เป็น null และคืนค่าออกจากฟังก์ชันเป็นเท็จ

รูปที่ 1จ. แสดงข้อความแจ้งว่าไม่สามารถเปิดภาพได้

2. การอ่าน DIB file

ชื่อฟังก์ชัน ReadDIBFile( )

คำอธิบาย

```
HDIB WINAPI ReadDIBFile(CFile& file)
{
    BITMAPFILEHEADER bmfHeader;
    DWORD dwBitsSize;
```

```
HDIB hDIB;
LPSTR pDIB;

// get length of DIB in bytes for use when reading
dwBitsSize = file.GetLength();
// Go read the DIB file header and check if it's valid.
if (file.Read((LPSTR)&bmfHeader, sizeof(bmfHeader)) != sizeof(bmfHeader))
        return NULL;

if (bmfHeader.bfType != DIB_HEADER_MARKER)
        return NULL;

// Allocate memory for DIB
hDIB = (HDIB) ::GlobalAlloc(GMEM_MOVEABLE | GMEM_ZEROINIT, dwBitsSize);

// Go read the bits.
if (file.ReadHuge(pDIB, dwBitsSize - sizeof(BITMAPFILEHEADER)) !=
        dwBitsSize - sizeof(BITMAPFILEHEADER) )
{
        ::GlobalUnlock((HGLOBAL) hDIB);
        ::GlobalFree((HGLOBAL) hDIB);
        return NULL;
}
::GlobalUnlock((HGLOBAL) hDIB);
return hDIB;
}
```

การอ่านข้อมูลไฟล์ภาพจะทำงานโดยรับไฟล์ภาพที่จะอ่านเข้ามาเก็บไว้ในหน่วยความจำ (GlobalAlloc( )) โดยมีตัวแปรประเภท hDIB เป็นตัวจัดการ ถ้า DIB file header ไม่ถูกต้อง ฟังก์ชันจะเคลียร์หน่วยความจำที่จองไว้ (GlobalUnlock( ), GlobalFree( )) และคืนค่าของ hDIB กลับมาเป็น null

3. การกำหนดขนาดและ palette ของภาพที่นำมาแสดงบนหน้าจอ

ชื่อฟังก์ชัน InitDIBData( )

คำอธิบาย

DibDoc.cpp

```
void CDibDoc::InitDIBData()
{
        ...
        LPSTR lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB);
        m_sizeDoc = CSize((int) ::DIBWidth(lpDIB), (int) ::DIBHeight(lpDIB));
        ::GlobalUnlock((HGLOBAL) m_hDIB);
        // Create copy of palette
        m_palDIB = new CPalette;
        if (::CreateDIBPalette(m_hDIB, m_palDIB) == NULL)
        {
                // DIB may not have a palette
                delete m_palDIB;
                m_palDIB = NULL;
                return;
        }
}
```

การนำข้อมูลของภาพที่เก็บไว้ในหน่วยความจำมาใช้ จะใช้ตัวแปรประเภทพอยน์เตอร์ชี้ไปยังหน่วยความจำนั้น เมื่อเริ่มการทำงานของฟังก์ชันจึงต้องใช้ตัวแปร lpDIB ซึ่งเป็น long pointer string ชี้ไปยังหน่วยความจำที่จองไว้โดยตัวแปร m_hDIB จากนั้นตัวแปร m_sizeDoc ซึ่งเป็นตัวแปรประเภท Csize จะใช้ในการเก็บขนาดความกว้างและความยาวของภาพโดยสามารถหาได้จากการใช้ฟังก์ชัน DIBWidth( ) และ DIBHeight( ) ตามลำดับ เมื่อทำงานกับหน่วยความจำนั้นเสร็จแล้วจะต้องเรียกฟังก์ชัน GlobalUnlock ( ) เสมอ เพื่อยกเลิกการจองหน่วยความจำ ต่อมาเราจะทำการสร้าง palette สำหรับภาพที่จะแสดงบนหน้าจอ โดยสร้าง palette ใหม่โดยใช้คำสั่ง new CPalette จากนั้นจึงเรียกฟังก์ชัน CreateDIBPalette( ) เพื่อสร้าง logical palette และจะคืนค่าของ palette handle กลับมาในตัวแปร m_palDIB ถ้าการสร้าง logical palette ไม่สำเร็จจะลบค่าของ m_palDIB นั้นทิ้งและกำหนดให้มีค่าเป็น null

4. การนำข้อมูลที่อยู่ใน Class Document มาใช้ใน Class View

<u>ชื่อฟังก์ชัน</u> GetDocument( )

<u>คำอธิบาย</u>

```
CDibDoc* GetDocument()
        {
                ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CDibDoc)));
                return (CDibDoc*) m_pDocument;
        }
```

5. การแสดงภาพของไฟล์ที่เปิดบนหน้าจอ

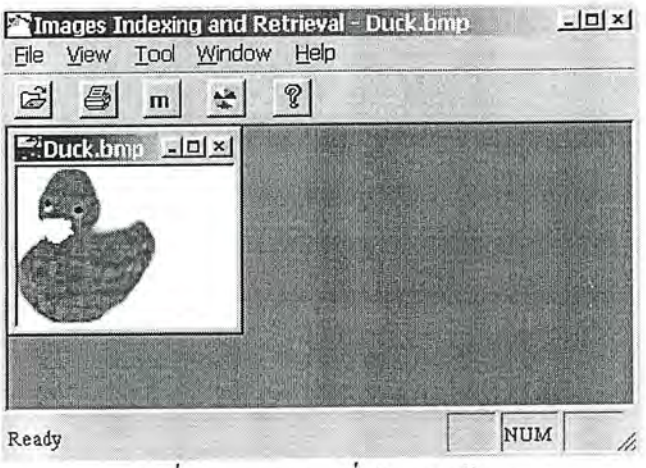
ชื่อฟังก์ชัน OnDraw( )

คำอธิบาย

การแสดงภาพบนหน้าจอจะต้องมีพอยน์เตอร์ชี้ไปยังหน่วยความจำที่เก็บไว้โดย handle ของ window และวาดภาพบนหน้าจอโดยเรียกฟังก์ชัน PaintDIB( )

```
void CDibView::OnDraw(CDC* pDC)
{
        CDibDoc* pDoc = GetDocument();

        HDIB hDIB = pDoc->GetHDIB();
        if (hDIB != NULL)
        {
                LPSTR lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) hDIB);
                int cxDIB = (int) ::DIBWidth(lpDIB);        // Size of DIB - x
                int cyDIB = (int) ::DIBHeight(lpDIB);       // Size of DIB - y
                ::GlobalUnlock((HGLOBAL) hDIB);

                CRect rcDIB;
                rcDIB.top = rcDIB.left = 0;
                rcDIB.right = cxDIB;
                rcDIB.bottom = cyDIB;

                CRect rcDest;
                rcDest = rcDIB;

                ::PaintDIB(pDC->m_hDC, &rcDest, hDIB,
                        &rcDIB, pDoc->GetDocPalette());
                }
        ...
}
```

การสั่งให้วาดโดยใช้ฟังก์ชัน PaintDIB( ) จะต้องมีการส่งค่าของตัวแปรต่างๆคือ m_hDC ซึ่งเป็น handle ของ device context, &rcDest ซึ่งเก็บตำแหน่งของภาพที่จะแสดงบนหน้าจอ, &rcDIB ซึ่งเก็บตำแหน่งตามขนาดของภาพ และ logical palette ของภาพ จากฟังก์ชันนี้เรากำหนดให้ rcDest = rcDIB ทำให้ขนาดของภาพที่แสดงบนหน้าจอจะเท่ากับขนาดของภาพจริง ดังแสดงในรูปที่ 2จ.



รูปที่ 2จ. แสดงภาพที่เปิดบนหน้าจอ

6. การทำให้ขนาดของภาพที่แสดงบนหน้าจอเป็นไปตามที่กำหนด

ชื่อฟังก์ชัน PaintDIB( )

คำอธิบาย

```
// Dibapi.cpp
BOOL WINAPI PaintDIB (HDC hDC, LPRECT lpDCRect, HDIB hDIB,
                      LPRECT lpDIBRect, Cpalette* pPal)
{
        LPSTR   lpDIBHdr;       // Pointer to BITMAPINFOHEADER
        LPSTR   lpDIBBits;      // Pointer to DIB bits
        BOOL    bSuccess=FALSE;  // Success/fail flag
        HPALETTE hPal=NULL;      // Our DIB's palette
        HPALETTE hOldPal=NULL;   // Previous palette

        /* Check for valid DIB handle */
        if (hDIB == NULL)
                return FALSE;

        // Lock down the DIB, and get a pointer to the beginning of the bit  buffer
        lpDIBHdr  = (LPSTR) ::GlobalLock((HGLOBAL) hDIB);
        lpDIBBits = ::FindDIBBits(lpDIBHdr);

        // Get the DIB's palette, then select it into DC
        if (pPal != NULL)
```

```
    {
        hPal = (HPALETTE) pPal->m_hObject;

        // Select as background since we have
        // already realized in forground if needed
        hOldPal = ::SelectPalette(hDC, hPal, TRUE);
    }

    /* Make sure to use the stretching mode best for color pictures */
    ::SetStretchBltMode(hDC, COLORONCOLOR);

    /* Determine whether to call StretchDIBits() or SetDIBitsToDevice() */
    if ((RECTWIDTH(lpDCRect)  == RECTWIDTH(lpDIBRect)) &&
      (RECTHEIGHT(lpDCRect) == RECTHEIGHT(lpDIBRect)))
        bSuccess = ::SetDIBitsToDevice(
        ...
    else
        bSuccess = ::StretchDIBits(
        ...
    return bSuccess;
}
```

การวาดภาพบนหน้าจอ จะวาดลงบน device context ที่ระบุตามตำแหน่งที่ชี้โดยตัวแปร lpDCRect โดยมีขอบเขตของภาพที่แสดงตามที่ชี้โดยตัวแปร lpDIBRect ซึ่งถ้าขนาดของภาพที่แสดงบนหน้าจอเท่ากับขนาดภาพจริงจะวาดโดยเรียกฟังก์ชัน SetDIBitsToDevice( ) ถ้าขนาดไม่เท่ากันจะวาดโดยใช้ฟังก์ชัน StretchDIBits( )

7. การแสดงการตีกรอบบนภาพเป้าหมาย

คำอธิบาย (เป็นส่วนหนึ่งของฟังก์ชัน OnDraw( ))

```
DibView.cpp
void CdibView::OnDraw(CDC* pDC)
{
    ...
    // Draw box boundary
```

```
UINT x1 = pDoc->start_point.x;
UINT y1 = pDoc->start_point.y;
UINT x2 = pDoc->end_point.x;
UINT y2 = pDoc->end_point.y;

boxWidth  = x2 – x1;
boxHeight = y2 – y1;

Cbrush brush(boxColor);

pDC->FrameRect(Crect(x1,y1,x2,y2),&brush);
...
}
```

การตีกรอบลงบนภาพทำได้โดยนำค่าของจุดเริ่มต้นในการลากกรอบ (มุมบนซ้าย:x1,y1) และจุดสิ้นสุดของกรอบ (มุมล่างขวา:x2,y2) มาใช้วาดกรอบสี่เหลี่ยม คำนวณขนาดของกรอบภาพเป้าหมายเก็บไว้ กำหนดตัวแปรประเภท Cbrush ขึ้นมาโดยจะมีค่าสีตามที่กำหนดในตัวแปร boxColor และทำการวาดกรอบโดยใช้ฟังก์ชัน FrameRect( ) ดังแสดงในรูปที่ 3จ.

รูปที่ 3จ. แสดงการตีกรอบบนภาพเป้าหมาย

8. การกำหนดจุดเริ่มต้นของกรอบที่ลาก

ชื่อฟังก์ชัน OnLButtonDown( )

คำอธิบาย

ทุกครั้งที่กดปุ่มเมาส์ด้านซ้ายจะเป็นการกำหนดจุดเริ่มต้นของกรอบใหม่ และสั่งให้วาดกรอบใหม่ทุกครั้งที่กดปุ่มเมาส์ด้านซ้ายโดยใช้ฟังก์ชัน Invalidate( ) เป็นตัวเรียกให้ฟังก์ชัน OnDraw( ) ของ DibView มีการทำงาน

```
// DibView.cpp
void CDibView::OnLButtonDown(UINT nFlags, CPoint point)
{
        CDibDoc* pDoc = GetDocument();
        pDoc -> start_point = point;
        Invalidate();
        CScrollView::OnLButtonDown(nFlags, point);
}
```

9. การส่งค่าในการตีกรอบขณะที่ทำการลากเมาส์ (drag and drop)

ชื่อฟังก์ชัน OnMouseMove( )

คำอธิบาย

เมื่อเมาส์มีการกดปุ่มซ้ายค้างไว้ในขณะที่ลากเมาส์จะทำให้ภาพบนหน้าจอและกรอบสี่เหลี่ยมถูกวาดใหม่ตามค่าของตำแหน่งจุดสิ้นสุดของกรอบ

```
// DibView.cpp
void CDibView::OnMouseMove(UINT nFlags, CPoint point)
{
        CDibDoc* pDoc = GetDocument();
                while (nFlags)
                {
                        pDoc -> end_point = point;
                        nFlags = 0;
                        Invalidate();
                }
 CScrollView::OnMouseMove(nFlags, point);
}
```

การส่งค่าตำแหน่งจุดสิ้นสุดของกรอบจะทำการวนลูปส่งไปเรื่อยๆ โดยมีตัวแปร nFlags ที่ระบุค่าว่าเมาส์, ปุ่ม shift หรือปุ่ม ctrl มีการกดอยู่หรือไม่ เมื่อใดก็ตามที่มีการกดอยู่ก็จะมีการนำค่าของ point ไปเก็บไว้ในตัวแปร end_point เสมอ และสั่งให้ทำการวาดกรอบสีเหลี่ยมโดยใช้ฟังก์ชัน Invalidate( )

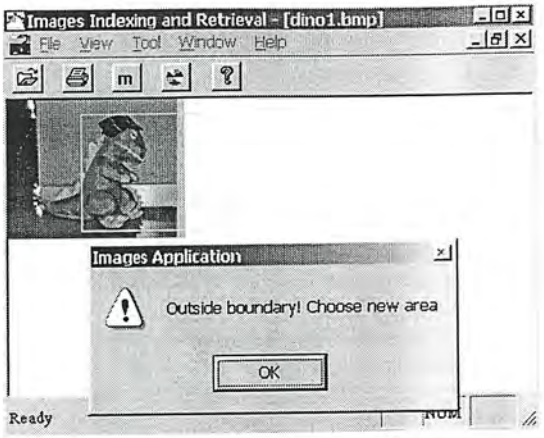
10. การเช็คก่อนทำการคำนวณค่า RGB

คำอธิบาย (เป็นส่วนหนึ่งของฟังก์ชัน Rgbvalue( ))

```
// DibView.cpp
void CDibView::Rgbvalue()
```

```
{
    ...
    boxWidth = x2 - x1;
    boxHeight = y2- y1;

    HDIB hDIB = pDoc->GetHDIB();
    LPSTR lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) hDIB);
    int width = (int) ::DIBWidth(lpDIB);        // Size of DIB - x
    int height = (int) ::DIBHeight(lpDIB);      // Size of DIB - y
    ::GlobalUnlock((HGLOBAL) hDIB);             // check that already defined boundary yet

    if (boxHeight > 0 && boxWidth > 0)
    {
        // check box boundary if outside picture not store histogram & file
        if (x2 > width || y2 > height)
        {
            AfxMessageBox("Outside boundary! Choose new area");
            if (!target.IsEmpty());
                target.Empty();
        }
        else
        {
        ...
    }
```

ก่อนคำนวณค่า RGB จะมีการเช็คการตีกรอบสำหรับภาพเป้าหมาย โดยเช็คจากความกว้างและความยาวของกรอบและเช็คจากตำแหน่งของจุดสิ้นสุดของการตีกรอบกับความกว้างและความยาวของภาพ ถ้าขอบเขตของกรอบเกินออกมานอกภาพแล้วเรียกฟังก์ชัน Matching picture( ) จะมีกล่องข้อความเตือนดังแสดงในรูปที่ 4จ.



รูปที่ 4จ. แสดงกล่องข้อความเตือนว่ากรอบอยู่นอกขอบเขตของภาพ

ภาคผนวก ฉ
การทำงานของโปรแกรมในส่วนของการคำนวณ

1. การส่งค่า RGB ของแต่ละพิกเซลในกรอบที่กำหนดไปคำนวณ

คำอธิบาย (เป็นส่วนหนึ่งของฟังก์ชัน Rgbvalue( ))

```
//DibView.cpp
void CDibView::Rgbvalue()
{
    ...
        CPalette* cPal = pDoc->GetDocPalette();
        BeginWaitCursor();
        ToScreen hisbuff = InitHistogram(hisbuff);

        for (int v=0; v<boxHeight; ++v)
        {
            for (int h=0; h<boxWidth; ++h)
            {
                HSVQUAD value = ::findHSV(hDIB,cPal,x1,y1);
                hisbuff = :: StoreHistogram(hisbuff,value);
                x1 += 1;
            }
            x1 -= boxWidth;  y1 += 1;
        }
        EndWaitCursor();
    ...
}
```

ก่อนเก็บฮิสโตแกรมของภาพจะต้องเคลียร์บัฟเฟอร์ของฮิสโตแกรมก่อนเสมอโดยใช้ฟังก์ชัน InitHistogram( ) ซึ่งจะทำให้ฮิสโตแกรมบัฟเฟอร์มีค่าเริ่มต้นเป็น 0 ทั้งหมด จากนั้นจึงวนลูปทำงานตั้งแต่พิกเซลที่ตำแหน่งจุดเริ่มต้นกรอบโดยใช้ฟังก์ชัน findHSV( ) เปลี่ยนค่า RGB ของแต่ละพิกเซลไปเป็น HSV และเก็บลงในฮิสโตแกรมโดยใช้ฟังก์ชัน StoreHistogram( ) และเพิ่มตำแหน่งพิกเซลและนำไปคำนวณเรื่อยๆ จนครบทุกพิกเซลที่อยู่ในขอบเขตขอบกรอบภาพ

2. **การดึงค่า RGB ของพิกเซลจากไฟล์**

ชื่อฟังก์ชัน GetPixel( )

คำอธิบาย

การนำพิกเซลของภาพไปหาค่า RGB จะใช้ตำแหน่งของพิกเซลบนจอภาพที่แสดงไปคำนวณหาตำแหน่งของพิกเซลนั้นในไฟล์ โดยแยกการคำนวณออกเป็น 3 กรณี คือ ภาพ 24 บิทต่อพิกเซล, ภาพ 8 บิทต่อพิกเซล และภาพ 4 บิทต่อพิกเซล กรณีอื่นๆ นอกจากนี้จะไม่สามารถหาค่า RGB ในพิกเซลได้

```
// DibOper.cpp
RGBQUAD WINAPI GetPixel(HDIB m_hDIB, CPalette* m_palDIB, int x, int y)
{
        static RGBQUAD pixColor = {0, 0, 0, 0};
        UINT RowByteCount;

        LPSTR lpDIB = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB);

        // Get starting address of pixel data and color table
        LPSTR lpDIBBits;
        lpDIBBits = ::FindDIBBits(lpDIB);
        BYTE * dibits = (BYTE *) lpDIBBits;
        LPRGBQUAD pDibQuad = (LPRGBQUAD) m_palDIB;

        // Now extract the color data
        int temp = (int)::BitCount(lpDIB);

        ULONG width = ::DIBWidth(lpDIB);
        ULONG height =          ::DIBHeight(lpDIB);
        ::GlobalUnlock((HGLOBAL) m_hDIB);

        switch(temp)
        {
                case 24:
                // Calculate the number of bytes in a row
                RowByteCount = ((((width * 3) -1) / 4) + 1) * 4;
                // Move to the correct row (correct for bottom up DIB storage)
                dibits -= (height - y) * RowByteCount;
```

```
// Move to the correct pixel address
dibits += x * 3;
// Load up pixColor
pixColor.rgbBlue = dibits[0];
pixColor.rgbGreen = dibits[1];
pixColor.rgbRed = dibits[2];
break;

case 8:
// Calculate the number of bytes in a row
RowByteCount = (((width -1) / 4) + 1) * 4;
// Move to the correct row (correct for bottom up DIB //storage)
dibits += (height - y) * RowByteCount;
// Move to the correct pixel address
dibits += x;
pDibQuad += dibits[0];
pixColor.rgbBlue = pDibQuad->rgbBlue;
pixColor.rgbGreen = pDibQuad->rgbGreen;
pixColor.rgbRed = pDibQuad->rgbRed;
break;

case 4:
// Calculate the number of bytes in a row
RowByteCount = ((((width >> 1)-1) / 4) + 1) * 4;
// Move to the correct row (correct for bottom up DIB //storage)
dibits += (height - y) * RowByteCount;
// Move to the correct pixel address
dibits += (x >> 1);
if ( x%2 )
pDibQuad += dibits[0] & 0x0f;
else
pDibQuad += (dibits[0] & 0xf0) >> 4;
pixColor.rgbBlue = pDibQuad->rgbBlue;
pixColor.rgbGreen = pDibQuad->rgbGreen;
```

```
                pixColor.rgbRed = pDibQuad->rgbRed;
                break;

                default:
                AfxMessageBox("This BMP format is not supported!", MB_ICONSTOP | MB_OK);
                pixColor.rgbBlue = pixColor.rgbGreen = pixColor.rgbRed = 0;
                break;
        }
        return pixColor;
}
```

3. การแปลงสีและลดระดับสี

ชื่อฟังก์ชัน findHSV( )

คำอธิบาย

ฟังก์ชัน findHSV จะทำการแปลงสีและลดระดับสีจากพิกเซลของ RGB color space ให้เป็น 166 สี HSV color space โดยจะรับตัวแปร hDIB ที่เป็น handle ของภาพ , Palette และพิกัด x, y ของพิกเซล

```
//DibOper.cpp
HSVQUAD WINAPI findHSV(HDIB hDIB, CPalette* cPal, int pointX, int pointY)
```

จากที่ได้กล่าวไปแล้ว ในส่วนของการแปลงสี เราจะทำการตรวจสอบก่อนว่าพิกเซลสีอยู่ในกลุ่มสีเทาหรือไม่ (มีค่า R, G, B เท่ากัน) โดยใช้คำสั่ง if..then

```
RGBQUAD in;
HSVQUAD out;
if (((float) in.rgbRed == (float)in.rgbGreen) && ((float)in.rgbRed == (float)in.rgbBlue))
        {
                if ((float)in.rgbRed > 191.25)
                {       out.Gray = 3;}
                else if ((float)in.rgbRed > 127.5)
                {       out.Gray = 2;}
                else if ((float)in.rgbRed > 63.75)
                {       out.Gray = 1;}
                else {out.Gray =0;}
                out.Saturation = 0;
          out.Value = 255;  // not add in histogram
```

```
out.Hue = 255;
}
```

คำสั่ง (float) จะทำการแปลงค่า จาก integer ให้เป็น float เนื่องจากจะต้องเอาค่าของ R, G, B มาเข้าสมการการแปลงสี หากว่าพิกเซลสีเป็นสีเทาจะไม่ต้องทำการแปลงสี ซึ่งในที่นี้จะเข้าสู่ statement then คือทำการลดระดับสีได้เลย ส่วน out.Gray จะเป็นเหมือนออปเจ็คที่เก็บระดับของสีเทาไว้ ซึ่งสุดท้ายเมื่อฟังก์ชัน FindHSV นี้ทำงานเสร็จสิ้นก็จะส่งค่าออปเจ็ค out นี้ออกไป เช่นเดียวกับ out.Saturation, out.Value และ out.Hue การกำหนดให้ out.Value และ out.Hue เป็น 255 จะเป็นค่าที่บอกว่าไม่สามารถระบุค่าของสีและความสว่างให้ได้ ซึ่งค่านี้จะนำไปใช้ในการคำนวณเพื่อสร้างฮิสโตแกรม

แต่ถ้าพิกเซลสีไม่ใช่สีเทา จะมาทำที่สเตทเมนต์ else ซึ่งจะทำการปรับอัตราส่วนให้อยู่ระหว่าง 0 ถึง 1 ทั้งนี้เพื่อให้ ค่าของ R, G, B เป็นค่าที่พร้อมจะคำนวณ (pre-condition) ซึ่งคำนวณได้ดังสเตทเมนต์ดังนี้

```
else {
        RGB =  (float) in.rgbRed+(float)in.rgbGreen+(float)in.rgbBlue;
        r =  (float)in.rgbRed/RGB;
        g =  (float)in.rgbGreen/RGB;
        b =  (float)in.rgbBlue/RGB;                    //normalize
```

ค่า r, g, b เป็นค่าของ R, G, B ที่ผ่านการปรับอัตราส่วนจนอยู่ในรูปแบบที่เหมาะสม เมื่อปรับอัตราส่วนค่าของสีได้แล้ว จึงเข้าสู่สมการการแปลงสี ดังนี้คือ

```
MaxValue = max(r,g);
MaxValue = max(b,MaxValue);
V = MaxValue;          //find max of three value
MinValue = min(r,g);
MinValue = min(b,MinValue);    // find min of three value
S = (V-MinValue)/V ;

tempr = (V-r)/(V-MinValue);
tempg = (V-g)/(V-MinValue);
tempb = (V-b)/(V-MinValue);

if ((r == MaxValue) && (g == MinValue))
{ H =  5+tempb;}
else if ((r == MaxValue) && (g != MinValue))
```

```
        { H = 1-tempg;}
else if ((g == MaxValue) && (b == MinValue))
        { H = 1+tempr;}
else if ((g == MaxValue) && (b != MinValue))
        { H = 3-b;}
else if ((b == MaxValue) && (r == MinValue))
        { H =  3+tempg;}
    else  { H = 5-tempr;}
H = H/6;     // real Hvalue
```

ฟังก์ชัน min และ max ได้จากการกำหนดมาโคร ในส่วนของการ define การกำหนดมาโครทำให้เรียกฟังก์ชันนี้ได้เมื่อต้องการ การกำหนดมาโคร ในส่วนของการ define ทำได้ดังนี้

```
#define max(a, b)  (((a) > (b)) ? (a) : (b))
#define min(a, b)  (((a) < (b)) ? (a) : (b))
```

การลดระดับของความเข้มสี จะมีสเตทเมนต์เหมือนกับการลดระดับค่าความสว่าง เนื่องจากมีการแบ่งระดับที่เท่ากัน และเช่นเดียวกับการแบ่งระดับของสีเทาคือ ระดับของความเข้มสีและความสว่างจะถูกเก็บอยู่ในออปเจ็ค out

```
if (S > 0.333333333)
        { if (S >0.666666666)
                { out.Saturation = 2;}
          else { out.Saturation = 1;} }
else { out.Saturation = 0;}                          // get quantize S

if (V > 0.333333333)
        { if (V >0.666666666)
                { out.Value = 2;}
          else { out.Value = 1;} }
else { out.Value = 0;}                                  // get quantize V
```

การลดระดับค่าของสี จะทำการคำนวณโดยจำลองคล้ายวิธีของ binary search คือแบ่งทีละครึ่งของการค้นหาก่อนและค่อยๆแบ่งทีละครึ่งไปเรื่อยๆจนกระทั่งพบระดับที่มีช่วงของสีตรงกัน แล้วกำหนดให้ออปเจ็ค out เก็บค่าระดับของสี

```
if (H > 0.5)
 if (H > 0.777777777)
        if (H > 0.888888888)
              {if (H > 0.944444444)
                    { out.Hue = 17;}          // >0.944->1.0
              else {   out.Hue = 16;} }  // >0.888-0.944
        else if (H > 0.833333333)
                          { out.Hue = 15;}  // >0.833-0.888
               else { out.Hue = 14;}              // >0.777-0.833

 else if (H > 0.666666666)
              {if (H > 0.722222222)
                    { out.Hue = 13;}              // >0.722-0.777
              else {   out.Hue = 12;} }          // >0.666-0.722
        else if (H > 0.555555555)
              { if (H > 0.611111111)
                    { out.Hue = 11;}          // >0.611-0.666
                 else {out.Hue = 10;}} // >0.555-0.611
              else { out.Hue = 9;}              // >0.5-0.555
else if (H > 0.277777777)
        if (H > 0.388888888)
              if (H >0.444444444)
                    { out.Hue = 8 ;}          // >0.444-0.5
              else { out.Hue = 7;}              // >0.388-0.444
           else if (H > 0.333333333)
                    { out.Hue = 6;}          // >0.333-0.388
                 else { out.Hue = 5;}        // >0.277-0.333
        else if (H > 0.166666666)
              if (H > 0.222222222)
                    { out.Hue = 4;}          // >0.222-0.277
              else {  out.Hue = 3;}              // >0.166-0.222
            else if (H > 0.055555555)
                    if (H > 0.111111111)
                          { out.Hue = 2 ;}  // >0.111-0.166
```

```
                else {out.Hue = 1;}          // >0.055-0.111
                else { out.Hue = 0;}                  // <0.0-0.055
                                                      // get quantize H
        out.Gray = 255;
        } // end else
        return out;
```

ในส่วนของการลดระดับสีค่าของสี จะสังเกตว่า มีการกำหนดให้ out.Gray เป็น 255 ในตอนท้าย เนื่องจากเป็นการระบุว่าพิกเซลนี้ไม่ใช่สีเทา และค่านี้จะนำไปนับเป็นส่วนหนึ่งของฮิสโตแกรม คำสั่ง return จะทำการส่งค่าของออปเจ็ค out ซึ่งจะประกอบไปด้วยแอททริบิวต์ของระดับของสี ระดับความเข้มสี ระดับความสว่างของสี และระดับของสีเทา กลับคืนไปยังฟังก์ชัน OnToolRBGvalue( )

4. การสร้างฮิสโตแกรมของสี

ชื่อฟังก์ชัน StoreHistogram( )

คำอธิบาย

```
//DibOper.cpp
ToScreen StoreHistogram(ToScreen bu,HSVQUAD value)
```

เป็นฟังก์ชันที่ใช้นับจำนวนสีในแต่ละระดับเพื่อสร้างฮิสโตแกรม โดยฟังก์ชันจะรับค่าของ bu ซึ่งเป็นฮิสโตแกรมว่างที่ได้จากการสร้างของฟังก์ชัน InitHistogram( ) และ value เป็นค่าของออปเจ็ค out ที่รับมาจากฟังก์ชัน OnToolRGBvalue( ) มีรายละเอียดของฟังก์ชันดังนี้

```
        switch(value.Hue) {     // store hue
                        case 0: ++bu.hue[0];break;
                        case 1: ++bu.hue[1];break;
                        case 2: ++bu.hue[2];break;
                        case 3: ++bu.hue[3];break;
                        case 4: ++bu.hue[4];break;
                        case 5: ++bu.hue[5];break;
                        case 6: ++bu.hue[6];break;
                        case 7: ++bu.hue[7];break;
                        case 8: ++bu.hue[8];break;
                        case 9: ++bu.hue[9];break;
                        case 10: ++bu.hue[10];break;
                        case 11: ++bu.hue[11];break;
                        case 12: ++bu.hue[12];break;
                        case 13: ++bu.hue[13];break;
```

```
                case 14: ++bu.hue[14];break;
                case 15: ++bu.hue[15];break;
                case 16: ++bu.hue[16];break;
                case 17: ++bu.hue[17];break;
                default: ++bu.hue[18];
        }
        switch (value.Saturation){  //store Saturation
                case 0 : ++bu.sat[0];break;
                case 1 : ++bu.sat[1];break;
                case 2 : ++bu.sat[2];break;
                default : ++bu.sat[3];
        }
        switch (value.Value){                   //store Value
                case 0 : ++bu.val[0];break;
                case 1 : ++bu.val[1];break;
                case 2 : ++bu.val[2];break;
                default : ++bu.val[3];
        }
        switch (value.Gray){                    //store Gray
                case 0 : ++bu.gra[0];break;
                case 1 : ++bu.gra[1];break;
                case 2 : ++bu.gra[2];break;
                case 3 : ++bu.gra[3];break;
                default : ++bu.gra[4];
        }
        return bu;
```

คำสั่ง switch(value.Hue) จะทำการเลือกระดับของสีเพื่อทำการสร้างฮิสโตแกรมของสี เช่นเดียวกับ switch (value.Saturation), switch(value.Value) และ switch(value.Gray) จะทำการสร้างฮิสโตแกรมของความเข้มสี ความสว่างของสี และ สีเทาตามลำดับ ทั้งนี้ออปเจ็ค out ที่มีแอททริบิวต์เป็น 255 จะถูกจัดให้อยู่ใน default ซึ่งเป็นระดับที่สร้างขึ้นเพื่อบอกความผิดพลาดของการคำนวณ หรือบ่งบอกว่าเป็นสีเทา แต่ไม่ใช่เป็นระดับสี

5. การกำหนดค่าเริ่มต้นก่อนการสร้างฮิสโตแกรม

ชื่อฟังก์ชัน InitHistogram( )

คำอธิบาย

เป็นฟังก์ชันในการกำหนดค่าเริ่มต้นให้กับแต่ละรับสีของฮิสโตแกรมก่อนที่จะทำการสร้างฮิสโตแกรมสีขึ้นมาก โดยจะทำการวนลูปเพื่อกำหนดค่าในแต่ละระดับสี ฟังก์ชันนี้รับค่า bu ซึ่งเป็นออปเจ็คที่มีแอททริบิวต์เป็นอะเรย์ข้อมูลค่าของสี ค่าความเข้มสี ค่าความสว่างของสี จากฟังก์ชัน OnToolRGBvalue() และส่งค่าฮิสโตแกรมกลับฟังก์ชัน OnToolRGBvalue( ) ด้วยคำสั่ง return bu

```
//DibView.cpp
ToScreen CDibView::InitHistogram(ToScreen bu)
for (int ih = 0;ih < 19;++ih)
        bu.hue[ih]=0;
for (int is = 0;is < 5;++is)
        bu.sat[is]=0;
for (int iv = 0;iv < 5;++iv)
        bu.val[iv]=0;
for (int ig =0;ig <6;++ig)
        bu.gra[ig] = 0;
return bu;
```

6. การแปลงค่าของฮิสโตแกรมของภาพโมเดลจากรหัสแอสกี้ให้เป็นจำนวนเต็ม

คำอธิบาย (เป็นส่วนหนึ่งของฟังก์ชัน OnToolMatching( ))

ในตอนแรกข้อมูลของฮิสโตแกรมที่รับเข้ามาจะเป็นฮิสโตแกรมของภาพโมเดล ซึ่งเป็นสตรีมของรหัสแอสกี้ เนื่องจากการใช้งานต้องใช้ฮิสโตแกรมที่เป็นจำนวนเต็มในการคำนวณ จึงทำการแปลงรหัสแอสกี้ให้เป็นจำนวนเต็มโดยใช้ฟังก์ชัน Find เพื่อหาตำแหน่ง " | " และทำการเซตตำแหน่งของ " | " ให้เป็น " x " โดยใช้คำสั่ง SetAt

```
//DibView.cpp ฟังก์ชัน OnToolMatching( )
expo = model.Find('|');
model.SetAt(expo,'x');
```

การทำเช่นนี้จะเป็นการกำหนดตัวเลขแรกในสตรีม ที่จะทำการแปลงจากรหัสแอสกี้ให้เป็นจำนวนเต็ม และเข้าสู่การวนลูปเพื่อทำการแปลงรหัสแอสกี้ทีละตัวในแต่ละลูป จนกระทั่งพบอักษร "*" จึงหยุดการวนลูป

```
        while (model.GetAt(c) != '*')
        if (model.GetAt(c) == 'x' )
        {
                if (model.GetAt(++c) != ' ')
```

```
                {
                        expo = model.Find('|');
                        model.SetAt(expo,'x');
                        expo = expo - (c + 2) ;
                        ++c;
                        num = 0;
                }
        else
                {
                        expo = model.Find('|');
                        model.SetAt(expo,'x');
                        expo = expo -( c + 2);
                        num = 0 ;
                        ++c;
                }
        }
        else
        {
                cnum = model.GetAt(c++);
                count = asctoint(cnum);
                num = pow(10,expo)*count+num;
                --expo;
                 if (expo == -1)
                        {model_buff[order++] = num;}
        }
}
```

ในส่วนของ if (model.GetAt(c) == 'x') จะทำการตรวจสอบตำแหน่งของข้อมูลที่ตำแหน่ง c ถ้าเป็น x จะทำการค้นหาตำแหน่งถัดไปซึ่งเป็นตำแหน่งของ " " ก่อนจากสเตทเมนท์ if (model.GetAt(++c) != ' ') ซึ่งเป็นช่องว่างที่แบ่งประเภทของฮิสโตแกรมออกจากกันในสตรีมข้อมูล หากเป็น " " จะค้นหาตำแหน่ง " | " ตำแหน่งต่อไปและเซตตำแหน่งของ " | " ให้เป็น "x" โดยจะนับรวมตำแหน่งช่องว่างนี้ไปด้วย แต่ถ้าตำแหน่งถัดไปไม่เป็น " " ก็จะทำการแทนตำแหน่ง " | " ด้วย "x" ก่อน จึงค้นหาตำแหน่งของ " | " ที่อยู่ในสตรีมต่อไป

ตัวแปร expo จะเป็นตัวนับตำแหน่งของสตรีมที่เป็นหลักของตัวเลขเพื่อจะทำการแปลงเลขจากรหัสแอสกี้ 1 ตัวให้เป็นจำนวนเต็ม 1 หลัก เมื่อตำแหน่งในสตรีมข้อมูลไม่เป็น " | " จะทำการแปลงตัวเลขที่อยู่ระหว่าง "x" และเริ่มแปลงตัวเลขหลักมากสุดก่อนที่ตำแหน่งของตัวแปร cnum การแปลงใช้ฟังก์ชัน asctoint ซึ่งเป็นฟังก์ชันที่สร้างขึ้นโดยใช้สมการ total = 10*total+(cnum-'0') โดยที่ total เป็นจำนวนเต็มของรหัสแอสกี้ cnum ฟังก์ชัน asctoint จะส่งค่ากลับมาให้ count จากนั้นจะเข้าสมการ num = pow (10,expo)*count+num เพื่อให้ได้ค่าจำนวนเต็มที่อยู่ในหลักที่ถูกต้อง และค่า expo นี้จะลดลงทีละ 1 ยกตัวอย่างการแปลงค่าเช่น สตรีมข้อมูลที่เป็นเลข 146 จะแปลง 1 ซึ่งเป็นหลักร้อยก่อน และทำการแปลง 4 ซึ่งเป็นหลักสิบและเอาค่า 100 มาบวก 40 เป็น 140 ก่อนจากนั้นจะแปลง 6 ให้เป็นหลักหน่วยแล้วนำมาบวกกับอีก 140 เป็น 146 ซึ่งเป็นจำนวนเต็ม(ตอนนี้ค่า expo จะเป็น –1) สเตทเมนท์ต่อไปจะเช็คค่า expo ถ้ามีค่าเป็น –1 แสดงว่าได้ทำการแปลงตัวเลขได้ระดับสีหนึ่งระดับแล้วก็จะทำการเก็บเลขจำนวนเต็มนี้ลงใน model_buff ซึ่งเป็นอะเรย์ข้อมูลฮิสโตแกรมทีละระดับเรียงกันไป ซึ่งฮิสโตแกรมที่จะเอาไปคำนวณค่าความเหมือนต่อไป จากนั้นจะทำการหาตำแหน่ง " | " ในสตรีมต่อไปจนกระทั่งหมดสตรีมข้อมูล

การแปลงฮิสโตแกรมของภาพเป้าหมายซึ่งเป็นสตรีมข้อมูลให้เป็นจำนวนเต็มทำได้เช่นเดียวกับการแปลงฮิสโตแกรมของภาพโมเดลเพียงแต่เปลี่ยนจากสตรีมข้อมูลของภาพโมเดลให้เป็นภาพเป้าหมายเท่านั้น

7. การคำนวณอัตราส่วนของภาพเป้าหมายและภาพโมเดล (Normalization)

<u>คำอธิบาย</u> (เป็นส่วนหนึ่งของฟังก์ชัน OnToolMatching( ))

ส่วนนี้จะทำการคำนวณให้จำนวนพิกเซลรวมในฮิสโตแกรมของภาพโมเดลมีขนาดเท่ากับจำนวนพิกเซลรวมในฮิสโตแกรมของภาพเป้าหมาย โดยจำนวนพิกเซลรวมของภาพเป้าหมายได้จากสเตทเมนท์ Target_Size = boxWidth * boxHeight และต้องมีการเก็บพิกเซลรวมของภาพเป้าหมายไว้ในตัวแปร Old_Target เพื่อให้พิกเซลรวมของภาพเป้าหมายคงที่เสมอ

```
//DibView.cpp ฟังก์ชัน OnToolMatching( )
Target_Size = boxWidth * boxHeight;
Old_Target = Target_Size;
for (int k=0;k<=32;++k)
        {old_target[k] = target_buff[k];}
for (int i=1;i<=5;++i)
        {Model_Size = model_buff[i+26]+ Model_Size;}
```

for (int k=0;k<=32;++k) จะเป็นลูปที่ทำการเก็บฮิสโตแกรมของภาพเป้าหมายไว้เพื่อจะนำไปใช้ในการแปลงอัตราส่วนรอบต่อไป และ for (int i=1;i<=5;++i) จะเป็นลูปคำนวณจำนวนพิกเซลรวมของภาพโมเดล เก็บไว้ในตัวแปร Model_Size

```
        Top = max(Target_Size,Model_Size);
        Down = min(Target_Size,Model_Size);
```

```
Normalize = Top/Down;// always more than or equal 1
```

ส่วนนี้จะทำการหาค่าอัตราส่วนของภาพระหว่างภาพเป้าหมายและภาพโมเดล ค่าอัตราส่วนที่ได้จะมีค่ามากกว่า 1 เสมอ และเก็บค่าไว้ในตัวแปร Normalize

```
if (Normalize == 0.0)
        {Normalize = 1;}
if (Top == Model_Size) // model is bigger than target
        {  for (int i = 0;i <=32;++i)
                {target_buff[i] = target_buff[i]*Normalize;}
           Target_Size = Target_Size*Normalize;
           }
```

ถ้าอัตราส่วนของภาพเป็น 0 จะกำหนดให้เป็น 1 เพื่อป้องกันไม่ให้ผลคูณระหว่างฮิสโตแกรมของภาพและอัตราส่วนนี้ทำให้ฮิสโตแกรมเป็น 0 ทั้งหมด สเตทเมนท์ if (Top == Model_Size) จะตรวจสอบว่าภาพโมเดลมีขนาดใหญ่กว่าภาพเป้าหมาย จะทำการคูณอัตราส่วนของภาพกับฮิสโตแกรมของภาพเป้าหมายในแต่ละระดับจนครบทุกระดับ และคูณอัตราส่วนของภาพกับจำนวนพิกเซลรวมของภาพเป้าหมายด้วย

```
else  {
        for (int j = 0;j <=32;++j)
        { model_buff[j] = model_buff[j]*Normalize;}
        Model_Size = Model_Size*Normalize;
      }
```

แต่ถ้าภาพโมเดลมีขนาดเล็กกว่าภาพเป้าหมาย โปรแกรมจะมาทำงานที่สเตทเมนท์ else คือจะคูณอัตราส่วนของภาพกับฮิสโตแกรมของภาพโมเดลในแต่ละระดับจนครบทุกระดับ และคูณอัตราส่วนของภาพกับจำนวนพิกเซลรวมของภาพโมเดลด้วย

8. การสร้างโหนดเพื่อเก็บข้อมูลต่างๆ

<u>คำอธิบาย</u> (เป็นส่วนหนึ่งของฟังก์ชัน OnToolMatching( ))

เมื่อทำการคำนวณค่าความเหมือนระหว่างคู่ของฮิสโตแกรมแล้วจะเก็บข้อมูลต่างๆเช่น ค่าความเหมือนของคู่ฮิสโตแกรม ชื่อไฟล์ภาพโมเดล จำนวนพิกเซลรวมของภาพโมเดล ลงในโหนด เริ่มต้นจะทำการกำหนดให้เรดคอร์ด Result เป็น node, list_h, list_t และ insert เป็นพอยเตอร์ ได้โดย

```
//DibView.cpp ฟังก์ชัน OnToolMatching( )
Result *node = NULL, *list_h = NULL, *list_t = NULL , *insert = NULL;
```

และทำการสร้างโหนดได้โดยให้ node = new Result และเก็บข้อมูลต่างๆนี้ละฟิลด์ในโหนด และกำหนดให้ list_h และ list_t เป็นพอยเตอร์ให้ชี้อยู่ที่โหนดนี้ และให้ฟิลด์ next_down เป็นพอยเตอร์ชี้ไปที่ค่า NULL ซึ่งหมายถึงสุดลิสต์ และให้ฟิลด์ next_up เป็นพอยเตอร์ชี้ไปที่ NULL เช่นกัน

```
node = new Result;
node->filename = his_file.name;
node->avg = Ma;
node->H = Mh;
node->ModelSize = Model_Size;
list_h = node;
list_t = node;
list_t->next_down = NULL;
list_t->next_up = NULL;
```

ในโปรแกรมจะมีการแปลงสตรีมข้อมูลภาพโมเดลและการคำนวณอัตราส่วนของภาพโมเดลกับภาพเป้าหมาย 2 ครั้ง โดยที่การแปลงสตรีมข้อมูลภาพโมเดลจะมีการทำงานเหมือนกัน แต่การคำนวณอัตราส่วนของภาพจะต่างกันเล็กน้อยตรงที่จะมีการนำค่าของฮิสโตแกรมของภาพเป้าหมายเริ่มต้น (ซึ่งเก็บอยู่ในตัวแปร Old_target จากการแปลงอัตราส่วนรอบแรก) มาใช้ โดยคืนค่าให้กับ target_buff เพื่อให้ target_buff ใช้คำนวณหาอัตราส่วนของภาพต่อไป การคืนค่าทำได้โดยใช้ลูป for ดังนี้

```
for(int k=0;k>=32;++k)
  {target_buff[k] = old_target[k];}
```

9. การคำนวณค่าความเหมือน

ชื่อฟังก์ชัน HisIntersectH( )

คำอธิบาย

เป็นฟังก์ชันในการคำนวณค่าความเหมือน โดยใช้ค่าของระดับสี 18 ระดับ ฟังก์ชันทำการรับค่าอะเรย์ของข้อมูลชนิดสตริงของฮิสโตแกรมเฉพาะค่าของสีของภาพเป้าหมาย target_buff[19] และอะเรย์ของข้อมูลชนิดสตริงของฮิสโตแกรมเฉพาะค่าของสีของภาพโมเดล model_buff[19] ดังนี้

```
//DibView.cpp
float CDibView::HisIntersectH(double target_buff[19],double model_buff[19])
```

จากนั้นจะทำการวนลูปเพื่อคำนวณหาค่าความเหมือนตามสมการหาค่าความเหมือน โดย total_model จะเป็นจำนวนพิกเซลรวมของภาพโมเดล total_min จะเป็นจำนวนรวมของพิกเซลสีที่ภาพโมเดลและภาพเป้าหมายมีสีเหมือนกัน และ mini_val เป็นค่าที่น้อยที่สุดในแต่ละระดับของคู่ฮิสโตแกรม

```
while (order < 18)
```

```
{
        total_model = total_model + model_buff[++order]; // sum of model
        mini_val = min(target_buff[order],model_buff[order]); // min of model and target
        total_min = total_min + mini_val;   // total of model and target compare
}
```

ค่าความเหมือนค่าของสี ซึ่งเก็บอยู่ในตัวแปร Mh จะถูกส่งกลับไปยังฟังก์ชัน OnToolMatching( ) ด้วยคำสั่ง return

```
float Mh ;
Mh =  (float)total_min/(float)total_model;
return Mh;
```

10.การจัดลำดับโหนด

คำอธิบาย (เป็นส่วนหนึ่งของฟังก์ชัน OnToolMatching( ))

การจัดลำดับโหนดเพื่อใช้ในการแสดงผลภาพโมเดล 5 อันดับแรกที่มีความคล้ายกับภาพเป้าหมาย โดยจะใช้ค่าความเหมือนของสีในการจัดลำดับ ในการจัดลำดับจะรับโหนดที่มีการกำหนดค่าต่างๆ เรียบร้อยแล้ว และเพิ่มเข้าไปในลิสต์ เริ่มต้นจะกำหนดให้พอยเตอร์ insert และ list_t มีค่าเท่ากับ list_h ซึ่งเป็นการกำหนดให้พอยเตอร์ insert และ list_t ชี้ไปที่โหนดแรกของลิสต์

```
//DibView.cpp ฟังก์ชัน OnToolMatching( )
insert = list_h;
list_t = list_h;
node = new Result; // create a new node
```

ลูป while(insert) จะทำการวนลูปเพื่อหาตำแหน่งของโหนดที่เหมาะสมในลิสต์ ซึ่งจะเป็นตำแหน่งที่จะทำให้ลิสต์เรียงลำดับตามค่าความเหมือนของสี และจะหยุดวนลูปก็ต่อเมื่อโหนดได้ทำการเพิ่มเข้าไปในลิสต์แล้ว

```
while(insert)
{
        pot = insert->H ;
        list_t = insert;
        if (Mh  >  pot) // top
        {
```

ตัวแปร pot จะเก็บค่าของสีจากโหนดในลิสต์ โดยใช้พอยเตอร์ insert เป็นตัวชี้ และเอามาเปรียบเทียบกับ ค่า Mh ที่ได้จากการคำนวณค่าความเหมือนของคู่ฮิสโตแกรม สเตทเมนท์ if (Mh > pot) จะทำการเปรียบเทียบค่าความเหมือนที่ได้จากการคำนวณมีค่ามากกว่าค่าความเหมือนในลิสต์ จะลงมาทำสเตทเมนท์ต่อไปคือ

```
if (insert->next_up == NULL) // one child
```

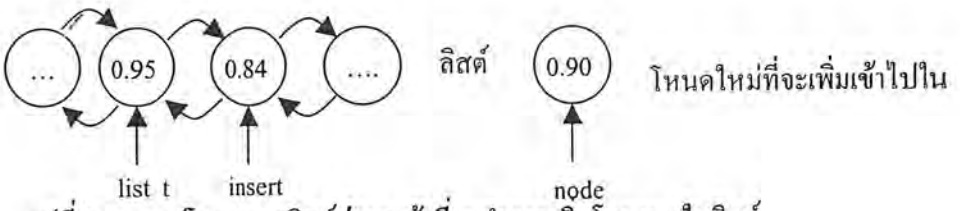
ซึ่งจะทำการตรวจสอบว่า มีโหนดก่อนหน้าโหนดที่พอยเตอร์ insert ชี้อยู่ แล้วจะลงมาทำสเตทเมนท์

```
{
    list_h = node;
    node->next_down = insert;
    node->next_up = NULL;
    insert->next_up = node;
    insert = NULL;
}
```

เป็นสเตทเมนท์ที่ทำการเพิ่มโหนดเข้าไปหน้าลิสต์ และกำหนดให้โหนดที่เพิ่มเข้าใหม่นี้เป็นหัวของลิสต์ ในสเตทเมนท์นี้จะมีพอยเตอร์ที่สำคัญ 2 พอยเตอร์คือ พอยเตอร์ node และ พอยเตอร์ insert ซึ่งพอยเตอร์ node จะชี้โหนดที่จะเพิ่มเข้าไปในลิสต์และพอยเตอร์ insert จะชี้ที่โหนดเริ่มต้นของลิสต์ ฟิลด์ next_down ของโหนดที่เพิ่มเข้าใหม่นี้จะชี้ไปที่โหนดแรกในลิสต์ที่พอยเตอร์ insert ชี้อยู่ ฟิลด์ next_up ของโหนดใหม่จะชี้ไปที่ NULL แสดงว่าโหนดที่เพิ่มเข้าใหม่นี้เป็นโหนดเริ่มต้น และกำหนดให้โหนดที่ insert ชี้อยู่เป็น NULL

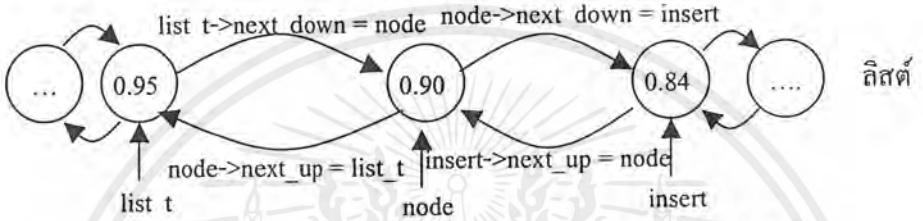
```
else  // there is node up and down child
{
        list_t = insert->next_up;
        list_t->next_down = node;
        node->next_up = list_t;
        insert->next_up = node;
        node->next_down = insert;
}
insert = NULL;
```

สเตทเมนท์ else จะทำการตรวจสอบว่าโหนดที่เพิ่มเข้ามาใหม่นี้อยู่ระหว่างโหนด 2 โหนดในลิสต์ การเพิ่มโหนดใหม่ที่เข้ามาในลิสต์นี้แสดงได้เป็นขั้นตอนดังรูปที่ ๑ฉ.



รูปที่ 1ฉ. แสดงโหนดและลิสต์ก่อนหน้าที่จะทำการเพิ่มโหนดลงในลิสต์

พอยเตอร์ที่สำคัญจะมีอยู่ 3 พอยเตอร์คือ พอยเตอร์ list_t, พอยเตอร์ insert และพอยเตอร์ node ซึ่งจะชี้ที่โหนดดังรูป จะทำการเพิ่มโหนดใหม่ระหว่างโหนดที่พอยเตอร์ list_t และพอยเตอร์ insert ชี้อยู่ ดังรูปที่ 2ฉ.



รูปที่ 2ฉ. แสดงลิสต์หลังจากที่ได้เพิ่มโหนดเข้าไปในลิสต์แล้ว

```
else if (insert->next_down == NULL)  //end of list  // rear
{
        node->next_up = insert;
        insert->next_down = node;
        node->next_down = NULL;
        insert = NULL;
} // end else if
```

สเตทเมนท์ else if จะตรวจสอบว่าโหนดที่จะเพิ่มเข้ามาใหม่จะเป็นโหนดที่มาต่อท้ายลิสต์ พอยเตอร์ที่สำคัญมีอยู่ 2 พอยเตอร์คือ พอยเตอร์ node และพอยเตอร์ insert โดยที่พอยเตอร์โหนดจะชี้โหนดที่จะเพิ่มเข้ามาใหม่และพอยเตอร์ insert จะชี้ที่โหนดสุดท้ายของลิสต์ การเพิ่มโหนดที่ท้ายลิสต์จะให้ฟิลด์ next_up ของโหนดที่เพิ่มเข้ามาชี้ไปที่โหนดท้ายสุดของลิสต์และ ให้ฟิลด์ next_down ของโหนดสุดท้ายของลิสต์ ชี้ไปที่โหนดที่เพิ่มเข้าใหม่และกำหนดให้โหนดที่เข้ามาใหม่นี้เป็นโหนดท้ายสุดด้วยสเตทเมนท์ node->next_down = NULL

แต่ถ้าโหนดยังไม่อยู่ในตำแหน่งที่จะเพิ่มเข้าไปในลิสต์ได้ จะค้นหาโหนดต่อไปเรื่อยในลิสต์ตามสเตทเมนท์ดังนี้

```
else { insert = insert->next_down ;}
```

เมื่อเพิ่มโหนดใหม่เข้าไปในลิสต์แล้ว จะทำการเพิ่มข้อมูลเข้าไปในโหนดใหม่นี้ โดยข้อมูลที่เพิ่มจะมี ชื่อไฟล์ภาพโมเดล ค่าความเหมือนของสี และจำนวนพิกเซลทั้งหมดของภาพโมเดล

```
node->filename = his_file.name;
node->H = Mh;
node->ModelSize = Model_Size;
```

สุดท้ายจะได้ลิสต์ที่เรียงลำดับตามค่าความเหมือนจากมากไปหาน้อย

ภาคผนวก ช.
การทำงานของโปรแกรมในส่วนของการติดต่อกับผู้ใช้

1 การกำหนดลำดับของ tool ต่างๆ ที่ปรากฏใน tool bar

ชื่อฟังก์ชัน _CODE buttons[ ]

คำอธิบาย

ลำดับจะเรียงตาม ID ของ tool และ ID_SEPARATOR ซึ่งเป็นตัวเว้นระยะตามที่กำหนด

```
// Mainfrm.cpp
static UINT _CODE buttons[ ] =
{
        // same order as in the bitmap 'toolbar.bmp'
        ID_FILE_OPEN,
                ID_SEPARATOR,
                ID_SEPARATOR,
        ID_FILE_PRINT,
                ID_SEPARATOR,
                ID_SEPARATOR,
        ID_TOOL_MATCHING,
                ID_SEPARATOR,
                ID_SEPARATOR,
        ID_OPTION_SETBOXCOLOR,
                ID_SEPARATOR,
                ID_SEPARATOR,
        ID_APP_ABOUT,
};
```

tool ต่างๆ จะปรากฏตามลำดับดังแสดงในรูปที่ 1ช.

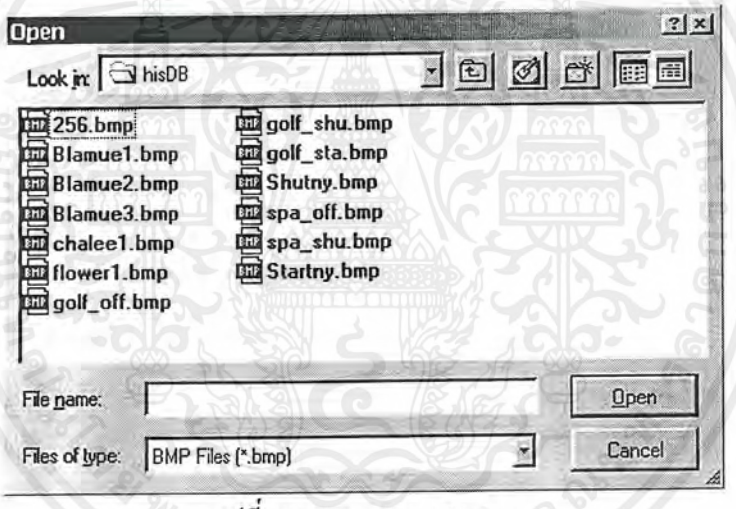
รูปที่ 1ช. แสดง tool bar ที่ปรากฏในโปรแกรม

2. การกำหนดให้ประเภทของไฟล์ที่แสดงใน Open dialog เป็นบิทแมพ

คำอธิบาย

```
// DibLook.rc
STRINGTABLE PRELOAD DISCARDABLE
BEGIN
        ...
        IDR_DIBTYPE   "\nDib\nDIBLOOK Document \nBMP Files (*.bmp) \n.bmp\nDibFileType \
                      NDIBLOOK File Type \nDIB \nDevice-Independent Bitmap \n"
        ...
END
```

ในการกำหนดประเภทไฟล์ที่เปิดสามารถเข้าไปกำหนดได้โดยเข้าไปแก้ไขใน string table ของ program resource โดยเมื่อใช้คำสั่งเปิดไฟล์ จะปรากฏ Open Dialog ดังแสดงในรูปที่ 2ช. ซึ่งจะเห็นว่าจะมีไฟล์ประเภทบิทแมพเป็น default



รูปที่ 2ช. แสดง Open Dialog

3 การเรียก Color Dialog เพื่อเลือกสีของกรอบที่จะลาก

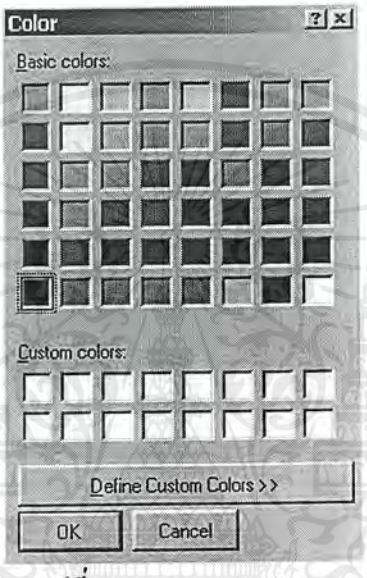
ชื่อฟังก์ชัน OnOptionSetboxcolor( )

คำอธิบาย

```
// DibView.cpp
void CDibView::OnOptionSetboxcolor()
{
        CColorDialog dlgColor(boxColor);
        if (dlgColor.DoModal() == IDOK)
        {
```

```
        boxColor = dlgColor.GetColor();
        Invalidate();
    }
}
```

การเรียก color dialog จะต้องกำหนดตัวแปรประเภท CColorDialog ก่อน โดยในฟังก์ชันที่แสดงใช้ตัวแปร dlgColor ซึ่งตัวแปรนี้จะสามารถทำงานต่างๆ ได้ตาม class ที่สืบทอดมา เมื่อเรียกฟังก์ชัน DoModal ของ dlgColor จะแสดง color dialog ออกมาดังรูปที่ 3ช. และจะเปลี่ยนค่าของตัวแปรที่เก็บค่าสี (COLORREF boxColor) ให้เป็นไปตามที่เลือกไว้ใน color dialog เมื่อมีการส่งค่าของ IDOK กลับมา



รูปที่ 3ช. แสดง color dialog

4 การกำหนดการเก็บภาพลงในแฟ้มข้อมูล

ชื่อฟังก์ชัน OnToolDatabase( ), OnUpdateToolDatabase( )

คำอธิบาย

```
// DibView.cpp
void CDibView::OnToolDatabase()
{
        m_database = !m_database;
}
```

การเก็บภาพลงแฟ้มข้อมูลจะเช็คจากตัวแปรที่เป็นบูลีนคือ m_database ซึ่งตัวแปรจะเปลี่ยนแปลงค่าโดยการกำหนดให้ m_database = !m_database เมื่อกดที่คำสั่ง Save to database บนเมนูบาร์

```
void CDibView::OnUpdateToolDatabase(CCmdUI* pCmdUI)
{
```

pCmdUI->SetCheck(m_database);

}

ภาพจะถูกเก็บลงแฟ้มข้อมูลเมื่อมีเครื่องหมายถูกแสดงอยู่ด้านหน้าของคำสั่ง Save to directory ดังแสดงในรูปที่ 4ช.

รูปที่ 4ช. แสดงเมนูการเลือกเก็บภาพลงในแฟ้มข้อมูล

5 การแสดงภาพผลลัพธ์ที่ได้

<u>คำอธิบาย</u> (เป็นส่วนหนึ่งของฟังก์ชัน OnToolMatching( ) ในไฟล์ DibView.cpp)

ภาพผลลัพธ์ที่ได้จะถูกเก็บชื่อไฟล์ไว้โดยเรียงลำดับตามความเหมือนที่ได้ โดยมีตัวแปรที่สำคัญ คือ times ใช้นับจำนวนภาพที่เก็บไว้ในลิสต์เพื่อทำการแสดงภาพแต่ละครั้ง, ตัวแปร reset ใช้ในการเคลียร์ค่าของ result dialog และตัวแปร Dlg ซึ่งใช้อ้างถึง CResultDlg

int times = 0;

BOOL reset;

CResultDlg Dlg;

reset = FALSE;

เมื่อมีภาพที่ต้องการแสดงผลอยู่ในลิสต์ (list_h != NULL) ก็จะทำการกำหนดค่าต่างๆเพื่อส่งให้กับ result dialog โดยให้ลิสต์ชี้ไปยังโหนด (node) ของข้อมูลที่ต้องการต่างๆและเก็บค่าเหล่านั้นไว้ในตัวแปร การเก็บค่าจะทำไปเรื่อยๆจนกว่าจะพบภาพที่มีค่าความเหมือนที่คำนวณได้น้อยกว่าค่าความเหมือนเฉลี่ยที่กำหนด ก็จะกำหนดให้ list_h เป็น null เพื่อออกจากลูป

while (list_h)

```
{
        node = list_h;
        list_h = list_h->next_down;
        CString n = node->filename + " ";
        match = node->H;
        info = node->DATA;

        if ( match < Tthres)        list_h = NULL;
        else
        ...
}
```

เพื่อที่จะสามารถแสดง result dialog ของภาพได้พร้อมกันทีละหลายๆ ภาพจึงทำการวนลูปเรียกฟังก์ชัน InitPath( ) โดยจะเพิ่มตัวแปร times ไปเรื่อยๆตามจำนวนครั้งที่วนลูป และหากมีการเคลียร์ค่าของ result dialog แล้วจะต้องเซ็ทค่า reset ให้เป็นเท็จใหม่ด้วย เมื่อมีการ InitPath( ) จะเซ็ทให้ตัวแปร firsttime มีค่าเป็นจริง เพื่อใช้ในการเช็คว่ามีภาพที่สามารถนำมาแสดงผลได้หรือไม่ ถ้าไม่มีให้แสดงข้อความแจ้งว่าไม่พบภาพที่ต้องการ และเซ็ทค่า list_h เป็น null เพื่อที่จะหลุดจากลูปและจบการแสดง result dialog

```
while (list_h)
{
        ...
        times += 1;
        Dlg.InitPath(fileResult,n,match,info,reset);
        if (reset == TRUE)          reset = FALSE;
        firsttime = true ;
        ...
        if (firsttime == false)
        {
                AfxMessageBox("NO picture found");
                list_h = NULL;
        }
        else
        ...
```

ในการวนลูปทุกๆ 5 ครั้ง จะเรียกแสดงผลของ result dialog และการแสดงผลแต่ละครั้งจะต้องเซ็ทค่า reset ให้เป็นจริงเพื่อเคลียร์ค่าเดิมทิ้งในการเรียก InitPath( ) ครั้งต่อไป เมื่อมีการเรียกแสดง result dialog โดยใช้ฟังก์ชัน DoModal( ) จะมีการส่งค่าคืนกลับมา ถ้าค่าที่คืนกลับมาเป็น IDCANCEL ก็จะต้อง

แสดงภาพผลลัพธ์ก่อนหน้านี้ 5 ภาพ โดยใช้ตัวแปร times เป็นตัวเช็คลำดับของภาพที่แสดงในไดอะล็อก ว่าเมื่อกดปุ่ม "BACK" แล้วต้องใช้พอยน์เตอร์ชี้กลับไปที่ใด

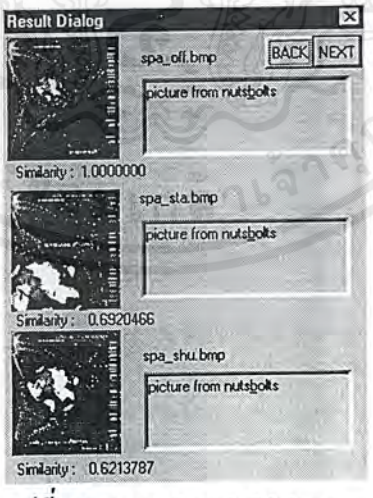
```
if (times % 5 == 0)
{       reset = TRUE;
        // IDOK = NEXT, IDCANCEL = BACK
        if (Dlg.DoModal() == IDCANCEL)
        {       if (times == 5)
                { list_h = NULL;}
                else
                {       if(list_h == NULL)
                        {       list_h = node;
                                times--;
                                for (int k=1;k<10;k++)
                                {       list_h = list_h->next_up;
                                        times--;
                                }
                        }
                        else
                        {       for (int i=1;i<11;i++)
                                {       list_h = list_h->next_up;
                                        times--;
                                }
                        }
                }
        }// end of clicked back
}// end if (times % 5 == 0)
```

ในการแสดงภาพบนไดอะล็อกอาจมีจำนวนของภาพที่ต้องแสดงในแต่ละครั้งไม่ครบ 5 ภาพ ก็ได้ ดังนั้นเราจึงต้องเขียนการทำงานส่วนนี้ ซึ่งจะทำงานก็ต่อเมื่อมีการ InitPath( ) ครบจน list_h เป็น null แล้ว

```
...
else {  if (list_h == NULL) // not divide by 5
        {       if (times < 5) //picture found less than 5
                {       Dlg.DoModal();
```

```
            reset = TRUE;
        }
        if (times > 5 && times % 5 != 0)
        {       reset = TRUE;
                if (Dlg.DoModal() == IDCANCEL)
                {       // it's next node that not show
                        list_h = node;
                        int m = times % 5;
                        for (int j=0;j < m;j++)
                        {       list_h = list_h->next_up;
                                times--;
                        }
                        for (int i=0;i<5;i++)
                        {       list_h = list_h->next_up;
                                times--;
                        }
                } // end of clicked back
        }
```

ผลลัพธ์ที่ได้จากการทำงานของโปรแกรมจะแสดง ดังรูปที่ 5ช. ซึ่งเป็นส่วนหนึ่งของ result dialog ในการทำงานจริงจะแสดงครั้งละ 5 ภาพ



รูปที่ 5ช. แสดง Result Dialog

6 การเก็บค่าต่างๆ ของภาพที่จะแสดงใน Result dialog

ชื่อฟังก์ชัน InitPath( )

คำอธิบาย

ฟังก์ชันนี้จะรับ path ของไฟล์, ชื่อไฟล์, ค่าความเหมือน, รายละเอียดของภาพนั้นมาเก็บไว้ เพื่อนำไปแสดงผลเมื่อมีการเรียกฟังก์ชัน OnPaint( ) และรับตัวแปร reset มาเพื่อเช็คการเคลียร์ค่าของตัวแปร no ที่เป็นเหมือนตัวแทนของภาพแต่ละภาพที่จะแสดงผลออกมา

```
// ResultDlg.cpp
void CResultDlg::InitPath(CString filepath,CString name, float match, CString detail, BOOL reset)
{
        BOOL clear;
        clear = reset;
        if (clear == TRUE)
                no = 0;

        no += 1;
        if (no == 1)
        {
        // initial value
        m_hDIB2 = NULL;                 m_hDIB3 = NULL;                 m_hDIB4 = NULL;
        m_hDIB5 = NULL;

        m_palDIB2 = NULL;       m_palDIB3 = NULL;       m_palDIB4 = NULL;
        m_palDIB5 = NULL;

        // read file and copy handle
        CFile fileshow(filepath,CFile::modeRead);
                m_hDIB1 = ::ReadDIBFile(fileshow);
        fileshow.Close();
                m_palDIB1 = new CPalette;
                ::CreateDIBPalette(m_hDIB1, m_palDIB1);

                m_matching1 = Convert(match);
                m_name1 = name;
                m_data1 = detail;
        }

        if (no == 2)
```

```
{
// read file and copy handle
CFile fileshow(filepath,CFile::modeRead);
        m_hDIB2 = ::ReadDIBFile(fileshow);
fileshow.Close();
        m_palDIB2 = new CPalette;
        ::CreateDIBPalette(m_hDIB2, m_palDIB2);
        m_matching2 = Convert(match);
        m_name2 = name;
}

if (no == 3)
{
        ...
}

...
}
CFile fileshow(filepath,CFile::modeRead);
        m_hDIB1 = ::ReadDIBFile(fileshow);
fileshow.Close();
        m_palDIB1 = new CPalette;
        ::CreateDIBPalette(m_hDIB1, m_palDIB1);

        m_matching1 = Convert(match);
        m_name1 = name;
```

ใน Class ResultDlg จะมีตัวแปร no เป็นตัวนับ เพื่อที่จะสามารถเก็บ handle และข้อมูลต่างๆ ของแต่ละภาพแยกกันได้ ตามจำนวนครั้งของการวนลูป โดยในการเก็บข้อมูลครั้งแรก (no = 1) จะทำการกำหนดค่าเริ่มต้นของ handle และ palette ต่างๆ ของภาพให้เป็น null นอกนั้นจะมีการทำงานที่เหมือนกัน คือ นำตัวแปร filepath ที่รับมาใช้เป็นตัวบอกค่า path และชื่อไฟล์ที่ทำการอ่านและเก็บไฟล์ที่อ่านได้ในตัวแปร fileshow เพื่อนำไปใช้ในการเรียกฟังก์ชัน ReadDIBFile( ) เพื่อนำ handle ของภาพมาเก็บไว้ในตัวแปร m_hDIB เมื่อทำงานกับไฟล์เสร็จแล้วจะต้องปิดไฟล์เสมอโดยการเรียกฟังก์ชัน Close( ) ของไฟล์ จากนั้นกำหนด palette สำหรับแสดงภาพโดยใช้คำสั่ง new CPalette และสร้าง palette สำหรับภาพที่จะ

แสดงโดยใช้ฟังก์ชัน CreateDIBPalette( ) นำค่าความเหมือนของภาพมาเปลี่ยนจาก integer ให้กลายเป็น string เพื่อที่จะสามารถนำมาแสดงผลบนหน้าจอได้โดยใช้ฟังก์ชัน Convert( ) เก็บค่าที่ได้ในตัวแปร m_matching ซึ่งเป็นตัวแปรที่ผูกค่ากับ IDC_SIMVALUE ของไดอะล็อก (การผูกค่าของตัวแปรกับไดอะล็อกทำโดยการเข้าไปกำหนดตัวแปรใน member variable ของคลาสไดอะล็อกนั้นโดยใช้ class wizard

7 การแสดงผลของ result dialog

ชื่อฟังก์ชัน PaintDIB( )

คำอธิบาย

การแสดงผลของไดอะล็อกทำโดยเช็คจำนวนของภาพที่ต้องวาดโดยดูจากค่าของตัวแปร no และวาดโดยใช้ฟังก์ชัน PaintDIB( )

```
// ResultDlg.cpp
void CResultDlg::OnPaint()
{
        // device context for painting
        CPaintDC dc(this);

        if (no > 0)
        {
                // define image rectangle
                LPSTR lpDIB1 = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB1);
                int cxDIB = (int) ::DIBWidth(lpDIB1);
                int cyDIB = (int) ::DIBHeight(lpDIB1);
                ::GlobalUnlock((HGLOBAL) m_hDIB1);
                CRect rcDIB1;
                rcDIB1.top = rcDIB1.left = 0;
                rcDIB1.right = cxDIB;
                rcDIB1.bottom = cyDIB;

                // define display area rectangle
                CRect rcDest1;
                rcDest1.top = rcDest1.left = 5;
                rcDest1.right = 90;
                rcDest1.bottom = 100;
```

```
            // paint image to defined area
            ::PaintDIB(dc, &rcDest1, m_hDIB1,
            &rcDIB1, m_palDIB1);
    }

    if (no > 1)
    {
            // define image rectangle
            LPSTR lpDIB2 = (LPSTR) ::GlobalLock((HGLOBAL) m_hDIB2);
            ...
            // define display area rectangle
            CRect rcDest2;
            rcDest2.top = 120;
            rcDest2.left = 5;
            rcDest2.right = 90;
            rcDest2.bottom = 215;

            // paint image to defined area
            ::PaintDIB(dc, &rcDest2, m_hDIB2,
            &rcDIB2, m_palDIB2);
    }
    ...
}
```

บรรณานุกรม

Rafael C. Gonzalez and Richard E. Woods,

Digital Image Processing, Addison Wesley หน้า 191-195

John C. Russ,

The Image Processing Handbook, CRC press หน้า 41-42

John R. Smith,

Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression,

Columbia University,

1997, บทที่ 2 หน้า 11-13, 15-17

John R. Smith and Shin-Fu Chang,

Automated Image Retrieval Using Color and Texture,

Columbia University technical Report,

TR# 414-95-20, July, 1995

John R. Smith and Shin-Fu Chang,

Tools and Techniques use for color Image Retrieval,

Columbia University, หน้า 4-6

Michael J. Swain and Dand H. Ballard,

Color Indexing,

International Journal of Computer Vision, หน้า 14-20

Greg Pass and Ramin Zabih,

Comparing Images Using Joint Histograms,

Computer Science Department of Cornell University,หน้า 3-11

Markus Stricker and Michale Swain

Capacity of Color Histogram Indexing

Jai Wang, Wen-jann Yang* and Raj Acharya

Color Clustering Techniques for Color-Content-Based Image Retrieval from Image Database,

Santhana Krishnamachari and Mohamed Abdel-Mottaleb

A Scalable Algorithm for Image Retrieval by Color