การค้นหาข้อมูลภาพบนอินเตอร์เน็ตด้วยวิธีการวิเคราะห์เนื้อหาและคำอธิบาย

Image Retrieval on Internet Using Metadata and Content Analysis Method

นางสาวสาวิตี โคเรียน รหัส 45360492

นายธีระพงค์ หมั่นคิด รหัส 45380057

นายชุมพล สมัครการ รหัส 45380192

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ใบรับรองโครงงานวิจัยวิศวกรรม

หัวข้อโครงงาน	การค้นหาข้อมูลภาพบนอินเตอร์เน็ตด้วยวิธีการวิเคราะห์เนื้อหาและคำอธิบาย
ผู้ดำเนินโครงงาน	นางสาวสาวิตี โคเรียน รหัส 45360492
	นายธีระพงค์ หมั่นคิด รหัส 45380057
	นายชุมพล สมัครการ รหัส 45380192
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. สุชาติ แย้มเม่น
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2548

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. สุชาติ แย้มเม่น)

.....กรรมการ
(ดร.ไพศาล มุณีสว่าง)

.....กรรมการ
(ดร.พนมขวัญ ริยะมงคล)

.....กรรมการ
(ดร.สุรเชษฐ์ กานต์ประชา)

หัวข้อโครงงาน	การค้นหาข้อมูลภาพบนอินเตอร์เน็ตด้วยวิธีการวิเคราะห์เนื้อหาและคำอธิบาย		
ผู้ดำเนินโครงงาน	นางสาวสาวิตี	โคเรียน	รหัส 45360492
	นายธีระพงค์	หมั่นคิด	รหัส 45380057
	นายชุมพล	สมัครการ	รหัส 45380192
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. สุชาติ		แย้มเม่น
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2548		

บทคัดย่อ

โครงงานนี้ศึกษาและพัฒนาโปรแกรมการค้นหาข้อมูลรูปภาพบนอินเตอร์เน็ต โดยจะมีสองส่วนหลักๆ ด้วยกัน ในส่วนแรกจะใช้หลักการ Meta data Search ซึ่งในส่วนนี้จะทำการร้องขอบริการจาก Google Search Engine โดยให้ไปค้นหารูปภาพที่มีความคล้ายคลึงกับคำอธิบายที่ทำการร้องขอไป และในส่วนที่สองเป็นการค้นหาข้อมูลรูปภาพโดยใช้หลักการ Content-base image retrieval ซึ่งใช้ระบบสี RGB และHSV แล้วเก็บค่าองค์ประกอบของสีทั้งสองระบบนั้นลงในฐานข้อมูล เมื่อเราทำการค้นหาจะนำค่าองค์ประกอบสี RGBและHSV มาทำการเปรียบเทียบหาค่าผลต่างและทำการเรียงลำดับค่าผลต่างจากน้อยไปมาก ซึ่งวิธีการค้นหาแบ่งออกเป็น 2 วิธีด้วยกัน คือ ค้นหาโดยเปรียบเทียบค่าองค์ประกอบสี RGB และค้นหาโดยเปรียบเทียบค่าองค์ประกอบสี HSV และแนวทางการค้นหาจะประกอบด้วย การค้นหาจากกลุ่มเดียวกัน, การค้นหาจากประเภทเดียวกัน, การค้นหาจากรูปภาพทั้งหมด โดยโปรแกรมที่พัฒนานี้จะใช้ C# Window Application ของ Microsoft visual studio.NET 2005 และมีการจัดการฐานข้อมูลโดยใช้ Microsoft Access 2003 เป็นเครื่องมือในการจัดการ

Project title	Image Retrieval on Internet Using Metadata and Content Analysis Method
Name	Miss Sawitee Korean ID. 45360492
	Mr. Theeraphong Mankhit ID. 45380057
	Mr. Chumphol Samakran ID. 45380192
Project advisor	Assistant Professor Suchart Yammen , Ph.D.
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic year	2005

.....

Abstract

The project studies and develops metadata/content-based search engine for searching digital images in distributed databases on internet. The system consists of two parts. First, we use metadata search technique to call service from Google search engine to find images on internet by matching the metadata (keywords) of the request query and then pulling relevant images into a local computer. Second, we apply content-based image retrieval technique to the pulled images using color descriptors conducted in RGB and HSV color spaces. These spaces are quantized with some suitable levels to reduced complexity of the content matching. The proposed search engine offers two query interface methods which are search-by-RGB Color Model and search-by-HSV Color Model. The system also offers threes search options: search within group, search within class, and search for all images in the achieved database. We implement the system by C# Window Application in Microsoft Visual Studio .NET 2005 platform and use Microsoft Access 2003 for database management system.

กิตติกรรมประกาศ

ในการทำโครงงานวิศวกรรมครั้งนี้ คณะผู้จัดทำขอกราบขอบพระคุณ ดร.ไพศาล มุณีสว่าง ที่ได้ให้คำปรึกษาโครงงานนี้ ทั้งทฤษฎีและขั้นตอนการปฏิบัติงานต่างๆ และขอกราบขอบพระคุณ ผศ.ดร.สุชาติ แย้มเม่นที่ให้คำแนะนำและติชมเพื่อแก้ไขในส่วนของโปรแกรมและรายงานโครงงาน ขอกราบขอบพระคุณ ดร.พนมขวัญ ริยะมงคล และดร.สุรเชษฐ์ กานต์ประชา ที่ได้เสียสละเวลาเพื่อทำการตรวจสอบการทำงานและชี้แนวทางในการแก้ไขปัญหาโครงงานนี้

และทางคณะผู้จัดทำใคร่กราบขออภัยบุคคลที่มีส่วนเกี่ยวข้องในการทำโครงงานนี้แต่ไม่ได้กล่าวนามและใคร่ขอขอบพระคุณ มา ณ ที่นี้ด้วย

นางสาวสาวิตี โคเรียน
นายธีระพงค์ หมั่นคิด
นายชุมพล สมัครการ

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาและความและความสำคัญของโครงงาน

ในปัจจุบันเทคโนโลยีอินเตอร์เน็ต ได้เข้ามามีบทบาทกับมนุษย์มากขึ้น เพราะในอินเตอร์เน็ต ได้รวบรวมข่าวสารไว้อย่างมากมาย จึงมีผู้สนใจและใช้บริการเป็นจำนวนมาก เนื่องจากเป็นการใช้ที่ไม่จำกัดการใช้และไม่จำกัดสถานที่ และค่าใช้จ่ายน้อยเมื่อเทียบกับสื่ออื่นๆ ผู้ใช้สามารถเข้าไปค้นหาข้อมูลต่างๆ ที่ต้องการได้ ไม่ว่าจะเป็น ทางด้านการศึกษา การบันเทิง ข่าวสาร และบริการอื่นๆ อีกมากมาย รวมทั้งการค้นหาข้อมูลภาพด้วย และในโลกของ Internet มีข้อมูลมากมาย ถ้าจะใช้เวลาในการอ่านทุกสิ่งบน Internet คงต้องใช้เวลาหลายชั่วอายุคน จริงแล้วเราคงไม่มีความสนใจในทุกเรื่อง แต่คงสนใจเฉพาะเรื่องที่เราสนใจเท่านั้น และการค้นหาข้อมูลที่รู้จัก มี 2 วิธี คือ

1.1.1 การค้นหาในรูปแบบ Index Directory

ซึ่งข้อมูลจะถูกแบ่งออกเป็นหมวดหมู่ และจัดแบ่งแยก Site ต่างๆออก เป็นประเภท สำหรับวิธีใช้งาน สามารถที่จะเลือกประเภทเลือกข้อมูลตามที่ต้องการจะดูได้เลยใน Web Browser และเว็บไซต์ที่แสดงออกมานั้นทางผู้ให้บริการยังได้เรียบเรียงโดยนำเอา เว็บไซต์ ที่มีความเกี่ยวข้องมากที่สุดเอามาไว้ตอนบนสุดของรายชื่อที่แสดง

1.1.2 การค้นหาในรูปแบบ Search Engine

ซึ่งเป็นวิธีการที่นิยมใช้ในการค้นหาข้อมูล เป็นส่วนใหญ่กว่า 70% จะใช้วิธีการค้นหานี้ หลักการทำงานของ Search Engine จะเป็นฐานข้อมูลขนาดใหญ่มหาศาลที่กระจัดกระจายอยู่ทั่วไปบน Internet ไม่มีการแสดงข้อมูลออกมาเป็นลำดับขั้นของความสำคัญ การใช้งานจะเหมือนการสืบค้นฐานข้อมูล อื่นๆคือ จะต้องพิมพ์คำสำคัญ (Keyword) ซึ่งเป็นการอธิบายถึงข้อมูลที่ต้องการค้นหานั้นๆเข้าไปจากนั้น Search Engine ก็จะแสดงข้อมูลและ เว็บไซต์ ต่างๆที่เกี่ยวข้องออกมา

ซึ่งการทำโครงงานนี้เราจะใช้ความสามรถของการ Search แบบ Search Engine เข้ามาช่วยในการค้นหารูปภาพ

ในการค้นหารูปภาพ เพื่อสนองตอบความต้องการของผู้ใช้ โดยทั่วไปแล้วไม่ว่าจะค้นหาจากเน็ตเวิร์ค(Network) เช่นอินเตอร์เน็ต (Internet) หรือจาก ฐานข้อมูล (Database) ผู้ใช้ส่วนมากจะนิยมใช้การ search หาข้อมูลโดยผ่านทางเว็บไซต์ต่างๆที่ให้บริการค้นหาข้อมูล และเว็บที่ให้บริการค้นหาข้อมูลที่เรารู้จักกันดีก็คือ Google ซึ่งจะเห็นว่ามีความสะดวกสบายและง่ายสำหรับการหา แต่แค่เพียงลำพังของการทำงานของ Google นั้นก็ยังไม่มีประสิทธิภาพเพียงพอต่อการค้นหา เพราะในบางครั้งรูปภาพที่ได้มาได้อาจไม่ตรงตามความต้องการของเราเอง เพราะการทำงานของการ search

engine ใน Google นั้นเป็นการค้นหาข้อมูลจาก Keyword จึงอาจทำให้ได้ภาพที่ไม่ต้องการได้ หากภาพนั้นมีคำอธิบายตรงกับ Keyword ที่เราต้องการ ไม่ว่าภาพนั้นๆ ที่จริงแล้วจะเป็นภาพอะไรก็ตาม

จากปัญหาที่พบดังกล่าว ดังนั้นจึงได้มีการนำเอาความรู้ทางด้าน Image Processing เข้ามาช่วยลดความล่าช้าในการเข้าถึงข้อมูลและให้ได้มาซึ่งข้อมูลที่ตรงกับที่ต้องการ โดยจะอาศัยหลักการของ Content-based image retrieval ในแบบของการค้นหาของ Google search engine ซึ่งหลักการค้นหา (search) ดังกล่าวได้พูดถึงไปแล้วในส่วนก่อนหน้าเข้ามาช่วยในการค้นหาข้อมูลเพื่อจะนำไปใช้ในการทำการประมวลผลภาพในลักษณะการนำเอาภาพที่ได้จาก Google search engine มาทำการวิเคราะห์และจัดเรียงความสำคัญตามความต้องการของผู้ใช้โดยจะนำเอาความรู้ทางด้านการวิเคราะห์ค่าองค์ประกอบสี (Color Histogram) เข้ามาช่วยในการแยกความต่างของภาพโดยการแยกความแตกต่างของสี ให้ออกมาอยู่ในรูปของเวกเตอร์ (Vector) เพื่อทำการหาผลต่างขององค์ประกอบสีของรูปภาพที่ผู้ใช้ต้องการหรือภาพต้นแบบกับภาพที่จะนำมาเปรียบเทียบ

ซึ่งที่ได้กล่าวมาขั้นต้นนั้นจะทำให้ผู้ใช้งานสามารถที่จะค้นหาข้อมูลรูปภาพตามที่ต้องการจากระบบเครือข่ายอินเตอร์เน็ต (Internet) ได้ง่ายและรวดเร็วขึ้นและได้ภาพที่ตรงตามความต้องการมากที่สุด

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 ศึกษาการใช้ภาษา Microsoft Visual C#.NET ในการพัฒนาโปรแกรมประเภท Search Engine และ GUI (Graphic User Interface) บนระบบเครือข่ายอินเตอร์เน็ต โดยอาศัย Google เป็น Web-Base Application ซึ่งรองรับการใช้งานจากผู้ใช้หลายคน (Multi user)

1.2.2 ใช้ Visual C#.NET มาสร้าง Search Engine โดยใช้คีย์เวิร์ด (Keyword) ในการค้นหาโดยทำงานบนระบบเครือข่ายอินเตอร์เน็ตผ่าน Web-Browser โดยอาศัย Google เป็น Web-Base Application

1.2.3 สร้างโปรแกรมค้นหาข้อมูลประเภทรูปภาพ (Image Search Engine) บนระบบฐานข้อมูล และใช้องค์ประกอบพื้นฐานของรูปภาพ (CBIR) ในการช่วยจัดเรียงข้อมูลรูปภาพที่มีองค์ประกอบพื้นฐาน (CBIR) ใกล้เคียงกันมากที่สุดไปจนถึงน้อยที่สุด

1.2.4 ช่วยลดช่องว่างของการใช้เทคโนโลยีกับผู้ใช้ลง โดยให้มีการทำงานที่สะดวกและรวดเร็วในการค้นหาข้อมูลประเภทรูปภาพ และยังรวมไปถึงจะสามารถหาภาพที่มีองค์ประกอบพื้นฐาน (CBIR) ใกล้เคียงกันได้ด้วย

1.2.5 เพื่อพัฒนาระบบฐานข้อมูลประเภทรูปภาพที่ใช้ภายใต้ระบบการจัดการแบบอัตโนมัติและกึ่งอัตโนมัติ (Fully Automated image analysis)

1.3 ขอบข่ายของโครงงาน

1.3.1ใช้ Visual C#.NET มาสร้าง Search Engine โดยใช้คีย์เวิร์ด (Keyword) ในการค้นหา โดยทำงานบนระบบเครือข่ายอินเตอร์เน็ตผ่าน Web-Browser โดยอาศัย Google เป็น Web-Base Application

1.3.2 ใช้ Microsoft Access สร้างระบบฐานข้อมูลรูปภาพอัตโนมัติ

1.3.3 สร้างโปรแกรมที่มีการติดต่อผู้ใช้แบบ GUI (Graphic User Interface) ด้วยภาษา Visual C#.NET เพื่อทำการค้นหาข้อมูลรูปภาพที่อยู่ในฐานข้อมูลและทำการจัดเรียงรูปภาพที่มีองค์ประกอบพื้นฐานของรูปภาพ (CBIR) ที่ใกล้เคียงกันโดยอาศัยหลักการของ Content-Base Image Retrieval using Color Histogram

1.4 ขั้นตอนการดำเนินงานของโครงการ

1.4.1 ศึกษาและค้นหาข้อมูลเกี่ยวกับหลักการเบื้องต้นเกี่ยวกับ Search Engine Meta data search, content–base image retrieval

1.4.2 ศึกษาและค้นคว้าข้อมูลเกี่ยวกับการวิเคราะห์องค์ประกอบสีของรูปภาพ (RGB) โดยอาศัยหลักการของ Color Histogram เข้ามาช่วย

1.4.3 ศึกษาและค้นคว้าการสร้าง Search Engine และการจัดเรียงข้อมูลภาพโดยอาศัยองค์ประกอบพื้นฐานของรูปภาพ (CBIR)

1.4.4 ศึกษาภาษา Microsoft Visual C#.NET และการออกแบบ GUI ด้วยภาษา Visual C#.NET

1.4.5 ศึกษา Microsoft Access เกี่ยวกับการจัดการฐานข้อมูลรูปภาพ

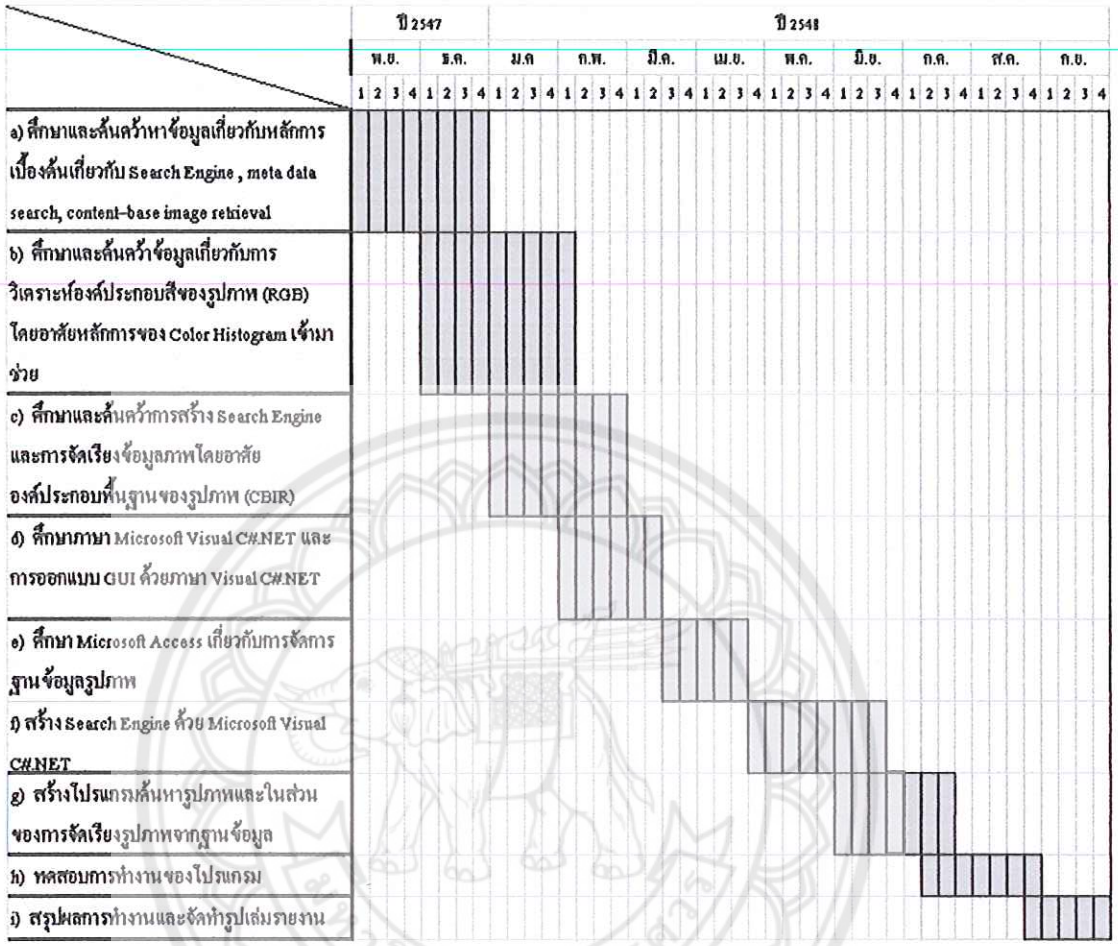
1.4.6 สร้าง Search Engine ด้วย Microsoft Visual C#.NET

1.4.7 สร้างโปรแกรมค้นหารูปภาพและในส่วนของการจัดเรียงรูปภาพจากฐานข้อมูล

1.4.8 ทดสอบการทำงานของโปรแกรม

1.4.9 สรุปผลการทำงานและจัดทำรูปเล่มรายงาน

1.5 แผนการดำเนินโครงการ



1.6 ผลที่คาดว่าจะได้รับ

1.6.1 Search Engine เพื่อการค้นหาข้อมูลรูปภาพบนระบบเครือข่ายอินเตอร์เน็ตด้วยวิธีการวิเคราะห์เนื้อหาและคำ โดยอาศัย Google เป็น Web-Base Application ซึ่งพัฒนาบน Microsoft Visual C#.NET ที่สามารถรองรับการใช้งานจากผู้ใช้งานที่ต้องการความลดความล้าช้าในการค้นหาข้อมูลภาพ บนระบบเครือข่ายอินเตอร์เน็ต

1.6.2 โปรแกรมค้นหาข้อมูลรูปภาพบนระบบฐานข้อมูลและจัดเรียงข้อมูลรูปภาพโดยอาศัยองค์ประกอบพื้นฐานของรูปภาพ (CBIR) ซึ่งเป็นแอพพลิเคชั่นที่ใช้งานง่ายไม่จำกัดขอบเขตเฉพาะและไม่จำกับจำนวนผู้ใช้งาน

1.7 งบประมาณที่ใช้

1.7.1 ค่าใช้จ่ายในการซื้ออุปกรณ์ทางคอมพิวเตอร์	1000 บาท
1.7.2 ค่าใช้จ่ายในการทำรายงาน	1500 บาท
1.7.3 ค่าใช้จ่ายเบ็ดเตล็ด	500 บาท
รวมทั้งสิ้น	3000บาท

บทที่ 2
หลักการและทฤษฎี

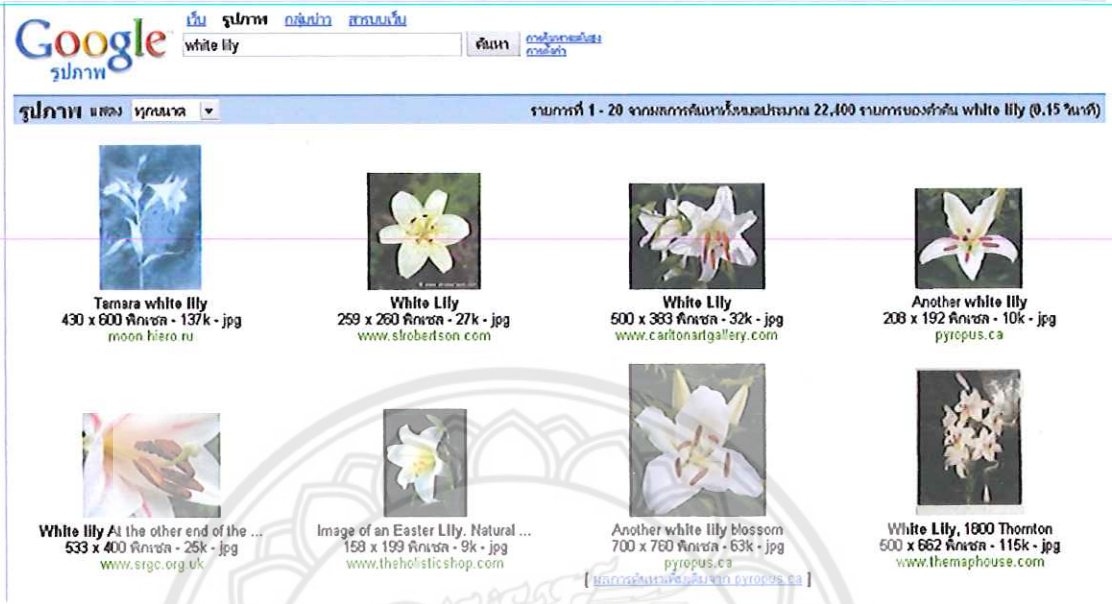
2.1 หลักการทำงานของ Google Search Engine

Google เป็นเว็บไซต์ที่ให้บริการค้นหาข้อมูลโดยใช้ดัชนีข้อความ และจัดเป็นแหล่งรวบรวมดัชนีข้อมูลที่ใช้ในการค้นหาข้อมูล โดยจะใช้โปรแกรมที่เรียกว่า spiders หรือ robots เพื่อคืบคลานเข้าไปค้นหาข้อมูลตามเว็บไซต์ใหม่ๆ ที่เปิดขึ้นมา ซึ่งตัว spider นี้ทำงานได้ดีพอๆกับการทำงานของเว็บมาสเตอร์เอง และในปัจจุบันหลายๆ เว็บไซต์ได้มีการติดตั้ง search engine ของ Google ไว้ในเว็บของตนเอง เพื่อใช้เป็นเครื่องมือตัวหนึ่งในการค้นหาข้อมูลจากเว็บต่างๆ

คุณสมบัติของ Google

- กลุ่มคำที่ใส่ในช่องค้นหา Google จะอ่านเรียงจากซ้ายไปขวา และถ้าระบุคำหรือวลีได้เฉพาะเจาะจงเท่าไรก็จะได้ผลลัพธ์ที่รวดเร็วและตรงใจมากขึ้นเท่านั้น เพราะ Google จะไม่ต้องเสียเวลาในการแสดงผลลัพธ์ที่มีความนิยมสูงจำนวนมากกลับมาให้เรา
- Google จะรองรับการใช้ wildcard(*) เช่น ถ้าใส่คำว่า color* มันก็จะแสดงผลลัพธ์ทั้งคำว่า colors และ coloring ออกมา ซึ่งกระบวนการนี้บางทีถูกเรียกว่า stemming คือค้นหาทุกคำที่เริ่มต้นด้วยคำว่า color
- เครื่องหมายและสัญลักษณ์ ในSearch Engineเครื่องหมาย (+) หรือ (-) จะใช้แทน AND หรือ NOT เช่น ในกรณีที่ใส่คำว่า Laptop OR notebook review การใช้ OR ในประโยคจะเป็นตัวบอกให้ Search Engineเรียกข้อมูลเว็บเพจที่มีทั้งสองคำนี้ปรากฏอยู่ขึ้นมา ผลลัพธ์ที่ได้จะแสดงรายชื่อเว็บต่างๆ มากมายที่จะช่วยคุณเปรียบเทียบราคา Laptop รุ่นต่างๆ ได้ตามที่ต้องการ และถ้าคุณตัดสินใจว่าจะเลือกดูข้อมูลเฉพาะของบริษัท Dell คุณอาจจะใส่เงื่อนไขในการค้นหาเป็น Laptop OR notebook + Dell + price ซึ่งจะบอกSearch Engineให้ส่งผลลัพธ์เฉพาะหน้าเว็บเพจที่มีคำว่า laptop หรือ notebook โดยที่มีคำว่า Dell กับ Price อยู่ด้วย
- Google จะจัดลำดับผลลัพธ์ โดยดูที่จำนวนของเว็บไซต์อื่นๆ ที่ลิงค์ที่เว็บเพจผลลัพธ์นี้ และจำนวนครั้งที่กลุ่มคำที่ใช้ในการค้นหา ปรากฏบนเว็บเพจนั้น และเงื่อนไขอื่นๆ
- สามารถจำกัดขอบเขตในการค้นหา โดยการเซตที่เซอร์วิสของทางเว็บไซต์ อย่างเช่น การบอกให้ส่งผลลัพธ์ออกมาในภาษาอังกฤษเท่านั้น ส่วนผลลัพธ์ที่เป็นภาษาอื่นให้ฟิลเตอร์ออกไป เช่น การค้นหาชื่อของนักปรัชญาชาวสวิตเซอร์แลนด์ชื่อ Charles Bonnet จะได้ผลลัพธ์ผสมกันระหว่างภาษาเยอรมัน และภาษาอังกฤษ ใน Google ใช้คุณสมบัติ translate-this-page ของ Google เพื่อแปลลิงค์ในภาษาเยอรมัน แต่คุณสมบัติ

นี้ก็ยังคงอยู่ในขั้นทดสอบเท่านั้น จึงเปลี่ยน preferences เพื่อเรียกดูเฉพาะข้อมูลที่มีเนื้อหาเป็นภาษาอังกฤษเท่านั้นแทน



รูปที่2.1 ตัวอย่างการค้นหาโดยใช้ "White lily"ในการค้นหา

ปัญหาและข้อจำกัดเราพิจารณาโดยนำหลักการวิเคราะห์องประกอบพื้นฐานของรูปภาพ Content Base Image Retrieval (CBIR) เข้ามาช่วย สิ่งที่ได้นั่นคือรูปภาพที่มีองค์ประกอบพื้นฐานหรือรูปภาพตามที่ต้องการหรือใกล้เคียงกับความต้องการมากที่สุด ดังนั้นจะเห็นว่า CBIR เป็นเทคโนโลยีที่กำลังได้รับความนิยมในโลกของอินเตอร์เน็ต

รูปที่2.2 ตัวอย่างการใช้ CBIR

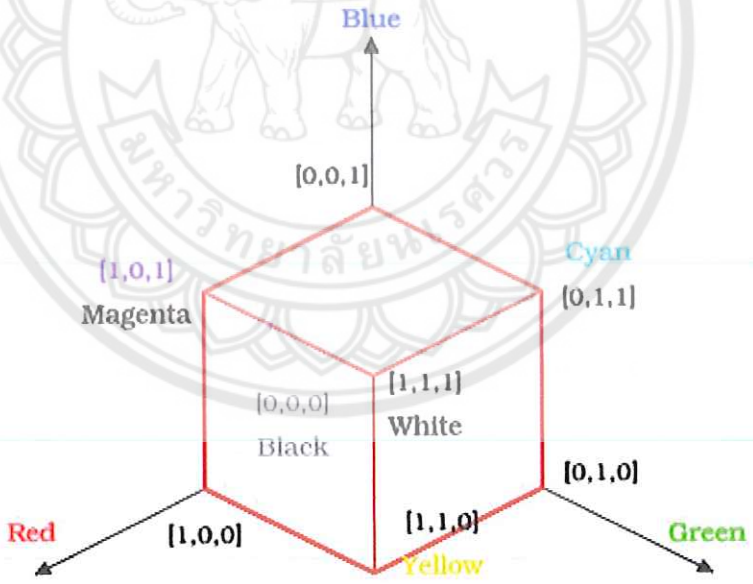
2.2 มาตรฐานของสี

มาตรฐานของสีที่ใช้อยู่ในปัจจุบันมีอยู่หลายระบบด้วยกัน ทั้งนี้จะขึ้นอยู่กับการนำไปใช้ แต่โดยทั่วไปแล้วทุกมาตรฐานจะมีแนวคิดเดียวกันคือ การแทนจุดสีด้วยจุดที่อยู่ภายในสเปส 3 มิติ โดยจะมีแกนอ้างอิง

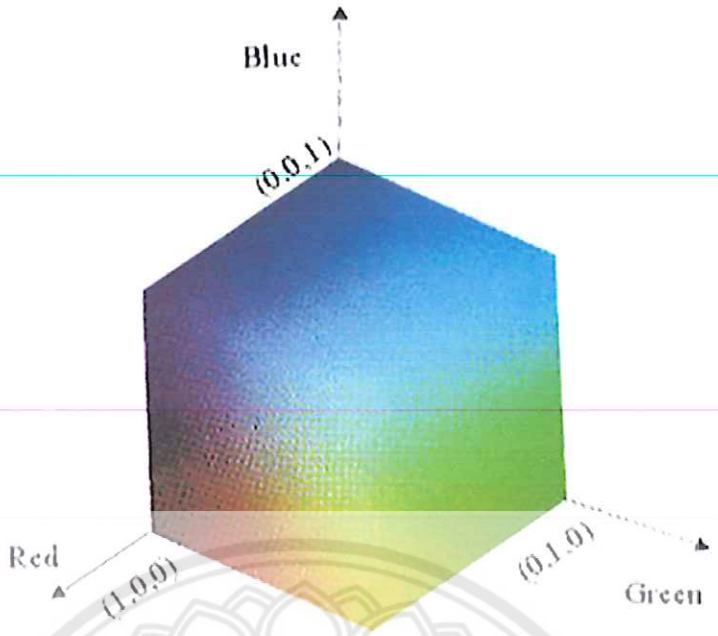
สำหรับจุดสี นั้นในสเปสซึ่งแต่ละแกนจะมีความเป็นอิสระต่อกัน ตัวอย่างเช่นในระบบ RGB จะมีแกนสีคือ แกนสีแดง เขียว และน้ำเงินในระบบ HLS จะมีแกนเป็น ค่าสี (hue) ความสว่าง (lightness) และความบริสุทธิ์ของสี (saturation) ตัวอย่างระบบสีที่เราสนใจในการพิจารณาคือ ระบบ RGB (Red Green Blue)

2.2.1 ระบบสี RGB

ระบบสี RGB เป็นระบบสีที่เกิดจากการรวมกันของแสงสีแดง เขียวและน้ำเงินโดยมีการรวมกันแบบ Additive ก็จะเป็นกราฟที่มี 3 แกน โดยแต่ละแกนก็จะแทนค่าสี R, G, B โดยไล่จากค่า 0 ที่มีความเข้มสีมาก จนไปถึงค่า 1 ที่มีความเข้มสีน้อยดังรูป



รูปที่ 2.3 RGB coordinates system



รูปที่ 2.4 RGB color model เมื่อมองจากทางด้าน white

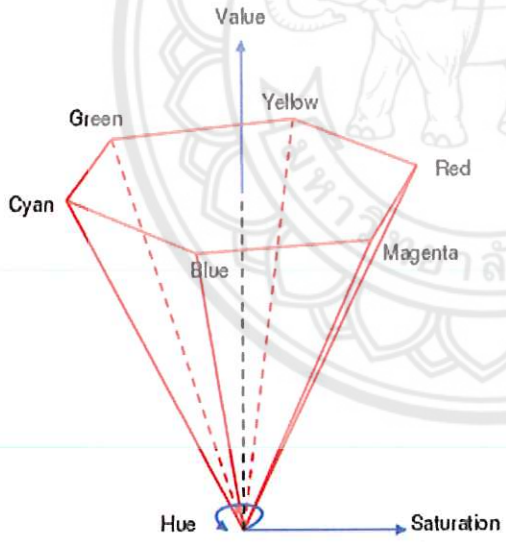
การบอกหรือระบุสีต่างๆ โดยใช้ "พิกัด" หรือ "ค่าของแม่สี" สามค่า ในการแทนค่าของสีต่างๆที่เราใช้งาน เช่น ในงานพิมพ์แยกสี จึงทำให้ดูเหมือนว่า "สี" นั้นมี สามมิติ ตามมิติของ "แม่สี" ทั้งสาม คือ แดง เหลือง และฟ้า โดยสมมติ ให้เป็นค่า R Y B ต่างๆ (ตัวย่อของ Red Yellow และ Blue) หรือเราอาจจะใช้ "ค่าแม่สีของแสง" คือ R G B (Red Green Blue) ในการสร้างแสงสีจาก "แม่สีของแสง" สามสีซึ่งต่างจาก "แม่สีของสี" ตรงที่สีเหลืองกับ สีเขียว กล่าวคือแม่สีของสีจะเป็น "สีเหลือง" แต่ แม่สีของแสงจะเป็น "สีเขียว ซึ่งสามารถเทียบเคียงได้กับการระบุพิกัดแบบใช้ค่า X Y Z ในการระบุตำแหน่งของวัตถุต่างๆในระบบพิกัดแบบคาร์ทีเซียนนั่นเอง เมื่อเป็นเช่นนี้ จึงทำให้ดูเหมือนว่า สี ต่างๆนั้นมี สามมิติ คือค่า R Y B หรือ ค่า R G B ต่างๆ ตามส่วนประกอบของมัน ซึ่งเราก็สามารถ สร้าง "ลูกบาศก์ของสี" (Color Cube) ขึ้นได้จากค่า R Y Bหรือ R G Bทั้งสามค่านั้นได้ ดังนั้นสีจึงดูกลายเหมือนเป็น"สามมิติ"ไปในทันที ทั้งๆที่ความจริงแล้ว สีต่างๆนั้นมันอยู่ในมิติเดียวกัน คือถ้าหาก เราเอาสีต่างๆมาเรียงต่อๆกัน มันจะ กลายเป็น "แถบสเปคตรัมของแสง"หรือ" แถบสีรุ้ง" อย่างที่เราคุ้นเคยกันดี เพราะในธรรมชาติมันก็เป็นเช่นนั้นจริงๆคือ เวลาที่เราเห็น รุ้งกินน้ำบนท้องฟ้าหรือ เราเห็นแสงหักเหผ่านแท่งแก้วปริซึมที่กระจายออกเป็นแถบสีรุ้ง ตาม Spectrum ของแสงคือ VIBGYOR หรือ ROYGBIV กล่าวคือ แดง ส้ม เหลือง เขียว น้ำเงิน คราม ม่วง(Red Orange Yellow Green Blue Indigo Violet)เรียงลำดับตามความยาวคลื่นหรือความถี่ของแสงนั่นเอง นั่นคือแสงสีต่างๆนั้นต่างอยู่ในมิติเชิงเส้นอันเดียวกัน ดังนั้นการที่ เกิดค่า R Y B หรือ R G B ของสีต่างๆ นั้นจึงเป็น "รงคมิติ" หรือเกิด"มิติเทียม" ของสี ขึ้น จากส่วนผสมของแม่สีทั้งสาม ซึ่งไม่ใช่มิติที่แท้จริง

2.2.2 ระบบสี HSV/I

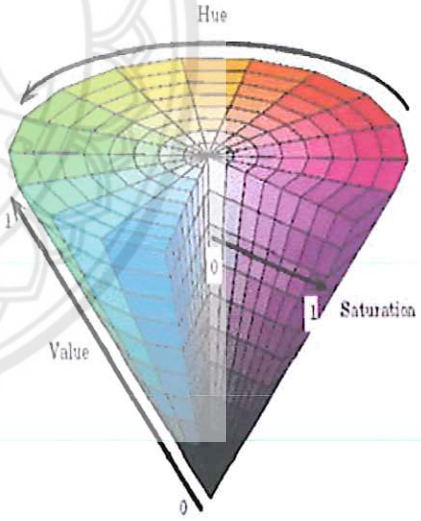
HSV (Hue Saturation and Value) หรือ HIS (Hue, Saturation and Intensity) ยึดหลักการแลกเปลี่ยนในความเข้มแสง ค่า HSV สามารถรับจาก ค่า RGB และ HSV color space สามารถหาค่าง่ายกว่า RGB color space

สีเป็นรูปแบบที่เห็นได้ชัดที่สุดสำหรับข้อมูลรูปภาพ เนื่องจากสิ่งแรกที่สังเกตเห็นในรูปภาพคือสี ก่อนที่สามารถจะวิเคราะห์รายละเอียดภายในรูปภาพได้ ในการดึงข้อมูลเพื่อมาเปรียบเทียบความแตกต่างระหว่างสีจะต้องมีโมเดลของสีที่ใช้ อย่างเช่น การแบ่งช่องว่างระหว่างสีให้สอดคล้องกับการแยกความแตกต่างของสีโดยสายตาของมนุษย์ โมเดลสีที่ใช้ชื่อว่า HSV (hue, saturation, value) ค่า hue หมายถึงโทนสี ค่า saturation หมายถึง ค่าความสดใสที่เกิดจากแสงขาว และค่า value แสดงถึงความเข้มแสง มนุษย์จะรับรู้สีจากโทนสี, ค่าความสดใส และความเข้มแสง

รูปภาพทุกรูปจะต้องแปลงจากโมเดลสี RGB ให้อยู่ในโมเดลสี HSV ซึ่งค่าความเข้มแสงมีผลกระทบไม่มากนักต่อการมองเห็นของมนุษย์ การไม่พิจารณาค่าความเข้มแสงจะช่วยในการคำนวณให้เร็วขึ้นและเนื้อที่ในการเก็บข้อมูลน้อยลง ในการคำนวณจะนำค่าแผนภูมิของค่าโทนสีกับแผนภูมิของค่าความสดใสแปลงให้อยู่ในรูปเวกเตอร์



รูปที่ 2.5 HSV coordinates system



รูปที่ 2.6 HSV color model

2.3 Content Base Image Retrieval

Color retrieval (Color histogram) เป็นเทคนิคที่สำคัญสำหรับการสร้างดัชนีฐานข้อมูลรูปภาพและการค้นหาโดยหลักการคือ ทำการวิเคราะห์รูปภาพแต่ละรูปเพื่อคำนวณหาค่าองค์ประกอบหรือลักษณะที่แสดงสัดส่วนของ pixels ของแต่รูปภาพ เช่น ค่าองค์ปรกอบของสี สำหรับรูปภาพแต่ละรูปถูกวิเคราะห์แล้วจะถูกเก็บไว้ที่ฐานข้อมูลและเมื่อเวลาค้นหาผู้ใช้สามารถจะ

เลือกภาพเจาะจงลงไปอันใดอันหนึ่งตามสัดส่วนที่ต้องการของแต่ละสี (เช่น สีเขียว 75% สีแดง 10% และสีน้ำเงิน 15%) หรือส่งกลับรูปภาพตัวอย่างที่เลือกซึ่งค่าองค์ประกอบของสีถูกคำนวณไว้แล้วทำการการจับคู่รูปภาพที่ค่าองค์ประกอบของสีที่มีความใกล้เคียงกันมากที่สุดแล้วคืนรูปภาพเหล่านั้นให้ผู้ใช้งาน พัฒนาโดย Swain และ Ballard ในปี1991 เทคนิคนี้ถูกใช้ในส่วนของระบบ CBIR ปัจจุบัน

ซึ่งบางครั้งแล้วรูปภาพที่มีค่าองค์ประกอบสีที่ใกล้เคียงกันอาจจะไม่ตรงตามการตีความหมายของผู้ใช้งาน(User Symmetric) ซึ่งเรียกว่า Symmetric gap

Symmetric gap เป็นปัญหาของ Content base image retrieval มันสะท้อนถึงความขัดแย้งกันระหว่างlow-level imagery features use by the retrieval algorithm กับ high-level concepts require by user system คือความขัดแย้งของ user algorithm กับ ความต้องการหรือเหตุผลของตัวผู้ใช้เอง

ในการแก้ปัญหา Symmetric gap ทำได้โดย 2วิธีใหญ่ๆ คือ

วิธีที่ 1 Relevance feedback ซึ่งจะมีปริสิทธิภาพในบาง Application เท่านั้นอย่างไรก็ System อาจจะเพิ่มภาระให้แก่ User เอง โดยเวลาที่มีข้อมูลมากๆ แทนที่จะเป็นแค่ Boolean feedback (relevant or non-relevant)ซึ่ง Relevance feedback สามารถเป็นส่วนหนึ่งของการทำ Search engine โดย keyword-base จะสนับสนุนการมองภาพในฐานข้อมูลโดยการจัดกลุ่มของรูปภาพ ซึ่งรูปภาพที่เหมือนกับภาพที่ต้องกาเท่านั้นถึงจะได้รับความสนใจ

วิธีที่ 2 Classification methods ทำการจัดกลุ่มภาพซึ่งทำให้การค้นหาทำได้ง่ายจะมีปัญหาเมื่อฐานข้อมูลมีขนาดใหญ่

2.3.1 Color histogram in RGB space

RGB space (สีแดง, สีเขียวและสีน้ำเงิน) สนับสนุนโปรแกรมเกี่ยวกับ image processing ดังนั้นจึงไม่ยากที่จะสร้างโปรแกรมซึ่งอ่านรูปภาพจาก Diskถึง RAM ของคอมพิวเตอร์

RGB space มีข้อเสียหาย 2 ข้อคือ

- RGB space ไม่ใช่ perceptually รูปแบบเดียวกัน

ปัญหา Perceptual สามารถหลีกเลี่ยงโดยการใช้ weight matrix ระหว่าง 2 Histogram

$$d_{hist}(x, y) = (x-y)' A (x-y) \qquad (2.1)$$

โดยที่ A คือ weight matrix x และ y คือกราฟแท่งแสดงค่าของสถิติความถี่ ถ้าตัวเลขที่แตกต่างของ binsที่สถานะแตกต่างกัน สถานะทั้งหมดมันก็ต้องการ matrix ของมันเองตัวอย่างเช่น 64 bin histogram ก็จะต้อง weight matrix ซึ่งมีขนาดคือ 64*64

- ส่วนประกอบทั้งหมด (R, G, B) มีความสำคัญที่เท่ากันดังนั้นค่าเหล่านั้นต้อง บอกค่าจำนวน กับความประณีตเดียวกัน

วิธีผลรวม bins ที่เป็นไปได้นั้นคือของจำนวนเต็ม (เช่น 13=1, 23= 8,33=27, 43=64, 53=125, 63=216, 73=343...)ขั้นตอนเหล่านี้เราสามารถหลีกเลี่ยงปัญหาโดยการกำหนด color map เพื่อรูปภาพทุกรูปได้ใช้ร่วมกันก่อนที่ จะถูก histogramแปลงเป็นค่า color space ซึ่งตามความเป็นจริงค่า color spaceนี้ มีหน้าที่ตรงกันกับbinsและค่าของ Histogramสามารถได้รับการเพิ่มค่าที่ Histogramระหว่างการแปลงซึ่งทำให้ Algorithmกระชับขึ้น

2.3.2 Normalize RGB

Intensity	I	= (R+G+B)/3	(2.2)
Normalize red	r	= R/(R+G+B)	(2.3)
Normalize green	g	= G/(R+G+B)	(2.4)
Normalize blue	b	= B/(R+G+B)	(2.5)
Luminance	L	= r+g+b	(2.6)

2.3.3 Histogram Distance Measure

Histogram: $h_{A,B,C}(a,b,c) = N \cdot \Pr ob(A=a, B=b, C=c)$ (2.8)

Histogram distance

Euclidean distance

$$d^2(h,g) = \sum_A \sum_B \sum_C (h(a,b,c) - g(a,b,c))^2 \quad (2.9)$$

Intersection distance

$$d(h,g) = \frac{\sum_A \sum_B \sum_C \min(h(a,b,c), g(a,b,c))}{\min(|h|,|g|)} \quad (2.10)$$

Quadratic distance (Cross distance)

$$d(h,g) = (h-g)^t A (h-g) \quad (2.11)$$

Difference distance

$$dif(h,g) = \sqrt{(h_a - g_a)^2 + (h_b - g_b)^2 + (h_c - g_c)^2} \quad (2.12)$$

2.4 ภาษา C#.net

เนื่องจากทุกภาษาใน .Net จะถูกแปลเป็นภาษา Microsoft Intermediate Language (MSIL) ถ้าขนาดของโปรเจ็กต์ที่คุณคิดว่าจะต้องทำในอนาคตใหญ่มากๆ ก็ควรจะเลือก C#

C# เป็นภาษา โครงสร้างบล็อก (Block structured) หมายความว่า ทุกๆ Statement ถือเป็นส่วนหนึ่งของ Block Code และ Block เหล่านี้ถูกจำกัดด้วย เครื่องหมาย { } เหมือนกับ C++ ทุกประการ รูปแบบจึงเหมือนกับ C++ และมีโครงสร้างเหมือน C++ ดังนั้นหากเราได้ศึกษาโครงสร้างของ C++ แล้ว ก็จะเข้าใจ

C# เป็นภาษาที่ถูกสร้างขึ้นมาเพื่อทำงานบน .NET Framework สร้างและก็ทำงานในลักษณะของ Object Oriented ได้อย่างสมบูรณ์ ซึ่งเมื่อเปรียบเทียบกับ C++ ที่ยังทำงานในลักษณะของ Object Oriented Programming (OOP) ได้บางส่วน, ไลบรารีของ C# ถูกสร้างขึ้นเพื่อให้ทำงานได้ครอบคลุมตั้งแต่การสร้างรูปแบบการติดต่อแบบ GUI ไปจนถึงการAccess ฐานข้อมูลผ่านอินเตอร์เน็ตหรือแม้แต่การทำงานร่วมกับ XML เพื่อทำให้การแลกเปลี่ยนข้อมูลระหว่างแอพพลิเคชั่นซึ่งทำได้อย่างสมบูรณ์ไม่ว่าข้อมูลนั้นจะอยู่บนPlatformใดก็ตาม

เมื่อเปรียบเทียบกับ C++ แล้ว การสร้างแอพพลิเคชั่นจะทำได้ง่ายกว่ามาก เนื่องจาก C# ถูกออกแบบมาเพื่อการสร้างแอพพลิเคชั่นให้ทำงานบนอินเตอร์เน็ตโดยตรง (.NET Framework) นอกจากนี้ C# เป็น Object Oriented Programming (OOP) อย่างสมบูรณ์ ไม่ว่าจะเป็น

- Encapsulation การรวมกลุ่มฟังก์ชั่นการทำงานของออบเจ็กต์ต่างๆ (Object Blueprint, Class) เพื่อให้โค้ดถูกเขียนขึ้นมาอย่างเป็นระเบียบ
- Polymorphism (Inheritance, Interfacing และ Overloading) การนำโค้ดที่เขียนขึ้นมาแล้วนั้นมาใช้ในงานอื่นได้อีก

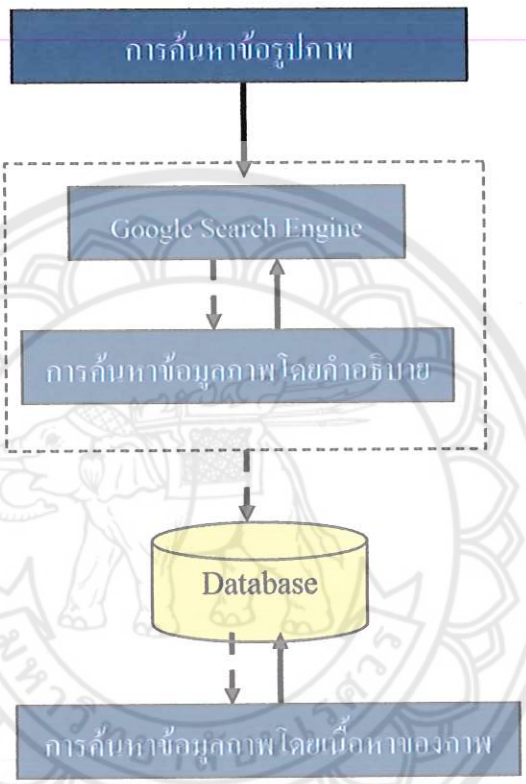
การเขียนโปรแกรมโดยใช้ C#.NET เมื่อเปรียบเทียบกับการเขียนโดยใช้ Visual Basic หรือ Visual Basic.NET แน่นอนว่าการเขียนโดยใช้ Visual Basic หรือ Visual Basic.NET ทำได้ง่ายกว่า แต่ถ้าพูดถึงประสิทธิภาพของโปรแกรมแล้วจะด้อยกว่าโปรแกรมที่เขียนขึ้นมาจาก C#.NET มากแต่ข้อด้อยของC#.NET ก็คือเมื่อเราต้องการเขียนApplication เพื่อทำการติดต่อกับฮาร์ดแวร์ซึ่งก็จะทำได้ยากเพราะโครงสร้างของC#.NET นั้นถูกออกกแบบมาทำงานบน .NET Framework โดยเฉพาะซึ่งก็จะทำงานมีประสิทธิภาพน้อยเมื่อเทียบกับ C++ แต่อย่างไรก็ตามก็ต้องขึ้นอยู่กับผู้ใช้งานเองว่ามีความถนัดมากน้อยแค่ไหนและต้องการพัฒนา Application ลักษณะใด แต่ถ้าต้องการความง่ายในการเขียนโปรแกรม โดยไม่ต้องคำนึงถึงประสิทธิภาพการทำงานมากนัก ควรเลือกใช้ Visual Basic, C# ซึ่งรวมเอาลักษณะการเขียนโปรแกรมจากภาษาทั้งสอง เข้ามาไว้ เช่น C# จะไม่มี Overhead มากนัก เมื่อเทียบกับ Visual Basic

ข้อดีของ C# .Net

- รองรับ XML documentation คล้ายๆ javadoc คือเอาคอมเม้นต์ในโค้ดมาแปลง
- เป็นเอกสาร technical manual ได้เลย
- สามารถทำ operator overloading ได้
- รองรับ unsigned data type
- มีประโยค using เพื่อใช้จัดการกับ resource ที่เป็นแบบ unmanaged
- รองรับ unsafe code
- C# เป็น Object Oriented Programming (OOP) อย่างสมบูรณ์ ไม่ว่าจะเป็น
- Encapsulation การรวมกลุ่มฟังก์ชั่นการทำงานของออบเจ็กต์ต่างๆ
- (Object Blueprint, Class) เพื่อให้โค้ดถูกเขียนขึ้นมาอย่างเป็นระเบียบ

บทที่ 3
วิธีการดำเนินการ

ในการทดลองการค้นหาข้อมูลภาพบนอินเตอร์เน็ตโดยใช้Application นั้น จะแบ่งขั้นตอนการดำเนินการงานออกเป็นส่วนต่างๆ ได้ดังนี้



รูปที่ 3.1 ส่วนประกอบของขั้นตอนการค้นหารูปภาพ

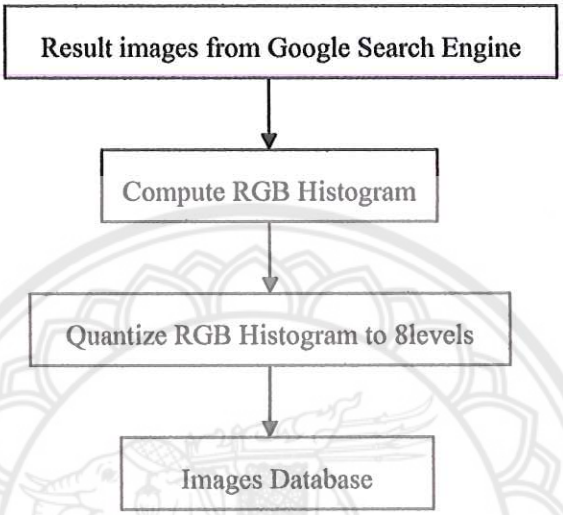
โดยจะมีการทำงานย่อยๆ ลงไปอีกในส่วนต่างๆ โดยจะมีขั้นตอนการทำงานดังต่อไปนี้

3.1 การค้นหาข้อมูลภาพโดยคำอธิบาย (Meta data Search)

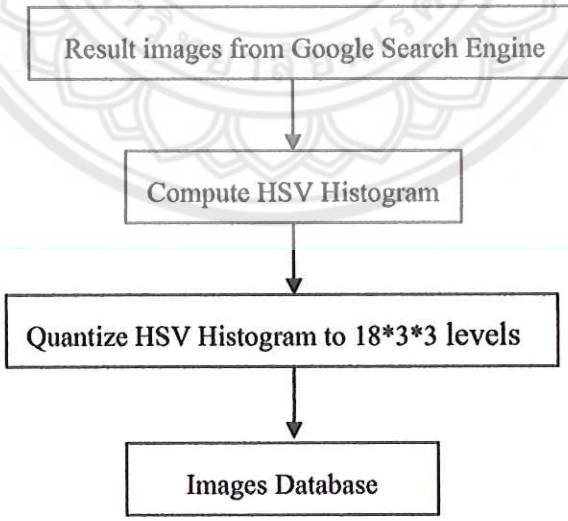
ในส่วนนี้เป็นส่วนแรกของการค้นหาโดยจะมี Interfaceที่จะช่วยในการไปร้องขอบริการจาก Google Search Engine ให้ทำการค้นหารูปภาพตามคำอธิบายของภาพหรือKeywordต่างๆที่โปรแกรมทำการร้องขอบริการไป เมื่อทำการค้นหาแล้วจะส่งผลลัพธ์ทั้งหมดมายังโปรแกรมดังกล่าว ในรูปแบบของรูปภาพ

3.1.1การจัดการฐานข้อมูล

โปรแกรมจะทำการวิเคราะห์องค์ประกอบของภาพโดยส่วนนี้จะเลือกองค์ประกอบของสี RGB และHSV (จะอธิบายในส่วนต่อไป) แล้วทำการเก็บข้อมูลองค์ประกอบของสีดังกล่าว รวมถึงข้อมูลที่สำคัญเพื่อที่จะช่วยดึงรูปภาพจากฐานข้อมูลขึ้นมาแสดงในโปรแกรมได้ เช่น ตำแหน่งที่อยู่ของรูปภาพ, ชื่อรูปภาพ เป็นต้น ซึ่งการวิเคราะห์ค่าองค์ประกอบของสีนั้นจะมีหลักการดังนี้



รูปที่ 3.2 ขั้นตอนการวิเคราะห์องค์ประกอบสี R, G, Bของรูปภาพ



รูปที่ 3.3 ขั้นตอนการวิเคราะห์องค์ประกอบสี H, S, Vของรูปภาพ

3.2 การหาค่า RGB Color histogram

3.2.1 Normalization RGB

โดยจะทำการวิเคราะห์ค่าสีทุกๆ จุด Pixel ของรูปภาพแล้วทำการหาค่าเฉลี่ยขององค์ประกอบสีแต่ละสี จะทำให้ได้ค่าองค์ประกอบสีที่ออกมาในรูปเวกเตอร์ ซึ่งแต่ละรูปนั้นจะมีค่าเวกเตอร์ 3 ค่า คือ

R ซึ่งจะเป็นค่าเวกเตอร์สีแดงที่เป็นองค์ประกอบของภาพนั้น

G ซึ่งจะเป็นค่าเวกเตอร์สีเขียวที่เป็นองค์ประกอบของภาพนั้น

B ซึ่งจะเป็นค่าเวกเตอร์สีน้ำเงินที่เป็นองค์ประกอบของภาพนั้น

ซึ่งจะหาได้จากสมการที่ 3.1, 3.2, 3.3 ตามลำดับ

$$R = \frac{\sum_{i=0}^{y}\sum_{j=0}^{x} R_{(i,j)}}{xy} \quad (3.1)$$

$$G = \frac{\sum_{i=0}^{y}\sum_{j=0}^{x} G_{(i,j)}}{xy} \quad (3.2)$$

$$B = \frac{\sum_{i=0}^{y}\sum_{j=0}^{x} B_{(i,j)}}{xy} \quad (3.3)$$

i, j : ตำแหน่ง dimension ของจุดบนรูปภาพ

$R_{(i,j)}$: ค่าขององค์ประกอบสีแดงบนจุดPixel (i, j) ที่ทำการวิเคราะห์

$G_{(i,j)}$: ค่าขององค์ประกอบสีเขียวบนจุดPixel (i, j) ที่ทำการวิเคราะห์

$B_{(i,j)}$: ค่าขององค์ประกอบสีน้ำเงินบนจุดPixel (i, j) ที่ทำการวิเคราะห์

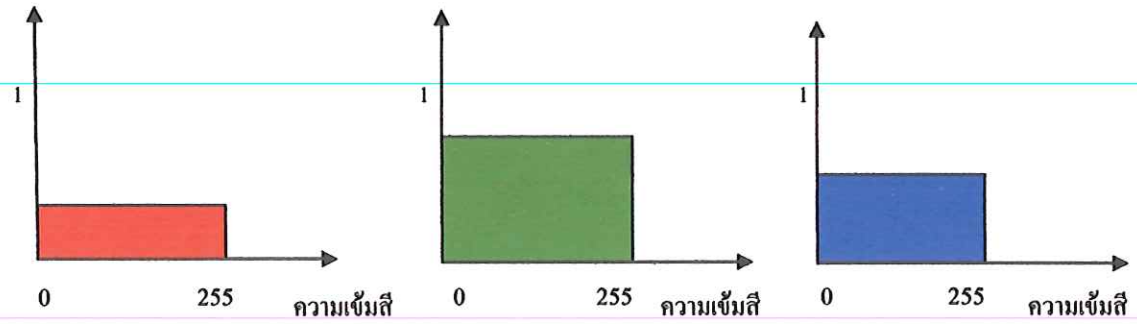
x ,y : ความกว้างและยาวของรูปภาพตามลำดับ

เมื่อเราได้ค่าเฉลี่ยของแต่ละสีแล้ว จากนั้นทำการ Normalization ค่าของ RGB Color Histogram จากสมการ

Normalize red r = R/(R+G+B) (3.4)

Normalize green g = G/(R+G+B) (3.5)

Normalize blue b = B/(R+G+B) (3.6)



รูปที่ 3.4 รูปกราฟแสดงองค์ประกอบสีแต่ละสีของรูปภาพใน Unit plane

เมื่อทำการวิเคราะห์แล้วจะได้กราฟแสดงความถี่ของค่าสีแต่ละสีดังรูปที่ 3.4 จากการวิเคราะห์จะได้ค่าองค์ประกอบสี 3 ค่าดังกล่าวจากนั้นทำการจัดเก็บลงฐานข้อมูล ดังรูปที่ 3.5

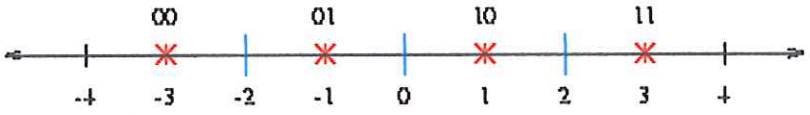
	img_id	location	rValue	gValue	bValue
	69	C:\Images\sunflower1.jpg	.39626	.36225	.24149
	70	C:\Images\sunflower2.jpg	.39893	.36626	.23481
	71	C:\Images\sunflower3.jpg	.44008	.39378	.16614

รูปที่ 3.5 ค่าองค์ประกอบของสี R, G, B ในฐานข้อมูล

3.2.2 Vector Quantization (VQ)

การทำเวกเตอร์ 1 มิติ

เป็นการเก็บค่าองค์ประกอบสีของภาพที่มีขนาดใหญ่ให้มีขนาดเล็กลง โดยการเก็บให้อยู่ในรูปของ Vector เราจะประมาณค่าของสีที่มีความใกล้เคียงกัน ประมาณค่าให้เป็นค่าเดียวกัน ดังตัวอย่าง Vector ใน 1 มิติ



รูปที่ 3.6 ภาพแสดงการหาค่าเวกเตอร์ใน 1 มิติ

โดยค่าที่น้อยกว่า -2 จะประมาณให้เป็น -3, ค่าที่อยู่ระหว่าง -2 กับ 0 ให้มีค่าเป็น -1, ค่าที่อยู่ระหว่าง 0 กับ 2 ให้มีค่าเป็น 1, ค่าที่มากกว่า 2 ให้มีค่าเป็น 3 จะเห็นว่าจากที่เราต้องเก็บค่าทั้งหมด 3 bit เราสามารถให้เหลือ 2 bit ทำให้เก็บค่าได้ง่ายขึ้น

3.2.3 Quantize RGB to 8*8*8 Levels

แบ่งระดับความเข้มสีของแต่และสีเป็น 8 ช่วงความถี่สี ซึ่งระดับสีของสีแต่ละสีจะมีค่าตั้งแต่ 0-255 ค่า ดังนั้นเราจะสามารถแบ่งช่วงระดับความเข้มสีเป็นช่วง ๆ ละ 32 ค่า เราสามารถจะหาค่าความถี่ของระดับความเข้มสีแต่ละช่วงได้จากสมการ

$$R = \frac{\sum_{i=0}^{y}\sum_{j=0}^{x} R[r]}{xy} \quad (3.7)$$

$$G = \frac{\sum_{i=0}^{y}\sum_{j=0}^{x} G[g]}{xy} \quad (3.8)$$

$$B = \frac{\sum_{i=0}^{y}\sum_{j=0}^{x} B[b]}{xy} \quad (3.9)$$

i, j : ตำแหน่ง dimension ของจุดบนรูปภาพ

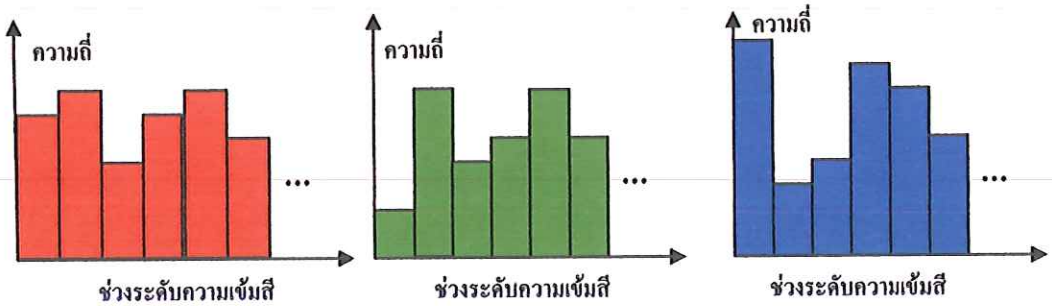
r, g, b : ช่วงของระดับสีที่แบ่งแบ่งเป็นช่วง

R[r] : Pixelที่มีค่าขององค์ประกอบสีแดงอยู่ในช่วง r ใด ๆ

G[g] : Pixelที่มีค่าขององค์ประกอบสีแดงอยู่ในช่วง g ใด ๆ

B[b] : Pixelที่มีค่าขององค์ประกอบสีแดงอยู่ในช่วง b ใด ๆ

x, y : ความกว้างและยาวของรูปภาพตามลำดับ



รูปที่ 3.7 รูปกราฟแสดงค่าองค์ประกอบของสี R, G, B เมื่อทำการ Quantize เป็นเวกเตอร์

จากการวิเคราะห์จะได้เวกเตอร์ของค่าองค์ประกอบสีดังกล่าวจากแล้วทำการ Quantize โดยแต่ละสีเป็น 8 เวกเตอร์ แล้วมารวมกันทำให้เป็น 24 เวกเตอร์แล้วจัดเก็บลงฐานข้อมูล ดังรูปที่ 3.8-3.11

v1	v2	v3	v4	v5	v6	v7	v8
6899	1290	368	301	292	265	287	265
896	1538	1863	3353	1612	19	398	19
6293	350	567	370	245	230	192	230
831	1740	2356	2295	314	160	264	160
5302	2785	891	247	71	140	237	140
6876	281	404	283	207	233	300	233

รูปที่ 3.8 ภาพแสดงค่าเวกเตอร์ของสีแดง

v9	v10	v11	v12	v13	v14	v15	v16
7399	791	379	289	290	272	278	382
882	1602	1831	2920	2004	42	398	401
6293	350	567	370	245	230	192	1833
900	991	2142	2954	488	223	327	2055
3809	2276	1916	1002	281	99	271	426
6818	428	390	268	266	308	423	1179

รูปที่ 3.9 ภาพแสดงค่าเวกเตอร์ของสีเขียว

v17	v18	v19	v20	v21	v22	v23	v24
5365	2810	333	279	291	327	353	322
1685	1674	1767	1784	2233	138	398	401
6293	350	567	370	245	230	192	1833
1915	2949	1038	1489	302	231	559	1597
5379	2620	796	459	66	102	275	383
6824	815	324	211	271	386	474	775

รูปที่ 3.10 ภาพแสดงค่าเวกเตอร์ของสีน้ำเงิน

3.2.4 Quantize HSV to 18*3*3 Levels

แบ่งระดับสีของแต่และสีเป็น 18, 3, 3 ตามลำดับ

- ค่า Hue จะมีระดับความเข้มอยู่ระหว่าง 0-360 จะแบ่งเป็นช่วงเท่าๆกัน ได้เป็นช่วงละ 18 ค่า
- ค่า Saturation มีระดับความเข้มระหว่าง 0-1 และจะแบ่งออกเป็น 3 ช่วงเท่าๆ กัน
- ค่า Value มีระดับความเข้มสีระหว่าง 0-1 และจะแบ่งออกเป็น 3 ช่วงเท่าๆ กัน

เราสามารถจะหาค่าความถี่ของระดับความเข้มสีแต่ละช่วงได้จากสมการ

$$H = \frac{\sum_{i=0}^{y}\sum_{j=0}^{x} H[h]}{xy} \quad (3.10)$$

$$S = \frac{\sum_{i=0}^{y}\sum_{j=0}^{x} S[s]}{xy} \quad (3.11)$$

$$V = \frac{\sum_{i=0}^{y}\sum_{j=0}^{x} V[v]}{xy} \quad (3.12)$$

i, j : ตำแหน่ง dimension ของจุดบนรูปภาพ

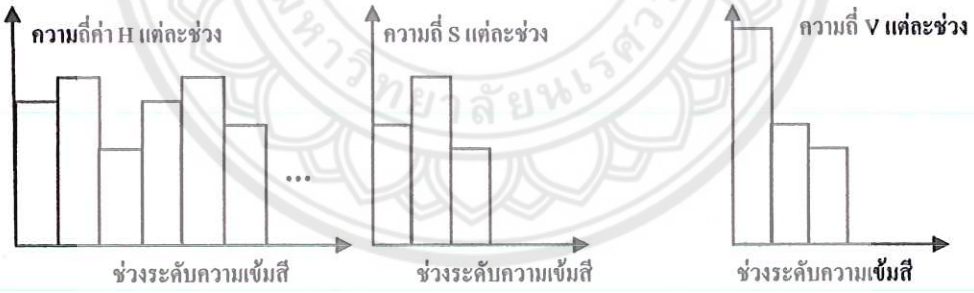
h, s, v : ช่วงของระดับสีที่แบ่งแบ่งเป็นช่วง

H[h] : Pixelที่มีค่าขององค์ประกอบสีแดงอยู่ในช่วง h ใด ๆ

S[s] : Pixelที่มีค่าขององค์ประกอบสีแดงอยู่ในช่วง s ใด ๆ

V[v] : Pixelที่มีค่าขององค์ประกอบสีแดงอยู่ในช่วง vใด ๆ

x, y : ความกว้างและยาวของรูปภาพตามลำดับ



รูปที่ 3.11 รูปกราฟแสดงค่าองค์ประกอบของสี H, S, V เมื่อทำการ Quantize เป็นเวกเตอร์

ซึ่งการทำ Normalize HSV จะทำในลักษณะเดียวกันกับการNormalize RGB และค่าของ H, S, V ก็จะเก็บในฐานข้อมูลในลักษณะเดียวกันดังนี้

	img_id	location	hValue	sValue	vValue
	621	C:\Images\white rose1.jpg	.656731	.131331	.202281
▶	622	C:\Images\white rose2.jpg	.300016	.129845	.389101
	623	C:\Images\white rose3.jpg	0	0	.276681

รูปที่ 3.12 ภาพแสดงค่าองค์ประกอบ HSV ของรูปภาพ

v25	v26	v27	v28	v29	v30	v31	v32	v33
6157	327	1926	82	952	567	33	567	2
233	272	824	743	415	296	364	296	843
1179	2227	284	24	27	5	27	5	54
0	234	9846	0	0	0	0	0	0
10080	0	0	0	0	0	0	0	0
68	2546	2669	1200	714	95	372	95	1017

รูปที่ 3.13 ภาพแสดงค่าเวกเตอร์ของ Hue (v25-v33)

v34	v35	v36	v37	v38	v39	v40	v41	v42
6	0	6	0	1	0	17	0	0
2117	3620	66	12	16	4	13	13	21
186	545	1414	2979	674	81	144	99	112
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
630	459	47	0	1	0	3	2	1

รูปที่ 3.14 ภาพแสดงค่าเวกเตอร์ของ Hue (v34-v42)

v43	v44	v45
7455	2186	439
9887	189	4
8365	1453	262

รูปที่ 3.15 ภาพแสดงค่าเวกเตอร์ของ Saturation

v46	v47	v48
7455	2186	439
9887	189	4
8365	1453	262

รูปที่ 3.16 ภาพแสดงค่าเวกเตอร์ของ Value

3.3 การค้นหารูปภาพโดยเนื้อหาของภาพ

ในส่วนนี้จะทำการค้นหาโดยใช้เนื้อหาของภาพซึ่งเราจะเลือกองค์ประกอบของสี R, G, B และ ค่าของ H, S, V เป็นเกณฑ์ในการวัดความสัมพันธ์และแบ่งกลุ่มของรูปภาพ หรือClass ของรูปภาพจากคำอธิบายของรูปภาพ

การแบ่งกลุ่มรูปภาพที่อยู่ในกลุ่มเดียวกันและอยู่ใน Class เดียวกัน

<u>ตัวอย่างเช่น</u>

กุหลาบแดง1, กุหลาบแดง2, กุหลาบแดง3 ทั้งหมดนี้อยู่ในกลุ่มเดียวกันคือกลุ่มกุหลาบแดง และ Class เดียวกันคือ Class กุหลาบ

การแบ่งประเภทและClassของรูปภาพ คือ อาจจะไม่อยู่ในกลุ่มเดียวกันแต่ต้องอยู่ใน Class เดียวกันหรือประเภทเดียวกัน

ตัวอย่างเช่น

กุหลาบแดง1, กุหลาบแดง2 ทั้งหมดนี้อยู่ในกลุ่มเดียวกันคือกลุ่มกุหลาบแดง กุหลาบขาว1, กุหลาบขาว2 ทั้งหมดนี้อยู่ในกลุ่มเดียวกันคือกลุ่มกุหลาบขาวและจะเห็นว่า กุหลาบแดง1,กุหลาบแดง2,กุหลาบขาว1, กุหลาบขาว2 จะอยู่ใน Class เดียวกันหรือประเภทเดียวกัน คือประเภทของกุหลาบซึ่งการแบ่งประเภทนี้จะขึ้นอยู่กับการแยกกลุ่มหรือประเภทของรูปภาพโดยผู้ใช้งาน (User) เอง

การกำหนดความสัมพันธ์ของรูปภาพ รูปภาพจะมีความพันธ์กันจะต้องอยู่ในกลุ่มเดียวกันและค่าขององค์ประกอบสีจะต้องอยู่ในช่วงที่เรากำหนดไว้

เมื่อเราทราบความสัมพันธ์แล้ว จากนั้นเราจะเริ่มจากนำรูปภาพมาจากผลการค้นหาในส่วนของการค้นหาโดยคำอธิบาย (Meta data Search) มาทำการค้นหาโดย CBIR ซึ่งการค้นหาสามารถเลือกวิธี ได้ 2 วิธี ด้วยกัน คือ

1. Quantize RGB color Histogram to 8 * 8 * 8 levels
2. Quantize HSV color Histogram to 18 *3 *3 levels

การค้นหารูปภาพที่ต้องการ โดยเราจะนำค่าColor Histogram ของภาพที่ต้องการนำไปเปรียบเทียบกับภาพอื่นๆในฐานข้อมูลตามความสัมพันธ์ที่เราเลือก โดยใช้หลักการเปรียบเทียบ ดังนี้

รูปที่ 3.17 ภาพแสดงขั้นตอนการเปรียบเทียบค่าองค์ประกอบสี

การเปรียบเทียบ

โดยในขั้นตอนแรก เราจะนาค่า Vector 2 ค่าที่ต้องการเปรียบเทียบมาทำการหาค่าส่วนต่าง

$$d(Q,I)=\sum_{i=1}^{N}\left|H_Q[i]-H_I[i]\right| \qquad (3.13)$$

d (Q,I) : ค่าความต่าง

N : จำนวนของข้อมูลใน Vector

H_Q และ H_I : ค่าของ Vector ที่ตำแหน่ง i ดังนั้นจะได้สมการความแตกต่างดังนี้

โดยการเปรียบเทียบแต่ละครั้ง จะได้ค่า Distance มา 1 ค่า

$$\text{Distance1}=\sum_{i=1}^{24}(\bar{b}1_i-\bar{v}i) = \left|(\bar{b}1_1-\bar{v}_1)+(\bar{b}1_2-\bar{v}_2)+....+(\bar{b}1_{24}-\bar{v}_{24})\right|$$

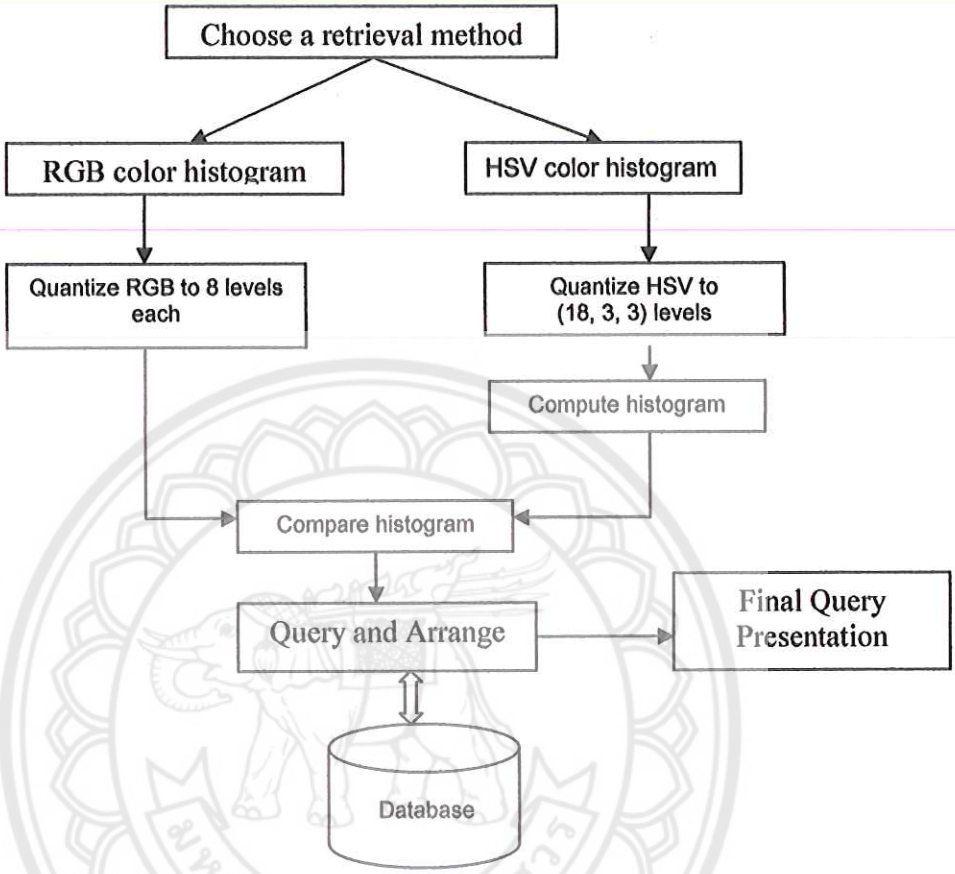
$$\text{Distance2}=\sum_{i=1}^{24}(\bar{b}2_i-\bar{v}i) = \left|(\bar{b}2_1-\bar{v}_1)+(\bar{b}2_2-\bar{v}_2)+....+(\bar{b}2_{24}-\bar{v}_{24})\right|$$

$$\vdots$$

$$\text{Distance N}=\sum_{i=1}^{24}(\bar{b}N_i-\bar{v}i) = \left|(\bar{b}N_1-\bar{v}_1)+(\bar{b}N_2-\bar{v}_2)+....+(\bar{b}N_{24}-\bar{v}_{24})\right| \qquad (3.14)$$

เมื่อเราได้ ค่า Distance แล้วทั้งหมดโปรแกรมจะทำการ Ranking ค่า Distance นั้นโดยเรียงจากค่า น้อยที่สุดไปหาค่ามากที่สุด แล้วนำค่าของ image location มาเก็บไว้ตามค่าdistance ที่เรียงแล้วแสดงผลการ Search รูปภาพทั้งหมด 20 ภาพ

3.4 แผนผังขั้นตอนการค้นหาโดย CBIR



รูปที่ 3.18 ภาพแสดงขั้นตอนการทำงาน

บทที่ 4

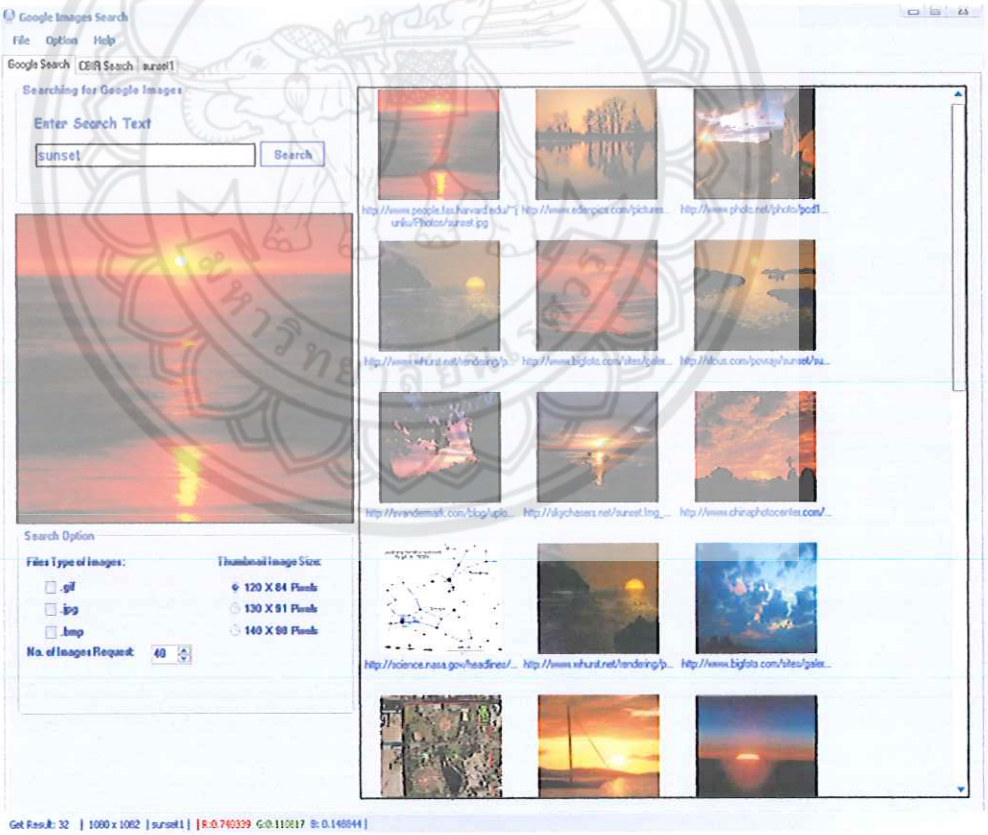
ผลการทดลอง

4.1 การค้นหารูปภาพโดยคำอธิบาย

สิ่งที่ต้องเตรียมคือ

1) ในเครื่องต้องมี .NET Framework version 2.0 ก่อน
2) เชื่อมต่อกับระบบ Internet ควรมีความเร็วที่สูง
3) โปรแกรมส่วนที่สามารถติดต่อกับ Google Search Engine ได้ ซึ่งพัฒนาด้วย C#.NET

โดยเราจะทำการค้นหาจากคำอธิบาย หรือที่เรียกว่า Keyword โดยอาศัย Google Search Engine ซึ่งผลการค้นหาด้วยคำอธิบายคำว่า "sunset" จะได้ผลการค้นหาดังรูปที่4.1



รูปที่4.1 ภาพผลการค้นหาโดยคำอธิบายของภาพ

4.2 การค้นหารูปภาพโดยเนื้อหาของภาพ

ในขั้นที่2จะทำการค้นหาโดยอาศัยเนื้อหาของรูปภาพซึ่งจะใช้ค่าองค์ประกอบสีของรูปภาพเป็นดัชนีในการค้นหาในส่วนนี้สามารถค้นหาได้วิธีหลัก 2 วิธี

และยังสามารถเจาะจงไปได้ว่าจะค้นหาโดยรูปแบบใด เช่นค้นหาจากกลุ่มเดี่ยวกัน ค้นหาจากประเภทเดียวกันหรือ Class เดียวกัน หรือเปรียบเทียบกับรูปภาพทั้งหมดในฐานข้อมูล

4.2.1 เกณฑ์ในการวัดความสัมพันธ์ของรูปภาพ

ในการค้นหาเราใช้

- ค่าของความคล้ายคลึงขององค์ประกอบสีของรูปภาพ
- ความคล้ายคลึงขององค์ประกอบของรูปภาพ

ซึ่งการกำหนดความสัมพันธ์นั้น รูปภาพเป้าหมายที่จะมีความสัมพันธ์กันนั้น จะต้องมีค่าขององค์ประกอบสีที่ใกล้เคียงกับค่าองค์ประกอบสีของรูปภาพต้นแบบ

ในการวัดว่ารูปภาพนั้นมีความสัมพันธ์กัน ซึ่งเราจะคำนวณค่าความถูกต้องในการค้นหาตามสมการที่ 4.1

$$\% \text{ ความถูกต้อง} = (\text{Relevant Images} / \text{Retrieved Images}) \times 100 \qquad (4.1)$$

4.2.2 การค้นหาข้อมูลภาพโดยระบบสี RGB (Quantize RGB Color Histogram to 8 levels)

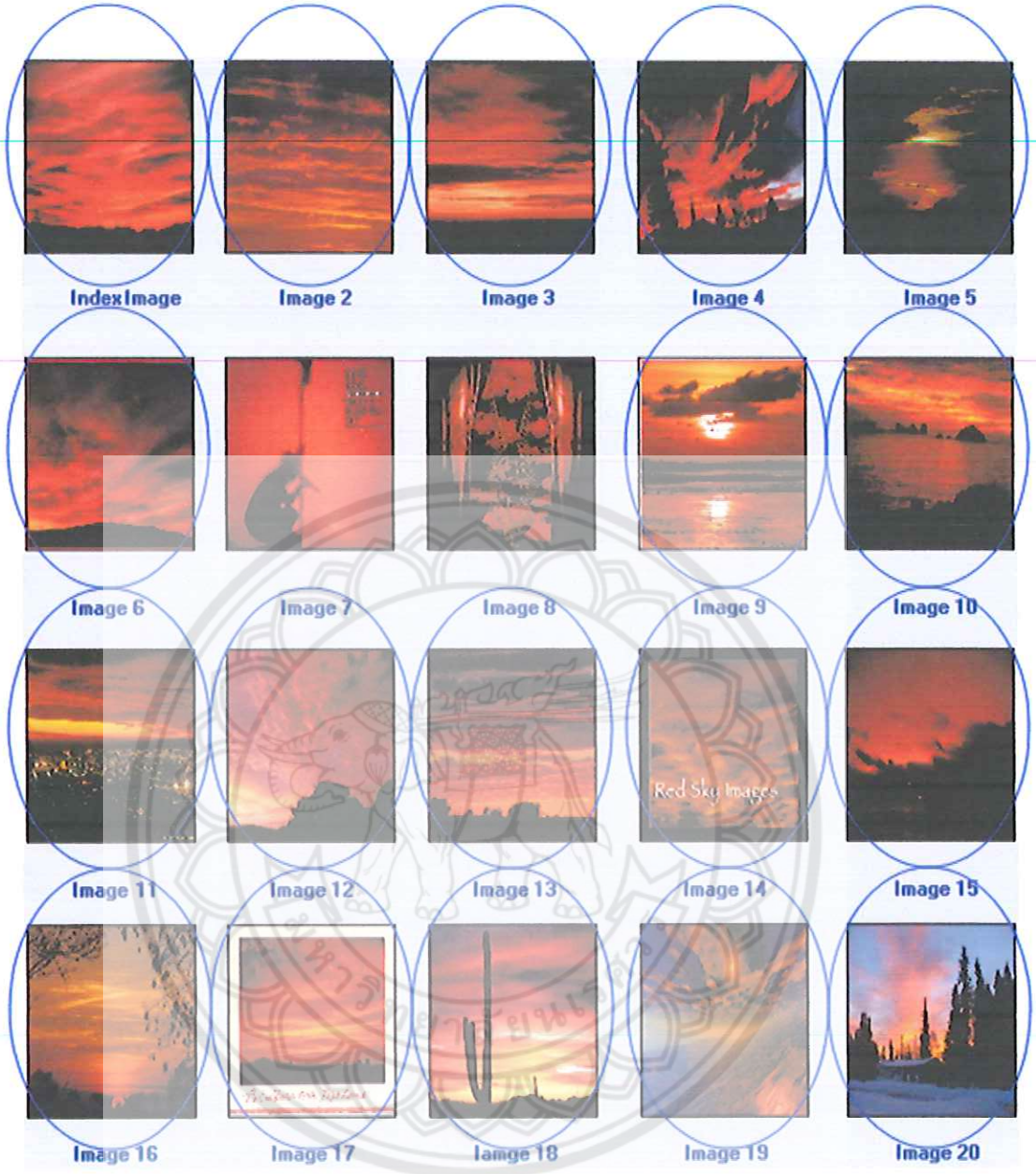
นำรูปภาพที่ต้องการค้นหาซึ่งได้จากผลการค้นหาในส่วนแรกมาทำการค้นหา โปรแกรมจะทำการเปรียบเทียบค่าองค์ประกอบสีของรูปภาพตัวอย่างกับรูปภาพที่อยู่ในฐานข้อมูล

เมื่อเปรียบเทียบค่าผลต่างขององค์ประกอบสีของรูปภาพแล้วจะทำการQuery โดยให้เรียงจากค่าผลรวมของผลต่างองค์ประกอบสีที่น้อยที่สุดก่อนไปจนถึงค่าผลรามของผลต่างขององค์ประกอบสีที่มากที่สุด

ในการค้นหาข้อมูลภาพจะสามารถค้นหาได้ 3 รูปแบบดังที่กล่าวมาคือ

1) รูปภาพที่อยู่ในกลุ่มเดียวกัน
2) รูปภาพที่อยู่ในประเภทหรือ class เดียวกัน
3) เปรียบเทียบกับรูปภาพทั้งหมดในฐานข้อมูล

ในส่วนของรูปแบบของรูปภาพที่อยู่ในกลุ่มเดียวกันจะแสดงในรูปที่ 4.2 ซึ่งจะเป็นการค้นหาภายในกลุ่มเดียวกันคือ ค้นหาท้องฟ้าที่มีลักษณะเป็นสีแดง



รูปที่ 4.2 ภาพตัวอย่างการค้นหาจากกลุ่มเดียวกันโดย RGB Color Model

จากรูปที่ 4.2 เราหาค่าความถูกต้อง(Precision) ของการค้นหาโดยรูปภาพ จะเห็นได้ว่ารูปที่ถูกต้องที่ถูกเลือกมา 18 รูป

ซึ่งการหาค่า Precision นั้นเราจะกำหนดความสัมพันธ์ของรูปภาพในฐานข้อมูลโดยอาศัยความคล้ายคลึงขององค์ประกอบสี ตามสมการ 4.1

$$\text{จะได้} \quad \%\ \text{ความถูกต้อง} = \left(\frac{18}{20}\right)\times 100 = 90\%$$

ในส่วนของรูปแบบของรูปภาพที่อยู่ในประเภทหรือ Class เดียวกันจะแสดงในรูปที่ 4.3

รูปที่ 4.3 ภาพตัวอย่างการค้นหาจากประเภทหรือClass เดียวกันโดย RGB Color Model

จากรูปที่ 4.3 เราหาค่าความถูกต้อง(Precision) ของการค้นหาโดยรูปภาพ จะเห็นได้ว่ารูปที่ถูกต้องที่ถูกเลือกมา 13 รูป จากสมการ 4.1

$$\text{จะได้} \quad \%\ \text{ความถูกต้อง} = \left(\frac{13}{20}\right)\times 100 = 65\%$$

ในส่วนของการค้นหาโดยการเปรียบเทียบรูปแบบของรูปภาพกับรูปภาพทั้งหมดในฐานข้อมูล จะแสดงในรูปที่ 4.4

รูปที่ 4.4 ภาพตัวอย่างการค้นหาจากรูปภาพทั้งหมดในฐานข้อมูลโดย RGB Color Model

จากรูปที่ 4.4 เราหาค่าความถูกต้อง(Precision) ของการค้นหาโดยรูปภาพ จะเห็นได้ว่ารูปที่ถูกต้องที่ถูกเลือกมา 13 รูป จากสมการ 4.1

$$\text{จะได้} \quad \%\ \text{ความถูกต้อง} = \left(\frac{13}{20}\right) \times 100 = 65\%$$

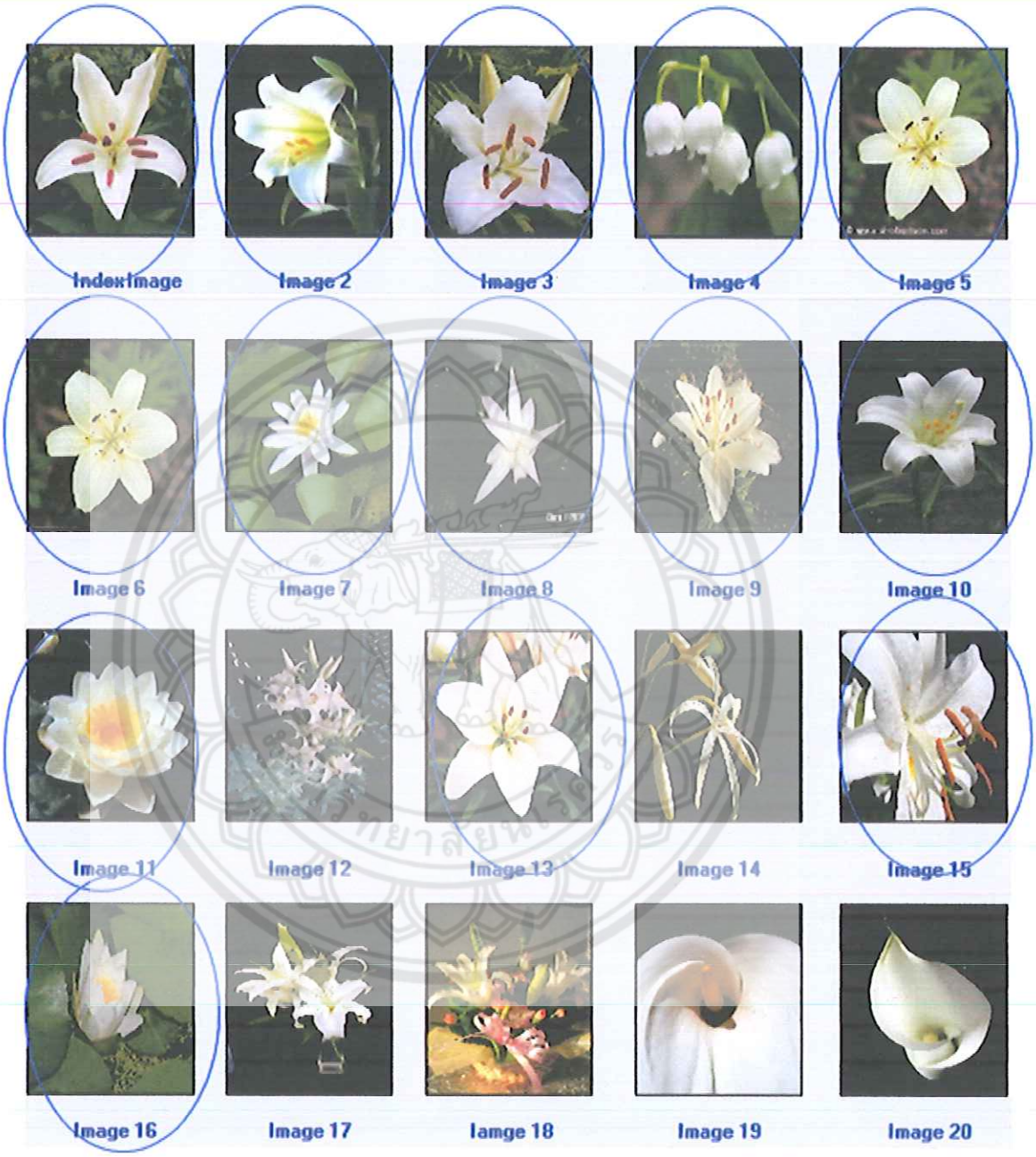
จากผลการทดลองทำการสุ่มตัวอย่างรูปภาพ 30 รูปภาพแล้วทำการค้นโดยใช้ RGB Color modelในการค้นหาแล้วบันทึกค่า Precision ในตาราง 4.1

ตารางที่ 4.1 บันทึกการทดลองค้นหารูปภาพโดยพิจารณาค่าองค์ประกอบสี RGB

ลำดับที่	ชื่อไฟล์รูปภาพ	จำนวนรูปภาพที่ถูกเลือก (จาก 20 ภาพ)	ค่าความถูกต้องการค้นหา (Precision)
1	white lily2.jpg	5	25%
2	white lily3.jpg	9	45%
3	white lily4.jpg	6	30%
4	white lily5.jpg	7	35%
5	white lily34.jpg	3	15%
6	white lily35.jpg	5	25%
7	gray cat4.jpg	7	35%
8	gray cat13.jpg	6	30%
9	gray cat27.jpg	6	30%
10	mazda31.jpg	6	30%
11	mazda32.jpg	9	45%
12	mazda39.jpg	10	50%
13	mazda323.jpg	9	45%
14	mazda317.jpg	9	45%
15	mazda329.jpg	12	60%
16	orange lily1.jpg	11	55%
17	orange lily6.jpg	17	85%
18	orange lily11.jpg	14	70%
19	orange lily22.jpg	10	50%
20	orange lily27.jpg	18	90%
21	orange lily33.jpg	13	65%
22	orange lily42.jpg	7	35%
23	white calla1.jpg	9	45%
24	white calla2.jpg	10	50%
25	white calla11.jpg	9	45%
26	white calla34.jpg	8	40%
27	vaio laptop1.jpg	10	50%
28	vaio laptop23.jpg	11	55%
29	vaio laptop24.jpg	12	60%
30	vaio laptop26.jpg	8	40%
SUMATION		274	1370%
AVERAGE		9.13	45.65%

4.2.3 HSV Color Histogram (Quantize HSV Color Histogram to 18 *3*3 levels)

ในการค้นหาก็จะทำในลักษณะเดียวกับการค้นหารูปภาพโดยใช้วิธี Quantize RGB Color Histogram to 8 levels

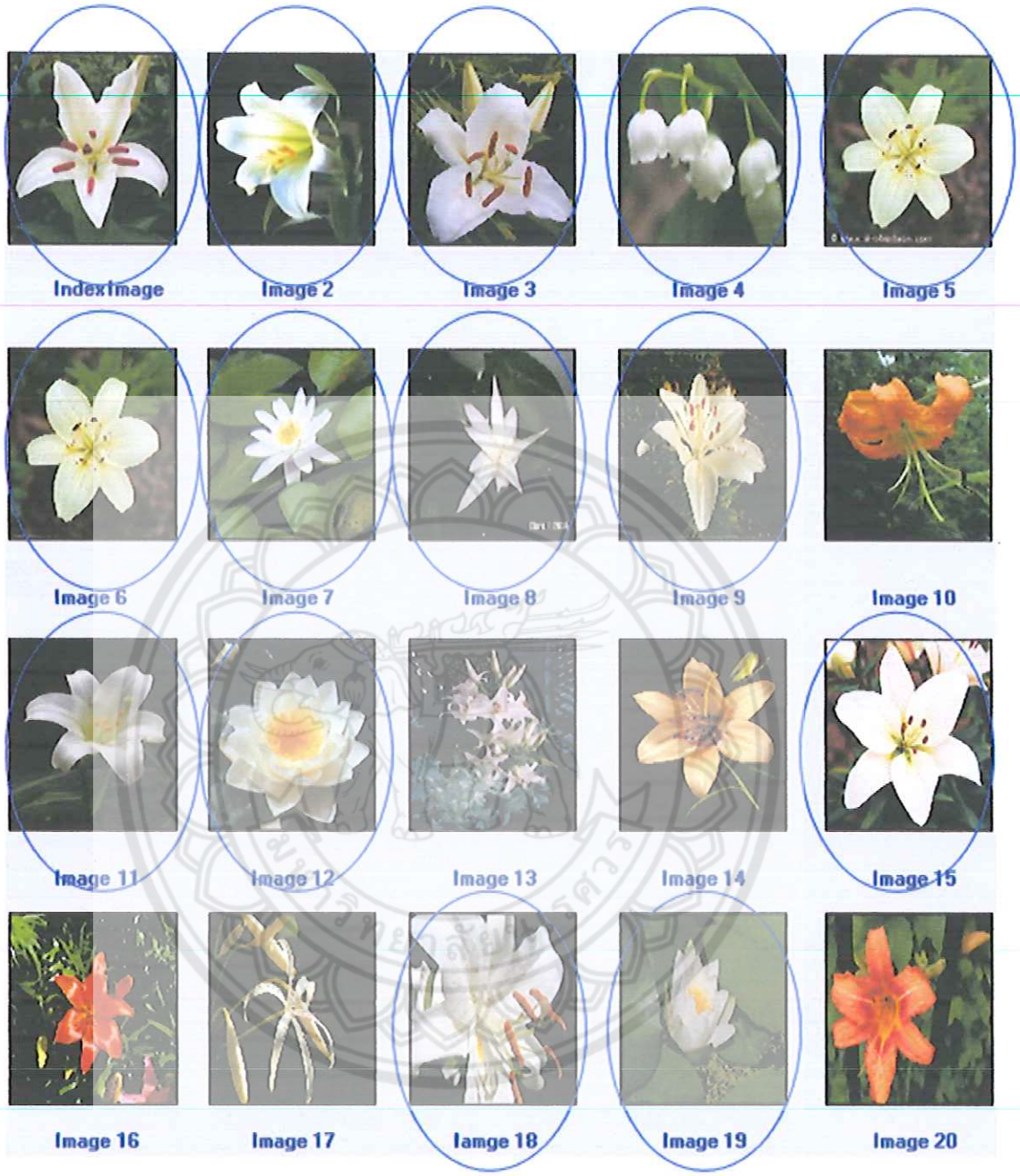


รูปที่ 4.5 ภาพตัวอย่างการค้นหาจากกลุ่มเดียวกันโดย HSV Color Model

จากรูปที่ 4.5 เราหาค่าความถูกต้อง(Precision) ของการค้นหาโดยรูปภาพ จะเห็นได้ว่ารูปที่ถูกต้องที่ถูกเลือกมา 16 รูปจะได้ จากสมการ 4.1

$$\% \text{ ความถูกต้อง} = \left(\frac{14}{20}\right) \times 100 = 70\%$$

ในส่วนของความสัมพันธ์แบบรูปภาพที่อยู่ในประเภทหรือ Class เดียวกันจะแสดงในรูปที่4.6



รูปที่ 4.6 ภาพตัวอย่างการค้นหาจากประเภทหรือClass เดียวกันโดย HSV Color Model

จากรูปที่ 4.6 เราหาค่าความถูกต้อง(Precision) ของการค้นหาโดยรูปภาพ จะเห็นได้ว่ารูปที่ถูกต้องที่ถูกเลือกมา 14 รูป

จะได้ $\%\ \text{ความถูกต้อง} = \left(\frac{14}{20}\right) \times 100 = 70\%$

ในส่วนของการค้นหาโดยการเปรียบเทียบกับรูปภาพทั้งหมดในฐานข้อมูล จะแสดงในรูปที่4.7

รูปที่ 4.7 รูปภาพตัวอย่างการค้นหาจากรูปภาพทั้งหมดในฐานข้อมูลโดย HSV Color Model

จากรูปที่ 4.7 เราหาค่าความถูกต้อง(Precision) ของการค้นหาโดยรูปภาพ จะเห็นได้ว่ารูปที่ถูกต้องที่ถูกเลือกมา 15 รูป จากสมการ 4.1

$$\text{จะได้} \quad \%\ \text{ความถูกต้อง} = \left(\frac{15}{20}\right) \times 100 = 75\%$$

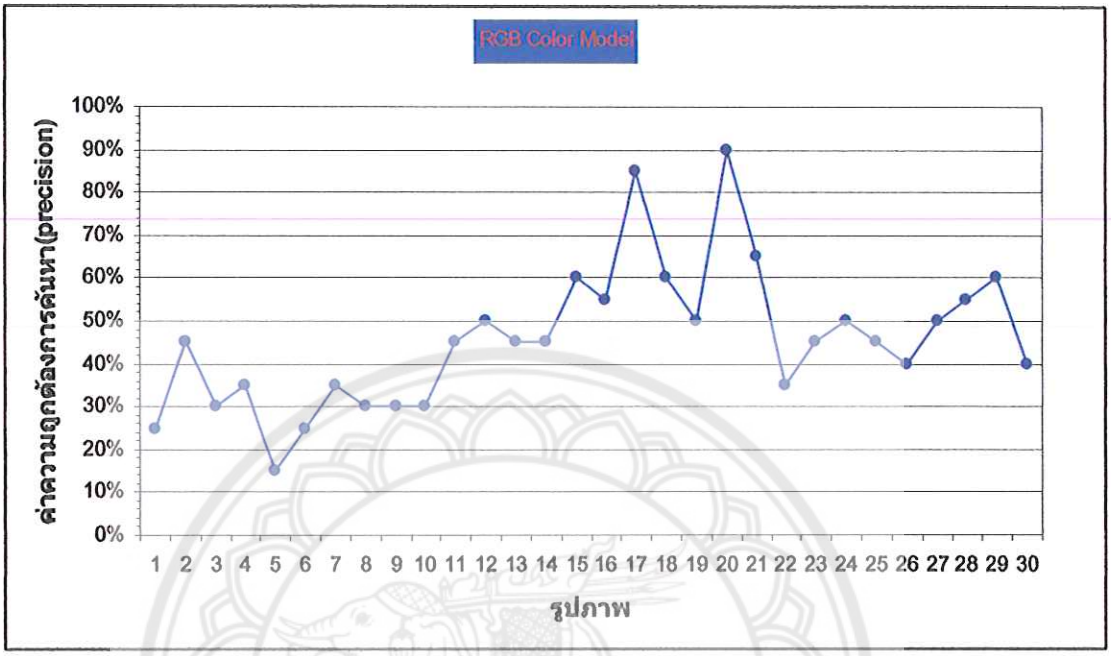
จากผลการทดลองทำการสุ่มตัวอย่างรูปภาพ 30 รูปภาพแล้วทำการค้นหาโดยใช้ HSV Color modelในการค้นหาแล้วบันทึกค่า Precision ในตาราง 4.2

ตารางที่ 4.2 บันทึกการทดลองค้นหารูปภาพโดยพิจารณาค่าองค์ประกอบสี HSV

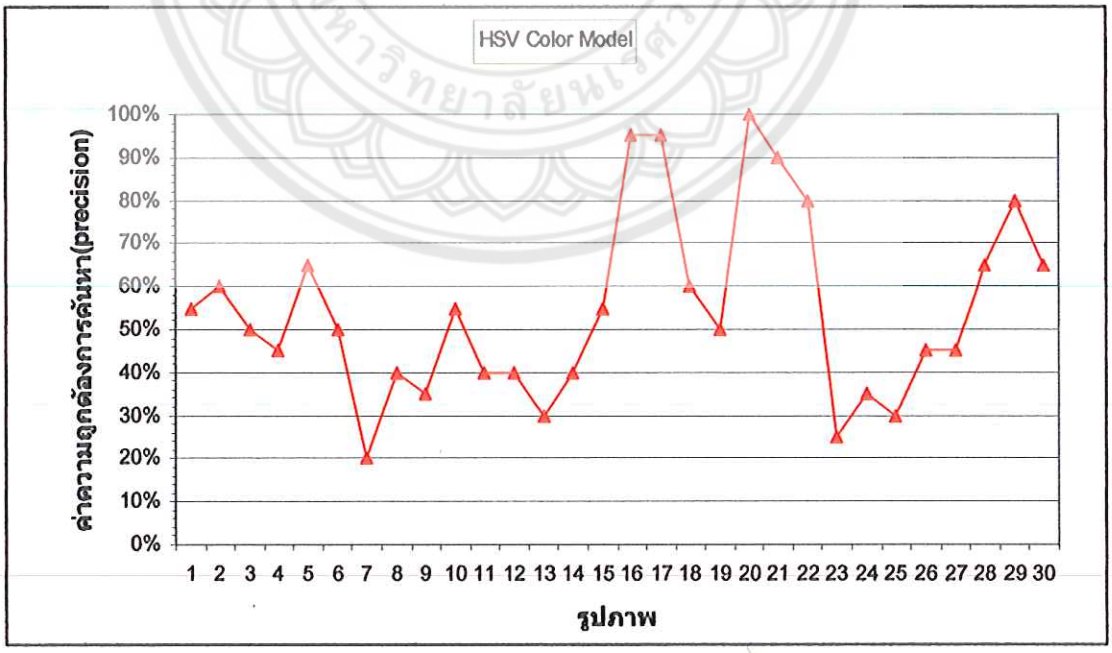
ลำดับที่	ชื่อไฟล์รูปภาพ	จำนวนรูปภาพที่ถูกเลือก (จาก 20 ภาพ)	ค่าความถูกต้องการค้นหา (Precision)
1	white lily2.jpg	11	55%
2	white lily3.jpg	12	60%
3	white lily4.jpg	10	50%
4	white lily5.jpg	7	45%
5	white lily34.jpg	11	65%
6	white lily35.jpg	10	50%
7	gray cat4.jpg	4	20%
8	gray cat13.jpg	8	40%
9	gray cat27.jpg	7	35%
10	mazda31.jpg	11	55%
11	mazda32.jpg	8	40%
12	mazda39.jpg	8	40%
13	mazda323.jpg	6	30%
14	mazda317.jpg	8	40%
15	mazda329.jpg	11	55%
16	orange lily1.jpg	19	95%
17	orange lily6.jpg	19	95%
18	orange lily11.jpg	12	60%
19	orange lily22.jpg	10	50%
20	orange lily27.jpg	20	100%
21	orange lily33.jpg	18	90%
22	orange lily42.jpg	16	80%
23	white calla1.jpg	5	25%
24	white calla2.jpg	7	35%
25	white calla11.jpg	6	30%
26	white calla34.jpg	9	45%
27	vaio laptop1.jpg	9	45%
28	vaio laptop23.jpg	13	65%
29	vaio laptop24.jpg	16	80%
30	vaio laptop26.jpg	13	65%
SUMATION		414	1640%
AVERAGE		13.8	69%

4.3 กราฟแสดงค่าความถูกต้องของการค้นหาทั้ง 2 วิธีจากผลการทดลอง

จากตาราง 4.1และ 4.2 เมื่อเรานำมาพล็อตกราฟจะได้ดังรูปกราฟที่ 4.8 และ 4.9 ตามลำดับ



รูปที่ 4.8 รูปกราฟแสดงค่าความถูกต้องของการค้นหารูปภาพโดยใช้ RGB Color Model



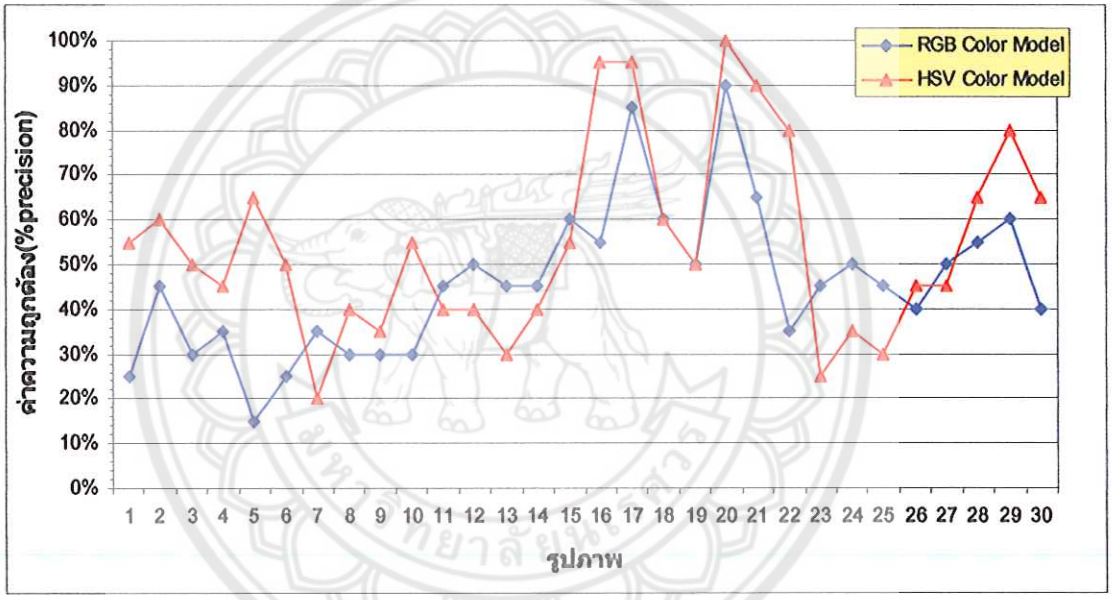
รูปที่ 4. 9 รูปกราฟแสดงค่าความถูกต้องของการค้นหารูปภาพโดยใช้ HSV Color Model

4.4 เปรียบเทียบผลการทดลอง

จากการที่เราสุ่มตัวอย่างรูปภาพทั้งหมด30รูป ที่ได้มาจากการค้นหาในส่วนแรกโดยใช้หลักการ Meta data search มาทำการค้นหาโดยใช้หลักการ Content base image retrieval ซึ่งแบ่งเป็น 2 Model คือ

1. RGB Color Model
2. HSV Color Model

เมื่อทำการค้นหาและบันทึกแล้วจึงนำผลการทดลองหาค่าความถูกต้องของการค้นหา (Precision) ของรูปภาพที่สุ่มตัวอย่างแต่ละรูปภาพมาทำการพล๊อตเป็นกราฟเปรียบเทียบการค้นหาทั้ง 2 วิธี จะเห็นความแตกต่างของการทดลองได้จากรูปกราฟที่ 4.8



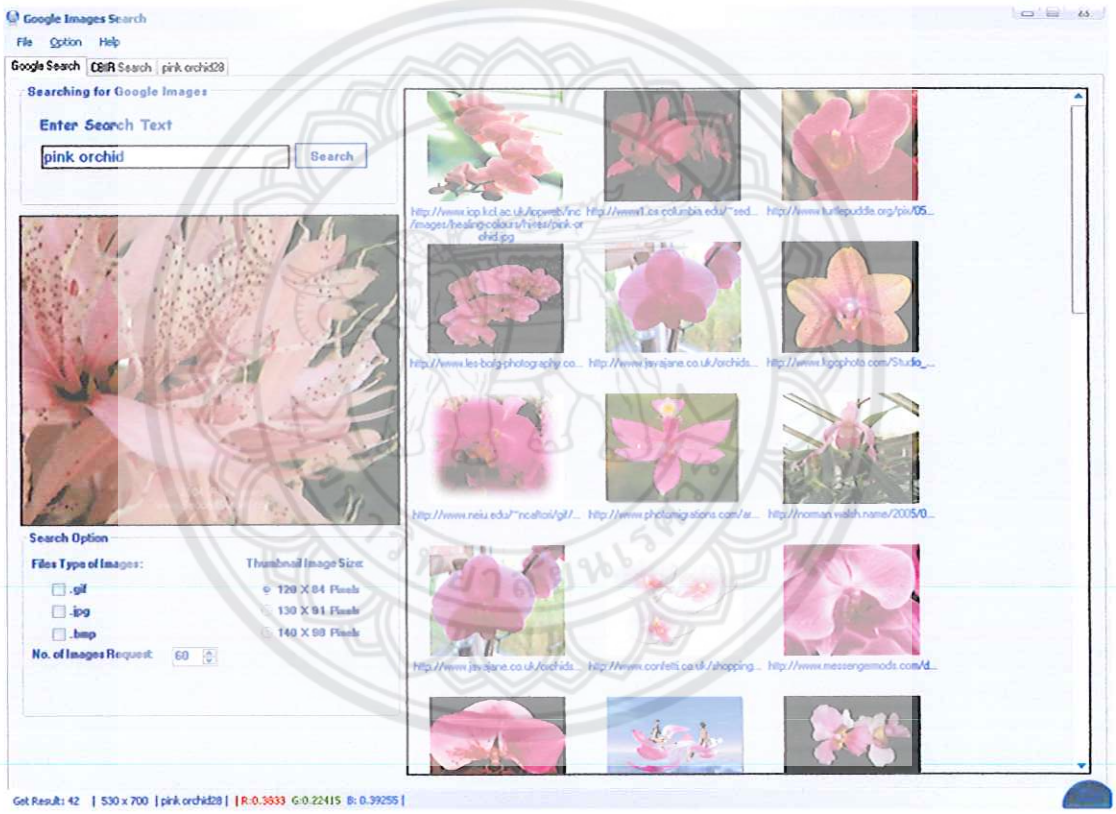
รูปที่ 4.10 รูปกราฟเปรียบเทียบค่าความถูกต้องการค้นหาของทั้ง 2วิธี

เมื่อสังเกตผลการเปรียบเทียบค่าความถูกต้องของการค้นหาแล้ว จะสังเกตเห็นความแตกต่างของทั้ง2วิธี ได้อย่างชัดเจนว่าค่าความถูกต้องของการค้นหาโดยใช้ HSV Color Model ภาพที่ใกล้เคียงกันและองค์ประกอบของภาพที่มีความใกล้เคียงกันจะถูกเลือกมามากกว่าการค้นหาโดยใช้ RGB Color Model

บทที่ 5
บทสรุป

5.1 สรุปผลการทดลอง

ผลการค้นหารูปภาพบนอินเตอร์เน็ตโดยใช้หลักการของ Meta data search จะเห็นได้ว่ารูปภาพที่ได้นั้นอาจจะไม่ตรงตามความต้องการของเรามากนักซึ่งผลที่ได้นั้นอาจจะไม่อยู่ในโทนสีเดียวกันทั้งหมดดังรูปที่ 5.1



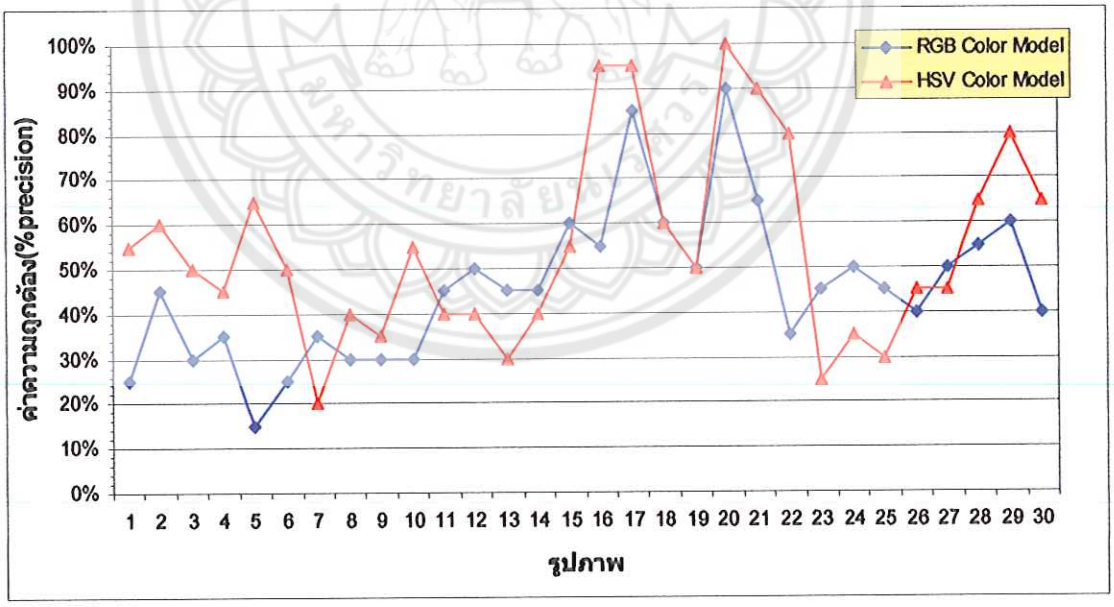
รูปที่ 5.1 ภาพผลการค้นหารูปภาพบนอินเตอร์เน็ตโดยใช้หลักการของ Meta data search

ดังนั้นเราจึงสร้าง Application ในส่วนของการค้นหาโดยใช้หลักการของ Content base image retrieval จากนั้นเราจะทำการทดลองสุ่มตัวอย่างรูปภาพทั้งหมด30รูป ที่ได้มาจากการค้นหาในส่วนแรกที่ใช้หลักการ Meta data search มาทำการค้นหาโดยใช้หลักการ Content base image retrieval

เราจะได้ผลการทดลองดังตารางที่ 4.1และ 4.2 เมื่อเปรียบเทียบค่าความถูกต้องในการค้นหาทั้ง 2 วิธี ทำให้เราสามารถสรุปได้ว่า

- การค้นหาจากไฟล์รูปภาพโดยใช้หลักการของ Content base image retrieval จะสามารถพบไฟล์รูปภาพที่มีองค์ประกอบภายในของรูปภาพคล้ายกับไฟล์รูปภาพที่นำมาเป็นตัวอย่างในการค้นหาโดยภาพที่ถูกเลือกอาจจะมีภาพที่ถูกต้องตรงตามความต้องการของเรา ซึ่งจะเห็นว่าการค้นหาโดยใช้ RGB Color Model ช่วยในการวิเคราะห์องค์ประกอบสีของไฟล์รูปภาพจะเห็นว่าเปอร์เซ็นต์ค่าความถูกต้องของการค้นหาเฉลี่ยแล้วจะอยู่ในระดับปานกลางเมื่อเทียบกับการค้นหาโดยใช้ HSV Color Model ช่วยในการวิเคราะห์องค์ประกอบสีของไฟล์รูปภาพ ซึ่งผลการทดลองจะแสดงดังตารางที่ 4.1

- การค้นหาโดยใช้ HSV Color Model ช่วยในการวิเคราะห์องค์ประกอบสีของไฟล์รูปภาพ จากผลการทดลองแล้วจะเห็นว่าเปอร์เซ็นต์ค่าความถูกต้องของการค้นหาเฉลี่ยจะอยู่ในระดับที่ดีเมื่อเทียบกับการค้นหาโดยใช้ RGB Color Model ช่วยในการวิเคราะห์องค์ประกอบสีของไฟล์รูปภาพซึ่งผลการทดลองจะแสดงดังตารางที่ 4.2 เนื่องจากจะมีการวิเคราะห์องค์ประกอบอื่นๆ อีกนอกจากองค์ประกอบสีเพียงอย่างเดียว เช่น ค่าความสว่าง และค่าความอิ่มตัว เป็นต้น ดังนั้นจึงสรุปได้ว่าการหาโดยใช้ HSV Color Model ช่วยในการวิเคราะห์องค์ประกอบสีของไฟล์รูปภาจะมีประสิทธิภาพในการค้นหาที่ดีกว่าและตรงตามความต้องการมากกว่า การค้นหาโดยใช้ RGB Color Model ช่วยในการวิเคราะห์องค์ประกอบสีของไฟล์รูปภาพ ดังแสดงให้เห็นดังรูปกราฟที่ 5.2



รูปที่ 5.2 รูปกราฟเปรียบเทียบค่าความถูกต้องการค้นหาของทั้ง 2 วิธี

เมื่อเราพิจารณาและเปรียบเทียบค่าเฉลี่ยของความถูกต้องของการค้นหา(Precision) ในแต่ละวิธีจะได้ดังตารางที่ 5.1

Methods	Average Precision
RGB Color Model	45.65%
HSV Color Model	69%

ตารางที่ 5.1 เปรียบเทียบค่า Average Precision ของ 2 วิธี

จะเห็นได้ว่า ค่าเฉลี่ยความถูกต้องในการค้นหาโดยใช้ RGB Color Modelจะมีค่าน้อยกว่าการค้นหาแบบการใช้ HSV Color Model เนื่องจากจะมีการวิเคราะห์องค์ประกอบอื่นๆอีกนอกจากองค์ประกอบสีเพียงอย่างเดียว เช่น ค่าSector ของแต่ละสี, ค่าความสว่าง, และค่าความอิ่มตัว

5.2 ปัญหาที่พบ

5.2.1 **Application** นี้ ในส่วนของการค้นหาโดยหลักการ Meta data search นั้นเมื่อเราทำการร้องขอบริการจาก Google และได้ URL ของรูปภาพแล้วแต่บางครั้งเราอาจจะ access URL นั้นไม่ได้ดั้งนั้นเราก็ไม่สามารถดึงภาพนั้นมาได้

5.2.2 ในส่วนที่ 2 ของ Application นี้เราใช้หลักการ Content Base Image Retrieval โดยจะใช้ Model ของสี 2 Model ด้วยกัน คือ RGB Color Model และ HSV Color Model ข้อผิดพลาดของการค้นหาอาจจะเกิดได้ เนื่องจากบางรูปภาพค่าองค์ประกอบของสีจะมีค่าใกล้เคียงกันแต่ลักษณะรูปร่างอาจจะแตกต่างกัน ซึ่งใน Application นี้เราไม่ได้ใช้หลักการของ Shape, Texture Analysis เพื่อเข้ามาช่วยวิเคราะห์

5.2.3 การหาค่าความถูกต้องของการค้นหา เมื่อเราทำการค้นหาในกลุ่มเดียวกันจะเห็นว่าภาพทุกภาพที่อยู่ในกลุ่มนั้นจะได้รับความสนใจและจะถูกเลือกมาทั้งหมด แต่ในบางครั้งค่าความถูกต้อง (precision) ซึ่งจะแสดงค่าความถูกต้องออกมาอาจจะไม่สมบูรณ์มากนัก ทั้งนี้อาจจะเป็นเพราะรูปภาพบางรูปอาจจะมีค่าขององค์ประกอบสีที่ไม่ใกล้เคียงกับภาพต้นแบบจึงถือว่าค่าองค์ประกอบสีของรูปภาพนั้นไม่มีความสัมพันธ์กันรูปภาพต้นแบบ

5.3 ข้อเสนอแนะ

5.3.1 ต้องสร้าง Algorithm ในส่วนที่ช่วยตรวจสอบความถูกต้องของ URL ของรูปภาพ

5.3.2 ในการค้นหาแต่ละครั้ง ในบางครั้งผลการค้นหานั้นเราอาจจะไม่ได้รูปภาพที่ตรงตามความต้องการหรือคล้ายคลึงกับภาพต้นแบบมากนัก เนื่องจากApplication นี้เราใช้การเปรียบเทียบความเหมือนของค่าองค์ประกอบสี RGB และ HSV เป็นหลัก ซึ่งเป็นการหาค่าของผลต่างของค่าองค์ประกอบสีของรูปต้นแบบกับรูปเป้าหมายโดยไม่ได้คำนึงถึงว่ารูปภาพเป้าหมายจะมีลักษณะรูปร่างคล้ายคลึงหรือแตกต่างกันกับรูปภาพต้นแบบหรือไม่ ดังนั้นหากมีการพัฒนาApplication นี้ต่อไป ควรจะศึกษาในส่วนของ Texture และ Shapes Analysis เพิ่มเติม เพื่อจะนำมาช่วยในการวิเคราะห์ค่าองค์ประกอบของรูปภาพในส่วนอื่นๆนอกเหนือจากองค์ประกอบสีเพียงอย่างเดียว เพื่อจะทำให้การค้นหามีประสิทธิภาพมากขึ้น

เอกสารอ้างอิง

[1] Sangoh Jeong. **"Histogram-Base Color Image Retrieval."** [Online]. Available: http://scien.stanford.edu/class/psych221/projects/02/sojeong/

[2] A.M. Kuchling . **"Regular Expression HOWTO."** [Online]. Available: http://www.amk.ca/python/howto/regex/

[3] Dan Fernandez. **"Pulling Images from Google using C# Express."** [Online]. Available: http://blogs.msdn.com/danielfe/archive/2004/07/26/197811.aspx

[4] Data-Compression.com. **"Vector Quantization." [Online]**. Available: http://www.data-compression.com/vq.shtml

ภาคผนวก ก

Source Code ติดต่อกับ Google

ก-1 Source Code ที่ทำการติดต่อเพื่อร้องขอบริการจาก Google ให้ทำการค้นหาข้อมูลรูปภาพโดยคำอธิบายที่ต้องการค้นหา

```
public static List<string> GetImagesFromGoogle(string search,int
requestResult, int page,int count )
 {
   List<string> urlList = new List<string>();

    for(count = 0; count < requestResult; count+= RESULT_PER_QUERY)
    {
    string googleUrl=string.Format("Http://images.google.co.th/images?q="
      + search +"&svnum=10&hl=en&lr=&start=" + Convert.ToString(page
      + count));

    string html = GetHtml(googleUrl);
    Regex r = new Regex(@"(\x3Ca\s+href=/imgres\x3Fimgurl=)
   (?<imgurl>http[^&>]*)([>&]{1})([^>]*)(>{1})(<img\ssrc\x3D)(""{0,1})
   (?<images>/images[^>\s]*)([\s])+(width=)(?<width>[09,]*)\s+(height=)
   (?<height>[0-9,]*)");

     MatchCollection matches = r.Matches(html);
       for (int n = 0; n < matches.Count; n++)
       {
                //Provide full path for image
          string imageUrl = @matches[n].Groups["imgurl"].Value;
                urlList.Add(imageUrl);
        }

     }

        return urlList;
 }
```

ก-2 Source Code ที่ทำการอ่านค่าDefault proxy

```
public static Stream GetStreamFromUrl(string url)
 {

      System.Net.WebProxy wp = System.Net.WebProxy.GetDefaultProxy();
      wp.Credentials = System.Net.CredentialCache.DefaultCredentials;
      System.Net.GlobalProxySelection.Select = wp;

      return new WebClient().OpenRead(url);

  }
```

ก-3 Source Code ที่ทำการอ่านค่าข้อมูลรูปภาพจาก URL

```
int requestResult = Decimal.ToInt32(imageRequest.Value);
int page = 0;
int count = 0;
List<string> imgUrls
=WebUtilities.GetImagesFromGoogle((string)e.Argument,
requestResult,page,count);

            //Load images
            Image img = null;

foreach (string url in imgUrls)
{
  try
   {
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
    request.Method = "GET";
    request.Timeout = 999999999;
    request.ProtocolVersion = HttpVersion.Version11;
using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
{
 using (Stream responseStream = response.GetResponseStream())
  {
    img = Image.FromStream(responseStream);
    if (count < requestResult)
    {
      img.Save("C:\\Images\\" + txtSearch.Text + Convert.ToString(i + 1)
      + ".jpg");

       UrlAndImage.Add(url, img);
       lbResult.Text = (" Get Result: " + Convert.ToString(i + 1));
       responseStream.Close();
       img_result += 1;

      }

   }

 }     count = count + 1;
       i = i + 1;
       img = null;


}
catch
 {
 }
finally
 {
   googleBackgroundWorker.ReportProgress(progress++);
 }

}
        //Send result back to Form
      e.Result = UrlAndImage;
```

ก-4 การหาค่า Color histogram แล้วนำมาทำ vector ของสี RGB โดยแต่ละค่าสีจะเก็บทั้งหมด 8 bins และแต่ละรูปจะมี vector 24 bins ที่เกิดจากการเอา ค่าvector ของ สี แดง, เขียว, น้ำเงิน มาต่อกัน

```
string stimg_id = string.Empty;
string location = string.Empty;
string stmax_id = string.Empty;
string sqlcmd = string.Empty;
img_tag = getText;
double rmean = 0; double gmean = 0; double bmean = 0; double resultR = 0;
double resultG=0; double resultB=0; double r = 0; double g = 0;
double b = 0;double b1 = 0, b2 = 0, b3 = 0, b4 = 0, b5 = 0, b6 = 0,
b7 = 0, b8 = 0, b9 = 0, b10 = 0, b11 = 0, b12 = 0;double b13 = 0,
b14 = 0, b15 = 0, b16 = 0, b17 = 0, b18 = 0, b19 = 0, b20 = 0, b21 = 0,
b22 = 0, b23 = 0, b24 = 0;
int x = 0; int y = 0; int pixels; int n_count = 0; int group_id = 1;
byte Hgreen;byte Hred;
byte Hblue;
byte pixelColorRed;
byte pixelColorGreen;
byte pixelColorBlue;

Bitmap pic_his = null;
pic_his = new Bitmap("C:\\Images\\ThumbImages\\" + getText +
Convert.ToString(n_count + 1) + ".jpg");

int height = pic_his.Height;
int width = pic_his.Width;
pixels = width * height;

      for (y = 0; y < height; y++)
      {
            for (x = 0; x < width; x++)
            {

                  Hred = pic_his.GetPixel(x, y).R;
                  double rpoint = Convert.ToDouble(Hred);
                  resultR = resultR + rpoint;

                  Hgreen = pic_his.GetPixel(x, y).G;
                  double gpoint = Convert.ToDouble(Hgreen);
                  resultG = resultG + gpoint;

                  Hblue = pic_his.GetPixel(x, y).B;
                  double bpoint = Convert.ToDouble(Hblue);
                  resultB = resultB + bpoint;
```

```
#region Quatize RGB Histogram
//*********Quantize Red Color
          pixelColorRed = pic_his.GetPixel(x, y).R;

          if (pixelColorRed >= 0 && pixelColorRed < 32)
           {
              b1 = b1 + 1;
           }
          else if (pixelColorRed >= 32 && pixelColorRed < 64)
           {
              b2 = b2 + 1;
           }
          else if (pixelColorRed >= 64 && pixelColorRed < 96)
           {
              b3 = b3 + 1;
           }
          else if (pixelColorRed >= 96 && pixelColorRed < 128)
           {
              b4 = b4 + 1;
           }
          else if (pixelColorRed >= 128 && pixelColorRed < 160)
           {
              b5 = b5 + 1;
           }
          else if (pixelColorRed >= 160 && pixelColorRed < 192)
           {
              b6 = b6 + 1;
           }
          else if (pixelColorRed >= 192 && pixelColorRed < 224)
           {
              b7 = b7 + 1;
           }
          else if(pixelColorRed >= 224)
           {
              b8 = b8 + 1;
           }

//*********Quantize Green Color
          pixelColorGreen = pic_his.GetPixel(x, y).G;

          if (pixelColorGreen >= 0 && pixelColorGreen < 32)
          {
             b9 = b9 + 1;
          }
          else if (pixelColorGreen >= 32 && pixelColorGreen < 64)
          {
             b10 = b10 + 1;
          }
          else if (pixelColorGreen >= 64 && pixelColorGreen < 96)
          {
             b11 = b11 + 1;
          }
          else if (pixelColorGreen >= 96 && pixelColorGreen < 128)
          {
             b12 = b12 + 1;
          }
```

```
            else if (pixelColorGreen >= 128 && pixelColorGreen < 160)
            {
               b13 = b13 + 1;
            }
            else if (pixelColorGreen >= 160 && pixelColorGreen < 192)
            {
               b14 = b14 + 1;
            }
            else if (pixelColorGreen >= 192 && pixelColorGreen < 224)
            {
               b15 = b15 + 1;
            }
            else if (pixelColorGreen >= 224)
            {
               b16 = b16 + 1;
            }

//*********Quantize Blue Color
            pixelColorBlue = pic_his.GetPixel(x, y).B;

            if (pixelColorBlue >= 0 && pixelColorBlue < 32)
            {
               b17 = b17 + 1;
            }
            else if (pixelColorBlue >= 32 && pixelColorBlue < 64)
            {
               b18 = b18 + 1;
            }
            else if (pixelColorBlue >= 64 && pixelColorBlue < 96)
            {
               b19 = b19 + 1;
            }
            else if (pixelColorBlue >= 96 && pixelColorBlue < 128)
            {
               b20 = b20 + 1;
            }
            else if (pixelColorBlue >= 128 && pixelColorBlue < 160)
            {
               b21 = b21 + 1;
            }
            else if (pixelColorBlue >= 160 && pixelColorBlue < 192)
            {
               b22 = b22 + 1;
            }
            else if (pixelColorBlue >= 192 && pixelColorBlue < 224)
            {
               b23 = b23 + 1;
            }
            else if (pixelColorBlue >= 224)
            {
               b24 = b24 + 1;
            }

            #endregion
      }

}
```

ก-5 การทำNormalize Color histogram ของสี RGB

```
rmean = (resultR / pixels);
gmean = (resultG / pixels);
bmean = (resultB / pixels);
r = (rmean / (rmean + gmean + bmean));
g = (gmean / (rmean + gmean + bmean));
b = (bmean / (rmean + gmean + bmean));

string rVal = r.ToString("F5");
string gVal = g.ToString("F5");
string bVal = b.ToString("F5");
rmean = 0; gmean = 0; bmean = 0;

b1  = b1  / pixels; b7  = b7  / pixels; b13 = b13 / pixels;
b19 = b19 / pixels; b2  = b2  / pixels; b8  = b8  / pixels;
b14 = b14 / pixels; b20 = b20 / pixels; b3  = b3  / pixels;
b9  = b9  / pixels; b15 = b15 / pixels; b21 = b21 / pixels;
b4  = b4  / pixels; b10 = b10 / pixels; b16 = b16 / pixels;
b22 = b22 / pixels; b5  = b5  / pixels; b11 = b11 / pixels;
b17 = b17 / pixels; b23 = b23 / pixels; b6  = b6  / pixels;
b12 = b12 / pixels; b18 = b18 / pixels; b24 = b24 / pixels;
```

ก-6 การนำค่า Vector ที่หาได้ไปใส่ในฐานข้อมูล

```
OleDbConnection Conn = new OleDbConnection();
Conn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Application.StartupPath + "\\GoogleImage.mdb";
Conn.Open();

sqlcmd = "INSERT INTO
[Images](location,rValue,gValue,bValue,img_tag,group_id)" +
" VALUES('" + location + "','" + Convert.ToDouble(rVal) +
"','" + Convert.ToDouble(gVal) + "','" + Convert.ToDouble(bVal) +
"','" + getText + "','" + group_id + "')";
OleDbCommand command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();

sqlcmd = "INSERT INTO
[S_Images](location,img_tag,group_id,v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v
12,v13,v14,v15,v16,v17,v18,v19,v20,v21,v22,v23,v24)" +
" VALUES('" + location + "','" + getText + "','" + group_id + "','" +
b1.ToString("F5") + "','" + b2.ToString("F5") + "','" +
b3.ToString("F5") + "','" + b4.ToString("F5") + "','" +
b5.ToString("F5") + "','" + b6.ToString("F5") + "','" +
b7.ToString("F5") + "','" + b8.ToString("F5") + "','" +
b9.ToString("F5") + "','" + b10.ToString("F5") +"','" +
b11.ToString("F5") + "','" + b12.ToString("F5") + "','" +
b13.ToString("F5") + "','" + b14.ToString("F5") + "','" +
b15.ToString("F5") + "','" + b16.ToString("F5") + "','" +
b17.ToString("F5") + "','" + b18.ToString("F5") + "','" +
b19.ToString("F5") + "','" + b20.ToString("F5") + "','" +
b21.ToString("F5") + "','" + b22.ToString("F5") + "','" +
b23.ToString("F5") + "','" + b24.ToString("F5") + "')";
command = new OleDbCommand(sqlcmd, Conn)
command.ExecuteNonQuery();
```

ก-7 การหาค่า Color histogram แล้วนำมาทำ vector ของสี HSV โดยค่า H จะเก็บทั้งหมด 18 Bins ค่า S จะเก็บทั้งหมด 3 bins และ V จะเก็บทั้งหมด 3 แต่ละรูปจะมี vectors 24 bins ที่เกิดจากการเอา ค่าvector ของ สี แดง, เขียว, น้ำเงิน มาต่อกัน

```
string stimg_id = string.Empty;
string location = string.Empty;
string stmax_id = string.Empty;
string sqlcmd = string.Empty;
img_tag = getText;
double resultH = 0;double resultV = 0;double resultS = 0;double Hmean= 0;
double Smean = 0; double Vmean = 0; double Hmean = 0;
double b = 0;double b1 = 0, b2 = 0, b3 = 0, b4 = 0, b5 = 0, b6 = 0,
b7 = 0, b8 = 0, b9 = 0, b10 = 0, b11 = 0, b12 = 0;double b13 = 0,
b14 = 0, b15 = 0, b16 = 0, b17 = 0, b18 = 0, b19 = 0, b20 = 0, b21 = 0,
b22 = 0, b23 = 0, b24 = 0;
int x = 0; int y = 0; int pixels; int n_count = 0; int group_id = 1;
float HS;float HH; float HV;
double pixelHue = 0;
double pixelSaturation=0;
double pixelValue=0;
double HueValue = 0

Bitmap pic_his = null;
pic_his = new Bitmap("C:\\Images\\ThumbImages\\" + getText +
Convert.ToString(n_count + 1) + ".jpg");

int height = pic_his.Height;
int width = pic_his.Width;
pixels = width * height;

      for (y = 0; y < height; y++)
      {
            for (x = 0; x < width; x++)
            {

            HH = pic_his.GetPixel(x, y).GetHue();
            double hpoint = (Convert.ToDouble(HH));
            resultH = resultH + hpoint;


            HS = pic_his.GetPixel(x, y).GetSaturation();
            double spoint = (Convert.ToDouble(HS));
            resultS = resultS + spoint;


            HV = pic_his.GetPixel(x, y).GetBrightness();
            double vpoint = (Convert.ToDouble(HV));
            resultV = resultV + vpoint;


            pixelHue = pic_his.GetPixel(x, y).GetHue();
            HueValue = (pixelHue / 360);
```

```
//Hue*****************************************
            if (HueValue >= 0 && HueValue < 0.0556)
            {
              b1 = b1 + 1;
            }
            else if (HueValue >= 0.0556 && HueValue < 0.1111)
            {
              b2 = b2 + 1;
            }
            else if (HueValue >= 0.1111 && HueValue < 0.1667)
            {
              b3 = b3 + 1;
            }
            else if (HueValue >= 0.1667 && HueValue < 0.2222)
            {
              b4 = b4 + 1;
            }
            else if (HueValue >= 0.2222 && HueValue < 0.2778)
            {
              b5 = b5 + 1;
            }
            else if (HueValue >= 0.2778 && HueValue < 0.3333)
            {
              b6 = b6 + 1;
            }
            else if (HueValue >= 0.3333 && HueValue < 0.3889)
            {
              b7 = b7 + 1;
            }
            else if (HueValue >= 0.3889 && HueValue < 0.4444)
            {
              b8 = b8 + 1;
            }
            else if (HueValue >= 0.4444 && HueValue < 0.5000)
            {
              b9 = b9 + 1;
            }
            else if (HueValue >= 0.5000 && HueValue < 0.5556)
            {
              b10 = b10 + 1;
            }
            else if (HueValue >= 0.5556 && HueValue < 0.6111)
            {
              b11 = b11 + 1;
            }
            else if (HueValue >= 0.6111 && HueValue < 0.6667)
            {
              b12 = b12 + 1;
            }
            else if (HueValue >= 0.6667 && HueValue < 0.7222)
            {
              b13 = b13 + 1;
            }
            else if (HueValue >= 0.7222 && HueValue < 0.7778)
            {
              b14 = b14 + 1;
            }
```

```
            else if (HueValue >= 0.7778 && HueValue < 0.8333)
            {
               b15 = b15 + 1;
            }
            else if (HueValue >= 0.8333 && HueValue < 0.8889)
            {
               b16 = b16 + 1;
            }
            else if (HueValue >= 0.8889 && HueValue < 0.9444)
            {
               b17 = b17 + 1;
            }
            else if (HueValue >= 0.9444)
            {
               b18 = b18 + 1;
            }
            else if(HueValue==Double.NaN)
            {
               b1 = b1 + 1;
            }

 //Saturation ***********************************
         pixelSaturation = pic_his.GetPixel(x, y).GetSaturation();

         if (pixelSaturation >= 0 && pixelSaturation < 0.3333)
         {
            b19 = b19 + 1;
         }
         else if (pixelSaturation >= 0.3333 && pixelSaturation < 0.6667)
         {
            b20 = b20 + 1;
         }
         else if (pixelSaturation >= 0.6667)
         {
            b21 = b21 + 1;
         }
         else if(pixelSaturation==Double.NaN)
         {
            b19 = b19 + 1;
         }

 //Value ****************************************
         pixelValue = pic_his.GetPixel(x, y).GetBrightness();

         if (pixelValue >= 0 && pixelValue < 0.3333)
         {
            b22 = b22 + 1;
         }
         else if (pixelValue >= 0.3333 && pixelValue < 0.6667)
         {
            b23 = b23 + 1;
         }
         else if(pixelValue >=  0.6667)
         {
            b24 = b24 + 1;
         }
      }
}
```

ก-8 การทำNormalize Color histogram ของสี HSV

```
Hmean = ((resultH / pixels) / 360);
Smean = (resultS / pixels);
Vmean = (resultV / pixels);
string hVal = Hmean.ToString("F5");
string sVal = Smean.ToString("F5");
string vVal = Vmean.ToString("F5");

b1  = b1  / pixels; b7  = b7  / pixels; b13 = b13 / pixels;
b19 = b19 / pixels; b2  = b2  / pixels; b8  = b8  / pixels;
b14 = b14 / pixels; b20 = b20 / pixels; b3  = b3  / pixels;
b9  = b9  / pixels; b15 = b15 / pixels; b21 = b21 / pixels;
b4  = b4  / pixels; b10 = b10 / pixels; b16 = b16 / pixels;
b22 = b22 / pixels; b5  = b5  / pixels; b11 = b11 / pixels;
b17 = b17 / pixels; b23 = b23 / pixels; b6  = b6  / pixels;
b12 = b12 / pixels; b18 = b18 / pixels; b24 = b24 / pixels;
```

ก-9 การนำค่า Vector ที่หาได้ไปUpdate ค่า HSVในฐานข้อมูล

```
OleDbConnection Conn = new OleDbConnection();
Conn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Application.StartupPath + "\\GoogleImage.mdb";
Conn.Open();

sqlcmd = "UPDATE [Images] SET location='" + location + "'," + "hValue='"
+ Convert.ToDouble(hVal) +"'," + "sValue='" + Convert.ToDouble(sVal) +
"'," + "vValue='" + Convert.ToDouble(vVal) +"'," + "img_tag='" +
txt_imgTag.Text + "'," + "group_id=" + group_id + " WHERE(img_id=" +
Convert.ToInt32(stimg_id) +"AND location='" + location + "')";

command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();

sqlcmd = "UPDATE [S_Images] SET location='" + location +"'," +
"img_tag='" + txt_imgTag.Text +"'," + "group_id='" + group_id +
"'," + "v25='" + b1.ToString("F5") + "'," + "v33='" + b9.ToString("F5") +
"'," + "v41='" + b17.ToString("F5") +"'," + "v26='" + b2.ToString("F5") +
"'," + "v34='" + b10.ToString("F5") + "'," + "v42='" +b18.ToString("F5")+
"'," + "v27='" + b3.ToString("F5") + "'," + "v35='" + b11.ToString("F5")+
"'," + "v43='" + b19.ToString("F5") +"'," + "v28='" + b4.ToString("F5") +
"'," + "v36='" + b12.ToString("F5") + "',"+ "v44='"+b20.ToString("F5")+
"'," + "v29='" + b5.ToString("F5") + "'," + "v37='" + b13.ToString("F5")+
"'," + "v45='" + b21.ToString("F5") +"'," + "v30='" + b6.ToString("F5") +
"'," + "v38='" + b14.ToString("F5")+ "'," + "v46='" + b22.ToString("F5")+
"',"+ "v31='" + b7.ToString("F5") + "'," + "v39='" + b15.ToString("F5") +
"',"+ "v47='" + b23.ToString("F5") +"'," + "v32='" + b6.ToString("F5") +
"',"+ "v40='" + b16.ToString("F5") + "'," + "v48='" + b24.ToString("F5")+
"' WHERE(img_id=" + Convert.ToInt32(stimg_id) + "AND location='" +
location + "')";
command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();
```

การค้นหารูปภาพโดยการเปรียบเทียบองค์ประกอบสี RGB

ก-10 Source Code อ่านค่าของ vectors องค์ประกอบสี RGB ของภาพตัวอย่าง

```
OleDbConnection Conn = new OleDbConnection();
Conn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Application.StartupPath + "\\GoogleImage.mdb";
Conn.Open();

sqlcmd = "SELECT * FROM [S_Images] WHERE (location='" +
this.txt_lastimage_location.Text + "')";
OleDbCommand command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();
OleDbDataReader rdr = command.ExecuteReader();
while (rdr.Read())
 {
     index1 = Convert.ToDouble(rdr.GetValue(4).ToString());
     index2 = Convert.ToDouble(rdr.GetValue(5).ToString());
     index3 = Convert.ToDouble(rdr.GetValue(6).ToString());
     index4 = Convert.ToDouble(rdr.GetValue(7).ToString());
     index5 = Convert.ToDouble(rdr.GetValue(8).ToString());
     index6 = Convert.ToDouble(rdr.GetValue(9).ToString());
     index7 = Convert.ToDouble(rdr.GetValue(10).ToString());
     index8 = Convert.ToDouble(rdr.GetValue(11).ToString());
     index9 = Convert.ToDouble(rdr.GetValue(12).ToString());
     index10 = Convert.ToDouble(rdr.GetValue(13).ToString());
     index11 = Convert.ToDouble(rdr.GetValue(14).ToString());
     index12 = Convert.ToDouble(rdr.GetValue(15).ToString());
     index13 = Convert.ToDouble(rdr.GetValue(16).ToString());
     index14 = Convert.ToDouble(rdr.GetValue(17).ToString());
     index15 = Convert.ToDouble(rdr.GetValue(18).ToString());
     index16 = Convert.ToDouble(rdr.GetValue(19).ToString());
     index17 = Convert.ToDouble(rdr.GetValue(20).ToString());
     index18 = Convert.ToDouble(rdr.GetValue(21).ToString());
     index19 = Convert.ToDouble(rdr.GetValue(22).ToString());
     index20 = Convert.ToDouble(rdr.GetValue(23).ToString());
     index21 = Convert.ToDouble(rdr.GetValue(24).ToString());
     index22 = Convert.ToDouble(rdr.GetValue(25).ToString());
     index23 = Convert.ToDouble(rdr.GetValue(26).ToString());
     index24 = Convert.ToDouble(rdr.GetValue(27).ToString());
 }
 rdr.Close();
```

ก-11 Source Code อ่านค่าของ vectors องค์ประกอบสี RGB ของภาพทั้งหมดในฐานข้อมูล

```
 #region check search by
 if (radioButton1.Checked == true)
 {
   sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag='" +
this.txt_imgTag.Text + "')" + "ORDER BY group_id"; ;

  }
```

```
if (radioButton2.Checked == true)
{
 if (word2 == "")
  {
      sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%'))" + "ORDER BY
group_id";
  }
 else
 {
   if (word1 == "white" || word1 == "red" || word1 == "orange" ||
       word1 == "black" || word1 == "blue" || word1 == "pink" ||
       word1 ==  "green" || word1 == "gray"|| word1 == "yellow")
   {
      sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word2 + "%'))" +
"ORDER BY group_id";
    }
    else
    {
sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%')
OR(img_tag LIKE'%" + word2 + "%'))" + "ORDER BY group_id";
    }
 }
}
if (radioButton3.Checked == true)
{
   sqlcmd = "SELECT * FROM [S_Images] ORDER BY group_id";
}
command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();
rdr = command.ExecuteReader();
while (rdr.Read())
{
  string stgp_id = rdr.GetValue(1).ToString();
  string text3 = rdr.GetValue(3).ToString();
  string img_location = rdr.GetValue(2).ToString();
  test1  = Convert.ToDouble(rdr.GetValue(4).ToString());
  test2  = Convert.ToDouble(rdr.GetValue(5).ToString());
  test3  = Convert.ToDouble(rdr.GetValue(6).ToString());
  test4  = Convert.ToDouble(rdr.GetValue(7).ToString());
  test5  = Convert.ToDouble(rdr.GetValue(8).ToString());
  test6  = Convert.ToDouble(rdr.GetValue(9).ToString());
  test7  = Convert.ToDouble(rdr.GetValue(10).ToString());
  test8  = Convert.ToDouble(rdr.GetValue(11).ToString());
  test9  = Convert.ToDouble(rdr.GetValue(12).ToString());
  test10 = Convert.ToDouble(rdr.GetValue(13).ToString());
  test11 = Convert.ToDouble(rdr.GetValue(14).ToString());
  test12 = Convert.ToDouble(rdr.GetValue(15).ToString());
  test13 = Convert.ToDouble(rdr.GetValue(16).ToString());
  test14 = Convert.ToDouble(rdr.GetValue(17).ToString());
  test15 = Convert.ToDouble(rdr.GetValue(18).ToString());
  test16 = Convert.ToDouble(rdr.GetValue(19).ToString());
  test17 = Convert.ToDouble(rdr.GetValue(20).ToString());
  test18 = Convert.ToDouble(rdr.GetValue(21).ToString());
  test19 = Convert.ToDouble(rdr.GetValue(22).ToString());
  test20 = Convert.ToDouble(rdr.GetValue(23).ToString());
  test21 = Convert.ToDouble(rdr.GetValue(24).ToString());
  test22 = Convert.ToDouble(rdr.GetValue(25).ToString());
  test23 = Convert.ToDouble(rdr.GetValue(26).ToString());
  test24 = Convert.ToDouble(rdr.GetValue(27).ToString());
```

ก-12 Source Code เปรียบเทียบค่า vectors องค์ประกอบสี RGB ของภาพตัวอย่างกับทั้งหมดในฐานข้อมูลแล้วเก็บค่าDistance ลงในฐานข้อมูล

```
sum_dif = (Math.Abs((index1 - test1)) + Math.Abs((index2 - test2)) +
Math.Abs((index3 - test3)) + Math.Abs((index4 - test4)) +
Math.Abs((index5 - test5)) + Math.Abs((index6 - test6)) +
Math.Abs((index7 - test7)) + Math.Abs((index8 - test8)) +
Math.Abs((index9 - test9)) + Math.Abs((index10 - test10)) +
Math.Abs((index11 - test11)) + Math.Abs((index12 - test12)) +
Math.Abs((index13 - test13)) + Math.Abs((index14 - test14)) +
Math.Abs((index15 - test15)) + Math.Abs((index16 - test16)) +
Math.Abs((index17 - test17)) + Math.Abs((index18 - test18)) +
Math.Abs((index19 - test19)) + Math.Abs((index20 - test20)) +
Math.Abs((index21 - test21)) + Math.Abs((index22 - test22)) +
Math.Abs((index23 - test23)) + Math.Abs((index24 - test24)));

sqlcmd = "UPDATE [S_Images] SET [rgb_dHist]=" + sum_dif.ToString("F5") +
" WHERE (img_tag='" + text3 + "'" + " AND " + "group_id=" +  stgp_id +
"AND location='" + img_location + "')";
command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();
```

ก-13 Source Code นำรูปภาพที่ทำการQuery มาเรียงลำดับจากค่าDistance น้อยสุดไปมากสุด

```
if (radioButton1.Checked == true)
 {
  sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag='" +
this.txt_imgTag.Text + "')" + "ORDER BY rgb_dHist";
}
if (radioButton2.Checked == true)
 {
if (word2 == "")
 {
  sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
  this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%'))" +
  "ORDER BY rgb_dHist";
  }
else
 {
   if (word1 == "white" || word1 == "red" ||  word1 == "orange" ||
   word1 == "black" || word1 == "blue" || word1 == "pink" ||
   word1 == "green" || word1 == "gray" || word1 == "yellow")
   {
    sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
    this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word2 + "%'))" +
    "ORDER BY rgb_dHist ";
   }
   else
   {
    sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
    this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%')
    OR(img_tag LIKE'%" + word2 + "%'))" + "ORDER BY rgb_dHist";
   }
 }
}
```

```
if (radioButton3.Checked == true)
 {
   sqlcmd = "SELECT * FROM [S_Images] ORDER BY rgb_dHist";
 }
 command = new OleDbCommand(sqlcmd, Conn);
 command.ExecuteNonQuery();
 rdr = command.ExecuteReader();
 while (rdr.Read())
 {
    if (radioButton1.Checked == true)
     {
       location = rdr.GetValue(2).ToString();
       img_tag = rdr.GetValue(3).ToString();

       sqlcmd = "SELECT
location,group_id,rValue,gValue,bValue,hValue,sValue,vValue FROM [Images]
WHERE (location='" + location + "')";
       command = new OleDbCommand(sqlcmd, Conn);
       command.ExecuteNonQuery();
       OleDbDataReader _rdr = command.ExecuteReader();
       while (_rdr.Read())
       {
          st_gp_id = _rdr.GetValue(1).ToString();
          gp_id = Convert.ToInt32(st_gp_id);
          pic = new PictureBox();
          pic.BorderStyle = BorderStyle.FixedSingle;
          pic.Cursor = Cursors.Hand;
          pic.SizeMode = PictureBoxSizeMode.StretchImage;
          pic.Image = Image.FromFile(location);
          pic.Size = imageSize;
          tag = img_tag + Convert.ToString(gp_id);
          pic.Tag = tag;
          pic.ContextMenuStrip = DeleteMenuStrip;
          maxgp_id = Convert.ToInt32(stmaxgp_id);

          #region Switch-Case Image

          if (n_pic >= 1 && n_pic <= 20)
          {

            switch (n_pic)
            {
              case 1:
                  pic.Location = new Point(20, 12);
                  this.label1.Visible = true;
                  break;
              case 2:
                  pic.Location = new Point(142, 12);
                  this.label2.Visible = true;
                  break;
              case 3:
                  pic.Location = new Point(265, 12);
                  this.label3.Visible = true;
                  break;
              case 4:
                  pic.Location = new Point(394, 12);
                  this.label4.Visible = true;
                  break;
```

```
case 5:
    pic.Location = new Point(520, 12);
    this.label5.Visible = true;
    break;
case 6:
    pic.Location = new Point(20, 193);
    this.label6.Visible = true;
    break;
case 7:
    pic.Location = new Point(142, 193);
    this.label7.Visible = true;
    break;
case 8:
    pic.Location = new Point(265, 193);
    this.label8.Visible = true;
    break;
case 9:
    pic.Location = new Point(394, 193);
    this.label9.Visible = true;
    break;
case 10:
    pic.Location = new Point(520, 193);
    this.label10.Visible = true;
    break;
case 11:
    pic.Location = new Point(20, 371);
    this.label11.Visible = true;
    break;
case 12:
    pic.Location = new Point(142, 371);
    this.label12.Visible = true;
    break;
case 13:
    pic.Location = new Point(265, 371);
    this.label13.Visible = true;
    break;
case 14:
    pic.Location = new Point(394, 371);
    this.label14.Visible = true;
    break;
case 15:
    pic.Location = new Point(520, 371);
    this.label15.Visible = true;
    break;
case 16:
    pic.Location = new Point(20, 539);
    this.label16.Visible = true;
    break;
case 17:
    pic.Location = new Point(142, 539);
    this.label17.Visible = true;
    break;
case 18:
    pic.Location = new Point(265, 539);
    this.label18.Visible = true;
    break;
case 19:
    pic.Location = new Point(394, 539);
    this.label19.Visible = true;
    break;
```

```
            case 20:
                pic.Location = new Point(520, 539);
                this.label20.Visible = true;
                break;
    }

      pics.Add(pic);
      AddPictureBoxHandler handler1 = new
      AddPictureBoxHandler(AddPictureBox_To_pnl1);
      object[] args1 = new object[1];
      args1[0] = pic;
      this.Invoke(handler1, args1);
      pic.Click += new EventHandler(pic_click);

}
```

การค้นหารูปภาพโดยการเปรียบเทียบองค์ประกอบสี HSV

ก-14 Source Code อ่านค่าของ vectors องค์ประกอบสี HSV ของภาพตัวอย่าง

```
OleDbConnection Conn = new OleDbConnection();
Conn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Application.StartupPath + "\\GoogleImage.mdb";
Conn.Open();

sqlcmd = "SELECT * FROM [S_Images] WHERE (location='" +
this.txt_lastimage_location.Text + "')";
OleDbCommand command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();
OleDbDataReader rdr = command.ExecuteReader();
while (rdr.Read())
 {
     index1 = Convert.ToDouble(rdr.GetValue(28).ToString());
     index2 = Convert.ToDouble(rdr.GetValue(29).ToString());
     index3 = Convert.ToDouble(rdr.GetValue(30).ToString());
     index4 = Convert.ToDouble(rdr.GetValue(31).ToString());
     index5 = Convert.ToDouble(rdr.GetValue(32).ToString());
     index6 = Convert.ToDouble(rdr.GetValue(33).ToString());
     index7 = Convert.ToDouble(rdr.GetValue(34).ToString());
     index8 = Convert.ToDouble(rdr.GetValue(35).ToString());
     index9 = Convert.ToDouble(rdr.GetValue(35).ToString());
     index10 = Convert.ToDouble(rdr.GetValue(37).ToString());
     index11 = Convert.ToDouble(rdr.GetValue(38).ToString());
     index12 = Convert.ToDouble(rdr.GetValue(39).ToString());
     index13 = Convert.ToDouble(rdr.GetValue(40).ToString());
     index14 = Convert.ToDouble(rdr.GetValue(41).ToString());
     index15 = Convert.ToDouble(rdr.GetValue(42).ToString());
     index16 = Convert.ToDouble(rdr.GetValue(43).ToString());
     index17 = Convert.ToDouble(rdr.GetValue(44).ToString());
     index18 = Convert.ToDouble(rdr.GetValue(45).ToString());
     index19 = Convert.ToDouble(rdr.GetValue(46).ToString());
     index20 = Convert.ToDouble(rdr.GetValue(47).ToString());
     index21 = Convert.ToDouble(rdr.GetValue(48).ToString());
     index22 = Convert.ToDouble(rdr.GetValue(49).ToString());
     index23 = Convert.ToDouble(rdr.GetValue(50).ToString());
     index24 = Convert.ToDouble(rdr.GetValue(51).ToString());
 }
 rdr.Close();
```

ก-15 Source Code อ่านค่าของ vectors องค์ประกอบสี RGB ของภาพทั้งหมดในฐานข้อมูล

```
 #region check search by
 if (radioButton1.Checked == true)
 {
   sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag='" +
this.txt_imgTag.Text + "')" + "ORDER BY group_id"; ;

 }
```

```
if (radioButton2.Checked == true)
{
 if (word2 == "")
  {
      sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%'))" + "ORDER BY
group_id";
  }
 else
 {
   if (word1 == "white" || word1 == "red" || word1 == "orange" ||
       word1 == "black" || word1 == "blue" || word1 == "pink" ||
       word1 ==  "green" || word1 == "gray"|| word1 == "yellow")
   {
      sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word2 + "%'))" +
"ORDER BY group_id";
    }
    else
    {
sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%')
OR(img_tag LIKE'%" + word2 + "%'))" + "ORDER BY group_id";
    }
 }
}
if (radioButton3.Checked == true)
{
   sqlcmd = "SELECT * FROM [S_Images] ORDER BY group_id";
}
command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();
rdr = command.ExecuteReader();
while (rdr.Read())
{
  string stgp_id = rdr.GetValue(1).ToString();
  string text3 = rdr.GetValue(3).ToString();
  string img_location = rdr.GetValue(2).ToString();
  test1  = Convert.ToDouble(rdr.GetValue(28).ToString());
  test2  = Convert.ToDouble(rdr.GetValue(29).ToString());
  test3  = Convert.ToDouble(rdr.GetValue(30).ToString());
  test4  = Convert.ToDouble(rdr.GetValue(31).ToString());
  test5  = Convert.ToDouble(rdr.GetValue(32).ToString());
  test6  = Convert.ToDouble(rdr.GetValue(33).ToString());
  test7  = Convert.ToDouble(rdr.GetValue(34).ToString());
  test8  = Convert.ToDouble(rdr.GetValue(35).ToString());
  test9  = Convert.ToDouble(rdr.GetValue(36).ToString());
  test10 = Convert.ToDouble(rdr.GetValue(37).ToString());
  test11 = Convert.ToDouble(rdr.GetValue(38).ToString());
  test12 = Convert.ToDouble(rdr.GetValue(39).ToString());
  test13 = Convert.ToDouble(rdr.GetValue(40).ToString());
  test14 = Convert.ToDouble(rdr.GetValue(41).ToString());
  test15 = Convert.ToDouble(rdr.GetValue(42).ToString());
  test16 = Convert.ToDouble(rdr.GetValue(43).ToString());
  test17 = Convert.ToDouble(rdr.GetValue(44).ToString());
  test18 = Convert.ToDouble(rdr.GetValue(45).ToString());
  test19 = Convert.ToDouble(rdr.GetValue(46).ToString());
  test20 = Convert.ToDouble(rdr.GetValue(47).ToString());
  test21 = Convert.ToDouble(rdr.GetValue(48).ToString());
  test22 = Convert.ToDouble(rdr.GetValue(49).ToString());
  test23 = Convert.ToDouble(rdr.GetValue(50).ToString());
  test24 = Convert.ToDouble(rdr.GetValue(51).ToString());
```

ก-16 Source Code เปรียบเทียบค่า vectors องค์ประกอบสี RGB ของภาพตัวอย่างกับทั้งหมดในฐานข้อมูลแล้วเก็บค่าDistance ลงในฐานข้อมูล

```
sum_dif = (Math.Abs((index1 - test1)) + Math.Abs((index2 - test2)) +
Math.Abs((index3 - test3)) + Math.Abs((index4 - test4)) +
Math.Abs((index5 - test5)) + Math.Abs((index6 - test6)) +
Math.Abs((index7 - test7)) + Math.Abs((index8 - test8)) +
Math.Abs((index9 - test9)) + Math.Abs((index10 - test10)) +
Math.Abs((index11 - test11)) + Math.Abs((index12 - test12)) +
Math.Abs((index13 - test13)) + Math.Abs((index14 - test14)) +
Math.Abs((index15 - test15)) + Math.Abs((index16 - test16)) +
Math.Abs((index17 - test17)) + Math.Abs((index18 - test18)) +
Math.Abs((index19 - test19)) + Math.Abs((index20 - test20)) +
Math.Abs((index21 - test21)) + Math.Abs((index22 - test22)) +
Math.Abs((index23 - test23)) + Math.Abs((index24 - test24)));

sqlcmd = "UPDATE [S_Images] SET [hsv_dHist]=" + sum_dif.ToString("F5") +
" WHERE (img_tag='" + text3 + "'" + " AND " + "group_id=" +  stgp_id +
"AND location='" + img_location + "')";
command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();
```

ก-17 Source Code นำรูปภาพที่ทำการQuery มาเรียงลำดับจากค่าDistance น้อยสุดไปมากสุด

```
if (radioButton1.Checked == true)
 {
  sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag='" +
this.txt_imgTag.Text + "')" + "ORDER BY hsv_dHist";
}
if (radioButton2.Checked == true)
 {
if (word2 == "")
 {
  sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
  this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%'))" +
  "ORDER BY hsv_dHist";
  }
else
 {
   if (word1 == "white" || word1 == "red" ||  word1 == "orange" ||
   word1 == "black" || word1 == "blue" || word1 == "pink" ||
   word1 == "green" || word1 == "gray" || word1 == "yellow")
   {
    sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
    this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word2 + "%'))" +
    "ORDER BY hsv_dHist ";
   }
   else
   {
    sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
    this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%')
    OR(img_tag LIKE'%" + word2 + "%'))" + "ORDER BY hsv_dHist";
   }
 }
}
```

```
if (radioButton3.Checked == true)
 {
   sqlcmd = "SELECT * FROM [S_Images] ORDER BY hsv_dHist";
 }
 command = new OleDbCommand(sqlcmd, Conn);
 command.ExecuteNonQuery();
 rdr = command.ExecuteReader();
 while (rdr.Read())
 {
    if (radioButton1.Checked == true)
     {
       location = rdr.GetValue(2).ToString();
       img_tag = rdr.GetValue(3).ToString();

       sqlcmd = "SELECT
location,group_id,rValue,gValue,bValue,hValue,sValue,vValue FROM [Images]
WHERE (location='" + location + "')";
       command = new OleDbCommand(sqlcmd, Conn);
       command.ExecuteNonQuery();
       OleDbDataReader _rdr = command.ExecuteReader();
       while (_rdr.Read())
       {
          st_gp_id = _rdr.GetValue(1).ToString();
          gp_id = Convert.ToInt32(st_gp_id);
          pic = new PictureBox();
          pic.BorderStyle = BorderStyle.FixedSingle;
          pic.Cursor = Cursors.Hand;
          pic.SizeMode = PictureBoxSizeMode.StretchImage;
          pic.Image = Image.FromFile(location);
          pic.Size = imageSize;
          tag = img_tag + Convert.ToString(gp_id);
          pic.Tag = tag;
          pic.ContextMenuStrip = DeleteMenuStrip;
          maxgp_id = Convert.ToInt32(stmaxgp_id);

          #region Switch-Case Image

          if (n_pic >= 1 && n_pic <= 20)
          {

            switch (n_pic)
            {
              case 1:
                  pic.Location = new Point(20, 12);
                  this.label1.Visible = true;
                  break;
              case 2:
                  pic.Location = new Point(142, 12);
                  this.label2.Visible = true;
                  break;
              case 3:
                  pic.Location = new Point(265, 12);
                  this.label3.Visible = true;
                  break;
              case 4:
                  pic.Location = new Point(394, 12);
                  this.label4.Visible = true;
                  break;
```

```
            case 5:
                pic.Location = new Point(520, 12);
                this.label5.Visible = true;
                break;
            case 6:
                pic.Location = new Point(20, 193);
                this.label6.Visible = true;
                break;
            case 7:
                pic.Location = new Point(142, 193);
                this.label7.Visible = true;
                break;
            case 8:
                pic.Location = new Point(265, 193);
                this.label8.Visible = true;
                break;
            case 9:
                pic.Location = new Point(394, 193);
                this.label9.Visible = true;
                break;
            case 10:
                pic.Location = new Point(520, 193);
                this.label10.Visible = true;
                break;
            case 11:
                pic.Location = new Point(20, 371);
                this.label11.Visible = true;
                break;
            case 12:
                pic.Location = new Point(142, 371);
                this.label12.Visible = true;
                break;
            case 13:
                pic.Location = new Point(265, 371);
                this.label13.Visible = true;
                break;
            case 14:
                pic.Location = new Point(394, 371);
                this.label14.Visible = true;
                break;
            case 15:
                pic.Location = new Point(520, 371);
                this.label15.Visible = true;
                break;
            case 16:
                pic.Location = new Point(20, 539);
                this.label16.Visible = true;
                break;
            case 17:
                pic.Location = new Point(142, 539);
                this.label17.Visible = true;
                break;
            case 18:
                pic.Location = new Point(265, 539);
                this.label18.Visible = true;
                break;
            case 19:
                pic.Location = new Point(394, 539);
                this.label19.Visible = true;
                break;
```

```
            case 20:
                pic.Location = new Point(520, 539);
                this.label20.Visible = true;
                break;
    }

      pics.Add(pic);
      AddPictureBoxHandler handler1 = new
      AddPictureBoxHandler(AddPictureBox_To_pnl1);
      object[] args1 = new object[1];
      args1[0] = pic;
      this.Invoke(handler1, args1);
      pic.Click += new EventHandler(pic_click);

}
```

ภาคผนวก ข

Source Code หาค่า Precision

การหาค่าPrecision นั้นเราจะอาศัยความสัมพันธ์ของรูปภาพในฐานข้อมูลโดยจะทำการแบ่งกลุ่มของรูปภาพตามคำอธิบายของรูปภาพหรือชื่อของกลุ่มรูปภาพที่ได้มาจากการค้นหาในส่วนของการค้นหาบนอินเตอร์เน็ตหรือในส่วนแรก

ข-1 การหาค่า Precision ของ HSV Color Model

```
private void Retrieved_CheckedPresion()
 {
    this.PreReMenuItem.Checked = false;
    string sqlcmd = string.Empty;
    string Img_Tag = string.Empty;
    double count = 0;
    int n_count = 0;
    int releventImg = 0;
    int Top_K = Decimal.ToInt32(this.TopKNum.Value);

    string word1 = string.Empty;
    string word2 = string.Empty;

    #region Separetor split word
    try
    {
       string keyword = this.txt_imgTag.Text;
       char[] separetor ={ ' ' };
       word1 = keyword.Split(separetor).GetValue(0).ToString();
       word2 = keyword.Split(separetor).GetValue(1).ToString();


     }
     catch
     {
       try
       {
         string keyword = this.txt_imgTag.Text;
         char[] separetor ={ ' ' };
         word1 = keyword.Split(separetor).GetValue(0).ToString();
         word2 = keyword.Split(separetor).GetValue(1).ToString();
        }
        catch
        {
          string keyword = this.txt_imgTag.Text;
          char[] separetor ={ ' ' };
          word1 = keyword.Split(separetor).GetValue(0).ToString();

        }
     }
     #endregion
```

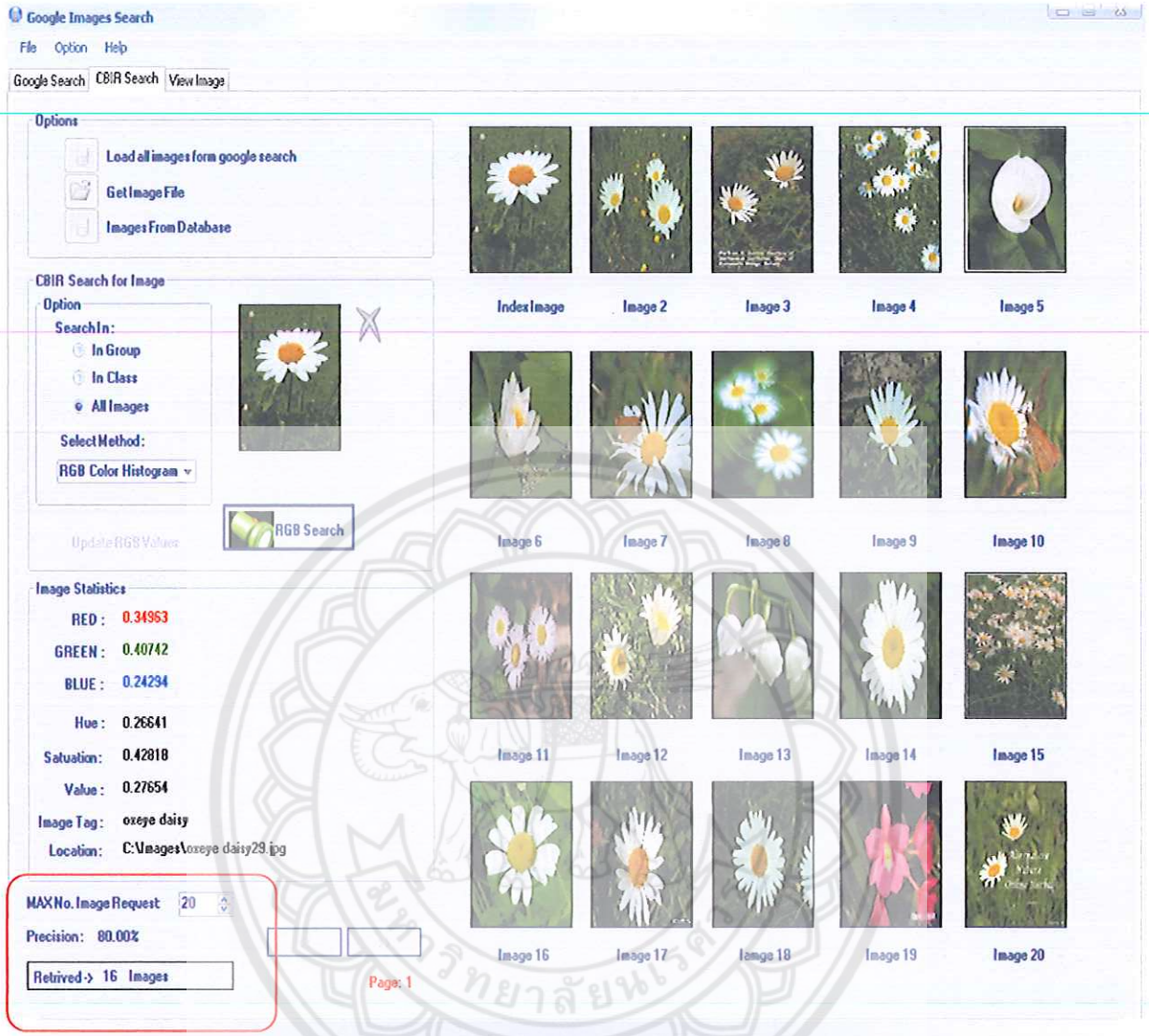
```
OleDbConnection Conn = new OleDbConnection();
Conn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Application.StartupPath + "\\GoogleImage.mdb";
Conn.Open();
sqlcmd ="SELECT * FROM [Images] WHERE(img_tag='"+ txt_imgTag.Text + "')";
OleDbCommand command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();
OleDbDataReader rdr = command.ExecuteReader();
while (rdr.Read())
{
  string img_tag = rdr.GetValue(6).ToString();
  if (img_tag != "")
  {
       releventImg += 1;
  }
}
rdr.Close();
  try
  {
   #region checked search by
   if (this.comboBox1.Text == "HSV Color Histogram")
   {
      if (radioButton1.Checked == true)
      {
        sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "')" + "ORDER BY hsv_dHist";
       }
       if (radioButton2.Checked == true)
       {
          if (word2 == "")
           {
             sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%'))" + "ORDER BY
hsv_dHist";
           }
        else
        {
          if (word1 == "white" || word1 == "red" || word1 == "orange" ||
word1 == "black" || word1 == "blue" || word1 == "pink" || word1 ==
"green" || word1 == "gray" || word1 == "yellow")
           {
             sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word2 + "%'))" + "ORDER BY
hsv_dHist ";
           }
           else
           {
             sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%')OR(img_tag
LIKE'%" + word2 + "%'))" + "ORDER BY hsv_dHist";
           }
         }
       }
        if (radioButton3.Checked == true)
        {
             sqlcmd = "SELECT * FROM [S_Images] ORDER BY hsv_dHist";
                  }
       #endregion
```

```
command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();
rdr = command.ExecuteReader();

while (rdr.Read())
{
   if (n_count < Top_K)
    {
      Img_Tag = rdr.GetValue(3).ToString();
      if (count < Top_K)
      {
         if (Img_Tag == TextToPreRe)
          {
              count += 1;
          }
       }

       n_count += 1;
     }

   }
 rdr.Close();

 //MessageBox.Show("precision : " + (((count) /
20)*100).ToString("F2")+"(+5%)or(-5%)"+"\nRecall :
"+((count)/releventImg).ToString("F3"), "Precision&Recall");
 //MessageBox.Show("This query result have a precision : " + (((count) /
Top_K) * 100).ToString("F2") + "%", "Precision");
   this.label105.Text =(((count) / Top_K) * 100).ToString("F2")+"%";
   this.lbRetrieval.Text = count.ToString();
  }
```

ข-2 การหาค่า Precision ของ RGB Color Model

```
#region checked search by
if (this.comboBox1.Text == "RGB Color Histogram")
{
   if (radioButton1.Checked == true)
    {
     sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "')" + "ORDER BY rgb_dHist";
     }

    if (radioButton2.Checked == true)
    {
       if (word2 == "")
       {
        sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%'))" + "ORDER BY
rgb_dHist";
       }
```

```
else {

      if (word1 == "white" || word1 == "red" || word1 == "orange" ||
word1 == "black" || word1 == "blue" || word1 == "pink" || word1 ==
"green" || word1 == "gray" || word1 == "yellow")
       {
        sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word2 + "%'))" + "ORDER BY
rgb_dHist ";
        }
       else
        {
          sqlcmd = "SELECT * FROM [S_Images] WHERE (img_tag ='" +
this.txt_imgTag.Text + "'OR(img_tag LIKE'%" + word1 + "%')OR(img_tag
LIKE'%" + word2 + "%'))" + "ORDER BY rgb_dHist";
        }
      }
   }
if (radioButton3.Checked == true)
 {
      sqlcmd = "SELECT * FROM [S_Images] ORDER BY rgb_dHist";
                    }
#endregion

command = new OleDbCommand(sqlcmd, Conn);
command.ExecuteNonQuery();
rdr = command.ExecuteReader();

 while (rdr.Read())
 {
     if (n_count < Top_K)
      {
        Img_Tag = rdr.GetValue(3).ToString();
        if (count < Top_K)
         {
           if (Img_Tag == TextToPreRe)
            {
               count += 1;
            }
         }

         n_count += 1;
       }

     }
     rdr.Close();
     this.label105.Text =(((count) / Top_K) * 100).ToString("F2") + "%";
     this.lbRetrieval.Text = count.ToString();
   }
     Conn.Close();
  }
   catch
   {}
```



รูปที่ ข-1. แสดงการผลการค้นหาโดย RGB Color Model และแสดงค่า Precision

จะเห็นว่าผลการค้นหาจะแสดงรูปภาพที่มีค่าองค์ประกอบสีที่ใกล้เคียงกับรูปภาพต้นแบบ และแสดงค่าความถูกต้องของการค้นหาที่ได้คือ 80 % คือภาพที่ถูกเลือกมาทั้งหมด 16 ภาพจากทั้งหมด 20ภาพ แต่ทั้งนี้ค่าความถูกต้องนี้จะสอดคล้องกับผลการค้นหาหรือไม่นั้น อาจจะขึ้นอยู่กับความพอใจ, การตัดสินใจ หรือ เหตุผลของผู้ใช้งานด้วยดังได้กล่าวมาข้างต้น