รายงานการวิจัย
ระบบการค้นหารูปภาพด้วยรายละเอียดของภาพ
Content-Based Image Retrieval

ดร.เทอดศักดิ์ ลิ่วหาทอง

ได้รับทุนสนับสนุนงานวิจัยจากเงินงบประมาณแผ่นดิน ประจำปีงบประมาณ 2554
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

กิตติกรรมประกาศ

งานวิจัยฉบับนี้สำเร็จลุล่วงไปด้วยดี เนื่องจากผู้วิจัยได้รับความช่วยเหลือ ดูแลเอาใจใส่เป็นอย่างดีจากหลายๆฝ่าย โดยเฉพาะอย่างยิ่ง รองศาสตราจารย์ ดร. มนัส สังวรศิลป์ ผู้ช่วยศาสตราจารย์ ดร. สุพันธุ์ ตั้งจิตกุศลมั่น และ รองศาสตราจารย์ ดร. สุรพันธุ์ เอื้อไพบูลย์ ในความช่วยเหลือและให้คำแนะนำเกี่ยวกับงานทางด้านการประมวลผลภาพ ผู้วิจัยรู้สึกซาบซึ้งในความกรุณาของอาจารย์ทั้งสามท่านนี้เป็นอย่างยิ่ง และขอขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอขอบคุณสำนักงานคณะกรรมการวิจัยแห่งชาติ ที่ได้สนับสนุนทุนวิจัยสำหรับงานวิจัยนี้ ขอขอบคุณนักศึกษา สาขาวิชาอิเล็กทรอนิกส์ ที่ได้ให้ความร่วมมือในการดำเนินการทดลอง

นอกจากนี้ผู้วิจัยยังได้รับกำลังใจจากคุณพ่อ คุณแม่ และเพื่อนๆ ตลอดจนบุคคลต่างๆ ที่ให้ความช่วยเหลืออีกมาก ที่ผู้วิจัยไม่สามารถกล่าวนามได้หมดในที่นี้ ผู้วิจัยรู้สึกซาบซึ้งในความกรุณาและความปรารถนาดีของทุกท่านเป็นอย่างยิ่ง จึงกราบขอบพระคุณและขอบคุณไว้ในโอกาสนี้

เทอดศักดิ์ ลิ่วหาทอง

ชื่อโครงการ (ภาษาไทย) ระบบค้นหารูปภาพด้วยรายละเอียดของภาพ

ชื่อโครงการ(ภาษาอังกฤษ) Content-Based Image Retrieval

แหล่งเงิน งบประมาณ ประจำปีงบประมาณ พ.ศ. 2554 ตามมติคณะรัฐมนตรี

ประจำปีงบประมาณ 2554 จำนวนเงินที่ได้รับการสนับสนุน 300,000 บาท

ระยะเวลาทำการวิจัย 1 ปี ตั้งแต่ 1 ต.ค. 2553 ถึง 30 ก.ย. 2554

ดร. เทอดศักดิ์ ลิ่วหาทอง

สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Email : klthurds@kmitl.ac.th

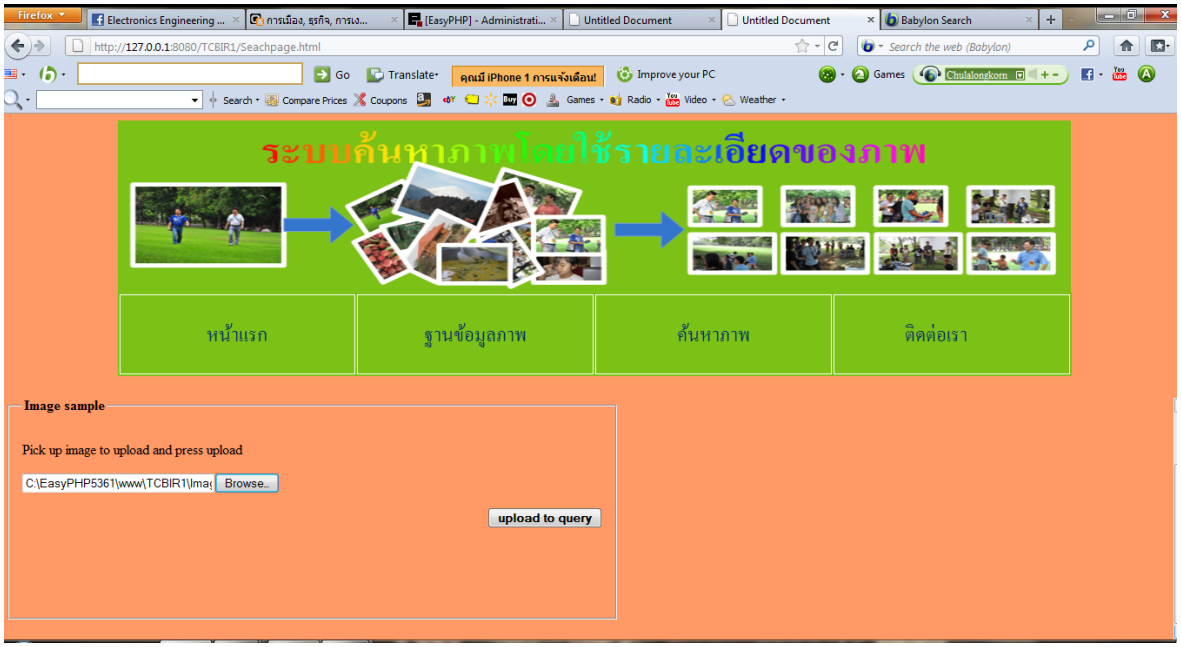
คำสำคัญ (Keywords) ระบบค้นหารูปภาพด้วยรายละเอียดของภาพ, เวคเตอร์ของลักษณะเฉพาะทางสี, Weighted Multidimensional Generalization of Wald-Wolfowitz Runs Test

บทคัดย่อ

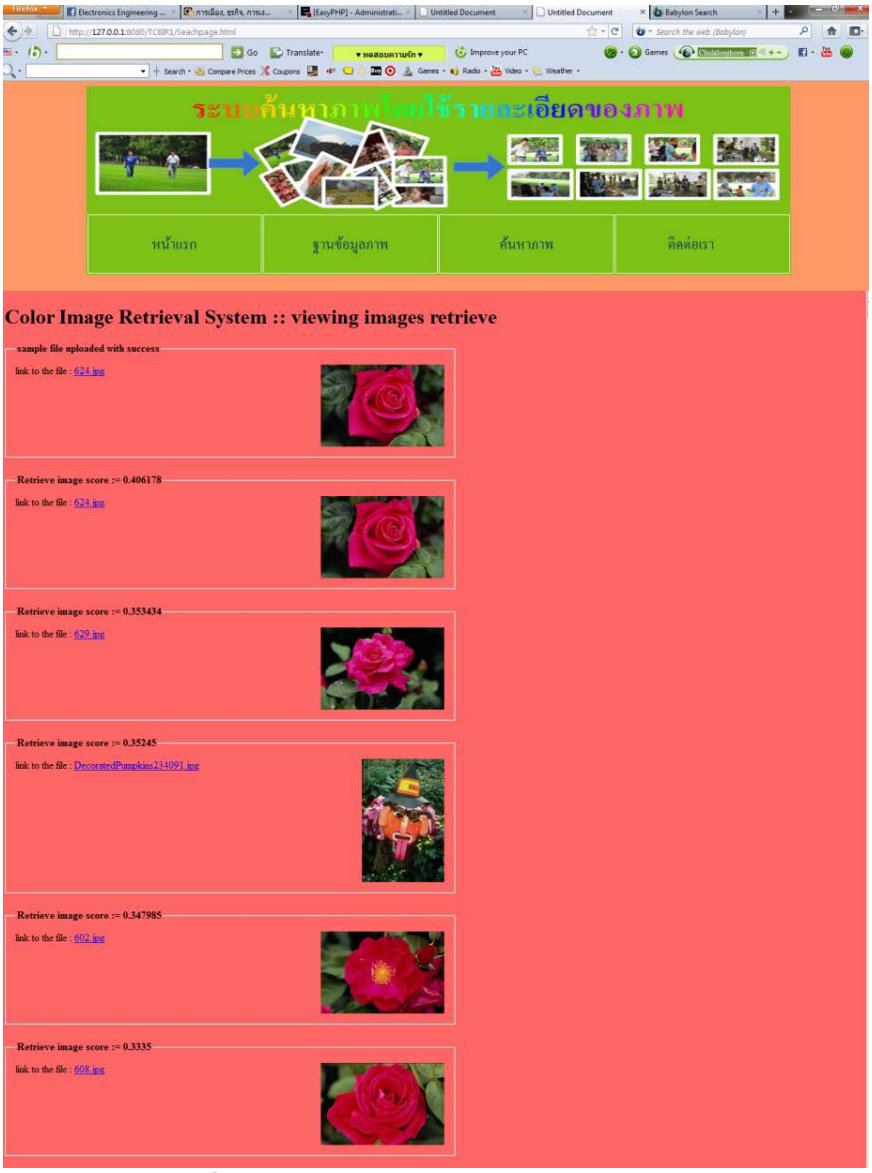
ปัจจุบันปริมาณภาพดิจิตอลที่เก็บไว้ในอินเตอร์เน็ต มีปริมาณเพิ่มขึ้นเป็นจำนวนมาก จึงมีความจำเป็นในการพัฒนาวิธีค้นหาภาพให้มีประสิทธิภาพและรวดเร็ว การค้นหาภาพในปัจจุบันใช้ข้อความในการบรรยายความหมายของภาพ และใช้ข้อความในการค้นหาภาพในฐานข้อมูล ปัญหาของการค้นหาภาพด้วยข้อความคือ ปัจจุบันยังไม่มีวิธีที่มีประสิทธิภาพในบรรยายความหมายของภาพแบบอัตโนมัติ ทำให้ต้องใช้มนุษย์ในการบรรยายแต่ละภาพ แต่เนื่องจากภาพมีจำนวนมากดังนั้นจึงต้องใช้แรงงาน, สิ้นเปลืองเงิน, และเวลาเป็นจำนวนมาก นอกจากนี้มนุษย์แต่ละคนเมื่อมองภาพเดียวกันมักจะให้ความเห็นไม่ตรงกัน ทำให้ผลลัพธ์ของการค้นหาภาพมีความไม่แน่นอน จากปัญหาดังกล่าวข้างต้น งานวิจัยนี้จึงพัฒนาระบบการค้นหาภาพด้วยรายละเอียดของภาพ (Content-Based Image Retrieval System : CBIRS) ซึ่งมีขั้นตอนการทำงานดังนี้คือ เริ่มต้นจากผู้ใช้ป้อนภาพที่ต้องการค้นหาให้กับระบบ จากนั้นระบบจะนำภาพดังกล่าวไปเปรียบเทียบกับภาพที่ถูกเก็บไว้ในฐานข้อมูล และแสดงภาพที่อยู่ในฐานข้อมูลที่ใกล้เคียงกับภาพที่ผู้ใช้ต้องการค้นหา

Abstract

Nowadays, the amount of digital images which are available on the World-Wide-Web has been massively increasing. Hence, efficient and flexible image retrieval systems for automatically browsing the entire database have been becoming real demand. The traditional image retrieval systems used text annotations to describe image semantics, and the images in the database were retrieved by their corresponding text annotations. However, since automatically generating text annotations for a wide range of images is not feasible, most text-annotation based image retrieval systems require manual annotation of images. Obviously, manually annotating images is a cumbersome and expensive task for large image databases, and the results are often subjective, context-sensitive, and incomplete. As a result, this research develops a content-based image retrieval system which can divide into three steps: 1) a user inputs a query image, 2) the system searches similar images in the database, and 3) the system browses the similar images.



รูปที่ 1. ผู้ใช้ป้อนภาพที่ต้องการค้นหาให้กับระบบ



รูปที่ 2. แสดงภาพผลลัพธ์ของการค้นหา

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1.

บทนำ

ปัจจุบันปริมาณภาพดิจิตอลมีปริมาณเพิ่มขึ้นเป็นจำนวนมากเนื่องจากการเปลี่ยนมาใช้กล้องดิจิตอล, ราคาของอุปกรณ์ข้อมูลดิจิตอลที่ถูกลง และการปริมาณภาพดิจิตอลในอินเตอร์เน็ตที่เพิ่มขึ้น ทำให้จำเป็นต้องพัฒนาระบบค้นหารูปภาพที่มีประสิทธิภาพและรวดเร็วเพื่อค้นหาภาพในฐานข้อมูล

การพัฒนาระบบค้นหาภาพสามารถย้อนหลังไปถึงช่วงหลังขอทศวรรษที่ 1970 ระบบดังกล่าวใช้ข้อความในการบรรยายความหมายของภาพและใช้ข้อความในการค้นหาภาพที่อยู่ในฐานข้อมูล [1-3] เพื่อให้ค้นหาภาพได้อย่างเที่ยงตรงจำเป็นต้องบรรยายความหมายของภาพอย่างถูกต้องและสมบูรณ์ แต่เทคโนโลยีที่มีในปัจจุบันยังไม่สามารถพัฒนาระบบอัตโนมัติสำหรับสร้างข้อความบรรยายความหมายของภาพได้ ดังนั้นจึงต้องใช้มนุษย์ในการกรอกข้อความบรรยายความหมายของแต่ละภาพ ซึ่งเป็นงานที่ต้องใช้แรงงานเป็นจำนวนมาก ใช้เวลานาน และสิ้นเปลืองเงิน และผลลัพธ์ที่ได้มักจะขึ้นกับความเห็นของผู้กรอกข้อความและไม่สมบูรณ์ ดังนั้นการค้นหาภาพโดยใช้ข้อความจึงไม่สามารถนำมาใช้งานได้อย่างกว้างขวาง

มันค่อนข้างจะชัดเจนว่าการค้นหาภาพที่มีประสิทธิภาพ ควรจะเป็นการค้นหาโดยพิจารณาจากคุณสมบัติที่อยู่ภายในภาพ นักวิจัยจากหลากหลายกลุ่มเช่น Computer Vision, Database Management, Human-Computer Interface และ Information Retrieval จึงได้หันมาพัฒนาระบบค้นหารูปภาพโดยรายละเอียดของภาพ [4-9]

1.1 ปัญหาของระบบค้นหารูปภาพโดยรายละเอียดของภาพ

ปัญหาของการค้นหารูปภาพโดยรายละเอียดของมีอยู่หลายอย่างที่จำเป็นจะต้องแก้ไขก่อนที่จะสามารถนำไปใช้งานจริงได้ ดังนี้

1.1.1 การทำความเข้าใจรายละเอียดภาพของมนุษย์

เนื่องจากมนุษย์เป็นผู้ใช้ระบบค้นหารูปภาพ ดังนั้นจึงมีความจำเป็นอย่างมากที่จะต้องศึกษาวิธีที่มนุษย์ทำความเข้าใจรายละเอียดของภาพ งานวิจัยในหัวข้อนี้ได้รับความสนใจ

เป็นอย่างมากในปัจจุบัน โดยมุ่งไปที่การศึกษาวิธีการที่มนุษย์ทำความเข้าใจรายละเอียดของภาพและวิธีการรวมแบบจำลองของวิธีการทำความเข้าใจเข้ากับระบบค้นหารูปภาพ [13-20]

1.1.2 คุณลักษณะและการวัดความเหมือนของภาพ

รายละเอียดของภาพสามารถถูกบรรยายโดยสี, รูปร่าง, พื้นผิว และการจัดวางองค์ประกอบของภาพ ในระบบการค้นหารูปภาพการบรรยายดังกล่าวสามารถสร้างจากคุณลักษณะพื้นฐานของภาพเช่น Color Histogram, Tamura Feature หรือ Fourier Descriptor คุณลักษณะของภาพที่ดีจะต้องมีคุณสมบัติดังนี้ ถ้ามีภาพสองภาพที่คล้ายกัน คุณลักษณะของภาพทั้งสองจะต้องอยู่ใกล้กันด้วย

นอกจากนี้ความเปลี่ยนแปลงต่างๆในการถ่ายภาพยังทำให้เกิดปัญหาในระบบการค้นหาภาพ ความเปลี่ยนแปลงของการถ่ายภาพที่มีปัญหามีดังต่อไปนี้

1. การหมุนวัตถุ
2. อัตราส่วนภาพ
3. การเลื่อนตำแหน่งกล้อง
4. มุมกล้อง
5. ทิศทางของแสง
6. ความเข้มของแสงและเงา
7. ความเข้มของสี
8. การบดบังวัตถุในภาพ
9. ความสับสนระหว่างวัตถุกับพื้นหลัง
10. วัตถุอื่นๆที่อยู่ในกลุ่มเดียวกัน

ดังนั้นคุณลักษณะของภาพที่ดีจะต้องไม่ขึ้นกับความเปลี่ยนแปลงต่างๆดังกล่าว แต่อย่างไรก็ตามคุณลักษณะที่ดีจะต้องรักษาสมดุลระหว่างการไม่ขึ้นกับการเปลี่ยนแปลงกับการแยกแยะความแตกต่าง เนื่องจากถ้าคุณลักษณะไม่ขึ้นกับความเปลี่ยนแปลงมากเกินไปจะทำให้ไม่สามารถแยกแยะความแตกต่างที่มีความสำคัญได้

นอกจากนี้งานที่สำคัญมากอีกอย่างของระบบการค้นหารูปภาพคือการเปรียบเทียบความเหมือนระหว่างสองภาพ ดังนั้นปัญหาคือวิธีการวัดความเหมือนของภาพสองภาพผลลัพธ์ของการวัดความเหมือนจะต้องตรงกับวิธีการวัดความเหมือนของมนุษย์

1.1.3 ความเชื่อมโยงระหว่างคุณลักษณะพื้นฐานของภาพและแนวความคิดชั้นสูง

มนุษย์มักจะใช้มักนิยมใช้แนวความคิดชั้นสูงในทุกๆวัน ตัวอย่างแนวความคิดแบบง่ายๆคือ รถ บ้าน ต้นไม้ และเครื่องบิน จากแนวความคิดแบบง่ายๆก็พัฒนาเป็นแนวความคิดที่ซับซ้อนเช่น เมือง กีฬา สัตว์ และป่า

แต่อย่างไรก็ตามเทคนิคในงาน Computer Vision ทั่วไปในปัจจุบันสามารถทำได้แค่ดึงคุณลักษณะพื้นฐานของภาพได้เท่านั้น ในงานบางอย่างเช่นระบบจดจำหน้าและการตรวจลายนิ้วมือที่สามารถเชื่อมโยงระหว่างคุณลักษณะพื้นฐานกับแนวความคิดชั้นสูงได้ แต่ไม่สามารถทำได้ในกรณีทั่วไป มันเป็นการยากมากที่จะเชื่อมโยงคุณลักษณะพื้นฐานกับแนวความคิดชั้นสูง

เพื่อลดช่องว่างในการแปลความหมาย (Semantic Gap) จำเป็นต้องใช้เทคนิคที่ รวมคุณลักษณะหลายๆอย่างไว้ด้วยกัน ตัวอย่างเช่นควรจะรวมคุณลักษณะทางสีและพื้นผิวเข้ากับวงกลม เพื่อแยกความแตกต่างระหว่างลูกบอลและจาน หรือดวงอาทิตย์ ขั้นต่อไปจำเป็นต้องใช้การประมวลผลแบบ Off-line และ On -line การประมวลผลแบบ Off-line สามารถทำได้โดยใช้เทคนิค Supervised Learning, Unsupervised Learning หรือรวมทั้งสองวิธี เครื่องมือในการเรียนรู้เหล่านี้คือ Neural network, Genetic Algorithm และ Clustering สำหรับการประมวลผลแบบ On-line จำเป็นต้องใช้การติดต่อกับผู้ใช้ที่มีความฉลาดและเป็นมิตร มันจะต้องให้ผู้ใช้สามารถประเมินผลลัพธ์ของการค้นหารูปภาพ เทคนิคดังกล่าวคือ Relevance Feedback

1.1.4 ระบบติดต่อผู้ใช้

ความแตกต่างระหว่างระบบจดจำรูปแบบและระบบค้นหารูปภาพคือผู้ใช้มีส่วนสำคัญอย่างมากในระบบที่สอง ในระบบค้นหารูปภาพ การค้นหารูปภาพเป็นปัญหาที่ไม่ชัดเจน ขึ้นกับผู้ใช้แต่ละคน คนสองคนหรือคนเดียวกันแต่ต่างเวลาอาจจะมองภาพเดียวกันในมุมมองที่แตกต่างกันได้ เราเรียกปัญหานี้ว่า ความเห็นส่วนตัวของการรับรู้ของมนุษย์ (Human Perception Subjectivity) ความเห็นส่วนตัวนี้เกิดขึ้นในหลายระดับ ตัวอย่างเช่นคน

หนึ่งอาจจะสนใจเฉพาะสีของภาพ ในขณะที่อีกคนอาจจะสนใจรูปร่างเป็นต้น ถึงแม้ว่าทั้งสองคนอาจจะสนในรูปร่างแต่การรับรู้รูปร่างของแต่ละคนก็ค่อนข้างแตกต่างกัน

1.1.5 ตัวชี้ในมิติชั้นสูง

เพื่อให้ระบบค้นหารูปภาพสามารถใช้ได้กับฐานข้อมูลขนาดใหญ่ จำเป็นต้องใช้เทคนิคตัวชี้หลายมิติ เป้าหมายของการใช้ตัวชี้คือเพื่อให้การค้นหาข้อมูลทำได้รวดเร็ว มีปัญหา 2 อย่างในระบบการค้นหารูปภาพคือ

1. ปัญหามิติชั้นสูง : โดยปกติจำนวนมิติของเวคเตอร์คุณลักษณะจะมีค่าประมาณ 10^2 [11] แต่อย่างไรก็ตามโครงสร้างของตัวชี้ที่ถูกนำเสนอจะสามารถทำงานได้อย่างมีประสิทธิภาพเมื่อจำนวนมิติมีขนาดเล็กมาก [32]
2. การวัดความเหมือนในอวกาศที่ไม่ใช่แบบยูคลิเดียน : เนื่องจากการวัดในอวกาศแบบยูคลิเดียนอาจจะไม่ใช่การวัดที่มีประสิทธิภาพมากที่สุด จึงจำเป็นต้องพัฒนาวิธีการวัดในแบบอื่นๆเช่น Histogram Intersection, Cosine, Correlation

1.1.6 การประเมินประสิทธิภาพและตัวทดสอบมาตรฐาน

การพัฒนาเทคนิคต่างๆให้ก้าวหน้าขึ้นจำเป็นจะต้องมีการประเมินประสิทธิภาพที่มีประสิทธิภาพ ตัวอย่างเช่นค่า SNR ใช้ในการประเมินการลดข้อมูล และ Precision และ Recall ใช้ในการประเมินการค้นหาข้อความ วิธีการประเมินที่ดีจะทำให้เทคนิคต่างๆถูกพัฒนาไปในทิศทางที่ถูกต้อง ระบบการค้นหารูปภาพบางระบบในปัจจุบันใช้วิธีการประเมินแบบ Cost/Time ส่วนระบบอื่นๆที่เหลือใช้วิธีการประเมินแบบ Precision/Recall ที่ยืมมาจากระบบการค้นหาข้อความ

ถึงแม้ว่าวิธีการประเมินเหล่านี้จะเป็นวิธีที่ดี แต่ก็ยังห่างไกลจากความต้องการ เหตุผลหลักอันหนึ่งที่ทำให้วิธีการประเมินเหล่านี้ด้อยประสิทธิภาพลงคือปัญหาความคิดเห็นส่วนตัวของการรับรู้รายละเอียดของภาพ นั่นคือความคิดเห็นส่วนตัวทำให้ไม่สามารถประเมินผลลัพธ์ของการค้นหาได้อย่างถูกต้อง

ปัญหาที่สำคัญอีกอย่างคือการหาตัวทดสอบมาตรฐาน ในการลดขนาดภาพเราใช้ภาพเลนา ซึ่งเป็นภาพที่มีความสมดุลย์ระหว่างพื้นผิวหลายแบบ การลดขนาดภาพวีดีโอก็มีภาพวีดีโอที่เหมาะสมสำหรับการทดสอบ ในการค้นหาข้อความก็มีตัวทดสอบมาตรฐาน

สำหรับการค้นหาภาพ เริ่มมีการรวบรวมฐานข้อมูลภาพที่ใช้เป็นตัวทดสอบมาตรฐาน ตัวทดสอบที่ดีสำหรับการค้นหารูปภาพ จะต้องเป็นฐานข้อมูลภาพที่มีขนาดใหญ่ และต้องรักษาสมดุลย์ระหว่างรายละเอียดของภาพ เพื่อทดสอบประสิทธิภาพของคุณลักษณะของภาพและประสิทธิภาพโดยรวมของระบบ

1.2 คุณลักษณะของภาพ

รายละเอียดของภาพอาจจะประกอบด้วยรายละเอียดทางสายตาและรายละเอียดทางความหมาย รายละเอียดทางสายตาประกอบด้วย สี พื้นผิว รูปร่าง การวางองค์ประกอบ และอื่นๆ รายละเอียดทางความหมายอาจได้จากการกรอกข้อความโดยมนุษย์หรือใช้กระบวนการที่ซับซ้อนซึ่งอยู่บนพื้นฐานของรายละเอียดทางสายตา งานวิจัยนี้จะมุ่งเน้นเฉพาะการบรรยายรายละเอียดทางสายตาเท่านั้น

ในระบบการค้นหารูปภาพ การบรรยายรายละเอียดทางสายตาจะต้องถูกแทนที่ด้วยคุณลักษณะภาพที่เหมาะสม ในหัวข้อนี้จะแนะนำคุณลักษณะภาพอย่างกว้างๆที่ใช้ในการแสดงสี พื้นผิว รูปร่างและองค์ประกอบของภาพ

สี

สีเป็นรายละเอียดทางสายตาที่นิยมใช้มากที่สุดในการค้นหารูปภาพ [31, 50-58] ค่าสีใน 3 มิติทำให้มันมีศักยภาพในการแยกแยะความแตกต่างได้ดีกว่าค่าระดับสีเทาใน 1 มิติของภาพ ก่อนที่จะเลือกคุณลักษณะของสีที่เหมาะสม จะต้องเลือกปริภูมิสีก่อน

ปริภูมิสี (Color Space)

แต่ละจุดในภาพสามารถแสดงเป็นจุดในระบบ 3 มิติปริภูมิสี ปริภูมิสีที่นิยมใช้ในการค้นหารูปภาพคือ RGB, Munsell, CIE L*a*b*, CIE L*u*v*, HSV, และปริภูมิสีตรงข้าม แต่อย่างไรก็ตามจนถึงตอนนี้ยังคงไม่มีปริภูมิสีใดที่ดีที่สุด แต่ปริภูมิสีที่เหมาะสมสำหรับระบบการค้นหารูปภาพจะต้องมีคุณสมบัติสม่ำเสมอในการรับรู้ (Perceptually Uniform) ปริภูมิสีที่มีความสม่ำเสมอในการรับรู้หมายความว่า ระยะทางระหว่างสี 2 สีในปริภูมิสีจะต้องมีค่าเท่ากับความเหมือนที่วัดได้โดยการรับรู้ของมนุษย์

ปริภูมิ RGB เป็นปริภูมิสี ที่ใช้กันอย่างกว้างขวางในการแสดงภาพ มันประกอบด้วยองค์ประกอบของแสงสีแดง เขียวและน้ำเงิน แต่ปริภูมิ RGB เป็นปริภูมิสีที่รู้จักกันโดยดีว่าไม่มีคุณสมบัติความสม่ำเสมอในการรับรู้

ปริภูมิ CIE L*a*b* และ CIE L*u*v* เป็นปริภูมิสีที่มีความสม่ำเสมอในการรับรู้ มันประกอบด้วยองค์ประกอบทางความสว่าง (Lightness) และองค์ประกอบทางสี (a และ b หรือ u และ v)

ปริภูมิ HSV นิยมใช้กันอย่างกว้างขวางในงานทางด้าน Computer Graphic องค์ประกอบของมันคือ Hue, Saturation และ Value ค่าของ Hue จะไม่มีการเปลี่ยนแปลงตามความสว่างของแสงและทิศทางของกล้อง ดังนั้นจึงเหมาะสมสำหรับการค้นหารูปภาพ

ปริภูมิสีตรงข้าม ประกอบด้วยองค์ประกอบ R-G, 2B-R-G, R+G+B ข้อดีของมันคือการแยกข้อมูลความสว่างไว้ในองค์ประกอบที่ 3 และองค์ประกอบที่ 1 และ 2 เป็นองค์ประกอบทางสี ซึ่งสามารถที่จะลดขนาดลงมาได้ เนื่องจากการรับรู้ของมนุษย์มีความไวต่อการเปลี่ยนแปลงของความสว่างมากกว่าการเปลี่ยนแปลงของสี

คุณลักษณะของสีที่นิยมใช้กันมีดังต่อไปนี้

Color Histogram

Color Histogram เป็นวิธีการแสดงรายละเอียดของสีของภาพได้อย่างมีประสิทธิภาพ เราสามารถคำนวณ Color Histogram ของภาพได้อย่างมีประสิทธิภาพและรวดเร็ว นอกจากนี้มันยังทนทานต่อการเปลี่ยนแปลงทางการเลื่อนภาพ การหมุน และเปลี่ยนแบบช้าๆตามสัดส่วนของภาพ การบดบังและมุมมอง

Color Histogram คือการแจกแจงของจุดภาพในแต่ละ Bin ของ Color Space Color Histogram ที่มีจำนวน Bin มากจะทำให้มีศักยภาพสูงในการแยกแยะความแตกต่าง แต่ก็ทำให้สิ้นเปลืองเวลาในการคำนวณ และไม่เหมาะสำหรับการสร้างตัวชี้ของฐานข้อมูลภาพที่มีประสิทธิภาพ

1.3 การวัดความเหมือน

ระบบการค้นหารูปภาพใช้วิธีการวัดความเหมือนของรายละเอียดทางสายตาในการเปรียบเทียบภาพที่ต้องการค้นหากับภาพที่อยู่ในฐานข้อมูล และเรียงลำดับความเหมือนของ

ภาพในฐานข้อมูลกับภาพที่ต้องการค้นหา วิธีการวัดความเหมือนของภาพหลายวิธีถูกพัฒนาขึ้นมาเพื่อใช้กับระบบการค้นหารูปภาพ วิธีการวัดที่แตกต่างกันจะทำให้ประสิทธิภาพของระบบที่แตกต่างกัน ในหัวข้อนี้จะแนะนำวิธีการวัดความเหมือนที่นิยมใช้กันบางวิธี โดยกำหนดให้ $D(I,J)$ เป็นระยะทางระหว่างภาพ I ที่ต้องการค้นหา และภาพ J ที่อยู่ในฐานข้อมูล และ $f_i(I)$ เท่ากับจำนวนจุดภาพที่อยู่ใน Bin ที่ i ของภาพ I

Minkowski-Form Distance

Minkowski-Form Distance เหมาะสมสำหรับการคำนวณระยะทางระหว่างภาพ 2 ภาพถ้าแต่ละมิติของคุณลักษณะของภาพไม่ขึ้นต่อกันและมีความสำคัญเท่ากัน เราสามารถคำนวณ Minkowski-Form Distance โดย

$$D(I,J)=\left(\sum_i \left|f_i(I)-f_i(J)\right|^p\right)^{\frac{1}{p}} \qquad (1)$$

ถ้า $p=1,2$ และ ∞ เราสามารถ $D(I,J)$ แบบย่อๆเป็น L_1, L_2 และ L_∞ Mindkowski-Form Distance เป็นวิธีการวัดความเหมือนที่นิยมใช้ กันมากในระบบการค้นหารูปภาพ

Histogram Intersection

Histogram Intersection เป็นกรณีพิเศษของ L_1 ซึ่งถูกพัฒนาโดย Swain และ Ballard [57] เพื่อใช้คำนวณความเหมือนระหว่างภาพ 2 ภาพ Histogram Intersection สามารถคำนวณโดย

$$S(I,J)=\frac{\sum_{i=1}^{N}\min\left(f_i(I),f_i(J)\right)}{\sum_{i=1}^{N} f_i(J)} \qquad (2)$$

Histogram Intersection ได้แสดงให้เห็นว่ามันไม่ไวต่อการเปลี่ยนความละเอียดของภาพ, ขนาดของ Histogram, การบดบัง ความลึกและมุมมอง

Quadratic Form (QF) Distance

Minkowski Distance กำหนดให้ทุก Bin ของ Histogram มีความสำคัญเท่ากัน โดยไม่สนใจความจริงที่ว่า Bin บางอันมีความสำคัญมากกว่าอันอื่น Quadratic Form Distance ได้ถูกพัฒนาขึ้นมาเพื่อแก้ปัญหานี้

$$D(I,J)=\sqrt{\left(F_I-F_j\right)^T A\left(F_I-F_J\right)} \qquad (3)$$

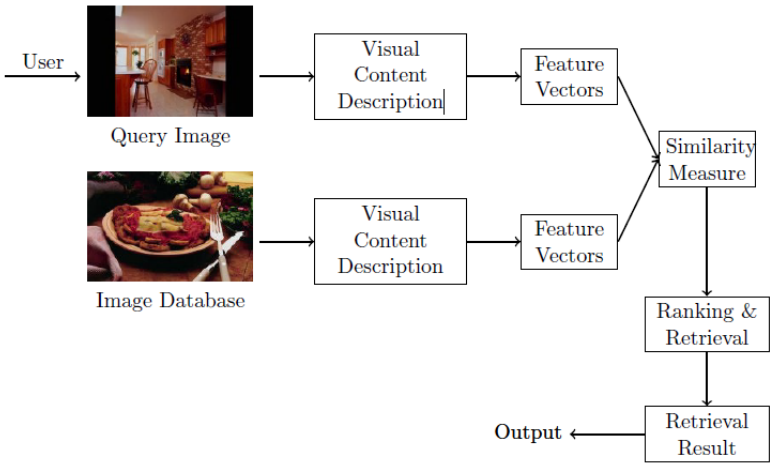
โดยที่ $A=\left[a_{ij}\right]$ เป็นเมทริกซ์ของความเหมือน และ a_{ij} หมายถึงความเหมือนระหว่าง Bin i และ Bin j

1.4 การประเมินประสิทธิภาพ

ในงานวิจัยนี้ใช้ค่า Precision ในการประเมินประสิทธิภาพของระบบการค้นหารูปภาพ สำหรับภาพที่ใช้ในการค้นหา q เซ็ทของภาพในฐานข้อมูลที่เกี่ยวข้องกับภาพ q กำหนดให้เป็น $R(q)$ และผลลัพธ์ของการค้นหาภาพ q กำหนดให้เป็น $Q(q)$ ค่า Precision คือสัดส่วนระหว่างจำนวนภาพที่เกี่ยวข้องกับภาพ q กับภาพผลลัพธ์ทั้งหมด

$$precision=\frac{Q(q)IR(q)}{\left|Q(q)\right|} \tag{4}$$

1.5 ระบบการค้นหารูปภาพโดยรายละเอียดของภาพ



รูปที่ 3. แผนผังของระบบการค้นหารูปภาพโดยรายละเอียดของภาพ

แผนผังของระบบการค้นหารูปภาพโดยรายละเอียดของภาพที่ใช้ในงานวิจัยนี้

บทที่ 2.
ทฤษฎีกราฟ, ต้นไม้แผ่ทั่วที่น้อยที่สุด และทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz

มีการใช้งานการทดสอบรันของ Wald และ Wolfowitz ในงานวิจัยทางด้านสถิติเป็นเวลานานก่อนที่จะถูกประยุกต์ในการค้นหารูปภาพโดยรายละเอียดของภาพ Wald และ Wolfowitz ได้พัฒนาทดสอบรันของพวกเขาในปี 1940 เพื่อใช้เปรียบเทียบความเหมือนระหว่างเซ็ทของจุด 2 เซ็ท [117] ผลลัพธ์ของทดสอบรันอยู่ในรูปของค่าความน่าจะเป็นที่เซ็ทของจุดทั้ง 2 เซ็ทถูกสร้างมาจากการแจกแจงแบบเดียวกัน จากนั้นในปี 1979 Friedman และ Rafsky ได้พัฒนาทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz (Multidimensional Generalization of Wald and Wolfowitz Runs Test) [114] ทฤษฎีของทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz อยู่บนพื้นฐานของการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุด (Minimal Spanning Tree) จากยูเนียนเซ็ทของจุดแบบหลายมิติ Theoharatos และคณะได้ประยุกต์ทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ในการค้นหารูปภาพโดยรายละเอียดของภาพในปี 2005 [113] พวกเขานำทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz มาใช้เปรียบเทียบความเหมือนระหว่างเซ็ทของสีที่ได้จากภาพ 2 ภาพและทำการพิสูจน์ให้เห็นว่า ทดสอบรันแบบหลายมิติมีประสิทธิภาพสูงกว่าวิธีการวัดความเหมือนแบบเดิมเช่นวิธี Histogram Intersection, Kullback-Leibler Divergtence, χ^2 Statistics และ Earth Mover's Distance [112]

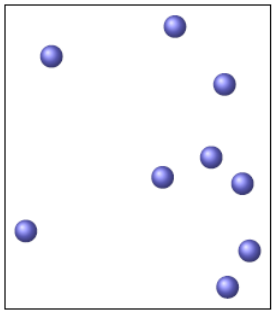
2.1 ทฤษฎีกราฟและต้นไม้แผ่ทั่วที่น้อยที่สุด

กำหนดให้ $V=\left\{v_i \middle| v_i \in \mathbf{R}^d\right\}_{i=1}^{n}$ โดยที่ $d \geq 2$ เป็นเซ็ทของจุดในปริภูมิเวคเตอร์หลายมิติ กราฟของเซ็ท V คือคู่ $G=(V,E)$ ของเซ็ทโดยที่ $E \subseteq \left\{(v_i,v_j) \middle| v_i,v_j \in V\right\}_{i,j=1}^{n}$ สมาชิกของ V และ E เรียกว่า Vertex และ Edge ตามลำดับ Vertex v_i และ v_j จะถูกเรียกว่า Adjacent หรือ

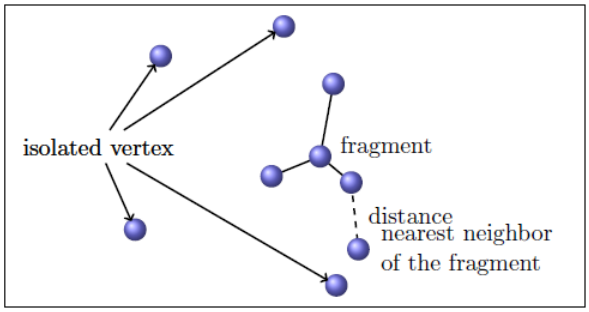
Neightbor ซึ่งกันและกันถ้า Edge (v_i, v_j) เป็นสมาชิกของ G Degree D_i ของ v_i คือจำนวน Edge ที่ต่อกับ v_i กราฟย่อย (Subgraph) ของ $G=(V,E)$ คือกราฟ $H=(V',E')$ โดยที่ $V' \subset V$ และ $E' \subset E$ ทางเดิน (Walk) ที่มีความยาวเท่ากับ k ของกราฟ G คืออนุกรมไม่ว่างของการสลับ Vertex และ Edge $v_0 e_0 v_1 e_1 \cdots e_{k-1} v_k$ ของ G โดยที่ $e_i = (v_i, v_{i+1})$ สำหรับทุกๆ $i < k$ เส้นทาง (Path) ของ G คือทางเดินที่ไม่มี Vertex ซ้ำกัน เส้นทาง $v_0 e_0 v_1 e_1 \cdots e_{k-1} v_k$ เป็นรอบ (Cycle) ถ้า $v_0 = v_k$ กราฟไม่ว่าง G เป็นกราฟเชื่อมต่อ (Connected) หมายถึงกราฟที่ Vertex ทุกคู่ของมันถูกเชื่อมด้วยเส้นทางใดเส้นทางหนึ่งของ G ต้นไม้ (Tree) คือกราฟเชื่อมต่อที่ไม่มีรอบ ต้นไม้แผ่ทั่วที่ (Spanning Tree) หมายถึงต้นไม้ที่ประกอบด้วย Vertex ทุกอันของ G และต้นไม้แผ่ทั่วที่น้อยที่สุด (Minimal Spanning Tree) หมายถึงต้นไม้แผ่ทั่วที่ที่ผลรวมของความยาวของ Edge ของมันมีค่าน้อยที่สุด

$$\sum_{e \in T} |e| = \min_{T'} \left\{ \sum_{e \in T'} |e| \right\} \tag{5}$$

โดยที่ T' คือต้นไม้แผ่ทั่วใดๆของ V และ $|e| = \|v_i - v_j\|$ เป็นความยาวของ $e = (v_i, v_j)$



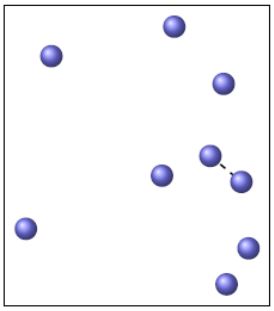
ก. Vertex ของต้นไม้แผ่ทั่วที่น้อยที่สุด



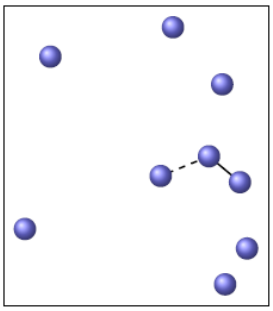
ข. Isolated Vertex, Fragment, ระยะทาง, และ Nearest Neighbor ของ Fragment

รูปที่ 4. ตัวอย่างของเครื่องมือที่ใช้ในการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุดที่นิยามโดย Prim

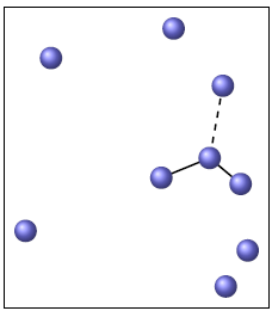
Prim ได้เสนอวิธีในการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุดจากเซ็ท V และได้นิยามเครื่องมือต่างๆที่ใช้ในการสร้างดังต่อไปนี้ Isolated Vertex หมายถึง Vertex ที่ไม่ได้เชื่อมต่อกับ Vertex อื่นๆในระหว่างกระบวนการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุด Fragment หมายถึงกราฟเชื่อมต่อที่สร้างจากซับเซ็ทของ V ระยะทาง (distance) จาก Vertex ถึง Fragment หมายถึงระยะทางที่สั้นที่สุดจาก Vertex ที่กำลังพิจารณาไปยัง Vertex ใดๆใน Fragment และ Nearest Neighbor หมายถึง Vertex ที่อยู่ใกล้ Fragment มากที่สุด ตัวอย่างของเครื่องมือในการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุดถูกแสดงไว้ในรูปที่ 4.



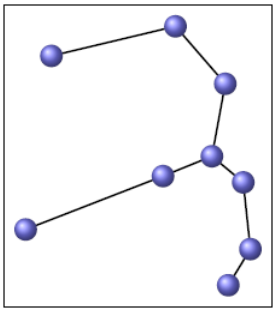
ก. Isolated Vertex เริ่มต้น และ Nearest Neighbor ของมัน



ข. Fragment เริ่มต้นและ Nearest Neighbor ของมัน



ค. Fragment ที่สองและ Nearest Neighbor ของมัน



ง. ต้นไม้แผ่ทั่วที่น้อยที่สุด

รูปที่ 5. ตัวอย่างการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุดโดยใช้วิธีของ Prim

วิธีการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุดมีดังนี้

- สุ่มเลือก Isolated Vertex เริ่มต้นและค้นหา Nearest Neighbor ของมันเพื่อสร้าง Fragment เริ่มต้น
- สำหรับ $i = 1, \cdots, N-2$

  เชื่อมต่อ Fragement ที่ i กับ Nearest Neighbor ของมัน

ตัวอย่างการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุดโดยใช้วิธีของ Prim ถูกแสดงไว้ในรูปที่ 5.

2.2 ทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz

2.2.1 นิยามของทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz

กำหนดให้ $X_1, X_2, \cdots, X_m$ เป็นจุดที่เป็นอิสระต่อกันและกันและมีการแจกแจงที่เหมือนกันในปริภูมิ $\mathbf{R}^d$ โดยมีฟังก์ชันการแจกแจงเป็น f และ $Y_1, Y_2, \cdots, Y_n$ เป็นจุดที่เป็นอิสระต่อกันและกันและมีการแจกแจงที่เหมือนกันในปริภูมิ $\mathbf{R}^d$ โดยมีฟังก์ชันการแจกแจงเป็น g Friedman และ Rafsky ได้พัฒนาทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz เพื่อใช้ทดสอบสมมติฐานหลัก $H_0 : f = g$ ซึ่งมีทฤษฎีดังนี้

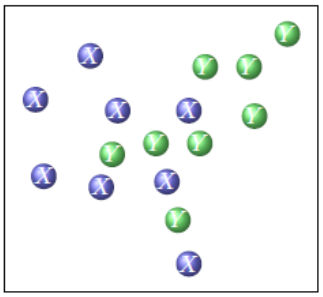
กำหนดให้ T เป็นต้นไม้แผ่ทั่วที่น้อยที่สุดของเซ็ทยูเนียนของเซ็ท $\mathbf{X} = \{X_i\}_{i=1}^m$ และ $\mathbf{Y} = \{Y_i\}_{i=1}^n$ เราเรียก Edge ที่เชื่อมต่อ Vertex ที่ได้มาจากเซ็ทที่ต่างกันว่า Inter-Set Edge Friedman และ Rafsky นิยามค่าผลลัพธ์ R ของทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz คือจำนวนต้นไม้ที่แยกจากกันที่ได้จากการลบ Inter-Set Edge ออกทั้งหมด

เนื่องจาก T เป็นต้นไม้ที่ไม่มีรอบ ดังนั้นการลบ Edge ออกจากต้นไม้ 1 ต้นจะทำให้ต้นไม้ถูกแยกจากกันเป็น 2 ต้น (ดูทฤษฎีใน [114]) ดังนั้นจำนวนต้นไม้ที่แยกจากกันจะเท่ากับจำนวน Inter-Set Edge + 1 เราสามารถคำนวณค่า R ได้จาก

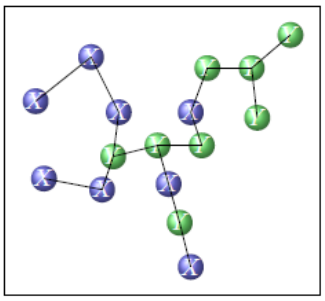
$$R = \sum_{i=1}^{m+n-1} z_i + 1$$

โดยที่ $z_i = 1$ ถ้า Edge ที่ i เป็น Inter-Set Edge และ $z_i = 0$ ถ้าไม่ใช่

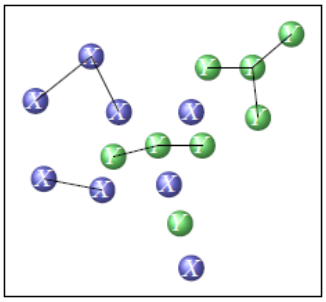
ตัวอย่างของทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ถูกแสดงไว้ในรูปที่ 6. รูปที่ 6. ก. แสดง Vertex ของยูเนียนเซ็ท $\mathbf{X} \cup \mathbf{Y}$ โดยที่ ● หมายถึง Vertex ที่มาจากเซ็ท $\mathbf{X}$ และ ● หมายถึง Vertex ที่มาจากเซ็ท $\mathbf{Y}$ รูปที่ 6. ข. แสดงต้นไม้แผ่ทั่วที่น้อยที่สุดของ $\mathbf{X} \cup \mathbf{Y}$ Inter-Set Edge คือ Edge ที่เชื่อมระหว่าง ● และ ● ซึ่งมีทั้งหมด 7 Edge และรูปที่ 6. ค. แสดงต้นไม้แยกจากกันที่ได้จากการลบ Inter-Set Edge ทั้งหมด ซึ่งจะมีจำนวนต้นไม้แยกจากกันทั้งหมด 8 ต้น ดังนั้นผลลัพธ์ของทดสอบรันของ Wald และ Wolfowitz คือ $R = 8$



ก. Vertex ของยูเนียนเซ็ท $\mathbf{X} \cup \mathbf{Y}$



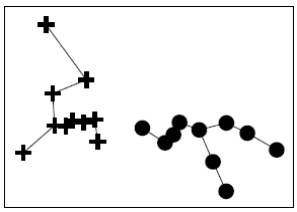
ข. กราฟต้นไม้แผ่ทั่วที่น้อยที่สุด



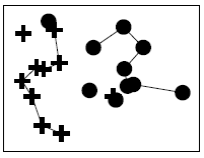
ค. ต้นไม้ที่แยกจากกันจำนวน 9 ต้นที่ได้จากการลบ Inter-Set Edge

รูปที่ 6. ตัวอย่างของทดสอบรันแบบ 2 มิติของ Wald และ Wolfowitz

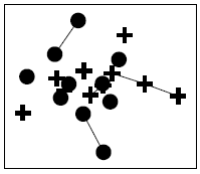
จากวิธีการของ Prim เราสามารถพิจารณาได้ว่า Inter-Set Edge คือ Edge ของการเชื่อมต่อจุดจากเซ็ท **X** และเซ็ท **Y** ที่ใกล้กันที่สุด ดังนั้นถ้า R มีค่าสูงก็หมายความว่ามีจำนวนจุดในเซ็ท **X** และเซ็ท **Y** ที่ใกล้กันอยู่มาก ดังนั้นเราสามารถพิจารณาได้ว่าฟังก์ชันการแจกแจง f เหมือนกับฟังก์ชันการแจกแจง g ในทางตรงข้ามถ้า R มีค่าต่ำก็หมายความว่ามีจำนวนจุดในเซ็ท **X** และเซ็ท **Y** ที่ใกล้กันอยู่น้อย ดังนั้นเราสามารถพิจารณาได้ว่าฟังก์ชันการแจกแจง f ไม่เหมือนกับฟังก์ชันการแจกแจง g รูปที่ 7. แสดงตัวอย่างของทดสอบรันของ Wald และ Wolfowitz ที่มีเงื่อนไขต่างกัน 3 เงื่อนไข กำหนดให้ $N(\mu,\sigma)$ เป็นฟังก์ชันการแจกแจงปกติ โดยที่ μ และ σ เป็นค่าเฉลี่ยและค่าความแปรปรวนของการแจกแจง รูปที่ 7. ก. กำหนดให้ $f=N\big((0.0,0.0),1.0\big)$ และ $g=N\big((2.5,0.0),1.0\big)$ ฟังก์ชัน f แตกต่างจากฟังก์ชัน g มาก ค่า $R=2$ รูปที่ 7. ข. กำหนดให้ $f=N\big((0.0,0.0),1.0\big)$ และ $g=N\big((1.4,0.0),1.0\big)$ ฟังก์ชัน f ค่อนข้างเหมือนฟังก์ชัน g มาก ค่า $R=7$ และรูปที่ 7. ค. กำหนดให้ $f=N\big((0.0,0.0),1.0\big)$ และ $g=N\big((0.0,0.0),1.0\big)$ ฟังก์ชัน $f=g$ ค่า $R=15$ จากการทดลองนี้แสดงให้เห็นว่ายิ่งฟังก์ชันการแจกแจง f เหมือนกับ g เท่าไหร่ยิ่งทำให้ R มีค่าสูงขึ้น



ก. ทดสอบรันของ $f=N\big((0.0,0.0),1.0\big)$ และ $g=N\big((2.5,0.0),1.0\big)$



ข. ทดสอบรันของ $f = N\big((0.0,0.0),1.0\big)$ และ $g = N\big((1.4,0.0),1.0\big)$



ค. ทดสอบรันของ $f = N\big((0.0,0.0),1.0\big)$ และ $g = N\big((0.0,0.0),1.0\big)$

รูปที่ 7. ตัวอย่างการทดสอบรันของ Wald และ Wolfowitz ที่เงื่อนไขแตกต่างกัน

Friedman และ Rafsky คาดเดาว่าสมมติฐานหลัก H_0 จะถูกต้องยิ่งขึ้นถ้า R มีค่าสูงขึ้น

2.2.2 การแจกแจงการเรียงสับเปลี่ยนของทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz

ฟังก์ชันการแจกแจงการเรียงสับเปลี่ยนของทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz เป็นพฤติกรรมที่มีประโยชน์มากและนำมาใช้ในการพัฒนาวิธีการวัดความเหมือนที่ใช้ในงานวิจัยฉบับนี้

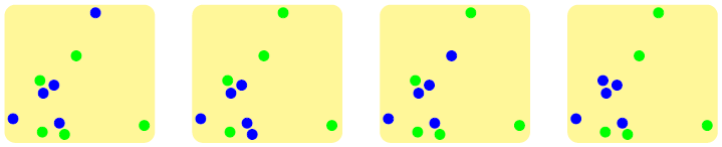
ในการคำนวณค่า R จำเป็นต้องใช้ข้อมูลตำแหน่งของจุดในเซ็ท **X** และ **Y** ทั้งหมด เพื่อสร้างต้นไม้แผ่ทั่วที่น้อยที่สุด ถ้าเราทราบแต่เพียงจำนวนของจุดในเซ็ท **X** และ **Y** เท่ากับ m และ n ตามลำดับ แต่ไม่ทราบว่าแต่ละจุดอยู่ที่ตำแหน่งใดบ้าง ย่อมเป็นไปไม่ได้ที่จะสรางต้นไม้แผ่ทั่วที่น้อยที่สุด ดังนั้นจึงไม่สามารถคำนวณหาค่า R ได้ การศึกษาฟังก์ชันการแจกแจงการเรียงสับเปลี่ยนของทดสอบรันเป็นการศึกษาค่าความน่าจะเป็นที่ค่า $R = k$ หรือ $\Pr[R = k]$ การทดลองเพื่อหาค่า $\Pr[R = k]$ ประกอบด้วย 3 ขั้นตอนดังนี้

1. กำหนดตำแหน่งเริ่มต้นทั้งหมดจำนวน $m+n$ ตำแหน่งแบบสุ่ม
2. จากตำแหน่งที่กำหนดในข้อที่ 1. ทำการเรียงสับเปลี่ยนทั้งหมด m ตำแหน่งเพื่อเป็นจุดในเซ็ท **X** และเรียงสับเปลี่ยน n ตำแหน่งเพื่อเป็นจุดในเซ็ท **Y**
3. คำนวณหาค่า R ของสำหรับการเรียงสับเปลี่ยนแต่ละวิธี
4. คำนวณ $\Pr[R = k]$

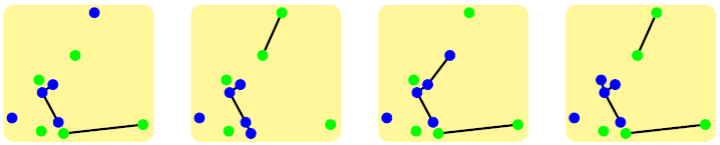
ตัวอย่างของการทดลองหาค่า $\Pr[R=k]$ ถูกแสดงไว้ในรูปที่ 8. กำหนดให้ $m,n=5$ รูปที่ 8.ก. แสดงการสุ่มตำแหน่งทั้งหมด 10 ตำแหน่ง รูปที่ 8.ข. แสดงการเรียงสับเปลี่ยนตำแหน่งของจุดในเซ็ท **X** และ **Y** เนื่องจากกำหนดให้ $m,n=5$ ดังนั้นจึงมีวิธีในการเรียงสับเปลี่ยนทั้งหมด $\frac{10!}{5!5!}=252$ วิธี รูปที่ 8.ค. แสดงวิธีการคำนวณหาค่า R ของการเรียงสับเปลี่ยนแต่ละวิธี และรูปที่ 8.ง. แสดงกราฟ $\Pr[R=k]$ ที่ได้จากการทดลอง

จากกราฟ $\Pr[R=k]$ ที่แสดงไว้ในรูปที่ 8.ง. เราสามารถคำนวณหาค่าเฉลี่ยได้เท่ากับ 6 ในทางปฏิบัติเราไม่สามารถทดลองหา $\Pr[R=k]$ เมื่อ m และ n มีค่ามากๆได้ Friedman และ Rafsky ได้พิสูจน์ทางคณิตศาสตร์จนได้ข้อสรุปว่าฟังก์ชันการแจกแจงการเรียงสับเปลี่ยนของต้นไม้แผ่ทั่วที่น้อยที่สุดของค่า

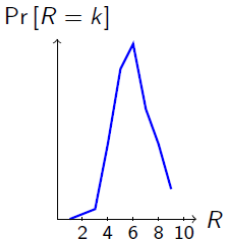
ก. การสุ่มเลือกตำแหน่งทั้งหมด 10 ตำแหน่ง



ข. การเรียงสับเปลี่ยนตำแหน่งของจุดในเซ็ท **X** และ **Y**



ค. คำนวณหาค่า R ของการเรียงสับเปลี่ยนแต่ละวิธี



ง. กราฟ $\Pr[R=k]$

รูปที่ 8. การทดลองหาค่า $\Pr[R=k]$

$$W=\frac{R-E[R]}{\sqrt{Var[R|C]}} \qquad (6)$$

ลู่เข้าสู่ฟังก์ชันการแจกแจงแบบปกติ [114] โดยที่ค่าเฉลี่ยและค่าความแปรปรวนของ R คำนวณจาก

$$E[R] = \frac{2mn}{N} + 1 \tag{7}$$

$$Var[R|C] = \frac{2mn}{N(N-1)}\left\{\frac{2mn-N}{N} + \frac{C-N+2}{(N-2)(N-3)}\left[N(N-1)-4mn+2\right]\right\} \tag{8}$$

โดยที่ $N = m+n$ และ C เท่ากับจำนวนคู่ของ Edge ทั้งหมดในต้นไม้แผ่ทั่วที่น้อยที่สุด กำหนดให้ D_i เป็น Degree ของ Vertex ที่ i จำนวนคู่ของ Edge ทั้งหมดของ Vertex ที่ i เท่ากับ $C_i = \frac{1}{2}D_i(D_i-1)$ ดังนั้นจำนวนคู่ของ Edge ทั้งหมดสามารถคำนวณได้จาก

$$C = \sum_{i=1}^{N} C_i = \frac{1}{2}\sum_{i=1}^{N} D_i(D_i-1) \tag{9}$$

W สามารถใช้ในการวัดความเหมือนในลักษณะที่ยิ่งค่า W เท่าไหร่ ความเหมือนยิ่งมากขึ้นเท่านั้น

บทที่ 3.

การวัดความเหมือนโดยใช้ทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz

ทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz เป็นวิธีที่มีประสิทธิภาพในการวัดความเหมือนของฟังก์ชันแจกแจงหลายมิติ ในงานวิจัยนี้ได้นำทดสอบรันมาใช้ในการวัดความเหมือนของฟังก์ชันการกระจายของสีที่ถูกใช้ในภาพสี 2 ภาพ จากทฤษฎีที่ได้อธิบายในหัวข้อที่ 2.2.1 เราสามารถวัดความเหมือนของฟังก์แจกแจงของสีโดยการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุดของจุดสีที่อยู่ในภาพทั้ง 2 ในปริภูมิสี 3 มิติ (ตัวอย่างเช่นในปริภูมิ RGB) และนับจำนวน Edge ที่เชื่อมต่อระหว่างสี 2 สีที่ได้มาจากภาพทั้ง 2 บวกด้วย 1 จากวิธีของ Prim จะได้ว่าค่า R ที่คำนวณได้จะเท่ากับจำนวนสี 2 สีที่อยู่ในภาพ 2 ภาพที่เหมือนกัน ถ้า R มีค่าสูงหมายความว่าภาพทั้ง 2 มีสีที่คล้ายกันมาก แสดงว่าภาพทั้ง 2 มีสีที่คล้ายกันมาก ดังนั้นภาพทั้ง 2 น่าจะเหมือนกัน ถ้า R มีค่าน้อยหมายความว่าภาพทั้ง 2 ไม่ค่อยมีสีที่คล้ายกัน แสดงว่าภาพทั้ง 2 ไม่ค่อยมีสีที่คล้ายกัน ดังนั้นภาพทั้ง 2 ไม่น่าจะเหมือนกัน แต่อย่างไรก็ตามในการนำทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz มาใช้วัดความเหมือนของภาพมีปัญหาที่ต้องได้รับการแก้ไขทั้งหมด 3 ปัญหาดังต่อไปนี้

1. **ปัญหาความสม่ำเสมอในการรับรู้ทางสี**

   เพื่อสร้างระบบค้นหารูปภาพโดยรายละเอียดของภาพให้มีประสิทธิภาพ ค่า R ที่คำนวณได้จะต้องเท่ากับจำนวนคู่สีจากภาพทั้ง 2 ที่เหมือนกันที่สังเกตุโดยมนุษย์ ดังนั้นการเลือกปริภูมิสีจึงเป็นเรื่องสำคัญมากในการคำนวณค่าทดสอบรัน ปริภูมิสีที่

ถูกเลือกจะต้องมีคุณสมบัติสม่ำเสมอในการรับรู้ในลักษณะที่ค่าความแตกต่างของสีที่คำนวณได้จากปริภูมิสีจะต้องเท่ากับค่าความแตกต่างของสีที่มนุษย์สังเกตุได้ ในงานวิจัยนี้ได้เลือกปริภูมิสี CIE L*a*b* เนื่องจากเป็นที่ทราบกันดีว่าปริภูมิสี CIE L*a*b* เป็นปริภูมิสีที่มีความสม่ำเสมอในการรับรู้

2. ปัญหาขนาดข้อมูล

การสร้างต้นไม้แผ่ทั่วที่น้อยที่สุดสำหรับสีทั้งหมดที่ได้จากภาพทั้ง 2 ใช้เวลานานมาก ซึ่งไม่เหมาะสมในทางปฏิบัติ ตัวอย่างเช่นความซับซ้อนของการสร้างต้นไม้จากภาพสีขนาด 192x128 จุดเท่ากับ $O\left((2\times192\times128)^2\right)$ ดังนั้นจึงจำเป็นต้องใช้เทคนิคในการลดขนาดข้อมูลเพื่อลดจำนวนสีที่อยู่ในภาพสีทั้ง 2 ให้มีขนาดเล็กลง นอกจากนี้เนื่องจากเราต้องการวัดความเหมือนของฟังก์ชันแจกแจงของสีของภาพทั้ง 2 ดังนั้นฟังก์ชันการแจกแจงสีที่ได้จากการลดข้อมูล จะต้องคล้ายกับฟังก์ชันการแจกแจงสีของเดิมให้มากที่สุดเท่าที่จะเป็นไปได้ ในงานวิจัยนี้ได้ใช้เทคนิคการแบ่งคลัสเตอร์แบบ k-means ในการลดข้อมูล

3. ปัญหาการรักษาความสมดุลย์ระหว่างความเร็วในการคำนวณและประสิทธิภาพของทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz

การรักษาความสมดุลย์ระหว่างความเร็วในการคำนวณ กับประสิทธิภาพของการวัดความเหมือนเป็นเรื่องที่สำคัญ ในระบบการค้นหารูปภาพโดยรายละเอียดของภาพ ระบบที่ดีจะต้องให้ผลลัพธ์ของการค้นหาที่มีประสิทธิภาพสูงในขณะที่ใช้เวลาในการคำนวณน้อย แต่เนื่องจากจำเป็นต้องลดขนาดข้อมูลให้เล็กลงก่อนที่จะคำนวณทดสอบรัน ดังนั้นประสิทธิภาพของการวัดความเหมือนจึงต่ำลง ซึ่งทำให้ไม่เหมาะสมสำหรับระบบการค้นหารูปภาพ ในงานวิจัยนี้ได้เสนอวิธีในการเพิ่มประสิทธิภาพของทดสอบรันให้เพิ่มขึ้นโดยไม่เพิ่มเวลาในการคำนวณ

งานวิจัยนี้ได้เสนอวิธี 2 วิธีในการวัดความเหมือนโดยใช้ทดสอบรัน โดยที่วิธีแรกเรียกว่าวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid วิธีนี้ถูกพัฒนาขึ้นเพื่อแก้ปัญหาความสม่ำเสมอในการรับรู้ทางสี และปัญหาขนาดข้อมูล วิธีที่สองเรียกว่าวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมี

น้ำหนักเป็นการปรับปรุงวิธีแรกเพื่อแก้ปัญหาการรักษาความสมดุลย์ระหว่างความเร็วและประสิทธิภาพ

3.1 ปริภูมิ CIE L*a*b* และการวัดความแตกต่างของสี

เราสามารถคำนวณความแตกต่างของสีในปริภูมิ RGB ระหว่างสี (R_1, G_1, B_1) และ (R_2, G_2, B_2) สามารถคำนวณได้ดังนี้

$$\Delta E_{RGB} = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2} \qquad (10)$$

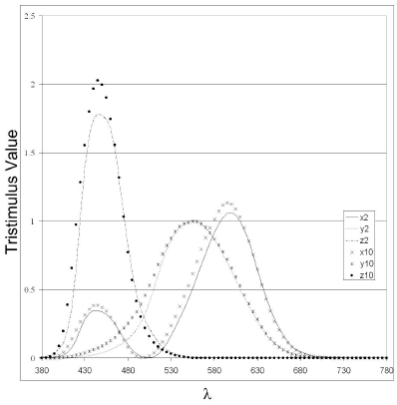
แต่อย่างไรก็ตามเป็นที่รู้กันดีว่าปริภูมิ RGB ไม่มีความสม่ำเสมอในการรับรู้ ปริภูมิสีที่มีความสม่ำเสมอในการรับรู้มากที่สุดในปัจจุบันคือ ปริภูมิ CIE L*a*b* ซึ่งทฤษฎีพื้นฐานสามารถอธิบายได้ดังต่อไปนี้

3.1.1 ค่า Tristimulus ของ XYZ-CIE

ในปี 1931 ได้มีการทดลองเพื่อศึกษาว่ามนุษย์รับรู้เรื่องสีได้อย่างไร [121] การทดลองเริ่มจากการฉายแสงสีเดี่ยวลงบนฉากและให้ผู้สังเกตุทำสังเกตุแสงที่ผสมจากแสงสีแดง เขียว และน้ำเงินที่มีสีตรงกับแสงสีเดี่ยวที่ฉายลงบนฉาก ผู้สังเกตุสามารถปรับค่าความเข้มของแสงสีแดง เขียว และน้ำเงิน จนกระทั่งได้แสงสีผสมที่ตรงกับแสงสีเดี่ยว การทดลองดังกล่าวทำโดยการกำหนด Field of View = 2^o ข้อมูลที่ได้จากการทดลองถูกปรับโดยสมการทางคณิตศาสตร์จนกระทั่งค่าที่เป็นบวกทั้งหมด และค่า $\bar{y}$ เป็นฟังก์ชันความสว่าง กราฟ $\bar{x}, \bar{y}$ และ $\bar{z}$ ที่ได้จะเรียกว่าฟังก์ชันการสังเกตุมาตรฐานของ CIE 2^o

ในปี 1964 ได้มีการทดลองลักษณะเดียวกันแต่เปลี่ยนไปใช้ File of View ขนาด 10^o ซึ่งผู้ทำการทดลองในตอนนั้นมีความเข้าใจในกายวิภาคและการทำงานต่างของดวงตามากขึ้น ทำให้ทราบว่าการใช้ File of View ขนาด 10^o น่าจะทำให้เข้าใจการรับรู้ทางสีของมนุษย์ได้ดีกว่า Field of View ขนาด 2^o

ฟังก์ชันการสังเกตุมาตรฐานของ CIE 2^o และ 10^o ถูกแสดงไว้ในรูปที่ 9.



รูปที่ 9. กราฟ CIE XYZ 2^{o} และ 10^{o}

ค่า Tristimulus ของ CIE (XYZ) สามารถคำนวณได้จาก

$$X = 100\frac{\int E(\lambda)R(\lambda)\bar{x}(\lambda)d\lambda}{\int E(\lambda)\bar{y}(\lambda)d\lambda}$$
$$Y = 100\frac{\int E(\lambda)R(\lambda)\bar{y}(\lambda)d\lambda}{\int E(\lambda)\bar{y}(\lambda)d\lambda} \qquad (11)$$
$$Z = 100\frac{\int E(\lambda)R(\lambda)\bar{z}(\lambda)d\lambda}{\int E(\lambda)\bar{y}(\lambda)d\lambda}$$

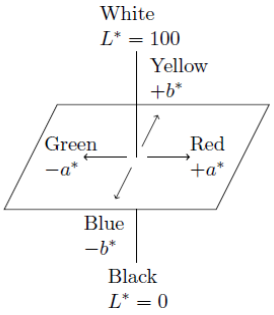
โดยที่ $\bar{x}, \bar{y}$ และ $\bar{z}$ เป็นฟังก์ชันการสังเกตุของ CIE $E(\lambda)$ คือพลังงานที่แหล่งกำเนิดแสงเปล่งออกมาที่ความยาวคลื่นเท่ากับ λ และ $R(\lambda)$ เป็นค่าการสะท้อนแสงของตัวอย่างที่ความยาวคลื่นเท่ากับ λ ค่า XYZ ถูกคำนวณโดยเทียบกับการสะท้อนแสงของแผ่น Diffuser ที่สมบูรณ์แบบ ซึ่งจะสะท้อนแสง 100 เปอร์เซนต์ที่ทุกความยาวคลื่น ผลรวมจะถูกหารโดยผลรวมของพลังงานคูณด้วยค่า $\bar{y}$ ที่ทุกความยาวคลื่นเพราะว่า Y สำหรับแสงขาวสมบูรณ์แบบเท่ากับ 100 เปอร์เซนต์ตามนิยาม

ปริภูมิ XYZ อาจจะถูกใช้ในการวัดค่าสีของวัตถุ กราฟการสะท้อนของวัตถุต่างสีจะไม่เหมือนกัน ดังนั้นค่าของ XYZ ก็แตกต่างกันเช่นกัน

แต่อย่างไรก็ตามค่า XYZ ไม่เข้ากันได้กับค่าของสี จึงยากที่จะใช้ค่า XYZ ในการอ้างอิงสีที่ต้องการ

3.1.2 **ปริภูมิ CIE L*a*b***

ในปี 1978 สมาคม CIE ได้แนะนำปริภูมิ CIE L*a*b* ซึ่งมีคุณสมบัติความสม่ำเสมอในการรับรู้ทางสี ทำให้ง่ายต่อการเปรียบเทียบสี 2 สีใดๆ ค่าระยะทางระหว่างสี 2 สีที่อยู่ในปริภูมิ CIE L*a*b* จะเท่ากับค่าความแตกต่างของสีที่มนุษย์สามารถสังเกตุได้ ปริภูมิ CIE L*a*b* อยู่ในรูปของลูกบาศก์ แกน L* จะมีค่าระหว่าง 100 และ 0 L*=100 หมายถึงมีค่าการสะท้อนแสงเท่ากับ 100 และ L*=0 หมายถึงสีดำ ส่วนแกน a* และ b* มีค่าไม่จำกัด ค่าทางบวกของแกน a* หมายถึงสีแดง ทางลบหมายถึงสีเขียว ค่าทางบวกของ b* หมายถึงสีเหลือง ทางลบหมายถึงสีน้ำเงิน ปริภูมิ CIE L*a*b* ถูกแสดงไว้ในรูปที่ 10.



รูปที่ 10. ปริภูมิ CIE L*a*b*

เราสามารถแปลงปริภูมิ RGB ให้เป็นปริภูมิ CIE L*a*b* ตามสมการต่อไปนี้

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{100}{255} \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{12}$$

$$L^* = 116 f\left(\frac{Y}{Y_0}\right) - 16$$

$$a^* = 500\left[f\left(\frac{X}{X_0}\right) - f\left(\frac{Y}{Y_0}\right)\right] \tag{13}$$

$$b^* = 200\left[f\left(\frac{Y}{Y_0}\right) - f\left(\frac{Z}{Z_0}\right)\right]$$

โดยที่ $R,G,B \in [0,255]$ X_0, Y_0 และ Z_0 เป็นค่า Tristimulus ของ CIE ของแสงสีขาวมาตรฐาน ระบบการค้นหารูปภาพโดยใช้รายละเอียดของภาพในทางปฏิบัติ เราไม่สามารถทราบค่าแสง

สีขาวมาตรฐานที่ใช้ในภาพแต่ละภาพ แต่เนื่องจากกล้องดิจิตอลทั่วไปจะใช้ค่า CIE Daylight D65 ดังนั้นในงานวิจัยนี้เราจึงกำหนดให้ $X_0 = 95.047, Y_0 = 100$ และ $Z_0 = 108.883$ [10] ฟังก์ชัน $f(q)$ คำนวณได้จาก

$$f(q) = \begin{cases} \sqrt[3]{q}, & (q > 0.008856) \\ 7.787q & (q \le 0.008856) \end{cases}$$

ค่าความแตกต่างของสีที่นิยมใช้ใน CIE L*a*b* สามารถคำนวณได้ดังนี้

$$\Delta E_{CIEL^*a^*b^*} = \sqrt{\left(L_1^* - L_2^*\right)^2 + \left(a_1^* - a_2^*\right)^2 + \left(b_1^* - b_2^*\right)^2} \tag{14}$$

นอกจากนี้ยังมีการวัดความแตกต่างของสีแบบ CIE94 ซึ่งให้ความสม่ำเสมอในการรับรู้ที่ดีกว่า ซึ่งสามารถคำนวณได้จาก [6]

$$\Delta E_{CIE94} = \sqrt{\left(\frac{\Delta L^*}{k_L S_L}\right)^2 + \left(\frac{\Delta C_{ab}^*}{k_C S_C}\right)^2 + \left(\frac{\Delta H_{ab}^*}{k_H S_H}\right)^2} \tag{15}$$

โดยที่

$$\Delta L^* = L_1^* - L_2^*$$
$$\Delta C_{ab}^* = C_1^* - C_2^*$$
$$C_S^* = \sqrt{\left(a_S^*\right)^2 + \left(b_S^*\right)^2}, (s = 1, 2)$$
$$\Delta H_{ab}^* = \sqrt{\left(\Delta a^*\right)^2 + \left(\Delta b^*\right)^2 + \left(\Delta C_{ab}^*\right)^2}$$
$$\begin{Bmatrix} \Delta a^* \\ \Delta b^* \end{Bmatrix} = \begin{Bmatrix} a_1^* - a_2^* \\ b_1^* - b_2^* \end{Bmatrix}$$
$$k_L = k_C = k_H = 1$$
$$S_L = 1, S_C = 1 + 0.045\overline{C}_{ab}^*, S_H = 1 + 0.015\overline{C}_{ab}^*$$
$$\overline{C}_{ab}^* = \sqrt{C_1^* C_2^*}$$

ในงานวิจัยนี้กำหนดให้ Ω_{RGB}, $\Omega_{CIEL*a*b*}$ และ Ω_{CIE94} เป็นปริภูมิ RGB ที่มีการวัดความแตกต่างของสีคือ ΔE_{RGB} ปริภูมิ CIE L*a*b* ที่มีการวัดความแตกต่างของสีเป็น $\Delta E_{CIEL*a*b*}$ และปริภูมิ CIE L*a*b* ที่มีการวัดความแตกต่างของสีเป็น ΔE_{CIE94} ตามลำดับ

3.2 Vector Quantization และการแบ่งคลัสเตอร์แบบ k-Means

กำหนดให้ Ω เป็นปริภูมิสีที่มีการวัดความแตกต่างของสีตามที่นิยามในหัวข้อที่ 3.1 กำหนดให้ $\mathbf{X} = \{x_i | x_i \in \Omega\}_{i=1}^{w \times h}$ เป็นเซ็ทของสีของจุดภาพในภาพที่มีความกว้างและยาวเป็น w และ h ตามลำดับ และกำหนดให้ $p(x)$ เป็นฟังก์ชันการแจกแจงของสีที่ใช้ในภาพ เราสามารถพิจารณา $p(x)$ เป็นฟังก์ชันความหนาแน่นของจุด โดยที่ $p(x)\Delta$ เท่ากับจำนวนจุดของสีที่ถูกใช้ในภาพที่อยู่ในปริมาตร Δ รอบๆตำแหน่ง x เพื่อลดเวลาในการคำนวณทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz จำเป็นต้องลดขนาดของเซ็ท $\mathbf{X}$ ที่มีจำนวนสมาชิกเท่ากับ $w \times h$ ให้เป็นเซ็ท $\overline{\mathbf{X}} = \{\overline{x}_i | \overline{x}_i \in \Omega\}_{i=1}^{N}$ โดยที่ $N << w \times h$ กำหนดให้ $\lambda(x)$ เป็นฟังก์ชันความแน่นของจุดของเซ็ท $\overline{\mathbf{X}}$ เพื่อให้ค่าทดสอบรันของเซ็ท $\overline{\mathbf{X}}$ มีค่าใกล้เคียงกับทดสอบรันของเซ็ท $\mathbf{X}$ ฟังก์ชัน $\lambda(x)$ จะต้องใกล้เคียงกับฟังก์ชัน $p(x)$ ให้มากที่สุดเท่าที่จะเป็นไปได้

ในงานวิจัยนี้ได้ใช้เทคนิคการแบ่งคลัสเตอร์แบบ k-Means เนื่องจากมันได้รับการพิสูจน์มาแล้วว่าเป็นวิธีที่มีประสิทธิภาพในการลดขนาดข้อมูลโดยที่ยังคงรักษาฟังก์ชันความหนาแน่นของจุดให้ใกล้เคียงกับฟังก์ชันเดิมไว้ได้มากที่สุด

3.2.1 Vector Quantization, Integral ของ Bennett และฟังก์ชันความหนาแน่นของจุด

กำหนดให้ $\Omega \subseteq \mathbf{R}^d$, โดยที่ $d \geq 2$ และเซ็ทของจุด $C = \{c_i | c_i \in \Omega\}_{i=1}^{N}$ เรียกว่า Codebook และเรียกจุด c_i ว่า Codeword นอกจากนี้บริเวณที่อยู่รอบๆ Codeword แต่ละจุดที่เรียกว่า Voronoi Cell S_i หมายถึงจุดทั้งหมดที่อยู่ใกล้กับ c_i มากกว่า Codeword อื่นๆ หรือสามารถเขียนเป็นสมการดังนี้

$$S_i = \left\{ z \in \Omega \middle| \|z - c_i\| \leq \|z - c_j\| \right\} \tag{16}$$

สำหรับทุกค่า $j \in [1,\cdots,N]$ โดยที่ $\|\bullet\|$ กำหนดให้เป็นการวัดระยะทางแบบยูคลิด จากสมการที่ (16) เราสามารถเรียก c_i ว่าเป็น Centroid ของ S_i และน้ำหนักของ c_i หมายถึงจำนวนสมาชิกทั้งหมดของ S_i

การทำ Vector Quantization N ระดับ ด้วย Codebook C : $Q_{NC}(z)$ หมายถึงฟังก์ชันส่ง $\Omega \to \Omega$ โดยที่ทุกๆ $z \in S_i$ ถูกแทนที่ด้วยค่า c_i ดังนั้นถ้า z มีฟังก์ชันการความน่าจะเป็นของความหนาแน่นเป็น $p(z):\Omega \to \mathbf{R}$ เราสามารถคำนวณหาค่าความบิดเบี้ยวยกกำลัง r ได้จาก

$$\begin{aligned} D\left(Q_{N,C,p}\right) &= \frac{1}{d}\int \left\| z - Q_{N,C}(z) \right\|^r p(z)\,dz \\ &= \frac{1}{d}\sum_{i=1}^{N}\int_{S_i} \left\| z - c_i \right\|^r p(z)\,dz \end{aligned} \tag{17}$$

จากงานบุกเบิกของ Bennett [123] แสดงให้เห็นว่าเราสามารถประมาณค่าความผิดพลาดเฉลี่ยกำลังสอง (r=2) ของการควอนไตซ์แบบสเกล่า (d=1) โดยที่ Voronoi Cell มีขนาดเล็ก (N มีค่าสูง) และมี c_i อยู่ที่กึ่งกลาง ได้จาก

$$D\left(Q_{N,C,p}\right) \cong \frac{1}{12N^2}\int \frac{1}{\lambda(z)^2}\,p(z)\,dz \tag{18}$$

โดยที่ $\lambda(z)$ เป็นฟังก์ชันความหนาแน่นจุดของ Centroid ด้านขวามือของสมการถูกเรียกว่า Integral ของ Bennett นอกจากนี้เขายังได้แสดงให้เห็นว่าในกรณีที่เป็นการทำ Vector Quantization ที่ดีที่สุด เราจะได้ $\lambda(z) \cong p(z)^{1/3} / \int p(z)^{1/3}\,dz$

จากนั้น Na และ Neuhoff ได้ศึกษา Integral ของ Bennett แบบหลายมิติ [124] และได้ผลลัพธ์ดังนี้

สมมติให้ Vector Quantization $Q_{N,C}$ เป็นแบบ d มิติ ที่มี Centroid จำนวนมาก (N มีค่าสูง) สมมติให้ขนาดของ Voronoi Cell มีขนาดเล็กและสมมติให้ฟังก์ชัน $p(z)$ เรียบ ดังนั้นเราสามารถประมาณสมการที่ (17) ดังนี้

$$D\left(Q_{N,C,p}\right) \cong \frac{1}{d}\sum_{i=1}^{N} p(z)\int_{S_i}\left\|z-c_i\right\|^r dz$$
$$= \sum_{i=1}^{N} p(c_i)M(S_i)v(S_i)^{1+r/d} \qquad (19)$$

โดยที่ $v(S_i)$ เป็นปริมาตรของ S_i และ $M(S_i)$ เป็นโมเมนต์ความเฉื่อยมาตรฐาน (Normalized Moment of Inertia) ของ S_i ที่จุด c_i สำหรับความบิดเบี้ยวยกกำลัง r ซึ่งสามารถคำนวณจาก

$$M(S_i) = \frac{\int_{S_i}\left\|z-c_i\right\|^r dz}{d \times v(S_i)^{1+r/d}} \qquad (20)$$

การทำให้เป็นมาตรฐานในสมการที่ (20) ทำให้ $M(S_i)$ ไม่ขึ้นกับมาตราส่วนของ S_i และสำหรับ r=2 เราจะได้ค่า $M(cube)=1/12$ สำหรับทุกๆมิติ d

สมมติว่ามีฟังก์ชัน $\lambda(z)$ (โดยปกติต้องเรียบ) ที่ทำให้ค่าทุกค่า z น้ำหนักของ Centroid ที่อยู่ในบริเวณเล็กๆ Δ ที่บรรจุ z อยู่สามารถประมาณได้เป็น $\lambda(z)v(z)$ เราเรียกฟังก์ชันนี้ว่าความหนาแน่นของจุดของ $Q_{N,C}$ ซึ่งสามารถคำนวณได้ดังนี้

$$\lambda(z) \cong \frac{1}{Nv(S_i)} \text{ สำหรับ } z \in S_i \qquad (21)$$

การมีอยู่ของฟังก์ชันความหนาแน่นของจุดทำให้คาดการณ์ได้ว่า เซลล์ที่อยู่ติดกันโดยส่วนใหญ่จะมีปริมาตรที่เท่ากัน

นอกจากนี้สมมติว่ามีฟังก์ชัน $m(z)$ (โดยปกติจะเรียบ) ที่ทำให้สำหรับทุกค่า z เซลล์ที่อยู่ใกล้กับ z มีโมเมนต์ความเฉื่อยแบบมาตรฐานใกล้เคียงกับ $m(z)$ ฟังก์ชันนี้จะเรียกว่ารูปโครงร่างของความเฉื่อย (Inertial Profile) ของ $Q_{N,C}$ การมีอยู่ของฟังก์ชันนี้ทำให้คาดการณ์ว่าเซลล์ที่อยู่ใกล้เคียงมีโมเมนต์ความเฉื่อยแบบมาตรฐานที่ใกล้เคียงกัน

เมื่อแทนฟังก์ชันความหนาแน่นของจุดและฟังก์ชันรูปโครงร่างความเฉื่อยลงในสมการที่ (19) เราจะได้

$$D\left(Q_{N,C,p}\right) \cong \frac{1}{N^{r/d}} \sum_{i=1}^{N} p\left(c_i\right) \frac{m\left(c_i\right)}{\lambda\left(c_i\right)^{r/d}} v\left(S_i\right)$$

ซึ่งสมการดังกล่าวเป็นการประมาณค่าของการอินทิเกรตแบบ Riemann ดังนั้นเราสามารถเขียนสมการใหม่ได้ดังนี้

$$D\left(Q_{N,C,p}\right) \cong \frac{1}{N^{r/d}} \int p\left(z\right) \frac{m\left(z\right)}{\lambda\left(z\right)^{r/d}} dz \qquad (22)$$

ซึ่งเป็นการ Integrall ของ Bennett แบบใหม่

สมการที่ (22) แสดงให้เห็นว่าการบิดเบี้ยวจะลดลงถ้า $\frac{1}{N^{r/d}}$ มีค่ามากขึ้นและแปรผันตรงกับค่าคงที่ที่ขึ้นกับค่าฟังก์ชันความหนาแน่น ความหนาแน่นของจุด และรูปโครงร่างความเฉื่อย สมการที่ (22) ทำให้ทราบว่าประสิทธิภาพของ Vector Quantization ขึ้นกับฟังก์ชันความหน่าแน่นจุดและรูปโครงร่างความเฉื่อย

จากการคาดเดาที่มีชื่อเสียงของ Gersho [125] ถ้า N มีค่าสูงขึ้น จะทำให้เซลล์เกือบทั้งหมดของการควอนไทซ์ที่ดีที่สุดใน d มิติ มีรูปร่างคล้ายกับ Tessellating Polytope $H_{r,d}^{*}$ ที่มีค่าโมเมนต์ความเฉื่อยแบบมาตรฐานต่ำที่สุด สมมติว่ารูปโครงร่างความเฉื่อยคงที่ตามสมการ

$$m_{r,d}^{*}\left(z\right) \square M\left(H_{r,d}^{*}\right) \square M_{r,d}^{*} \qquad (23)$$

และจากทฤษฎีอสมการของ Holder เราสามารถพิสูจน์ได้ว่าสมการที่ (22) จะมีค่าต่ำสุดเมื่อ [125]

$$\lambda_{r,d}^{*}\left(z\right) \square \frac{p\left(z\right)^{d/(d+r)}}{\int p\left(z'\right)^{d/(d+r)} dz'} \qquad (24)$$

เราสามารถพิจารณาสมการที่ (24) ว่าเป็นฟังก์ชันการแจกแจงของข้อมูลที่ถูกลดขนาดโดยใช้วิธี Vector Quantization

3.2.2 วิธีการแบ่งคลัสเตอร์แบบ k-Means

ในงานวิจัยนี้ใช้วิธีการแบ่งคลัสเตอร์แบบ k-Means ซึ่งวิธีที่รู้จักกันดีสำหรับการทำ Vector Quantization การแบ่งคลัสเตอร์แบบ k-Means จะให้ผลลัพธ์ออกมาเป็นเซ็ท $\{(c_i, w_i)\}_{i=1}^{N}$ โดยที่ c_i เป็น Centroid ของเซลล์ที่ i และ w_i เป็นจำนวนสีทั้งหมดที่ถูกใช้ในภาพที่เป็นสมาชิกของเซลล์ที่ i เพื่อแก้ปัญหาเรื่องการกำหนดตำแหน่งเริ่มต้นของ Centroid ในงานวิจัยนี้ได้ใช้วิธี Binary Splitting ที่ถูกพัฒนาโดย Linde และคณะ [115]

กำหนดให้ Ω และ $\|\bullet\|$ เป็นปริภูมิสีและการวัดความแตกต่างที่ได้อธิบายในหัวข้อที่ 3.1 ขั้นตอนการทำงานของ k-Means ที่ใช้ในงานวิจัยนี้สามารถเขียนเป็นรหัสจำลองได้ดังนี้

1. กำหนดค่าเริ่มต้น
   - กำหนดให้เซ็ทของสีเป็น $\mathbf{X} = \{x_i | x_i \in \Omega\}_{i=1}^{w \times h}$ และกำหนดค่าความผิดพลาด $\varepsilon > 0$ ที่มีขนาดเล็กมากๆ
   - กำหนดให้ $N^* = 1$ และ

     $$c_1^* = \frac{1}{w \times h} \sum_{i=1}^{w \times h} x_i$$

     คำนวณ

     $$D_{ave}^* = \frac{1}{w \times h \times d} \sum_{i=1}^{w \times h} \left\| x_i - c_1^* \right\|^2$$

2. การแยก Centroid
   - สำหรับทุกค่า $i = 1, 2, \cdots, N^*$

     ตั้งค่า

     $$c_i^{(0)} = (1 + \varepsilon) c_i^*$$
     $$c_{N^*+i}^{(0)} = (1 - \varepsilon) c_i^*$$
   - ถ้า $2N^* \le N$ ดังนั้น

     ตั้งค่า

     $$N^* = 2N^*$$

     ถ้าไม่ใช่

     ตั้งค่า

     $$N^* = N$$
3. การวนรอบ

- กำหนดให้ $D_{ave}^{(0)} = D_{ave}^{*}$ และตั้งค่าดัชนีการวนรอบ $j = 0$

(ก) สำหรับ $i = 1, \cdots, w \times h$ ให้หาค่าต่ำสุดของ

$$\left\| x_i - c_n^{(j)} \right\|^2$$

สำหรับทุกค่า $n = 1, \cdots, N*$ กำหนดให้ $n*$ เป็นค่าที่ทำให้ $\left\| x_i - c_n^{(j)} \right\|^2$ ต่ำที่สุด กำหนดให้

$$Q(x_i) = c_{n*}^{(j)}$$

(ข) สำหรับ $n = 1, \cdots, N*$ ปรับค่าของ Centroid ให้เป็นค่าปัจจุบันดังนี้

$$c_n^{j+1} = \frac{\sum_{Q(x_i)=c_n} x_i}{\sum_{Q(x_i)=c_n} 1}$$

(ค) ตั้งค่า $j = j + 1$

(ง) คำนวณ

$$D_{ave}^{(j)} = \frac{1}{w \times h \times d} \sum_{i=1}^{w \times h} \left\| x_i - Q(x_i) \right\|^2$$

(จ) ถ้า $\frac{D_{ave}^{(j-1)} - D_{ave}^{(j)}}{D_{ave}^{(j-1)}} > \varepsilon$ ดังนั้นกลับไปทำงานในขั้นตอน (ก)

(ฉ) ตั้งค่า $D_{ave}^{*} = D_{ave}^{(j)}$

สำหรับ $n = 1, \cdots, N*$

ตั้งค่า

$$c_n^{*} = c_n^{(i)}$$

4. ย้อนกลับไปทำในขั้นตอนที่ 2 และ 3 จนกระทั่ง $N* = N$

3.3 นิยามของวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid

กำหนดให้ $\{x_i\}_{i=1}^{w \times h}$ และ $\{y_i\}_{i=1}^{w \times h}$ เป็นเซ็ทของสีของจุดภาพสำหรับภาพ I_1 และ I_2 ตามลำดับ และกำหนดให้ f และ g เป็นฟังก์ชันการแจกแจงของสีของภาพ I_1 และ I_2 ตามลำดับ นอกจากนี้กำหนดให้ $A = \{(a_i, w_{ai})\}_{i=1}^{N}$ และ $B = \{(b_i, w_{bi})\}_{i=1}^{N}$ เป็นเซ็ทของ Centroid และน้ำหนักของ Centroid หรือเรียกว่าเวคเตอร์ของลักษณะเฉพาะ (Feature Vector) ที่คำนวณได้จากภาพ I_1 และ I_2 ตามลำดับ จากรหัสจำลองของ k-Means ที่ได้อธิบายในหัวข้อที่ 3.2.2 เราจะได้ $r = 2$ และมิติของปริภูมิสี $d = 3$ เมื่อแทนค่าลงในสมการที่ (24) เราจะได้ฟังก์ชันความหนาแน่นของจุดของ $\{a_i\}_{i=1}^{N}$ และ $\{b_i\}_{i=1}^{N}$ ดังนี้

$$\lambda_A^*(z) = \frac{f(z)^{0.6}}{\int f(z')^{0.6}\,dz'}$$

และ

$$\lambda_B^*(z) = \frac{g(z)^{0.6}}{\int g(z')^{0.6}\,dz'}$$

วิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid ระหว่างเซ็ท A และ B เป็นดังต่อไปนี้

กำหนดให้ $V = \{a_1, \cdots, a_N, b_1, \cdots, b_N\}$ เป็นเซ็ทของ Vertex และกำหนดให้ $e = (c_i, c_j)$ โดยที่ $c_i, c_j \in V$ เป็น Edge ของ V และความยาวของ Edge คำนวณได้จาก $|e| = \Delta E$ โดยที่ ΔE เป็นความแตกต่างของสีที่เลือกใช้ในวิธี k-Means

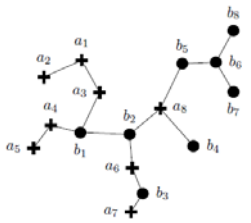
กำหนดให้ T เป็นต้นไม้แผ่ทั่วที่น้อยที่สุดของ V ที่ได้นิยามไว้ในหัวข้อที่ 2.1 ดังนั้นกทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz เท่ากับจำนวน Inter-Set Edge + 1 Degree ของการวัดความเหมือนสามารถคำนวณได้จากค่า W ในสมการที่ (6)

ค่า W สามารถใช้ในการวัดความเหมือนระหว่างภาพ I_1 และ I_2 ในลักษณะที่ว่า ยิ่งค่า W สูงเท่าไหร่ ภาพ I_1 ยิ่งเหมือนกับภาพ I_2 มากเท่านั้น

ตัวอย่างของต้นไม้แผ่ทั่วที่น้อยที่สุดของ $\{a_1, \cdots, a_8, b_1, \cdots, b_8\}$ ถูกแสดงไว้ในรูปที่ 11. จากรูปเราจะได้ $R = 9, C = 18, E[R] = 9$ และ $Var[R|C] = 3.56923$ ดังนั้นสามารถคำนวณค่า W จากสมการที่ (6) ได้ดังนี้

$$W = \frac{9-9}{\sqrt{3.56923}} = 0$$

ค่า W ที่ได้จากวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid เป็นการวัดความเหมือนระหว่างฟังก์ชันความหนาแน่นจุด $\lambda_A^*(z)$ และ $\lambda_B^*(z)$ ของภาพ I_1 และ I_2 ตามลำดับ

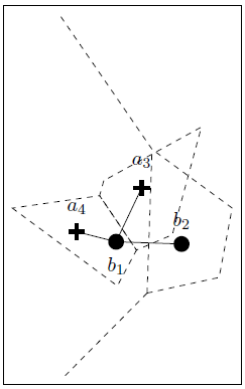


รูปที่ 11. ตัวอย่างของต้นไม้แผ่ทั่วที่น้อยที่สุดของ $\{a_1,\cdots,a_8,b_1,\cdots,b_8\}$

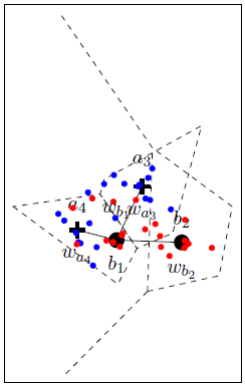
3.4 นิยามของวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมีน้ำหนัก

กำหนดให้ $\mathbf{X}=\{X_i|X_i\in\Omega\}_{i=1}^{w\times h}$ และ $\mathbf{Y}=\{Y_i|Y_i\in\Omega\}_{i=1}^{w\times h}$ เป็นเซ็ทของสีของจุดภาพในภาพ I_1 และ I_2 ตามลำดับ ในทางทฤษฎีการทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ได้จากต้นไม้แผ่ทั่วที่น้อยที่สุดของ $\mathbf{X}\cup\mathbf{Y}$ เป็นวิธีการวัดความเหมือนที่มีประสิทธิภาพมาก แต่ในทางปฏิบัติการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุดของ $\mathbf{X}\cup\mathbf{Y}$ ใช้เวลานานมาก ดังนั้นงานวิจัยนี้จึงได้เสนอวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมีน้ำหนัก ซึ่งเป็นวิธีที่ใช้ในการประมาณจำนวน Inter-Set Edge

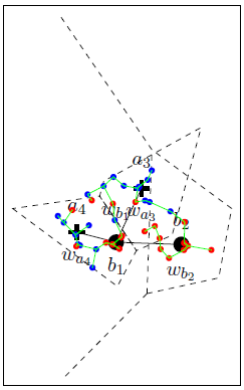
กำหนดให้ $A=\{(a_i,w_{a_i})\}_{i=1}^{N}$ และ $B=\{(b_i,w_{b_i})\}_{i=1}^{N}$ เวคเตอร์ของลักษณะเฉพาะที่ได้จากภาพ I_1 และ I_2 ตามลำดับ กำหนดให้ S_{a_i} และ S_{b_i} เป็น Voronoi Cell ตามนิยามในสมการที่ (16) ของ a_i และ b_i ตามลำดับ



ก. ตัวอย่างของ b_1, Neighbor และ Voronoi cell



ข. สีของจุดภาพที่เป็นสมาชิกของบริเวณ $S_{a_3} \cup S_{a_4} \cup S_{b_1} \cup S_{b_2}$



ค. ตัวอย่างของทดสอบรันในบริเวณ $S_{a_3} \cup S_{a_4} \cup S_{b_1} \cup S_{b_2}$

รูปที่ 12. ตัวอย่างทดสอบรันในบริเวณ $S_{a_3} \cup S_{a_4} \cup S_{b_1} \cup S_{b_2}$

เมื่อพิจารณาต้นไม้แผ่ทั่วที่น้อยที่สุดของ $A \cup B$ ที่แสดงไว้ในรูปที่ 11. จะเห็นว่า Neighbor ของ Centroid b_1 คือ Centroid a_3, a_4 และ b_2 ดังแสดงไว้ในรูปที่ 12.ก. จากทฤษฎีของ Vector Quantization ทำให้ทราบว่า มีสีของจุดภาพจากภาพ I_1 จำนวนเท่ากับ w_{a_3} และ w_{a_4} อยู่รอบๆ Centroid a_3, a_4 และมีสีของจุดภาพจากภาพ I_2 จำนวนเท่ากับ w_{b_1} และ w_{b_2} อยู่รอบๆ Centroid b_1, b_2 ดังแสดงในรูปที่ 12.ข. เราสามารถคำนวณทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ในบริเวณ $S_{a_3} \cup S_{a_4} \cup S_{b_1} \cup S_{b_2}$ ได้ดังแสดงในรูปที่ 12.ค.

ในทางปฏิบัติการสร้างต้นไม้แผ่ทั่วที่น้อยที่สุดของเซ็ทที่มีจุดจำนวนมาก จำเป็นต้องใช้เวลานานมาก แต่เราสามารถนำทฤษฎีของการแจกแจงการเรียงสับเปลี่ยนที่ได้อธิบายไว้ในหัวข้อที่ 2.2.2 มาใช้ประมาณค่าของทดสอบรันในบริเวณ $S_{a_3} \cup S_{a_4} \cup S_{b_1} \cup S_{b_2}$ ค่าทดสอบรันที่มีความเป็นไปได้มากที่สุดคือ ค่าเฉลี่ยของการแจกแจงการเรียงสับเปลี่ยนซึ่งสามารถคำนวณได้จาก สมการที่ (7) ค่าโดยประมาณของจำนวน Inter-Set Edge ในบริเวณ $S_{a_3} \cup S_{a_4} \cup S_{b_1} \cup S_{b_2}$ คำนวณได้ดังนี้

$$R(b_1) = \frac{2\left(w_{b_1} + w_{b_2}\right)\left(w_{a_3} + w_{a_5}\right)}{\left(w_{b_1} + w_{b_2}\right)\left(w_{a_3} + w_{a_5}\right)}$$

เราสามารถใช้วิธีการประมาณค่าดังกล่าวในการประมาณจำนวน Inter-Set Edge รอบๆ Centroid ที่เหลือทั้งหมด

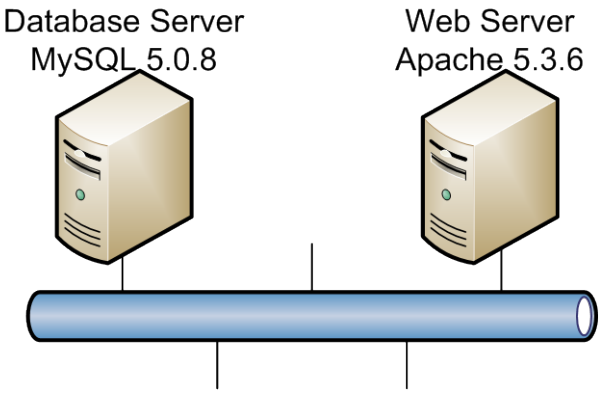
เราเรียกผลรวมของการประมาณจำนวน Inter-Set Edge ของ Cent roid ทั้งหมดว่าการทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมีน้ำหนัก ซึ่งสามารถเขียนเป็นสมการได้ดังนี้

$$R(I_1, I_2) = \sum_{i=1}^{M} R(a_i) + \sum_{i=1}^{M} R(b_i) \qquad (25)$$

บทที่ 4.

ระบบค้นหารูปภาพโดยใช้รายละเอียดของภาพ

ระบบค้นหารูปภาพโดยใช้รายละเอียดของภาพในงานวิจัยนี้ประกอบด้วย



รูปที่ 13. ระบบค้นหารูปภาพโดยรายละเอียดของภาพ

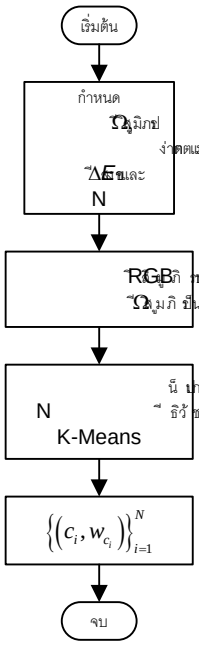
1. คอมพิวเตอร์ CPU Intel Core i5 ความเร็ว 2.53 GHz หน่วยความจำ 4 GB ระบบปฏิบัติการ Windows 7 32 bit จำนวน 2 เครื่องเพื่อใช้เป็น Web Server และ Database Server
2. เครื่องที่ใช้เป็น Database Server ได้ลงโปรแกรม MySQL 5.0.8 เพื่อใช้ในการจัดการฐานข้อมูลภาพที่ใช้ในงานวิจัย
3. เครื่องที่ใช้เป็น Web Server ได้ลงโปรแกรม PHP 5.3.6
4. ในเครื่อง Web Server ได้ติดตั้ง Web Service ที่พัฒนาโดยใช้ภาษา Visual C++ 2008 ร่วมกับ OpenCV 2.3.0 จำนวน 2 โปรแกรมดังนี้

a. VqKmeanCalculate ทำหน้าที่ในการคำนวณเวคเตอร์ของลักษณะเฉพาะ $\{(c_i, w_{c_i})\}_{i=1}^{N}$ จากภาพที่ผู้ใช้ต้องการค้นหา โดยใช้ภาษา Visual C++ 2010 ร่วมกับ OpenCV 2.3.0

b. VqRetrieval ทำหน้าที่คำนวณทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ระหว่างเวคเตอร์ของลักษณะเฉพาะของภาพที่ต้องการค้นหากับภาพที่อยู่ในฐานข้อมูล

5. ในเครื่อง Web Server ได้ติดตั้ง Web Page ที่พัฒนาโดยภาษา PHP และ JQuery เพื่อใช้เก็บภาพที่ต้องการลงในฐานข้อมูลภาพ
6. ในเครื่อง Web Server ได้ติดตั้ง Web Page ที่พัฒนาโดยภาษา PHP และ JQuery เพื่อให้ผู้ใช้บริการค้นหารูปภาพ ป้อนภาพที่ต้องการค้นหา และแสดงภาพที่ได้จากการค้นหาภาพ

4.1 การทำงานของโปรแกรม VqKmeanCalculate



รูปที่ 14. Flowchart การทำงานของโปรแกรม VqKmeanCalculate

ขั้นตอนการทำงานของโปรแกรม VqKmeanCalculate สามารถ เขียนเป็น Flowchart ดังในรูปที่ 14. ซึ่งสามารถอธิบายได้ดังนี้

1. กำหนดปริภูมิสี Ω การวัดความแตกต่างของสี ΔE และจำนวนคลัสเตอร์ N
2. แปลงปริภูมิ RGB ให้เป็นปริภูมิสี Ω ตามที่กำหนดในข้อที่ 1.
3. ทำการแบ่งคลัสเตอร์ออกเป็น N คลัสเตอร์โดยใช้วิธี k-Means โดยกำหนดให้การวัดระยะทางใน k-Means $\|\bullet\|$ เป็นวิธีการวัดความแตกต่างของสี ΔE ตามที่กำหนดในข้อที่ 1.
4. นำผลลัพธ์ของวิธี k-Means มาใช้สร้างเวคเตอร์ลักษณะเฉพาะ $\{(c_i, w_i)\}_{i=1}^{N}$ และเก็บเวคเตอร์ลักษณะเฉพาะในไฟล์แบบ YAML

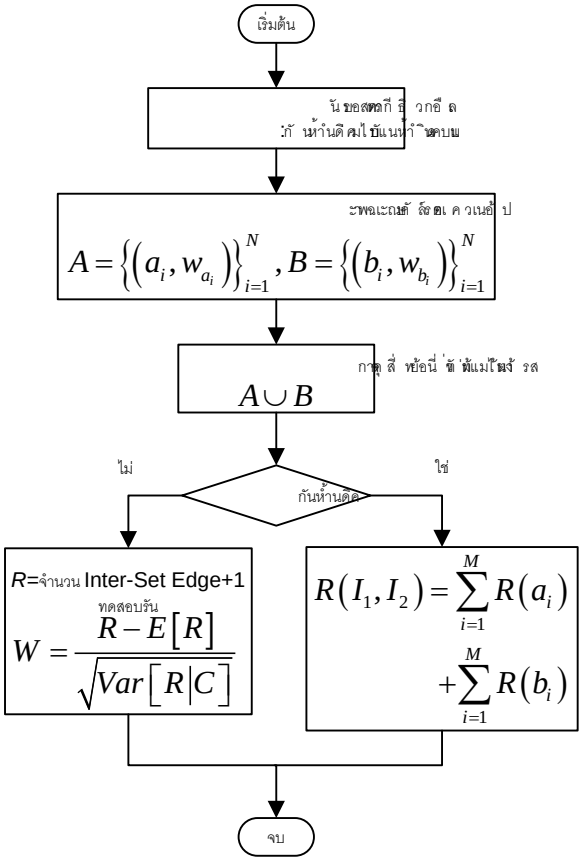
4.2 การทำงานของโปรแกรม VqRetrieval

ขั้นตอนการทำงานของโปรแกรม VqRetrieval สามารถ เขียนเป็น Flowchart ดังในรูปที่ 15. ซึ่งสามารถอธิบายได้ดังนี้

1. เลือกว่าต้องการทดสอบรันแบบมีน้ำหนักหรือไม่มีน้ำหนัก
2. อ่านไฟล์ YAML ที่เก็บเวคเตอร์ลักษณะเฉพาะ $A=\{(a_i, w_{a_i})\}_{i=1}^{N}$ ของภาพในฐานข้อมูล และสร้างเวคเตอร์ลักษณะเฉพาะ $B=\{(b_i, w_{b_i})\}_{i=1}^{N}$ ของภาพที่ต้องการค้นหา
3. สร้างต้นไม้แผ่ทั่วที่น้อยที่สุดจากยูเนียนเซ็ท $A \cup B$
4. ถ้าต้องการคำนวณทดสอบรันแบบคิดน้ำหนัก

   ใช่ ทดสอบรัน $R(I_1, I_2)=\sum_{i=1}^{M} R(a_i)+\sum_{i=1}^{M} R(b_i)$

   ไม่ใช่ ทดสอบรัน $W=\dfrac{R-E[R]}{\sqrt{Var[R|C]}}$



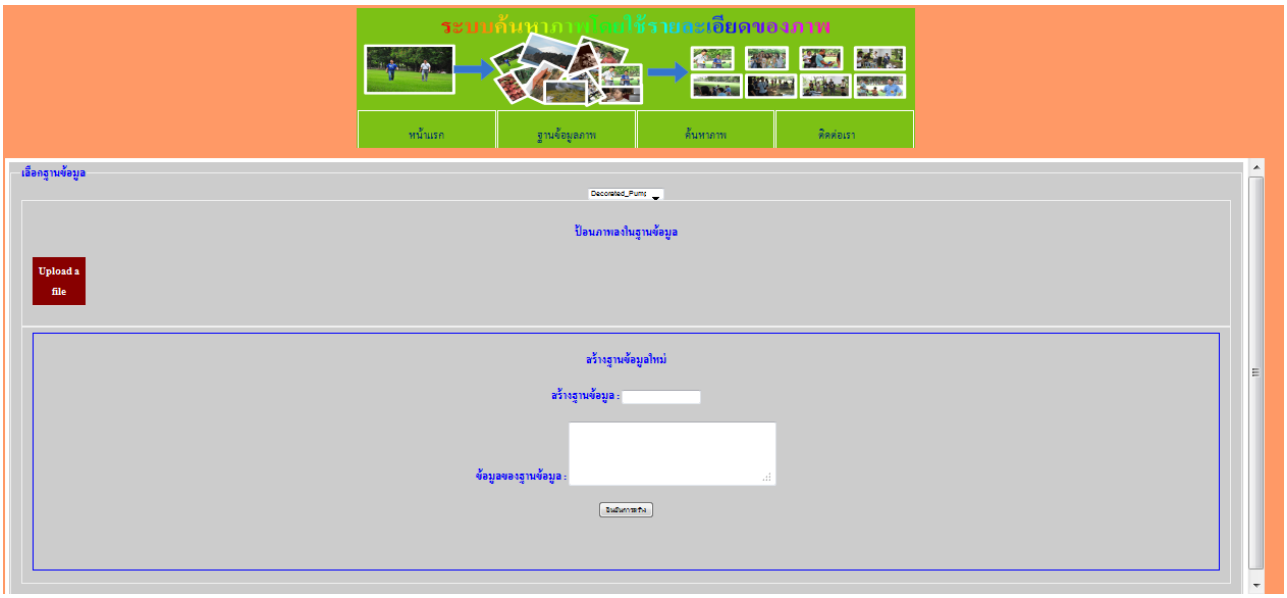
รูปที่ 15. Flowchart การทำงานของโปรแกรม VqRetrieval

4.3 การทำงานของ Web Page ที่ใช้ในการเก็บภาพลงในฐานข้อมูล

ระบบค้นหารูปภาพโดยรายละเอียดของภาพจำเป็นต้องมี Web page สำหรับให้ผู้บริหารระบบใช้ในการเก็บภาพที่ต้องการลงในฐานข้อมูลของระบบ Web Page ถูกพัฒนาโดยภาษา PHP ทำงานร่วมกับ JQuery รูปที่ 16. แสดง Web Page ที่ใช้ในการเก็บภาพลงในฐานข้อมูล ซึ่งสามารถแบ่งออกเป็น 2 ส่วนคือ

1. ส่วนป้อนภาพลงในฐานข้อมูล ผู้บริหารสามารถเลือกฐานข้อมูลที่ต้องการนำภาพไปเก็บและกดปุ่ม Upload a File เพื่อเลือกภาพที่ต้องการนำไปเก็บไว้ในฐานข้อมูล จากนั้นโปรแกรม VqKmeanCalculate จะถูกเรียกเพื่อสร้างไฟล์ YAML ของลักษณะเฉพาะของภาพ จากนั้นไฟล์ภาพและไฟล์ YAML จะถูกนำไปเก็บไว้ในฐานข้อมูล
2. ส่วนสร้างฐานข้อมูลใหม่ เพื่อให้ผู้บริหารสามารถสร้างฐานข้อมูลภาพสำหรับระบบค้นหารูปภาพสำหรับการใช้งานที่แตกต่างกันได้ การใช้งานเริ่มจากกรอกชื่อของฐานข้อมูลภาพลงในช่อง “สร้างฐานข้อมูล” และกรอกคำอธิบายของ

ฐานข้อมูลที่จะสร้างลงในช่อง “ข้อมูลของฐานข้อมูล” จากนั้นจึงกดปุ่ม “ยืนยันการสร้าง” เพื่อสร้างฐานข้อมูลภาพใหม่

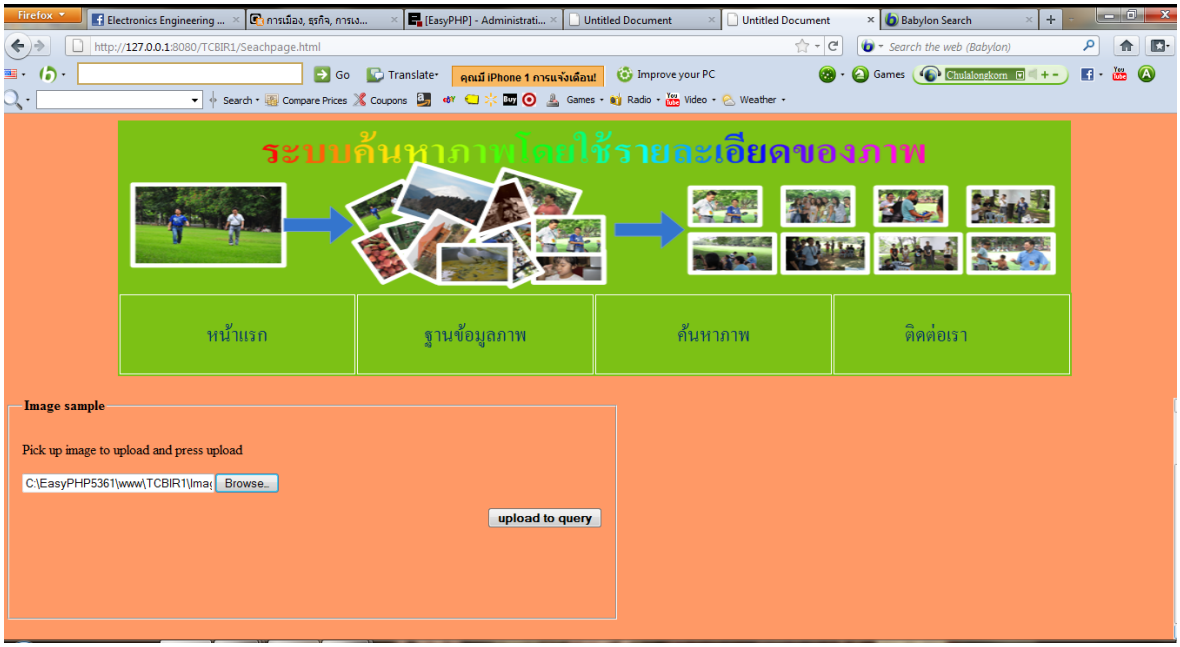


รูปที่ 16. Web page สำหรับเก็บภาพลงในฐานข้อมูล

4.4 การทำงานของ Web Page ที่ใช้ในการค้นหาภาพในฐานข้อมูล

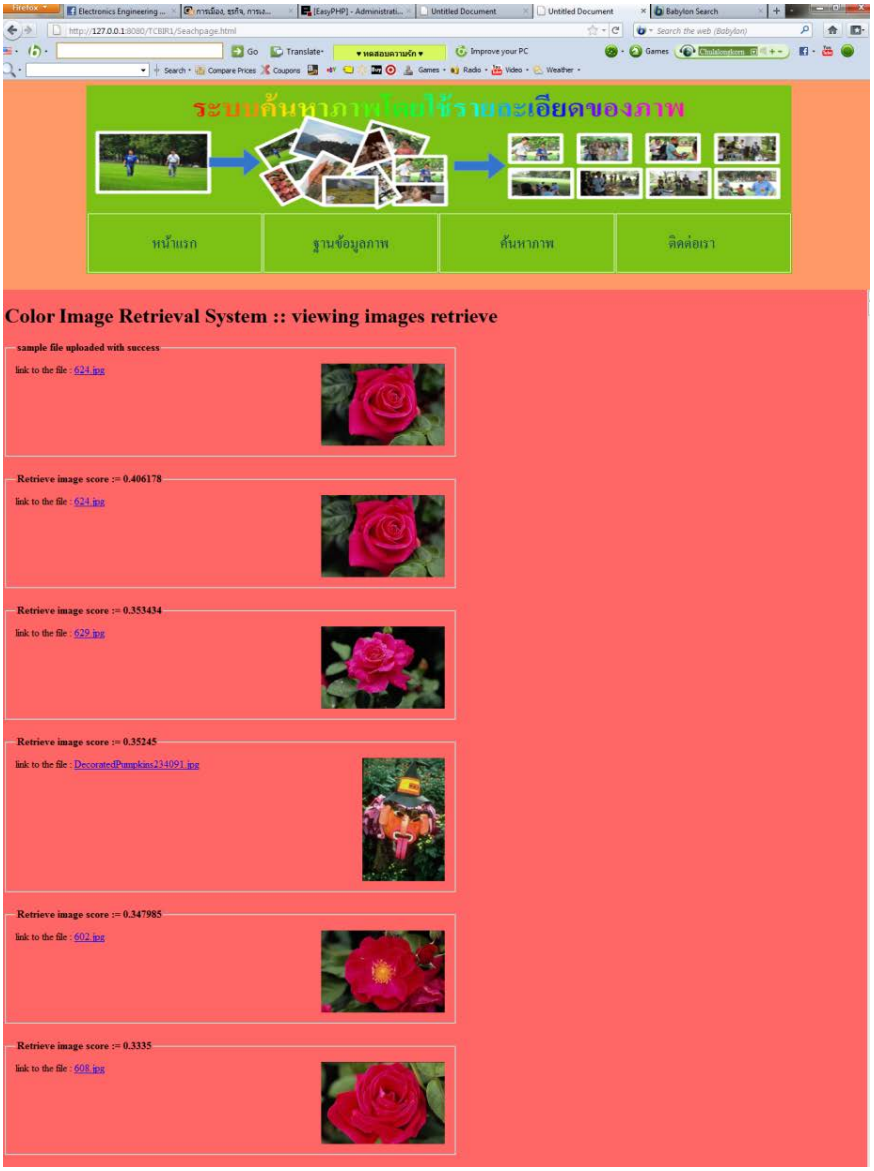
Web Page ที่ใช้ในการค้นหาภาพแบ่งออกเป็น 2 Web Page คือ

1. Web Page ที่ใช้ป้อนภาพที่ต้องการค้นหา ถูกพัฒนาโดยโปรแกรมภาษา PHP ร่วมกับ JQuery เพื่อใช้ป้อนภาพที่ต้องการค้นหาให้กับระบบ ดังแสดงในรูปที่ 17. การใช้งานเริ่มจากผู้ใช้กดปุ่ม “Browse” เพื่อเลือกไฟล์ภาพที่ต้องการค้นหา จากนั้นกดปุ่ม “upload to query” เพื่อส่งไฟล์ภาพที่ต้องการค้นหาให้กับระบบค้นหารูปภาพ



รูปที่ 17. Web page สำหรับป้อนภาพที่ต้องการค้นหา

2. Web Page สำหรับแสดงผลลัพธ์ของการค้นหา ดังแสดงในรูปที่ 18. หลังจากระบบทำการค้นหาภาพในฐานข้อมูลแล้ว ก็จะนำเอาภาพที่เหมือนกับภาพที่ต้องการค้นหามากที่สุดจำนวน 10 รูปแรกมาแสดงผล การแสดงผลเริ่มจากภาพที่ต้องการค้นหาอยู่บนสุด จากนั้นเป็นภาพผลลัพธ์ของการค้นหา โดยเริ่มจากภาพที่เหมือนที่สุด จากนั้นจึงเป็นรูปที่เหมือนอันดับถัดไปจนครบ 10 รูป



รูปที่ 18. Web Page สำหรับแสดงภาพผลลัพธ์ของการค้นหา

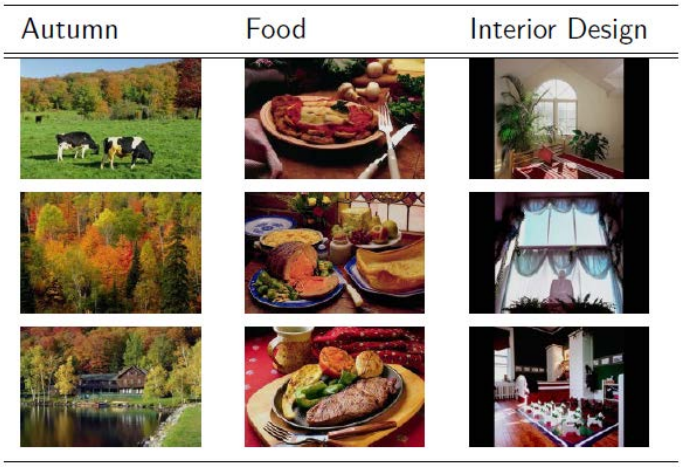
บทที่ 5.

ผลการทดลอง

ฐานข้อมูลภาพที่ใช้ในการทดลองประกอบด้วยรูปภาพทั้งหมด 20 กลุ่ม แต่ละกลุ่มประกอบด้วยรูปทั้งหมด 50 รูป รวมเป็น 1000 รูป โดยที่แต่ละรูปมีขนาดเท่ากับ 192x128 หรือ 128x192 จุดภาพ เพื่อพิสูจน์ว่าระบบการค้นหารูปภาพที่ใช้สามารถนำไปใช้ได้จริง ผู้วิจัยได้พิจารณากลุ่มของรูปภาพอย่างพิถีพิถัน ตัวอย่างเช่นรูปภาพที่ถูกเลือกมีสภาพแสงที่แตกต่างกัน มีฉากหลังของภาพที่หลากหลาย และมีวัตถุที่แตกต่างกันเช่น มนุษย์ สัตว์ และดอกไม้เป็นต้น รายชื่อของกลุ่มของรูปที่ถูกเลือกถูกแสดงไว้ในตารางที่ 1. และตัวอย่างของรูปภาพในแต่ละกลุ่มถูกแสดงไว้ในรูปที่ 19.

ตารางที่ 1. รายชื่อกลุ่มของรูปภาพที่ใช้ในการทดลอง

Africans	Autumn	Buses	Decorated Pumpkins
Decoys	Dinosaurs	Easter Eggs	Firework1
Flowers	Food	French Doors	Glaciers & Mountains
Horses	Interior Design	Kungfu	Land of Pyramids
Mediterranean Cruises	Owls	Royal Guards	Wolves



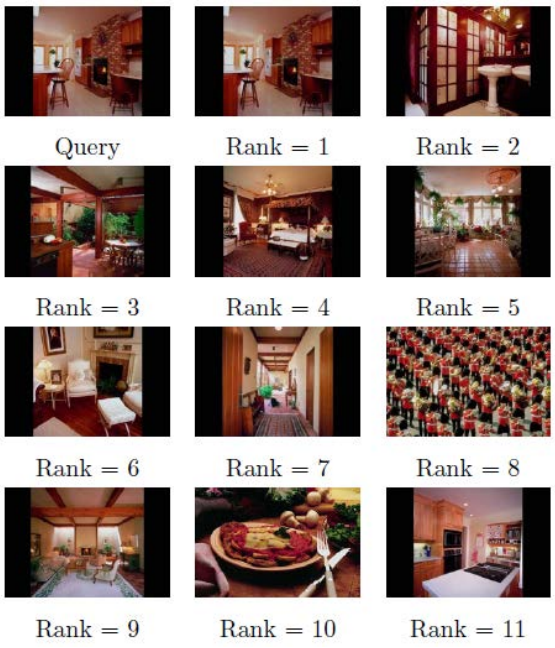
รูปที่ 19. ตัวอย่างของรูปที่ใช้ในการทดลอง

5.1 ขั้นตอนการทดลอง

การทดลองสำหรับวัดประสิทธิภาพของระบบการค้นหารูปภาพในงานวิจัยนี้ เป็นดังนี้คือ

1. เลือกรูปภาพจากฐานข้อมูลจำนวน 1 รูปเพื่อใช้เป็นรูปที่ต้องการค้นหา
2. นำรูปที่ถูกเลือกไปค้นหารูปในฐานข้อมูลภาพ
3. แสดงเฉพาะรูปภาพที่เหมือนที่สุดจำนวน 10 รูปแรก

รูปที่ 20. และ 21. แสดงตัวอย่างผลลัพธ์ของการค้นหาของรูปในกลุ่ม Interior Design และ Autumn จากรูปที่ 20. จะเห็นได้ว่าผลลัพธ์ของการค้นหาใน 10 รูปแรกเป็นรูปในกลุ่ม Interior Design จำนวน 8 รูป และจากรูปที่ 21. มีรูปที่อยู่ในกลุ่ม Autumn จำนวน 8 รูป



รูปที่ 20. ตัวอย่างผลลัพธ์ของการค้นหาของรูปในกลุ่ม Interior Design

รูปที่ 21. ตัวอย่างผลลัพธ์ของการค้นหาของรูปในกลุ่ม Autumn

จากตัวอย่างในรูปที่ 20. และ 21. จะเห็นได้ว่าภาพผลลัพธ์ที่ถูกต้องคือภาพที่อยู่ในกลุ่มเดียวกับภาพที่ต้องการค้นหา เราจะเรียกภาพที่อยู่ในกลุ่มเดียวกับภาพที่ต้องการค้นหาว่าภาพที่เกี่ยวข้อง (Relevant Image) และภาพที่ไม่ได้อยู่ในกลุ่มเดียวกับภาพที่ต้องการค้นหาว่าภาพที่ไม่เกี่ยวข้อง (Irrelevant Image) ระบบการค้นหารูปภาพที่ดีจะต้องให้ผลลัพธ์เป็นภาพที่เกี่ยวข้องจำนวนมาก ดังนั้นค่าประสิทธิภาพของผลลัพธ์ของการค้นหาภาพที่ใช้ในงานวิจัยนี้เราจะใช้ค่า Precision ซึ่งสามารถคำนวณได้จาก

$$\mathrm{Pr} = \frac{N_R}{10} \tag{26}$$

โดย N_R เท่ากับจำนวนรูปภาพผลลัพธ์ที่เกี่ยวข้อง ตัวอย่างเช่นค่า Pr ของตัวอย่างในรูปที่ 20. และ 21. คือ 0.8

และในการทดลองนี้ใช้ค่าเฉลี่ยของ Pr (Average Precision : AP) เพื่อใช้วัดประสิทธิภาพของระบบการค้นหาภาพ ซึ่งขั้นตอนการวัดประสิทธิภาพของระบบการค้นหาภาพที่ใช้ในการทดลองนี้เป็นดังนี้

1. สุ่มเลือกรูป 3 รูปจากแต่ละกลุ่ม รวมเป็นทั้งหมด 60 รูป เพื่อใช้เป็นรูปที่ต้องการค้นหา
2. ป้อนรูปที่ถูกเลือกจากข้อที่ 1 เข้าไปให้ระบบทำการค้นหารูปภาพ
3. ทำการคำนวนหาค่า Pr สำหรับรูปภาพที่ต้องการค้นหาแต่ละรูป
4. คำนวณหาค่าเฉลี่ยของ Pr ทั้งหมดที่ได้จากข้อที่ 3.

ค่าเฉลี่ยของ Pr ที่ได้จากการทดลองดังกล่าวหมายถึงค่าความน่าจะพบภาพที่เกี่ยวข้องในภาพผลลัพธ์ของการค้นหา

5.2 ผลการทดลอง

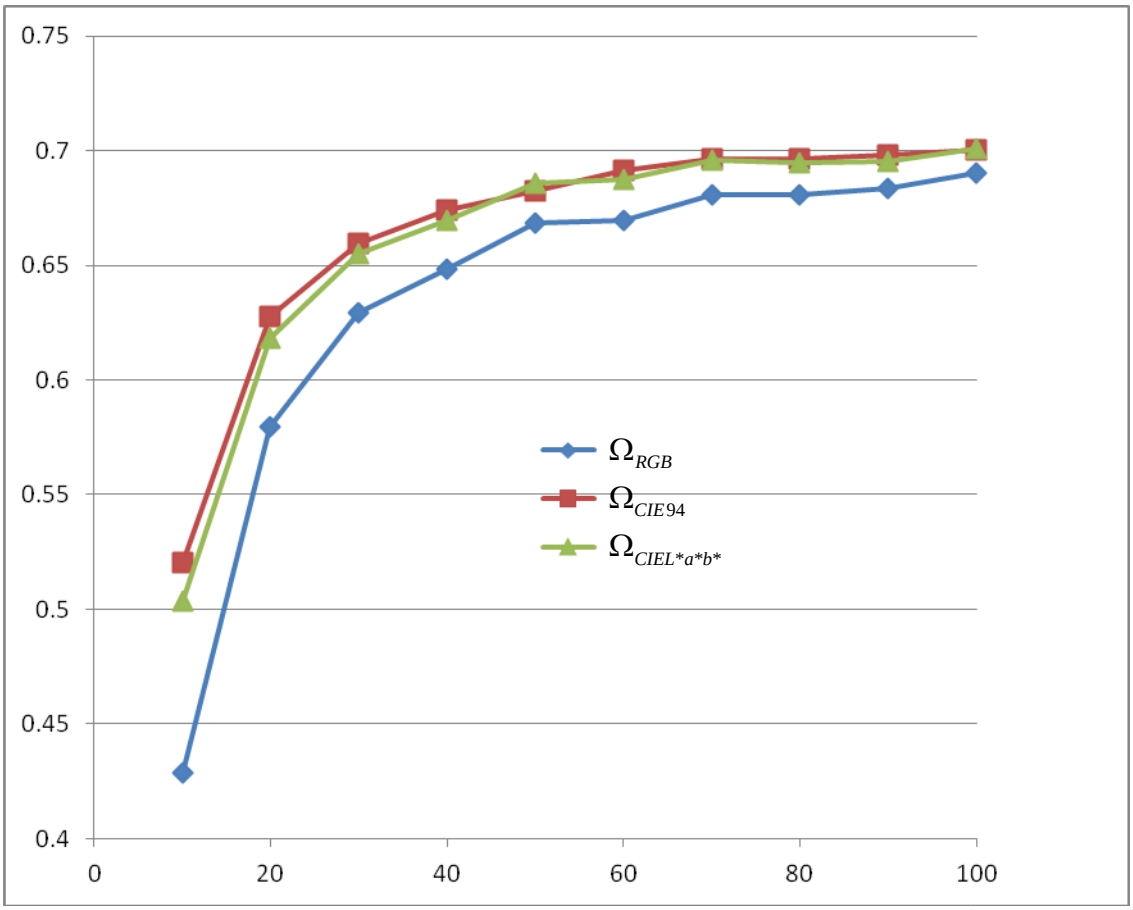
กำหนดให้ Ω_{RGB}, $\Omega_{CIEL*a*b*}$, และ Ω_{CIE94} เป็นปริภูมิ RGB ที่มีการวัดความแตกต่างของสีแบบ ΔE_{RGB}, ปริภูมิ CIE L*a*b* ที่มีการวัดความแตกต่างของสีแบบ $\Delta E_{CIEL*a*b*}$ และปริภูมิ CIE L*a*b* ที่มีการวัดความแตกต่างของสีแบบ ΔE_{CIE94}

ตารางที่ 2. ค่าเฉลี่ยของ Pr สำหรับวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid

จำนวนคลัสเตอร์	Ω_{RGB}	$\Omega_{CIEL*a*b*}$	Ω_{CIE94}
10	0.4286	0.5205	0.5037
20	0.5796	0.6274	0.6184
30	0.6296	0.6594	0.6549
40	0.6481	0.674	0.6694
50	0.6687	0.6824	0.686
60	0.6698	0.6916	0.6875
70	0.6808	0.6966	0.6958
80	0.681	0.6964	0.6948
90	0.6833	0.6983	0.6954
100	0.6903	0.7002	0.7011

งานวิจัยนี้ได้ทำการทดลองเพื่อคำนวณหาค่าเฉลี่ยของ Pr สำหรับทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid และทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมีน้ำหนัก สำหรับปริภูมิ Ω_{RGB}, $\Omega_{CIEL^*a^*b^*}$, และ Ω_{CIE94} และจำนวนคลัสเตอร์ต่างๆ เพื่อค้นหาปริภูมิสีและจำนวนคลัสเตอร์ที่ดีที่สุด และเปรียบเทียบว่าวิธีทดสอบรันแบบใดเป็นวิธีที่ดีที่สุด

ตารางที่ 2. แสดงค่าเฉลี่ยของ Pr สำหรับวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid สำหรับปริภูมิ Ω_{RGB}, $\Omega_{CIEL^*a^*b^*}$, และ Ω_{CIE94} และจำนวนคลัสเตอร์ตั้งแต่ 10 จนถึง 100 และรูปที่ 22. แสดงกราฟของค่าเฉลี่ย Pr ที่ได้จากการทดลอง จากรูปจะเห็นว่าปริภูมิ $\Omega_{CIEL^*a^*b^*}$ และ Ω_{CIE94} มีค่าเฉลี่ยที่ใกล้เคียงกัน โดยที่ปริภูมิ Ω_{CIE94} มีค่าเฉลี่ยที่สูงกว่าเล็กน้อย ดังนั้นปริภูมิ Ω_{CIE94} เป็นปริภูมิที่เหมาะสมที่สุดสำหรับทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid



รูปที่ 22. กราฟค่าเฉลี่ยของ Pr สำหรับวิธีทดสอบรันแบบไม่คิดน้ำหนัก

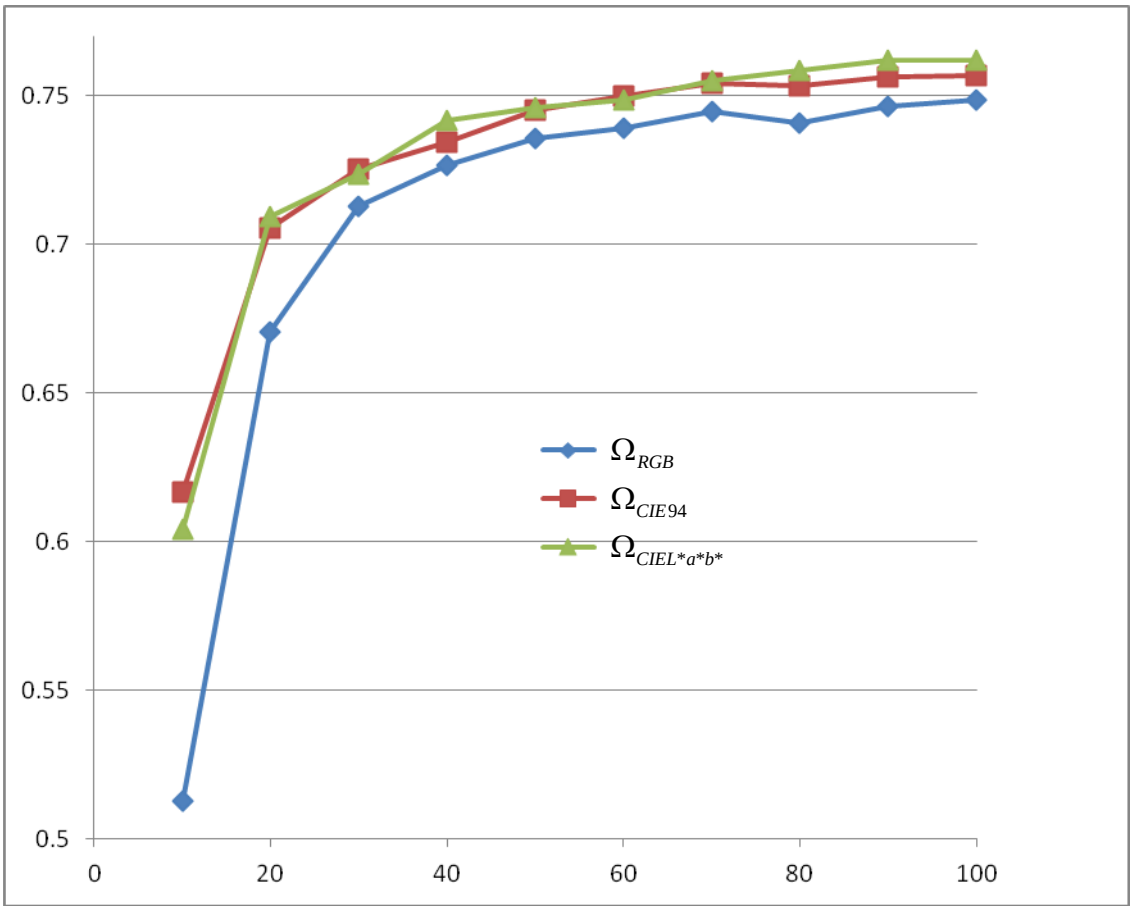
ตารางที่ 3. แสดงค่าเฉลี่ยของ Pr สำหรับวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมีน้ำหนัก สำหรับปริภูมิ Ω_{RGB}, $\Omega_{CIEL*a*b*}$, และ Ω_{CIE94} และจำนวนคลัสเตอร์ตั้งแต่ 10 จนถึง 100 และรูปที่ 23. แสดงกราฟของค่าเฉลี่ย Pr ที่ได้จากการทดลอง จากรูปจะเห็นว่าปริภูมิ $\Omega_{CIEL*a*b*}$ และ Ω_{CIE94} มีค่าเฉลี่ยที่ใกล้เคียงกัน โดยที่ปริภูมิ $\Omega_{CIEL*a*b*}$ มีค่าเฉลี่ยที่สูงกว่าเล็กน้อย ดังนั้นปริภูมิ $\Omega_{CIEL*a*b*}$ เป็นปริภูมิที่เหมาะสมที่สุดสำหรับทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมีน้ำหนัก

ตารางที่ 3. ค่าเฉลี่ยของ Pr สำหรับวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมีน้ำหนัก

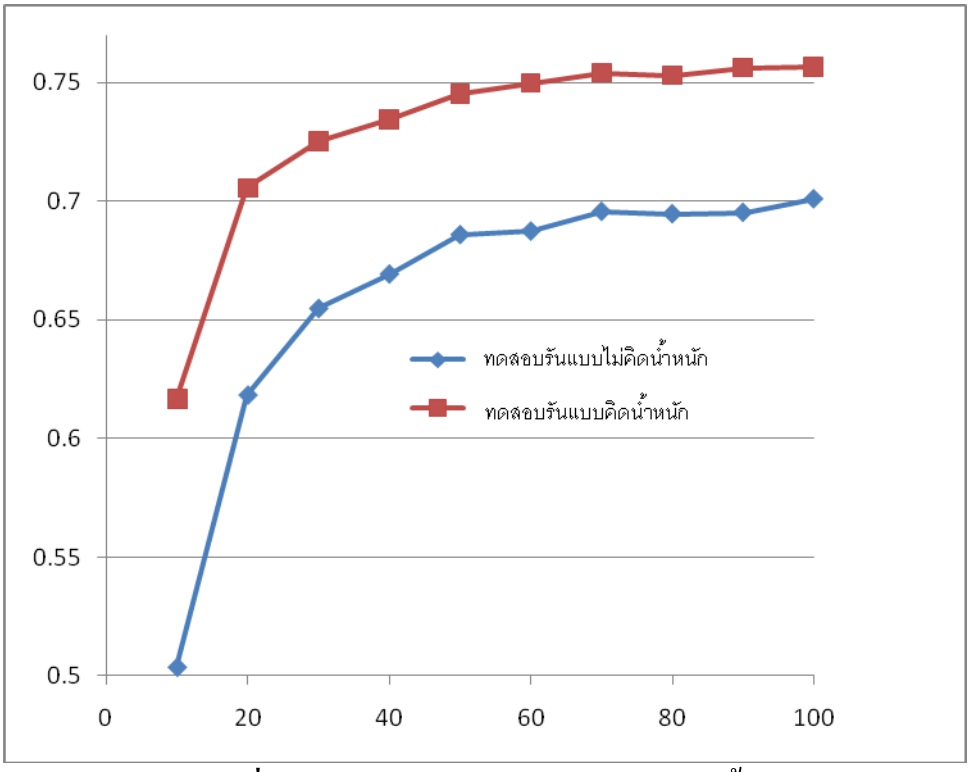
จำนวนคลัสเตอร์	Ω_{RGB}	$\Omega_{CIEL*a*b*}$	Ω_{CIE94}
10	0.5126	0.6165	0.604
20	0.6705	0.7056	0.7093
30	0.7128	0.7251	0.7237
40	0.7264	0.7343	0.7418
50	0.7357	0.7451	0.7459
60	0.7389	0.7499	0.7485
70	0.7445	0.7541	0.7548
80	0.7408	0.7531	0.7585
90	0.7464	0.7563	0.7617
100	0.7486	0.7567	0.762

รูปที่ 24. แสดงการเปรียบเทียบระหว่างกราฟค่าเฉลี่ยของ Pr ของทดสอบรันแบบไม่คิดน้ำหนักสำหรับปริภูมิ Ω_{CIE94} และของทดสอบรันแบบคิดน้ำหนักสำหรับปริภูมิ $\Omega_{CIEL*a*b*}$ จะเห็นได้ว่าค่าเฉลี่ยของ Pr ของทดสอบรันแบบคิดน้ำหนักมีค่าสูงกว่าของทดสอบรันแบบไม่คิดน้ำหนักอย่างมาก ดังนั้นจึงสรุปได้ว่าระบบการค้นหารูปภาพโดยใช้รายละเอียด

ของภาพที่ใช้ทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมีน้ำหนัก และใช้ปริภูมิแบบ $\Omega_{CIEL^*a^*b^*}$ เป็นระบบที่มีประสิทธิภาพมากที่สุด



รูปที่ 23. กราฟค่าเฉลี่ยของ Pr สำหรับวิธีทดสอบรันแบบมีน้ำหนัก



รูปที่ 24. เปรียบเทียบค่าเฉลี่ยของ Pr ของทดสอบรันแบบคิดน้ำหนัก และไม่คิดน้ำหนัก

ระบบการค้นหาภาพที่ดีจะต้องค้นหาภาพผลลัพธ์อย่างมีประสิทธิภาพ และทำงานได้อย่างรวดเร็ว และเนื่องจากความเร็วในการคำนวณของการทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid ทั้งแบบคิดน้ำหนักและไม่คิดน้ำหนัก จะขึ้นกับจำนวนของคลัสเตอร์ ดังนั้นเราจึงจำเป็นต้องเลือกจำนวนคลัสเตอร์ที่เหมาะสม คือได้ค่าเฉลี่ยของ Pr ที่สูงในขณะที่ใช้จำนวนคลัสเตอร์ไม่มาก ซึ่งจากการพิจารณากราฟในรูปที่ 24. จะเห็นว่าจำนวนคลัสเตอร์ที่เหมาะสมของการทดสอบรันแบบคิดน้ำหนักคือ 70 คลัสเตอร์ ซึ่งจะได้ค่าเฉลี่ยเท่ากับ 0.7541

บทที่ 6.

สรุปผลและวิจารณ์

ผู้วิจัยได้ทดลองระบบการค้นหารูปภาพโดยรายละเอียดของภาพ กับภาพสีที่มีคุณสมบัติหลากหลาย เช่นมีฉากหลังและสภาพแสงที่แตกต่างกัน และเป็นภาพถ่ายของมนุษย์ สัตว์และดอกไม้เป็นต้น เพื่อให้แน่ใจว่าระบบที่พัฒนาขึ้นสามารถนำไปใช้งานได้จริง และจากการทดลองสามารถสรุปได้ว่าวิธีทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมีน้ำหนัก โดยใช้ปริภูมิ $\Omega_{CIEL^*a^*b^*}$ และใช้คลัสเตอร์จำนวน 70 คลัสเตอร์ เป็นวิธีการค้นหาที่มีประสิทธิภาพมากที่สุด ซึ่งจะให้ค่าเฉลี่ยของ Pr เท่ากับ 0.7541

แนวทางในการพัฒนาในอนาคตคือ จะนำเอาวิธีการทดสอบรันแบบหลายมิติของ Wald และ Wolfowitz ที่ขึ้นกับ Centroid แบบมีน้ำหนักไปประยุกต์ใช้กับระบบค้นหาภาพชนิดอื่นๆ เช่นระบบค้นหารูปภาพทางการแพทย์ ซึ่งแพทย์สามารถนำไปใช้ประโยชน์ในการเพิ่มประสิทธิภาพในการรักษาผู้ป่วยให้ดียิ่งขึ้นได้

บทที่ 7.

โปรแกรม

7.1 โปรแกรม VqKmeanCalculate

7.1.1 VQ_KmeanKDTree.h

```
#include "opencv2/opencv.hpp"
#include <vector>
#include <algorithm>

using namespace cv;

typedef struct{
	double B,G,R;
} RgbPixelDouble;

typedef struct{
	float B,G,R;
} RgbPixelFloat;

typedef struct{
	double C0 ,C1 ,C2;
} centerDouble;

typedef struct{
	float C0 ,C1 ,C2;
} centerFloat;

#pragma once
class VQ_KmeanKDTree
{
public:
	VQ_KmeanKDTree(void);
	~VQ_KmeanKDTree(void);
	void setDataImageBGR(Mat &srcImg);
	void setDataImageBGROpenCV(Mat &srcImg,bool isUnique);
	void kmeansExecute(int cluster,int loopNumber);
	void openCVKmeansExecute(int cluster,int loopNumber);
	bool kmeansExecuteOpenCV(int cluster_count,int loopNumber);
	bool writeDataYAML(char *filename);
	vector<int> eachClusterNumber;
	vector<centerDouble> eachDoubleClusterCenter;
	vector<centerFloat>  eachFloatClusterCenter;
private:
	vector<int> vectorPixelColorAll;
	vector<RgbPixelDouble> vectorDoubleUniquePixelRGB;
	vector<float>  vectorFloatUniquePixelRGB;
	Mat pointSamples;
	Mat centerCluster;
	Mat eachClusterCount;
	void uniqueData(void);
	void prepare2Double(void);
	void prepare2FloatForOpenCV(void);
	void prepareDataWithUnique(Mat &srcImg,Mat &pointSamples);
	void prepareDataWithOutUnique(Mat &srcImg,Mat &pointSamples);
};
```

7.1.2 VQ_KmeanKDTree.cpp

```
#include "VQ_KmeanKDTree.h"

VQ_KmeanKDTree::VQ_KmeanKDTree(void)
{
}

VQ_KmeanKDTree::~VQ_KmeanKDTree(void)
{
	if(this->pointSamples.refcount!=NULL)
		this->pointSamples.release();
}

void VQ_KmeanKDTree::setDataImageBGROpenCV(Mat &srcImg,bool isUnique)
{
	if (isUnique)
	{
		this->prepareDataWithUnique(srcImg,this->pointSamples);
	}else
	{
		this->prepareDataWithOutUnique(srcImg,this->pointSamples);
	}
}

void VQ_KmeanKDTree::prepareDataWithUnique(Mat &srcImg,Mat &pointSamples)
{
	int lenghtPixel = srcImg.rows * srcImg.cols;

	vector<int> vectorIntTemp(lenghtPixel);

	MatIterator_<Vec3b> it = srcImg.begin<Vec3b>(),it_end =
srcImg.end<Vec3b>();

	int tempRGBint;
	for(int i=0; it != it_end; ++it,++i)
	{
		tempRGBint = (int)(*it)[0];
		tempRGBint += ((int)(*it)[1])<<8;
		tempRGBint += ((int)(*it)[2])<<16;
		vectorIntTemp[i] = tempRGBint;
	}

	// unique process using STL
	sort( vectorIntTemp.begin(), vectorIntTemp.end() );
	vectorIntTemp.erase( unique( vectorIntTemp.begin(), vectorIntTemp.end() ),
vectorIntTemp.end() );

	Mat processMat(vectorIntTemp.size(),1, CV_MAKETYPE(CV_8U,3));

	it = processMat.begin<Vec3b>();
	it_end = processMat.end<Vec3b>();

	int intBGR;
	int mask = 255;
	for(int i=0; it != it_end; ++it,++i)
	{
		intBGR = vectorIntTemp[i];
		(*it)[0] =  saturate_cast<uchar>(intBGR & mask);
		(*it)[1] =  saturate_cast<uchar>((intBGR>>8) & mask);
		(*it)[2] =  saturate_cast<uchar>((intBGR>>16) & mask);
	}
	//Mat pointSamples;
```

```
	processMat.convertTo(pointSamples, CV_32FC3,1.0/255.0,0.0);
	processMat.release();

	//return pointSamples;
}

void VQ_KmeanKDTree::prepareDataWithOutUnique(Mat &srcImg,Mat &pointSamples)
{
	//Mat pointSamples;
	srcImg.convertTo(pointSamples, CV_32FC3,1.0/255.0,0.0);
	pointSamples = pointSamples.reshape(3, srcImg.rows*srcImg.cols);
	//return pointSamples;
}

void VQ_KmeanKDTree::setDataImageBGR(Mat &srcImg)
{
	int lenghtPixel = srcImg.rows * srcImg.cols;
	this->vectorPixelColorAll.resize(lenghtPixel);
	uchar* data    = (uchar *) srcImg.data;
	int intBRG;
	for (int i =0;i<lenghtPixel;i++)
	{
		intBRG = (int)(*(data++));
		intBRG += (int)(*(data++))<<8;
		intBRG += (int)(*(data++))<<16;
		this->vectorPixelColorAll[i] = intBRG;
	}
	this->uniqueData();
}

void VQ_KmeanKDTree::uniqueData(void)
{
	sort( vectorPixelColorAll.begin(), vectorPixelColorAll.end() );
	vectorPixelColorAll.erase( unique( vectorPixelColorAll.begin(),
vectorPixelColorAll.end() ), vectorPixelColorAll.end() );
}
void VQ_KmeanKDTree::prepare2Double(void)
{
	int nPts = vectorPixelColorAll.size();
	this->vectorDoubleUniquePixelRGB.resize(nPts);
	int intBGR;
	int mask = 255;
	for (int j = 0; j < nPts; j++) {
		intBGR = vectorPixelColorAll[j];
		this->vectorDoubleUniquePixelRGB[j].B =  ((double)(intBGR &
mask))/255.0;
		this->vectorDoubleUniquePixelRGB[j].G =  ((double)((intBGR>>8) &
mask))/255.0;
		this->vectorDoubleUniquePixelRGB[j].R =  ((double)((intBGR>>16) &
mask))/255.0;
	}
}

void VQ_KmeanKDTree::prepare2FloatForOpenCV(void)
{
	int nPts = vectorPixelColorAll.size();
	this->vectorFloatUniquePixelRGB.resize((nPts*3));

	int intBGR;
	int mask = 255;
	for (int j = 0; j < nPts; j++) {
		intBGR = vectorPixelColorAll[j];
		this->vectorFloatUniquePixelRGB[(j*3)] =  ((float)(intBGR &
mask))/255.0f;
```

```
            this->vectorFloatUniquePixelRGB[(j*3+1)] =  ((float)((intBGR>>8) &
mask))/255.0f;
            this->vectorFloatUniquePixelRGB[(j*3+2)] =  ((float)((intBGR>>16) &
mask))/255.0f;
      }
}

void VQ_KmeanKDTree::openCVKmeansExecute(int cluster,int loopNumber)
{
      int cluster_count = cluster; /* number of cluster */
      this->prepare2FloatForOpenCV();

      Mat src_img = imread("c:””baboon.jpg");

      int lenghtPixel = src_img.rows * src_img.cols;

      vector<int> vectorIntTemp(lenghtPixel);

      MatIterator_<Vec3b> it = src_img.begin<Vec3b>(),
            it_end = src_img.end<Vec3b>();
      int tempRGBint;


      for(int i=0; it != it_end; ++it,++i)
      {
            tempRGBint = (int)(*it)[0];
            tempRGBint += ((int)(*it)[1])<<8;
            tempRGBint += ((int)(*it)[2])<<16;
            vectorIntTemp[i] = tempRGBint;
      }

      cout << "non unique number :" << vectorIntTemp.size() << endl;
      sort( vectorIntTemp.begin(), vectorIntTemp.end() );
      vectorIntTemp.erase( unique( vectorIntTemp.begin(), vectorIntTemp.end() ),
vectorIntTemp.end() );
      cout << "unique number :" << vectorIntTemp.size() << endl;

      Mat processMat(vectorIntTemp.size(),1, CV_MAKETYPE(CV_8U,3));

      it = processMat.begin<Vec3b>();
      it_end = processMat.end<Vec3b>();

      int intBGR;
      int mask = 255;
      for(int i=0; it != it_end; ++it,++i)
      {
            intBGR = vectorIntTemp[i];
            (*it)[0] =  saturate_cast<uchar>(intBGR & mask);
            (*it)[1] =  saturate_cast<uchar>((intBGR>>8) & mask);
            (*it)[2] =  saturate_cast<uchar>((intBGR>>16) & mask);
      }

      Mat pointSamples;
      processMat.convertTo(pointSamples, CV_32FC3,1.0/255.0,0.0);
      Mat_<int> clusters(pointSamples.size(), CV_32SC1);
      // (3)run k-means clustering algorithm to segment pixels in RGB color
space
      //Mat_<int> clusters(points.size(), CV_32SC1);
      //cv::Mat centers(cluster_count,1,CV_32FC3);
      cv::Mat centers;
      //kmeans(pointSamples, cluster_count,
clusters,cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, loopNumber, 1.0), 1,
KMEANS_PP_CENTERS, &centers);
```

```
	kmeans(pointSamples, cluster_count,
clusters,cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, loopNumber, 1.0), 1,
KMEANS_PP_CENTERS, &centers);
	centers = centers.reshape(3, cluster_count);

	MatIterator_<Vec3f> itf     = centers.begin<Vec3f>();
	MatIterator_<Vec3f> itf_end = centers.end<Vec3f>();

	for(int i=0; itf != itf_end; ++itf,++i) {
		cout << (*itf)[0] << " ";
		cout << (*itf)[1] << " ";
		cout << (*itf)[2]<< endl;
	}
}

bool VQ_KmeanKDTree::kmeansExecuteOpenCV(int cluster_count,int loopNumber)
{
	if ((this->pointSamples.refcount==NULL))
		return false;

	Mat clusters = cv::Mat_<int>(this->pointSamples.size(),CV_32SC1);

	// (3)run k-means clustering algorithm to segment pixels in RGB color
space
	//kmeans(pointSamples, cluster_count,
clusters,cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, loopNumber, 1.0), 1,
KMEANS_PP_CENTERS, &centers);
	//kmeans(pointSamples, cluster_count,
clusters,cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, loopNumber, 1.0), 1,
KMEANS_USE_INITIAL_LABELS, &centers);
	cv::kmeans(pointSamples, cluster_count,
clusters,cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, loopNumber, 0.5), 1,
KMEANS_PP_CENTERS, &this->centerCluster);

	this->centerCluster = this->centerCluster.reshape(3, cluster_count);
	MatIterator_<Vec3f> itf     = this->centerCluster.begin<Vec3f>();
	MatIterator_<Vec3f> itf_end = this->centerCluster.end<Vec3f>();

	this->eachClusterCount = cv::Mat_<float>(cluster_count,1,CV_32FC1);
	this->eachClusterCount.setTo(cv::Scalar(0.0));
	MatIterator_<Vec<float, 1>> itff         = this-
>eachClusterCount.begin<Vec<float, 1>>();

	Mat tempCount = cv::Mat_<int>(cluster_count,1,CV_32SC1);
	tempCount.setTo(cv::Scalar(0.0));

	MatIterator_<Vec<int, 1>> itI         = tempCount.begin<Vec<int, 1>>();
	MatIterator_<Vec<int, 1>> itI_end     = tempCount.end  <Vec<int, 1>>();
	MatIterator_<Vec<int, 1>> itItmp;

	MatIterator_<Vec<int, 1>> itIC     = clusters.begin<Vec<int, 1>>();
	MatIterator_<Vec<int, 1>> itIC_end = clusters.end<Vec<int, 1>>();

	for(int i = 0; itIC != itIC_end; ++itIC) {
		int temp = (*itIC)[0];
		itItmp = itI+temp;
		(*itItmp)[0]++;
	}

	float totalPixel  = (float)this->pointSamples.rows;
	for(int i = 0; itI != itI_end; ++itI,++itff,++i) {
		(*itff)[0] = ((float)((*itI)[0]))/totalPixel;
		cout <<i<<" :: "<< (*itff)[0] << " ";
		cout << endl;
```

```
	}
	return true;
}

bool VQ_KmeanKDTree::writeDataYAML(char *filename)
{
	if (this->centerCluster.refcount==NULL)
		return false;
	cv::FileStorage fs(filename, cv::FileStorage::WRITE);
	fs << "ClusterCenter" << this->centerCluster;
	fs << "ClusterWeight" << this->eachClusterCount;
	fs.release();
	return true;
}

void VQ_KmeanKDTree::kmeansExecute(int clusterNumber,int loopNumber)
{
	KMterm  term(loopNumber, 0, 0, 0,		// run for 100 stages
		0.10, 0.10, 3,			// other typical parameter values
		0.50, 10, 0.95);

	this->prepare2Double();
	int dim = 3;
	int nPts = this->vectorDoubleUniquePixelRGB.size();
	KMpointArray source = kmAllocPts(nPts, dim);
	for (int j =0;j<nPts;j++)
	{
		source[j][0] = this->vectorDoubleUniquePixelRGB[j].B;
		source[j][1] = this->vectorDoubleUniquePixelRGB[j].G;
		source[j][2] = this->vectorDoubleUniquePixelRGB[j].R;
	}
	KMdata *dataPts = new KMdata(dim, nPts);
	kmCopyPts(nPts,dim,source,dataPts->getPts());

	kmDeallocPts(source);
	dataPts->buildKcTree();			// build filtering structure

	KMfilterCenters *ctrs = new KMfilterCenters(clusterNumber, *dataPts);
	// allocate centers
	KMlocalEZ_Hybrid kmAlg(*ctrs, term);
	*ctrs = kmAlg.execute();			// execute

	double*		sqDist = new double[nPts];
	KMctrIdxArray newCands = new KMctrIdx[nPts];
	ctrs->getAssignments(newCands,sqDist);
	eachClusterNumber.resize(clusterNumber);

	for (int j =0;j<nPts;j++)
	{
		eachClusterNumber[newCands[j]]++;
	}
	delete []sqDist;
	delete []newCands;

	eachDoubleClusterCenter.resize(clusterNumber);
	KMcenterArray centerPoint = ctrs->getCtrPts();
	for (int j =0;j<clusterNumber;j++)
	{
		eachDoubleClusterCenter[j].C0 = centerPoint[j][0];
		eachDoubleClusterCenter[j].C1 = centerPoint[j][1];
		eachDoubleClusterCenter[j].C2 = centerPoint[j][2];
	}
	delete ctrs;
	delete dataPts;
```

```
	for (int j =0;j<clusterNumber;j++)
	{
		cout << eachDoubleClusterCenter[j].C0 << " , ";
		cout << eachDoubleClusterCenter[j].C1 << " , ";
		cout << eachDoubleClusterCenter[j].C2 << endl;
	}
	return;
}
```

7.1.3 kmeanCluster.h

```
#include "opencv2/opencv.hpp"
#include <vector>
#include <algorithm>

using namespace cv;

#pragma once
class kmeanCluster
{
public:
	kmeanCluster(void);
	~kmeanCluster(void);
	void setDataImageBGROpenCV(Mat &srcImg,bool isUnique);
	bool kmeansExecuteOpenCV(int cluster_count,int loopNumber);
	bool writeDataYAML(const char *filename);

private:
	Mat pointSamples;
	Mat centerCluster;
	Mat eachClusterCount;
	void prepareDataWithUnique(Mat &srcImg,Mat &pointSamples);
	void prepareDataWithOutUnique(Mat &srcImg,Mat &pointSamples);
};
```

7.1.4 kmeanCluster.cpp

```
#include "kmeanCluster.h"

kmeanCluster::kmeanCluster(void)
{
}

kmeanCluster::~kmeanCluster(void)
{
}

void kmeanCluster::setDataImageBGROpenCV(Mat &srcImg,bool isUnique)
{
	if (isUnique)
	{
		this->prepareDataWithUnique(srcImg,this->pointSamples);
	}else
	{
		this->prepareDataWithOutUnique(srcImg,this->pointSamples);
	}
}

void kmeanCluster::prepareDataWithUnique(Mat &srcImg,Mat &pointSamples)
{
	int lenghtPixel = srcImg.rows * srcImg.cols;
```

```
	vector<int> vectorIntTemp(lenghtPixel);

	MatIterator_<Vec3b> it = srcImg.begin<Vec3b>(),it_end =
srcImg.end<Vec3b>();

	int tempRGBint;
	for(int i=0; it != it_end; ++it,++i)
	{
		tempRGBint = (int)(*it)[0];
		tempRGBint += ((int)(*it)[1])<<8;
		tempRGBint += ((int)(*it)[2])<<16;
		vectorIntTemp[i] = tempRGBint;
	}

	// unique process using STL
	sort( vectorIntTemp.begin(), vectorIntTemp.end() );
	vectorIntTemp.erase( unique( vectorIntTemp.begin(), vectorIntTemp.end() ),
vectorIntTemp.end() );

	Mat processMat(vectorIntTemp.size(),1, CV_MAKETYPE(CV_8U,3));

	it = processMat.begin<Vec3b>();
	it_end = processMat.end<Vec3b>();

	int intBGR;
	int mask = 255;
	for(int i=0; it != it_end; ++it,++i)
	{
		intBGR = vectorIntTemp[i];
		(*it)[0] =  saturate_cast<uchar>(intBGR & mask);
		(*it)[1] =  saturate_cast<uchar>((intBGR>>8) & mask);
		(*it)[2] =  saturate_cast<uchar>((intBGR>>16) & mask);
	}
	//Mat pointSamples;
	processMat.convertTo(pointSamples, CV_32FC3,1.0/255.0,0.0);
	processMat.release();
	//return pointSamples;
}

void kmeanCluster::prepareDataWithOutUnique(Mat &srcImg,Mat &pointSamples)
{
	srcImg.convertTo(pointSamples, CV_32FC3,1.0/255.0,0.0);
	pointSamples = pointSamples.reshape(3, srcImg.rows*srcImg.cols);
}

bool kmeanCluster::kmeansExecuteOpenCV(int cluster_count,int loopNumber)
{
	if ((this->pointSamples.refcount==NULL))
		return false;

	Mat clusters = cv::Mat_<int>(this->pointSamples.size(),CV_32SC1);

	// (3)run k-means clustering algorithm to segment pixels in RGB color
space
	//kmeans(pointSamples, cluster_count,
clusters,cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, loopNumber, 1.0), 1,
KMEANS_PP_CENTERS, &centers);
	//kmeans(pointSamples, cluster_count,
clusters,cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, loopNumber, 1.0), 1,
KMEANS_USE_INITIAL_LABELS, &centers);
	cv::kmeans(pointSamples, cluster_count,
clusters,cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, loopNumber, 0.5), 1,
KMEANS_PP_CENTERS, this->centerCluster);
	this->centerCluster = this->centerCluster.reshape(3, cluster_count);
```

```
	MatIterator_<Vec3f> itf     = this->centerCluster.begin<Vec3f>();
	MatIterator_<Vec3f> itf_end = this->centerCluster.end<Vec3f>();

	this->eachClusterCount = cv::Mat_<float>(cluster_count,1,CV_32FC1);
	this->eachClusterCount.setTo(cv::Scalar(0.0));
	MatIterator_<Vec<float, 1>> itff         = this-
>eachClusterCount.begin<Vec<float, 1>>();

	Mat tempCount = cv::Mat_<int>(cluster_count,1,CV_32SC1);
	tempCount.setTo(cv::Scalar(0.0));

	MatIterator_<Vec<int, 1>> itI         = tempCount.begin<Vec<int, 1>>();
	MatIterator_<Vec<int, 1>> itI_end     = tempCount.end  <Vec<int, 1>>();
	MatIterator_<Vec<int, 1>> itItmp;

	MatIterator_<Vec<int, 1>> itIC     = clusters.begin<Vec<int, 1>>();
	MatIterator_<Vec<int, 1>> itIC_end = clusters.end<Vec<int, 1>>();

	for(int i = 0; itIC != itIC_end; ++itIC) {
		int temp = (*itIC)[0];
		itItmp = itI+temp;
		(*itItmp)[0]++;
	}

	float totalPixel  = (float)this->pointSamples.rows;
	for(int i = 0; itI != itI_end; ++itI,++itff,++i) {
		(*itff)[0] = ((float)((*itI)[0]))/totalPixel;
	}
	return true;
}

bool kmeanCluster::writeDataYAML(const char *filename)
{
	if (this->centerCluster.refcount==NULL)
		return false;
	cv::FileStorage fs(filename, cv::FileStorage::WRITE);
	fs << "ClusterCenter" << this->centerCluster;
	fs << "ClusterWeight" << this->eachClusterCount;
	fs.release();
	return true;
}
```

7.2 โปรแกรม VqRetrieval

7.2.1 yamlWriteRead.h

```
#include "opencv2/opencv.hpp"
#include <vector>
#include <algorithm>
using namespace cv;

#pragma once
class yamlWriteRead
{
public:
	bool readDataVq(char* filename, Mat &centerCluster, Mat
&eachClusterCount);
	bool readDataVqConst(const char* filename, Mat &centerCluster, Mat
&eachClusterCount);
	bool writeDataYAML(char* filename, Mat &centerCluster, Mat
&eachClusterCount);
	yamlWriteRead(void);
```

```
	~yamlWriteRead(void);
};
```

7.2.2 yamlWriteRead.cpp

```
#include "yamlWriteRead.h"

yamlWriteRead::yamlWriteRead(void)
{
}

yamlWriteRead::~yamlWriteRead(void)
{
}

bool yamlWriteRead::readDataVq(char* filename, Mat &centerCluster, Mat
&eachClusterCount)
{
	FileStorage fsRead(filename, FileStorage::READ);

	fsRead["ClusterCenter"] >> centerCluster;
	fsRead["ClusterWeight"] >> eachClusterCount;
	fsRead.release();
	return true;
}

bool yamlWriteRead::readDataVqConst(const char* filename, Mat &centerCluster,
Mat &eachClusterCount)
{
	FileStorage fsRead(filename, FileStorage::READ);

	fsRead["ClusterCenter"] >> centerCluster;
	fsRead["ClusterWeight"] >> eachClusterCount;
	fsRead.release();
	return true;
}

bool yamlWriteRead::writeDataYAML(char* filename, Mat &centerCluster, Mat
&eachClusterCount)
{
	cv::FileStorage fs(filename, cv::FileStorage::WRITE);
	fs << "ClusterCenter" << centerCluster;
	fs << "ClusterWeight" << eachClusterCount;
	fs.release();
	return true;
}
```

7.2.3 VqRetrieval.cpp

```
#include "yamlWriteRead.h"
#include <opencv2/opencv.hpp>
#include <emmintrin.h>
#include <vector>
#include <string>
#include <libmysqlwrapped.h>
#include <mysql.h>

#include <iostream>                 // for std::cout
#include <utility>                  // for std::pair
#include <algorithm>                // for std::for_each
#include <boost/graph/graph_traits.hpp>
#include <boost/graph/adjacency_list.hpp>
#include <boost/graph/dijkstra_shortest_paths.hpp>
```

```
#include <boost/config.hpp>
#include <iostream>
#include <boost/graph/prim_minimum_spanning_tree.hpp>

#include <vector>
#include <math.h>

#include "kmeanCluster.h"
using namespace boost;

using namespace cv;
using namespace std;

struct Centet3Dfloat {
	float c0;
	float c1;
	float c2;
} ;

struct dbInfo{
	long Id;
	string imagePath;
	string yamlPath;
};

struct dbConnect{
	string dbHost;
	string dbUser;
	string dbPass;
	string dbDBName;
};

vector<dbInfo> readDataDbMysql(dbConnect dbConn,string DBID);
void MWWTestCompute(Mat & sampleMat,Mat & compareMat,Mat &samplePopulaWeight,Mat
& comparePopulaWeight,int &MWWTestVaule,double & MWWTestWeightVaule);
void MWWTestWeightCompute(void);

int main(int argc, char **argv)
{
	// 0 name program
	// 1 filepath image input
	// 2 cluster number
	// 3 loop number
	// 4 db file for sqlite
	// 5 flag for 0 for MWWTest other for MWWTestWeight

	if (argc != 10)
		return -1;

	const char *imagename = argv[1];
	const int clusterNum  = atoi(argv[2]);
	const int loopNum     = atoi(argv[3]);
	const int flagChooseMWW = atoi(argv[4]);

	dbConnect dbConn;
	string DBID;
	dbConn.dbHost   = argv[5];
	dbConn.dbUser   = argv[6];
	dbConn.dbPass   = argv[7];

	if (dbConn.dbPass.compare("null")==0 || dbConn.dbPass.compare("NULL")==0)
	{
		dbConn.dbPass = "";
```

```
	}

	dbConn.dbDBName = argv[8];
	DBID            = argv[9];

	if (DBID.compare("null")==0 || DBID.compare("NULL")==0)
	{
		DBID = "";
	}

	if (!(clusterNum < 100 && clusterNum > 1 && loopNum < 200 && loopNum >
10))
		return -1;

	Mat src_img = imread(imagename, 1);
	if(!src_img.data || src_img.channels()!=3)
		return -1;

	kmeanCluster kCluster;
	kCluster.setDataImageBGROpenCV(src_img,false);

	if (!(kCluster.kmeansExecuteOpenCV(clusterNum,loopNum)))
		return -1;

	Mat sampleCenterCluster;
	Mat sampleEachClusterCount;
	kCluster.getKmeanData(sampleCenterCluster,sampleEachClusterCount);
	yamlWriteRead yamlWR;

	vector<dbInfo> vecListFile = readDataDbMysql(dbConn,DBID);

	if (vecListFile.size()<1)
	{
		return -1;
	}

	vector<std::pair<Mat,Mat>> vqDataAllImage(vecListFile.size());

	vector<dbInfo>::iterator it;
	vector<std::pair<Mat,Mat>>::iterator itYaml;

	map<int,string> MWWTestMapList;
	map<double,string> MWWTestWeightMapList;

	for ( it=vecListFile.begin(),itYaml = vqDataAllImage.begin()  ; it !=
vecListFile.end(); it++, itYaml++ )
	{
		string finalPath = it->yamlPath;

		Mat centerCluster;
		Mat eachClusterCount;
		yamlWR.readDataVqConst( finalPath.c_str()
,centerCluster,eachClusterCount);

		int MWWTestVaule = 0;
		double MWWTestWeightVaule = 0.0f;
		MWWTestCompute(sampleCenterCluster, centerCluster,
sampleEachClusterCount, eachClusterCount, MWWTestVaule, MWWTestWeightVaule);
		MWWTestMapList[MWWTestVaule] = it->imagePath;
		MWWTestWeightMapList[MWWTestWeightVaule] = it->imagePath;
	}

	int i =0;
	if (flagChooseMWW)
```

```
	{
		map<double,string>::reverse_iterator ritDouble;
		// show content:
		for ( ritDouble=MWWTestWeightMapList.rbegin() ; ritDouble !=
MWWTestWeightMapList.rend() && i<10 ; ritDouble++ ,i++)
			cout << ritDouble->first << ";" << ritDouble->second << endl;
		return 1;
	}
	map<int,string>::reverse_iterator ritInt;
	// show content:
	for ( ritInt=MWWTestMapList.rbegin() ; ritInt != MWWTestMapList.rend() &&
i<10; ritInt++ ,i++)
		cout << ritInt->first << ";" << ritInt->second << endl;
	return 1;
}

vector<dbInfo> readDataDbMysql(dbConnect dbConn,string DBID)
{
	vector<dbInfo> vecResult;
	Database db(dbConn.dbHost,dbConn.dbUser,dbConn.dbPass,dbConn.dbDBName);


	Query q(db);

	if (!db.Connected() || !q.Connected())
		return vecResult;
	// retrieve data
	string sql = "SELECT IDImage,imagePath,yamlPath from filedata";
	if (DBID.compare("")!=0)
	{
		sql = "SELECT IDImage,imagePath,yamlPath from filedata WHERE DBID =
"+ DBID;
	}
	q.get_result(sql);

	while (q.fetch_row())
	{
		dbInfo tmpAdd;
		tmpAdd.Id        = q.getval();
		tmpAdd.imagePath = q.getstr();
		tmpAdd.yamlPath  = q.getstr();
		vecResult.push_back(tmpAdd);
	}
	q.free_result();
	return vecResult;
}

void MWWTestCompute(Mat & sampleMat,Mat & compareMat,Mat &samplePopulaWeight,Mat
& comparePopulaWeight,int &MWWTestVaule,double & MWWTestWeightVaule)
{
	if (sampleMat.rows != compareMat.rows)
		return;

	int num_nodesMat         = sampleMat.rows + compareMat.rows;
	int num_EdgeCompleteMat  = (num_nodesMat*(num_nodesMat-1))/2;

	//first index of vertex
	//first index of vertex too
	typedef std::pair < int, int > EMat;

	//first index of all vertex
	//second type of each vertex
	//typedef std::pair < int, int > VMat;
	vector<EMat> allEdgesMat(num_EdgeCompleteMat);
```

```
	vector<int>   vertexInput(num_nodesMat);
	vector<Vec3f> vertexInputVec3f(num_nodesMat);
	vector<float> vertexInputWeightf(num_nodesMat);

	MatIterator_<Vec3f> sampleItf      = sampleMat.begin<Vec3f>();
	MatIterator_<Vec3f> sampleItf_end  = sampleMat.end<Vec3f>();
	MatIterator_<Vec3f> compareItf     = compareMat.begin<Vec3f>();
	MatIterator_<Vec3f> compareItf_end = compareMat.end<Vec3f>();

	MatIterator_<Vec<float, 1>> sampleWeightItf       =
samplePopulaWeight.begin<Vec<float, 1>>();
	MatIterator_<Vec<float, 1>> compareWeightItf      =
comparePopulaWeight.begin<Vec<float, 1>>();

	int countVertex = 0;

	for(; sampleItf != sampleItf_end; ++sampleItf,++sampleWeightItf) {
		vertexInput[countVertex]               = 0;
		vertexInputVec3f[countVertex]          = *sampleItf;
		vertexInputWeightf[countVertex]        = (*sampleWeightItf)[0];
		countVertex++;
	}
	for (; compareItf != compareItf_end; ++compareItf,++compareWeightItf){
		vertexInput[countVertex]               = 1;
		vertexInputVec3f[countVertex]          = *compareItf;
		vertexInputWeightf[countVertex]        = (*compareWeightItf)[0];
		countVertex++;
	}

	int countEdge   = 0;
	vector<double> weightsMat(num_EdgeCompleteMat);

	for(int i=0;i< num_nodesMat ;i++)
	{
		for(int j=i+1;j<num_nodesMat;j++)
		{
			allEdgesMat[countEdge].first  = i;
			allEdgesMat[countEdge].second = j;

			Vec3f tmpSrc  = vertexInputVec3f[i];
			Vec3f tmpDst  = vertexInputVec3f[j];

			double dim00 = (tmpSrc[0] - tmpDst[0]);
			double dim01 = (tmpSrc[1] - tmpDst[1]);
			double dim02 = (tmpSrc[2] - tmpDst[2]);

			weightsMat[countEdge] =
sqrt((dim00*dim00)+(dim01*dim01)+(dim02*dim02));
			countEdge++;
		}
	}
	typedef adjacency_list < vecS, vecS,
undirectedS,property<vertex_distance_t, double>, property < edge_weight_t,
double > > Graph;

	Graph g(num_nodesMat);
	property_map<Graph, edge_weight_t>::type weightmap = get(edge_weight, g);

	int sizeOfEdge = num_EdgeCompleteMat;

	for (std::size_t j = 0; j < sizeOfEdge ; ++j) {
		graph_traits<Graph>::edge_descriptor e;
		bool inserted;
```

```
            boost::tie(e, inserted) = add_edge(allEdgesMat[j].first,
allEdgesMat[j].second, g);
            weightmap[e] = weightsMat[j];
      }

      std::vector < graph_traits < Graph >::vertex_descriptor >
p(num_vertices(g));
      property_map<Graph, vertex_distance_t>::type distance =
get(vertex_distance, g);
      property_map<Graph, vertex_index_t>::type indexmap    = get(vertex_index,
g);
      prim_minimum_spanning_tree(g, *vertices(g).first, &p[0], distance,
weightmap, indexmap,default_dijkstra_visitor());

      int MWWValueInner = 0;
      for (std::size_t i = 0; i != p.size(); ++i)
            (p[i] != i) && (vertexInput[i] != vertexInput[p[i]]) ?
MWWValueInner++ : 0 ;

      vector<vector<int>> parentNode(num_nodesMat);
      for (std::size_t i = 0; i != p.size(); ++i)
            if (p[i] != i ){
                  parentNode[p[i]].push_back(i);
            }
      float MWWTestWeightInner = 0.0f;
      for (int i=0 ; i<num_nodesMat ;++i){
            // (term00 + term01) / term02
            // term00 = a0 + a1
            // term01 = b0 + b1
            // term02 = a0+a1+b0+b1
            int tmpComp = vertexInput[i];
            if (tmpComp == 0){
                  float term00 = vertexInputWeightf[i];
                  float term01 = 0.0f;
                  float term02 = vertexInputWeightf[i];
                  vector<int> tempInner = parentNode[i];
                  if (tempInner.size() > 0)
                  {
                        for (int j =0; j < tempInner.size();j++){
                              float tmpValue = vertexInputWeightf[tempInner[j]];
                              term02 += tmpValue;
                              if (vertexInput[tempInner[j]] == tmpComp){
                                    term00 += tmpValue;
                              }else{
                                    term01 += tmpValue;
                              }
                        }
                        if(term02>0.0f)
                              MWWTestWeightInner += (term00*term01)/term02;
                  }

            }
      }
      MWWTestVaule       = MWWValueInner ;
      MWWTestWeightVaule = MWWTestWeightInner;
      return;
}
```

7.3 โปรแกรม Web Page สำหรับเก็บภาพลงในฐานข้อมูลภาพ

7.3.1 DBCommand.php

```
<?php
session_start();
$info = pathinfo($_SERVER['REQUEST_URI']);
$path="http://".$_SERVER['SERVER_NAME'].$info['dirname'];
if (empty($info['extension']))
       $path.="/".$info['basename'];
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta  http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link  href="fileuploader.css" rel="stylesheet" type="text/css" media="all" />
<title>Untitled Document</title>
<style type="text/css">
      #DBSelect { font-family: AngsanaUPC;
                        font-size: 30px;
                        font-weight: bolder;
                        color: #00F;
                        background-color: #CCC;
                        text-align: center;
                        margin: 0px;
                        padding: 0px;
                      }
      #CreateDatabase {
                        font-family: AngsanaUPC;
                        font-size: 30px;
```

```
                font-weight: bolder;
                color: #00F;
                background-color: #CCC;
                text-align: center;
                margin: 0px;
                padding: 0px;
                border-top-style: solid;
                border-right-style: solid;
                border-bottom-style: solid;
                border-left-style: solid;
            }
#SelectDB {
                text-align: right;
                font-family: AngsanaUPC;
                font-size: 30px;
                font-weight: bolder;
            }
#Image #InputImage {
                font-family: AngsanaUPC;
                font-size: 30px;
                font-weight: bolder;
                color: #00F;
                background-color: #CCC;
                text-align: center;
                margin: 0px;
                padding: 0px;
            }
#apDiv1 {
                position:absolute;
                left:290px;
```

```
                        top:820px;
                        width:494px;
                        height:155px;
                        z-index:1;
                    }
      #JqPostForm fieldset legend {
                        text-align: left;
                    }
</style>
</head>

<body>
<label for="jumpmenu"></label>
<div>
<div id="DBSelect">
<form id="JqPostForm">
<fieldset>
<legend>เลือกฐานข้อมูล</legend>
<form id="form2" name="form2" method="post" action="" >
<select name="datafolder" id="datafolder" style="width:150px"
onchange="location.href='DBCommand.php?select='+this.value;">

<?php
@session_start();
session_unset();
mysql_connect("localhost","root","") or die(mysql_error());
mysql_select_db("TCBIR");
$query="SELECT DBID,Name FROM groupdatabase";
$result = mysql_query($query);
$checkValue = $_GET['select'];
while($nt=mysql_fetch_array($result))
```

```
{
	if ($checkValue == $nt[Name] || $checkValue =="" )
	{
		echo "<option selected value = $nt[Name]>$nt[Name]</option>";
		$checkValue = $nt[Name];
		$checkValueID = $nt[DBID];
	}else
	{
		echo "<option value = $nt[Name]>$nt[Name]</option>";
	}
}
?>
</select>
</form>
<fieldset>
<p>ป้อนภาพลงในฐานข้อมูล </p>

<div id="file-uploader">
<script src="fileuploader.js" language="javascript1.1"></script>
<script>
function createUploader()
{
	var uploader = new qq.FileUploader({
	//  pass the dom node (ex. $(selector)[0] for jQuery users)
	element: document.getElementById('file-uploader'),
	// path to server-side upload script. In our case server/php.php
	action: 'php.php',
	// additional data to send, name-value pairs
	params: {
		dataBaseSelect:  <?php echo "'"."$checkValue"."'"  ?>,
```

```
            dataBaseSelectID:  <?php echo "'"."$checkValueID"."'"  ?>
      }
      //debug: true
    });
}
window.onload = createUploader;
</script>
<script type="text/javascript">
var img_id=0
var image = new Array()
document.getElementById('send').onclick=function()
{
      img_id++
      var id="imgid"+img_id
      image = document.getElementById('file-uploader').value;
      document.getElementById('div').innerHTML="<img id='"+id+"' src='"+image+"'
width=500px height=200px>"
}
</script>
</div>
</fieldset>
</form>
<fieldset>
<div id="CreateDatabase">
<div>
<p>สร้างฐานข้อมูลใหม่ </p>
<form id="form1" name="form1" method="post" action="Newtable.php">
<p>
<label for="Namedatabase">สร้างฐานข้อมูล :   </label>
<input type="text" name="Namedatabase" id="Namedatabase" />
</p>
```

```
<p>
<label for="Detailofdatabase">ข้อมูลของฐานข้อมูล :</label>
<textarea name="Detailofdatabase" id="Detailofdatabase" cols="45"
rows="5"></textarea>
</p>
<p>
<input type="submit" name="submit" id="submit" value="ยืนยันการสร้าง" />
</p>
</form>
<p> </p>
</form>
</fieldset>
</body>
</html>
```

7.3.2 Php.php

```
<?php
require_once('Connections/conn.php');
session_start();
/**
* Handle file uploads via XMLHttpRequest
*/
class qqUploadedFileXhr {
/**
* Save the file to the specified path
* @return boolean TRUE on success
*/
function save($path) {
$input = fopen("php://input", "r");
$temp = tmpfile();
$realSize = stream_copy_to_stream($input, $temp);
```

```
fclose($input);
if ($realSize != $this->getSize()){
	return false;
}
$target = fopen($path, "w");
fseek($temp, 0, SEEK_SET);
stream_copy_to_stream($temp, $target);
fclose($target);
return true;
}

function getName() {
return $_GET['qqfile'];
}
function getSize() {
if (isset($_SERVER["CONTENT_LENGTH"])){
	return (int)$_SERVER["CONTENT_LENGTH"];
} else {
	throw new Exception('Getting content length is not supported.');
}
}
}

/**
* Handle file uploads via regular form post (uses the $_FILES array)
*/
class qqUploadedFileForm {
/**
* Save the file to the specified path
* @return boolean TRUE on success
```

```
*/
function save($path) {
if(!move_uploaded_file($_FILES['qqfile']['tmp_name'], $path)){
      return false;
}
return true;
}

function getName() {
return $_FILES['qqfile']['name'];
}

function getSize() {
return $_FILES['qqfile']['size'];
}
}

class qqFileUploader {
private $allowedExtensions = array();
private $sizeLimit = 10485760;
private $file;

function __construct(array $allowedExtensions = array(), $sizeLimit = 10485760){
$allowedExtensions = array_map("strtolower", $allowedExtensions);

$this->allowedExtensions = $allowedExtensions;
$this->sizeLimit = $sizeLimit;

$this->checkServerSettings();
```

```
if (isset($_GET['qqfile'])) {
//do this part
      $this->file = new qqUploadedFileXhr();
} elseif (isset($_FILES['qqfile'])) {
      $this->file = new qqUploadedFileForm();
} else {
      $this->file = false;
}
}

private function checkServerSettings(){
$postSize = $this->toBytes(ini_get('post_max_size'));
$uploadSize = $this->toBytes(ini_get('upload_max_filesize'));

if ($postSize < $this->sizeLimit || $uploadSize < $this->sizeLimit){
      $size = max(1, $this->sizeLimit / 1024 / 1024) . 'M';
      die("{'error':'increase    post_max_size    and    upload_max_filesize    to
$size'}");
}
}

private function toBytes($str){
$val = trim($str);
$last = strtolower($str[strlen($str)-1]);
switch($last) {
      case 'g': $val *= 1024;
      case 'm': $val *= 1024;
      case 'k': $val *= 1024;
}
return $val;
}
```

```
/**
* Returns array('success'=>true) or array('error'=>'error message')
*/
function handleUpload($uploadDirectory, $replaceOldFile = FALSE){
if (!is_writable($uploadDirectory)){
      return array('error' => "Server error. Upload directory isn't writable.");
}

if (!$this->file){
      return array('error' => 'No files were uploaded.');
}

$size = $this->file->getSize();

if ($size == 0) {
      return array('error' => 'File is empty');
}

if ($size > $this->sizeLimit) {
      return array('error' => 'File is too large');
}

$pathinfo = pathinfo($this->file->getName());

//filename
$filename = $pathinfo['filename'];
//echo $filename ;
//$filename = $pathinfo['filename'];
//$filename = md5(uniqid());
$ext = $pathinfo['extension'];
```

```
if($this->allowedExtensions && !in_array(strtolower($ext), $this->allowedExtensions)){
	$these = implode(', ', $this->allowedExtensions);
	return array('error' => 'File has an invalid extension, it should be one of '. $these . '.');
}

if(!$replaceOldFile){
	/// don't overwrite previous files that were uploaded
	while (file_exists($uploadDirectory . $filename . '.' . $ext)) {
		$filename .= rand(10, 99);
	}
}

try {
	/*** connect to SQLite database ***/
	$hostname_conn = "localhost";
	$database_conn = "ae";
	$username_conn = "root";
	$password_conn = "";
	$dbh = new PDO("mysql:host=$hostname_conn;dbname=$database_conn", $username_conn, $password_conn);
	$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
	$dbh->beginTransaction();

	$DBIDName = $_GET['dataBaseSelect'];
	$DBIDvalueID = $_GET['dataBaseSelectID'];
	$fullFileNameWithPath = $uploadDirectory . $filename . '.' . $ext;

	$yamlFilePath = "yaml/". $filename . '.yaml';
```

```
	$count = $dbh->exec("INSERT INTO filedata(DBID,imagePath,yamlPath) VALUES('$DBIDvalueID','$fullFileNameWithPath','$yamlFilePath')");
if ($this->file->save($uploadDirectory . $filename . '.' . $ext)){
	$return_var = 0;
	$out = array();
	$exec = exec("VqKmeanCalculate.exe ".$fullFileNameWithPath." ".$yamlFilePath . " 60 30",$out,$return_var);

	if($return_var < 0)
	{
		unlink($fullFileNameWithPath);
		$dbh->rollback();
		$dbh = null;
		return array('error'=> 'Could not produce yaml file.');
	}
	$resultBool = $dbh->commit();
	$dbh = null;
	return array('success'=>true);
} else {
	$dbh->rollback();
	$dbh = null;
	return array('error'=> 'Could not save uploaded file.' .
		'The upload was cancelled, or server error encountered');
}
}
catch(PDOException $e)
{
	$dbh->rollback();
	$dbh = null;
}
}
```

```
}

$allowedExtensions = array();
// max file size in bytes
$sizeLimit = 10 * 1024 * 1024;

$uploader = new qqFileUploader($allowedExtensions, $sizeLimit);

$DBIDName = $_GET['dataBaseSelect'];
$Class2 =  "Image/allMix/";
$result = $uploader->handleUpload($Class2);
echo htmlspecialchars(json_encode($result), ENT_NOQUOTES);
?>
```

7.4 โปรแกรม Web Page สำหรับค้นหาภาพในฐานข้อมูล

7.4.1 SearchInner.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>class.upload.php test forms</title>
<style>
fieldset {
      width: 50%;
      margin: 15px 0px 25px 0px;
      padding: 15px;
}
legend {
      font-weight: bold;
}
.button {
```

```
	text-align: right;
}
.button input {
	font-weight: bold;
}

body {
	background-color: #F96;
}
</style>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<h1>Color Image Retrieval System</h1>
<fieldset>
<legend>Image sample</legend>
<p>Pick up image to upload and press upload </p>
<form	name="form2"	enctype="multipart/form-data"	method="post"
action="upload.php" />
<p><input type="file" size="32" name="my_field" value="" /></p>
<p class="button"><input type="hidden" name="action" value="image" />
<input type="submit" name="Submit" value="upload to query" /></p>
</form>
<p> </p>
<p> </p>
</fieldset>
</body>
</html>
```

7.4.2 Retrieval.php

```
<?php
session_start();
error_reporting(E_ALL);

// recieve namedata form select to query each folder
$DBIDValue = "null" ;
// we first include the upload class, as we will need it here to deal with the uploaded file
include('class.upload.php');
// retrieve eventual CLI parameters
$cli = (isset($argc) && $argc > 1);
if ($cli) {
      if (isset($argv[1])) $_GET['file'] = $argv[1];
      if (isset($argv[2])) $_GET['dir'] = $argv[2];
      if (isset($argv[3])) $_GET['pics'] = $argv[3];
}

// set variables
$dir_dest = (isset($_GET['dir']) ? $_GET['dir'] : 'sampleImage');
$dir_pics = (isset($_GET['pics']) ? $_GET['pics'] : $dir_dest);

if (!$cli) {
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<meta http-equiv=Content-Type content="text/html; charset=utf-8">
<head>
```

```
<title>Color Image Retrieval System</title>
<style>
body {
}
fieldset {
	width: 50%;
	background: url(bg.gif);
	margin: 15px 0px 25px 0px;
	padding: 15px;
}
legend {
	font-weight: bold;
}
fieldset img {
	float: right;
}
fieldset p {
	font-size: 70%;
	font-style: italic;
}
.button {
	text-align: right;
}
.button input {
	font-weight: bold;
}
</style>
</head>

<body bgcolor="#FF6666">
```

```
<h1>Color Image Retrieval System :: viewing images retrieve</h1>

<?php

}

if ((isset($_POST['action']) ? $_POST['action'] : (isset($_GET['action']) ? $_GET['action'] : '')) == 'image') {

// ---------- IMAGE UPLOAD ----------

// we create an instance of the class, giving as argument the PHP object

// corresponding to the file field from the form

// All the uploads are accessible from the PHP object $_FILES

$handle = new Upload($_FILES['my_field']);

// then we check if the file has been uploaded properly

// in its *temporary* location in the server (often, it is /tmp)

if ($handle->uploaded) {

// yes, the file is on the server

// below are some example settings which can be used if the uploaded file is an image.

// now, we start the upload 'process'. That is, to copy the uploaded file

// from its temporary location to the wanted location

// It could be something like $handle->Process('/home/www/my_uploads/');

$handle->Process($dir_dest);

// we check if everything went OK

if ($handle->processed) {

      // everything was fine !

      echo '<fieldset>';

      echo '  <legend>sample file uploaded with success</legend>';

      echo '  <img src="'.$dir_pics.'/' . $handle->file_dst_name . '" />';

$info = getimagesize($handle->file_dst_pathname);

      echo '    link to the file : <a href="'.$dir_pics.'/' . $handle->file_dst_name . '">' . $handle->file_dst_name . '</a><br/>';

      echo '</fieldset>';

} else {
```

```
	// one error occured
	echo '<fieldset>';
	echo '  <legend>file not uploaded to the wanted location</legend>';
	echo '  Error: ' . $handle->error . '';
	echo '</fieldset>';
}

// we delete the temporary files
$handle-> Clean();

$return_var = 0;
$out = array();
$exec = exec("VqRetrieval.exe ".$dir_pics.'/' . $handle->file_dst_name . " 60 30 1 127.0.0.1 root null ae $DBIDValue",$out,$return_var);
if($return_var > 0)
	foreach ($out as $line) {  // process array line by line
		$pieces = explode(";", $line);
		$fileSplitPath = explode("/", $pieces[1]);
		$countData =  count($fileSplitPath);
		echo '<fieldset>';
		 echo '  <legend>Retrieve image score := '.$pieces[0].'</legend>';
		 echo '  <img src="'. $pieces[1] . '" />';
		 $info = getimagesize($handle->file_dst_pathname);
		 echo '    link to the file : <a href="'.$pieces[1] . '">' . $fileSplitPath[$countData-1] . '</a><br/>';
		echo '</fieldset>';
	}
} else {
	echo '<fieldset>';
	echo '  <legend>file not uploaded on the server</legend>';
	echo '  Error: ' . $handle->error . '';
```

```
      echo '</fieldset>';
}
}

if (!$cli) {
      echo '<p><a href="Seachpage.html">do another test</a></p>';
?>
</body>
</html>
<?php
}
?>
```

เอกสารอ้างอิง

[1] S.K. Chang and A. Hsu, “Image Information Systems: Where do we go from here?," IEEE Trans. Knowl. Data Eng., vol. 4, no. 5, pp. 431-442, Oct. 1992.

[2] H.T. Shen, B.C. Ooi, and K.L. Tan, “Giving Meanings to WWW Images," in Proc. ACM Int. Multimedia Conf., 2000, pp. 39-48.

[3] H. Tamura and N. Yokoya, “Image Database Systems: A Survey," Pattern Recognit., vol. 17, no. 1, pp. 29-43, 1984.

[4] R. Jain, Proc. US NSF Workshop Visual Information Management Systems, 1992.

[5] A. E. Cawkill, “The British Library's Picture Research Projects: Image, Word, and Retrieval," Advanced Imaging,Vol.8, No.10, pp.38-40, October 1993.

[6] J. Dowe, “Content-based retrieval in multimedia imaging," In Proc. SPIE Storage and Retrieval for Image and Video Database, 1993.

[7] C. Faloutsos et al, “E_cient and e_ective querying by image content," Journal of intelligent information systems, Vol.3, pp.231-262, 1994.

[8] Y. Gong, H. J. Zhang, and T. C. Chua, “An image database system with content capturing and fast image indexing abilities", Proc. IEEE International Conference on Multimedia Computing and Systems, Boston, pp.121-130, 14-19 May 1994.

[9] H. J. Zhang, and D. Zhong, “A Scheme for visual feature-based image indexing," Proc. of SPIE conf. on Storage and Retrieval for Image and Video Databases III, pp. 36-46, San Jose, Feb. 1995.

[10] B. Furht, S. W. Smoliar, and H.J. Zhang, “Video and Image Processing in Multimedia Systems," Kluwer Academic Publishers, 1995.

[11] Y. Rui, T. S. Huang, and S. F. Chang, “Image retrieval: current techniques, promising directions and open issues," Journal of Visual Communication and Image Representation, Vol.10, pp. 39-62, 1999.

[12] A.W.M. Smeulders, M.Worring, S. Santini, A. Gupta, and R. Jain : “Content-based Image Retrieval at the End of the Early Years,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol.22, no.12, pp.1349-1380, Dec. 2000.

[13] A. Smeulders, T. Gevers, J. M. Geusebroek, and M. Worring, “Invariance in Content-Based Retrieval,”

[14] H. Burkhardt, and S. Siggelkow, “Invariant features for discriminating between equivalence classes," Nonlinear Model-based Image Video Processing and Analysis, John Wiley and Sons, 2000.

[15] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, Computer graphics: principles and practice, 2nd ed., Reading, Mass, Addison-Wesley, 1990.

[16] J. Huang, S.R. Kumar, M. Metra, W. J. Zhu, and R. Zabith, “Spatial color indexing and applications," Intl J. Computer Vision, Vol.35, No.3, pp. 245-268, 1999.

[17] J. Huang, et al., “Image indexing using color correlogram," IEEE Int. Conf. on Computer Vision and Pattern Recognition, pp. 762-768, Puerto Rico, June 1997.

[18] M. Ioka, “A method of defining the similarity of images on the basis of color information," Technical Report RT-0030, IBM Tokyo Research Laboratory, Tokyo, Japan, Nov. 1989.

[19] A. K. Jain, Fundamental of Digital Image Processing, Englewood Cliffs, Prentice Hall, 1989.

[20] E. Mathias, “Comparing the influence of color spaces and metrics in content-based image retrieval," Proceedings of International Symposium on Computer Graphics, Image Processing, and Vision, pp. 371 -378, 1998.

[21] G.Pass, and R. Zabith, “Comparing images using joint histograms," Multimedia Systems, Vol.7, pp.234-240, 1999.

[22] M. Stricker, and M. Orengo, “Similarity of color images," SPIE Storage and Retrieval for Image and Video Databases III, vol. 2185, pp.381-392, Feb. 1995.

[23] M.J. Swain and D.H. Ballard : “Color Indexing," Int'l J. Computer Vision, Vol.7, No.1, pp. 11-32 (1991)

[24] H. J. Zhang, et al, “Image retrieval based on color features: An evaluation study," SPIE Conf. on Digital Storage and Archival, Pennsylvania, Oct. 25-27, 1995.
[25] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, “Query by image and video content: The QBIC system." IEEE Computer, Vol.28, No.9, pp. 23-32, Sept. 1995.
[26] W. Niblack et al., “Querying images by content, using color, texture, and shape," SPIE Conference on Storage and Retrieval for Image and Video Database, Vol. 1908, pp.173-187, April 1993.
[27] G. Pass, and R. Zabith, “Histogram re_nement for content-based image retrieval," IEEE Workshop on Applications of Computer Vision, pp. 96-102, 1996.
[28] T. Gevers, and A.W.M.Smeulders, “Pictoseek: Combining color and shape invariant features for image retrieval," IEEE Trans. on image processing, Vol.9, No.1, pp102-119, 2000.
[29] G. D. Finlayson, “Color in perspective," IEEE Trans on Pattern Analysis and Machine Intelligence, Vol.8, No. 10, pp.1034-1038, Oct. 1996.
[30] T. Gevers, and A. W. M. Smeulders, “Content-based image retrieval by viewpointinvariant image indexing," Image and Vision Computing, Vol.17, No.7, pp.475-488, 1999.
[31] P. Brodatz, “Textures: A photographic album for artists & designers," Dover, NY, 1966.
[32] T. Chang, and C.C.J. Kuo, “Texture analysis and classi_cation with tree-structured wavelet transform," IEEE Trans. on Image Processing, vol. 2, no. 4, pp. 429-441, October 1993.
[33] I. Daubechies, “The wavelet transform, time-frequency localization and signal analysis," IEEE Trans. on Information Theory, Vol. 36, pp. 961-1005, Sept. 1990.
[34] J. M. Francos. “Orthogonal decompositions of 2D random _elds and their applications in 2D spectral estimation," N. K. Bose and C. R. Rao, editors, Signal Processing and its Application, pp.20-227. North Holland, 1993.
[35] J. M. Francos, A. A. Meiri, and B. Porat, “A uni_ed texture model based on a 2d Wold like decomposition," IEEE Trans on Signal Processing, pp.2665-2678, Aug. 1993.
[36] J. M. Francos, A. Narasimhan, and J. W. Woods, “Maximum likelihood parameter estimation of textures using a Wold-decomposition based model," IEEE Trans. on Image Processing, pp.1655-1666, Dec.1995.
[37] A. K. Jain, and F. Farroknia, “Unsupervised texture segmentation using Gabor filters," Pattern Recognition, Vo.24, No.12, pp. 1167-1186, 1991.
[38] A. Kankanhalli, H. J. Zhang, and C. Y. Low, “Using texture for image retrieval," Third Int. Conf. on Automation, Robotics and Computer Vision, pp. 935-939, Singapore, Nov. 1994.
[39] W. J. Krzanowski, Recent Advances in Descriptive Multivariate Analysis, Chapter 2, Oxford science publications, 1995.
[40] A. Laine, and J. Fan, “Texture classification by wavelet packet signatures," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 15, No. 11, pp. 1186-1191, Nov. 1993.
[41] F. Liu, and R. W. Picard, “Periodicity, directionality, and randomness: Wold features for image modeling and retrieval," IEEE Trans. on Pattern Analysis and Machine Learning, Vol. 18, No. 7, July 1996.
[42] W. Y. Ma, and B. S. Manjunath, “A comparison of wavelet features for texture annotation," Proc. of IEEE Int. Conf. on Image Processing, Vol. II, pp. 256-259, Washington D.C., Oct. 1995.
[43] S. G. Mallat, “A theory for multiresolution signal decomposition: the wavelet representation," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 11, pp. 674-693, July 1989.
[44] B. S. Manjunath, and W. Y. Ma, “Texture features for browsing and retrieval of image data," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 18, No. 8, pp. 837-842, Aug. 1996.
[45] J. Mao, and A. K. Jain, \Texture classi_cation and segmentation using multiresolution simultaneous autoregressive models," Pattern Recognition, Vol. 25, No. 2, pp. 173-188, 1992.
[46] T. Ojala, M. Pietikainen, and D. Harwood, \A comparative study of texture measures with classification based feature distributions," Pattern Recognition, Vol.29, No.1, pp.51-59, 1996.

[47] R. W. Picard, T. Kabir, and F. Liu, “Real-time recognition with the entire Brodatz texture database," Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition, pp. 638-639, New York, June 1993.
[48] H. Tamura, S. Mori, and T. Yamawaki, “Texture features corresponding to visual perception," IEEE Trans. On Systems, Man, and Cybernetics, vol. Smc-8, No. 6, June 1978.
[49] H. Voorhees, and T. Poggio, “Computing texture boundaries from images," Nature, 333:364-367, 1988.
[50] A. Pentland, R.W. Picard and S. Sclar, “Photobook: Content-Based Manipulation of Image Databases," Proc. Storage and Retrieval for Image and Video Databases II, Vol. 2185, San Jose, CA, USA February, 1994.
[51] J. G. Daugman, “Complete discrete 2D Gabor transforms by neural networks for image analysis and compression," IEEE Trans. ASSP, vol. 36, pp. 1169-1179, July 1998.
[52] J. E. Gary, and R. Mehrotra, “Shape similarity-based retrieval in image database systems," Proc. of SPIE, Image Storage and Retrieval Systems, Vol. 1662, pp. 2-8, 1992.
[53] W. I. Grosky, and R. Mehrotra, “Index based object recognition in pictorial data management," CVGIP, Vol. 52, No. 3, pp. 416-436, 1990.
[54] H. V. Jagadish, “A retrieval technique for similar shapes," Proc. of Int. Conf. on Management of Data, SIGMOID91, Denver, CO, pp. 208-217, May 1991.
[55] D. Tegolo, “Shape analysis for image retrieval," Proc. of SPIE, Storage and Retrieval for Image and Video Databases -II, no. 2185, San Jose, CA, pp. 59-69, February 1994.
[56] E. M. Arkin, L.P. Chew, D..P. Huttenlocher, K. Kedem, and J.S.B. Mitchell, “An efficiently computable metric for comparing polygonal shapes," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 13, no. 3, pp. 209-226, 1991.
[57] S. Sclaro, and A. Pentland, “Modal matching for correspondence and recognition," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 17, No. 6, pp. 545-561, June 1995.
[58] K. Arbter, W. E. Snyder, H. Burkhardt, and G. Hirzinger, “Application of affine invariant Fourier descriptors to recognition of 3D objects," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 12, pp. 640-647, 1990.
[59] H. Kauppinen, T. Seppnen, and M. Pietikinen, “An experimental comparison of autoregressive and Fourier-based descriptors in 2D shape classification," IEEE Trans. Pattern Anal. and Machine Intell., Vol. 17, No. 2, pp. 201-207, 1995.
[60] E. Persoon, and K. Fu, “Shape discrimination using Fourier descriptors," IEEE Trans. Syst., Man, and Cybern., Vol. 7, pp. 170-179, 1977.
[61] M. K. Hu, “Visual pattern recognition by moment invariants," in J. K. Aggarwal, R. O. Duda, and A. Rosenfeld, Computer Methods in Image Analysis, IEEE computer Society, Los Angeles, CA, 1977.
[62] L. Yang, and F. Algregtsen, “Fast computation of invariant geometric moments: A new method giving correct results," Proc. IEEE Int. Conf. on Image Processing, 1994.
[63] R. C. Veltkamp, and M. Hagedoorn, “State-of-the-art in shape matching," Technical Report UU-CS-1999-27, Utrecht University, Department of Computer Science, Sept. 1999.
[64] S. K. Chang, Q. Y. Shi, and C. Y. Yan, “Iconic indexing by 2-D strings," IEEE Trans. on Pattern Anal. Machine Intell., Vol.9, No.3, pp. 413-428, May 1987.
[65] S. K. Chang, E. Jungert, and Y. Li, “Representation and retrieval of symbolic pictures using generalized 2D string", Technical Report, University of Pittsburgh, 1988.
[66] S. Y. Lee, and F. H. Hsu, “2D C-string: a new spatial knowledge representation for image database systems," Pattern Recognition, Vol. 23, pp 1077-1087, 1990.
[67] S. Y. Lee, M.C. Yang, and J. W. Chen, “2D B-string: a spatial knowledge representation for image database system," Proc. ICSC'92 Second Int. computer Sci. Conf., pp.609-615, 1992.
[68] H. Samet, “The quadtree and related hierarchical data structures," ACM Computing Surveys, Vol.16, No.2, pp.187-260, 1984.
[69] V. N. Gudivada, and V. V. Raghavan, “Design and evaluation of algorithms for image retrieval by spatial similarity," ACM Trans. on Information Systems, Vol. 13, No. 2, pp. 115-144, April 1995.

[70] M. Stricker, and M. Orengo, “Color indexing with weak spatial constraint," Proc. SPIE Conf. On Visual Communications, 1996.
[71] F. Guo, J. Jin, and D. Feng, “Measuring image similarity using the geometrical distribution of image contents", Proc. of ICSP, pp.1108-1112, 1998.
[72] H. Wang, F. Guo, D. Feng, and J. Jin, “A signature for content-based image retrieval using a geometrical transform," Proc. Of ACM MM'98, Bristol, UK, 1998.
[73] Y. Rui, T.S.Huang, and S. Mehrotra, “Content-based image retrieval with relevance feedback in MARS," Proceedings of International Conference on Image Processing, Vol.2, pp. 815 - 818, 1997.
[74] W. Y. Ma, and B. S. Manjunath, “Edge ow: a framework of boundary detection and image segmentation," IEEE Int. Conf. on Computer Vision and Pattern Recognition, pp. 744-749, Puerto Rico, June 1997.
[75] W. Y. Ma, and B. S. Manjunath, “Netra: A toolbox for navigating large image databases," Multimedia Systems, Vol.7, No.3, pp.:184-198, 1999.
[76] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik, “Blobworld: A system for region-based image indexing and retrieval," In D. P. Huijsmans and A. W. M. Smeulders, ed. Visual Information and Information System, Proceedings of the Third International Conference VISUAL99, Amsterdam, The Netherlands, June 1999, Lecture Notes in Computer Science 1614. Springer, 1999.
[77] J. Hafner, H.S. Sawhney, W. Equitz, M. Flickner, and W. Niblack : “Efficient Color Histogram Indexing for Quadratic Form Distance Functions", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.17, No. 7, pp.729-736 (July 1995)
[78] T. P. Minka, and R. W. Picard, “Interactive learning using a 'society of models', "IEEE Int. Conf. on Computer Vision and Pattern Recognition, pp. 447-452, 1996.
[79] J.A. Catalan, and J.S. Jin, “Dimension reduction of texture features for image retrieval using hybrid associative neural networks," IEEE International Conference on Multimedia and Expo, Vol.2, pp. 1211 -1214, 2000.
[80] N. Beckmann, et al, “The R*-tree: An efficient robust access method for points and rectangles," ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, May 1990.
[81] J. Vendrig, M. Worring, and A. W. M. Smeulders, “Filter image browsing: exploiting interaction in retrieval," Proc. Viusl'99: Information and Information System, 1999.
[82] J. T. Robinson, “The k-d-B-tree: a search structure for large multidimensional dynamic indexes," Proc. of SIGMOD Conference, Ann Arbor, April 1981.
[83] J. Nievergelt, H. Hinterberger, and K. C. Sevcik, “The grid file: an adaptable symmetric multikey file structure," ACM Trans. on Database Systems, pp. 38-71, March 1984.
[84] A. Vailaya, M. A. G. Figueiredo, A. K. Jain, and H. J. Zhang, “Image classification for content-based indexing," IEEE Trans. on Image Processing, Vol.10, No.1, Jan. 2001.
[85] J. Assfalg, A. D. Bimbo, and P. Pala, “Using multiple examples for content-based retrieval," Proc. Int'l Conf. Multimedia and Expo, 2000.
[86] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, “Relevance feedback: a power tool for interactive content-based image retrieval," IEEE Trans. on Circuits and Systems for Video Technology, 1998.
[87] Y. Rui, et al, “A relevance feedback architecture in content-based multimedia information retrieval systems," Proc of IEEE Workshop on Content-based Access of Image and Video Libraries, 1997.
[88] J. Huang, S. R. Kumar, and M. Metra, “Combining supervised learning with color correlograms for content-based image retrieval," Proc. of ACM Multimedia95, pp. 325-334, Nov. 1997.
[89] R.W. Picard and T. P. Minka, “Vision texture for annotation," Multimedia Systems: Special Issue on Contentbased Retrieval, vol. 3, no. 1, pp. 3-14, 1995.
[90] T. V. Papathomas, T. E. Conway, I. J. Cox, J. Ghosn, M. L. Miller, T. P. Minka, and P. N. Yianilos, “Psychophysical studies of the performance of an image database retrieval system," IS&T/SPIE Conf. on Human Vision and Electronic Imaging III, 1998.

[91] B. E. Rogowitz, T. Frese, J. Smith, C. A. Bouman, and E. Kalin, “Percpetual image similarity experiments," IS&T/SPIE Conf. on Human Vision and Electronic Imageing III, 1998.
[92] I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos, “Pichunter: Bayesian relevance feedback for image retrieval," in Intl. Conf. on Pattern Recognition, vol. 3, pp. 361-369, 25-29 Aug., 1996
[93] B. S. Manjunath and W. Y. Ma, “Image indexing using a texture dictionary," in Proceedings of SPIE Conference on Image Storage and Archiving System, Philadelphia, Vol. 2606, pp. 288-296, Oct., 1985
[94] W. Y. Ma and B. S. Manjunath, “Texture features and learning similarity," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 425-430, 1996.
[95] Y. Rui, T. S. Huang, and S. Mehrotra, “Relevance Feedback Techniques in Interactive Content-Based Image retrieval," in Proc. of IS&T SPIE Storage and Retrieval of Images/Video Databases VI, EI'98, 1998.
[96] D. Daneels, D. Campenhout, W. Niblack, W. Equitz, R. Barber, E. Bellon, and F. Fierens, “Interactive outlining: An improved approach using active contours,"' in Proc. SPIE Storage and Retrieval for Image and Video Databases, 1993.
[97] J. R. Smith and S.-F. Chang, “An image and video search engine for the world-wide web," in Proc. SPIE Storage and Retrieval for Image and Video Databases, 1997.
[98] Y. Rui, T. S. Huang, S. Mehrotra, and M. Ortega, Automatic matching tool selection using relevance feedback in MARS, in Proc. of 2nd Int. Conf. on Visual Information Systems, 1997.
[99] J. R. Smith and S.-F. Chang, “Visualseek: A fully automated content-based image query system," in Proc. ACM Multimedia 96, 1996.
[100] R. Jain, “Workshop report: NSF workshop on visual information management systems," in Proc. SPIE Storage and Retrieval for Image and Video Databases, 1993.
[101] R. Jain, A. Pentland, and D. Petkovic, “in NSF-ARPA Workshop on Visual Information Management Systems," Cambridge, MA, June 1995.
[102] B. Cheng, “Approaches to image retrieval based on compressed data for multimedia database systems," Ph.D. thesis, University of New York at Bu_alo, 1996.
[103] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and T. S. Huang, Supporting similarity queries in MARS, in Proc. of ACM Conf. on Multimedia, 1997.
[104] R. Fagin and E. L. Wimmers, “Incorporating user preferences in multimedia queries," in Proc. of Int. Conf. on Database Theory, 1997.
[105] J. P. Callan, W. B. Croft, and S. M. Harding, “The inquery retrieval system," in Proc. of 3rd Int. Conf. on Database and Expert System Application, Sept. 1992.
[106] G. Salton and M. J. McGill, “Introduction to Modern Information Retrieval," McGraw-Hill, New York, 1983.
[107] G. Salton and C. Buckley, “Term-weighting, approaches in automatic text retrieval," Information Processing and Management, 1988.
[108] W. M. Shaw, “Term-relevance computations and perfect retrieval performance," Information Processing and Management, vol. 31, issue 4, pp. 491-498, Jul. 1995.
[109] C. Buckley and G. Salton, “Optimization of relevance feedback weights," in Proc. of SIGIR95, 1995.
[110] J. Allan, “Relevance feedback with too much data," in Proc. of SIGIR95, 1995.
[111] S. Mehrotra, Y. Rui, O.-B. Michael, and T. S. Huang, “Supporting content-based queries over images in MARS," in Proc. of IEEE Int. Conf. on Multimedia Computing and Systems, 1997.
[112] R. K. Srihari, “Automatic indexing and content-based retrieval of captioned images," IEEE Computer Magzine 28(9), 1995.
[113] J. R. Smith and S.-F. Chang, “Multi-stage classi_cation of images from features and related text," in 4th Europe EDLOS Workshop, San Miniato, Italy, Aug. 1997.
[114] J. R. Smith and S.-F. Chang, “Enhancing image search engines in visual information environments," in IEEE 1st Multimedia Signal Processing Workshop, June 1997.

[115] I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos, “Target testing and the pichunter bayesian multimedia retrieval system," in Advanced Digital Libraries Forum, Washington, DC, May.
[116] I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos, Pichunter: Bayesian relevance feedback for image retrieval, in Intl. Conf. on Pattern Recognition.
[117] I. J. Cox, M. L. Miller, T. P. Minka, and P. N. Yianilos, An optimized interaction strategy for bayesian relevance feedback, in IEEE Conf. CVPR, 1998.
[118] M. Stricker and M. Swain : “The capacity of Color Histogram Indexing", Proc. Computer Vision and Pattern Recognition, pp. 704-708 (1994)
[119] C. Theoharatos, N.A. Laskaris, G. Economou, and S. Fotopoulos : “A Generic Scheme for Color Image Retrieval Based on the Multivariate Wald-Wolfowitz Test", IEEE Trans. on Knowledge and Data Engineering, Vol.17, No. 6, pp.808-819 (June, 2005)
[120] L. Mandic, S. Grgic, and M. Grgic : “Comparison of Color Difference Equations", 48th International Symposium ELMAR-2006, pp.107-110 (Jun., 07-09, 2006)
[121] R. C. Prim : “Shortest connection networks and some generalizations", Bell Sys. Tech. J., pp.1389-1401 (1957)
[122] J.H. Friedman and L.C. Rafsky : “Multivariate Generalizations of the Wolfowitz and Smirnov two-sample tests", Annals of Statistics, Vol.7, No.4, pp. 697-717 (July 1979)
[123] A. Wald and J. Wolfowitz : “On a Test Whether Two Samples Are from the Same Population", Ann. Math. Statist., Vol.11, pp.147-162 (1940)
[124] “Recommendations on Uniform Color Spaces, Color Difference Equations, Psychometric Color Terms", C.I.E. Supplement No. 2 to CIE publication No. 15(E-131) 1971/(TC-1.3), 1978. References 87
[125] Y. Linde, A. Buzo, and R. M. Gray : “An Algorithm for Vector Quantizer Design", IEEE Trans. Commun., Vol.COM-28, pp.84-95 (1980)
[126] Corel Gallery 1 million, Corel Corp., Ontario, Canada.
[127] A. Abdullah and M. A. Wiering : “CIREC : Clustering Correlogram Image Retrieval and Categorization using MPEG-7 Descriptors", Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing, pp. 431-437 (2007)
[128] S.P. Smith and A.K. Jain, “A Test to Determine the Multivariate Normality of a Data Set," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.10, No.5, pp. 757-761, (Sept. 1988)
[129] Y. Rubner, J. Puzicha, C. Tomasi, and J.M. Buhmann, “Empirical Evaluation of Dissimilarity Measures for Color and Texture," Computer Vision and Image Understanding, vol. 84, pp. 25-43, 2001.
[130] http://www.cie.co.at/main/freepubs.html
[131] S. Na and D. L. Neuho, “Bennett's Integral for Vector Quantizers," IEEE Trans. Information Theory, vol. 41, no. 4, July 1995.
[132] W.D. Wright, “The Historical and Experimental Background to the 1931 CIE System of Colorimetry," Golden Jubilee of Colour in the CIE, Bradford 1981.
[133] W.R. Bennett, “Spectra of Quantized Signals," Bell Syst. Tech. J., vol. 27, pp. 446-472, July 1948.
[134] A. Gersho, “Asymptotically optimal block quantization," IEEE Trans. Inform. Theory, vol. IT-25, pp. 373-380, July 1979.