การประยุกต์ใช้กระบวนการ Gaussian Mixture Model ในการสืบค้นสารสนเทศประเภทภาพดิจิตอล ผ่านทางระบบอินเตอร์เนต

Implementation of Gaussian Mixture Model Using Server-Based Application for Image Retrieval on the Internet

นายทวี คุณบิดา รหัส 42360511

นายวัฒนพงศ์ เตียมแสง รหัส 43360551

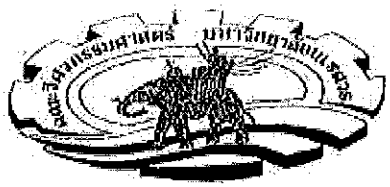
นายเฉลิมขวัญ ลาสอน รหัส 43360718

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2546



ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	การประยุกต์ใช้กระบวนการ Gaussian Mixture Model วินิจฉัยเนื้อหาเพื่อการสืบค้นสารสนเทศประเภทภาพดิจิตอล ผ่านทางระบบอินเตอร์เนต		
ผู้ดำเนินโครงงาน	นายทวี	คุณบิดา	รหัส 42360511
	นายวัฒนพงศ์	เตียมแสง	รหัส 43360551
	นายเฉลิมขวัญ	ลาสอน	รหัส 43360718
อาจารย์ที่ปรึกษา	ดร.ไพศาล	มุณีสว่าง	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2546		

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ดร.ไพศาล มุณีสว่าง)

.....กรรมการ

(ดร.สุรเชษฐ์ กานต์ประชา)

.....กรรมการ

(ดร.สุชาติ แย้มเม่น)

หัวข้อโครงงาน	การประยุกต์ใช้กระบวนการ Gaussian Mixture Model วินิจฉัยเนื้อหาเพื่อการ สืบค้นสารสนเทศประเภทภาพดิจิตอลผ่านทางระบบอินเตอร์-เนต		
ผู้ดำเนินโครงงาน	นายทวี	คุณบิดา	รหัส 42360511
	นายวัฒนพงศ์	เตียมแสง	รหัส 43360551
	นายเฉลิมขวัญ	ลาสอน	รหัส 43360718
อาจารย์ที่ปรึกษา	ดร.ไพศาล มุณีสว่าง		
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2546		

บทคัดย่อ

โครงงานนี้เป็นการศึกษาและพัฒนาโปรแกรมเพื่อใช้ในการสืบค้นหาสารสนเทศประเภทภาพดิจิตอลผ่านระบบอินเตอร์เนต โดยการประยุกต์ใช้ Gaussian mixture model (GMM) ในการพัฒนาร่วมกับเครื่องมือ ในการเขียนโปรแกรมได้แก่ โปรแกรม Java server page (JSP) เพื่อพัฒนาเว็บแอพลิเคชัน โปรแกรมฐานข้อมูล Cloudscape เพื่อใช้เป็นฐานข้อมูล และเซิร์ฟเวอร์ของ Java 2 Enterprise Edition (J2EE) โปแกรมที่สร้างขึ้นสามารถหาข้อมูลภาพโดยใช้วิธี Content-based image retrieval (CBIR) ซึ่งอาศัยคุณสมบัติพื้นฐานของภาพ (Low-level feature) ในการประมวลผล ส่วนสำคัญที่ใช้เพิ่มประสิทธิภาพในการสืบค้นหาภาพคือ กระบวนการป้อนกลับจากผู้ใช้ (Relevance feedback) ซึ่งเป็นการติดต่อกันระหว่างผู้ใช้กับคอมพิวเตอร์ (หรือ search engine) ผลลัพธ์ของการสืบค้นโดยใช้กระบวนการป้อนกลับนี้ค่อนข้างมีประสิทธิภาพมากกว่าการค้นหาแบบง่าย (Simple search method) เนื่องจากมีการปรับปรุงผลลัพธ์โดยผู้ใช้เอง ดังนั้นจึงได้ภาพที่ใกล้เคียงกับความต้องการของผู้ใช้มากที่สุด แต่ทั้งนี้โครงงานนี้ได้รวมเอาการค้นหาทั้งสองแบบเข้าไว้ด้วยกันเพื่อเพิ่มประสิทธิภาพการทำงานให้มากขึ้น

จากผลการทดลองกับฐานข้อมูลภาพดิจิตอลจำนวน 34,000 ภาพ พบว่ากระบวนการป้อนกลับ GMM มีประสิทธิภาพมาก และให้ผลลัพธ์ที่พึงพอใจแก่ผู้ใช้ได้ดีกว่าการหาแบบ Simple search method

Project Title	Implementation of Gaussian Mixture Model Using Server-Based Application for Image Retrieval on the Internet.		
Name	Mr.Tavee	Khunbida	ID.42360511
	Mr.Wattanapong	Thiamsang	ID.43360551
	Mr.Chalermkwan	Lasorn	ID.43360718
Project Advisor	Dr.Paisarn Muneesawang		
Major	Computer Engineering		
Department	Electrical and Computer Engineering		
Academic Year	2003		

.....

ABSTRACT

This project studies and implements Gaussian Mixture Model (GMM) algorithm using server-based application for image retrieval on the Internet. The proposed search engine uses query by example and relevance feedback (RF) methods to improve its retrieval performance. The search engine allows user to feedback on search results to allow a computer to know the user's needs, so that the precision results obtained arr significantly improved from the user viewpoints.

We implement retrieval algorithm using J2EE for the server side and JSP program for the client side. Cloudscape has been chosen to manage low-level features of 34,000 images. In the experiments, we applied the proposed search engine to the Corel image database and compared its performance with non-interactive retrieval system. Our conclusion is that, the proposed search engine employed GMM-based relevance feedback method gave good results than that of the traditional methods.

กิตติกรรมประกาศ

ขอขอบพระคุณอาจารย์ทุกท่านที่ได้ให้ความรู้ในการทำโครงงานในครั้งนี้ โดยเฉพาะอย่างยิ่ง อาจารย์ไพศาล มุณีสว่าง ซึ่งคอยให้คำปรึกษาและคำแนะนำการทำงานในหลายเรื่อง รวมทั้งยังเสียสละเวลาเพื่อดูแลโครงงานของเรา ขอบคุณหอพักกฤษสิริ ที่เอื้อเฟื้อสถานที่ทำโครงงาน ขอขอบคุณเพื่อนๆ ทุกคนที่เป็นกำลังใจ ขอบคุณมหาวิทยาลัยนเรศวรที่เปิดโอกาสให้ทำโครงงานนี้ รวมทั้งยังเอื้อเฟื้อค่าใช้จ่ายทั้งหมดในการทำโครงงาน ขอบคุณ คุณพ่อและคุณแม่ของพวกเราที่คอยเป็นกำลังใจ และท้ายสุดขอบคุณเพื่อน ๆ ทุกคนที่ให้ความช่วยเหลือตลอดมา

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญของโครงงาน

ในปัจจุบันข้อมูลข่าวสารกลายเป็นปัจจัยที่มีความสำคัญในการดำรงชีวิตของมนุษย์ ประกอบกับการเติบโตอย่างไม่หยุดยั้งของระบบบริการข้อมูลข่าวสารที่สำคัญต่าง ๆ อย่างเช่น อินเตอร์เนต (Internet) ซึ่งเป็นแหล่งที่อยู่ของข้อมูลปริมาณมหาศาล ทั้งข้อมูลที่อยู่ในรูปไฟล์เอกสาร ไฟล์ภาพ ไฟล์วิดีโอ และ ไฟล์เสียง ข้อมูลภาพดิจิตอล (Digital Image) ก็เป็นข้อมูลอีกประเภทหนึ่งที่มีผู้ต้องการใช้จำนวนมาก เช่นสถาบันการศึกษา วงการแพทย์ องค์กรทางธุรกิจ หรือวงการบันเทิง เป็นต้น แต่เครื่องมือที่ใช้ในการค้นหาภาพจากฐานข้อมูลภาพผ่านทางระบบ Internet นั้นในปัจจุบันยังนิยมการสืบค้นโดยใช้ คำสำคัญ (keyword search) ซึ่งให้ผลลัพธ์ที่ดีในระดับหนึ่งเท่านั้น แต่ก็ไม่ได้เป็นวิธีการสืบค้นที่ดีที่สุด ดังนั้นจึงได้มีผู้พยายามคิดหาวิธีการในการสืบค้นภาพแบบใหม่ขึ้น ซึ่งได้แก่วิธี "การค้นหาข้อมูลภาพโดยใช้เนื้อหาของภาพ" (Content-base image retrieval :CBIR) ซึ่งเป็นวิธีการค้นหาโดยการพิจารณาที่เนื้อหาของภาพ (visual content) เป็นหลัก เช่น พิจารณาจากสี,พื้นผิว,หรือรูปร่างของภาพ เป็นต้น ซึ่งวิธีการค้นหาแบบใหม่นี้ให้ผลลัพธ์ในการค้นหาที่ดีกว่าแบบเดิม โดยกระบวนการ (Algorithm) ที่ใช้ในการค้นหาและเปรียบเทียบเนื้อหาของภาพนั้นจนถึงปัจจุบันได้มีผู้คิดค้นขึ้นหลายกระบวนการด้วยกัน แต่ละกระบวนการก็มีความแตกต่างกันทั้งความสามารถ ข้อดีและข้อเสียที่แตกต่างกันออกไป ทั้งนี้ในการเลือกว่าจะใช้กระบวนการใดนั้นก็ขึ้นอยู่กับการตัดสินใจของผู้พัฒนาระบบค้นหา (Search Engine) เป็นหลัก

กระบวนการในการสืบค้นภาพ (CBIR) แบบธรรมดาที่สุดนั้น จะเป็นกระบวนการที่ทำการวัดความใกล้เคียงกันของภาพ (Similarity measurement) ว่ามีความใกล้เคียงกันเพียงใด ซึ่งจะออกมาเป็นตัวเลขที่แสดงถึงความแตกต่างระหว่างภาพต้นแบบ(Query image) และภาพจากฐานข้อมูล ต่อจากนั้นก็จะแสดงภาพที่มีความแตกต่างจากภาพต้นแบบน้อย ๆ ออกมาให้ผู้ใช้ (user) ผู้ใช้ก็จะทำการพิจารณาคัดเลือกภาพเหล่านั้นด้วยตนเองอีกทีหนึ่ง แต่ถ้าหากว่าภาพที่แสดงออกมานั้นยังไม่ใช่ภาพที่ตรงกับที่ผู้ใช้ต้องการแล้ว ก็ไม่สามารถที่จะสั่งให้ระบบค้นหาภาพใหม่อีกครั้งเพื่อให้ได้ภาพที่ตรงกับความต้องการมากกว่านี้ โดยการใช้ภาพต้นแบบอันเดิมได้ เพราะผลลัพธ์ก็จะออกมาเหมือนเดิม

แต่ยังมีกระบวนการที่มีความสามารถสูงขึ้นมาอีกและหนึ่งในกระบวนการเหล่านั้นได้แก่ Gaussian Mixture Model (GMM) Algorithm ซึ่งถูกออกแบบมาให้มีความสามารถมากขึ้น โดยสามารถทำงานแบบ User Interaction กล่าวคือ ระบบจะสามารถโต้ตอบกับผู้ใช้ในระหว่างที่กำลัง

ใช้งานระบบได้ โดยเมื่อระบบทำการเปรียบเทียบและแสดงผลลัพธ์ออกไปสู่ผู้ใช้แล้ว หากภาพที่ได้ยังไม่ตรงกับความต้องการของผู้ใช้แล้วข้อมูลของภาพที่เลือกก็จะถูกส่งกลับให้ระบบ เรียกกระบวนการนี้ว่า "Relevance Feedback(RF)" แล้วระบบก็จะการคำนวณและค้นหาภาพที่มีความใกล้เคียงกับความต้องการของผู้ใช้มากเข้าไปอีก โดยกระบวนการนี้จะดำเนินไปแบบซ้ำเดิม จนกว่าผู้ใช้จะได้ภาพที่ต้องการซึ่งจะเห็นได้ว่าระบบจะมีความฉลาด (Machine intelligence) เพราะว่าสามารถที่จะเรียนรู้ถึงความต้องการของผู้ใช้ได้ และนำมาประมวลผลเพื่อทำให้เกิดผลลัพธ์ของการค้นหาที่ดีที่สุด

1.2 วัตถุประสงค์

1.2.1 เพื่อพัฒนาระบบสืบค้นข้อมูลประเภทภาพถ่ายดิจิตอล จากฐานข้อมูลในเซิร์ฟเวอร์ (Database Server)ให้มีประสิทธิภาพและใช้งานได้จริงคือผลลัพธ์ที่ได้จะต้องมีความถูกต้อง แม่นยำ ประหยัดเวลาในการสืบค้น ง่ายต่อการใช้งาน การสืบค้นอาจจะใช้หลายแนวคิดมาพัฒนาร่วมกัน ดึงเอาความสามารถของแต่ละแนวคิดออกมาใช้งาน ไม่ว่าจะเป็น การค้นหาโดยใช้คำสำคัญ (keyword search) , การวัดค่าความแตกต่างของสี (color similarity),การวัดค่าความแตกต่างของพื้นผิว (texture similarity) และ การวัดค่าความแตกต่างของรูปร่าง (shape similarity) การพัฒนาโปรแกรมกระทำโดยกระบวนการ Content-based image retrieval โดยอาศัย Gaussian Mixture Model (GMM) เป็นอัลกอริทึมในการพัฒนา ซึ่งตัว GMM เองมีความสามารถที่สูงมาก เพราะสามารถเรียนรู้ได้เร็ว และมีประสิทธิภาพสูงในการเรียนรู้ GMM ยังถือได้ว่าเป็นหัวใจหลักของ Mechanism Learning ในลักษณะ User-interface Method

1.2.2 สามารถสร้างเว็บแอพลิเคชัน (web application) เพื่อใช้ในคณะวิศวกรรมศาสตร์ได้โดยตัวเว็บเพจ (web page) จะมีลักษณะที่ใช้งานได้สะดวก คือ มีรูปแบบการใช้งานที่ง่ายต่อผู้ใช้ทุกระดับ ผู้ใช้สามารถเรียนรู้และใช้งานได้ด้วยตนเองเว็บดังกล่าวคือ เว็บเซิร์ฟเวอร์ (web server) ที่ให้บริการสืบค้นข้อมูลผ่านทางระบบเครือข่าย (Lan Network) และทาง Internet Network

1.2.3 เพื่อศึกษาถึงหลักการของการสืบค้นข้อมูลประเภทภาพถ่ายดิจิตอล จาก Database Server หลักการทำงานของระบบ Server – Client มีความรู้ความเข้าใจในระบบฐานข้อมูลจนสามารถนำไปพัฒนาและประยุกต์ใช้ให้เกิดประโยชน์กับโครงงานได้ องค์ความรู้ดังกล่าวคือ ความรู้ในการพัฒนาแอพลิเคชันในการสืบค้นข้อมูลโดยใช้ Content-based image retrieval การพัฒนา web server และ database server โดยใช้โปรแกรม JSP และ J2EE

1.2.4 สามารถพัฒนาระบบการติดต่อสื่อสารผ่านทาง Networkได้ รวมถึงการพัฒนาระบบให้สามารถติดต่อสื่อสารผ่านเครือข่าย Internet ได้ทั้งนี้จะต้องเข้าใจการทำงานของระบบเครือข่ายอิน

เตอร์เนต การเชื่อมต่อผ่าน Protocol ต่าง ๆ ,เข้าใจมาตรฐานของภาษาที่ใช้ในระบบ อินเตอร์เนต นั่นคือภาษา HTML (Hyper Text Makeup Language)

1.2 ขอบข่ายของงาน

เพื่อพัฒนาระบบการสืบค้นข้อมูลประเภทภาพดิจิตอลจากเซิร์ฟเวอร์ โดยใช้ Gaussian Mixture Model(GMM) เป็นอัลกอริทึมในการทำงาน ซึ่ง Server และ Client จะติดต่อกันผ่านทางระบบเครือข่าย (Network) ในการพัฒนาระบบดังกล่าวได้ใช้ J2EE ของบริษัท Sun Microsystem พัฒนาร่วมกับโปรแกรมฐานข้อมูล Cloudscape การพัฒนาระบบแบ่งออกเป็นสองส่วนย่อยดังนี้

1.3.1 พัฒนาระบบ Server โดยใช้ JSP ซึ่งเป็นเครื่องมือในการพัฒนาเว็บเซิร์ฟเวอร์ ใช้J2EE เป็นเซิร์ฟเวอร์โดยเซิร์ฟเวอร์ จะเปรียบเสมือนทรัพยากรของระบบโดยจะมี Client เป็นตัวร้องขอเพื่อสืบค้นข้อมูล ในระบบของเซิร์ฟเวอร์จะเก็บข้อมูลประเภทภาพดิจิตอลไว้และในขณะเดียวกันก็จะเก็บแอพลิเคชันในการสืบค้นข้อมูลรวมอยู่ด้วย โดยลักษณะการแสดงผลของเว็บ เป็นแบบ Dynamic คือจะมีการตอบสนองกับผู้ใช้อยู่ตลอดเวลา

1.3.2 พัฒนาระบบสืบค้นข้อมูลประเภทภาพดิจิตอล ให้สามารถทำงานบนเซิร์ฟเวอร์ได้อย่างมีประสิทธิภาพ โดยใช้ Gaussian mixture model เป็น Algorithm ในการประยุกต์และพัฒนาลักษณะการทำงานของระบบสืบค้นข้อมูลนี้เป็นการค้นหาภาพดิจิตอล เป้าหมายที่มีลักษณะเหมือนหรือใกล้เคียงกับภาพต้นแบบ (Query) โดยพิจารณาจาก รูปร่าง (Shape) , สี (Color) , และเนื้อหา (Texture)

1.4 ผลที่คาดว่าจะได้รับ

1.4.1 สามารถพัฒนาและสร้างสรรค์เว็บเซิร์ฟเวอร์ เพื่อใช้ในคณะวิศวกรรมศาสตร์ ในลักษณะให้บริการข้อมูลประเภทภาพดิจิตอลผ่านทางระบบอินเตอร์เนต โดยสามารถรองรับการทำงานกับผู้ใช้หลายคนในเวลาเดียวกันได้ ทั้งนี้เซิร์ฟเวอร์ดังกล่าวจะต้องให้ผลการทำงานมีความถูกต้อง รวดเร็ว และมีเสถียรภาพด้วย

1.4.2 มีความรู้ความเข้าใจหลักการทำงานของระบบ Server–Client ในระดับที่สามารถพัฒนาและประยุกต์ใช้แอพลิเคชันให้เกิดประโยชน์บนระบบอินเตอร์เนตได้โดยอาศัย JSP และ J2EE เป็นเครื่องมือในการพัฒนาเว็บแอพลิเคชัน

1.4.3 มีความรู้ความเข้าใจในการที่จะพัฒนาเว็บแอพลิเคชัน เพื่อรองรับการทำงานในลักษณะผู้ใช้หลายคน (multi user) รวมถึงสามารถออกแบบโครงสร้างและการทำงานของ ระบบฐานข้อมูลได้อย่างมีเสถียรภาพ และมีประสิทธิภาพ สามารถที่จะเชื่อมโยงฟังก์ชันต่าง ๆ ในระบบที่พัฒนาขึ้นมาเข้าด้วยกันเพื่อให้เกิดประโยชน์สูงสุดโดยผ่านทางระบบอินเตอร์เนตได้

1.5 กิจกรรมการดำเนินงาน

ตารางที่ 1.1 แสดงกิจกรรมการดำเนินงาน

กิจกรรม	ปี 2545		ปี 2546									
	พ.ย	ธ.ค	ม.ค	ก.พ	มี.ค	เม.ย.	พ.ค	มิ.ย	ก.ค	ส.ค	ก.ย	ต.ค.
1. Learn for Theory of Content Base Image Retrieval (CBIR)	←	—	—	—	—	—	—	—	→			
2.Learn for Gaussian mixture model Algorithm and Application		←	→									
3.Image database System Construction and image indexing		←	—	→								
4. Learn J2EE Server for Network Programming		←	—	→								

ตารางที่ 1.1 แสดงกิจกรรมการดำเนินงาน (ต่อ)

กิจกรรม	ปี 2545		ปี 2546									
	พ.ย	ธ.ค	ม.ค	ก.พ	มี.ค	เม.ย	พ.ค	มิ.ย	ก.ค	ส.ค	ก.ย	ต.ค
5. Learn for Java Server Page Programming (JSP)		◄	►									
6. Create CBIR Application Server side			◄	►								
7.Create CBIR Application Client side			◄	►								
8. 1st System testing and Debugging					◄	►						
9. Learn for Internet System and Programming						◄	►					
10. Develop Application for Internet										◄	►	
11. 2nd System testing and debugging											◄	►
12. Document creation						◄						►

บทที่ 2
งานวิจัยที่เกี่ยวข้อง

ในการทำโครงงานแต่ละอย่างจะต้องมีการศึกษาค้นคว้าหาความรู้ที่จะนำมาใช้เพื่อให้เกิดความเข้าใจในหลักการ และทฤษฎีว่าเป็นอย่างไร จึงจะสามารถนำความรู้นั้นมาใช้เกิดประสิทธิผลได้มากที่สุด โครงงานนี้ต้องใช้ความรู้ในเรื่องการค้นหาในรูปแบบต่าง ๆ ไม่ว่าจะเป็นการใช้ คำสำคัญ หรือการค้นหาโดยใช้ลักษณะพื้นฐานของภาพ(Query by Example) เพื่อจะได้รู้ว่าวิธีการใดมีข้อดี-ข้อเสียแตกต่างกันอย่างไร เพื่อจะได้เปรียบเทียบหาข้อสรุปว่าควรจะใช้วิธีการใดจึงจะทำให้งานออกมามีประสิทธิภาพที่สุด นอกจากวิธีการค้นหาแบบต่าง ๆ แล้วความรู้ในเรื่อง Image Processsing ก็มีความสำคัญเช่นกัน โดยความรู้ในส่วนนี้นั้นจะเป็นความรู้เกี่ยวกับสาระของภาพ โดยจะอธิบายภาพแต่ละภาพด้วย ลักษณะพื้นฐานของภาพซึ่งมีหลายลักษณะ เช่น สี(Color) รูปร่าง(Shap) หรือ พื้นผิว(Texture) เป็นต้น

2.1 การค้นหาโดยใช้คำสำคัญ (Key word search)

การสืบค้นข้อมูลโดยใช้คำสำคัญเป็นที่นิยมใช้กันมากเพราะประหยัดเวลาในการค้นหาเป็นอย่างมาก การค้นหานั้นอาจจะได้ผลของการค้นหาเป็นกลุ่มของภาพ โดยภาพแต่ละภาพนั้นจะมีคุณสมบัติเฉพาะของตัวเอง กล่าวคือ มีชื่อภาพ ชื่อผู้สร้าง วันเดือนปีที่ผลิต และแหล่งที่มา การค้นหาภาพประเภทนี้อาจจะใช้เวลารวดเร็วถ้าจัดการข้อมูลโดยใช้การจัดการฐานข้อมูลที่มีประสิทธิภาพ เช่น SQL relational database language ซึ่งเป็นมาตรฐานสำหรับการจัดการฐานข้อมูล ตัวอย่างเช่น

```
SELECT * FROM IMAGEDB
WHERE CATEGORY = 'GEMS' AND SOURCE = 'SMITHSONIAN'
AND (KEYWORD = 'AMETHYST' OR KEYWORD = 'CRYSTAL'
OR KEYWORD = 'PURPLE'
```

จากตัวอย่างดังกล่าวเราจะได้รูปภาพที่มีดรรชนีที่ประกอบไปด้วย ชื่อ IMAGEDB จัดอยู่ในประเภท GEMS แหล่งที่มาคือ SMITHSONIAN และมี keyword คือ AMETHYST , CRYSTAL หรือ PURPLE

จากตัวอย่างดังกล่าวผู้ใช้สามารถสืบค้นข้อมูลโดยการใช้ keyword ได้ แต่อย่างไรก็ตามการสืบค้นโดยการใช้ keyword อาจจะได้ผลลัพธ์ที่ไม่ตรงกับความต้องการนักเนื่องจากมีข้อมูลมากจนทำให้ความซ้ำซ้อนขึ้น หรือการให้คำนิยามที่ไม่ครอบคลุมกับสิ่งที่จะค้นหาหรือภาพเหล่านั้นอาจมี

keyword ที่เหมือนกันแต่เนื้อหารายละเอียดแตกต่างกัน นอกจากการค้นหาข้อมูลโดยใช้ keyword แล้วยังมีแนวทางอื่นในการค้นหาอีก คือ Query by example(QBE)

2.2 การค้นหาโดย CBIR

QBE (Query By Example) เป็นอีกรูปแบบหนึ่งของวิธีการสืบค้นข้อมูลประเภทภาพถ่าย โดยการใส่ค่าที่ต้องการ และเพิ่มข้อจำกัดในการสืบค้นให้กับระบบ ในตอนแรก ๆ นั้น ระบบ QBE ถูกพัฒนาโดย IBM ตัวอย่างของ QBE ที่พบในปัจจุบันเช่น Microsoft Access

ในการค้นหาอาจใช้ภาพถ่ายดิจิตอล , ภาพร่าง , หรือภาพวาดร่วมกับเงื่อนไขต่าง ๆ เพื่อให้ได้ผลลัพธ์ที่ใกล้เคียงกับที่ต้องการ เงื่อนไขที่ใช้อาจเป็น keyword หรือลักษณะเฉพาะของภาพก็ได้แต่โดยทั่วไปจะใช้ภาพถ่ายดิจิตอลเป็นภาพต้นแบบโดยทำการเปรียบเทียบความแตกต่างระหว่าง ตัวอย่างกับข้อมูลในฐานข้อมูล ถ้าผลลัพธ์ที่ได้จากการเปรียบเทียบเป็น "0" (Zero)หมายความว่าค้นพบภาพให้ผู้ใช้เลือกต่อไป การวัดความแตกต่างของภาพเรียกว่า "Image distance measures" ซึ่งเป็นหลักในการทำงานของระบบสืบค้นภาพ

Image distance measures เป็นตัวชี้ว่าภาพมีความแตกต่างกันอย่างไร ซึ่งเกณฑ์ในการวัดจะใช้หลักการเหล่านี้

1. ความแตกต่างของสี (Color similarity)
2. ความแตกต่างของพื้นผิว (Texture similarity)
3. ความแตกต่างของรูปร่าง (Shape similarity)

2.2.1 Color similarity measures เป็นการเปรียบเทียบอย่างง่ายคือใช้สีเป็นตัวเปรียบเทียบระหว่างภาพตัวอย่างกับภาพในฐานข้อมูล เทคนิคคือใช้ Color histogram matching คือให้ผู้ใช้เลือกสีที่ต้องการโดยการกำหนดปริมาณหรืออัตราส่วนของสีในภาพ ผลลัพธ์ที่ออกมาจะได้ชุดของภาพที่มีคุณสมบัติตามที่ได้ระบุไว้ในเงื่อนไข คือมี โทนสีที่ใกล้เคียงกันแต่อาจไม่ใช่ภาพที่เกี่ยวข้องกับภาพต้นแบบ

2.2.2 Texture similarity measures มีความซับซ้อนกว่าแบบที่แล้วที่ได้กล่าวไปข้างต้น ภาพอาจจะมีเนื้อหาหรือ texture ที่คล้ายกันแต่ไม่จำเป็นต้องมีสีเหมือนกัน เทคนิคนี้ใช้ตัดสินความเหมือนกันของภาพ 2 ภาพ ส่วนมากจะอธิบาย texture ด้วยเวคเตอร์ (vector) เป็นเวคเตอร์ของตัวเลขทั้งหมดหรือบางส่วนที่รวมกันขึ้นเป็นค่าของ texture

2.2.3 Shape similarity measures วิธีการนี้จะใช้รูปร่างหรือรูปทรงของภาพ (shape) เป็นตัวเปรียบเทียบ การทำงานนั้นจะใช้ขอบของวัตถุในภาพเป็นตัวกำหนด shape โดยจะเปรียบเทียบรูปทรงของวัตถุในภาพโดยไม่คำนึงถึงสี หรือรายละเอียดอย่างอื่น

2.3 หลักการเปรียบเทียบความเหมือน (Image Distance Measure)

ในการวัดค่าความแตกต่างระหว่างภาพเรามีหลายวิธีการที่จะตัดสินความแตกต่างอยู่หลายรูปแบบ แล้วแต่ว่าจะพิจารณาตามแบบใด หรือว่าจะใช้มากกว่าหนึ่งแบบร่วมกันก็จะทำให้ประสิทธิภาพดียิ่งขึ้น โดยวิธีการพิจารณาแบ่งออกเป็น 3 อย่างหลักๆ ดังนี้

1. การวัดความแตกต่างของสี (Color Similarity Measure)
2. การวัดความแตกต่างของพื้นผิว ๖Texture Similarity Measure)
3.การวัดความแตกต่างของรูปราง (Shape Similarity Measure)

2.3.1 การวัดค่าความเหมือนของสี (Color Similarity Measure)

การวัดค่าความเหมือนของสีเป็นการเปรียบเทียบลักษณะของสีของรูปหนึ่งกับอีกรูปหนึ่งที่เป็นรูปที่ใช้เป็นรูปต้นแบบหรือเปรียบเทียบกับการกำหนดสีของรูปที่ต้องการโดยการโดยตรง (a query specification) ยกตัวอย่าง เช่น ระบบ QBIC สามารถให้ผู้ใช้กำหนดเปอร์เซ็นต์ของสีต่างๆ ในภาพโดยตรง โดยผู้ใช้จะเลือกสีจากตารางค่าสีและกำหนดเปอร์เซ็นต์ของสีต่างๆ ระบบ QBIC จะมองหาภาพที่มีค่าสี และเปอร์เซ็นต์ของสีใกล้เคียงกับผู้ใช้กำหนดที่สุด โดยที่การจัดวางตำแหน่งของสีบนภาพจะไม่มีผลต่อการเลือกแต่อย่างใด ดังนั้นภาพจะมีความแตกต่างขององค์ประกอบสี (Composition) ดังภาพตัวอย่าง

รูปที่ 2.1 แสดงภาพที่มีสีใกล้เคียงกัน [1]

โดยเทคนิคที่ใช้ในการเปรียบเทียบได้แก่ เทคนิค color histogram matching โดยระบบจะเตรียมภาพตัวอย่างเพื่อให้ระบบค้นหาภาพที่มีความแตกต่างระหว่าง color histogram น้อยๆ ยกตัวอย่างระบบ QBIC คำนวณหาค่าความแตกต่างระหว่าง color histogram โดยสมการ

$$d_{hist}(I,Q) = [h(I) - h(Q)]^T A[h(I) - h(Q)] \qquad (2.1)$$

โดย h(I) และ h(Q) คือ k-bin histograms ของภาพ I และ Q และ A เป็น k x k similarity matrix จากเมตริกซ์นี้ สีที่มีค่า ใกล้เคียงกันจะให้ค่าเข้าใกล้1 แต่ถ้าสีที่มีค่าที่แตกต่างกันมากๆ จะให้ค่าที่เท่ากับหรือใกล้เคียงกับ 0

ตำแหน่งของสี (color layout) เป็นอีกอย่างหนึ่งที่ใช้ในการวัดความแตกต่างเป็นสิ่งที่ง่ายมาก สำหรับการที่ผู้ใช้จะเลือกสีมาระบายลงในตำแหน่งต่างๆของภาพต้นแบบ เพื่อให้ระบบค้นหาภาพที่มีตำแหน่งการจัดวางของสีใกล้เคียงกับตำแหน่งการจัดวางสีของภาพ โดยการวัดค่าความแตกต่างจะถูกตัดสินโดยวิธี simple color layout distance measure โดยวิธีนี้จะทำการคำนวณโดยใช้ a grid square color distance measure ซึ่งจะเป็นการเปรียบเทียบค่าความแตกต่างของสีในแต่ละตำแหน่งพิกัด แล้วนำค่าทั้งหมดมาประมวลเป็นค่าความแตกต่างเพียงค่าเดียว

$$d_{gridded\_color}(I,Q) = \sum_{g} d_{color}(c^{I}(g), c^{Q}(g)) \qquad (2.2)$$

โดย C^{I} (g) คือ สี ณ ตำแหน่งของพิกัด g ของภาพต้นแบบ Iและ C^{Q}(g) คือ สี ณ ตำแหน่งของพิกัด g ของภาพที่นำมาเปรียบเทียบ Q

2.3.2 การวัดค่าความเหมือนของพื้นผิวภาพ(Texture Similarity Measures)

การวัดค่าความเหมือนของ Texture (Texture Similarity Measures) มีความซับซ้อนกว่าการวัดค่าความเหมือนของสี(Color Similarity Measures) รูปภาพที่มีพื้นผิวเหมือนกัน เป็นภาพที่มีการจัดวางองค์ประกอบต่าง ๆ คล้ายกันแต่ไม่จำเป็นต้องมีสีเหมือนกัน การวัดค่าความแตกต่างของ Texture สามารถทำได้โดยใช้วิธี Laws texture energy measures ซึ่งประกอบด้วย

1. ตัวแทนของพื้นผิว
2. การให้คำนิยามความเหมือนกันของพื้นผิว

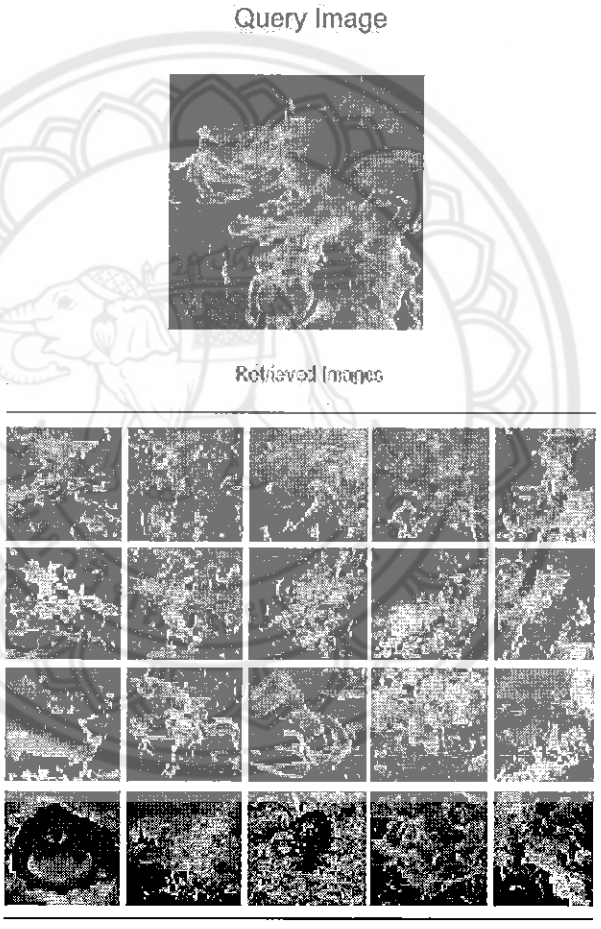
ตัวแทนของพื้นผิวที่นิยมใช้กันทั่วไปได้แก่ Texture description vector ซึ่งเป็นเวคเตอร์ของค่าของตัวเลขทั้งหมดของพื้นผิวของภาพทั้งหมดหรือบางส่วนของภาพเวคเตอร์ของ Haralick's five co-occurrence-based texture feature และของ Laws' mine texture energy feature เป็นตัวอย่างของ texture description vector เนื่องจาก texture description vector เป็นเวคเตอร์ที่คำนวณจากพื้นที่ทั้งหมดของรูปภาพ นี่จึงเป็นวิธีการที่ดีเพียงวิธีเดียวที่ใช้อธิบายภาพที่มี texture เพียง texture เดียวทั้งภาพ สำหรับภาพง่าย แล้ว texture description vector ได้จากการคำนวณกลุ่มของพิกเซลเล็ก ๆ ที่อยู่ใกล้กันเช่น 15 x 15 พิกเซลซึ่งพิกเซลถูกรวบรวมเข้าด้วยกันโดย clustering algorithm ซึ่งจะให้ชื่อที่เป็นชื่อเฉพาะกับแต่ละ texture ที่แตกต่างกันที่หาเจอบนภาพ

เมื่อมีการให้ชื่อเฉพาะกับแต่ละกลุ่มของพิกเซล ความหลากหลายของความแตกต่างกับระหว่างพื้นผิวก็จะเกิดขึ้น ซึ่งความแตกต่างกันอย่างง่าย ๆ ของ Texture ได้แก่ pick-and-click distance เช่นเมื่อผู้ใช้เลือกที่จุดใดจุดหนึ่งบนพื้นผิวของ texture ต้นแบบ texture ที่ถูกเลือกนั้นก็

จะถูกแทนที่ด้วย texture description vector ก็จะถูกคำนวณเก็บไว้ในฐานข้อมูลซึ่งการวัดความแตกต่างจะถูกนิยามโดย

$$d_{pick-and-click}(I,Q) = \text{mine}\sum\backslash\backslash T(i) - T(Q)\backslash\backslash^2 \qquad (2.3)$$

โดย T(i) ก็คือ texture description vector ของพิกเซล I ของภาพที่ถูกเลือกของภาพที่ใช้เป็นต้นแบบในการค้นหา โดย pick-and-click distance ต้องการให้ผู้ใช้เลือกพื้นที่ที่จะใช้ในการคำนวณ มันไม่สามารถจะกระทำได้อย่างอัตโนมัติบนภาพที่กำหนดมา



Weights: Perceptual Grouping = 0.1, Color = 0.3, Texture = 0.6, L, A, B channels

รูปที่ 2.2 แสดงการค้นหาโดยพิจารณาจากพื้นผิว [2]

2.3.3 การวัดค่าความเหมือนของรูปร่าง (Shape Similarity Measure)

สี (Color) และ พื้นผิว (texture) นับว่าเป็นลักษณะโดยพื้นฐานของรูปภาพ การวัดค่าความแตกต่าง (Distance Measure) ของลักษณะทั้งสองก็เพียงเป็นการหาว่าในรูปลักษณะพื้นฐานของ

รูปภาพ มันเป็นการยากที่จะตอบว่า shape ของภาพมีลักษณะใด นอกจากนี้ shape ยังเกี่ยวพันถึงการเจาะจงพื้นที่เฉพาะบนภาพอีกด้วย shape เป็นสิ่งที่มีลักษณะเหนือกว่า color และ texture อยู่หนึ่งขั้น ตรงที่ต้องมีการกำหนดพื้นที่เฉพาะบนภาพ ที่จะใช้สำหรับคำนวณหาค่าความเหมือนของ shape (Shape Similarity Measure) ในหลายๆ กรณีขั้นตอนนี้จะถูกกระทำแบบ manual แต่ในบางกรณีก็สามารถที่จะใช้การ segmentation โดยอัตโนมัติได้ แต่การ segmentation ก็ยังเป็นวิธีที่ยังใช้ได้ผลไม่ดีนัก อย่างน้อยก็ก่อนที่วิธี Shape-base retrieval จะได้รับความนิยม

Two-dimensional shape recognition ก็เป็นแนวคิดที่สำคัญในการวิเคราะห์ภาพ วิธี shape matching นับเป็นเทคนิคที่ shape จะถูกอธิบายจากส่วนประกอบและความสัมพันธ์ระหว่างส่วนประกอบของภาพ เมื่อสิ่งเหล่านี้ถูกแทนที่ด้วย relation graph แล้ววิธี graph matching ก็จะถูกนำมาใช้ในการเปรียบเทียบได้ แต่อย่างไรก็ตามวิธีนี้เป็นกระบวนการที่ช้ามาก การคำนวณจะมีลักษณะ exponential ของจำนวนองค์ประกอบที่พิจารณาของภาพ แต่ในระบบ CBIR เราต้องการวิธีที่รวดเร็วในการทดสอบความเหมือนกันของภาพ

Shape measure นิยมใช้กันมากในวงการ Computer Vision แต่ว่ามันถูกใช้เป็นกระบวนการช่วยที่ไม่ใช่วิธีหลักในการทำงาน object recognition เป็นเป็นวิธีการคำนวณในรายละเอียดเล็กๆ เพื่อช่วยในการค้นหา object ที่ต้องการ Shape histogram เป็นวิธีวัดอีกวิธีหนึ่งที่ใช้ในขณะที่ shape measure ใช้ไม่ได้ดี แต่ก็ยังเป็นวิธีที่ไม่ดีนัก เช่นเดียวกับ color histogram Boundary techniques ก็เป็นวิธีที่เฉพาะเจาะจงลงไปที่การแสดงถึงขอบของรูปร่าง และมองหา shape ที่มีลักษณะเหมือน Sketch matching ก็เป็นวิธีที่ใช้กับการที่เจาะจงยิ่งกว่า เพราะไม่เพียงแต่มองหา object เพียง object เดียวเท่านั้น แต่เป็นการมองหา object ได้หลาย object ในรูปภาพเดียว ซึ่งเป็น object ที่ผู้ใช้วาดขึ้นอย่างคร่าวๆ เท่านั้น

Shape Histograms การคำนวณค่าความแตกต่างโดยใช้ histogram เป็นวิธีที่ง่ายต่อการคำนวณ และยังใช้กับทั้ง color และ texture matching และเป็นการง่ายที่จะขยายต่อเป็น shape matching แต่ปัญหาหลักอยู่ที่การ define ตัวแปรที่จะใช้ใน histogram ในการพิจารณา shape คือ พื้นที่ที่มีค่าบิตเท่ากับ 1 ในขณะที่พิกเซลอื่นๆ มีค่าบิตเท่ากับ 0 Histogram matching ประเภทหนึ่งก็คือ projection matching โดยอาศัยหลักการ horizontal and vertical projection of the shape โดยสมมุติว่า shape มีลักษณะ n แถว และ m หลัก ในแต่ละแถวและในแต่ละหลักถูกแทนด้วยระบบเลขฐานสอง

Boundary Matching โดย Boundary matching algorithm ต้องการการค้นหา boundary ของวัตถุหรือภาพที่ต้องการ boundary สามารถที่จะแสดงในรูปของลำดับของพิกเซลหรือความใกล้เคียงกับรูปหลายเหลี่ยม สำหรับแบบลำดับของพิกเซล วิธีดั้งเดิมที่ใช้ matching ก็คือ Fourier

descriptors ใช้ในการเปรียบเทียบกันระหว่างสอง shape ในส่วนของ continuous mathematics. Fourier descriptors เป็นสัมประสิทธิ์ของ Fourier series ซึ่งนำมาใช้งานในการ define the boundary ของ shape แต่ในส่วนของ Discrete mathematics shape จะถูกแสดงโดย ลำดับของ m จุด (v0, v1,v2,...,vm-1) จากลำดับของจุดจะได้ลำดับของ Unit vectors

$$V_k = \frac{V_{k+1} - V_k}{|V_{k+1} - V_k|} \tag{2.4}$$

และลำดับของ Cumulative differences คำนวณจาก

$$I_k = \sum_{i=0}^{k} |V_i - V_{i-1}|, k > 0, I_o = 0 \tag{2.5}$$

และ Fourier descriptors $\{a_{-M},\ldots,a_0,\ldots a_M\}$ หาได้จาก

$$a_n = \frac{1}{L(\frac{n2\pi}{L})^2} \sum_{k=1}^{m} (v_{k-1} - v_k) e^{-jn(2\pi/L)t_k} \tag{2.6}$$

Sketch Matching ช่วยให้ผู้ใช้ค้นหาได้โดยใช้เพียงภาพร่างคร่าวๆของ สิ่งที่ต้องการ ซึ่งจะมีสีหรือเป็น gray-scale ก็ได้ ในพิพิธภัณฑ์ศิลปะ จะมีการเก็บข้อมูลภาพวาดด้วยสีที่มีชื่อเสียง ซึ่งภาพวาดบางภาพจะออกมาในแนว abstract image โดย

1. ประยุกต์ affine transform เพื่อลดขนาดของภาพให้มีขนาดเท่ากับที่กำหนดไว้ เช่น 64X64 พิกเซล แล้วทำการกลั่นกรองเพื่อขจัดสิ่งที่ไม่ต้องการ จะได้ภาพที่ normalized แล้ว
2. ทำการ detect edge โดยใช้ gradient-based edge-finding algorithm
3. ปรับปรุงเส้นขอบโดย การทำให้บางและเล็กลง ซึ่งผลลัพธ์ก็คือ ภาพ abstract นั่นเอง

กระบวนการที่เริ่มตั้งแต่การที่ผู้ใช้ป้อนภาพร่างที่ต้องการ ผ่านกระบวนการ normalized size , binarized , thinned and shrunk ผลลัพธ์ที่ออกมาเรียกว่า linear sketch และจะถูกนำไปค้นหาภาพ abstract โดยภาพทั้งสองจะถูกแบ่งออกเป็นช่องๆ (grid square) การวัดความเหมือนจะเป็นผลรวมทั้งหมดของ local correlation และ distance measure จะเป็นส่วนกลับของ similarity measure จากสมการที่ผ่านมา จะได้ว่า

$$d_{Flynter}(I,Q) = \left[\sum_{n=-M}^{M} \left| a_n^I - a_n^Q \right|^2 \right]^{\frac{1}{2}} \quad (2.7)$$

Descriptor นี้อาจจะใช้ใน shape distance measure โดย Q คือ query shape และ I คือ shape ที่ใช้เปรียบเทียบกับ Q

2.4 การสืบค้นโดยการป้อนกลับจากผู้ใช้ (Retrieval with User Interaction)

การสืบค้นโดยให้ผู้ใช้เป็นผู้ช่วยเหลือให้กับระบบสืบค้นในส่วนของ CBIR เป็นวิธีที่มีประสิทธิภาพมาก หลักการดังกล่าว ได้รับแนวคิดมาจาก RF technique ซึ่งใช้ในระบบสืบค้นข้อมูลแบบใหม่ ซึ่งช่วงเวลาไม่กี่ปีที่ผ่านมา RF technique ได้ถูกพัฒนาขึ้นอย่างรวดเร็ว [3]

งานวิจัยส่วนใหญ่ที่เกี่ยวกับ Image Retrieval จะเป็นการพัฒนาในส่วนของวิธีการเพื่อปรับปรุงงานวิจัยที่มีอยู่ในปัจจุบันการประยุกต์ใช้เทคนิคดังกล่าวที่รู้จักกันดีคือ Multimedia Analysis and Retrieval System (MARS) ได้รับการพัฒนาขึ้นที่มหาวิทยาลัย อิลลินอยส์ ลักษณะเด่นของระบบประกอบด้วย PicToSeek , Draw Search และ Viper หลักการพื้นฐานคือ จะพยายามปรับปรุงและสร้าง Query ตัวใหม่จาก Query ตัวเดิม โดยอาศัย Feedback มาจากผู้ใช้ (RF)

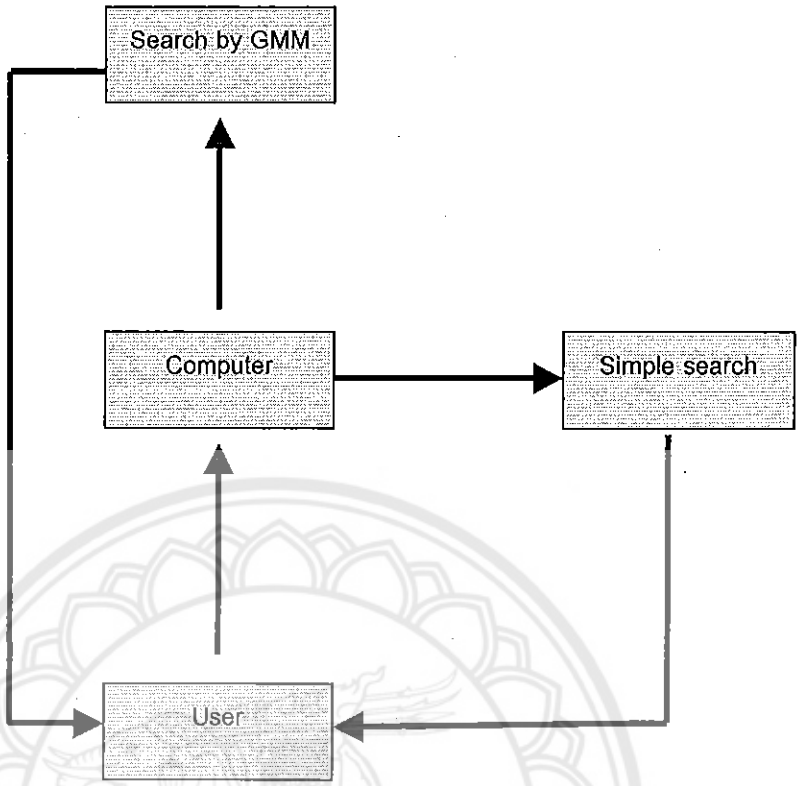
วิธีการดังกล่าวเป็นวิธีที่มีประสิทธิภาพ แต่อย่างไรก็ตามวิธีการนี้ต้องใช้กระบวนการในการเปลี่ยน Image ให้อยู่ในรูปของ Term-Weighting Model นอกจากนี้ในบางระบบการที่จะสามารถรันได้อาจต้องใช้ข้อมูลที่มีความซับซ้อนมาก ๆ แต่อย่างไรก็ตามงานวิจัยช่วงหลัง ๆ ที่มีวิธีการที่แตกต่างออกไป ก็ได้ถูกเสนอขึ้นมาเทคนิคดังกล่าวประกอบไปด้วย Multidimensional index structures และ Weighted average

Relevance feedback เป็นเทคนิคที่จะช่วยให้ระบบสามารถเรียนรู้และจำลองสภาพการรับรู้ของมนุษย์ได้ เป็นการป้อนกลับผลลัพธ์ที่ผู้ใช้ต้องการและผลลัพธ์ที่ผู้ใช้ไม่ต้องการคืนให้กับระบบค้นหา เพื่อเป็นการสอนให้ระบบเกิดการเรียนรู้และทำการค้นหาซ้ำอีกครั้ง โดยในการค้นหาครั้งใหม่จะได้ผลลัพธ์ที่ดีขึ้น มีความใกล้เคียงกับความต้องการของผู้ใช้มากยิ่งขึ้น กล่าวคือประสิทธิภาพในการทำงานของระบบจะเพิ่มขึ้น วิธีการนี้ได้มาจากแนวคิดพื้นฐานของ similarity functions รวมถึงการกำหนดระดับค่าความสำคัญโดยตัวเลข (Weighting numerical) การกำหนดระดับค่าความสำคัญเป็นกระบวนการที่ต้องพิจารณาถึงเซตของตัวแปรที่เราสนใจ และต้องใช้ Nerual network model ข้ามาช่วยรวมถึงต้องใช้ทฤษฎีการจัดแบ่งตามพื้นฐานความน่าจะเป็น (Probalitistic-based classification method) หรือใช้ Minkowski metrics ร่วมด้วย การ Weighting โดย similarity metric มักจะใช้ในรูปของ Euclidean distance ซึ่งจะช่วยเป็นแนวทาง

พื้นฐานในการออกแบบ similarity function อย่างไรก็ตามประสิทธิภาพการทำงานของ similarity function ก็มีขีดจำกัดอยู่ข้อนข้างมาก เนื่องจากตัวมันเองสามารถจัดการได้เฉพาะสมการในรูปกำลังสอง (Quadratic form) เท่านั้น ซึ่งไม่สามารถจะรับมือกับการทำงานของ Image similarity ที่เป็น complex ได้

Neural-Network Models ได้ถูกเสนอขึ้นมาเพื่อจัดการกับปัสญหาดังกล่าวในแง่ของความสามารถที่จะสามารถเรียนรู้ได้ และความสามารถในการจำลอง universal เพื่อ mapping แต่ Neural-Network ต้องการจำนวนข้อมูลในการประมวลผลของแต่ละ Image จำนวนมาก ดังนั้นจึงเกิดสถาปัตยกรรมใหม่ที่ใช้จัดการกับปัญหานี้ การใช้ Non linear Model โดยใช้ Gaussian-shaped และ RBF ได้ถูกเสนอขึ้นเพื่อจัดการกับปัญหา complex decision boundaries ประโยชน์หนึ่งที่เห็นได้ชัดของ Model นี้คือ มันต้องการปริมาณข้อมูลน้อยในการประมวลผล และมันสามารถทำงานได้อย่างรวดเร็ว

เราใช้ RF ในการปรับปรุงกลุ่มของ image , ปรับปรุงความน่าจะเป็นของฐานข้อมูล และปรับปรุง feedback ของผู้ใช้ Bayesian inference เป็นกระบวนการที่นิยมใช้มาก ใช้ในการทำนายหาเป้าหมายที่ต้องการจากตัวอย่างที่กำหนด ตัวอย่างเช่น PicHunter ใช้หลักการการกระจายของความน่าจะเป็นเข้าช่วย ซึ่งความน่าจะเป็นของแต่ละภาพจะมีลักษณะเฉพาะตัว เราใช้ Bayesian inference ดูว่าในฐานข้อมูลมีภาพที่เหมือนหรือต่างกันกับภาพเป้าหมาย จากนั้นเราจะใช้ Bayesian feedback นำเสนอทางเลือกให้กับผู้ใช้ เพื่อให้ผู้ใช้เลือกสิ่งที่ต้องการ

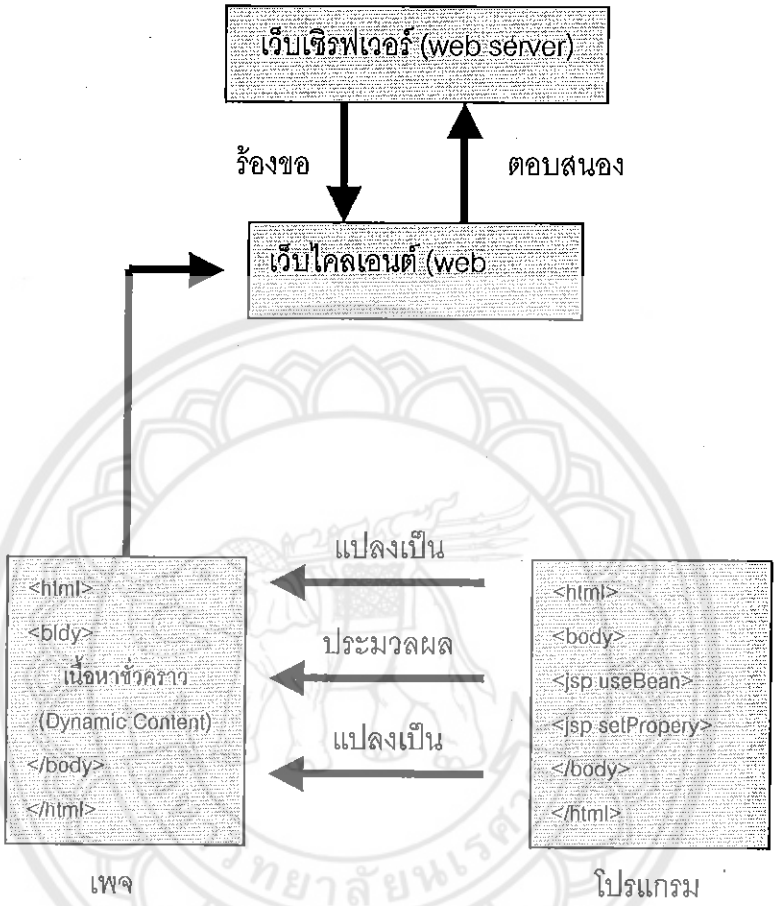


รูปที่ 2.3 แสดงแผนภาพแสดงการค้นหาโดยมีผู้ใช้เป็นผู้ป้อนกลับ

2.5 จาวาเซิร์ฟเวอร์เพจ

JSP (Java Server Page) [4] เป็นเทคโนโลยีสำหรับพัฒนาเว็บต่าง ๆ ที่มีเนื้อหาแบบไดนามิค (dynamic) ซึ่งสามารถเปลี่ยนแปลงได้และโต้ตอบกับผู้ใช้ ไม่เหมือนกับเพจของ HTML ตามปกติซึ่งจะบรรจุเนื้อหาคงที่ เพจของ JSP สามารถบรรจุเนื้อหาต่างๆ, การจำแนกผู้ใช้, ประเภทบราวเซอร์ของผู้ใช้, สารสนเทศต่างๆเกี่ยวกับผู้ใช้, และสิ่งต่าง ๆเกี่ยวกับเว็บเพจที่ผู้ใช้ชื่นชอบ เพจของ JSP จะบรรจุสมาชิกต่างๆของภาษาที่กำหนดเครื่องหมายมาตราฐาน เช่น แท็กต่างๆของ HTML ทั่วไป อย่างไรก็ตาม เพจของ JSP ที่บรรจุสมาชิกต่างๆ ของ JSP ใช้งานได้อย่างกว้างขวาง เช่น การนำสารสนเทศจากฐานข้อมูลออกมาดำเนินการหรือการลงทะเบียนผู้ใช้ เมื่อผู้ใช้ถามหาเพจของ JSP แล้วส่งผลกลับไปยังบราวเซอร์ (ดังรูปที่2.4) JSP จะกำหนดสมาชิกมาตรฐานต่างๆใช้ประโยชน์สำหรับแอพพลิเคชันของเว็บใด ๆ เช่น การเข้าถึงคอมโพเนนต์ ต่างๆของ JavaBeans, การผ่านคอนโทรลระหว่างเพจต่างๆ, และการใช้สารสนเทศร่วมกันระหว่างเพจต่าง ๆ ที่ร้องขอและผู้ใช้ต่าง ๆ โปรแกรมเมอร์สามารถเพิ่มความสามารถให้คำสั่งของ JSP โดยการสนับสนุนสมาชิกที่ระบุแอพลิเคชันที่ดำเนินงานต่าง ๆ เช่น การเข้าถึงฐานข้อมูลต่างๆ และ EJB(Enterprise

JavaBeans), การส่งอีเมล์, และการสร้างเอกสารของ HTML เพื่อป้องกันข้อมูลของแอพลิเคชัน สมาชิกมาตรฐานต่าง ๆ และสมาชิกต่าง ๆ ที่สร้างขึ้นจะช่วยสร้างแอพลิเคชันของเว็บที่มีประสิทธิภาพสูงมาก



รูปที่ 2.4 แสดงการสร้างเนื้อหาแบบไดนามิคด้วยสมาชิกต่างๆของ JSP

ในกระบวนการโต้ตอบระหว่างเครื่องเซิร์ฟเวอร์กับผู้ใช้ (Relevance Feed back) จำเป็นจะต้องมีเว็บแอพลิเคชัน ที่เป็นลักษณะของ ไดนามิคเว็บ (Dynamic web) โปรแกรมต่าง ๆ ของ CGI (Common Gateway Interface) เป็นเครื่องมือเฉพาะสำหรับพัฒนาเนื้อหาของเว็บประเภทไดนามิค ได้แก่ FastCGI ,mod_perl จาก Apache, NSAPI จาก Netscape, ISAPI จากไมโครซอฟท์ และ เซิรฟเล็ต (servlet) ของ Java จาก Sun Microsystems ได้ใช้งานมาหลายปีแล้ว

โปรแกรมดังกล่าวจะสร้างเว็บเพจต่าง ๆ โดยการฝัง HTML โดยตรงในโค้ดของภาษาโปรแกรม เพื่อทำให้เว็บเพจสามารถแสดงเนื้อหาแบบไดมามิค (Dynamic) หรือเนื้อหาชั่วคราวใช้ในการโต้ตอบกับผู้ใช้แต่ JavaServer Pages หรือ JSP ได้เปลี่ยนแปลงสิ่งต่างๆดังกล่าวทั้งหมดเพื่อให้มีสมรรถนะสูง จุดเด่นของ JSP ก็คือ เพจของ JSP จะคอมไพล์ก่อนที่เซิร์ฟเวอร์จะดำเนิน-

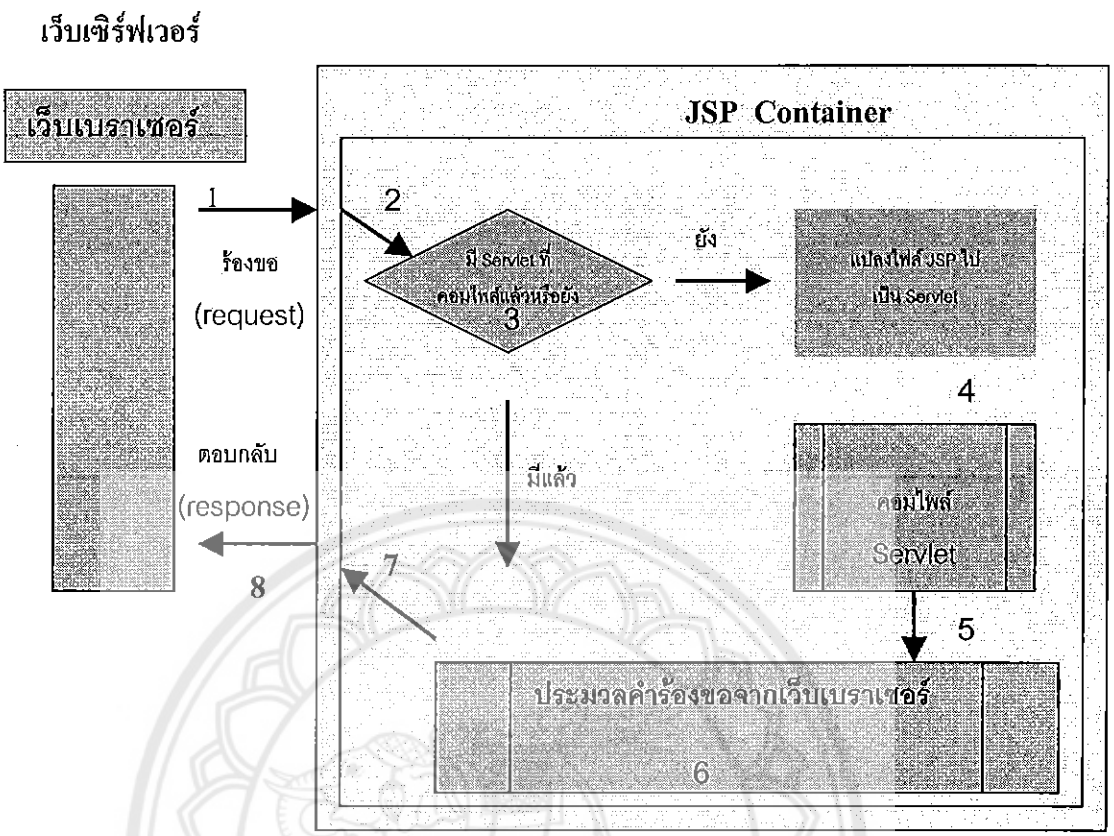
การ ถ้าเป็นเทคโนโลยีเก่า เช่น ภาษา Perl ของ CGI จะต้องการเซิร์ฟเวอร์โหลดตัวแปลภาษา(Interpreter) และสคริปต์แต่ละครั้งที่ร้องขอเพจจากบราวเซอร์ JSP แก้ปัญหาดังกล่าวโดยการคอมไพล์แต่ละเพจของ JSP เป็นไฟล์ .exe ในครั้งแรกที่ร้องขอเพจนั้น เมื่อ JVM (Java Virtual Machine) อยู่บนเว็บเซิร์ฟเวอร์ที่ให้การสนับสนุน JSP จะทำให้เซิร์ฟเวอร์เชื่อมโยงกับเพจต่าง ๆ ของ JSP ได้เร็วขึ้น และจุดเด่นอีกข้อของ JSP ก็คือ JSP สามารถทำงานได้หลายแพลตฟอร์ม เช่น Solaris, Linux, Windows NT/2000/XP, Mac OS, AIX, HP-UX, และ Unix อื่น ๆ

โครงสร้างและขั้นตอนการทำงานของ JSP

สิ่งที่มีบทบาทสำคัญในการทำงานของ JSP ได้แก่ JSP Container (หรือเรียกอีกอย่างว่า JSP Engine) ซึ่งเป็นส่วนประกอบสำคัญที่อยู่ในเว็บเซิร์ฟเวอร์ เพราะทำหน้าที่ควบคุมและประมวลผลไฟล์ JSP ที่มีการร้องขอ (request) เข้ามา และตอบสนอง (response) คำร้องขอนั้นไปยังไคลเอนต์

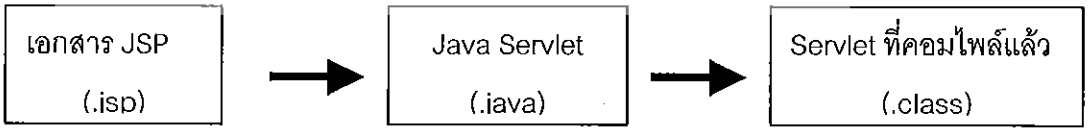
ขั้นตอนการประมวลไฟล์ JSP ทั้งหมด แบ่งเป็น 8 ขั้นตอน ดังนี้

1. ฝั่งไคลเอนต์ส่งคำร้องขอเอกสาร JSP ไปที่เว็บเซิร์ฟเวอร์
2. เว็บเซิร์ฟเวอร์ตรวจสอบการร้องขอ พบว่าเป็นไฟล์ JSP จึงส่งต่อไปให้ JSP Container
3. JSP Container ตรวจสอบว่าไฟล์ JSP ที่ร้องขอมา เคยแปลงเป็น Servlet และ คอมไพล์เป็นไฟล์ .class แล้วหรือยัง โดยดูว่ามีไฟล์ .class อยู่หรือเปล่า ถ้ายังไม่มี ก็จะกระโดดไปทำตามข้อ 4 ต่อ แต่ถ้ามีอยู่แล้ว ก็จะตรวจสอบอีกว่า หลังจากที่แปลงไฟล์ JSP เป็น Servlet และคอมไพล์เป็นไฟล์ .class ครั้งล่าสุดแล้ว ไฟล์ JSP นั้นมีการเปลี่ยนแปลงแก้ไขหรือเปล่า ถ้ามีการแก้ไข ก็จะกระโดดไปทำงานตามขั้นตอนที่ 4 ต่อเช่นกัน แต่ถ้าไม่มีการแก้ไข แสดงว่าไฟล์ JSP นั้นยังคงเดิมไม่เปลี่ยนแปลง จึงไม่มีความจำเป็นที่ต้องเปลี่ยนแปลงเป็น Servlet และคอมไพล์ใหม่ ก็ข้ามไปยังขั้นตอนข้อ 6 ได้เลย
4. JSP Container แปลงไฟล์ JSP เป็น Java Servlet
5. JSP Container คอมไพล์ไฟล์ Java Servlet เป็นไฟล์ .class
6. JSP Container ประมวลผลตามคำร้องขอนั้น
7. JSP container ส่งผลลัพธ์ที่ได้จากการประมวลผล ให้แก่เว็บเซิร์ฟเวอร์
8. เว็บเซิร์ฟเวอร์ส่งผลลัพธ์นั้นไปยังไปยังไคลเอนต์หรือเว็บเบราเซอร์อีกทอดหนึ่ง



รูปที่ 2.5 แสดงโครงสร้างและขั้นตอนการประมวลผลไฟล์ JSP

จากขั้นตอนการประมวลผลไฟล์ JSP ที่แจกแจงข้างต้น สามารถแบ่งออกได้เป็น 2 ช่วงหลักๆ คือ ช่วง translation และช่วง execution โดยช่วง translation ได้แก่ขั้นตอนข้อ 4 และขั้นตอนข้อ 5 ซึ่งเป็นการแปลงเอกสาร JSP (ไฟล์ .JSP) ให้เป็น Servlet (ไฟล์ .java) จากนั้นก็จะคอมไพล์ไฟล์ Servlet ให้เป็นไฟล์ .class ดังรูป



รูปที่ 2.6 แสดงขั้นตอนการประมวลไฟล์ JSP ในช่วง translation

ส่วนช่วง execution ได้แก่ขั้นตอนข้อ 6 ซึ่งเป็นการนำเอาไฟล์ .class ที่ได้จากการคอมไพล์มาประมวลผลหรือทำงานตามคำร้องขอจากไคลเอนต์นั่นเอง

ปกติแล้วกระบวนการทำงานในช่วง translation จะกินเวลาพอสมควร แต่โชคดีว่าช่วง translation จะไม่เกิดขึ้นทุกครั้งที่มีการร้องขอไฟล์ เพราะตราบใดที่ไฟล์ JSP ต้นฉบับไม่มีการเปลี่ยนแปลงอะไร เมื่อมีการร้องขอไฟล์เข้ามาใหม่ ก็ย่อมไม่มีความจำเป็นที่จะแปลงไฟล์เป็น Servlet และคอมไพล์เป็นไฟล์ .class อีก ระบบจะเข้าสู่ช่วง execution ทันทีโดยใช้ไฟล์ .class ที่มีอยู่แล้ว การทำงานจึงรวดเร็วยิ่งขึ้น แต่ถ้ามีการเปลี่ยนแปลงหรือแก้ไขไฟล์ JSP ใหม่ ก็จะต้องเข้าสู่กระบวนการ translation ใหม่ทุกครั้ง

สรุปว่ากระบวนการ translation มีโอกาสจะเกิดขึ้นได้ 2 กรณี กรณีแรกคือ ไฟล์ JSP ที่ร้องขอมา เป็นไฟล์ใหม่ที่ยังไม่เปลี่ยนแปลงและคอมไพล์มาก่อน กับอีกกรณีคือ ไฟล์ JSP ที่ร้องขอมา เคยผ่านการแปลงและคอมไพล์มาแล้ว แต่ภายหลังมีการเปลี่ยนแปลงแก้ไขไฟล์ JSP นั้นไปจากเดิม

บทที่ 3
วิธีการดำเนินงาน

จากหลักการของ Feature descriptor เราจะสามารถ Indexing Image ในฐานข้อมูลได้ ซึ่งเราจะ Indexing ในรูปของเวคเตอร์ซึ่งมีมิติแปรไปตามความต้องการของเราว่าเราจะให้แต่ละเวคเตอร์เก็บค่าอะไรไว้บ้าง เช่น ค่า Color Histogram หรือ ความเข้มของสี เป็นต้น ในขณะเดียวกัน Imageที่เราใช้เป็นต้นแบบ (Query Image) ก็จะต้อง Indexing ด้วยวิธีการเดียวกันกับฐานข้อมูล

ในส่วนของการทำงานของระบบเราใช้ GMM เป็นอัลกอริทึมในการวัดค่าความแตกต่างของ Query Image กับ Image ในฐานข้อมูล การใช้ GMM ในการค้นหาจะทำให้เราได้ภาพที่มีความเป็นไปได้ว่าใกล้เคียงกับภาพที่ผู้ใช้ต้องการเนื่องจาก GMM จะมีอัลกอริทึมในการวิเคราะห์ตัดสินใจคล้ายกับกระบวนการคิดของมนุษย์ กล่าวคือ มนุษย์จะไม่สามารถแยกแยะความแตกต่างของภาพได้ว่ามีเปอร์เซ็นต์สีของแต่ละสีเป็นเท่าใด มีความโปร่งแสงหรือทึบแสงมากน้อยเพียงใด ฯลฯ การบอกความแตกต่างของมนุษย์ใช้ความรู้สึกเป็นตัว เช่น ในภาพทิวทัศน์ที่เป็นทะเลยามเย็น มนุษย์สามารถจินตนาการได้ว่าควรจะเป็นแบบใดซึ่งต่างจากคอมพิวเตอร์ที่ไม่สามารถจินตนาการได้ จากหลักการของ GMM เราสามารถทำให้คอมพิวเตอร์มีความสามารถในการค้นหาภาพที่ใกล้เคียงของกับระบบการตัดสินใจของมนุษย์

3.1 ขั้นตอนเริ่มแรกของระบบ

ภาพทุกภาพในฐานข้อมูลจะถูกแทนด้วย Feature descriptor ซึ่งจะอยู่ในรูปของเวคเตอร์ (vector) ที่สมาชิกประกอบขึ้นจากค่าจากการคำนวณ ของ feature ต่างๆ แสดงได้ดังสมการ

$$
\begin{array}{l}
F_1=\{ f_{11}, f_{12}, f_{13}, \ldots, f_{1N} \} \\
F_2=\{ f_{21}, f_{22}, f_{23}, \ldots, f_{2N} \} \\
F_3=\{ f_{31}, f_{32}, f_{33}, \ldots, f_{3N} \} \\
\vdots \quad \vdots \quad \vdots \\
\vdots \quad \vdots \quad \vdots \\
F_M=\{ f_{M2}, f_{M2}, f_{M3}, \ldots, f_{MN} \}
\end{array}
\quad \Rightarrow \quad
\begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_M \end{pmatrix} =
\begin{pmatrix}
f_{11} & f_{12} & f_{13} & \cdots & f_{1N} \\
f_{21} & f_{22} & f_{23} & \cdots & f_{2N} \\
f_{31} & f_{32} & f_{33} & \cdots & f_{3N} \\
\vdots & \vdots & \vdots & & \\
\vdots & \vdots & \vdots & & \\
f_{M1} & f_{M2} & f_{M3} & \cdots & f_{MN}
\end{pmatrix}
$$

M (↓) N (→)

โดย

F_M = feature vector ของภาพที่ M ในฐานข้อมูล

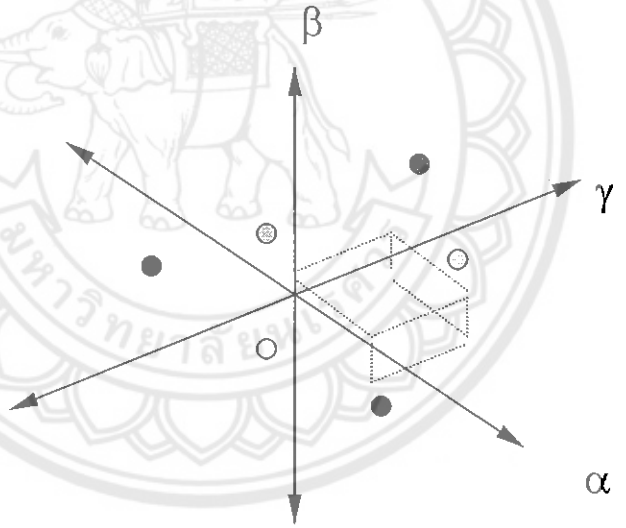
N = จำนวน feature ที่พิจารณา

ซึ่งในฐานข้อมูลหนึ่งๆ อาจเก็บภาพเป็นจำนวนมาก เช่น 100,000 ภาพ เป็นต้น ดังนั้น Feature descriptor ของภาพก็จะมี 100,000 ตัวเช่นกัน

$$F_1 , F_2 , F_3 , \dots , F_{100} , F_{200} , F_{300} , \dots , F_{1000} , F_{2000} , F_{3000} , \dots , F_{10000}$$

3.2 ปริภูมิเวคเตอร์ (Vector space) ของ feature vector

เพื่อความเข้าใจที่ง่ายขึ้นเราอาจมองว่าภาพใดๆก็คือจุดหนึ่งจุดบนปริภูมิเวคเตอร์ (Vector space) ซึ่งจำนวนแกนอ้างอิงของปริภูมิเวคเตอร์ จะมีจำนวนเท่ากับ Feature ที่เราสนใจพิจารณาจากภาพของฐานข้อมูล และพิกัดของจุด (รูปภาพ) จะเป็นการนำค่าของ feature vector มากำหนดตำแหน่งในปริภูมิเวคเตอร์ ดังภาพ



$F(\alpha , \gamma , \beta)$

$F(x_1 , y_1 , z_1)$

$F(x_2 , y_2 , z_2)$

:

รูปที่ 3.1 แสดง Feature Vector Space 3 มิติ

3.3 การค้นหาแบบง่าย (Simple search)

เมื่อผู้ใช้ต้องการจะค้นหาภาพ ผู้ใช้ก็จะต้องหารูปภาพที่จะใช้เป็นต้นแบบ (Query image) จากฐานข้อมูลภาพซึ่งได้ทำ Feature description ไว้แล้ว มาเป็นภาพต้นแบบในการค้นหาจำนวน 1 ภาพ ซึ่งสมมติว่าแทนด้วย Q ส่งให้กับระบบ ซึ่งระบบจะนำ Feature vector ของภาพ Q (F_Q)ไปทำการคำนวณหาค่าความแตกต่าง (Distance measure) ของภาพแต่ละภาพในฐานข้อมูลทั้งหมด โดยค่าความแตกต่างนิยามได้ดังนี้ [3]

$$D_M = \left[\sum_{i=1}^{N}(f_{iQ} - f_{iX})\right]^{1/2} \qquad (3.1)$$

สมการแสดงค่าความแตกต่างระหว่าง ภาพต้นแบบ Q กับภาพ X ใดๆ

$$D_1 = \left[\sum_{i=1}^{N}(f_{1Q} - f_{1X})\right]^{1/2}$$

$$D_2 = \left[\sum_{i=1}^{N}(f_{2Q} - f_{2X})\right]^{1/2}$$

$$D_M = \left[\sum_{i=1}^{N}(f_{MQ} - f_{MX})\right]^{1/2}$$

รูปที่ 3.2 แสดงการหา Distance ของเวคเตอร์ตัวอย่างในหัวข้อที่ 1

ซึ่งหากจะมองให้ง่ายขึ้น หากเราเปรียบภาพ 2 ภาพใด ๆ เป็นจุด 2 จุดในปริภูมิเวคเตอร์ (Vector space) ขั้นตอนนี้ก็คือขั้นตอนการหาระยะขจัด (Distance) ระหว่างจุดทั้งสองนั่นเอง โดยเมื่อเสร็จสิ้นขั้นตอนนี้แล้ว เราก็จะได้ภาพที่ถูกจัดลำดับจากค่าที่ D น้อยไปหาค่า D มากจำนวน K ภาพ ซึ่งจะถูกนำแสดงออกเป็นผลลัพธ์ให้กับผู้ใช้

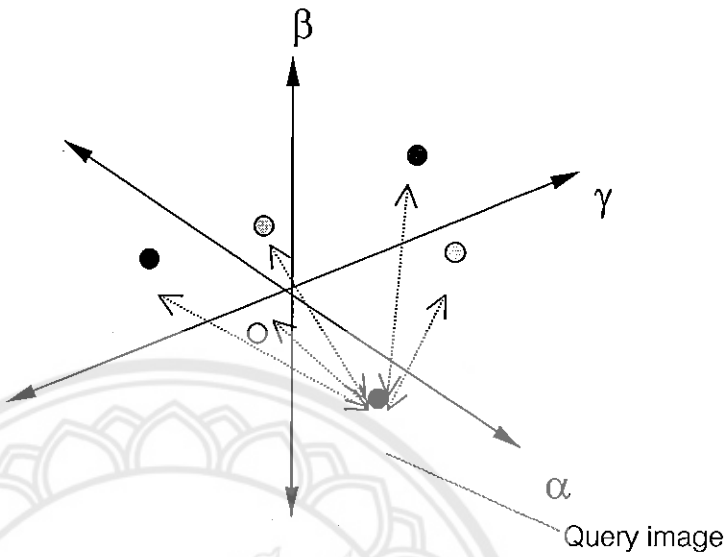
O = เซตของค่าความแตกต่างที่เป็นผลลัพธ์ของ Simple search

$O = \{ D_A , D_B , D_C , \dots , D_K \}$

โดย $D_A <= D_B <= D_C <= , \dots , <= D_K$

ดังนั้น F_O = เซตของภาพที่เป็นผลลัพธ์ของ Simple search

$F_O = \{ F_A , F_B , F_C , \dots , F_K \}$



รูปที่ 3.3 แสดงการหาค่าความแตกต่าง (Distance measure) บนปริภูมิเวคเตอร์ 3 มิติ

3.4 กระบวนการ Relevance feedback โดยใช้ Gaussian Mixture Model (GMM)

เมื่อมีการแสดงภาพซึ่งเป็นผลลัพธ์จากหัวข้อที่แล้ว คือ $F_O = \{ F_A , F_B , F_C , \dots , F_K \}$ ให้กับผู้ใช้แล้วถ้าหากว่ามีภาพที่ผู้ใช้ ต้องการก็ถือว่าการค้นหาประสบผลแล้ว แต่หากว่าในจำนวนนั้นยังไม่มีภาพที่ผู้ใช้ต้องการ ผู้ใช้ก็จะต้องทำการเลือกภาพที่มีลักษณะใกล้เคียงกับความต้องการแล้วส่งกลับให้ระบบ (positive feedback) ระบบก็จะทำการคำนวณหาครั้งใหม่ ด้วยกระบวนการ Gaussian Mixture Model โดยใช้ภาพที่ผู้ใช้เลือกกลับมา (positive feedback) ทุกๆ ภาพเป็นต้นแบบในการค้นหา เช่น กำหนดให้ B_O เป็นเซตของเวคเตอร์ของภาพที่ผู้ใช้ เลือกมาจำนวน T ภาพ

Relevant (B_O) = $\{ B_1 , B_2 , B_3 , \dots , B_T \}$

ค่า Relevant ที่ได้นี้ จะถูกนำมาใช้เป็น Center ของ GMM ดังนั้นจะได้ว่า

$Z_1 = B_1$

$Z_2 = B_2$

$Z_3 = B_3$

$\downarrow$

$Z_T = B_T$

เมื่อ Z คือ Center ของ GMM

จากสมการ Gaussian

$$G_m(x, z, \sigma_m) = \exp\left[\frac{\|x - z\|^2}{2\sigma_m^2}\right] \quad (3.2)$$

โดย σ_m = smouth parameter

และ $\|x - z\| = \left[\sum_{i=1}^{N}(x_i - z_j)^2\right]^{1/2}$

ดังนั้นจะได้ว่า

$$G_m(x, z, \sigma_m) = \exp\left[-\frac{\left[\sqrt{\sum_{i=1}^{N}(x_i - z_j)}\right]^2}{2\sigma_m^2}\right] \quad (3.3)$$

ทำการค้นหาอีกครั้งจะได้

$$\bar{D}_A = G_{mA1} + G_{mA2} + G_{mA3} + \ldots + G_{mAT} = \sum_{k=1}^{T} G_k \quad (3.4)$$

A = 1,2,3,...,M (รูปภาพทั้งหมดมี M รูป)

T = 1,2,3,...,T (Center of Gaussian)

$$G_{mA1} = \exp\left[-\frac{\left[\sqrt{\sum_{i=1}^{N}(x_{1i} - z_i)^2}\right]^2}{2\sigma_1^2}\right]$$

$$\downarrow$$

$$G_{mAT} = \exp\left[-\frac{\left[\sqrt{\sum_{i=1}^{N}(x_{2i} - z_T)^2}\right]^2}{2\sigma_T^2}\right]$$

จาก Center ที่ได้มาจากผู้ใช้จะได้ว่า

$$\bar{D}_A = G_{mA1} + G_{mA2} + G_{mA3} + \ldots + G_{mAT}$$

ทำการหาค่า $\bar{D}_A$ ไปเรื่อย ๆ จนครบทุก Image ในฐานข้อมูล จากนั้นนำค่า $\bar{D}_A$ ที่ได้มาเปรียบเทียบกัน แล้วเรียงลำดับจากมากไปหาน้อย เช่น จากการคำนวณแล้วได้ผลลัพธ์ดังนี้

$$\bar{D}_3 > \bar{D}_{80} > \bar{D}_{15} > \bar{D}_{50} > \ldots\ldots > \bar{D}_8 > \bar{D}_1$$

นำ Image ที่มีค่า D มากที่สุดแสดงให้ผู้ใช้ จากตัวอย่างถ้าต้องการแสดงภาพ 3 ภาพ จะได้ Image3,Image80 และ Image15 แสดงให้กับผู้ใช้ เมื่อผู้ใช้พบภาพที่ต้องการแล้วก็เป็นอันสิ้นสุดการค้นหา แต่หากว่าผู้ใช้ยังไม่พบภาพที่ต้องการผู้ใช้จะทำการเลือกภาพ และป้อนค่ากลับให้กับระบบอีกครั้ง ระบบจะทำการคำนวณและแสดงผลวนซ้ำไปอย่างนี้จนกว่าผู้ใช้จะยืนยันว่าพบภาพที่ต้องการแล้ว

นอกจากวิธีการดังกล่าวแล้ว ยังสามารถใช้วิธีการอื่นในการปรับปรุงผลลัพธ์ของการค้นหาได้ วิธีการดังกล่าวคือ การเลื่อน Center ที่ได้มาจาก feedback ของผู้ใช้ วิธีการนี้จะเป็นการเลื่อน กลุ่มของ Center ของภาพที่ผู้ใช้ป้อนกลับให้ห่างออกจาก กลุ่มของภาพที่ผู้ใช้ไม่ต้องการดังนี้

$$v_i(t+1) = v_i - \eta(t)\left[x_{ii}''(t) - v_i(t)\right] \qquad (3.5)$$

เมื่อ v_i คือ ภาพที่ผู้ใช้ต้องการ(relevance image) ที่ผู้ใช้ป้อนกลับ

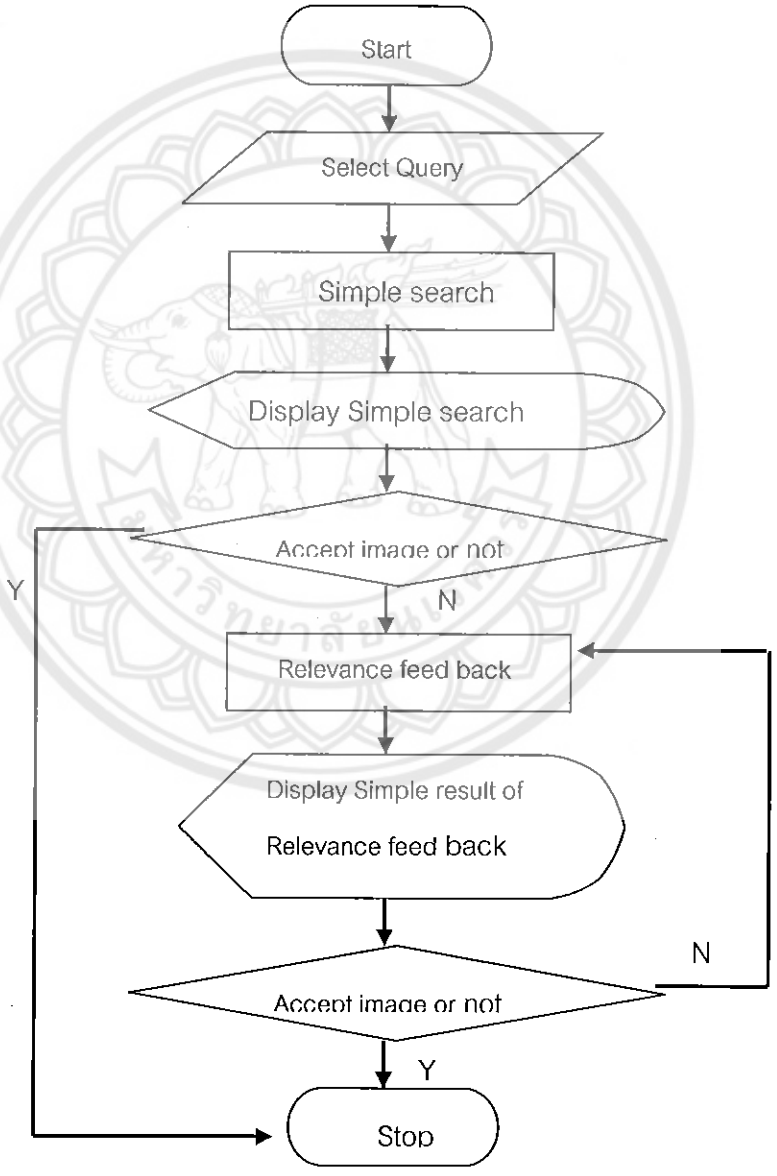
t คือ จำนวนครั้งของการป้อนกลับ

η คือ ค่า learning constant

x_n^- คือ ภาพที่ผู้ใช้ไม่ต้องการ (non-relevance image)

ค่า η ที่ใช้จะมีค่าอยู่ในช่วง 0.1 ถึง 0.8 และจะลดลงเป็นสัดส่วนที่แน่นอนกับจำนวนครั้งของการป้อนกลับ

เมื่อทำการเลื่อน Center แล้ว จะทำให้ได้ Center ใหม่ซึ่งสามารถใช้ GMM คำนวณหาผลลัพธ์ได้เหมือนกับวิธีที่ใช้เพียง Positive feedback อย่างเดียว

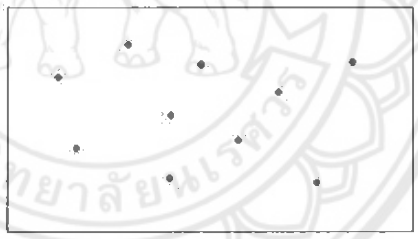


รูปที่ 3.4 แสดง Program flow

3.5 ตัวอย่างการคำนวณ

สมมุติให้ในฐานข้อมูลมี Image อยู่10 ภาพ และถูก Index ด้วยเวคเตอร์ที่ได้แสดงไว้ และมี Query Image คือ $X_q = [f_{q1}, f_{q2}] = [0.30\ \ 0.90]$

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \\ X_9 \\ X_{10} \end{bmatrix} = \begin{bmatrix} 0.50 & 0.70 \\ 0.50 & 0.10 \\ 0.40 & 0.20 \\ 0.70 & 0.10 \\ 0.10 & 0.80 \\ 0.90 & 0.90 \\ 0.10 & 0.60 \\ 0.60 & 0.20 \\ 0.20 & 0.40 \\ 0.50 & 0.40 \end{bmatrix}$$



รูปที่ 3.5 แสดงการกระจายตัวของ Image ในฐานข้อมูล

Simple Search

จากสมการ (3.1)

$D_1 = [(0.30-0.50)^2 + (0.90-0.70)^2]^{1/2} = [(0.04+0.04)]^{1/2} = 0.28$

$D_2 = [(0.30-0.50)^2 + (0.90-0.10)^2]^{1/2} = [(0.04+0.64)]^{1/2} = 0.82$

$D_3 = [(0.30-0.40)^2 + (0.90-0.70)^2]^{1/2} = [(0.01+0.49)]^{1/2} = 0.71$

$D_4 = [(0.30-0.70)^2 + (0.90-0.10)^2]^{1/2} = [(0.16+0.64)]^{1/2} = 0.89$

$D_5 = [(0.30-0.10)^2 + (0.90-0.80)^2]^{1/2} = [(0.04+0.01)]^{1/2} = 0.22$

$D_6 = [(0.30-0.90)^2 + (0.90-0.90)^2]^{1/2} = [(0.36+0.00)]^{1/2} = 0.60$

$D_7 = [(0.30-0.10)^2 + (0.90-0.60)^2]^{1/2} = [(0.04+0.09)]^{1/2} = 0.36$

$D_8 = [(0.30-0.60)^2 + (0.90-0.20)^2]^{1/2} = [(0.09+0.49)]^{1/2} = 0.76$

$D_9 = [(0.30-0.20)^2 + (0.90-0.40)^2]^{1/2} = [(0.01+0.25)]^{1/2} = 0.51$

$D_{10} = [(0.30-0.50)^2 + (0.90-0.40)^2]^{1/2} = [(0.04+0.25)]^{1/2} = 0.54$

เมื่อทำการคำนวณเปรียบเทียบค่าความแตกต่างของแต่ละภาพกับ Query Image แล้วจะทำการแสดงผลลัพธ์ให้กับผู้ใช้ โดยเรียงลำดับจาก Image ที่มีค่าความแตกต่างน้อยที่สุดไปหามากที่สุด (ในที่นี้แสดง 4 ภาพ)

$$D_5 < D_1 < D_7 < D_9 < D_{10} < D_6 < D_3 < D_8 < D_2 < D_4$$

Image 5 , Image 1 , Image 7 , Image 9

จากสมการ (3.3) ถ้าผู้ใช้ Feedback กลับมาว่าเลือก Image5 และ Image7 จะได้ว่า

$X_q = [0.30\ 0.90]$

$Z1 = [0.10\ 0.80] = X5$

$Z2 = [0.10\ 0.60] = X7$

จากสมการ (3.4) ให้และให้ $\sigma = 0.50$

หาค่า D1

$$G1=\exp[-[(0.50-0.10)^2+(0.70-0.80)^2]/0.50] = 0.35$$

$$G2=\exp[-[(0.50-0.10)^2+(0.70-0.60)^2]/0.50] = 0.71$$

$$D1=G1+G2$$

$$D1=1.42$$

หาค่า D2

$$G1=\exp[-[(0.50-0.10)^2+(0.10-0.80)^2]/0.50] = 0.27$$

$$G2=\exp[-[(0.50-0.10)^2+(0.10-0.60)^2]/0.50] = 0.44$$

$$D2=G1+G2$$

$$D2=0.71$$

หาค่า D3

$$G1=\exp[-[(0.04-0.10)^2+(0.20-0.80)^2]/0.50] = 0.41$$

$$G2=\exp[-[(0.04-0.10)^2+(0.20-0.60)^2]/0.50] = 0.61$$

$$D3=G1+G2$$

$$D3=1.02$$

หาค่า D4

$$G1=\exp[-[(0.70-0.10)^2+(0.10-0.80)^2]/0.50] = 0.18$$

$$G2=\exp[-[(0.70-0.10)^2+(0.10-0.60)^2]/0.50] = 0.30$$

$$D4=G1+G2$$

$$D4=0.48$$

หาค่า D5

$$G1=\exp[-[(0.10-0.10)^2+(0.80-0.80)^2]/0.50] = 1.00$$

$$G2=\exp[-[(0.10-0.10)^2+(0.80-0.60)^2]/0.50] = 0.92$$

$$D5=G1+G2$$

$$D5=1.92$$

หาค่า D6

$$G1=\exp[-[(0.90-0.10)^2+(0.90-0.80)^2]/0.50] = 0.27$$

$$G2=\exp[-[(0.90-0.10)^2+(0.90-0.60)^2]/0.50] = 0.23$$

$$D6=G1+G2$$

$$D6=0.50$$

หาค่า D7

$$G1=\exp[-[(0.10-0.10)^2+(0.60-0.80)^2]/0.50] = 0.92$$

$$G2=\exp[-[(0.10-0.10)^2+(0.60-0.60)^2]/0.50] = 1.00$$

$$D7=G1+G2$$

$$D7=1.92$$

หาค่า D8

$$G1=\exp[-[(0.60-0.10)^2+(0.20-0.80)^2]/0.50] = 0.30$$

$$G2=\exp[-[(0.60-0.10)^2+(0.20-0.60)^2]/0.50] = 0.44$$

$$D8=G1+G2$$

$$D8=0.74$$

หาค่า D9

$$G1=\exp[-[(0.20-0.10)^2+(0.40-0.80)^2]/0.50] = 0.71$$

$$G2=\exp[-[(0.20-0.10)^2+(0.40-0.60)^2]/0.50] = 0.90$$

$$D9=G1+G2$$

$$D9=1.61$$

หาค่า D10

$$G1=\exp[-[(0.50-0.10)^2+(0.40-0.80)^2]/0.50] = 0.53$$

$$G2=\exp[-[(0.50-0.10)^2+(0.40-0.60)^2/0.50] = 0.67$$

$$D10=G1+G2$$

$$D10=1.20$$

จากการคำนวณดังได้แสดงไว้แล้วจะสามารถนำค่า D มาจัดเรียงกันจากมากไปหาน้อยได้ดังนี้

$$D_5 > D_7 > D_9 > D_1 > D_{10} > D_3 > D_8 > D_2 > D_6 > D_4$$

ซึ่งสามารถแสดงผลลัพธ์ให้กับ User ได้ดังนี้ (แสดง 4 ภาพ) ดังนี้

Image5 , Image7, Image9 , Image1

เมื่อผู้ใช้ทำการเลือกภาพที่ต้องการแล้วผู้ใช้จะป้อนค่ากลับให้กับระบบอีกครั้งถ้าต้องการหาใหม่ และระบบจะทำการค้นหาแบบเดิมอีกรอบหนึ่ง การทำงานจะเป็นรอบวนซ้ำไปแบบนี้จนกว่าผู้ใช้จะพบภาพที่ต้องการ การค้นหาจึงจะเสร็จสิ้น

3.6 ขั้นตอนในการออกแบบการทำงานของโปรแกรม

ในการเขียนโปรแกรมภาษา Java เป็นภาษาโปรแกรมเชิงวัตถุ (Object Oriented Program :OOP) ซึ่งมีคุณสมบัติที่เอื้อต่อการเขียนโปรแกรมให้ง่ายขึ้น สามารถนำแต่ละส่วนของโปรแกรมย่อยมาประกอบกันขึ้นเป็นโปรแกรมใหญ่ ดังนั้นโครงงานนี้จึงใช้คุณสมบัติของภาษาในเชิงวัตถุให้เกิด ในโปรแกรมมีการใช้ Class หลักหนึ่ง Class เพื่อทำงานต่าง ๆ ทั้งหมด โดยมี method ย่อยหลาย methods

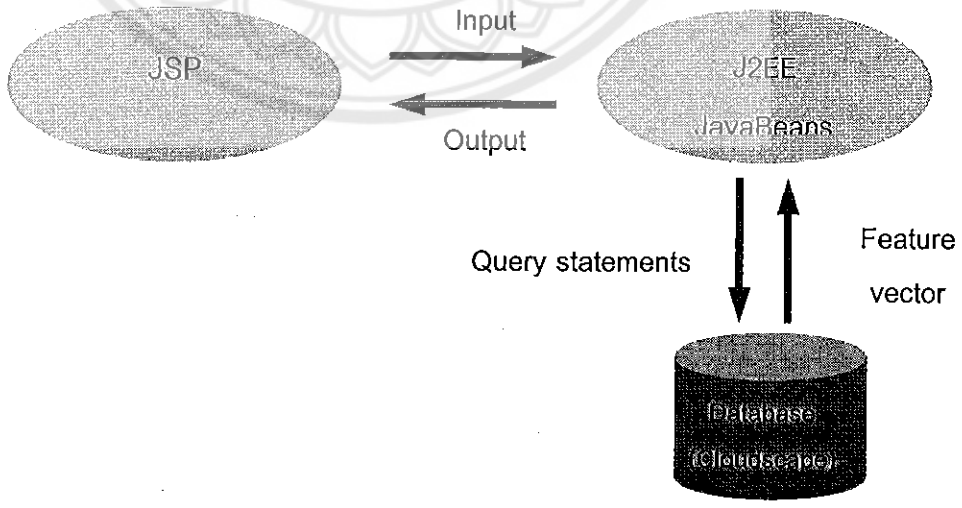
ในการออกแบบการทำงานของโปรแกรมนั้นจะแบ่งออกเป็น 2 ส่วนคือ ส่วนประมวลผลและส่วนแสดงผล

3.6.1 ส่วนประมวลผล

ส่วนประมวลผลทำหน้าที่ประมวลผลลัพธ์ทั้งหมด ทำหน้าที่ติดต่อกับฐานข้อมูล ,คำนวณค่า distance ,คำนวณค่า Gaussian distance หรือ คำนวณค่า Gaussian value เป็นต้น การทำงานของส่วนประมวลผลจะทำงานบนฝั่งของ server ทั้งหมด หลังทำการคำนวณเปรียบเทียบแล้วจะส่งผลลัพธ์ที่ได้ไปให้กับส่วนแสดงผลต่อไป การทำงานในส่วนนี้ใช้ Java Beans เป็นตัวควบคุมการทำงาน

3.6.2 ส่วนแสดงผล

ส่วนแสดงผลทำหน้าที่แสดงผลลัพธ์ที่ได้จากส่วนประมวลผลให้ผู้ใช้ การทำงานในส่วนแสดงผลจะทำงานโดยมี JSP เป็นตัวควบคุมการแสดงผล โดยรับค่าตัวแปรต่าง ๆ จากผู้ใช้,แสดงภาพที่ได้จากการประมวลผล การทำงานหลักในส่วนนี้คือ ทำหน้าที่ติดต่อระหว่าง ผู้ใช้กับระบบค้นหาภาพ การทำงานทั้งสองส่วนสามารถแสดงได้ด้วยรูปที่ 3.6



รูปที่ 3.6 แสดงการออกแบบการทำงานของโปรแกรม

บทที่ 4

การทดลองและผลการทดลอง

ในการทดสอบการทำงานของโปรแกรมนั้นจะแบ่งการทดสอบออกเป็นสองส่วนใหญ่ๆ คือ ในส่วน Text mode และ ในส่วนของ Graphic mode ซึ่งในส่วน Text mode นั้น เพื่อตรวจสอบว่าการทำงานของ method ที่สร้างขึ้นนั้นทำงานได้ถูกต้องหรือไม่ ส่วนการทดสอบในส่วน Graphic mode นั้นก็เพื่อตรวจสอบว่าโปรแกรมสามารถค้นหาภาพดิจิตอลในลักษณะที่ผู้ใช้ต้องการได้ถูกต้องมากน้อยเพียงใด ในส่วน Graphic mode นั้นแบ่งการทดลองออกเป็นสองส่วน คือส่วนแรกคือส่วนที่ใช้วิธี Positive feedback และส่วนที่สองคือส่วนที่ใช้ทั้ง Positive feedback และ Negative feedback

การทำงานในส่วนของText mode จะใช้โค้ดของโปรแกรมที่สร้างขึ้นเพื่อทดสอบหน้าที่การทำงานของแต่ละฟังก์ชันการทำงานว่าเป็นไปตามทฤษฎีหรือไม่ มีข้อผิดพลาดอะไรเกิดขึ้น และเกิดปัญหาอะไรบ้างที่ทำให้การทำงานไม่ประสบความสำเร็จ รวมถึงการแก้ไขข้อผิดพลาดที่เกิดขึ้นด้วย ในส่วนของการทดสอบใน Graphic mode จะใช้โค้ดโปรแกรม[9] อีกชุดหนึ่งซึ่งจะเป็นส่วนที่ใช้ติดต่อกับผู้ใช้ โดยรวมแล้วจะใช้โค้ดที่ได้จาก Text mode และเพิ่มโค้ดในส่วนติดต่อกับผู้ใช้เข้ามา ดังนั้นจุดประสงค์ของการทดสอบในแบบ Graphic mode คือเพื่อดูว่าหน้าตาของโปรแกรมเป็นอย่างไร มีความเหมาะสมหรือไม่ สาเหตุที่เน้นที่รูปแบบที่เหมาะสมมากกว่าเรื่องความถูกต้องเพราะความถูกต้องต่าง ๆ ได้ถูกทดสอบไปในขั้นตอนการทดสอบ Text mode แล้วนั่นเอง

ฐานข้อมูลที่ใช้ทดสอบคือ Cloudscape ซึ่งเก็บค่าของ feature vector ของภาพทั้งหมดเอาไว้ รวมทั้งเก็บชื่อของแต่ละภาพเอาไว้ด้วย แต่ละภาพจะมี 115 features ภาพทั้งหมดที่ใช้ในการทดสอบมีจำนวน 34,000 ภาพการรันโปรแกรมจะใช้ J2EE เป็นเซิร์ฟเวอร์จำลองเพื่อให้ระบบทำงานได้ เพราะ JSP จะสามารถทำงานได้เฉพาะบนเครื่องเซิร์ฟเวอร์เท่านั้น

การทดลองเริ่มต้นด้วยการ รันโปรแกรมใน Text mode ก่อนเริ่มต้นด้วยการทดสอบ method ทั้งหมด ทดสอบการคำนวณหาผลลัพธ์โดยกำหนดเซตของข้อมูลที่เหมาะสมขึ้นมาทดสอบว่าผลลัพธ์ที่ได้ถูกต้องหรือไม่ ส่วนการทดสอบใน graphic mode นั้นตอนแรกโปรแกรมจะทำการติดต่อกับฐานข้อมูล ผ่านทาง connector จากนั้นจะวนรอบเพื่อเก็บค่าข้อมูล feature vector ของภาพทุกภาพมาไว้ รวมทั้งเก็บชื่อของภาพทั้งหมดไว้ด้วย หลังจากนั้นเมื่อเก็บข้อมูลที่จำเป็นครบแล้วจะปิดการติดต่อกับฐานข้อมูล และทำการแสดงผลภาพที่มีอยู่ให้ผู้ใช้เลือก หลังจากนั้นจะเป็นการติดต่อกันระหว่างผู้ใช้กับคอมพิวเตอร์เพื่อคำนวณผลลัพธ์จาก feedback ที่ผู้ใช้ป้อนกลับ การทำสอบ

ในช่วงนี้จะเหมือนกับการทำงานจริงทุกอย่าง แตกต่างเพียงการทดสอบไม่ได้ทำบนเครื่อง server เท่านั้น

4.1 ผลการทดลอง

4.1.1 ผลการทดลองใน Text mode

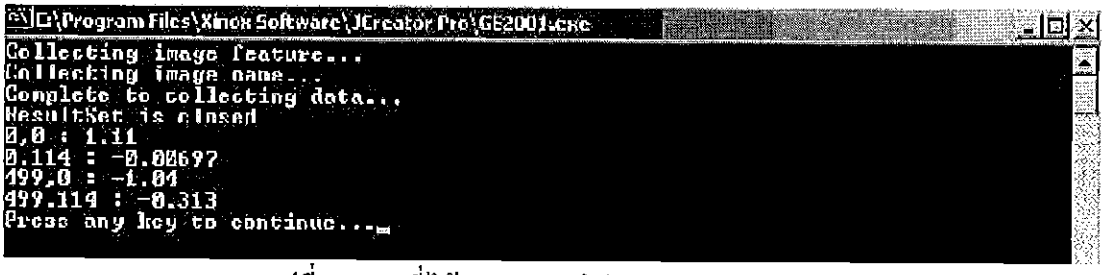
การทดลอง การหาค่า Distance

ในขั้นตอนนี้เป็นการทดสอบโค้ดโปรแกรม [10] ในส่วนของการหาค่า Distance ของ feature ของภาพสองภาพโดยทำการคำนวณเพียง 3 feature เท่านั้นเพื่อเป็นการประหยัดเวลา และไม่จำเป็นต้องทดสอบหลายค่า ผลการทดลองที่ปรากฏอยู่ในรูปที่ 4.1 เป็นการคำนวณหาค่าผลต่างกำลังสองของค่า {0.1 , 0.2 , 0.3} และ {0.5 , 0.6 , 0.8} จากรูปที่ 4.1 เป็นการหาค่า Distance ของภาพโดยใช้วิธี Simple search คือ นำ feature vector ของภาพ 2 ภาพที่เป็น feature ชนิดเดียวกันมาหาค่า euclidean distance

รูปที่ 4.1 ผลที่ได้จากการรันโปรแกรม Distance

การทดลอง การติดต่อกับฐานข้อมูล

โค้ดโปรแกรม[11] ในการทดลองนี้ เป็นการเรียกดู feature vector ในตำแหน่งต่าง ๆ ของฐานข้อมูล ผลที่ได้อันดับแรกจะได้จากการโหลดฐานข้อมูลในลำดับที่ 0 ตรงตำแหน่ง feature vector ที่ 0 และ 114 ส่วนถัดมาได้จากการโหลด ฐานข้อมูลในลำดับที่ 499 ตรงตำแหน่ง feature vector ที่ 0 และ 114 เช่นกัน การทดลองในส่วนนี้เพื่อเป็นการทดสอบการติดต่อกับฐานข้อมูลว่าเกิดปัญหาอะไรหรือไม่ เกิดความล่าช้าหรือไม่



รูปที่ 4.2 ผลที่ได้จากการรันโปรแกรม TestDatabase

การทดลอง การหาค่า Distance จากฐานข้อมูล

เป็นการหาค่า Distance ของภาพโดยใช้วิธี Simple search คือ นำ feature vector ของภาพ 2 ภาพที่เป็น feature ชนิดเดียวกันมาหาค่า euclidean distance การคำนวณในโค้ดส่วนนี้[12] แตกต่างจากในรูปที่ 4.1 ตรงข้อมูลที่ใช้ในการคำนวณของรูปนี้เป็นการดึงข้อมูลจริงจากฐานข้อมูล ส่วนในรูปที่4.1เป็นข้อมูลที่สมมุติขึ้นเพื่อตรวจสอบความถูกต้องของการคำนวณนั่นเอง

รูปที่ 4.3 ผลที่ได้จากการรันโปรแกรม TestDistance

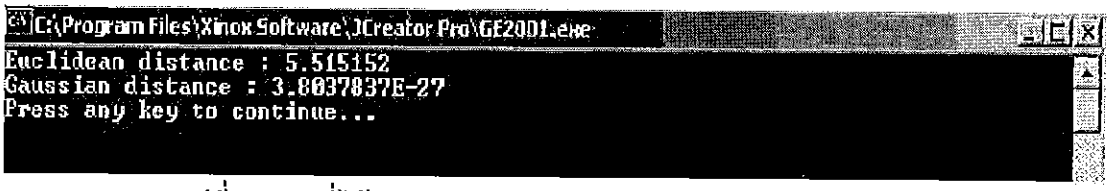
การทดลอง การหาค่า Sigma

จากโค้ดของโปรแกรม TestFindSigma[13] เป็นการหาค่า Sigma โดยค่า Sigma หาได้จากการนำเอาค่า Euclidean Distance ที่น้อยที่สุดในกลุ่มหารด้วย 2 จากตัวอย่างมี feature vector ของภาพ 6 ภาพ ทำการหาค่า Euclidean Distance ของทั้ง 6 ภาพ แล้วนำค่า Euclidean Distance ของแต่ละภาพมาหารด้วย 2 จะได้ค่า Sigma ของแต่ละภาพ

รูปที่ 4.4 ผลที่ได้จากการรันโปรแกรม TestFindSigma

การทดลอง การหาค่า Guassian Distance

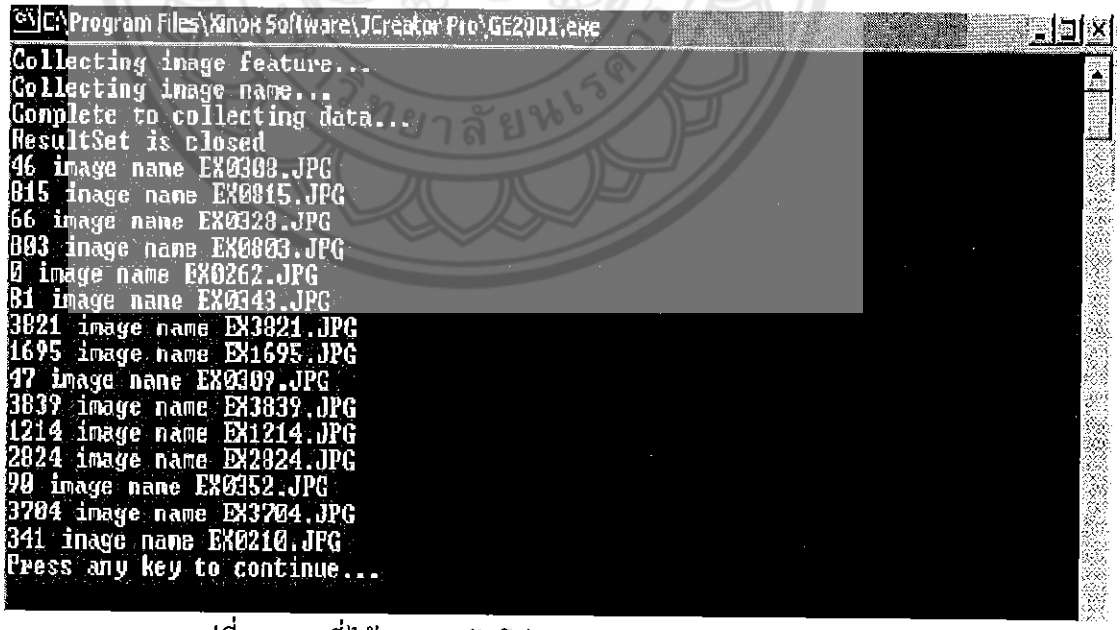
การหาค่า Gaussian Distance ในโค้ดโปรแกรม[14] ที่ทำการทดลองจะใช้ภาพ 2 ภาพ โดยในขั้นตอนแรกทำการหาค่า Euclidean Distance ก่อน จากนั้นนำค่าที่ได้ไปทำการคำนวณหาค่า Gaussian Distance



รูปที่ 4.5 ผลที่ได้จากการรันโปรแกรม TestGaussianDistance

การทดลอง Guassian Search

รูปที่ 4.6 เป็นการหาค่าของตำแหน่งภาพที่มีค่า Gaussian Distance ใกล้เคียงกับค่า feedback ที่ป้อนกลับเข้าไป (ภาพที่อยู่ในตำแหน่งที่ 46 , 66 , 803 และ 815) ในตัวอย่างจะมีการแสดงผลลัพธ์จำนวน 15 ภาพซึ่งถูกกำหนดไว้ในโค้ดโปรแกรม[15] โดยเรียงจากค่าที่มีความใกล้เคียงมากสุดไปหาน้อยสุด ในการคำนวณหาผลลัพธ์นอกจากจะได้ตำแหน่งของภาพแล้ว ยังแสดงชื่อของภาพแต่ละภาพด้วย



รูปที่ 4.6 ผลที่ได้จากการรันโปรแกรม TestGaussianSearch

การทดลอง การเรียงลำดับข้อมูลจากน้อยไปมาก

ในการคำนวณนั้นจะได้ผลลัพธ์ออกมาเป็นชุดของข้อมูลจำนวนหนึ่งซึ่งจำเป็นที่จะต้องนำไปจัดเรียงลำดับก่อนที่จะนำไปใช้งานในขั้นตอนต่อไปแล้วแต่ความต้องการ โค้ดโปรแกรม[16] ในส่วนนี้ เป็นการเรียงลำดับข้อมูลจากข้อมูลที่มีค่าน้อยสุดไปหาค่าที่มากสุด ตัวเลขที่แสดงนั้นหมายถึงตำแหน่งของข้อมูลว่าอยู่ตำแหน่งหรือลำดับที่เท่าใดในฐานข้อมูล

รูปที่ 4.7 ผลที่ได้จากการรันโปรแกรม TestLeastSorting

การทดลอง การเรียงลำดับข้อมูลจากมากไปน้อย

รูปที่ 4.8 เป็นการเรียงลำดับข้อมูลจากค่ามากสุดไปน้อยสุดโดยโค้ดโปรแกรม[17] จะแตกต่างจากการเรียงข้อมูลจากน้อยไปมากอยู่เล็กน้อย การทดลองนี้แสดงผลลัพธ์เป็นตำแหน่งของข้อมูล การทำงานจะคล้ายกับโค้ดโปรแกรมในรูปที่ 4.7

รูปที่ 4.8 ผลจากการรันโปรแกรม TestMaxSorting

การทดลอง Simple Search

โค้ดโปรแกรม[18] การหาค่า Simple Search ในที่นี้จะใช้ข้อมูลภาพจำนวนหนึ่ง โดยที่แต่ละภาพมี feature vector จำนวน 115 feature การคำนวณในขั้นตอนSimple search จะใช้ภาพในตำแหน่งที่ 0 เป็นภาพต้นแบบ(Query) แล้วทำการคำนวณหาค่า Euclidean Distance ของภาพต้นแบบกับภาพทั้งหมด จากนั้นจะแสดงค่า Distance ที่น้อยที่สุด 10 ค่าโดยบอกทั้งตำแหน่ง และชื่อของภาพ

รูปที่ 4.9 ผลที่ได้จากการรันโปรแกรม TestSimpleSearch

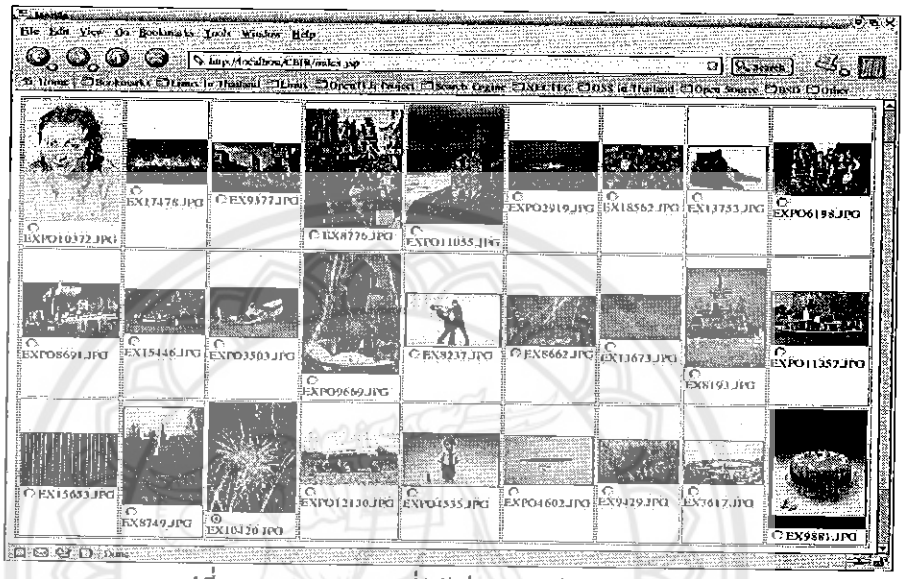
4.1.2 ผลการทดลองใน Graphic mode

การทดลองนี้มีจุดประสงค์เพื่อหาประสิทธิภาพของระบบค้นหาภาพว่ามีประสิทธิภาพมากน้อยเพียงใด ซึ่งภาพแต่ละภาพจะให้ผลลัพธ์ของการทำงานไม่เหมือนกัน ทั้งนี้ขึ้นอยู่กับรายละเอียดของแต่ละภาพว่ามีความยากง่ายในการค้นหาแค่ไหน การวัดประสิทธิภาพนั้นทำได้โดยหาค่า precision ซึ่งหาได้จากสมการดังต่อไปนี้

$$\text{Precision} = \frac{\text{จำนวนของภาพที่ถูกต้อง (Relevant image)}}{\text{จำนวนของ ผลลัพธ์ที่ได้ทั้งหมด}} \qquad (4.1)$$

ผลการทดลองดังต่อไปนี้เป็นการทดสอบใน Graphic mode โดยมีการหาค่าประสิทธิภาพในการค้นหาด้วยเพื่อเป็นเกณฑ์ในการอ้างอิงว่าการค้นหานั้นประสบผลสำเร็จมากน้อยเพียงใด ในการทดลองนี้แบ่งออกเป็นสองส่วนคือ ส่วนแรกทดสอบโปรแกรมโดยใช้เฉพาะ Positive feedback ในการคำนวณหาผลลัพธ์ และวิธีที่สองคือ ใช้ทั้ง Positive feedback และ Negative feedback

ในการคำนวณวิธี Positive feedback นี้จะใช้เฉพาะภาพที่ผู้ใช้ต้องการเท่านั้นในการปรับปรุง Query โดยภาพที่ผู้ใช้ไม่ต้องการนั้นจะไม่สนใจและละทิ้งไปโดยไม่นำมาคำนวณด้วย ดังนั้นผลการทดลองในรอบถัดไปจึงมีผลลัพธ์ที่มีลักษณะคล้ายภาพเดิมปรากฏออกมาด้วย เนื่องจากภาพเหล่านั้นมีค่า Gaussian value ใกล้เคียงกับ feedback ที่ผู้ใช้ป้อนกลับมากกว่าภาพอื่น ๆ ในฐานข้อมูลผลลัพธ์ที่ได้จึงมีลักษณะซ้ำไปซ้ำมาวนเวียนกันอยู่

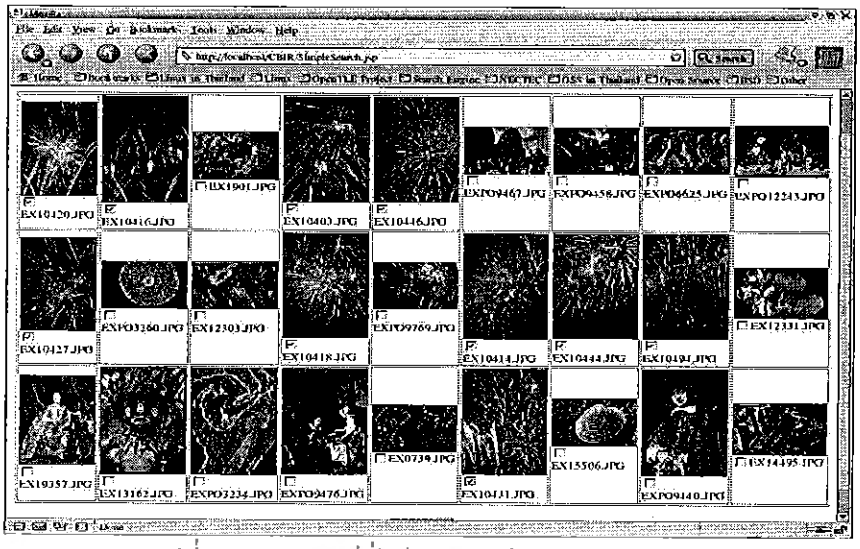


รูปที่ 4.10 แสดงภาพที่ใช้เป็นภาพต้นแบบ(Query)

รูปที่ 4.10 แสดงให้เห็นถึงภาพต้นแบบที่ใช้เพื่อสืบค้น ในที่นี้ใช้ภาพดอกไม้ไฟผลลัพธ์ครั้งแรกที่ได้นั้นจะมีภาพในลักษณะเดียวกันปรากฏให้เลือกดังแสดงในรูปที่ 4.11 ภาพเหล่านี้มีโทนสีที่คล้ายคลึงกัน คือ เป็นสีแดงและล้อมรอบด้วยสีดำ และผลลัพธ์ที่ต้องการมีจำนวน 10 ภาพ ดังนั้นจากสมาการที่ 4.1 สามารถคำนวณหาค่าประสิทธิภาพ(Precision) ได้ดังนี้

$$\text{Precision} = \frac{10}{27} \times 100$$

$$= 37.03\ \%$$

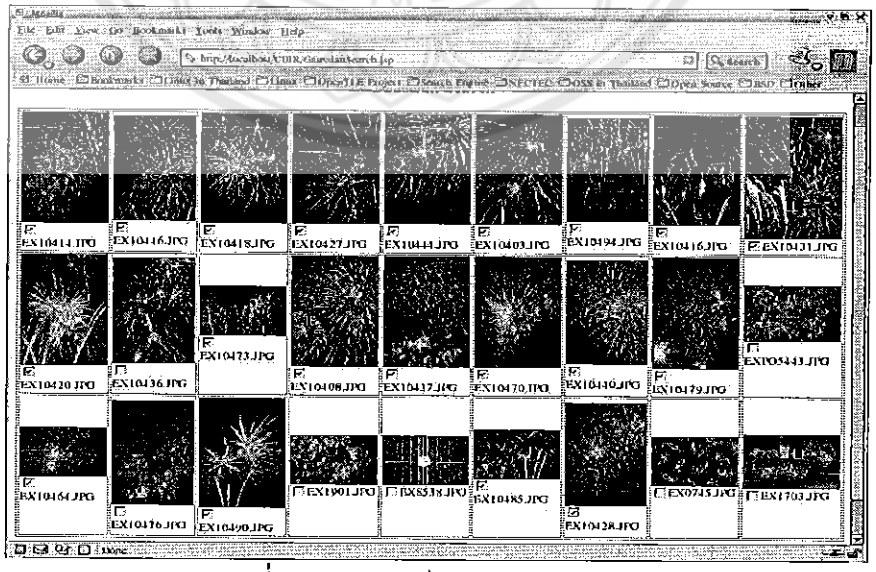


รูปที่ 4.11 ผลลัพธ์ที่ได้จากการค้นหาแบบ Simple search

ผลลัพธ์ที่ได้ครั้งที่ 2 ในรูปที่ 4.12 ได้มาจากการ Feedback ภาพกลับไปให้ระบบทำการค้นหาผลลัพธ์อีกครั้งหนึ่ง ภาพที่ใช้เป็น Feedback มีจำนวน10 ภาพ(Query) จากผลลัพธ์ที่ได้จะได้ค่าประสิทธิภาพคือ

$$\text{Precision} = \frac{10}{27} \times 100$$

$$= 37.03\ \%$$

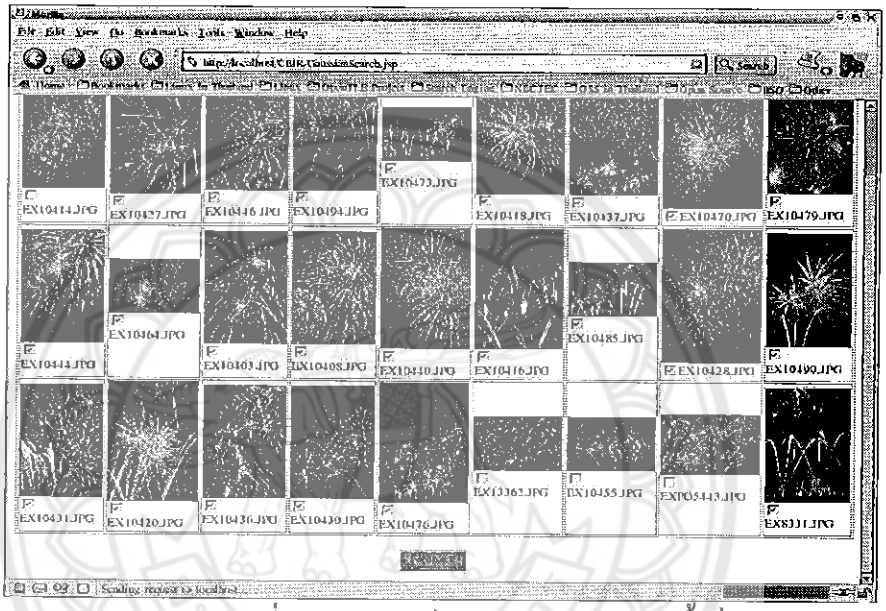


รูปที่ 4.12 ผลลัพธ์จากการ Feedback ภาพ 10 ภาพ

หลังจากทำการ Feedback ด้วยจำนวนภาพทั้งสิ้น 10 ภาพ แล้วทำการค้นหาอีกครั้งจะได้ผลลัพธ์ที่ดีขึ้นกว่าเดิมคือ มีภาพที่ต้องการปรากฏเพิ่มมากขึ้นจำ 10 ภาพเป็น ภาพ

$$\text{Precision} = \frac{22}{27} \times 100$$

$$= 81.48\ \%$$



รูปที่ 4.13 ผลลัพธ์จากการ Feedback ครั้งที่ 2

จากการทดลองข้างต้นจะสังเกตได้ว่า ผลลัพธ์ที่ได้มีความถูกต้องมากกว่าเมื่อเทียบกับขั้นตอน Simple search และค่าประสิทธิภาพที่วัดออกมาก็มีค่าที่สูง ทั้งนี้เพราะภาพที่ใช้สืบค้นเป็นภาพที่มีความง่ายที่จะค้นหา ภาพที่ง่ายคือภาพที่มีองค์ประกอบของภาพที่สามารถแบ่งแยกออกจากกันได้อย่างชัดเจน ไม่คลุมเครือ มีรูปร่างของวัตถุในภาพที่ชัดเจน มีลักษณะพื้นฐาน(feature) อันใดอันหนึ่งมีความโดดเด่นขึ้นมา

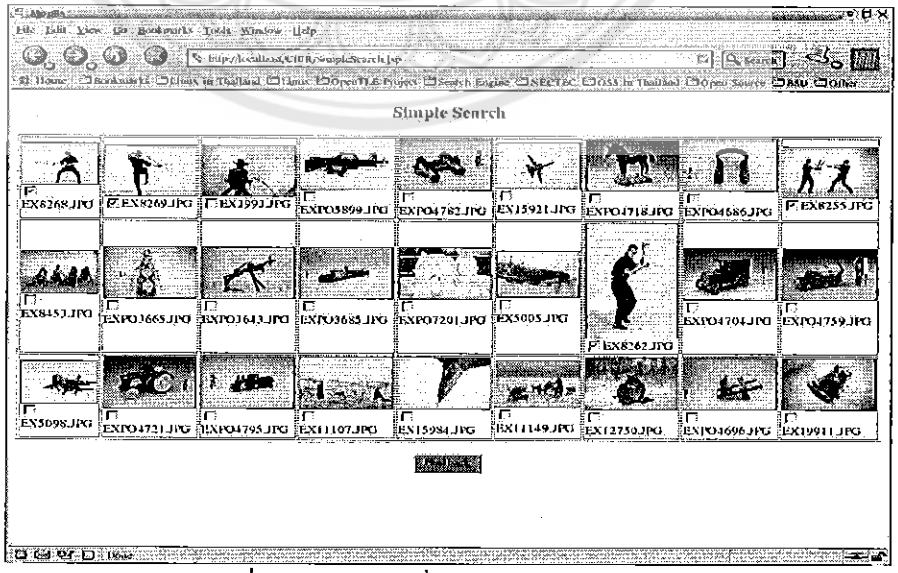
การทดลองต่อจากนี้ไปจะแสดงให้เห็นถึงประสิทธิภาพของการค้นหาอีกครั้งหนึ่งเพื่อที่จะสามารถเปรียบเทียบผลลัพธ์และค่าประสิทธิภาพกับการค้นหาครั้งแรกได้อย่างชัดเจนในรูปที่ 4.14 เป็นการค้นหาภาพโดยใช้ภาพศิลปะการการต่อสู้เป็นภาพต้นแบบ

รูปที่ 4.14 การค้นหาภาพโดยใช้ภาพศิลปะการต่อสู้

ผลที่ได้ครั้งแรกจาก Simple search จะได้ผลลัพธ์ที่ตรงกับความต้องการเพียง ภาพเท่านั้นดังนั้น ค่าประสิทธิภาพจึงหาได้ดังนี้

$$\text{Precision} = \frac{4}{27} \times 100$$

$$= 14.81\ \%$$

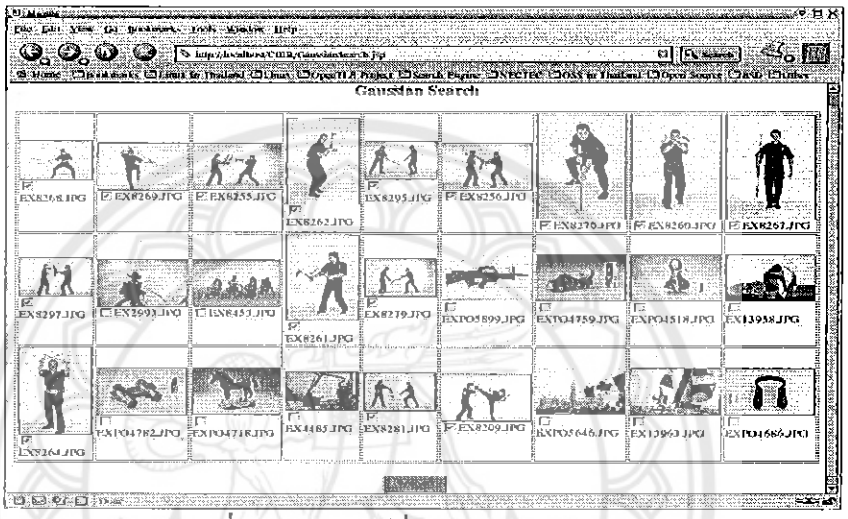


รูปที่ 4.15 ผลลัพธ์ที่ได้จาก Simple search

หลังจากการป้อน Feedback กลับไปให้ระบบทำการค้นหาอีกครั้งหนึ่ง จะได้ภาพที่มีความเหมือนเพิ่มมากขึ้น ดังรูปที่ 4.16

$$\text{Precision} = \frac{15}{27} \times 100$$

$$= 55.56\ \%$$

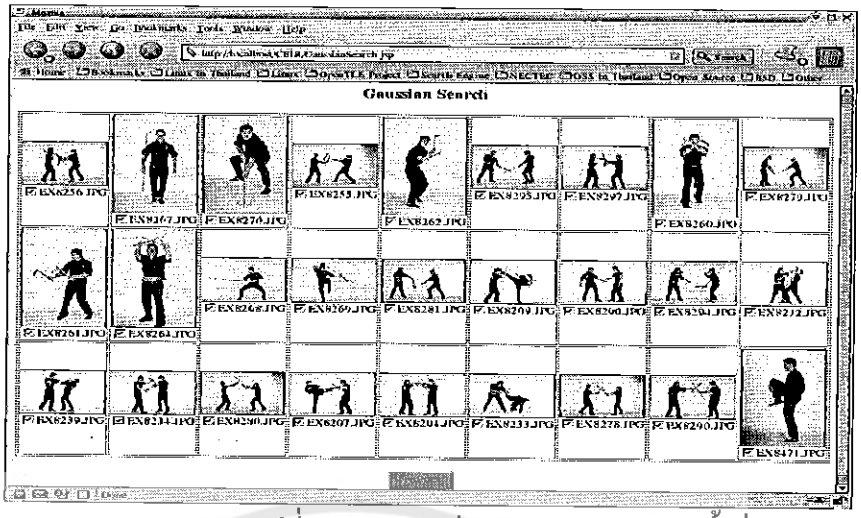


รูปที่ 4.16 ผลลัพธ์ที่ได้จากการ Feedback

ผลการค้นหาครั้งนี้ถือได้ว่ามีประสิทธิภาพมากเนื่องจากได้ภาพที่ต้องการทุกภาพที่สามารถแสดงให้เห็นได้ ดังนั้นประสิทธิภาพของการค้นหาครั้งนี้จึงเท่ากับ

$$\text{Precision} = \frac{27}{27} \times 100$$

$$= 100\ \%$$

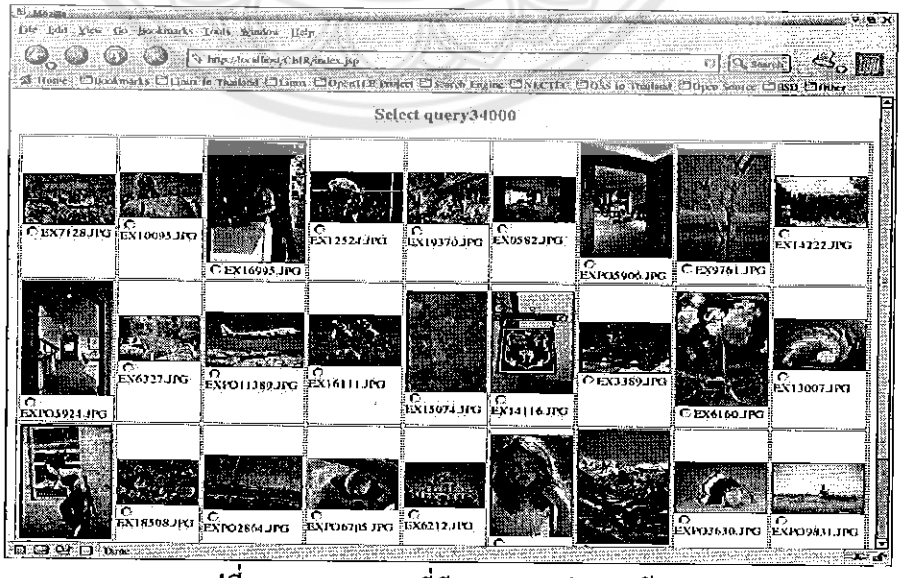


รูปที่ 4.17 ผลลัพธ์จากการ Feedback ครั้งที่ 2

การทดลองที่จะทดลองต่อไปนี้เป็นการเปลี่ยนรูปแบบของภาพให้เป็นภาพที่มีลักษณะที่ยากต่อการค้นหาดังแสดงในรูปที่ 4.18 เลือกภาพภูเขาเป็นภาพต้นแบบผลการค้นหาครั้งแรกจะได้ดังรูปที่ 4.19

$$\text{Precision} = \frac{4}{27} \times 100$$

$$= 14.81\ \%$$

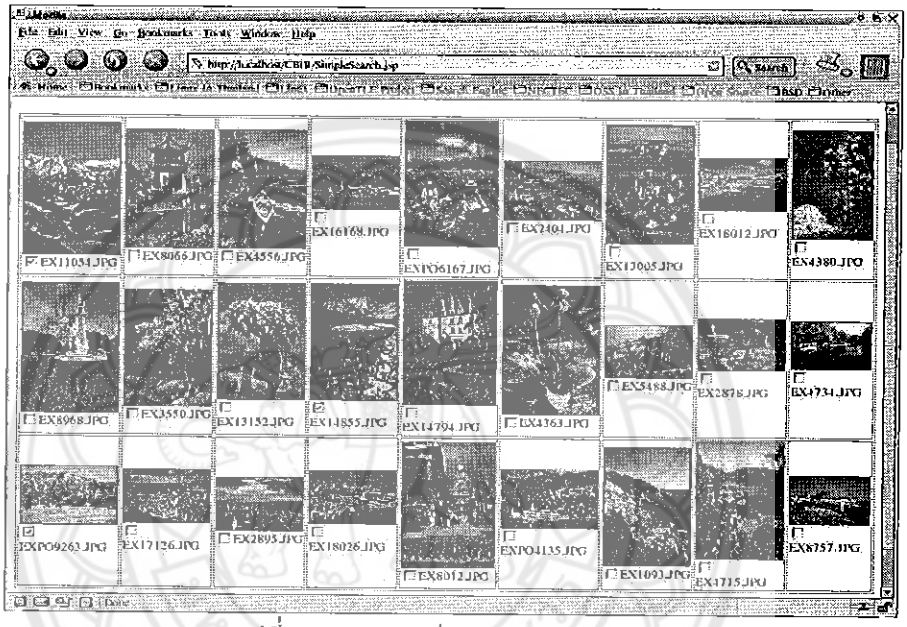


รูปที่ 4.18 ภาพภูเขาที่มีความยากต่อการค้นหา

ผลลัพธ์จากการค้นหาแบบ Simple search ในรูปที่ 4.20 ภาพที่ต้องการจะมีจำนวนใกล้เคียงกับตัวอย่างอื่น ๆ ก่อนหน้านี้ในขั้นตอนเดียวกัน

$$\text{Precision} = \frac{6}{27} \times 100$$

$$= 22.22\ \%$$

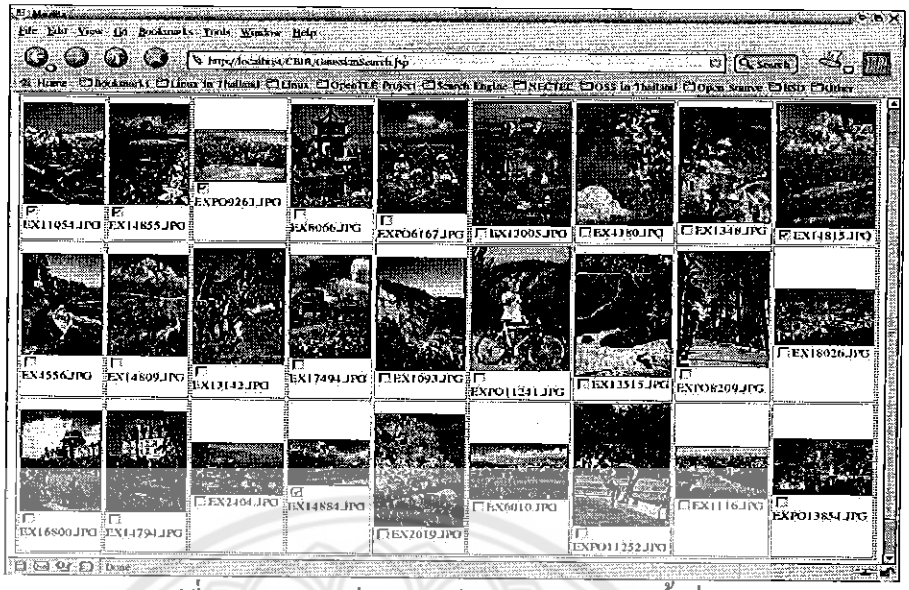


รูปที่ 4.19 ผลลัพธ์จาก Simple search

เมื่อป้อน Feedback ให้กับระบบเพื่อทำการค้นหาใหม่จะได้ผลลัพธ์ที่ไม่แตกต่างไปจากเดิมมากนัก เพราะภาพที่ต้องการเพิ่มขึ้นเพียงไม่กี่ภาพ

$$\text{Precision} = \frac{8}{27} \times 100$$

$$= 29.62\ \%$$



รูปที่ 4.20 ผลลัพธ์จากการป้อน Feedback ครั้งที่แรก

ผลลัพธ์สุดท้ายในรูปที่ 4.22 ที่ทำการทดลองนี้มีค่าประสิทธิภาพไม่ดีนักเพราะภาพมีความยากในการค้นหา

$$\text{Precision} = \frac{9}{27} \times 100$$

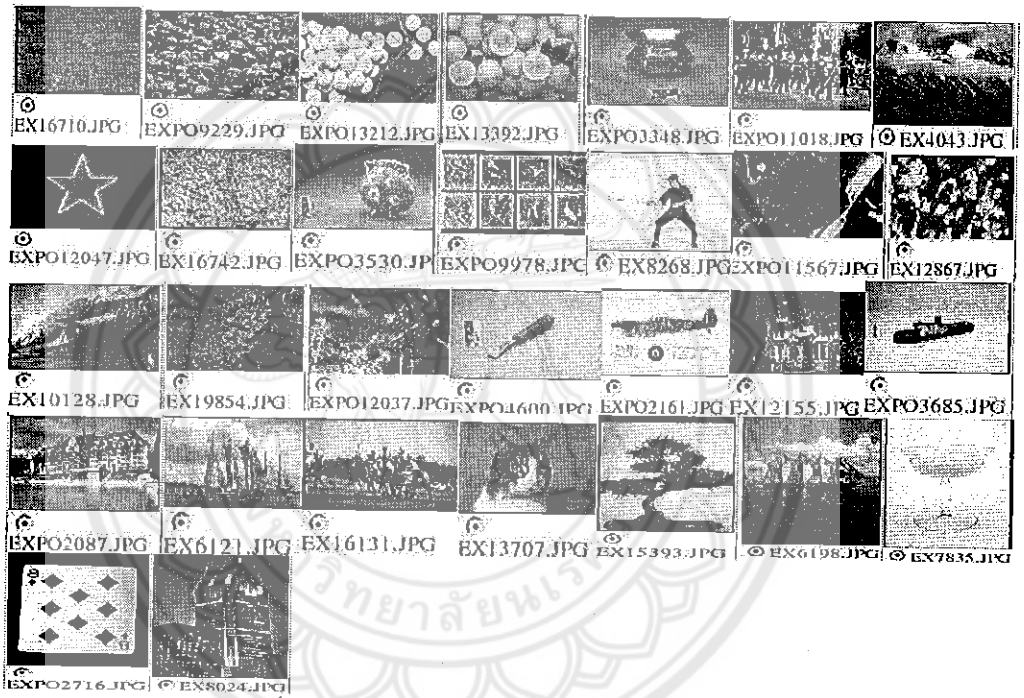
$$= 33.33\ \%$$

จากการทดลองกับภาพที่มีความยากในการค้นหาทำให้สามารถสรุปได้ว่าภาพที่มีความยากในการค้นหาคือ ภาพที่มีลักษณะขององค์ประกอบภายในภาพไม่เป็นระเบียบ ไม่สามารถแยกวัตถุหนึ่งออกจากอีกวัตถุหนึ่งได้ ภาพมีความคลุมเครือ ลักษณะพื้นฐาน(feature) มีค่าใกล้เคียงกัน

ในการทดสอบระบบเพื่อทำการค้นหาภาพนั้น ความต้องการของผู้ใช้แต่ละคนย่อมมีความแตกต่างกัน มุมมองต่อผลลัพธ์ที่ได้ก็แตกต่างกันด้วย เพื่อเป็นการแสดงให้เห็นถึงประสิทธิภาพการทำงาน จึงต้องมีการทดลองให้ผู้ใช้หลายคน ทำการค้นหาภาพตามที่กำหนดไว้จำนวน 30 ภาพ โดยมีข้อกำหนดคือ ผู้ใช้ทั้ง 3 คนจะต้องใช้ภาพต้นแบบ(Query) ภาพเดียวกันทั้ง 3 คนและทำการค้นหาโดยให้ป้อนผลกลับให้กับระบบ(Feedback) เป็นจำนวนครั้งที่แน่นอนทุกภาพ แล้วทำการบันทึกผลเก็บไว้ซึ่งแบ่งย่อยออกเป็นสองส่วนคือ ส่วนที่ใช้ Positive feedback และส่วนที่ใช้ทั้ง Positive feedback และ Negative feedback วิธีการหลังนี้จะใช้การการป้อน feedback แล้วนำค่าที่ป้อน

กลับนั้นไปทำการปรับปรุง Query โดยทำการเลื่อนจุดที่เป็นเซนเตอร์(Cetner) ของ Gaussian ให้ขยับห่างออกจากกลุ่ม ของ Negatieve feedback ดังสมการที่ (4.1)

หลังจากนั้นนำค่าผลลัพธ์ของแต่ละคนมาหาค่าเฉลี่ยของประสิทธิภาพการทำงานโดยรวมของระบบ ซึ่งแยกตามวิธีการได้สองวิธีดังที่ได้กล่าวไปแล้ว ตารางบันทึกผลการทดลองจะเก็บบันทึกข้อมูลของประสิทธิภาพการค้นหาจาก 2 วิธี คือ ผลลัพธ์จากวิธี Simple search และผลลัพธ์หลังจากการป้อน Relevance Feedback ได้แสดงไว้ในตารางที่ 4.1 ถึงตารางที่ 4.4 และตารางที่ 4.5 ถึงตารางที่ 4.8 ภาพต้นแบบที่ใช้ค้นหาทั้งหมดโดยวิธี Posive feedbackแสดงไว้แล้วในรูปที่ 4.21 และโดยวิธี Positive feedback ร่วมกับ Negative feedback แสดงไว้ในรูปที่ 4.30



รูปที่ 4.21 ภาพที่ใช้เป็นภาพต้นแบบในการค้นหา

ผลการทดลองโดยวิธีใช้เฉพาะ Positive feedback ปรับปรุง Query เพียงอย่างเดียวจากผู้ใช้ 3 คน ซึ่งแต่ละคนจะทำการเลือก Query ที่เหมือนกันทุกคนในแต่ละครั้งการทดลอง แล้วบันทึกผลลัพธ์ของแต่ละคนไว้ดังแสดงไว้ในตารางดังต่อไปนี้

ตารางที่ 4.1 ค่าประสิทธิภาพจากผู้ใช้คนที่ 1

Methods	Average Retrieval Rate (AVR) , %			
	0 Iter.	1 Iter.	2 Iter.	3 Iter.
Simple search	31.24	-	-	-
Interactive Search with GMM	31.24	48.89	61.73	65.68

ตารางที่ 4.2 ค่าประสิทธิภาพจากผู้ใช้คนที่ 2

Methods	Average Retrieval Rate (AVR) , %			
	0 Iter.	1 Iter.	2 Iter.	3 Iter.
Simple search	33.68	-	-	-
Interactive Search with GMM	33.68	52.93	63.69	70.68

ตารางที่ 4.3 ค่าประสิทธิภาพจากผู้ใช้คนที่ 3

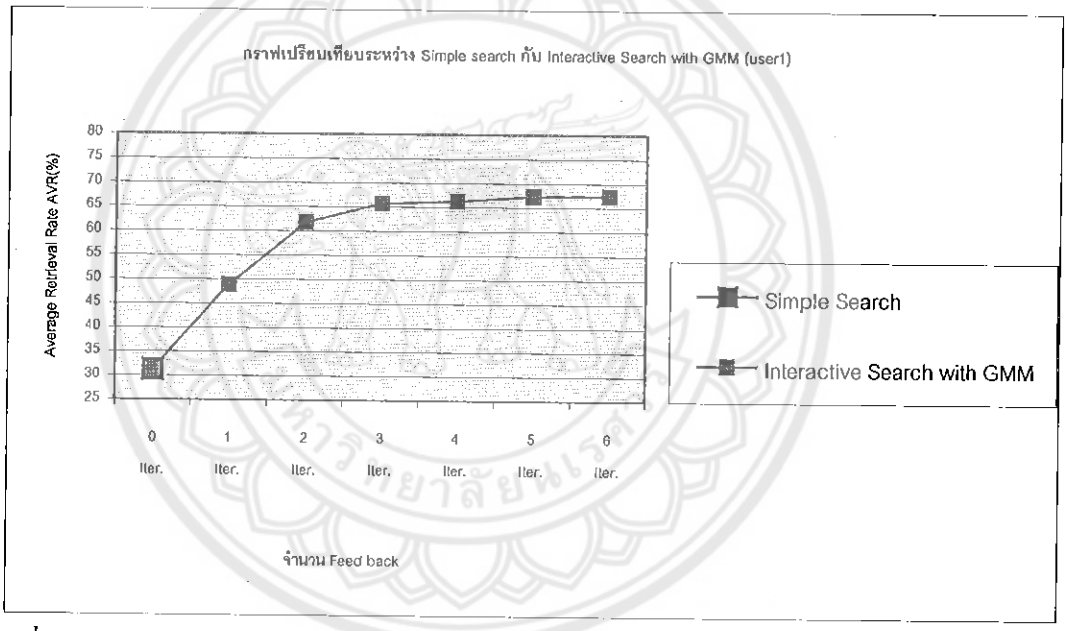
Methods	Average Retrieval Rate (AVR) , %			
	0 Iter.	1 Iter.	2 Iter.	3 Iter.
Simple search	32.59	-	-	-
Interactive Search with GMM	32.59	51.68	62.30	68.29

ตารางที่ 4.4 แสดงค่าเฉลี่ยของประสิทธิภาพการทำงานของผู้ใช้ทั้ง 3 คน จะสังเกตได้ว่าค่าประสิทธิภาพของการค้นหาที่ได้จากผู้ใช้แต่ละคนมีค่าต่างกันทั้งนี้เนื่องจากผู้ใช้แต่ละคนมีเกณฑ์การตัดสินใจเลือกภาพที่ใช้ป้อนเป็น Feedback ต่างกันดังที่ได้กล่าวไปแล้ว ในการทดลองค่าประสิทธิภาพของการสืบค้นมีค่าประมาณ 68% ซึ่งอาจจะไม่มากนักทั้งนี้เป็นเพราะการแสดงผลลัพธ์จากการค้นหาแต่ละครั้งได้กำหนดให้แสดงภาพทั้งหมด 27 ภาพซึ่งค่อนข้างมากเมื่อเทียบกับผลลัพธ์ที่ได้จากการค้นหาในแต่ละครั้ง

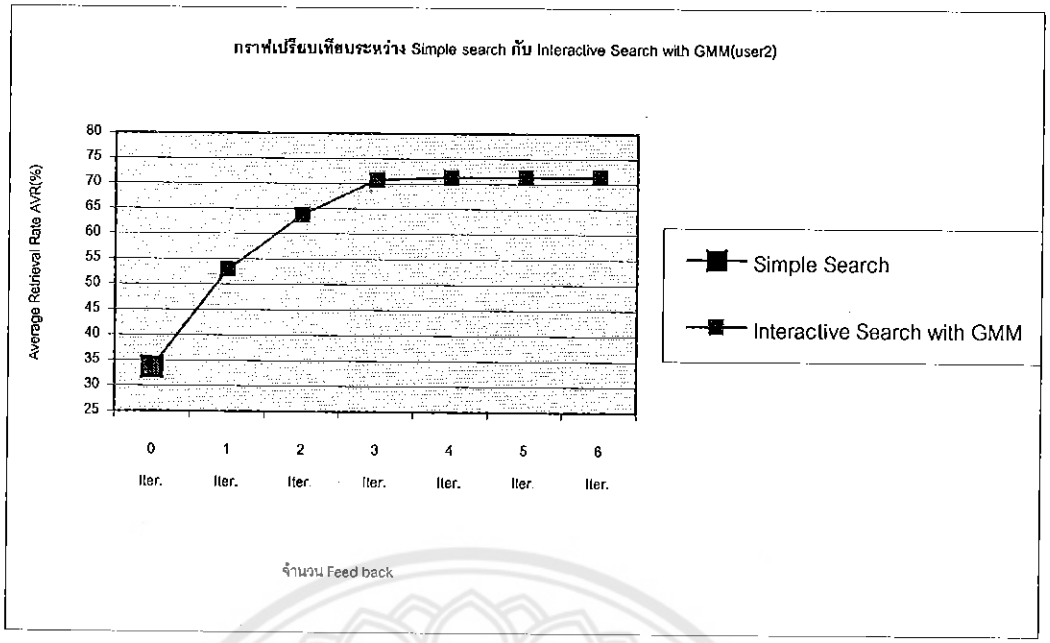
ตารางที่ 4.4 ค่าเฉลี่ยของประสิทธิภาพโดยรวมของระบบ

Methods	Average Retrieval Rate (AVR) , %			
	0 Iter.	1 Iter.	2 Iter.	3 Iter.
Simple search	32.50	-	-	-
Interactive Search with GMM	32.50	51.17	62.57	68.22

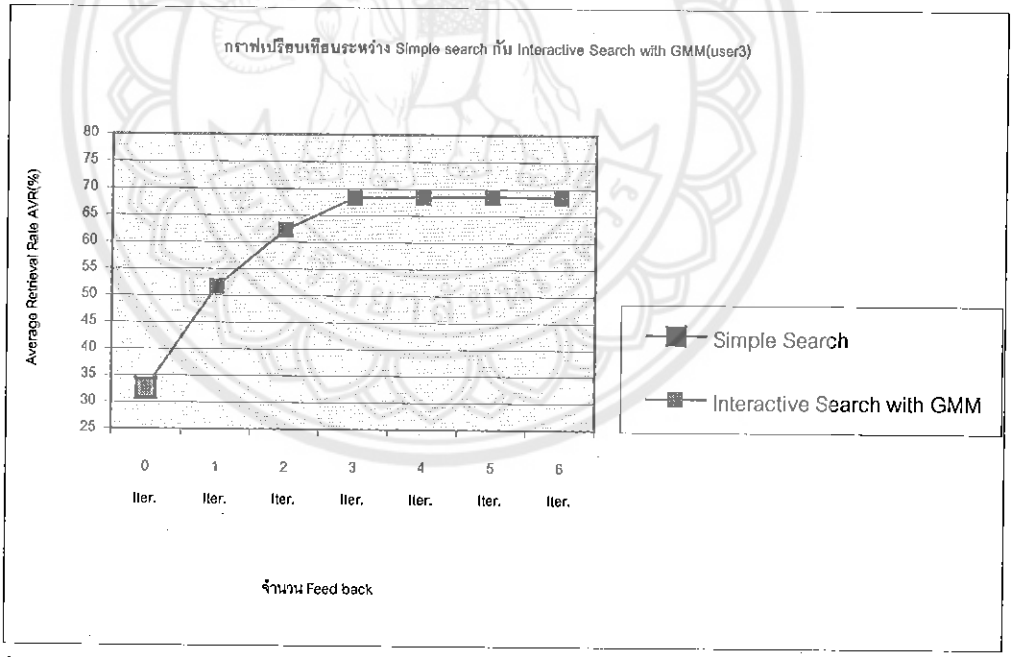
จากผลการทดลองของผู้ใช้ทั้ง 3 คนสามารถนำมาเขียนเป็นกราฟ เปรียบเทียบค่าประสิทธิภาพของแต่ละคนได้ นอกจากนี้ยังสามารถนำมาหาค่าเฉลี่ยของทั้ง 3 คนมาเปรียบเทียบเพื่อดูประสิทธิภาพโดยรวมได้ในรูปที่ 4.22 ถึงรูปที่ 4.24 เป็นกราฟเปรียบเทียบค่าประสิทธิภาพระหว่างวิธี Simple search และ Gaussian Mixture Modelของผู้ใช้คนที่ 1 คนที่ 2 และคนที่ 3 ตามลำดับ



รูปที่ 4.22 กราฟเปรียบเทียบค่าประสิทธิภาพระหว่างวิธี Simple search กับ Interactive Search with GMM ของผู้ใช้คนที่ 1

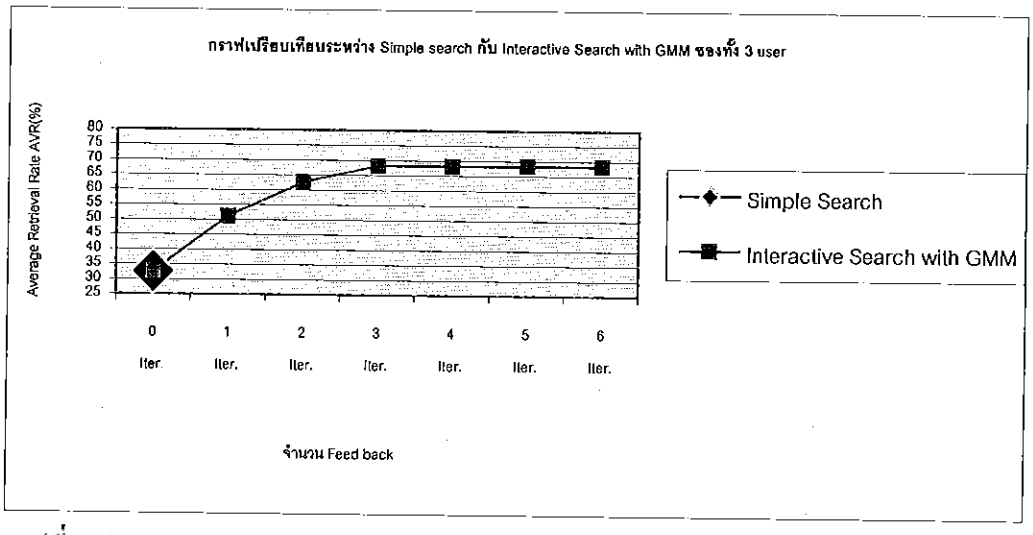


รูปที่ 4.23 กราฟเปรียบเทียบค่าประสิทธิภาพระหว่างวิธี Simple search กับ Interactive Search with GMM ของผู้ใช้คนที่ 2



รูปที่ 4.24 กราฟเปรียบเทียบค่าประสิทธิภาพระหว่างวิธี Simple search กับ Interactive Search with GMM ของผู้ใช้คนที่ 3

จากผลการทดลองของผู้ใช้ทั้ง3 คนสามารถหาค่าเฉลี่ยของค่าประสิทธิภาพในการค้นหาได้ซึ่งแสดงได้ดังรูปที่ 4.25



รูปที่ 4.25 กราฟเปรียบเทียบค่าประสิทธิภาพระหว่างวิธี Simple search กับ Interactive Search with GMM ของผู้ใช้ทั้ง 3 คน

ผลการทดลองโดยวิธีใช้ Positive feedback ร่วมกับ Negative feedback เพื่อ ปรับปรุง Query จะเก็บผลการทดลองจากผู้ใช้ 3 คนเหมือนกับการทดลองโดยใช้เฉพาะ Positive feedback ซึ่งแต่ละคนจะทำการเลือก Query ที่เหมือนกันทุกคนในแต่ละครั้งการทดลองซึ่งแสดงไว้ใน รูปที่ 4.30 แล้วบันทึกผลลัพธ์ของแต่ละคนไว้ดังแสดงไว้ในตารางดังต่อไปนี้

ตารางที่ 4.5 ค่าประสิทธิภาพจากผู้ใช้คนที่ 1 (วิธี Positive feedback และ Negative feedback)

Methods	Average Retrieval Rate (AVR) , %			
	0 Iter.	1 Iter.	2 Iter.	3 Iter.
Simple search	35.24	-	-	-
Interactive Search with GMM	35.24	54.89	66.73	71.68

ตารางที่ 4.6 ค่าประสิทธิภาพจากผู้ใช้คนที่ 2 (วิธี Positive feedback และ Negative feedback)

Methods	Average Retrieval Rate (AVR) , %			
	0 Iter.	1 Iter.	2 Iter.	3 Iter.
Simple search	37.68	-	-	-
Interactive Search with GMM	37.68	57.93	73.69	76.68

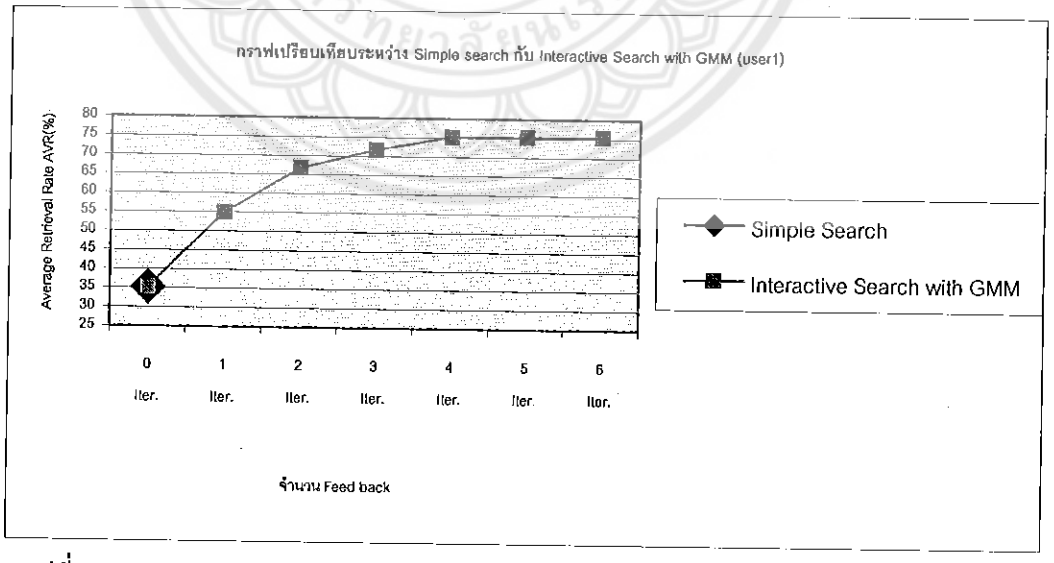
ตารางที่ 4.7 ค่าประสิทธิภาพจากผู้ใช้คนที่ 3 (วิธี Positive feedback และ Negative feedback)

Methods	Average Retrieval Rate (AVR) , %			
	0 Iter.	1 Iter.	2 Iter.	3 Iter.
Simple search	36.59	-	-	-
Interactive Search with GMM	36.59	55.68	67.30	73.29

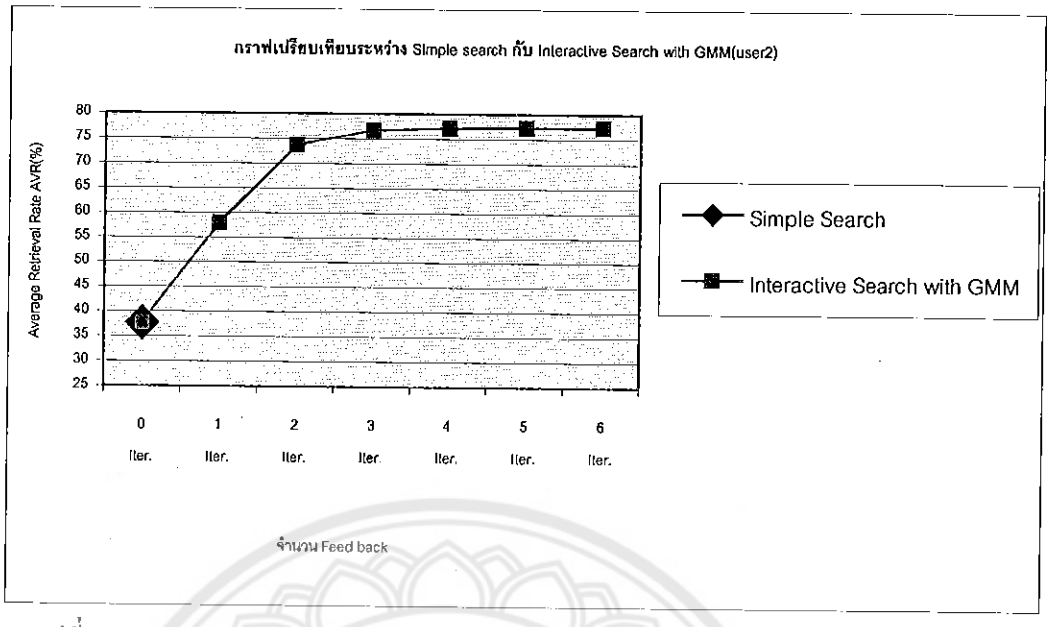
ตารางที่ 4.8 ค่าเฉลี่ยของประสิทธิภาพโดยรวมของระบบ (วิธี Positive feedback และNegative feedback)

Methods	Average Retrieval Rate (AVR) , %			
	0 Iter.	1 Iter.	2 Iter.	3 Iter.
Simple search	36.50	-	-	-
Interactive Search with GMM	36.50	56.17	69.24	73.88

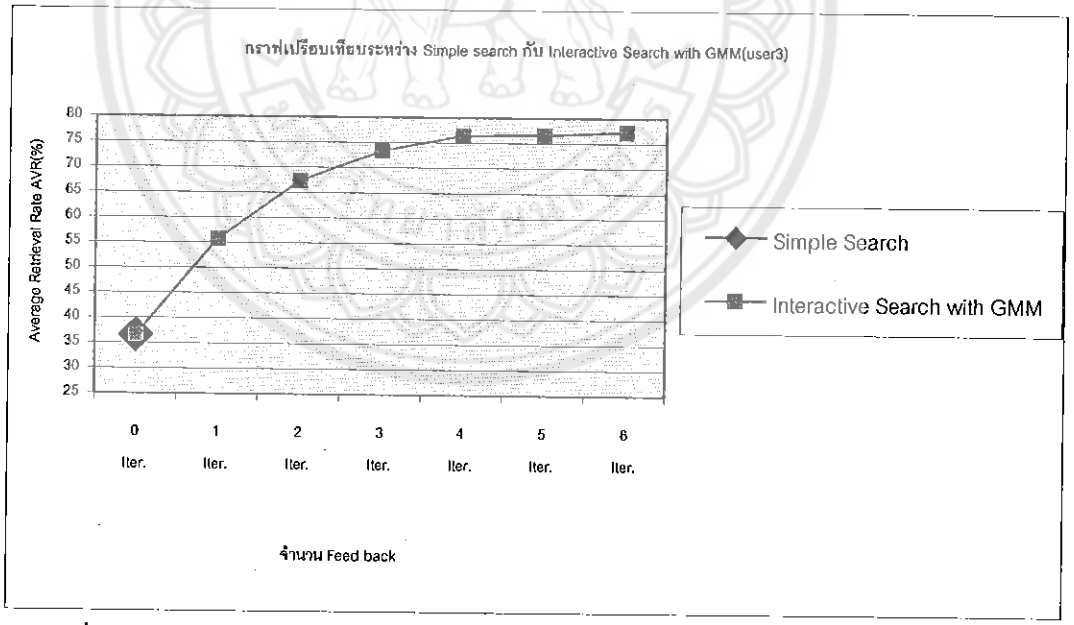
จากตารางผลการทดลองที่ 4.5 ถึงตารางที่ 4.8 สามารถทำมาเขียนเป็นกราฟแสดงค่าประสิทธิภาพได้ดังกราฟที่จะแสดงต่อไปนี้



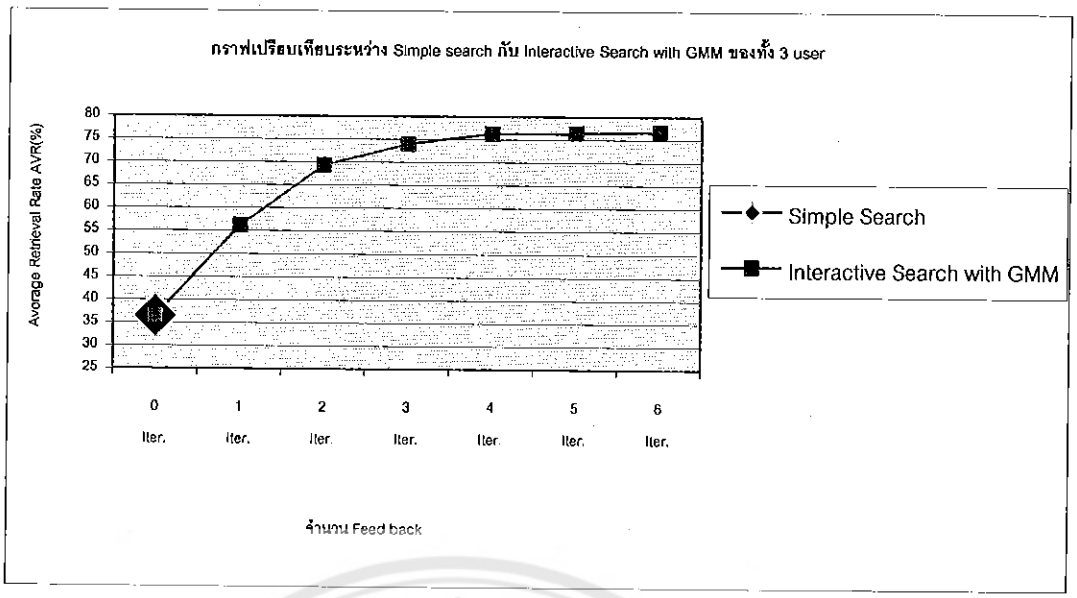
รูปที่ 4.26 กราฟเปรียบเทียบค่าประสิทธิภาพระหว่างวิธี Simple search กับ Interactive Search with GMM ของผู้ใช้คนที่ 1 (วิธี Positive feedback และ Negative feedback)



รูปที่ 4.27 กราฟเปรียบเทียบค่าประสิทธิภาพระหว่างวิธี Simple searchกับ Interactive Search with GMM ของผู้ใช้คนที่ 2 (วิธี Positive feedback และ Negative feedback)

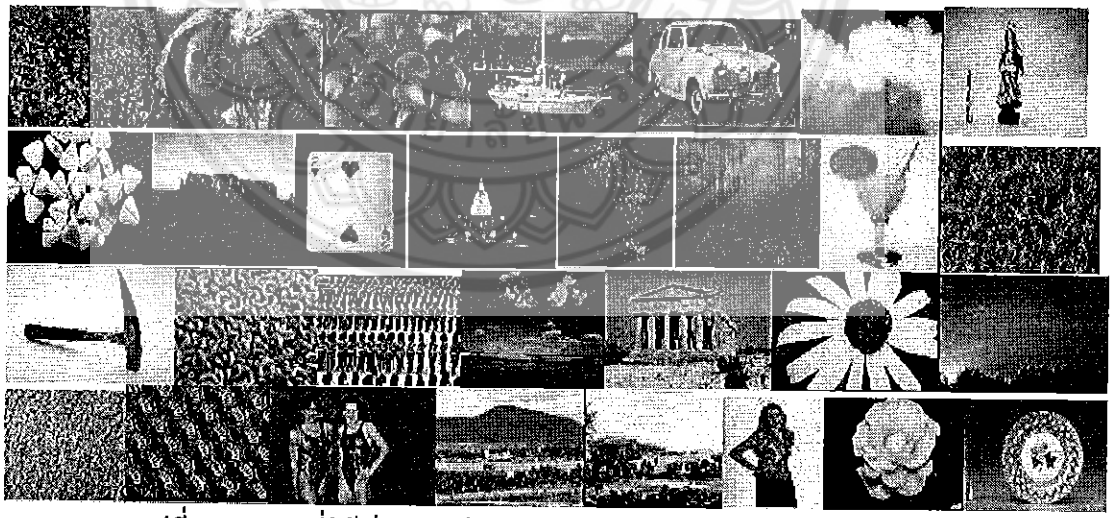


รูปที่ 4.28 กราฟเปรียบเทียบค่าประสิทธิภาพระหว่างวิธี Simple searchกับ Interactive Search with GMM ของผู้ใช้คนที่ 3 (วิธี Positive feedback และ Negative feedback)



รูปที่ 4.29 กราฟเปรียบเทียบค่าประสิทธิภาพระหว่างวิธี Simple search กับ Interactive Search with GMM ของผู้ใช้ทั้ง 3 คน (วิธี Positive feedback และ Negative feedback)

จากผลการทดลองของผู้ใช้ทั้ง 3 คนสามารถหาค่าเฉลี่ยของค่าประสิทธิภาพในการค้นหาได้ซึ่งแสดงได้ดังรูปที่ 4.29 ซึ่งมีประสิทธิภาพโดยเฉลี่ยอยู่ในเกณฑ์ที่ดี



รูปที่ 4.30 ภาพที่ใช้เป็นภาพต้นแบบในการค้นหาแบบ Positive feedback และ Negatifeedback

4.2 สรุปผลการทดลอง

จากการทดสอบการทำงานของโปรแกรมทำให้ทราบว่า algorithm ที่ใช้ในการ search ค่อนข้างมีประสิทธิภาพทั้งในส่วนของที่ใช้ Positive feedback เพียงอย่างเดียว และส่วนที่ใช้ทั้ง Positive feedback และ Negative feedback ดังจะเห็นได้จากตัวอย่างที่ได้เสนอไว้แล้วข้างต้น และผลการทดลองทั้งสองวิธีที่ได้นำเสนอไว้ จะเห็นว่า ภาพที่ได้จะมีลักษณะ หรือรายละเอียดที่เหมือนกันเช่น ถ้าเราเน้นที่ texture ของภาพ ผลการค้นหาก็จะได้ภาพที่มีลักษณะ texture ที่เหมือนกัน หรือ ถ้าเราเน้นที่ shape ก็จะได้ภาพที่มีรูปทรงที่ใกล้เคียงกัน โดยประสิทธิภาพของการใช้ทั้ง Positive feedback และ Negative feedback จะให้ผลที่ดีกว่าแต่โดยภาพรวมแล้วถือว่าทั้งสองวิธีมีประสิทธิภาพที่ดีมาก แต่ทั้งนี้ทั้งนั้นในการทำงานของโปรแกรมไม่ได้มีการทำงานเน้นหนักไปในทางใดทางหนึ่ง แต่เป็นการเรียนรู้จาก Input ที่ผู้ใช้ป้อนกลับคืนสู่โปรแกรม ดังนั้นภาพที่ได้มาจะมีลักษณะคล้ายกับกระบวนการคิดของมนุษย์ กล่าวคือ ผลลัพธ์ที่ปรากฏจะไม่เหมือนภาพต้นแบบเลยทีเดียว แต่จะมีลักษณะโดยรวมเหมือนกัน เช่น มีอัตราส่วนของสี ใกล้เคียงกัน มีรูปร่างรูปทรงที่คล้ายกัน เป็นต้น ซึ่งผลลัพธ์ดังกล่าวจะไม่สามารถบอกได้ว่าผิดหรือถูก แต่ขึ้นอยู่กับการตัดสินใจของผู้ใช้แต่ละคน ซึ่งภาพ 1 ภาพ อาจจะถูกมองแตกต่างกันออกไปในแต่ละบุคคล ดังนั้นการทำงานของโปรแกรมจะคล้ายกับกระบวนการตัดสินใจของมนุษย์โดยใช้ความรู้สึก

ผลการทดลองจะทำให้ได้ข้อสรุปเกี่ยวกับประสิทธิภาพในการค้นหาคือ ภาพที่มีความยากต่อการค้นหานั้นการค้นหาจะได้ประสิทธิภาพที่น้อยมากเมื่อเทียบกับภาพที่มีความง่ายต่อการค้นหา และปัจจัยอีกอย่างหนึ่งที่มีผลต่อประสิทธิภาพการค้นหาคือ ตัวผู้ใช้เองเนื่องจากแต่ละคนจะมีความชอบ และมุมมองที่ต่างกัน ดังนั้นภาพเดียวกันจึงอาจให้ผลลัพธ์จากการค้นหาได้หลายอย่างขึ้นอยู่กับผู้ใช้

จากการทำงานของโปรแกรมดังกล่าวจะเห็นว่ายังมีข้อบกพร่องของโปรแกรมอยู่บ้าง คือการทำงานที่อาจจะใช้เวลามากกับการโหลด feature vector จากฐานข้อมูล และการโหลดรูปภาพ อีกประการหนึ่งคือการที่จะให้ผู้ใช้เลือกภาพที่ต้องการยังทำได้ยากเพราะภาพมีจำนวนมาก ประมาณ 34,000 ภาพ ดังนั้นหากผู้ใช้ต้องการภาพใดจึงจำเป็นที่จะต้องค้นหาด้วยตัวเอง ซึ่งอาจจะต้องเสียเวลาไปบ้าง แต่อย่างไรก็ดีในตัวโปรแกรมก็จะมีส่วนที่จะช่วยให้ผู้ใช้ค้นหาภาพต้นแบบได้ง่ายขึ้น มีการ random ภาพมาให้ผู้ใช้เลือกซึ่งจะช่วยลดเวลาในการทดงานลงได้มากทีเดียววิธีการแก้ไขปัญหาที่พบดังกล่าวพอจะสรุปได้ดังนี้คือ

4.2.1 เวลาที่ใช้ในการโหลด feature vector นั้นอาจจะทำให้สั้นลงได้ด้วยการโหลด feature มาในปริมาณน้อย ๆ ก่อน ซึ่งจะทำให้การทำงานเร็วขึ้นแต่ข้อเสียคือจะทำให้การค้นหามีประสิทธิภาพต่ำลงตามไปด้วย เพราะการโหลด feature จำนวนน้อย ๆ จะทำให้ได้ภาพที่ถูกจำกัดอยู่ในวงแคบ ไม่มีความหลากหลาย

4.2.2 การค้นหาภาพต้นแบบอาจจะให้ผู้ใช้สามารถระบุได้ว่า ต้องการภาพในช่วงไหนของฐานข้อมูลจากนั้นจึงทำการโหลดเฉพาะภาพในช่วงที่ผู้ใช้ต้องการมาแสดงผล ซึ่งก็จะช่วยอำนวยความสะดวกและประหยัดเวลาได้ดียิ่งขึ้น

บทที่ 5
สรุปผลและข้อเสนอแนะ

จากทฤษฎีต่าง ๆ ที่เกี่ยวข้องกับการประยุกต์ใช้กระบวนการ Gaussian Mixture Model เพื่อใช้ในการสืบค้นหาข้อมูลประเภทภาพดิจิตอลนั้น เราสามารถนำความรู้เหล่านี้ไปใช้ให้เกิดเป็นรูปธรรมขึ้นได้นั่นคือ การสร้างโปรแกรมเพื่อสืบค้นหาภาพโดยอาศัยหลักการและทฤษฎีดังกล่าวแล้วจะทำให้ได้ระบบสืบค้นภาพที่มีประสิทธิภาพในระดับหนึ่ง สาเหตุที่กล่าวว่า มีประสิทธิภาพดีในระดับหนึ่ง คือ ระบบสามารถสืบค้นหาภาพได้ใกล้เคียงกับที่ผู้ใช้งานต้องการก็จริง แต่ผู้ใช้แต่ละคนย่อมมีทัศนคติหรือมุมมองต่อภาพเดียวกันแตกต่างกันออกไป ดังนั้นระบบอาจจะถูกมองว่าทำงานได้ดีในสายตาของผู้ใช้คนหนึ่ง แต่อาจจะไม่มีประสิทธิภาพในสายตาของผู้ใช้อีกคนหนึ่ง

อย่างไรก็ตามด้วยความสามารถและความฉลาดของอัลกอริทึมที่ใช้ในการพัฒนานั้นก็สามารถตอบสนองต่อความต้องการของผู้ใช้ได้เป็นอย่างดี ทั้งนี้เนื่องจาก Gaussian Mixture Model เป็นกระบวนการที่สามารถเรียนรู้ได้ โดยการเรียนรู้นั้นจะมีคนหรือผู้ใช้(user) เป็นผู้ป้อนข้อมูลให้กับระบบ ข้อมูลเหล่านั้นก็จะถูกระบบ ทำการประมวลผลและแสดงผลลัพธ์ให้ผู้ใช้ต่อไป ขั้นตอนการติดต่อระหว่างระบบกับผู้ใช้นี้เองที่ทำให้คอมพิวเตอร์เกิดการเรียนรู้ได้ เมื่อเกิดการเรียนรู้แล้วการค้นหาภาพในลำดับต่อ ๆ ไปก็จะมีประสิทธิภาพมากขึ้นเรื่อย ๆ

ในบทที่ 3 และบทที่ 4 มีการกล่าวถึงแนวทาง และวิธีในการพัฒนาระบบค้นหาภาพ เพื่อให้ระบบทำงานได้อย่างถูกต้องดังนั้นจึงจำเป็นจะต้องมีการทดสอบทดลองระบบก่อนการนำไปใช้งานจริง ผลจากการทดลองทำให้เราได้ทราบข้อมูลหลายอย่าง ไม่ว่าจะเป็น ประสิทธิภาพของระบบ ปัญหาต่าง ๆ ที่เกิดขึ้น สาเหตุของปัญหาดังกล่าว รวมทั้งแนวทางในการแก้ปัญหาต่าง ๆ ซึ่งข้อมูลเหล่านี้สามารถบอกได้ถึงภาพรวมของระบบซึ่งสามารถแบ่งได้เป็น 4 หัวข้อสำคัญ

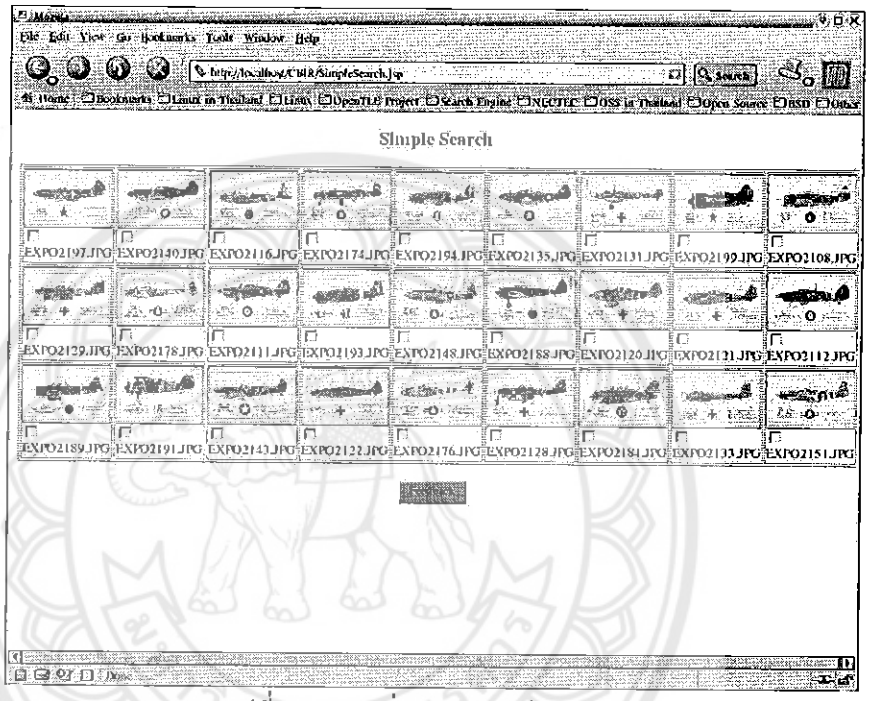
5.1 สรุปผล

5.1.1 ลักษณะของโปรแกรมนี้ เป็นโปรแกรมประเภท เว็บแอพลิเคชัน ซึ่งหมายถึงทำงานได้บนเซิร์ฟเวอร์(Server) และการทำงานต่าง ๆ จะเกิดขึ้นทางฝั่งของเซิร์ฟเวอร์ เนื่องจาก ใช้ JSP และ J2EE เป็นตัวพัฒนาซึ่งจะทำให้โปรแกรมมีการทำงานในลักษณะดัง กล่าว

5.1.2 โปรแกรมนี้ถูกพัฒนาขึ้นเพื่อใช้ในการสืบค้นหาภาพดิจิตอลบนระบบอินเตอร์เนท โดยการทำงานของโปรแกรมจะเป็นไปในลักษณะโต้ตอบกันระหว่าง Client และ Server แบบทันที (Real Time)

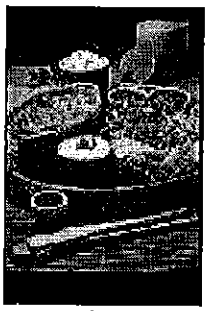
5.1.3 ประสิทธิภาพการทำงานของโปรแกรมค่อนข้างดี แต่ทั้งนี้ก็ขึ้นอยู่กับบริบทของภาพที่ผู้ใช้ต้องการด้วย ในการค้นหาจะพบลักษณะของภาพที่ถูกแบ่งออกเป็น 2 กลุ่มใหญ่ ๆ คือ กลุ่มของภาพที่ค้นหาได้ง่าย และกลุ่มของภาพที่ค้นหาได้ยาก

ภาพที่ค้นหาได้ง่ายคือ ภาพนั้นมีรายละเอียดต่าง ๆ ที่แบ่งแยกได้ชัดเจน เช่น สีในภาพมีการแบ่งกลุ่มกันอย่างชัดเจนไม่ผสมปนกัน วัตถุในภาพมีขอบที่ชัดเจน ไม่ปะปนกัน สามารถแบ่งแยกวัตถุหนึ่งออกจากอีกวัตถุหนึ่งได้ดังที่แสดงตัวอย่างไว้ในรูปที่ 5.1



รูปที่ 5.1 ภาพที่ง่ายต่อการค้นหา

ส่วนกลุ่มที่ค้นหาได้ยากคือ ภาพที่มีรายละเอียดที่คลุมเครือ มีการกระจายตัวกันขององค์ประกอบในภาพอย่างไม่เป็นระเบียบ ไม่สามารถแบ่งแยกวัตถุหนึ่งออกจากอีกวัตถุหนึ่งได้ ดังแสดงไว้ในรูปที่ 5.2



รูปที่ 5.2 ภาพที่ยากต่อการค้นหา

5.1.4 โปรแกรมค้นหาภาพสามารถปรับปรุงผลลัพธ์ให้ดีขึ้นได้ โดยอาศัยการเรียนรู้จากการ Feedback ที่ผู้ใช้ป้อนให้กับระบบ ซึ่งการทำงานในลักษณะนี้เป็นลักษณะเด่นของโปรแกรมนี้ เนื่องจากผู้ใช้สามารถระบุความต้องการให้กับระบบได้ เมื่อผู้ใช้ Feedback โต้ตอบกับระบบมากขึ้น ความต้องการต่าง ๆ ของผู้ใช้ก็จะถูกใส่เข้าไปในระบบมากขึ้น ทำให้ผลลัพธ์มีการปรับปรุงไปตามความต้องการของผู้ใช้ ซึ่งเปรียบเสมือนเกิดการเรียนรู้ขึ้นนั่นเอง

5.1.5 โปรแกรมนี้เป็นเว็บแอพลิเคชัน ดังนั้นจึงสามารถทำงานได้กับผู้ใช้หลายคนในเวลาเดียวกัน ดังนั้นการทำงานในด้านของความรวดเร็วในการประมวลผลนั้นขึ้นอยู่กับจำนวนผู้ใช้ ถ้าหากมีผู้ใช้หลายคน ก็จะส่งผลให้การทำงานช้าตามาไปด้วย เพราะการประมวลผลต่าง ๆ จะเกิดขึ้นบน Server แต่หากมีผู้ใช้น้อย การทำงานก็จะเร็วขึ้น แต่ไม่ว่าจำนวนผู้ใช้จะมีมากหรือน้อยก็จะไม่ส่งผลกระทบใด ๆ ให้กับผลลัพธ์ที่ได้

5.2 ปัญหาในการทำงาน

5.2.1 ในการสืบค้น ระบบจะพยายามเรียนรู้ความต้องการของผู้ใช้จาก feedback และจะการสืบค้นหาภาพโดยอาศัยคุณสมบัติพื้นฐานของภาพ (Low level feature) เป็นตัวเปรียบเทียบ เช่น สี (Color) ,รูปร่าง (Shape) และ พื้นผิว (Texture) ซึ่งภาพทั้งหมดที่นำมาเปรียบเทียบจะเก็บบันทึกไว้ในฐานข้อมูล ดังนั้นในบางกรณีระบบอาจจะไม่สามารถสืบค้นหาภาพที่ผู้ใช้ต้องการได้พบ เนื่องจากภาพในลักษณะที่ต้องการไม่มีอยู่ในฐานข้อมูล หรืออีกกรณีหนึ่งคือ ภาพที่ผู้ใช้ต้องการเป็นภาพที่มีความยากในการค้นหาหรือมีความคลุมเครือ

5.2.2 เนื่องจากความต้องการของผู้ใช้แต่ละคนไม่เหมือนกัน ซึ่งระบบก็จะพยายามเรียนรู้จาก feedback ของผู้ใช้และทำการค้นหา ดังนั้นการที่จะได้ภาพที่ตรงตามความต้องการอาจจะใช้จำนวนครั้งการ Feedback มากน้อยแตกต่างกัน

5.2.3 ในกรณีที่มีผู้ใช้บริการหลายคนพร้อมกัน อาจจะทำให้ประสิทธิภาพการทำงานของระบบในด้านความรวดเร็วลดลง ซึ่งส่งผลให้การประมวลผลแต่ละครั้งต้องใช้เวลานานขึ้นเมื่อเทียบกับในขณะที่มีผู้ใช้น้อยกว่า สาเหตุเนื่องจากการประมวลผลเกิดขึ้นทางฝั่งของเซิร์ฟเวอร์ ดังนั้นภาระการทำงานต่าง ๆ ของเซิร์ฟเวอร์จึงเพิ่มมากขึ้นตามจำนวนผู้ใช้บริการ

5.2.4 เนื่องจากการทำงานเป็นการทำงานทางฝั่งของเซิร์ฟเวอร์ (Server side script) ดังนั้นในขั้นตอนการทดสอบการทำงานของโปรแกรมนั้น จำเป็นต้องมีการจำลองเครื่องคอมพิวเตอร์ธรรมดา (Personal Computer : PC) ให้เป็นเครื่องเซิร์ฟเวอร์ ซึ่งประสิทธิภาพการทำงานก็จะลดลงเนื่องจากทรัพยากรที่จำเป็นต้องใช้ในการทำงานมีอยู่น้อย เช่น ความเร็วในการประมวลผลของหน่วยประมวลผลกลาง (CPU) และหน่วยความจำหลัก (RAM)

5.2.5 ในขั้นตอนเริ่มต้นการทำงาน จำเป็นต้องเก็บรวบรวมข้อมูลพื้นฐาน (feature)ของภาพทุกภาพในฐานข้อมูลมาเก็บไว้เพื่อพร้อมที่จะนำไปประมวลผลได้ทันที ข้อมูลเหล่านี้มีขนาดใหญ่ ซึ่งวิธีการนี้จะทำให้การประมวลผลแต่ละครั้งในการค้นหาเร็วขึ้น เนื่องจากไม่จำเป็นต้องติดต่อกับฐานข้อมูลทุกครั้งที่มีการเรียกใช้ข้อมูล แต่มีข้อเสียคือ ทำให้การทำงานช้าในช่วงเริ่มต้นการทำงานครั้งแรก

5.3 ข้อเสนอแนะ

5.3.1 การทำงานของโปรแกรมจะมีการประมวลผลทางฝั่งของเซิร์ฟเวอร์ ซึ่งอาจจะทำให้เกิดความล่าช้าได้ในกรณีที่มีผู้ใช้บริการหลายคนพร้อมกัน เนื่องจากข้อมูลที่ใช้ประมวลผลมีขนาดใหญ่ จึงจำเป็นต้องใช้Band Width ขนาดใหญ่ เพื่อแก้ไขปัญหานี้อาจจะเปลี่ยนการทำงานมาใช้ระบบที่มีการประมวลผลทางฝั่งของ Client (Client side script) ได้

5.3.2 เนื่องจากความต้องการของผู้ใช้บริการมีหลายลักษณะซึ่งบางครั้งข้อมูลในฐานข้อมูลอาจจะมีไม่ครอบคลุมความต้องการนั้น ดังนั้นอาจจะเพิ่มข้อมูลเข้าไปในฐานข้อมูลอีกเพื่อให้เพียงพอต่อความต้องการของผู้ใช้และเพื่อให้มีความหลากหลายของภาพมากขึ้น

5.3.3 ในการรันแอพลิเคชันนี้จำเป็นต้องกระทำบนเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ ซึ่งมีทรัพยากรที่จำเป็นต่อการประมวลผลอย่างพอเพียง แต่ถ้าหากจะทำให้ แอพลิเคชันหรือโปรแกรมทำงานได้บนคอมพิวเตอร์ส่วนบุคคลทั่วไป (PC) ก็อาจจะเพิ่มทรัพยากรเข้าไปให้เพียงพอ

5.3.4 จากการทำงานในขั้นตอนเริ่มต้นนั้นต้องใช้เวลาค่อนข้างนานกว่าการทำงานในส่วนอื่นของโปรแกรม เวลาที่ใช้ไปนี้เป็นไปเพื่อทำการติดต่อกับฐานข้อมูลแทบทั้งสิ้น ดังนั้นหากสามารถใช้ฐานข้อมูลที่มีประสิทธิภาพมากขึ้นก็จะช่วยให้เวลาในการทำงานช่วงนี้ลดลงไปได้

5.5.5 เนื่องจากเส้นทางการสื่อสารข้อมูล (Band Width)ที่มีอยู่อย่างจำกัดไม่เพียงพอต่อขนาดของข้อมูลที่มีขนาดใหญ่ ดังนั้นอาจจะเปลี่ยนการทำงานที่มีการติดต่อกันบ่อยครั้งระหว่างเซิร์ฟเวอร์ กับ คอมพิวเตอร์ของผู้ใช้ ให้เป็นการทำงานที่ไม่ต้องติดต่อกันบ่อย โดยให้การทำงานเกิดขึ้นทางฝั่งของคอมพิวเตอร์ของผู้ใช้ และให้ข้อมูลที่จำเป็นต้องใช้ในการประมวลผลอยู่ในเครื่องของผู้ใช้ เท่านี้ก็จะสามารถลดปัญหาของ เส้นทางการสื่อสารที่มีอยู่อย่างจำกัดลงได้

5.4 แนวทางในการพัฒนา

5.4.1 ความรู้และวิธีการในการทำโครงงานนี้อาจจะนำไปประยุกต์ใช้ในทางการแพทย์ได้ ในทางการแพทย์นั้นภาพจากฟิล์มเอกซเรย์ ถือว่ามีมีความสำคัญมากใช้ในการวินิจฉัยโรค ซึ่งในบางกรณี อาการหรือสาเหตุของโรคจะมีลักษณะจำเพาะดังนั้นถ้ามีฐานข้อมูลของภาพถ่ายเอกซเรย์เราก็จะอาจจะค้นหาภาพที่มีลักษณะเหมือนกับภาพที่มีอยู่จริง หรือค้นหาภาพตามความต้องการได้

ซึ่งจะทำให้การวินิจฉัยโรคเป็นไปอย่างรวดเร็ว อนึ่ง โดยลักษณะของตัวแอพลิเคชันแล้วเป็น เว็บแอพลิเคชัน ดังนั้นจึงสามารถทำงานบนระบบเนตเวิร์ค(Network) หรือระบบอินเตอร์เนต(Internet) ได้ ความสามารถจุดนี้เองอาจจะใช้ประโยชน์ได้ในกรณีที่ต้องการข้อมูลซึ่งไม่มีอยู่ในฐานข้อมูลของโรงพยาบาลแห่งหนึ่ง แต่อาจมีอยู่ที่โรงพยาบาลแห่งอื่นซึ่งอาจจะอยู่คนละซีกโลก ดังนั้นจึงสามารถหาข้อมูลได้ง่ายขึ้นโดยใช้ระบบอินเตอร์เนตร่วมกับระบบค้นหาภาพ

5.4.2 สามารถนำไปประยุกต์ใช้ในการค้นหาสภาพทางภูมิศาสตร์บนภาพถ่ายทางอากาศได้ ซึ่งรูปแบบทางภูมิศาสตร์บางอย่างจะมีลักษณะที่เป็นเอกลักษณ์ มีลักษณะที่มีรูปแบบเฉพาะเจาะจง เช่น พื้นที่ทำการกเกษตรกรรม พื้นที่ที่เป็นแอ่งน้ำ พื้นที่ป่าไม้ หรือพื้นที่ที่ประสบอุทกภัย ภาพเหล่านี้สามารถที่จะค้นหาได้โดยใช้ความรู้และวิธีการเดียวกันกับโครงงานนี้

5.4.3 ในระบบอินเตอร์เนตมีข้อมูลอยู่อย่างมากมายซึ่งไม่ได้แบกแยกกันไว้อย่างชัดเจน อาจจะมีทั้งข้อมูลที่เป็นประโยชน์เช่น ความรู้ต่าง ๆ และข้อมูลที่ไม่มีประโยชน์ต่อเยาวชนเช่น ภาพลามกอนาจาร เราสามารถตรวจสอบและดักจับข้อมูลเหล่านี้ได้โดยอาศัยหลักการและความรู้ที่ใช้ในการทำโครงงานนี้ได้

นอกจากที่กล่าวมาแล้วยังสามารถนำความรู้และตัวโปรแกรมไปประยุกต์ใช้ให้เกิดประโยชน์ในเรื่องอื่น ๆ ได้อีกมากมาย ทั้งนี้ขึ้นอยู่กับรูปแบบของปัญหาที่เกิดขึ้น หรืออยู่ที่วิธีการในการแก้ไขว่าวิธีใดถึงจะเหมาะสมที่สุด ปัญหาเดียวกันอาจจะมีวิธีแก้ไขได้แตกต่างกันหลายทาง ดังนั้นโครงงานนี้จึงไม่จำกัดว่าจะต้องสามารถนำไปใช้กับปัญหาในทางการแพทย์หรือการเกษตรกรรม ดังได้กล่าวไปแล้วเท่านั้น หากแต่ขึ้นอยู่กับวัตถุประสงค์ว่าต้องการจะใช้ให้เกิดผลสัมฤทธิ์ในด้านใด

เอกสารอ้างอิง

[1] http://vismod.www.media.mit.edu/vismod/imagery/VisionTexture

[2] http://amazon.ece.utexas.edu/~qasim/samples/sample_textures3.html

[3] Paisarn Muneesawang and Ling Guan "Automatic Machine Interaction for Content- Based Image Retrieval Using a Self-Organizing Tree Map Architecture" IEEE Transaction on Neural Networks, Vol.13, NO.4 JULY 2002 , pp. 821-827

[4] พ.อ.เจนวิทย์ เหลืองอร่าม , ปิยวิทย์ เหลืองอร่าม "การพัฒนา Web Applications ด้วย JavaServer Pages และ Servlet, JavaBeans,XML" Se-ed Ucation Lt.d , Bs2545 pp.4-5

[5] ดร. วีระศักดิ์ ซึงถาวร , กมลชนก เหมาะประสิทธิ์ , ศุภกานต์ ปิติธรรมภรณ์ " Enterprise JavaBenans " Se-ed Ucation Ltd , Bs2546

[6] ดร. วีระศักดิ์ ซึงถาวร "Java Programming Volume I " Se-ed Ucation Ltd , Bs2543

[7] วันชัย แซ่เตีย , สิทธิชัย ประสานวงศ์ " สร้างเว็บเพจด้วย HTML 4 " Se-ed Ucation Ltd , Bs2542

ภาคผนวก ก

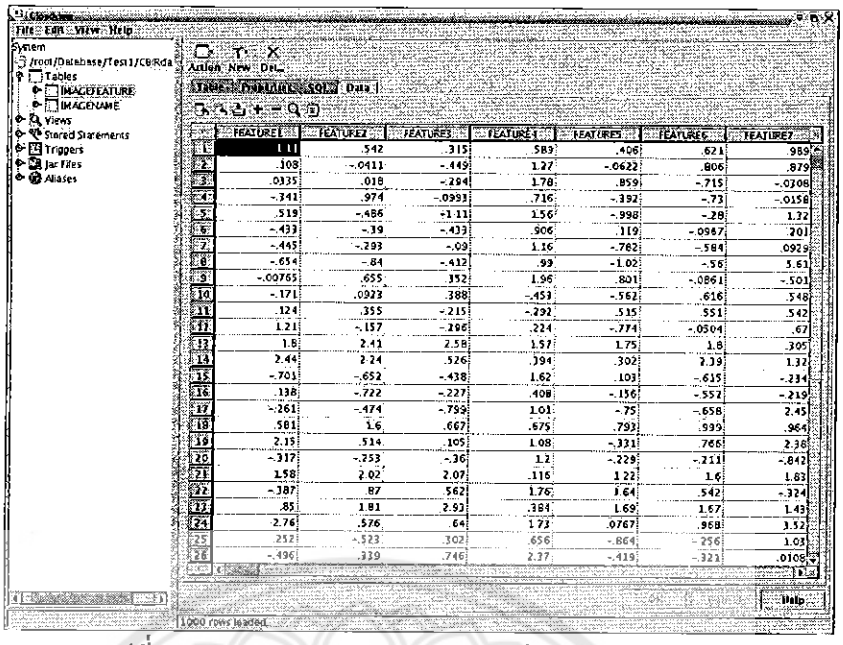
เนื้อหาในส่วนนี้เป็นการอธิบายลักษณะหน้าที่ของโปรแกรมฐานข้อมูล Cloudscape และ Java 2 Enterprise Edition (J2EE) เพื่อให้ทราบและเข้าใจถึงวิธี หลักการทำงาน และประโยชน์ของโปรแกรมที่ใช้ในการพัฒนาโครงงานนี้ ในส่วนของ Cloudscape ทำให้ทราบคุณสมบัติของตัวโปรแกรมว่าสามารถทำหน้าที่อย่างไรบ้าง ใช้ได้กับสภาวะแวดล้อมอย่างไร และในส่วนของ J2EE จะทำให้ทราบหลักการทำงานภายในว่าเป็นอย่างไร มีความสัมพันธ์ระหว่างคอมโพเนนต์ภายในอย่างไร รวมทั้งมีวิธีการติดตั้งอย่างไร

โปรแกรมฐานข้อมูล Cloudscape

โปรแกรมฐานข้อมูล Cloudscape เป็นของบริษัท Informix (Informix Software Inc.) Cloudscape เป็น Relational Database ซึ่งหน้าที่การทำงานจะเหมือนกับโปรแกรมฐานข้อมูลทั่วไป ตัวโปรแกรมทั้งหมดถูกสร้างขึ้นโดยใช้ภาษา Java ทั้งสิ้น(Pure Java Database Management System) ดังนั้นจึงสนับสนุนการทำงานกับภาษา Java ได้เป็นอย่างดี Cloudscape ถูกรวมเข้าไว้กับ J2EE(Java 2 Enterprise Edition)

การติดต่อกับ Cloudscape นั้นสามารถใช้กับภาษา Java ได้เป็นอย่างดี การเชื่อมต่อทำได้ง่าย ไม่สลับซับซ้อน และเพราะถูกสร้างจากภาษา Java ดังนั้นจึงทำงานกับภาษา Java ได้อย่างมีประสิทธิภาพ

Cloudscape มีคุณสมบัติที่ดีหลายประการ อย่างแรกคือ เป็นโปรแกรมฐานข้อมูลที่มากับ J2EE อยู่แล้วดังนั้นเมื่อติดตั้ง J2EE แล้ว Cloudscape ก็จะถูกติดตั้งลงไปอย่างอัตโนมัติไม่ต้องเสียเวลาติดตั้งเอง ประการต่อมาคือ ไม่ต้องเสียค่าใช้จ่าย เนื่องจากเป็นซอฟท์แวร์เสรี(Free ware) เหมือนกับภาษา Java และ J2EE ตัว Cloudscape เองสามารถใช้งานได้เป็นอย่างดีเพราะมีคุณสมบัติที่ฐานข้อมูลควรจะมีอยู่อย่างครบถ้วน ใช้งานง่าย แม้ว่าจะไม่ง่ายเหมือนกับฐานข้อมูลที่มีลิขสิทธิ์เช่น Microsoft Access ก็ตาม แต่ก็ถือว่ามีความสะดวกอยู่มากทีเดียว และที่สำคัญสามารถทำงานได้อย่างดีและมีประสิทธิภาพกับภาษา Java โดยภาพรวมแล้วถือว่า Cloudscape เป็นโปรแกรมฐานข้อมูลที่ดีอีกโปรแกรมหนึ่ง



รูปที่ ก-1 ลักษณะของโปรแกรมฐานข้อมูล Cloudscape

J2EE (Java 2 Enterprise Edition)

Java 2 Platform, Enterprise Edition ถูกเสนอขึ้นเพื่อลดความยุ่งยากในการสร้างโปรแกรม enterprise applications โดยกำหนดมาตรฐานสำหรับโปรแกรม application รวมทั้งสภาวะแวดล้อมสำหรับทำงาน application เหล่านี้รวมเรียกว่า J2EE architecture ประกอบด้วยข้อกำหนด(Specification) ของ application server (เรียกว่า J2EE Server) ซึ่งมีบริการมาตรฐานสนับสนุนการทำงานของ application ทำให้ผู้เขียนโปรแกรมสามารถมุ่งความสนใจไปที่ Business logic ไม่ต้องสร้างโปรแกรมที่เกี่ยวกับบริการและการติดต่อกับระบบอื่นขึ้นเอง จะช่วยให้ J2EE application ที่ได้ถูกนำไปใช้อีกสะดวกขึ้นเพราะไม่ผูกติดกับบริการ และการติดต่อกับระบบอื่น

J2EE Server มีลักษณะคล้ายกับ application server ของระบบ web-based แต่มีข้อกำหนดว่า J2EE Server ต้องสร้างสภาวะแวดล้อมสำหรับให้ J2ee application ทำงานได้ และต้องมีบริการเกี่ยวกับ transactions, instance management, persistence และ security รวมทั้งบริการเกี่ยวกับการติดต่อทางระบบเครือข่าย (Network) กับระบบอื่น

J2EE Server ต้องมี J2EE containers อย่างน้อยสองประเภทคือ

- web container สำหรับทำงาน servlets และ jsp
- ejb container สำหรับทำงาน enterprise java beans (EJB)

และอาจมี J2EE application container สำหรับทำงานกับ J2EE application client

Client ของ J2EE applications อาจเป็น browsers ที่ทำงานกับ html/sml pages หรือ applets ซึ่งติดต่อกับ web container โดย http protocol เพื่อเรียกใช้ servlets หรือ jsp แต่

clients อาจจะเป็นโปรแกรม java applications ใดแต่ต้องทำงานอยู่ใน container ของ J2EE application client ซึ่งจะสามารถติดต่อมาที่ EJB container เพื่อเรียกใช้งาน EJB เครื่องที่ทำงาน J2EE application client กับเครื่องที่ทำงาน EJB อาจเป็นเครื่องเดียวกัน หรือต่างเครื่องกันก็ได้ ถ้าอยู่ต่างเครื่องกัน ต้องเรียกผ่าน Network โดยใช้ RMI-IIOP protocol แต่ถ้าอยู่เครื่องเดียวกันจะใช้ method invocation ปกติ

นอกจากนี้ข้อกำหนดของ J2EE ก็ไม่ได้กำหนดว่า J2EE Server ต้องถูกสร้างขึ้นอย่างไร แต่กำหนดเป็น java api (class และ interface) ที่ใช้เป็นข้อตกลงในการติดต่อสื่อสารกันระหว่าง J2EE container กับ J2EE application

Component Contract

J2EE application จะถูกสร้างขึ้นและทำงานอยู่ภายใน JVM ที่อยู่ใน J2EE container ดังนั้นเพื่อให้ J2EE container สามารถสร้างและควบคุมการทดงานของ J2EE application ใด ๆ จะต้องถูกสร้างขึ้นตามข้อตกลงที่เรียกว่า component contract ซึ่งกำหนดเป็น interface โดยถ้า J2EE application เป็น

- servlet ต้อง extends javax.servlet.http.HttpServlet
- JSP ต้อง extends javax.servlet.jsp.HttpJspPage
- Enterprise java bean ต้อง implement javax.ejb.EnterpriseBean

Container Services

เป็นชุด Api สำหรับให้ J2EE applications เรียกใช้บริการที่ J2EE containers ต้องจัดหาไว้ให้โดย J2EE Server อาจทำงานสำหรับให้บริการนั้นขึ้นเอง หรือส่งคำขอใช้บริการไปยัง Server สำหรับบริการนั้นที่ทำงานอยู่ภายนอก J2EE Server ก็ได้J2EE application ไม่จำเป็นต้องทราบว่าโปรแกรมที่ให้บริการเป็นชนิดใด แต่เรียกใช้งานโดยใช้ชุด Api ของภาษา Java

Declarative Services

J2EE container สามารถให้บริการ อย่างเช่น transaction และ security แก่ J2EE application ได้โดยแต่ละ J2EE application จะมี xml ไฟล์เรียกว่า deployment descriptor สำหรับบอกแก่ J2EE container ว่าต้องการบริการใดบ้างเมื่อตอนทำงาน บริการแบบนี้เรียกว่า declarative services ถูกำหนดในไฟล์ deployment descriptor ไม่ได้กำหนดอยู่ในโปรแกรมของ J2EE application ทำให้สามารถเลือกบริการที่ต้องการได้โดยเปลี่ยนที่ไฟล์ deployment

descriptor ไม่ต้องคอมไพล์โปรแกรม J2EE application นั้นใหม่ช่วยให้ J2EE application หนึ่งถูกใช้งานภายใต้บริการที่แตกต่างกันได้เป็นการเพิ่มความสามารถในการนำมาใช้ใหม่ของโปรแกรม

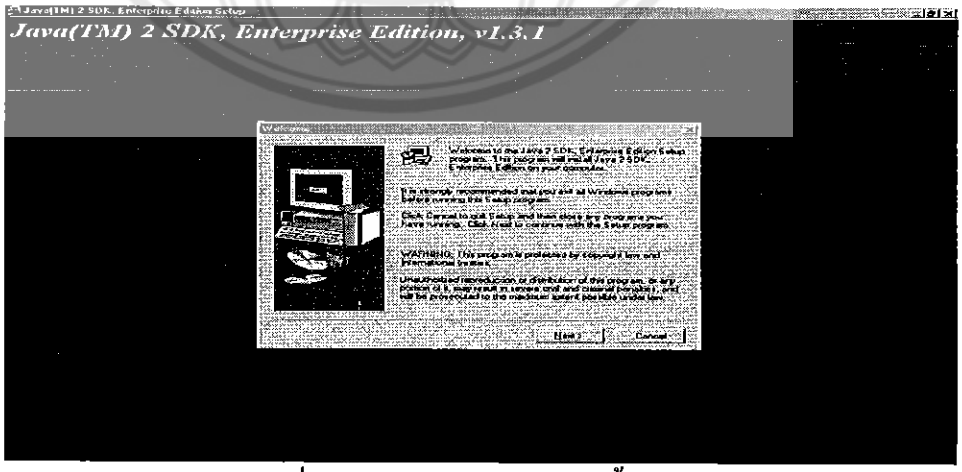
Runtime Services

เป็นบริการที่จำเป็นต้องมี สำหรับทำงานกับ J2EE application คือ

- Lifecycle management สำหรับการสร้าง และทำลาย instance ของ J2EE application รวมทั้งการทำ poling และควบคุมให้ทำงาน
- Resource Pooling สำหรับการทำ pooling ทรัพยากรอย่างเช่น database connections
- Population of the JNDI name space สำหรับนำข้อมูลจาก deployment descriptor กำหนดค่าแก่ data sources, message queur และ transactions
- Clustering เนื่องจากJ2EE containers อาจมีมากกว่าหนึ่ง JVM เพื่อช่วยกันทำงาน clustering คือการให้มี J2EE application หนึ่งถูกทำงานอยู่ใน JVM มากกว่าหนึ่งตัว และควบคุมให้ clients เข้าใช้งาน J2EE application ที่ JVMตัวใดตัวหนึ่ง เป็นการทำ load-balancing เพื่อให้ clients เข้าใช้งานได้เร็วขึ้น และระบบถูกขยายได้ง่ายขึ้น

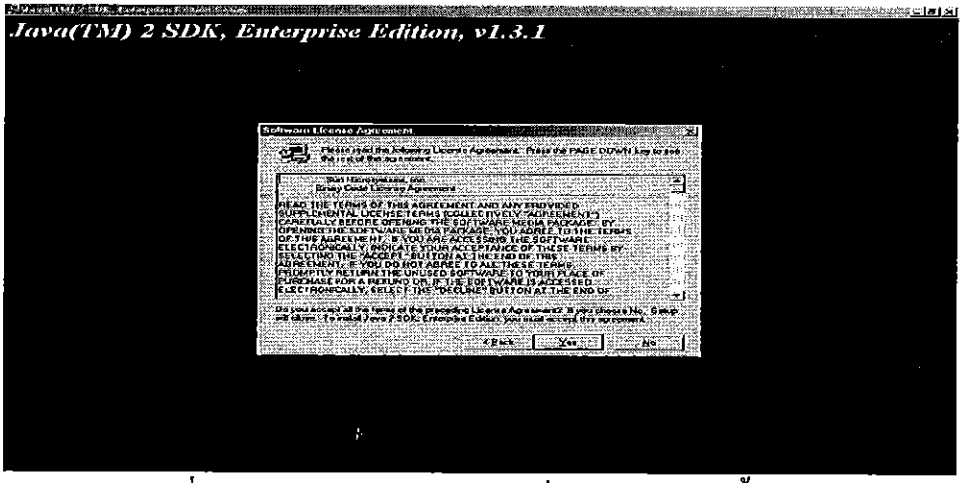
ขั้นตอนในการติดตั้ง J2EE

1. เมื่อทำการดับเบิ้ลคลิกที่ไฟล์ติดตั้งของ J2EE จะปรากฏหน้าจอการติดตั้งดังรูป ให้ทำการเลือก Next เพื่อจะติดตั้ง



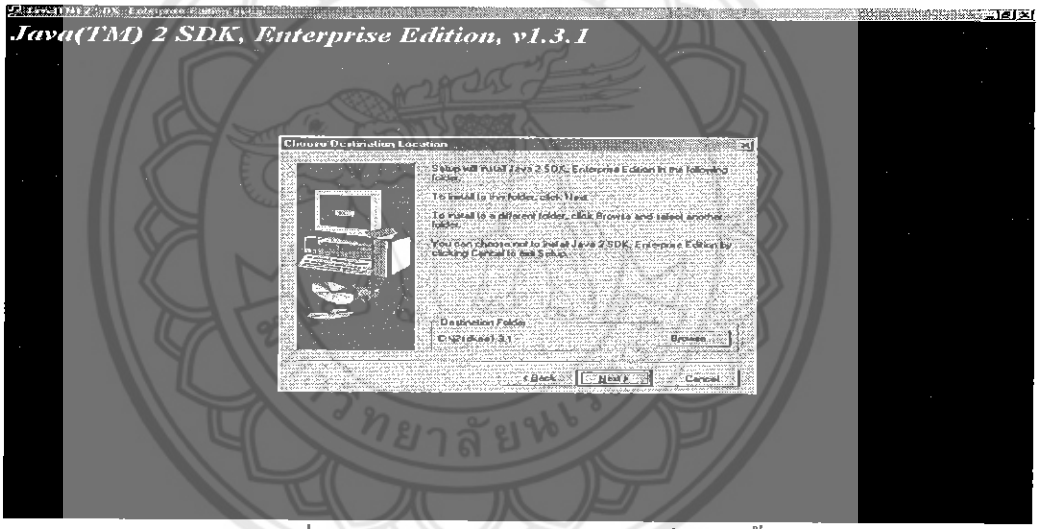
รูปที่ ก-2 แสดงหน้าต่างการติดตั้ง J2EE

2. จากนั้นให้ทำการเลือก Yes เพื่อตอบตกลงตามเงื่อนไขในการติดตั้ง



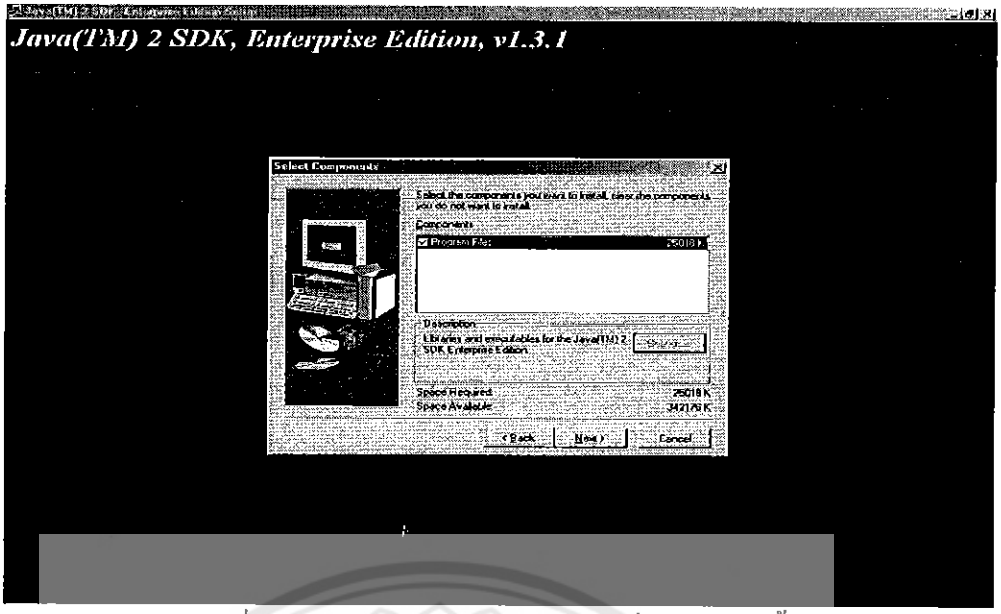
รูปที่ ก-3 แสดงรูปการเลือกยอมรับเงื่อนไขในการติดตั้ง

3. จากนั้นทำการเลือก Directory ที่ต้องการจะติดตั้ง แล้วเลือก Next



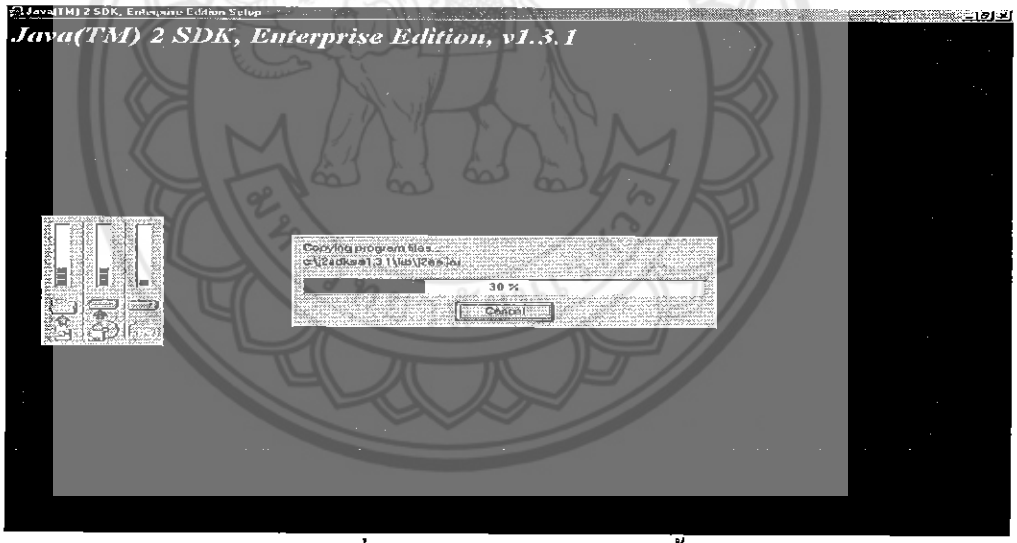
รูปที่ ก-4 แสดงการเลือก Directory ที่จะติดตั้ง

4. ทำการเลือก component ที่ต้องการจะติดตั้ง แล้ว เลือก Next



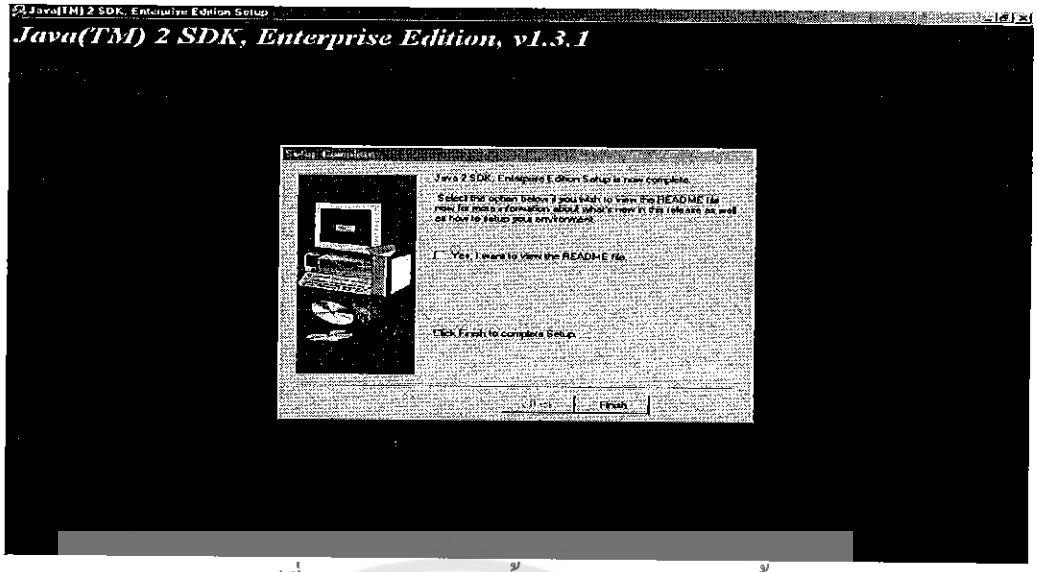
รูปที่ ก-5 แสดงการเลือก component ที่ต้องการติดตั้ง

5. จากนั้นโปรแกรมก็เข้าสู่กระบวนการติดตั้ง



รูปที่ ก-6 แสดงกระบวนการติดตั้ง

6. เป็นการสิ้นสุดกระบวนการติดตั้ง J2EE



รูปที่ ก-7 แสดงรูปการสิ้นสุดกระบวนการติดตั้ง

ภาคผนวก ข

ในส่วนนี้เป็นโค้ดโปรแกรมที่ใช้ทั้งหมดในโครงงานนี้เพื่อให้ง่ายต่อการอ้างอิงและสะดวกเมื่อต้องการค้นคว้าเพื่อเพิ่มเติมเพื่อให้มีความเข้าใจมากยิ่งขึ้น โค้ดโปรแกรมดังกล่าวมีทั้งโค้ดที่เป็นส่วนของการทดลองทั้ง text mode และ graphic mode ซึ่งโค้ดในการทดลอง Text mode เป็นโค้ดที่ยังไม่เสร็จสมบูรณ์ และในส่วนของโค้ดที่เสร็จสมบูรณ์แล้วใช้ในการทดลองใน Graphic mode คือโค้ดของโปรแกรมในโครงงานนี้นั้น จะแบ่งออกเป็นสองรูปแบบคือ โค้ดโปรแกรมที่ใช้วิธี Positive feedback เพียงอย่างเดียวในการปรับปรุง Query และโค้ดโปรแกรมที่ใช้ ทั้ง Positive feedback และ Negative feedback ปรับปรุง Query โค้ดของทั้งสองรูปแบบจะเหมือนกันเกือบทั้งหมด แตกต่างกันเพียง Class CBIRBean ของวิธีที่ใช้ทั้ง Positive และ Negative มี method getClosetNegative และ method getShiftedQuery เพิ่มขึ้นมาโปรแกรมทั้งหมดประกอบด้วยโค้ดโปรแกรมของ JavaBeans และโค้ดโปรแกรมของ JSP

Source Program

[8] source code ImageSearch.java

```
//*************************************//
//File name : ImageSearch.java    //
//*************************************//
import java.sql.*;
import java.util.*;
class ImageSearch {
   protected static int featureSize = 115;                    //Size of an image feature
   protected static int numOfImage = 5000;     //Size of an image database
   protected static float[][] imageFeature = new float[numOfImage][featureSize];
   //An array for collect feature data
   protected static String[] imageName = new String[numOfImage];
   //An array for clollect image name
   private Connection con;
        private ResultSet rs1,rs2;
```

```
//-------------------------------------------------------------------//
//  The constructor use to initialize image database     //
//-------------------------------------------------------------------//
ImageSearch() {
  //Jdbc Driver loading
  try {
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    }catch(ClassNotFoundException ex)
    {
       System.out.println(ex.getMessage());
       }
  //Connect to database and retrieve data
  try {
    //Make a connection
    con = DriverManager.getConnection("jdbc:odbc:CBIR_database","admin","");
    //Create Statement
    Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,

    ResultSet.CONCUR_READ_ONLY);

    //Retrieve image feature data from database
    String SQL1 = "SELECT * FROM MULTIFEATURE ";
    rs1 = stmt.executeQuery(SQL1);
    rs1.first();
    System.out.println("Collecting image feature...");
    for(int i=1;i<=numOfImage;i++)
    {
      for(int j=0;j<featureSize;j++)
      {
        imageFeature[i-1][j] = rs1.getFloat((j+2));
        }
```

```
            rs1.next();
            }

          System.out.println("Collecting image name...");
          String SQL2 = "SELECT * FROM IMAGELIST";
          rs2 = stmt.executeQuery(SQL2);
          rs2.first();
          for(int j=1;j<=numOfImage;j++)
          {
            imageName[j-1] = rs2.getString(2);
            rs2.next();
            }
          System.out.println("Complete to collecting data...");
          rs1.close();
          rs2.close();
          con.close();
          }catch(SQLException ex)
          {
            System.out.println(ex.getMessage());
                }
    }

 //-------------------------------------------------------------------//
// SimpleSearch method                                               //
 //-------------------------------------------------------------------//
  public int[] simpleSearch(int query,int numOfResult)
  {

    //A query is index of image feature that user select
    //A numOfResult is a number of images that require
    //after executed simple search
```

```
int[] resultIndex = new int[numOfResult];
float[] euclideanDistanceValue = new float[numOfImage];

Distance distance = new Distance();
System.out.println("Simple Search executing.....");

//Calculate Euclicean distance of all image feature
for(int i=0;i<numOfImage;i++)
{
   euclideanDistanceValue[i] =
   distance.calDistance(imageFeature[query],imageFeature[i]);
   }

//Sort an array of distance value
resultIndex = leastSorting(euclideanDistanceValue,numOfResult);

return resultIndex;
}
//-------------------------------------------------------------//
//       gaussianSearch method                                 //
//-------------------------------------------------------------//
      public int[] gaussianSearch(int[] relevanceIndex,int numOfResult) {

            int[] resultIndex = new int[numOfResult];
            float[] gaussianDistance = new float[relevanceIndex.length];
            float[] sigmaValue       = new float[relevanceIndex.length];
            float[] gaussianValue = new float[numOfImage];
            GaussianDistance gDistance = new GaussianDistance();

            sigmaValue = findSigma(relevanceIndex);
```

```
//calculate Gaussian distance
for(int i=0;i<numOfImage;i++)
{
        for(int j=0;j<relevanceIndex.length;j++)
        {

gaussianDistance[j]=gDistance.calGaussianDistance(imageFeature[relevanceIn
dex[j]],

imageFeature[i],sigmaValue[j]);
                }

        //sum all Gaussian distance of each image
        float tmp = 0.0f;
        for(int t=0;t<relevanceIndex.length;t++)
                tmp+=gaussianDistance[t];

        gaussianValue[i] = tmp;
        }

resultIndex = maxSorting(gaussianValue,numOfResult);

return resultIndex;
}
```

```
//-------------------------------------------------------------------//
//maxSorting method                                                  //
//-------------------------------------------------------------------//
        protected int[] maxSorting(float[] Xdata,int num)
 {
    float tmp;
                int[] result = new int[num];


                for(int i=0;i<num;i++)
                {
                        tmp = 0.0f ;
                        for(int j=0;j<Xdata.length;j++)
                        {
                                if(tmp<Xdata[j])
                                {
                                        tmp = Xdata[j];
                                        result[i] = j;
                                        }
                                }
                        Xdata[result[i]] = 0.0f;
                        }
               return result;
  }
```

```
//-------------------------------------------------------------------------//
//leastSorting method                                                      //
//-------------------------------------------------------------------------//
protected int[] leastSorting(float[] Xdata,int num)
    {
        float tmp;
        int[] result = new int[num];

        for(int i=0;i<num;i++)
        {
            tmp = 100f ;
            for(int j=0;j<Xdata.length;j++)
            {
                if(tmp>Xdata[j])
                {
                    tmp = Xdata[j];
                    result[i] = j;
                }
            }
            Xdata[result[i]] = 100;
        }
    return result;
    }
```

```
//-------------------------------------------------------------------//
//findSigma method                                                   //
//-------------------------------------------------------------------//
protected float[] findSigma(int[] relevanceIndex)
{
        float[] sigmaValue = new float[relevanceIndex.length];
        float[] tmp        = new float[relevanceIndex.length];
        Distance distance  = new Distance();

        Arrays.fill(tmp,99);
        for(int i=0;i<relevanceIndex.length;i++)
        {
            for(int j=0;j<relevanceIndex.length;j++)
            {
                if(i!=j)
                {
                    //find Euclidean distance
                    tmp[j] =
distance.calDistance(imageFeature[relevanceIndex[i]],

    imageFeature[relevanceIndex[j]]);
                }

            }
        //find least data
        float xxx = tmp[0];
        for(int t=0;t<tmp.length;t++)
        {
            if(xxx>tmp[t])
                xxx = tmp[t];
        }
```

```
                    sigmaValue[i] = (xxx/2);
                    }
                return sigmaValue;
                }

//-------------------------------------------------------------------------//
//getData method                                                           //
//-------------------------------------------------------------------------//
 public float getDataAt(int column,int row)
{
                return imageFeature[column][row];
                }

//-------------------------------------------------------------------------//
//getRow method                                                            //
//-------------------------------------------------------------------------//
public float[] getRow(int indexOfRow)
{
      return imageFeature[indexOfRow];
      }

//-------------------------------------------------------------------------//
//getImageNameAt method                                                    //
//-------------------------------------------------------------------------//
public String getImageNameAt(int index)
{
      return imageName[index];
      }
}//End of ImageSearch class
```

[9] Source code GaussianDistance.java

```
//*********************************************//
//File name: GaussianDistance.java        //
//*********************************************//
class GaussianDistance
{
        float calGaussianDistance(float[] feature1,float[] feature2,float sigmaValue)
        {
                float gaussianDistance;
                Distance distance = new Distance();

                float a = (float)Math.pow(distance.calDistance(feature1,feature2),2);
                float b = 2*(float)Math.pow(sigmaValue,2);
                gaussianDistance = (float)Math.exp(-a/b);

                return gaussianDistance;
                }
        }
```

[10] Source code Distance.java

```
//*********************************************//
//File name:Distance.java              //
//*********************************************//
    class Distance
    {
        public float calDistance(float[] Vect1,float[] Vect2)
        {
                float Result = 0.0f;
                //Both of Vect1 and Vect2 must same size.
                if(Vect1.length!=Vect2.length)
                {
```

```
}
                    System.out.println("Error! the two vector must be same size.");
                    return -1.00f;    //If Error then return -1
                    }
                else {
//-------------------------------------//
//Ucledian distance method  //
//-------------------------------------//
        for(int j=0;j<Vect1.length;j++)
                Result+=Math.pow((Vect1[j]-Vect2[j]),2);
                return (float)Math.sqrt(Result);
                //Return result
                    }
                }
        }
    /*
     *--------- Tester ----------
     */
    class DistanceTest
    {
        static Distance dist=new Distance();
        static float[] num1={0.1f,0.2f,0.3f};
        static float[] num2={0.5f,0.6f,0.8f};
        public static void main(String args[])
        {
                System.out.println(dist.calDistance(num1,num2));
                }
        }
```

[11] Source code TestDatabase.java

```
//***********************************//
// File name: TestDatabase.java //
//***********************************//
    class TestDatabase
    {
        public static void main(String args[])
        {
                ImageSearch database = new ImageSearch();
                System.out.println("0,0 : "+database.getDataAt(0,0));
                System.out.println("0,114 : "+database.getDataAt(0,114));
                System.out.println("499,0 : "+database.getDataAt(499,0));
                System.out.println("499,114 : "+database.getDataAt(499,114));
                }
        }
```

[12] Source code TestDistance.java

```
//*****************************************//
//File name: TestDistance.java          //
//*****************************************//
    class TestDistance
    {
        public static void main(String args[])
        {
                ImageSearch database = new ImageSearch();
                Distance distance = new Distance();
                System.out.println(distance.calDistance(database.getRow(1),database.g
                etRow( 2)));
                }
        }
```

[13] Source code TestFindSigma.java

```
//*****************************************//
//File name: TestFindSigma.java          //
//*****************************************//
    import java.util.*;
    class TestFindSigma
    {
      static float[][] imageFeature = {{0.1f,0.2f,0.3f,0.4f,0.5f,0.6f,0.7f,0.8f,0.9f,0.72f},
                                       {0.54f,0.8f,0.06f,0.42f,0.97f,0.81f,0.13f,0.06f,0.48f,0.32f},
                                       {0.65f,0.44f,0.7f,0.1f,0.2f,0.6f,0.19f,0.44f,0.73f,0.99f},
                                       {0.01f,0.09f,0.18f,0.56f,0.49f,0.37f,0.89f,0.79f,0.88f,0.65f},
                                       {0.32f,0.33f,0.67f,0.46f,0.91f,0.86f,0.74f,0.3f,0.11f,0.09f},
                                       {0.84f,0.71f,0.77f,0.84f,0.21f,0.5f,0.98f,0.14f,0.7f,0.01f}};
      public static void main(String args[])
      {
            int[] relevanceFeedback = {0,1,2,3,4,5};
            float[] result = new float[relevanceFeedback.length];
            result = findSigma(relevanceFeedback);
            for(int i=0;i<result.length;i++)
                  System.out.println(result[i]);
            }
      static float[] findSigma(int[] relevanceIndex)
      {
            float[] sigmaValue = new float[relevanceIndex.length];
            float[] tmp        = new float[relevanceIndex.length];
            Distance distance  = new Distance();
            for(int i=0;i<relevanceIndex.length;i++)
            {
                  Arrays.fill(tmp,99);
                  for(int j=0;j<relevanceIndex.length-1;j++)
                  {
```

```
                    if(i!=j)
                    {
                        tmp[j] =
distance.calDistance(imageFeature[relevanceIndex[i]],
        imageFeature[relevanceIndex[j]]);
                    }
                }
                float xxx = tmp[0];
                for(int t=0;t<tmp.length;t++)
                {
                    if(xxx>tmp[t])
                        xxx = tmp[t];
                }
                sigmaValue[i] = (xxx/2);
            }
        return sigmaValue;
        }
    }
```

[14] Source code TestGaussianDistance.java

```
//**********************************************//
//File name: TestGaussianDistance.java      //
//**********************************************//
    class TestGaussianDistance
    {
        public static void main(String args[])
        {
            float[][] data = {{0.1f,0.2f,0.3f,0.4f,0.5f,06f,0.7f,0.8f,0.9f,0.27f},
                    {0.4f,0.2f,0.6f,0.8f,0.7f,0.6f,0.1f,0.7f,0.2f,0.4f}};
            GaussianDistance gDistance = new GaussianDistance();
            Distance distance = new Distance();
```

```
                System.out.println("Euclidean distance :
                "+distance.calDistance(data[0],data[1]));
                System.out.println("Gaussian distance :
                "+gDistance.calGaussianDistance(data[0],data[1],0.5f));
                }
        }
```

[15] Source code TestGaussianSearch.java

```
//*************************************************//
//File name: TestGaussianSearch.java    //
//*************************************************//
      class TestGaussianSearch
      {
        public static void main(String args[])
        {
                int numOfResult = 15;
                int[] result    = new int[numOfResult];
                int[] relevanceIndex = {46,66,803,815};
                ImageSearch database = new ImageSearch();
                result = database.gaussianSearch(relevanceIndex,numOfResult);
                for(int i=0;i<result.length;i++)
    System.out.println(result[i]+" image name "+database.getImageNameAt (result[i]));
                }
        }
```

[16] Source code TestLeastSorting.java

```
//************************************************//
//File name: TestLeastSorting.java              //
//************************************************//
    class TestLeastSorting
    {
      public static void main(String args[])
      {
            int[] data = {8,3,7,6,8,2,4,5,6,1,0};
            int[] resultX = new int[4];
            resultX = leastSort(data,4);
            for(int i=0;i<data.length;i++)
            {
                //System.out.println(" "+data[resultX[i]]);
                System.out.println(data[i]);
                }
            }
    public static int[] leastSort(int[] kkk,int num)
    {
            int tmp;
            int[] result = new int[num];

            for(int i=0;i<num;i++)
            {
                  tmp = 100 ;
                  for(int j=0;j<kkk.length;j++)
                  {
                        if(tmp>kkk[j])
                        {
                              tmp = kkk[j];
                              result[i] = j;
```

```
                                        }
                                    }
                                kkk[result[i]] = 100;
                                }
                        return result;
                        }
                }
```

[17] Source code TestMaxSorting.java

```
//*****************************************//
//File name: TestMaxSorting.java        //
//*****************************************//
        class TestMaxSorting
        {
          public static void main(String args[])
          {
                int numOfResult = 5;
                int[] result = new int[numOfResult];
                float[] data = {0.1f,0.5f,0.4f,0.15f,0.87f,0.95f,0.65f,0.26f,0.43f,0.25f};
                result = maxSorting(data,numOfResult);
                for(int i=0;i<result.length;i++)
                        System.out.println(result[i]);
                }
        static int[] maxSorting(float[] Xdata,int num)
        {
           float tmp;
                int[] result = new int[num];

                for(int i=0;i<num;i++)
                {
                        tmp = 0.0f ;
```

```
                    for(int j=0;j<Xdata.length;j++)
                    {
                        if(tmp<Xdata[j])
                        {
                            tmp = Xdata[j];
                            result[i] = j;
                            }
                        }
                    Xdata[result[i]] = 0.0f;
                    }
                return result;
            }
        }
```

[18] Source code TestSimpleSearch.java

```
//**********************************************//
//File name: TestSimpleSearch.java        //
//**********************************************//
    class TestSimpleSearch
    {
      public static void main(String args[])
      {
            int numOfResult = 10;
            int[] result = new int[numOfResult];
            ImageSearch database = new ImageSearch();
            result = database.simpleSearch(0,numOfResult);
            for(int i=0;i<numOfResult;i++)
            { System.out.println(""+result[i]+" "+database.getImageNameAt(result[i]));
                }
            }
        }
```

[19] Source code CBIRBean.java

```
/**
 * Created by IntelliJ IDEA.
 * User: K. Tavee
 * Date: 28 ก.ย. 2546
 * Time: 23:20:35
 */
package searchBean;

import java.sql.*;
import java.util.Random;
import java.util.Arrays;

public class CBIRBean {


  private  int featureSize = 115;                  //Size of an image feature
  private  int numOfImage  = 35000;          //Size of an image database
  private  int numOfResult = 27;            //Num of image to display in JSP
  /* An array for collect feature data */
  private  float[][] imageFeature = new float[numOfImage][featureSize];
  /* An array for clollect image name */
  private  String[] imageName = new String[numOfImage];
  private Connection con;
        private ResultSet rs1,rs2;

  /* Load data base */
  public CBIRBean() {
    /* Jdbc Driver loading */
    try {
       //Class.forName("COM.cloudscape.core.RmiJdbcDriver");
```

```
            Class.forName("COM.cloudscape.core.JDBCDriver");
            //Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          }catch(ClassNotFoundException ex) {
            System.out.println(ex.getMessage());
            }
        /* Connect to database and retrieve data */
        try {
          /* Make a connection */
          //con =
DriverManager.getConnection("jdbc:cloudscape:rmi://192.168.0.5:1099//root/Database/
CBIR_database");
          con =
DriverManager.getConnection("jdbc:cloudscape://root/Database/CBIR_database");
          //con = DriverManger.getConnection("jdbc:odbc:CBIR_databse");
          /* Create Statement */
          Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,

ResultSet.CONCUR_READ_ONLY);
          System.out.println("Use "+numOfImage+" images.");
          /* Retrieve image feature data from database */
          rs1 = stmt.executeQuery("SELECT * FROM IMAGEFEATURE ");
          rs1.first();
          System.out.println("Collecting image feature...");
          for(int i=0; i<numOfImage; i++) {
            for(int j=0; j<featureSize; j++) {
              imageFeature[i][j] = rs1.getFloat((j+1));
              }
            rs1.next();
            }
          rs1.close();
```

```
            System.out.println("Collecting image name...");
            rs2 = stmt.executeQuery("SELECT * FROM IMAGELIST");
            rs2.first();
            for(int j=0;j<numOfImage;j++) {
               imageName[j] = rs2.getString(1);
               rs2.next();
               }
            System.out.println("Complete to collecting data...");
            System.out.println(numOfImage+" images are ready to use...");
            rs2.close();
            con.close();
            }catch(SQLException ex) {
               System.out.println(ex.getMessage());
                     }
}

/**
 * Simple Search method
 * @param query
 * return real index of result image
 */
public int[] simpleSearch(int query) {
   /**
    * A query is index of image feature that user select
    * A numOfResult is a number of images that require
    * after executed simple search
    */
   float[] euclideanDistanceValue = new float[numOfImage];

   System.out.println("Simple Search executing.....");
   /* Calculate Euclicean distance of all image feature */
```

```
    for(int i=0;i<numOfImage;i++) {
       euclideanDistanceValue[i] =
       calDistance(imageFeature[query],imageFeature[i]);
       }
    /* Sort an array of distance value and return*/
    return(leastRanking (euclideanDistanceValue,numOfResult));
    }

  /**
  * Gaussian Search method
  * @param relevanceIndex
  * return  real index of result image
  */
public int[] gaussianSearch(int[] relevanceIndex) {
     float[] gaussianDistance = new float[relevanceIndex.length];
     float[] sigmaValue       = new float[relevanceIndex.length];
     float[] gaussianValue = new float[numOfImage];

     System.out.println("Gaussian Search executing.....");
     sigmaValue = findSigma(relevanceIndex);
     /* calculate Gaussian distance */
     for(int i=0;i<numOfImage;i++) {
        for(int j=0;j<relevanceIndex.length;j++) {
           gaussianDistance[j] =
              calGaussianDistance(imageFeature[relevanceIndex[j]],
                                  imageFeature[i],sigmaValue[j]);
           }

        /* sum all Gaussian distance of each image */
        float tmp = 0.0f;
        for(int t=0;t<relevanceIndex.length;t++)
```

```
            tmp+=gaussianDistance[t];
        gaussianValue[i] = tmp;
        }
    return(maxRanking(gaussianValue,numOfResult));
    }

 /**
* cal SIGMA value of Gaussian equation
* @param relevanceIndex
* @return SIGMA value of each new center
*/
 protected float[] findSigma(int[] relevanceIndex) {
        float[] sigmaValue = new float[relevanceIndex.length];
        float[] tmp        = new float[relevanceIndex.length];

        Arrays.fill(tmp,99);
        for(int i=0;i<relevanceIndex.length;i++) {
           for(int j=0;j<relevanceIndex.length;j++) {
              if(i!=j) {
                 /* find Euclidean distance */
                 tmp[j] = calDistance(imageFeature[relevanceIndex[i]],
                                      imageFeature[relevanceIndex[j]]);
                 }
              }
           /* find 'n' least data */
           float xxx = tmp[0];
           for(int t=0;t<tmp.length;t++) {
              if(xxx>tmp[t])
                 xxx = tmp[t];
              }
           sigmaValue[i] = (xxx/2);
```

```
            }
        return sigmaValue;
    }

    /**
     * Cal Gaussian distance
     * @param feature1
     * @param feature2
     * @param sigmaValue
     * @return float value of Gaussian distance between any feature and a new center
     */
    public float calGaussianDistance(float[] feature1,float[] feature2,float sigmaValue) {
                float gaussianDistance;
                float a = (float)Math.pow(calDistance(feature1,feature2),2);
                float b = 2*(float)Math.pow(sigmaValue,2);
                gaussianDistance = (float)Math.exp(-a/b);

                return gaussianDistance;
                }

    /**
     * least to max ranking
     * @param Xdata
     * @param num
     * @return array of result index
     */
    protected int[] leastRanking(float[] Xdata,int num) {
                float tmp;
                int[] result = new int[num];

                for(int i=0; i<num; i++) {
```

```
                tmp = 100f ;
                for(int j=0; j<Xdata.length; j++) {
                        if(tmp > Xdata[j]) {
                                tmp = Xdata[j];
                                result[i] = j;          //Collect an index of image
                                }
                        }
                Xdata[result[i]] = 100;
                }
            return result;
            }

/**
 * max to least ranking
 * @param Xdata
 * @param num
 * @return array of result index
 */
protected int[] maxRanking(float[] Xdata,int num) {
  float tmp;
  int[] result = new int[num];

  for(int i=0;i<num;i++) {
    tmp = 0.0f ;
    for(int j=0;j<Xdata.length;j++) {
      if(tmp<Xdata[j]) {
        tmp = Xdata[j];
        result[i] = j;  //Collect an index of image
        }
      }
    Xdata[result[i]] = 0.0f;
```

```
        }
    return result;
    }

/**
 * Cal Ucledean distance
 * @param Vect1
 * @param Vect2
 * @return float value of Euclidean distance between Vect1 and Vect2
 */
public float calDistance(float[] Vect1,float[] Vect2) {
            /* result */
  float Result = 0.0f;
            /* Both of Vect1 and Vect2 must same size. */
            if(Vect1.length!=Vect2.length) {
                System.out.println("Error! the two vector must be same size.");
                return -1.00f;                    //If Error then return -1
                }
            else {
                /**
                 *      Eucledian distance method
                 */
                for(int j=0;j<Vect1.length;j++)
                        Result+=Math.pow((Vect1[j]-Vect2[j]),2);
                return (float)Math.sqrt(Result);      // Return result
                }
        }

/**
* get data at specific 'column' and 'row'
* @param column
```

```
     * @param row
     * @return float value of data at 'column' and 'row'
     */
    public float getImageFeature(int column,int row) {
                    return imageFeature[column][row];
                    }

    /**
     * get data at specific 'row'
     * @param indexOfRow
     * @return feature array of an image
     */
    public float[] getImageFeature(int indexOfRow) {
            return imageFeature[indexOfRow];
            }

    /**
     * get name of image at 'index'
     * @param index
     * @return name string of image
     */
    public String getImageName(int index) {
            return imageName[index];
            }

    /**
     * get num of image
     * @return  int value of total number of image
     */
    public int getNumOfImage() {
       return numOfImage;
```

```
        }

    /**
     * get random index
     * @return array of random number
     */
    public int[] getRdmImage() {
        Random r = new Random();
        int[] rdmNum = new int[numOfResult];
        for(int i=0; i<numOfResult; i++) {
            rdmNum[i] = Math.abs((r.nextInt()%numOfImage));
        }
        return(rdmNum);
    }
}
```

[20] Source code GuassianSearch.jsp

```
<%@ page contentType="text/html; charset=ISO-8859-1"%>
<html>
<body>
  <jsp:useBean id="bean" class="searchBean.CBIRBean" scope="application"/>
  <%!
    int[]   index;
    int[]   result;
  %>
  <form Method='POST' Action='GaussianSearch.jsp'>
  <%
    String[] input = request.getParameterValues("feedback");
    index = new int[input.length];
    for(int i=0; i<input.length; i++) {
      index[i] = Integer.parseInt(input[i]);
```

```
            }
            result = bean.gaussianSearch(index);
            out.println("<center><h2>Gaussian Search</h2></center>");
            out.println("<center><Table border=1 heigth='80%' width='100%' >");
            int counter=0;
            for(int i=0; i<3; i++){
               out.println("<tr>");
               for(int j=0; j<9; j++) {
                  out.println("<td>");
                  String imgName = bean.getImageName(result[counter]);
                  out.println("<img src='/corelImage/"+imgName+"' width='100%'><br>");
                  out.println("<input type='checkbox' name=" +
                        "'feedback' value='"+result[counter]+"'>"+imgName);
                  out.println("</td>");
                  counter++;
               }
               out.println("</tr>");
            }
            out.println("</Table></center><br>");
            out.println("<center><input type='submit' name='submit'
value='Feedback'></center>");
      %>
      </form>
   </body>
</html>
```

[21] Source code idex.jsp

```
<!-- index.jsp ->
<!-- forward to simpleSearch.jsp ->
<%@ page contentType="text/html; charset=ISO-8859-1"%>
<html>
<body>
   <jsp:forward page="Selectquery.jsp"/>
</body>
</html>
```

[22] Source code Selectquery.jsp

```
<%@ page contentType="text/html; charset=ISO-8859-1"%>
<html>
<body>
   <jsp:useBean id="bean" class="searchBean.CBIRBean" scope="application"/>
   <%!
      int result[];
   %>
   <%
      result = bean.getRdmImage();
   %>
   <form Method='POST' Action='SimpleSearch.jsp'>
   <%
      out.println("<center><h2>Select
query"+bean.getNumOfImage()+"</h2></center>");
      out.println("<center><Table border=1 heigth='80%' width='100%' >");
      int counter=0;
      for(int i=0; i<3; i++){
         out.println("<tr>");
         for(int j=0; j<9; j++) {
            out.println("<td>");
```

```
                String imgName = bean.getImageName(result[counter]);
                out.println("<img src='/corelImage/"+imgName+"' width='100%'><br>");
                out.println("<input type='radio' name=" +
                    "'query' value='"+result[counter]+"'>"+imgName);
                out.println("</td>");
                counter++;
            }
            out.println("</tr>");
        }
        out.println("</Table></center><br>");
        out.println("<center><input type='submit' name='submit' value=' submit
'></center>");
    %>
    </form>
    <form Method='POST' Action='Selectquery.jsp'>
      <center><input type='submit' name='radom' value='Random'></center>
    </form>
</body>
</html>
```

[23] Source code SimpleSearch.jsp

```
<%@ page contentType="text/html; charset=ISO-8859-1"%>
<html>
<body>
  <jsp:useBean id="bean" class="searchBean.CBIRBean" scope="application"/>
  <%!
     int[] result;
  %>
  <form Method='POST' Action='GaussianSearch.jsp'>
  <%
     String query = request.getParameter("query");
```

```
        result = bean.simpleSearch(Integer.parseInt(query));
        out.println("<center><h2>Simple Search</h2></center>");
        out.println("<center><Table border=1 heigth='80%' width='100%' >");
        int counter=0;
        for(int i=0; i<3; i++){
          out.println("<tr>");
          for(int j=0; j<9; j++) {
            out.println("<td>");
            String imgName = bean.getImageName(result[counter]);
            out.println("<img src='/corelImage/"+imgName+"' width='100%'><br>");
            out.println("<input type='checkbox' name=" +
                "'feedback' value='"+result[counter]+"'>"+imgName);
            out.println("</td>");
            counter++;
          }
          out.println("</tr>");
        }
        out.println("</Table></center><br>");
        out.println("<center><input type='submit' name='submit'
value='Feedback'></center>");
    %>
    </form>
</body>
</html>
```

[24] Source code CBIRBean.java (ใช้วิธี Positive feedback และ Negative feedback)

```
/**
 * Created by IntelliJ IDEA.
 * User: K. Tavee
 * Date: 28 ก.ย. 2546
 * Time: 23:20:35
 */
package searchBean;

import java.sql.*;
import java.util.Random;
import java.util.Arrays;

public class CBIRBean {


   private  int featureSize = 115;                  //Size of an image feature
   private  int numOfImage  = 35000;          //Size of an image database
   private  int numOfResult = 27;             //Num of image to display in JSP
   /* An array for collect feature data */
   private  float[][] imageFeature = new float[numOfImage][featureSize];
   /* An array for clollect image name */
   private  String[] imageName = new String[numOfImage];
   private Connection con;
       private ResultSet rs1,rs2;


   /* Load data base */
   public CBIRBean() {
      /* Jdbc Driver loading */
      try {
         //Class.forName("COM.cloudscape.core.RmiJdbcDriver");
```

```
        Class.forName("COM.cloudscape.core.JDBCDriver");
        //Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      }catch(ClassNotFoundException ex) {
        System.out.println(ex.getMessage());
        }
/* Connect to database and retrieve data */
try {
  /* Make a connection */
  //con =
DriverManager.getConnection("jdbc:cloudscape:rmi://192.168.0.5:1099//root/Data
base/CBIR_database");
  con =
DriverManager.getConnection("jdbc:cloudscape://root/Database/CBIR_database")
;
  //con = DriverManger.getConnection("jdbc:odbc:CBIR_databse");
  /* Create Statement */
  Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,

ResultSet.CONCUR_READ_ONLY);
   System.out.println("Use "+numOfImage+" images.");
  /* Retrieve image feature data from database */
  rs1 = stmt.executeQuery("SELECT * FROM IMAGEFEATURE ");
  rs1.first();
  System.out.println("Collecting image feature...");
  for(int i=0; i<numOfImage; i++) {
    for(int j=0; j<featureSize; j++) {
      imageFeature[i][j] = rs1.getFloat((j+1));
      }
    rs1.next();
    }
  rs1.close();
```

```
        System.out.println("Collecting image name...");
        rs2 = stmt.executeQuery("SELECT * FROM IMAGELIST");
        rs2.first();
        for(int j=0;j<numOfImage;j++) {
           imageName[j] = rs2.getString(1);
           rs2.next();
           }
        System.out.println("Complete to collecting data...");
        System.out.println(numOfImage+" images are ready to use...");
        rs2.close();
        con.close();
        }catch(SQLException ex) {
           System.out.println(ex.getMessage());
        }
}

/**
 * Simple Search method
 * @param query
 * return real index of result image
 */
public int[] simpleSearch(int query) {
   /**
    * A query is index of image feature that user select
    * A numOfResult is a number of images that require
    * after executed simple search
    */
   float[] euclideanDistanceValue = new float[numOfImage];

   System.out.println("Simple Search executing.....");
```

```
    /* Calculate Euclicean distance of all image feature */
    for(int i=0;i<numOfImage;i++) {
       euclideanDistanceValue[i] =
       calDistance(imageFeature[query],imageFeature[i]);
       }
    /* Sort an array of distance value and return*/
    return(leastRanking (euclideanDistanceValue,numOfResult));
    }

  /**
  * Gaussian Search method
  * @param relevanceIndex
  * return  real index of result image
  */
 public int[] gaussianSearch(int[] relevanceIndex) {
      float[] gaussianDistance = new float[relevanceIndex.length];
      float[] sigmaValue       = new float[relevanceIndex.length];
      float[] gaussianValue = new float[numOfImage];

      System.out.println("Gaussian Search executing.....");
      sigmaValue = findSigma(relevanceIndex);
      /* calculate Gaussian distance */
      for(int i=0;i<numOfImage;i++) {
         for(int j=0;j<relevanceIndex.length;j++) {
            gaussianDistance[j] =
               calGaussianDistance(imageFeature[relevanceIndex[j]],
                                   imageFeature[i],sigmaValue[j]);
            }

         /* sum all Gaussian distance of each image */
         float tmp = 0.0f;
```

```
        for(int t=0;t<relevanceIndex.length;t++)
            tmp+=gaussianDistance[t];
        gaussianValue[i] = tmp;
        }
    return(maxRanking(gaussianValue,numOfResult));
    }

 /**
* cal SIGMA value of Gaussian equation
* @param relevanceIndex
* @return SIGMA value of each new center
*/
 protected float[] findSigma(int[] relevanceIndex) {
        float[] sigmaValue = new float[relevanceIndex.length];
        float[] tmp        = new float[relevanceIndex.length];

        Arrays.fill(tmp,99);
        for(int i=0;i<relevanceIndex.length;i++) {
          for(int j=0;j<relevanceIndex.length;j++) {
            if(i!=j) {
              /* find Euclidean distance */
              tmp[j] = calDistance(imageFeature[relevanceIndex[i]],
                                   imageFeature[relevanceIndex[j]]);
              }
            }
          /* find 'n' least data */
          float xxx = tmp[0];
          for(int t=0;t<tmp.length;t++) {
            if(xxx>tmp[t])
              xxx = tmp[t];
            }
```

```
            sigmaValue[i] = (xxx/2);
            }
        return sigmaValue;
    }

/**
 * Cal Gaussian distance
 * @param feature1
 * @param feature2
 * @param sigmaValue
 * @return float value of Gaussian distance between any feature and a new center
 */
public float calGaussianDistance(float[] feature1,float[] feature2,float sigmaValue) {
        float gaussianDistance;
        float a = (float)Math.pow(calDistance(feature1,feature2),2);
        float b = 2*(float)Math.pow(sigmaValue,2);
        gaussianDistance = (float)Math.exp(-a/b);

        return gaussianDistance;
        }

/**
 * least to max ranking
 * @param Xdata
 * @param num
 * @return array of result index
 */
protected int[] leastRanking(float[] Xdata,int num) {
        float tmp;
        int[] result = new int[num];
```

```
    for(int i=0; i<num; i++) {
            tmp = 100f ;
            for(int j=0; j<Xdata.length; j++) {
                    if(tmp > Xdata[j]) {
                            tmp = Xdata[j];
                            result[i] = j;          //Collect an index of image
                            }
                    }
            Xdata[result[i]] = 100;
            }
    return result;
    }

/**
 * max to least ranking
 * @param Xdata
 * @param num
 * @return array of result index
 */
protected int[] maxRanking(float[] Xdata,int num) {
    float tmp;
    int[] result = new int[num];

    for(int i=0;i<num;i++) {
        tmp = 0.0f ;
        for(int j=0;j<Xdata.length;j++) {
            if(tmp<Xdata[j]) {
                tmp = Xdata[j];
                result[i] = j;   //Collect an index of image
                }
            }
```

```
            Xdata[result[i]] = 0.0f;
            }
        return result;
        }

/**
 * Cal Ucledean distance
 * @param Vect1
 * @param Vect2
 * @return float value of Euclidean distance between Vect1 and Vect2
 */
public float calDistance(float[] Vect1,float[] Vect2) {
        /* result */
    float Result = 0.0f;
        /* Both of Vect1 and Vect2 must same size. */
        if(Vect1.length!=Vect2.length) {
                System.out.println("Error! the two vector must be same size.");
                return -1.00f;                          //If Error then return -1
                }
        else {
                /**
                 *      Eucledian distance method
                 */
                for(int j=0;j<Vect1.length;j++)
                        Result+=Math.pow((Vect1[j]-Vect2[j]),2);
                return (float)Math.sqrt(Result);        // Return result
                }
        }

/**
* get data at specific 'column' and 'row'
```

```
 * @param column
 * @param row
 * @return float value of data at 'column' and 'row'
 */
public float getImageFeature(int column,int row) {
        return imageFeature[column][row];
        }

/**
 * get data at specific 'row'
 * @param indexOfRow
 * @return feature array of an image
 */
public float[] getImageFeature(int indexOfRow) {
    return imageFeature[indexOfRow];
    }

/**
 * get name of image at 'index'
 * @param index
 * @return name string of image
 */
public String getImageName(int index) {
    return imageName[index];
    }

/**
 * get num of image
 * @return  int value of total number of image
 */
public int getNumOfImage() {
```

```
        return numOfImage;
        }

    /**
     * get random index
     * @return array of random number
     */
    public int[] getRdmImage() {
        Random r = new Random();
        int[] rdmNum = new int[numOfResult];
        for(int i=0; i<numOfResult; i++) {
            rdmNum[i] = Math.abs((r.nextInt()%numOfImage));
        }
        return(rdmNum);
    }
}
```

ประวัติผู้เขียนโครงงาน

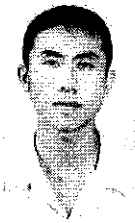
ชื่อ นาย ทวี คุณบิดา

ภูมิลำเนา 76 ม.4 ต.ทุ่งนาเลา อ.คอนสาร จ.ชัยภูมิ

ประวัติการศึกษา

- จบมัธยมศึกษาตอนปลายจาก โรงเรียนคอนสาร วิทยาคม จ.ชัยภูมิ
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรี ชั้นปีที่ 4 สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

Email: Taveek@hotmail.com



ชื่อ นายวัฒนพงศ์ เตียมแสง

ภูมิลำเนา 297/1 ม.10 ต.ห้วยข้าวก่ำ อ.จุน จ.พะเยา

ประวัติการศึกษา

- จบมัธยมศึกษาจากโรงเรียน จุนวิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรี ชั้นปีที่ 4 สาขาวิศวกรรม คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

Email: w_thiamsang@hotmail.com

ชื่อ นาย เฉลิมขวัญ ลาสอน

ภูมิลำเนา 380 ม.1 ต.นิเวศน์ อ.ธวัชบุรี จ.ร้อยเอ็ด

ประวัติการศึกษา

- จบมัธยมศึกษาจากโรงเรียน ร้อยเอ็ดวิทยาลัย
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรี ชั้นปีที่ 4 สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

Email: The_mixer6@hotmail.com