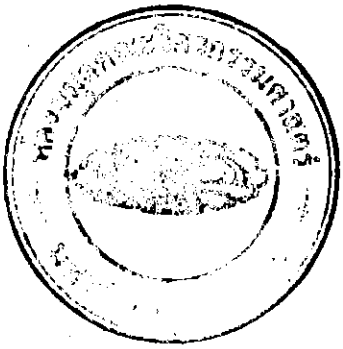


การค้นหาวีดิโอบนเครือข่ายเพียร์ทูเพียร์โดยใช้โมเดลการปรับตัวอัตโนมัติ

Content Based Video retrieval on P2P Network by using adaptive model

นายกฤษฎา ศตกาญจน์ รหัส 48364623

นายมงคล อักโข รหัส 48364883

นายแสวงชัย สาโรจน์ รหัส 48365088

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2551

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงการ	การค้นหาวีดิโอบนเครือข่ายเพียร์ทูเพียร์โดยใช้โมเดลการปรับตัวอัตโนมัติ		
ผู้ดำเนินโครงงาน	นายกฤษฎา	ศตกาญจน์	รหัส 48364623
	นายมงคล	อักโข	รหัส 48364883
	นายแสวงชัย	สาโรจน์	รหัส 48365088
อาจารย์ที่ปรึกษา	ดร.ไพศาล	มุณีสว่าง	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2551		

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ดร.ไพศาล มุณีสว่าง)

.....กรรมการ

(ดร.พนมขวัญ ริยะมงคล)

.....กรรมการ

(ดร.สุรเดช จิตประไพกุลศาล)

หัวข้อโครงงาน	การค้นหาวีดิโอบนเครือข่ายเพียร์ทูเพียร์โดยใช้โมเดลการปรับตัวอัตโนมัติ		
ผู้ดำเนินโครงงาน	นายกฤษฎา	ศตกาญจน์	รหัส 48364623
	นายมงคล	อักโข	รหัส 48364883
	นายแสวงชัย	สาโรจน์	รหัส 48365088
อาจารย์ที่ปรึกษา	ดร. ไพศาล	มุณีสว่าง	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2551		

บทคัดย่อ

โครงงานนี้ได้กล่าวถึงการพัฒนาโปรแกรมที่มีความสามารถในการค้นหาไฟล์วีดีโอโดยใช้ไฟล์วีดีโอตัวอย่างบนระบบเครือข่ายเพียร์ทูเพียร์ ซึ่งไฟล์วีดีโอที่จะนำมาใช้ในระบบต้องถูกทำดัชนีตามหลักการทำดัชนีแบบ AVI และมีภาพของไฟล์วีดีโอที่ใช้สำหรับแสดงผลลัพธ์ ในการค้นหาใช้หลักการ Cosine Similarity โดยการนำเวกเตอร์ที่ได้จากการทำดัชนีมาทำการเปรียบเทียบความเหมือน และเพื่อให้ได้ผลลัพธ์ที่มีคุณภาพมากขึ้นจึงต้องมีการปรับปรุงค่าเวกเตอร์ของวีดีโอต้นแบบที่ใช้ในการค้นหา ซึ่งการปรับปรุงค่าเวกเตอร์นี้จะทำการปรับปรุงเพื่อให้ได้เวกเตอร์ใหม่ที่มีประสิทธิภาพแบบอัตโนมัติ โดยที่โครงงานนี้พัฒนาด้วยภาษาจาวา (Java) และใช้ NetBeans IDE 6.5 เป็นเครื่องมือในการพัฒนา

ผลที่ได้จากการทำโครงงานนี้ คือ สามารถพัฒนาแอพลิเคชั่นที่สามารถค้นหาไฟล์วีดีโอโดยใช้ไฟล์วีดีโอตัวอย่างบนระบบเครือข่ายเพียร์ทูเพียร์ได้อย่างถูกต้องและมีประสิทธิภาพ

Project Title	Content Based Video Retrieval on P2P Network by Using Adaptive Model
Name	Mr. Krisda Satakarn ID. 48364623
	Mr. Mongkol Akko ID. 48364883
	Mr. Sawangchai Saroj ID. 48365088
Project Advisor	Paisarn Muneesawang, Ph.D.
Major	Computer Engineering.
Department	Electrical and Computer Engineering.
Academic Year	2551

ABSTRACT

This project develops video retrieval on peer-to-peer network using an adaptive model and AVI index encoding of video clips. The searching is based on the Cosine Similarity measure between indexed vectors to compare for content similarity. The adaptive system adjusts query vector automatically to improve retrieval accuracy. This application was developed by Socket Java API and NetBeans IDE 6.5.

This project is properly successfully done with a quality satisfying application for searching video clips on a peer-to-peer network.

กิตติกรรมประกาศ

โครงงานวิศวกรรมคอมพิวเตอร์สำเร็จลุล่วงได้ด้วยดี เนื่องด้วยความอนุเคราะห์จากท่านอาจารย์ที่ปรึกษาโครงงาน คือ ดร.ไพศาล มุณีสว่าง รวมถึงท่านคณะกรรมการทั้ง 2 ท่าน คือ ดร.สุรเดช จิตประไพกุลศาล และ ดร.พนมขวัญ ริยะมงคล ที่ได้ให้แนวคิดและคำแนะนำในการแก้ไขปัญหาต่างๆ ตลอดจนได้สละเวลาอันมีค่าเพื่อตรวจสอบและแก้ไขข้อบกพร่องต่างๆ อีกทั้งยังสอนให้รู้จักการวางแผนการทำงาน และขอขอบคุณภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ที่ได้ให้เงินสนับสนุนการทำโครงงาน

ในโอกาสนี้ทางคณะผู้จัดทำโครงงานจึงขอขอบพระคุณทุกๆ ท่านที่มีส่วนร่วมในการทำโครงงานนี้ตลอดจนผู้เขียน ผู้คิดค้นทฤษฎีต่างๆ ที่โครงงานฉบับนี้ได้นำความรู้ที่ได้มาพัฒนาระบบทำให้โครงงานนี้สำเร็จลุล่วงไปด้วยดี

นายกฤษฎา ศตกาญจน์
นายมงคล อักโข
นายแสวงชัย สาโรจน์

สารบัญ

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ในปัจจุบันการค้นหาข้อมูลบนอินเตอร์เน็ตมีบทบาท และได้รับความสนใจเป็นอย่างมาก ไม่ว่าจะเป็นการค้นหาข้อมูล เว็บไซต์ รูปภาพ และอื่นๆ ซึ่งวีดิโอเป็นข้อมูลประเภทหนึ่งที่ได้รับความนิยมในการค้นหา โดยส่วนใหญ่จะใช้คำสำคัญในการค้นหาข้อมูลวีดิโอที่ต้องการ แต่ผลลัพธ์ที่ได้อาจไม่ตรงตามความต้องการ อันเนื่องมาจากคำสำคัญ (Keyword) ที่ใช้ค้นหาไม่ตรงกับคำอธิบายหรือคำที่เกี่ยวข้องกับวีดิโอที่ต้องการค้นหา และข้อมูลนั้นถูกเก็บแบบรวมศูนย์ (Centralize Database) ทำให้ไม่มีความหลากหลายในการค้นหาข้อมูล เช่น ถ้าใช้บริการการค้นหาของกูเกิ้ล (Google) ข้อมูลที่ได้จะมาจากฐานข้อมูลของกูเกิ้ลเท่านั้น แต่ข้อมูลวีดิโออาจถูกเก็บไว้ในเครื่องของผู้ใช้รายอื่น นอกจากนั้นการเก็บข้อมูลแบบรวมศูนย์ อาจทำให้ระบบล่มได้เนื่องจากปัญหาคอขวด (Bottleneck) ซึ่งปัญหานี้จะเกิดเมื่อมีการค้นหาข้อมูลโดยผู้ใช้จำนวนมาก

การเก็บข้อมูลแบบรวมศูนย์นั้น เป็นการเก็บข้อมูลรวบรวมไว้อยู่ในฐานข้อมูลเดียวเมื่อผู้ใช้งานทำการค้นหาข้อมูลก็จะทำการติดต่อไปยังฐานข้อมูลนั้นเท่านั้น ซึ่งเมื่อมีผู้ใช้งานหลายคนอาจจะทำให้สร้างภาระให้กับฐานข้อมูลเป็นอย่างมาก เนื่องจากมีการทำการติดต่อมายังฐานข้อมูลนี้มาก อีกทั้งยังไม่มีความหลากหลายของข้อมูล แต่ข้อดี คือ สะดวกต่อการเก็บข้อมูลเพราะจะทำการเพิ่ม ลบ หรือเปลี่ยนแปลงข้อมูลได้ง่ายในฐานข้อมูลนั้น และยังสะดวกต่อการบำรุงรักษาเนื่องจากข้อมูลอยู่ในเครื่องเซิร์ฟเวอร์เดียว ส่วนการเก็บข้อมูลแบบกระจาย (Distributed Database) นั้นเป็นการเก็บข้อมูลที่ไม่ได้เก็บไว้ในฐานข้อมูลเพียงฐานข้อมูลเดียวเท่านั้น แต่ถูกเก็บไว้ในหลายแห่ง ซึ่งส่งผลให้แต่ละฐานข้อมูลมีการทำงานที่เป็นอิสระต่อกันหากข้อมูลบางส่วนเสียหายจะไม่ส่งผลกระทบต่อข้อมูลส่วนอื่น อีกทั้งยังมีประสิทธิภาพสูงเนื่องจากฐานข้อมูลมีอยู่หลายแห่งทำให้ผู้ใช้สามารถติดต่อฐานข้อมูลหลายแห่งในเวลาเดียวกันเพราะมีการทำงานที่เป็นอิสระต่อกัน นอกจากนั้นการเก็บข้อมูลแบบกระจายทำให้ประหยัดค่าใช้จ่ายเมื่อเทียบกับเครื่องคอมพิวเตอร์ขนาดใหญ่และง่ายต่อการขยายระบบในอนาคต เหมาะสำหรับข้อมูลที่มีการเปลี่ยนแปลงตลอดเวลา แต่ข้อเสียคือมีความซับซ้อนมากกว่าแบบรวมศูนย์ และยากต่อการดูแลความปลอดภัยของข้อมูล

การค้นหาข้อมูลโดยส่วนใหญ่แล้วจะใช้คำสำคัญ ในการค้นหาวีดิโอไฟล์ ซึ่งก่อนการค้นหาข้อมูลโดยใช้คำสำคัญ ต้องมีการทำดัชนี (Index) ให้แก่ข้อมูลนั้นก่อนจึงสามารถนำมาใช้ค้นหาข้อมูลได้ ข้อดีคือง่ายต่อการระบุคำสำคัญและ มีความรวดเร็วในการค้นหาเพราะมีการทำดัชนีอีกทั้งยังเป็นการเปรียบเทียบระหว่างตัวอักษรกับตัวอักษรเท่านั้น ส่วนข้อเสียคือเสียเวลาในการทำดัชนีและ ข้อมูลที่ได้มีความเที่ยงตรงแต่ขาดความแม่นยำ เพราะคำที่ระบุถึงวีดิโอกับเนื้อหาในวีดิโอ

อาจไม่ตรงกัน หรือหากไม่มีคำที่อธิบายถึงวีดิโอดังกล่าวก็จะไม่สามารถทำการค้นหาได้ ซึ่งแตกต่างจากการค้นหาโดยใช้ตัวอย่างไฟล์วีดิโอ (CBVR: Content Based Video Retrieval) ที่มีการเปรียบเทียบสี รูปร่างและ รายละเอียดของไฟล์วีดิโอ ทำให้การค้นหามีความเที่ยงตรงและแม่นยำอีกทั้งมีการทำดัชนีแบบอัตโนมัติทำให้การทำดัชนีมีความรวดเร็วมากขึ้น

ส่วนโมเดลการปรับตัวอัตโนมัติ (Adaptive Model) คือโมเดลที่จะช่วยปรับปรุงคุณภาพของการค้นหาโดยการนำผลลัพธ์ในการค้นหาของและรอบมาปรับปรุงสำหรับการค้นหาในรอบต่อไปเพื่อให้ได้ผลลัพธ์สุดท้ายที่มีความถูกต้องแม่นยำมากขึ้น โดยอาศัยหลักการปัญญาประดิษฐ์ (Artificial intelligent) มาใช้ในโมเดลเพื่อให้สามารถมีการปรับตัวแบบอัตโนมัติ

ดังนั้นโครงงานการค้นหาวีดิโอบนเครือข่ายเพียร์ทูเพียร์ โดยใช้โมเดลการปรับตัวอัตโนมัติจึงได้นำหลักการการเก็บข้อมูลแบบกระจายมาใช้เพื่อแก้ปัญหาของระบบการเก็บข้อมูลแบบรวมศูนย์และใช้ตัวอย่างไฟล์วีดิโอในการค้นหาไฟล์วีดิโอ โดยอาศัยโมเดลการปรับตัวแบบอัตโนมัติเพื่อให้การค้นหาไฟล์วีดิโอมีประสิทธิภาพมากขึ้น

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อสร้างแอพพลิเคชั่นที่ใช้ค้นหาไฟล์วีดิโอบนเครือข่ายเพียร์ทูเพียร์

1.2.2 เพื่อศึกษาการเปรียบเทียบระหว่างไฟล์วีดิโอ

1.2.3 เพื่อศึกษาการเก็บข้อมูลแบบกระจาย

1.2.4 เพื่อศึกษาการสร้างโมเดลการปรับตัวแบบอัตโนมัติ โดยใช้หลักการปัญญาประดิษฐ์

1.3 ขอบข่ายโครงงาน

1.3.1 สร้างซอฟต์แวร์ที่รองรับการค้นหาไฟล์วีดิโอบนเครือข่ายเพียร์ทูเพียร์

1.3.2 ซอฟต์แวร์มีโมเดลในการปรับปรุงตนเองเพื่อให้ได้ผลลัพธ์ที่มีความถูกต้อง

1.3.3 ซอฟต์แวร์มีความสามารถในการค้นหาได้อย่างถูกต้อง โดยใช้หลักการ Query-by-Example

1.3.4 ข้อมูลวีดิโอที่นำมาใช้งานทั้งหมดมีการทำดัชนีโดยใช้หลักการ AVI เรียบร้อยแล้ว

1.3.5 ซอฟต์แวร์ใช้ภาษาจาวา (Java) ที่สามารถรองรับการใช้งานบนเครือข่ายเพียร์ทูเพียร์ในการพัฒนา

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษาฐานข้อมูลแบบกระจาย

1.4.2 ศึกษาการค้นหาโดยใช้ตัวอย่างไฟล์วีดิโอ

1.4.3 ศึกษาโมเดลการปรับตัวอัตโนมัติ

1.4.4 ศึกษาเครือข่ายเพียร์ทูเพียร์

1.4.5 ฝึกเขียนโปรแกรมทำงานบนระบบเพียร์ทูเพียร์ โดยภาษาจาวา

1.4.6 รวบรวมปัญหาและความต้องการของซอฟต์แวร์

1.4.7 ออกแบบซอฟต์แวร์

1.4.8 พัฒนาซอฟต์แวร์

1.4.9 ทดสอบและปรับปรุงซอฟต์แวร์

1.4.10 จัดทำคู่มือโปรแกรม

1.4.11 สรุปและปิดโครงงาน

1.4.12 เขียนโครงงาน

ตารางที่ 1.1 ตารางแสดงระยะเวลาการดำเนินงาน

กิจกรรม	ปี 2551						ปี 2552	
	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.
1. ศึกษาฐานข้อมูลแบบกระจาย	↔							
2. ศึกษาการค้นหาโดยใช้ตัวอย่างไฟล์วีดิโอ	↔							
3. ศึกษาโมเดลการปรับตัวอัตโนมัติ		↔						
4. ศึกษาเครือข่ายเพียร์ทูเพียร์		↔						
5. ฝึกเขียนโปรแกรมทำงานบนระบบเพียร์ทูเพียร์ โดยภาษาจาวา			↔					
6. รวบรวมปัญหาและความต้องการของซอฟต์แวร์				↔				
7. ออกแบบซอฟต์แวร์				↔				
8. พัฒนาซอฟต์แวร์				←	—	—	→	
9. ทดสอบและปรับปรุงซอฟต์แวร์								↔
10. จัดทำคู่มือโปรแกรม								↔
11. สรุปและปิดโครงงาน								↔
12. เขียนโครงงาน	←	—	—	—	—	—	—	→

1.5 ผลที่คาดว่าจะได้รับ

1.5.1 แอพพลิเคชั่นที่ใช้ค้นหาไฟล์วีดิโอบนเครือข่ายเพียร์ทูเพียร์ ที่มีประสิทธิภาพ

1.5.2 แอพพลิเคชั่นสามารถใช้ตัวอย่างไฟล์วีดิโอในการค้นหาได้

1.5.3 แอพพลิเคชั่นมีโมเดลการปรับตัวแบบอัตโนมัติสำหรับการค้นหาได้

1.5.4 ได้ความรู้ความเข้าใจในเรื่องการเปรียบเทียบระหว่างไฟล์วีดิโอ

1.5.5 ได้ความรู้เรื่องระบบเพียร์ทูเพียร์

1.5.6 ได้ความรู้เรื่องการเก็บข้อมูลแบบกระจาย

1.5.7 ได้ความรู้เรื่องการสร้างโมเดลการปรับตัวแบบอัตโนมัติ โดยใช้หลักการปัญญาประดิษฐ์

1.6 งบประมาณที่ใช้

1.6.1 ค่าเอกสารโครงงาน	1,000	บาท
1.6.2 ค่าแผ่น CD-ROM และกล่องใส่ CD-ROM	500	บาท
1.6.3 อื่น ๆ	1,500	บาท
รวม	3,000	บาท

หมายเหตุ ขออนุมัติโดยถัวเฉลี่ยทุกรายการ

บทที่ 2
ทฤษฎีเบื้องต้น

2.1 การค้นหาแบบเปรียบเทียบข้อมูลวีดิโอกับวีดิโอ (Content-based Video Retrieval)

ในปัจจุบันเทคโนโลยีดิจิตอลสามารถสร้างไฟล์มัลติมิเดียโดยใช้ตัวเลขต่างๆ ได้ เช่น การนำไปสร้าง รูปภาพ เพลง หรือวีดิโอ ซึ่งการค้นหาและใช้งานจะเปรียบเทียบเฟรม (Frame) ต่างๆ กับตัวหนังสือ ดังนั้นระบบการค้นหา (Retrieval) จะใช้คำสำคัญ หรือกลุ่มคำสำคัญ เพื่อใช้ในระบบการค้นหา ตัวอย่างเช่น การค้นหารูปภาพบนเว็บ Google และ การค้นหาวีดิโอบนเว็บ YouTube แต่เนื่องจากการกำหนดคำสำคัญให้กับสิ่งที่ไม่ใช่วัตถุตัวอักษรจะต้องใช้วิธีการอื่น โดยทั่วไปการกำหนดคำสำคัญจะใช้บุคคลหนึ่งๆ เป็นคนกำหนดซึ่งอาจมีความผิดพลาดได้ เพื่อที่จะบรรเทาปัญหาดังกล่าวจึงมีการค้นคว้า การค้นหาแบบเปรียบเทียบข้อมูลชนิดเดียวกัน (Content-Based Information Retrieval) ขึ้นในปีค.ศ. 2006 [1]

การค้นหาแบบเปรียบเทียบข้อมูลชนิดเดียวกัน หรือ CBIR เป็นโมเดลระบบการค้นหาที่ใช้กับสิ่งที่ไม่ใช่วัตถุตัวอักษร ซึ่งเป็นพื้นฐานของระบบการทำงานแบบอัตโนมัติของวัตถุนี้ จะใช้งานแบบให้ผู้ใช้เป็นคนค้นหา ในการค้นหารูปภาพจะใช้มุมมองต่างๆ และจุดสี ส่วนการค้นหาเพลงจะใช้เวลาหรือคอร์ด ส่วนวีดิโอซึ่งประกอบไปด้วยเสียงและการเคลื่อนไหวของภาพ การค้นหาจะใช้พื้นฐานโดยทั่วไปของมัลติมิเดีย เช่น ระบบรูปภาพ ระบบเคลื่อนไหว และระบบเสียงพูด [1]

ซึ่งการนำมาใช้ในวีดิโอจะเรียกว่า การค้นหาแบบเปรียบเทียบข้อมูลวีดิโอกับวีดิโอ คือ การเปรียบเทียบไฟล์วีดิโอกับไฟล์วีดิโอ โดยใช้หลักการหลายด้าน เช่น ด้านความรู้เรื่องทั่วไปของคอมพิวเตอร์ ปัญญาประดิษฐ์ และระบบตัวอักษรอัตโนมัติ (automatic text processing) ซึ่งต้องใช้อัลกอลิทึมที่มีความเร็ว [1]

องค์ประกอบของระบบการค้นหาแบบเปรียบเทียบข้อมูลวีดิโอกับวีดิโอ แบ่งได้เป็น 3 ส่วน ได้แก่

1. การแบ่งส่วนของวีดิโอ (Video segmentation) เพื่อต้องการวีดิโอเป็นที่ส่วนเล็กๆ (clips) โดยแต่ละส่วนเล็กๆ ของวีดิโอจะต้องมีช็อต (shot) และส่วนที่ต่อไปยังส่วนอื่นๆ ของวีดิโอ โดยช็อต ที่ได้จะต้องมาจากการคัต (cuts) หรือเบรค (breaks) จากส่วนเล็กๆ ของวีดิโอนั้นๆ

2. การทำดัชนี (Indexing) เป็นระบบการใส่แทค (tag) ให้กับช็อตต่างๆ เพื่อที่จะสามารถนำไปใช้ค้นหาในขั้นตอนต่อไป โดยแทคจะได้มาจากการคำนวณของฟังก์ชันในโมเดล ซึ่งจะสามารถระบุส่วนที่สำคัญใน วีดิโอเป็นที่ส่วนเล็กๆ นั้นๆ ได้

3. ระบบการค้นหา (Retrieval and Browsing) เป็นการให้ผู้ใช้ (user) สามารถที่จะทำการค้นหาในฐานข้อมูลที่เก็บวีดิโอ และนำออกมาแสดงได้

2.1.1 เทคนิคที่ใช้ในการแบ่งส่วนของวีดิโอ (Video segmentation)

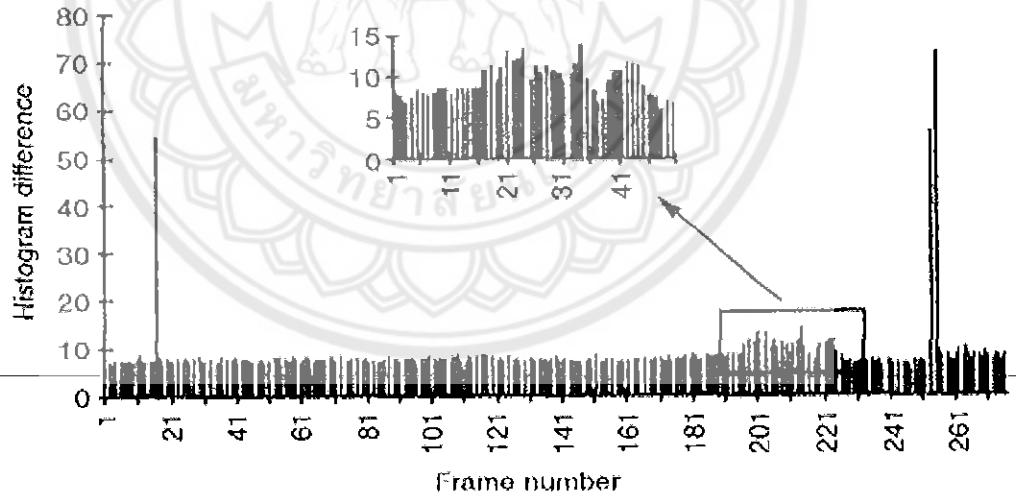
การที่จะได้วีดิโอที่เป็นส่วนเล็กๆ มานั้นต้องใช้การคัต ซึ่งการคัตจะเป็นการนำรูปออกมาจากวีดิโอแบบฉับพลัน แล้วนำไปปรับปรุงคุณภาพของรูปนั้นๆ ทีละน้อย โดยใช้การลบ (dissolve) การเลือน (fade) หรือการขจัด (wipe) เพื่อที่จะกำจัดสิ่งรบกวนออกซึ่งเป็นวิธีที่ยุ่งยาก จึงมีการค้นคว้าต่างๆ

อัลกอลิทึมที่นำไปใช้ในการคัตต่างกันทำให้ผลของการคัตออกมาแตกต่างกันไปเนื่องจากตัดคนละส่วนของไฟล์วีดิโอ (film) เช่น ไฟล์วีดิโอที่ไม่ถูกบีบอัด (uncompressed video) และไฟล์วีดิโอที่ถูกบีบอัด (compressed video) มีการต่างกันอย่างมากในด้านคุณภาพและการแบ่งส่วนของไฟล์วีดิโอ ทำให้มีการแบ่งงานวิจัยได้ 2 ส่วนตามการแบ่งส่วนของไฟล์วีดิโอ

2.1.1.1 การแบ่งส่วนของไฟล์วีดิโอที่ไม่ถูกบีบอัด (Uncompressed Video)

1) วิธีการแบ่งโดยใช้ฮีสโตแกรม (Histogram-based)

รูปภาพสามารถนำมาเทียบกับฟอร์มของฮีสโตแกรมได้ โดยนำทุกแถบสีมาเทียบกับสีเฉพาะหรือสีที่มีมากในภาพ แล้วดูว่าแต่ละแถบสีมีจำนวนของจุดสีจากในภาพอยู่เท่าไร โดยเทคนิควิธีการแบ่งโดยใช้ฮีสโตแกรมจะคัตเฉพาะที่ค่าความแตกต่างของภาพเหมือนกันและเมื่อค่าความแตกต่างมีค่ามากกว่าค่าที่กำหนด



รูปที่ 2.1 รูปแสดงกราฟฮีสโตแกรมระหว่างลำดับของเฟรมกับค่าความแตกต่างของฮีสโตแกรม

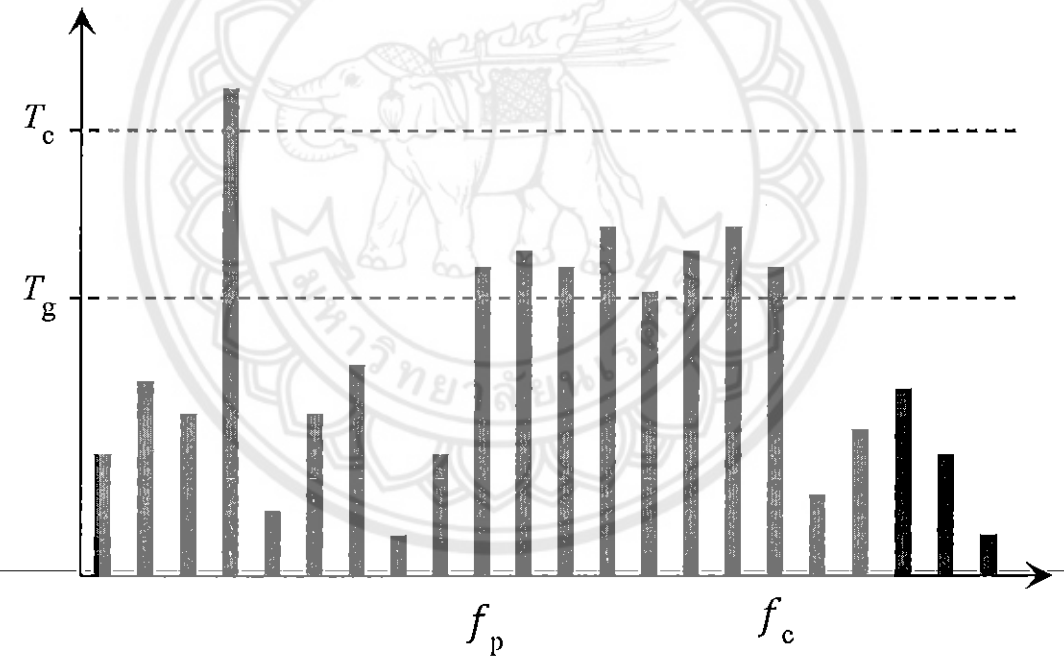
จากรูปที่ 2.1 จะเห็นว่ามีส่วนที่เปลี่ยนแปลงอย่างฉับพลัน เทคนิคฮีสโตแกรมจะมีผลเล็กน้อยต่อการทำงานของกล้องเมื่อเทียบกับเทคนิคการเทียบโครงสร้าง (template matching) และมีคุณภาพมากกว่าในขั้นตอนทดลอง แต่อย่างไรก็ตามเทคนิคฮีสโตแกรมไม่เหมาะสมกับการเทียบสิ่งที่เปลี่ยนแปลงเล็กน้อยของจุดสีที่มีมากในภาพ [2]

2) วิธีการแบ่งโดยใช้บล็อค (Block-based)

เทคนิคการแบ่งโดยใช้บล็อคจะแบ่งเฟรมให้เป็นหลายส่วน (Section) หรือบล็อค (Block) และเทียบความแตกต่างระหว่างส่วนข้างเคียง เทคนิคแบ่งโดยใช้บล็อคมีพื้นฐานเบื้องต้นเพื่อช่วยในการลดสัญญาณรบกวนและหลีกเลี่ยงแฟลตของกล้อง[2] อัลกอลิทึมของการแบ่งโดยใช้บล็อคเขียนโดย Idris และ Panchanathan ค่าความแตกต่างที่ได้จากทุกๆ บล็อคระหว่าง 2 เฟรม ถ้าไม่มีค่าความแตกต่างเลย ค่าความแตกต่างจะมีค่าเป็นศูนย์ และเมื่อมีการเปลี่ยนแปลงของค่าความแตกต่างจะทำการคัต

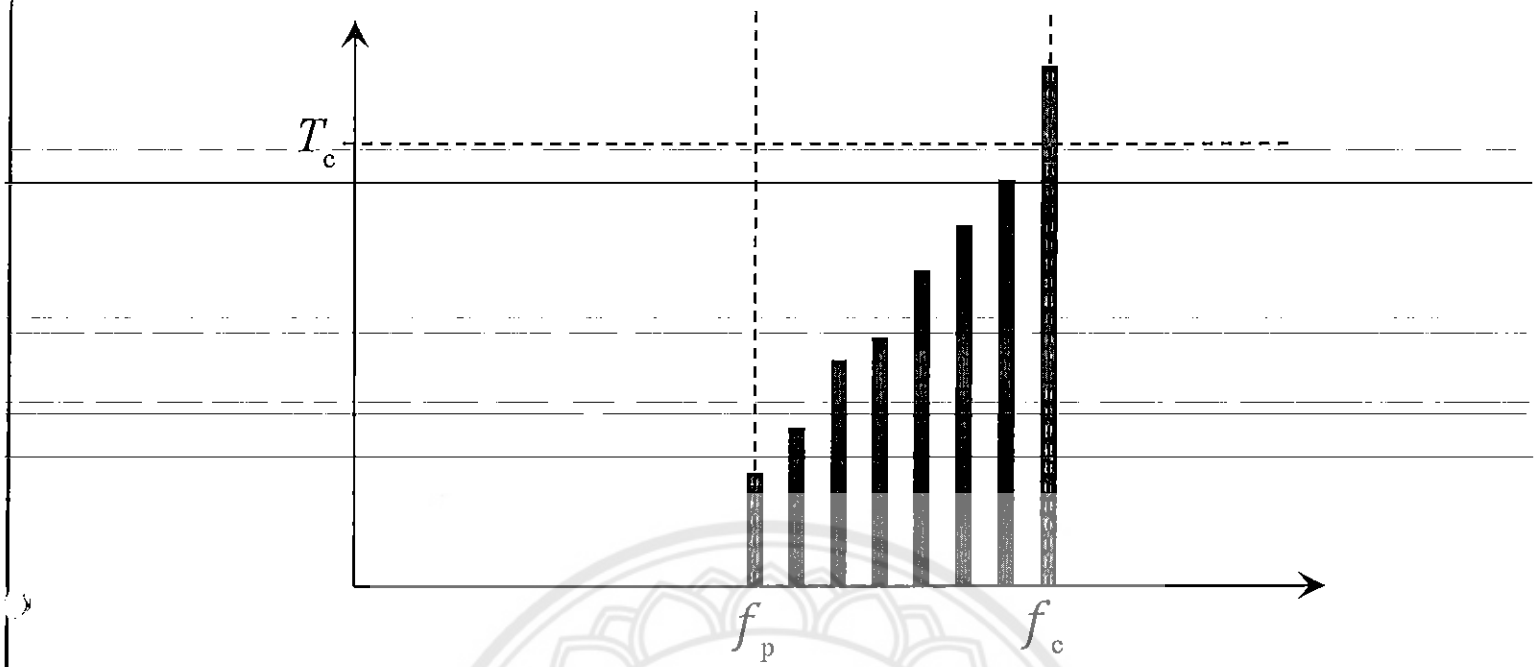
3) วิธีการแบ่งโดยใช้การเปรียบเทียบ 2 จุด (twin comparison)

เทคนิคการแบ่งโดยใช้การเปรียบเทียบ 2 จุดเป็นกลยุทธ์ที่ใช้ในการตรวจจับส่วนที่แตกต่างกัน ซึ่งไม่เหมือนกับสองอัลกอลิทึมที่ผ่านมา เทคนิคการแบ่งโดยใช้การเปรียบเทียบ 2 จุดจะใช้การเปรียบเทียบส่วนที่ต่างกันของจุดสีที่มีอยู่มาก โดยขั้นแรกต้องประมาณส่วนที่แตกต่างเพื่อที่จะนำไปประมวลผล (T_c)



รูปที่ 2.2 แสดงการประมาณค่า T_c เพื่อใช้ในการแบ่งโดยใช้การเปรียบเทียบ 2 จุด

ขั้นที่สองจะเป็นการเจาะจงเฉพาะส่วนที่จะนำมาหาจุดที่ไม่เหมือนกัน



รูปที่ 2.3 แสดงส่วนของเฟรมที่มีค่าเกิน T_c และจะทำการคัดตั้งแต่ f_p ถึง f_c

จากรูปที่ 2.3 เป็นการเจาะจงระหว่างจุดสองจุด โดยใช้อัลกอลิทึมค้นหาและนำมาคำนวณค่าความแตกต่างระหว่างเฟรมภายในสองจุด ถ้าค่าความเปลี่ยนแปลงมีค่าเท่ากับหรือมากกว่า T_c จะแสดงว่ามีการเปลี่ยนแปลงมากถึงจุดที่ได้ตั้งไว้และทำการคัด

2.1.1.2 การแบ่งส่วนของไฟล์วีดิโอที่ถูกบีบอัด (Compressed Video)

เนื่องจากไฟล์เป็นไฟล์ที่ถูกบีบอัดค่าต่างๆ จึงไม่เท่ากันกับไฟล์วีดิโอทั่วไป การเปรียบเทียบจึงต้องใช้ข้อตกลงกันระหว่างไฟล์วีดิโอที่ถูกบีบอัด (Compressed video) กับ (Uncompressed video) หรือบางครั้งจะไม่ได้นำส่วนของไฟล์วีดิโอที่ถูกบีบอัดมาทำการเปรียบเทียบ เช่น การนำหน้าปกของวีดิโอมาเป็นตัวเปรียบเทียบหรือการนำชื่อเรื่องของวีดิโอมาเปรียบเทียบ [3]

2.1.2 การทำดัชนี (Indexing)

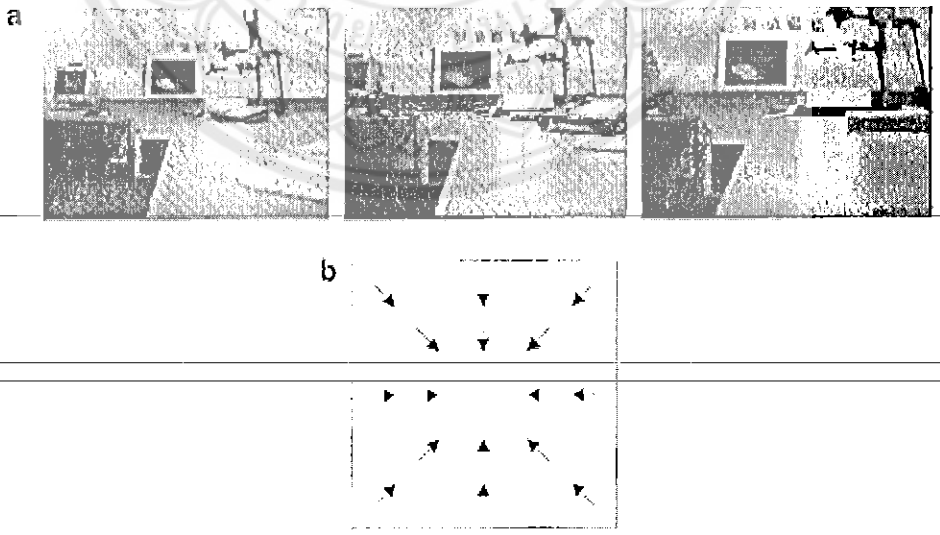
หลังจากที่ทำการแบ่งไฟล์วีดิโอให้เป็นช็อต แล้วขั้นต่อไปคือใส่ดัชนี (index) ให้กับช็อตแล้วนำไปใส่ในฐานข้อมูล การทำดัชนีมีอยู่สามชนิดใหญ่ๆ ซึ่งจะใช้งานแยกแต่ละประเภท ชนิดแรกเป็นการทำดัชนีโดยใช้ตัวแทน (spatial features) ซึ่งจะสนใจในเซตของเฟรมที่อ้างอิงและวีดิโอที่จะทำดัชนี เพื่อที่จะได้รูปภาพในเฟรมนั้นๆ มา ชนิดที่สองเป็นการคำนวณคุณภาพของวีดิโอแบบชั่วคราว ซึ่งจะเป็นการตรวจจับจากกล้องและการเคลื่อนที่ของวัตถุที่อยู่ในช็อต ชนิดที่สามเป็นจับการแปลงให้อยู่ในรูปของตัวอักษร (textual information) ซึ่งค่าที่ได้มาจะอยู่ในบทสนทนาหรือเป็นตัวอักษรที่อยู่บนหน้าจอ ซึ่งบรรยายต่างๆ เช่น รายการสาระแนโชว์ 07show หรือ CNN เป็นต้น และชนิดที่สามไม่ต้องใช้ผู้กระทำใดๆ แต่ใช้การเชื่อมของ 2 ส่วนย่อย

2.1.2.1 การดัชนีโดยใช้ตัวแทน (spatial features)

การทำดัชนีโดยใช้ตัวแทน (spatial features) ในจากไฟล์วีดิโอ จะใช้การดึงภาพจากเทคนิควิเคราะห์ภาพในเฟรมอ้างอิง ซึ่งเฟรมอ้างอิงจะเป็นเฟรมเดียวที่เป็นตัวแทนของแต่ละช็อตภายในวีดิโอ การวิเคราะห์ภาพของเฟรมจะใช้ได้ทั้งการอ้างอิงระดับต่ำ คือเปรียบเทียบจุดสีหรือจุดที่มีความเข้มสีมาก (เหมือนกับการทำ histograms) ไปจนถึงการวิเคราะห์ระดับสูง คือการเปรียบเทียบใบหน้าและวัตถุที่สนใจ ขั้นตอนของการวิเคราะห์จะใช้เซตของคำสำคัญ หรือแทคต่างๆ ที่สามารถสื่อความหมายของช็อตนั้นได้ เช่น การวิเคราะห์จุดต่างๆ เพื่อให้ได้สีเด่นของ เฟรมและการกระจาย ซึ่งซอฟต์แวร์ที่เปรียบเทียบวัตถุจะสามารถบ่งบอกได้ว่ามีหรือไม่มีวัตถุหรือคนได้

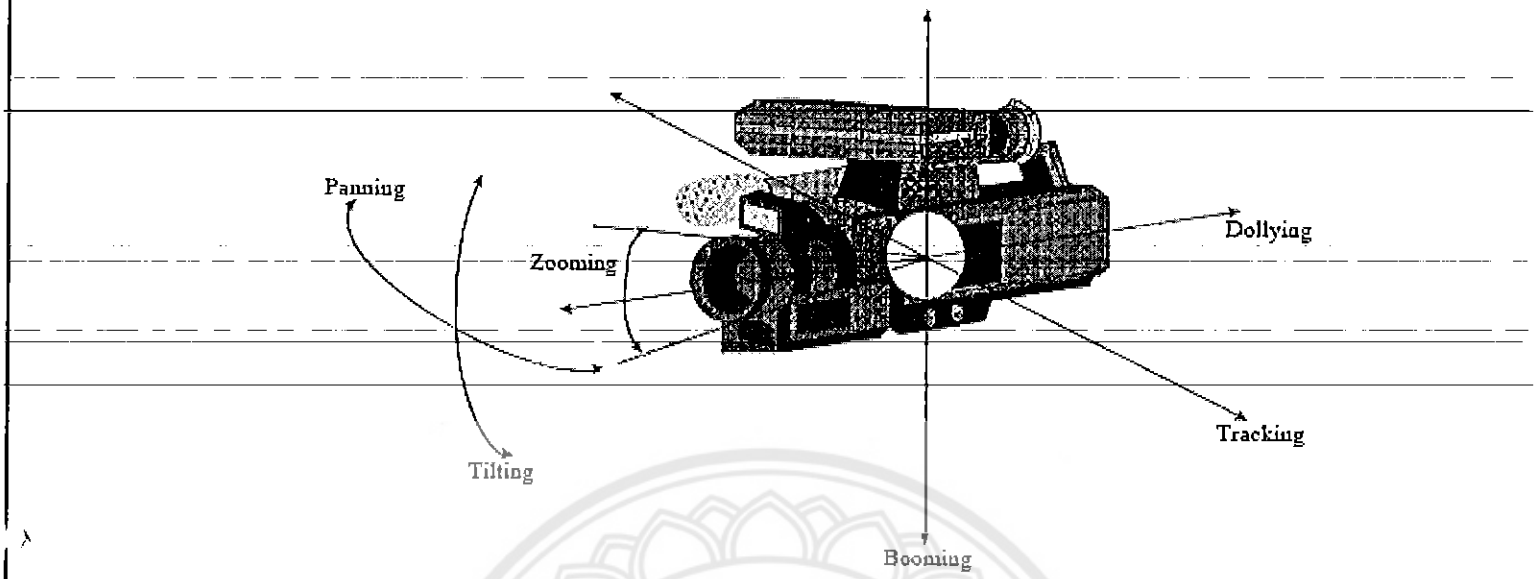
2.1.2.2 การดัชนีโดยใช้การคำนวณคุณภาพของวีดิโอแบบชั่วคราว (temporal features)

สิ่งสำคัญสองประการของการคำนวณคุณภาพของวีดิโอแบบชั่วคราว (temporal features) ที่จะสามารถระบุไฟล์วีดิโอได้ ได้แก่ ประการแรกการเคลื่อนไหวและประการที่สองการเคลื่อนไหวของกล้อง วัตถุที่เคลื่อนไหวได้จะง่ายที่อ้างอิงการเคลื่อนไหววัตถุภายในไฟล์วีดิโอได้เป็นอย่างดี การเคลื่อนไหวของกล้องเพื่อที่จะได้พื้นที่ใหม่ เช่น การซูม การส่ายหน้ากล้อง เป็นต้น หลายอัลกอลิทึมสามารถนำมาใช้คำนวณในรูปแบบต่างๆ นี้ได้ แต่โดยส่วนมากจะใช้เวกเตอร์การเคลื่อนไหว (motion vector) ที่มายังจุดศูนย์กลาง เวกเตอร์การเคลื่อนไหวจะถูกสร้างโดยการแทคกิ้ง (tracking) เซตของจุดที่คล้ายกัน ซึ่งจะเปลี่ยนสถานที่จากเฟรมไปอีกเฟรมหนึ่ง เช่น มีกล่องสี่เหลี่ยมสีดำอยู่ที่ตรงกลางของเฟรมแรก เมื่อกล่องเลื่อนขึ้น 1 จุดในเฟรมต่อๆ มา จะทำให้เวกเตอร์การเคลื่อนไหวมีการชี้ขึ้น และเมื่อมีหลายเฟรมจะทำให้มีเวกเตอร์การเคลื่อนไหวหลายทิศทางได้



รูปที่ 2.4 แสดงภาพตัวอย่าง a แสดงการซูม และ b แสดงทิศทางของเวกเตอร์การเคลื่อนไหว

ภาพแสดงการกระทำของกล้องที่จะมีผลต่อภาพ



รูปที่ 2.5 แสดงภาพการเคลื่อนไหวของกล้องที่จะมีผลต่อภาพ

2.1.2.3 การดัชนีด้วยการตรวจจับตัวอักษร (Derived Text)

การตรวจจับภาพ (Caption Detection)

การตรวจจับภาพจะใช้แสดงข้อมูลในวีดิโอประเภทข่าว สารคดี และกีฬา การตรวจจับภาพสามารถทำให้ทราบถึงส่วนสำคัญๆ ในวีดิโอและสามารถนำมาใช้ในการทำดัชนีได้ โดยสามารถวิเคราะห์คำในแต่ละเฟรมและอ่านคำโดยใช้ optical character recognition (OCR) ซึ่งเป็นซอฟต์แวร์ที่สามารถเก็บคำอธิบายภาพที่มีลักษณะเป็นตัวอักษรมาเก็บไว้ในรูปแบบของไฟล์ตัวอักษร (คิดค้นโดย Hauptmann et al.) ซึ่งสามารถนำไฟล์ตัวอักษรไปใช้ในการค้นหาโดยการเปรียบเทียบกับแทคที่ได้จากการทำดัชนี

มีหลายอัลกอลิทึมที่สามารถช่วยในการตรวจจับภาพที่ Brunelli et al. ได้ทำการศึกษาและสรุปว่ามีหลายอัลกอลิทึมที่มีคุณสมบัติในการ caption เพื่อที่จะบ่งบอกลักษณะของวีดิโอ เช่น ข้อความหรือหัวข้อส่วนมากจะอยู่ตรงกลางหรือส่วนล่างของภาพ อัลกอลิทึมสามารถอ้างอิงไปยัง frame นั้นได้และนำภาพเหล่านั้นมาหาตัวอักษรที่ปรากฏโดยใช้โปรแกรม OCR และยังมีอีกหลายอัลกอลิทึมที่สามารถระบุการเคลื่อนไหวของตัวอักษรและตัวอักษรที่อยู่บนพื้นหลังได้

การจับเสียง (Speech Recognition)

โปรแกรม speech recognition เป็นโปรแกรมที่สามารถแปลงเสียงของวีดิโอให้เป็นไฟล์ตัวอักษรได้ Hauptmann et al. ผู้คิดค้นระบบการค้นหาวีดิโอที่มีการตรวจจับภาพ (caption detection) และสัญญาณเสียง (audio detection) มีการปรับปรุงคุณภาพของระบบ โดยพัฒนาการทำดัชนีให้มีความแม่นยำขึ้น เพื่อบ่งบอกรายละเอียดได้มากขึ้น และสามารถนำไปใช้ร่วมกับส่วนอื่นๆ

มีการทดลองใช้การตรวจจับภาพและบันทึกของเสียงที่มีคุณภาพต่ำ แต่เมื่อนำทั้งสองส่วนมาทำการวิเคราะห์กลับได้ผลเป็นอย่างดี อย่างไรก็ตามเมื่อนำบันทึกของเสียงไปใช้ร่วมกับสิ่งอื่นเพื่อวิเคราะห์รูปภาพจะไม่สามารถปรับปรุงคุณภาพการตรวจจับได้ แต่ถ้านำการตรวจจับภาพมาใช้ร่วมกับสิ่งอื่นจะได้ผลลัพธ์ที่ดีกว่าในการวิเคราะห์ จากที่ได้กล่าวมาทำให้ทราบว่าการจับเสียงไม่สามารถที่จะนำไปใช้ในการทำดัชนีโดยใช้การจับเสียงเพียงอย่างเดียวได้ แต่ถ้านำการตรวจจับภาพมาช่วยจะได้ผลลัพธ์ที่ดีได้

2.1.3 การเปรียบเทียบและค้นหา (Retrieval and Browsing)

หลังจากที่ได้แบ่งส่วนวีดิโอ ทำดัชนี และเก็บเข้าฐานข้อมูลแล้ว ขั้นสุดท้ายจะเป็นระบบการค้นหาวีดิโอซึ่งจะเป็นหน้าที่ใช้ติดต่อกับผู้ใช้เพื่อที่จะค้นหาและทำการเปรียบเทียบกับดัชนีของข้อมูลวีดิโอ โดยทั่วไปรูปภาพและตัวอักษรจะเป็นตัวตัดสินว่าข้อมูลเหล่านั้นเหมือนกันหรือไม่ ซึ่งไม่ได้เหมือนกับการที่ใช้คนดูวีดิโอเพื่อแยกแยะไฟล์วีดิโอนั้นๆ ซึ่งจะใช้เวลานานเป็นชั่วโมง อย่างไรก็ตามสิ่งสำคัญคือสร้างการติดต่อกับผู้ใช้ เพื่อให้ทำการเปรียบเทียบวีดิโอที่ถูกแบ่งเป็นส่วนย่อยๆ ไว้แล้ว โดยจะใช้การค้นหาและการแสดงผลของการค้นหาที่เกิดจากการเทียบส่วนย่อยของวีดิโอกับไฟล์วีดิโอ

ในปีค.ศ.1994 Smoliar และ Zhang ได้เสนอระบบ multiple user interfaces ซึ่งใช้ในการค้นหาไฟล์วีดิโอ โดยขั้นตอนแรกเรียกว่า Clipmap เป็นการแสดงช็อตหรือส่วนย่อยของวีดิโอทั้งหมด เพื่อที่จะนำไปทำการค้นหา ซึ่งส่วนย่อยของวีดิโอเหล่านั้นจะถูกแสดงและสามารถคลิกเพื่อเข้าไปดูได้ พร้อมทั้งแสดงส่วนที่อยู่รอบของส่วนย่อยของวีดิโอ ซึ่งมีเนื้อหาติดกันเพื่อให้ผู้ใช้สามารถเลือกส่วนต่างๆ มาแสดงได้ Smoliar และ Zhang ได้แสดง iHierarchical Video Magnifieri ซึ่งเป็นการแบ่งไฟล์วีดิโอออกเป็น 5 ส่วนที่เท่ากันและแสดงเป็นแถวแนวตั้งที่มีปุ่มให้เลือก 5 ปุ่ม อยู่ส่วนบนของหน้าจอ เมื่อมีปุ่มถูกเลือกไฟล์วีดิโอที่อยู่ภายในจะถูกแบ่งออกเป็นไฟล์วีดิโอย่อยอีก 5 ส่วนที่เท่ากันและจะมีอีก 5 ปุ่มใหม่อยู่ด้านล่างของ 5 ปุ่มที่กดในครั้งแรก และเมื่อกดครั้งที่ต่อๆ ไปจะมีแถวต่อลงไปเรื่อยๆ ซึ่งแสดงว่าให้เห็นความสามารถการค้นหาแบบลึกลงไปในแต่ละส่วนของวีดิโอ

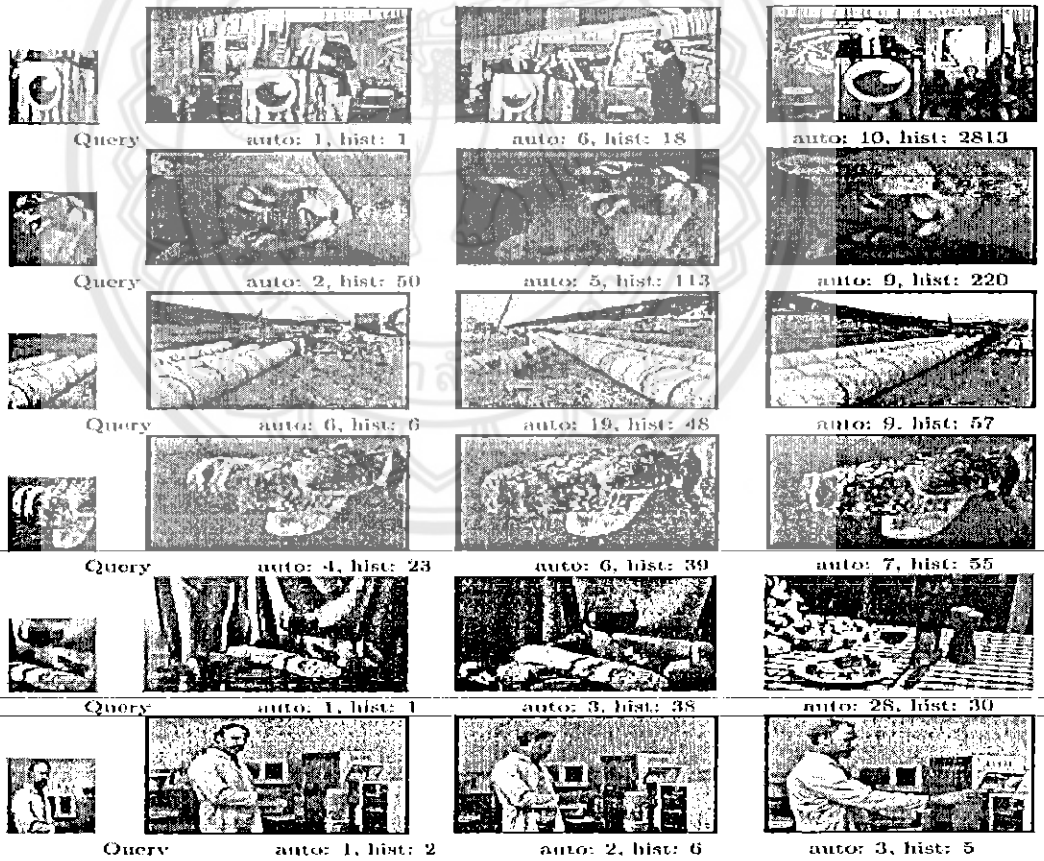
ในปีค.ศ.1997 Chang et al. ได้นำระบบ CBVR ไปใช้แบบออนไลน์ (online) ซึ่งเรียกว่า VideoQ และได้จัดตั้งทีมเพื่อที่จะพัฒนาส่วนการติดต่อกับผู้ใช้ VideoQ เป็นระบบที่ลึกซึ้งเพราะผู้ใช้สามารถสร้างการค้นหาโดยใช้การวาดวัตถุและเส้นทางการเคลื่อนที่บนหน้าจอ (movement vectors onscreen) การวาดต่างๆ สามารถสร้างและสั่งการโดยผู้ใช้ซึ่งบ่งบอกถึงลำดับเหตุการณ์เพื่อที่จะนำไปค้นหา ระบบฐานข้อมูลจะเป็นตัวค้นหารูปภาพที่สอดคล้องกับภาพผู้ใช้วาด และแสดงผลออกทางด้านขวาของจอภาพ ผู้ใช้สามารถคลิกไปที่ปุ่มเพื่อที่จะเล่น clip ได้ (ซึ่งการแสดงของปุ่มเหมือนกับของ Smoliar & Zhang) โดยผลของการเปรียบเทียบกับขึ้นอยู่กับรูปภาพและการแสดงต่างๆ (สี เวลา การเคลื่อนที่ และอื่นๆ)

ตัวอย่างสมการเมื่อนำภาพที่ได้จากการช็อตของวีดิโอที่ต้องการค้นหานำไปเทียบกับช็อตของวีดิโอที่อยู่ในฐานข้อมูลจะได้ค่าสมการการเปรียบเทียบโดยใช้วิธีการเปรียบเทียบฮีสโตแกรม (Histograms) 2 ฮีสโตแกรม (h_1 และ h_2) ดังนี้

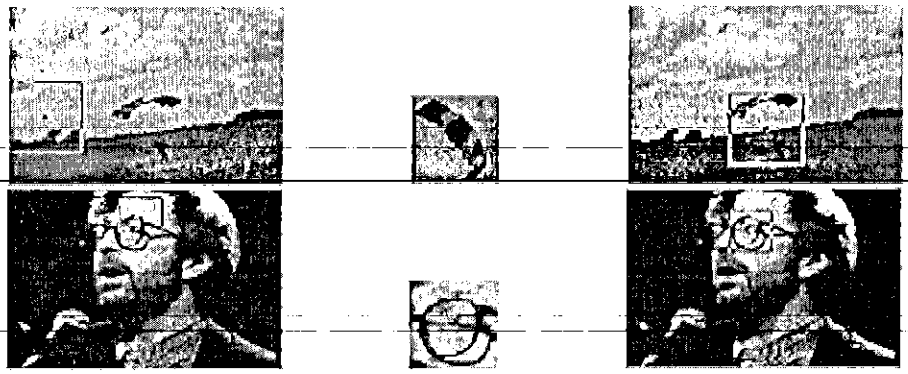
$$D(h_1, h_2) = -\theta \sum_{i=1}^{n_b} \min\{\mu(h_1[i]), \mu(h_2[i])\} \quad (2.1)$$
$$+\alpha \sum_{i=1}^{n_b} \min\{\mu(h_1[i]) - \mu(h_2[i], 0)\}$$
$$+\beta \sum_{i=1}^{n_b} \min\{\mu(h_1[i]) - \mu(h_2[i], 0)\}$$

เมื่อ n_b เป็นตัวเลขแสดงจำนวนที่ต้องนำช็อตไปเปรียบเทียบ α และ β เป็นค่าคงที่ที่บ่งบอกว่าต้องเทียบสีจำนวนเท่าไรในการเปรียบเทียบฮีสโตแกรม [4]

ตัวอย่างการเปรียบเทียบและผลลัพธ์ที่ได้โดยใช้สีที่มีมากในภาพ [5]



รูปที่ 2.6 แสดงการนำภาพไปทำการเปรียบเทียบภาพที่ได้จากการทำดัชนีตัวอย่างการเปรียบเทียบภาพที่ได้จากการคัดและผลลัพธ์โดยใช้การเทียบคู่เหมือน (Correlogram) [5]



รูปที่ 2.7 แสดงการเปรียบเทียบภาพโดยใช้การเทียบคู่เหมือน (Correlogram)

2.2 การทำดัชนีวีดิโอแบบ AVI (Adaptive Video Indexing) [6]

เป็นการทำดัชนีให้กับวีดิโออีกรูปแบบหนึ่งซึ่งการทำดัชนีแบบ AVI นี้เองเป็นการทำดัชนีโดยอาศัยรายละเอียดสำคัญ (content) ของวีดิโอเป็นตัวกำหนดดัชนีที่จะมีหลายๆระดับด้วยกันแยกเป็น การทำดัชนีโดยอาศัยภาพวีดิโอตอนหนึ่งในเรื่อง กลุ่มของภาพวีดิโอตอนนั้นๆในเรื่อง หรือภาพวีดิโอของทั้งเรื่องเลยก็ได้ โดยจะเห็นได้ว่าการทำดัชนีแบบนี้จะเป็นการใช้ภาพของวีดิโอเป็นตัวทำดัชนีซึ่งแตกต่างการทำดัชนีในแบบทั่วไปที่ทำดัชนีตามคำสำคัญ

การทำดัชนีแบบ AVI ไม่เหมือนกับการทำดัชนีโดยทั่วไปที่สามารถค้นหาได้เฉพาะวีดิโอบางส่วนของเรื่องเท่านั้น ซึ่งนั้นก็เนื่องมาจากข้อจำกัดของการค้นหาที่ใช้คำสำคัญในการทำดัชนีนั่นเอง แต่เมื่อใช้การทำดัชนีแบบ AVI นั้นจะสามารถค้นหากลุ่มของวีดิโอ หรือภาพวีดิโอทั้งหมดของเรื่องที่ทำการค้นหาได้ เนื่องจากมีการอาศัยรายละเอียดสำคัญของวีดิโอเป็นตัวกำหนดการทำดัชนี และยังสามารถใช้ส่วนหนึ่งของวีดิโอเป็นตัวกำหนดการค้นหาได้

การทำดัชนีแบบ AVI นั้นยังสามารถปรับปรุงตัวดัชนีที่ใช้ค้นหาได้เองแบบอัตโนมัติ โดยสามารถปรับปรุงตัวเองได้เมื่อมีการค้นหาไปยังเพียร์ต่อๆ ไปในทุกๆ ครั้งที่มีการค้นหา เพื่อเพิ่มความแม่นยำในการค้นหาวีดิโอให้มีความถูกต้องมากยิ่งขึ้น ซึ่งมีขั้นขั้นตอนการทำดัชนีแบบ AVI ดังต่อไปนี้

2.2.1 การสร้างต้นแบบ (Template Generation)

ขั้นตอนนี้เป็นขั้นตอนแรกในการทำดัชนีแบบ AVI คือจะทำการจัดกลุ่มเฟรมของวีดิโอที่อยู่ภายในฐานข้อมูลโดยนำแต่ละเฟรมที่มีความเหมือนหรือคล้ายกันนำมาจัดไว้เป็นกลุ่มๆ โดยขั้นตอนการทำ Template Generation นี้ก็จะแบ่งย่อยเป็นอีกสองขั้นตอนคือ การนำเฟรมวีดิโอมาจัดกลุ่มและการหาตัวแทนของกลุ่ม (Visual Templates) เพื่อใช้ในการเปรียบเทียบกันระหว่างวีดิโอ

โดยในขั้นตอนแรกคือการจัดกลุ่มเฟรมของวีดิโอที่เหมือนกัน จะอาศัยการคำนวณ Color Histogram จาก CHSD (Color-Histogram-Based Shot Detection) อัลกอริทึม โดยอัลกอริทึมนี้จะทำ

การวิเคราะห์และคำนวณค่า Color Histogram ของวีดิโอแต่ละเฟรมออกมาให้ เพื่อนำมาใช้แยกความแตกต่างหรือความเหมือนกันของแต่ละเฟรมวีดิโอในฐานข้อมูลโดยหลักการก็คือพิจารณาสถิติความถี่ของ Color Histogram โดยที่แต่เฟรมที่มีค่าสถิติความถี่ของตัวเอง (เช่น ระบบ RGB สีแดง 80 % สีน้ำเงิน 15% สีเขียว 5%) และเมื่อนำค่าสถิติไปเปรียบเทียบกับวีดิโออีกเฟรม ถ้าหากมีค่าสถิติความถี่ที่ใกล้เคียงของเฟรมนั้นก็จะจัดไว้อยู่ในกลุ่มเดียวกันโดยจะเรียกกลุ่มนี้ว่า Training Set ที่มีค่า Histogram คือ H_t และ H_v คือค่า Color Histogram ของเฟรมวีดิโอทั้งหมดภายในฐานข้อมูลโดยให้ $\overrightarrow{x} = [x_1 \ldots x_{48}] \in R^{48}$ เป็นเวกเตอร์ของ Color Histogram ตาม CHSD อัลกอริทึม ดังนี้

$$H_v = [\vec{x}_1, \ldots, \vec{x}_v, \ldots, \vec{x}_V]^T = [x_{vi}], \; v = 1, \ldots, V, \; i = 1, \ldots, 48 \qquad (2.2)$$

โดยที่ V คือ จำนวนของวีดิโอทั้งหมด จากนั้นนำมาทำการจัดกลุ่มจะได้ว่า

$$H_t \subset H_v$$

$$H_t = [x_{ji}], \; j = 1, \ldots, J, \; J < V, \; i = 1, \ldots, 48 \qquad (2.3)$$

จากนั้นจะมาถึงขั้นตอนต่อไปคือการหาเฟรมวีดิโอเฟรมหนึ่งที่จะนำมาเป็นตัวแทน (Visual Templates) เพื่อใช้ในการเปรียบเทียบกันระหว่างวีดิโอซึ่งจะเหมือนกันกับ Color Histogram ที่ใช้ค่าความถี่ของ RGB เป็นตัวพิจารณาว่าเฟรมนั้นมีความใกล้เคียงกันหรือไม่ จากที่กล่าวมาข้างต้นแต่ใน Histogram นี้ก็จะมีตัวแทนกลุ่มใช้เป็นตัวพิจารณาว่าแต่ละเฟรมมีความใกล้เคียงกันมากน้อยเพียงใด โดยจะทำการหาค่า Mean และ Standard Deviation จากเวกเตอร์ของ Color Histogram ของวีดิโอแต่ละกลุ่ม (H_t) ซึ่งก็คือ $\vec{x}_i = [x_{1i}, \ldots, x_{ji}, \ldots, x_{Ji}]^T$ มีขั้นตอนดังต่อไปนี้

หาค่า Mean

$$\bar{x}_{ji} = \frac{1}{J} \sum_{j=1}^{J} x_{ji} \qquad (2.4)$$

หาค่า Standard Deviation

$$S_{ji} = \left(\frac{1}{J-1} \sum_{j=1}^{J} (x_{ji} - \bar{x}_{ji})^2 \right)^{\frac{1}{2}} \quad (2.5)$$

เมื่อได้ค่าเหล่านี้แล้วขั้นตอนต่อไปก็คือการนำไปหาเวกเตอร์ที่เป็นตัวแทนของวีดิโอกลุ่ม (H_r) คือ $\tilde{x}_{ji}$ ซึ่งมี Histogram คือ $\tilde{H}_t$

$$f(\vec{x}_i) = \left\{ \tilde{x}_{ji} \,\middle|\, j = 1, \ldots, J \right\} \quad (2.6)$$

$$\tilde{x}_{ji} = \frac{(x_{ji} - \bar{x}_{ji})}{s_{ji}} \quad (2.7)$$

จากนั้นจะทำการนำค่าที่ได้นี้เป็นตัวพิจารณาความใกล้เคียงกันของวีดิโอ โดยการเปรียบเทียบจะใช้ Histogram เพื่อบ่งบอกว่าวีดิโอแต่ละเฟรมนั้นมีค่าใกล้เคียงตัวแทนกลุ่มไหนมากที่สุด โดยจะกล่าวในขั้นตอนต่อไปของการทำดัชนีแบบ AVI

2.2.2 การกำหนดรูปแบบตามต้นแบบโดยอาศัยความถี่ (Template-Frequency Modeling)

ขั้นตอนนี้จะเป็นขั้นตอนของการทำ Histogram ให้วีดิโอในแต่ละไฟล์โดยนำเฟรมของวีดิโอนั้นเทียบกับวีดิโอตัวแทนของแต่ละกลุ่มที่ได้มาในขั้นตอนของการทำ Template Generation หลักการคือเมื่อทำการเปรียบเทียบแล้วมีค่าใกล้เคียงกับตัวแทนของกลุ่มไหนเยอะมากที่สุดก็เก็บค่าสถิติความถี่ (Template-Frequency Factor) ของต้นแบบที่เหมือนนั้นไว้ (เช่น มี Visual Template 4 กลุ่มคือ M1, M2, M3 และ M4 มี Video1.mpg 10 เฟรม F1… F10 และ F1 (M1, M2) หมายความว่าเฟรม F1 นั้นใกล้เคียง M1 กับ M2 มากที่สุด ซึ่งจะมีค่าดัชนีนี้อยู่ในทุกๆ เฟรมของวีดิโอ) โดยจะนำค่าเหล่านี้ไปเปรียบเทียบกับเฟรมของวีดิโออื่น ถ้าหากมีเฟรมที่มีค่าที่ใกล้เคียงกันก็จะพบวีดิโอที่เหมือนกัน

กำหนดให้แต่ละ Template มีเวกเตอร์ $\vec{g}_r$, r $\in$ $[1, R]$ โดยจะอยู่ในขอบเขตของ Voronoi Space R $\subset \mathcal{R}^{48}$ (Voronoi Space เป็นขอบเขตที่เป็น Optimized-Model ของเวกเตอร์ $\vec{g}_r$, r = 1, …, R) จากนั้นจะกำหนดให้ลำดับเฟรมของวีดิโอที่จะนำมาทำดัชนี (Interval) คือ I โดยทำการเปรียบเทียบวีดิโอทั้งหมดนั่นคือภายในขอบเขตของ $\Re_r$, r = 1, …, R เพื่อให้ได้เวกเตอร์

น้ำหนักเป็น $\vec{v} = [w_1, \ldots, w_r, \ldots, w_R]$ ซึ่งแต่ละค่าของ w_r ก็คือค่าสถิติความถี่ของวีดิโอแต่ละเฟรมนั่นเอง

กำหนดให้ $\vec{x}_m \in R^{48}$ แทนเวกเตอร์ที่แตกออกมาจากทั้งหมดวีดิโอ m เฟรมที่อยู่ในเซท $D_{I_j} = \{(\vec{x}_1, f_1), \ldots, (\vec{x}_m, f_m), \ldots, (\vec{x}_M, f_M)\}$ ซึ่งจะมีตามวีดิโอที่นำมาทำดัชนี (Interval) I_j และให้เซทของวีดิโอที่เป็นตัวแทนหรือต้นแบบมีค่า $C = \{\vec{g}_r, | \ r = 1, 2, \ldots, R\}$ ที่มีขอบเขตตั้งแต่ $R^{48} \rightarrow \mathcal{C}$ ภายใน Voronoi Space หลังจากนั้นนำเฟรมวีดิโอเหล่านี้มาทำการวิเคราะห์ว่ามีความใกล้เคียงกับวีดิโอตัวแทนอันไหนบ้าง อธิบายเป็นสมการได้ดังนี้

$$\vec{x}_m \Rightarrow \langle \phi_v(C, \vec{x}_m), \Re^{\eta}_{r^*} \rangle \Rightarrow \rho^{(\vec{x}_m)}$$

$$\phi_v(C, \vec{x}_m) = \arg\min_r (\|\vec{x}_m - \vec{g}_r\|)$$

$$\Re^{\eta}_{r^*} = \bigcup^{\eta}_{i=1} \vec{g}_i \qquad (2.8)$$

โดยที่ $\Re^{\eta}_{r^*}$ คือขอบเขตที่ประกอบด้วยตั้งแต่ η Voronoi Cell ไปจนถึง $\vec{g}_{r^*}$ ซึ่ง r^* แทนค่าดัชนีที่มีความใกล้เคียงมากที่สุดกับวีดิโอต้นแบบหรือตัวแทน (เช่น M1, M2, M3, M4)

โดยทั่วไปแล้ววีดิโอเฟรมหนึ่งๆ นั้นอาจจะใกล้เคียงกับวีดิโอต้นแบบได้มากกว่าหนึ่งต้นแบบด้วยกัน (เช่น เฟรม F1 นั้นอาจจะมีค่า r^* ได้มากกว่าหนึ่งค่า คืออาจจะใกล้เคียงกับวีดิโอต้นแบบ M1, M2 และ M3 ก็ได้) ดังนั้นเพื่อให้ได้ความแม่นยำและความถูกต้องมากที่สุดในการค้นหาวีดิโอจึงได้มีการทำให้สามารถหาค่าความใกล้เคียงกับวีดิโอต้นแบบได้มากกว่าหนึ่งขึ้นไป (Multiple-Label Indexing) ตามสมการดังต่อไปนี้

$$\iota^{\vec{x}_m}_{r^*}, \iota^{\vec{x}_m}_{r^*,1}, \ldots, \iota^{\vec{x}_m}_{r^*,(\eta-1)} \qquad (2.9)$$

เมื่อได้ค่าดัชนีของแต่ละเฟรมของวีดิโอมาแล้วหลังจากนั้นก็ทำการวิเคราะห์ว่าแต่ละเฟรมของวีดิโอนั้นๆ มีสถิติความถี่ที่ใกล้เคียงกับต้นแบบไหนเยอะมากที่สุด ($\rho^{(\vec{x}_m)}$, $m = 1, \ldots, M$) โดยจะต้องทำทุกเฟรมของวีดิโอนั้นๆ แล้วนำค่านี้มาเก็บไว้เพื่อนำไปเปรียบเทียบกับวีดิโออื่นๆ ที่ต้องนำมาทำการหาค่านี้เหมือนกันจากนั้นนำมาพิจารณาดูว่า วีดิโอทั้งสองไฟล์ที่นำมาเปรียบเทียบนั้นมีความเหมือนหรือใกล้เคียงกันหรือไม่ โดยพิจารณาจาก Histogram (เช่น video1.mpg ใกล้เคียง M1

ทั้งหมด 4 เฟรม M2 1 เฟรม M3 2 เฟรม และ video2.mpg ใกล้เคียง M1 5 เฟรม M2 2 เฟรม สรุปได้ว่าวีดิโอทั้งสองเป็นวีดิโอที่กล่าวถึงเรื่องเดียวกันหรือมีความใกล้เคียงกันนั่นเอง) โดยสามารถหาค่าสถิติเหล่านี้ได้ตามสมการ

$$\vec{v}_j = (w_{j1}, \ldots, w_{jr}, \ldots, w_{jR}) \quad (2.10)$$

$$w_{jr} = \frac{freq_{jr}}{max_r} \times \log \frac{N}{n_r} \quad (2.11)$$

จะได้ $\vec{v}_j$ ที่ประกอบด้วยค่าสถิติของแต่ละเฟรมของวีดิโอนั้นๆ ออกมาโดยที่ $freq_{jr}$ นั้นคือค่าความถี่ที่ทำการเปรียบเทียบกับวีดิโอต้นแบบ $\vec{g}_r$ ของการทำดัชนี

2.2.3 วิธีการเปรียบเทียบตามหลักการของ AVI

จากขั้นตอนการทำงานของ AVI ดังกล่าวแล้ว สุดท้ายแล้วจะทำให้ได้ค่าเวกเตอร์มาในแต่ละวีดิโอไฟล์ซึ่งแสดงได้ดังต่อไปนี้ $\vec{v}_n = [t_1\ t_2\ t_3\ t_4 \ldots t_n]$ โดยที่ t คือตัวต้นแบบทั้งหมดที่อยู่ภายในฐานข้อมูล และ n คือจำนวนของต้นแบบทั้งหมดภายในฐานข้อมูล ซึ่งค่าภายในเวกเตอร์นี้จะเรียกว่าพารามิเตอร์น้ำหนักซึ่งเป็นค่าเฉพาะของไฟล์วีดิโอนั้นๆ เท่านั้น โดยแต่ละไฟล์ก็จะแตกต่างกันออกไป โดยค่าพารามิเตอร์นี้จะได้มาจากขั้นตอนการกำหนดรูปตามต้นแบบ โดยอาศัยความถี่นั่นเอง จากนั้นเมื่อได้ค่าเหล่านี้มาแล้วก็จะนำค่าพารามิเตอร์น้ำหนักของแต่ละไฟล์เหล่านี้มาเปรียบเทียบกันโดยใช้หลักการของ Similarity Measure Cosine ซึ่งมีวิธีทำดังต่อไปนี้ กำหนดให้ค่าเวกเตอร์ต่างๆ มีค่าดังต่อไปนี้

$\vec{v}_n = [t_1\ t_2\ t_3\ t_4\ t_5]$
$\vec{v}_1 = [1\ 0.2\ 0.2\ 0\ 0]$
$\vec{v}_2 = [1\ 0.5\ 0.2\ 0\ 0]$

สูตรการคำนวณตามหลักของ Similarity Measure Cosine

$$sim(Q, D_i) = \frac{\sum_i w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2} \sqrt{\sum_i w_{i,j}^2}} \quad (2.11)$$

$sim(Q, D_i)$ = ระยะห่าง (Cosine Similarity)
$w_{Q,j}$ = เป็นค่าดัชนี (Index) ของรูปภาพต้นแบบ
$w_{i,j}$ = เป็นค่าดัชนี (Index) ของรูปภาพในฐานข้อมูล

$$\vec{v}_1 = [1\ 0.2\ 0.2\ 0\ 0]$$
$$\vec{v}_2 = [1\ 0.5\ 0.2\ 0\ 0]$$

$$sim(\vec{v}_1, \vec{v}_2) = \frac{(1\times1) + (0.2\times0.5) + (0.2\times0.2) + (0\times0) + (0\times0)}{\sqrt{1^2+0.1^2+0.2^2} \times \sqrt{1^2+0.1^2+0.2^2}}$$

$$(\vec{v}_1, \vec{v}_2) = \frac{(1\times1) + (0.2\times0.5) + (0.2\times0.2) + (0\times0) + (0\times0)}{\sqrt{1^2+0.1^2+0.2^2} \times \sqrt{1^2+0.5^2+0.2^2}}$$

$$(\vec{v}_1, \vec{v}_2) = \frac{1.14}{1.16} = 0.98$$

จากการเปรียบเทียบค่าที่ได้นี้คือค่าที่บ่งบอกความเหมือนกันระหว่างวีดิโอ 2 เฟรม โดยที่ถ้ามีค่าความเหมือนออกมาคือมีค่ามากว่า 0 ก็แสดงว่าวีดิโอทั้ง 2 สองเฟรมนี้เหมือนกันและถ้ามีค่าเป็น 0 ก็แสดงว่าวีดิโอทั้ง 2 เฟรมนี้ไม่เหมือนกัน

2.2.4 ตัวอย่างการทำดัชนีแบบ AVI (Adaptive Video Indexing)

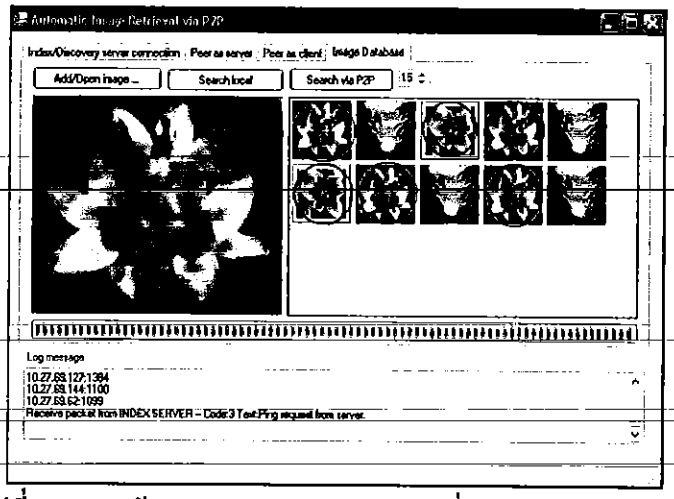
เทคนิคการทำดัชนีวีดิโอแบบ AVI นี้ยังไม่มีการแพร่หลายนักเนื่องจากเป็นเทคนิคใหม่แต่ก็ได้มีการนำไปใช้ในวิทยานิพนธ์อยู่บ้าง ซึ่งจะนำไปประยุกต์ใช้กับการค้นหาภาพและวีดิโอดังตัวอย่างที่ได้นำมา

จากวิทยานิพนธ์ "การค้นหาภาพอัตโนมัติบน Peer-to-Peer Network (Automatic Retrieval on Peer-To-Peer Network)" ซึ่งได้มีการนำการทำดัชนีแบบ AVI มาใช้ในการทำดัชนีให้กับรูปภาพที่อยู่ภายในฐานข้อมูลโดยมีขั้นตอนต่อไปนี้

2.2.3.1 ผลการทดลองจากวิทยานิพนธ์ดังกล่าว

ผลการทดลองของการค้นหาภาพเมื่อมีการใช้การทำดัชนีแบบ AVI โดยดูจากเปอร์เซ็นต์ของความแม่นยำในการค้นหา (Precision) โดยคำนวณจาก

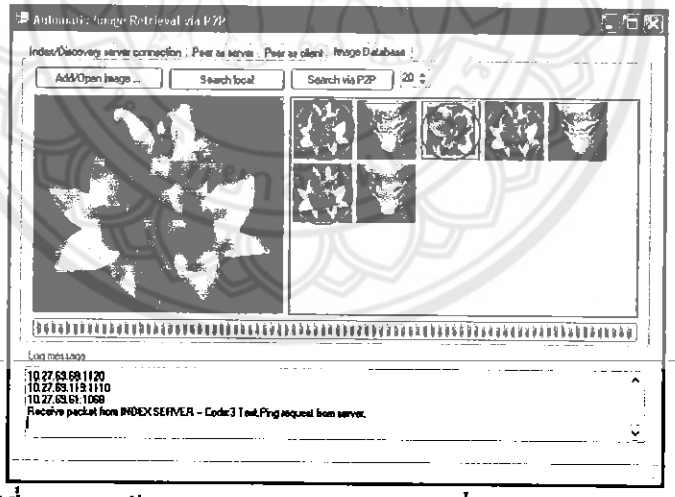
%Precision = (จำนวนรูปภาพที่ถูกเลือก / จำนวนรูปภาพทั้งหมด) × 100



รูปที่ 2.8 การค้นหารูปภาพบน Client 7 เครื่อง (Search via P2P)

จากรูปการทดสอบที่ 2.8 จะเห็นได้ว่ามีรูปที่มีลักษณะที่อยู่ในกลุ่มเดียวกับไฟล์รูปตัวอย่างทั้งหมด 6 ไฟล์ ซึ่งเวลาที่ใช้ในการค้นหาภาพเท่ากับ 21.99 วินาที เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

$$\%\text{Precision} = \left(\frac{4}{10}\right) \times 100 = 40\ \%$$



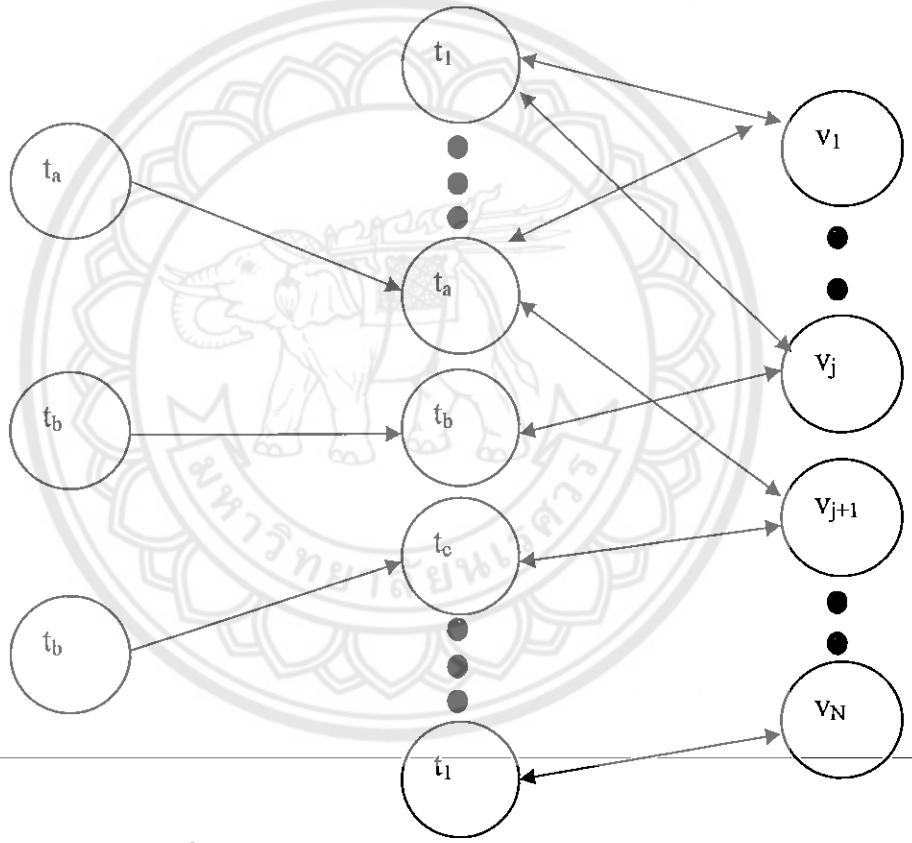
รูปที่ 2.9 การค้นหารูปภาพบน Client 10 เครื่อง (Search via P2P)

จากรูปการทดสอบที่ 2.9 จะเห็นได้ว่ามีรูปที่มีลักษณะที่อยู่ในกลุ่มเดียวกันกับไฟล์รูปตัวอย่างทั้งหมด 4 ไฟล์ เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

$$\%\text{Precision} = \left(\frac{4}{7}\right) \times 100 = 57.14\ \%$$

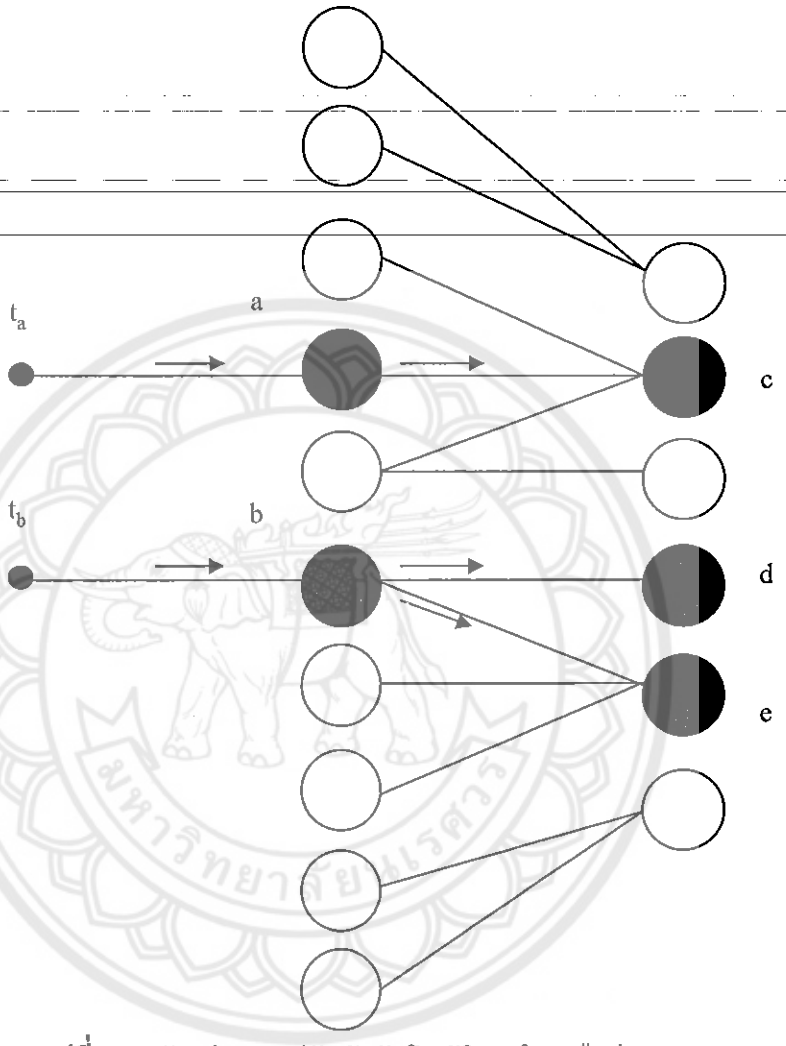
2.3 การปรับตัวอัตโนมัติภายในเครือข่าย (The Automatic Relevance Feedback Network (ARFN)) [6]

จากการทำดัชนีแบบ AVI จะทำให้ได้ต้นแบบของกลุ่มวีดิโอภายในฐานข้อมูลมาในรูปแบบของเวกเตอร์ โดยที่ต้นแบบแต่ละอันจะมีค่าพารามิเตอร์อยู่ค่าหนึ่งที่เรียกว่าค่าพารามิเตอร์ของน้ำหนัก ซึ่งค่าพารามิเตอร์น้ำหนักนี้จะทำการเปลี่ยนแปลงค่าใหม่ทุกครั้งเมื่อมีการคิวรี่หรือการค้นหาเกิดขึ้น จากนั้นจะนำเวกเตอร์ที่มีค่าพารามิเตอร์น้ำหนักใหม่นี้ไปทำดัชนีอีกรอบ เพื่อปรับปรุงการค้นหาในแต่ละรอบแล้วจึงค่อยคิวรี่ใหม่อีกรอบ จนได้วีดิโอที่มีความใกล้เคียงกันมากที่สุด โดยที่จะทำการปรับปรุงตัวของมันเองโดยอัตโนมัติ



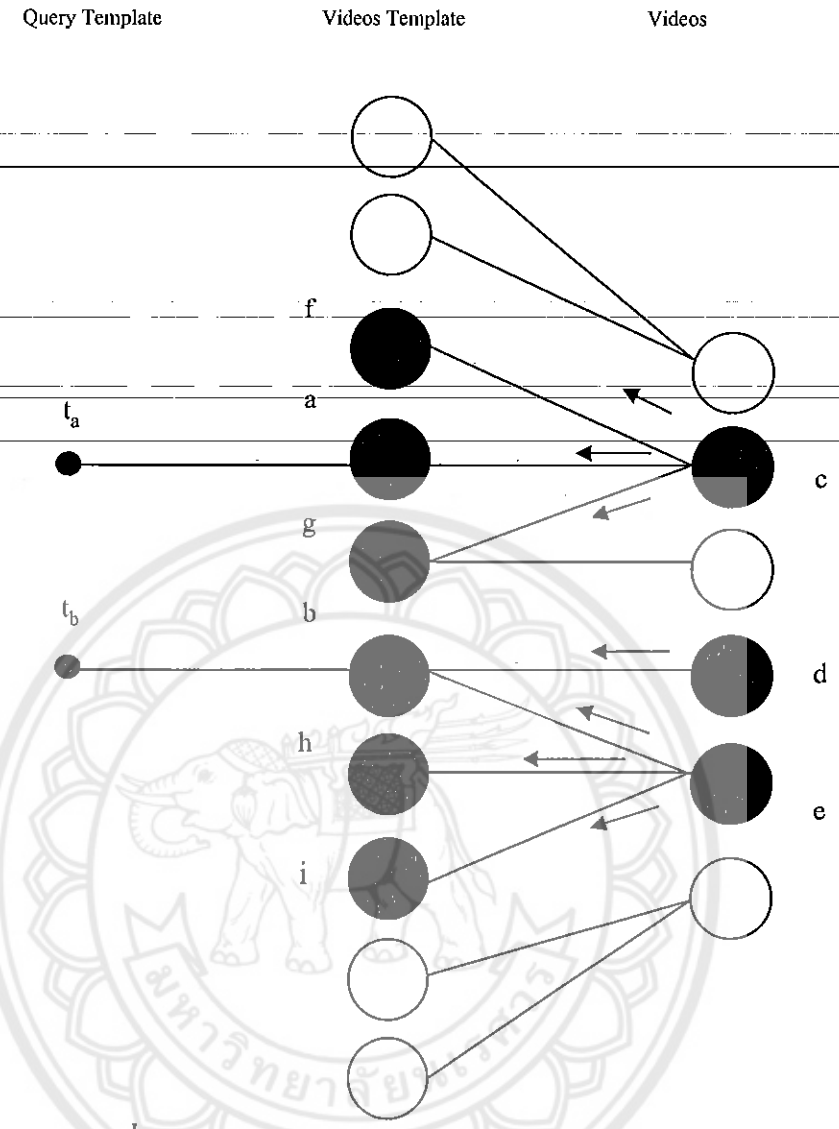
รูปที่ 2.10 รูปแสดงการปรับตัวอัตโนมัติภายในเครือข่าย

หลักการทำงานของ ARFN



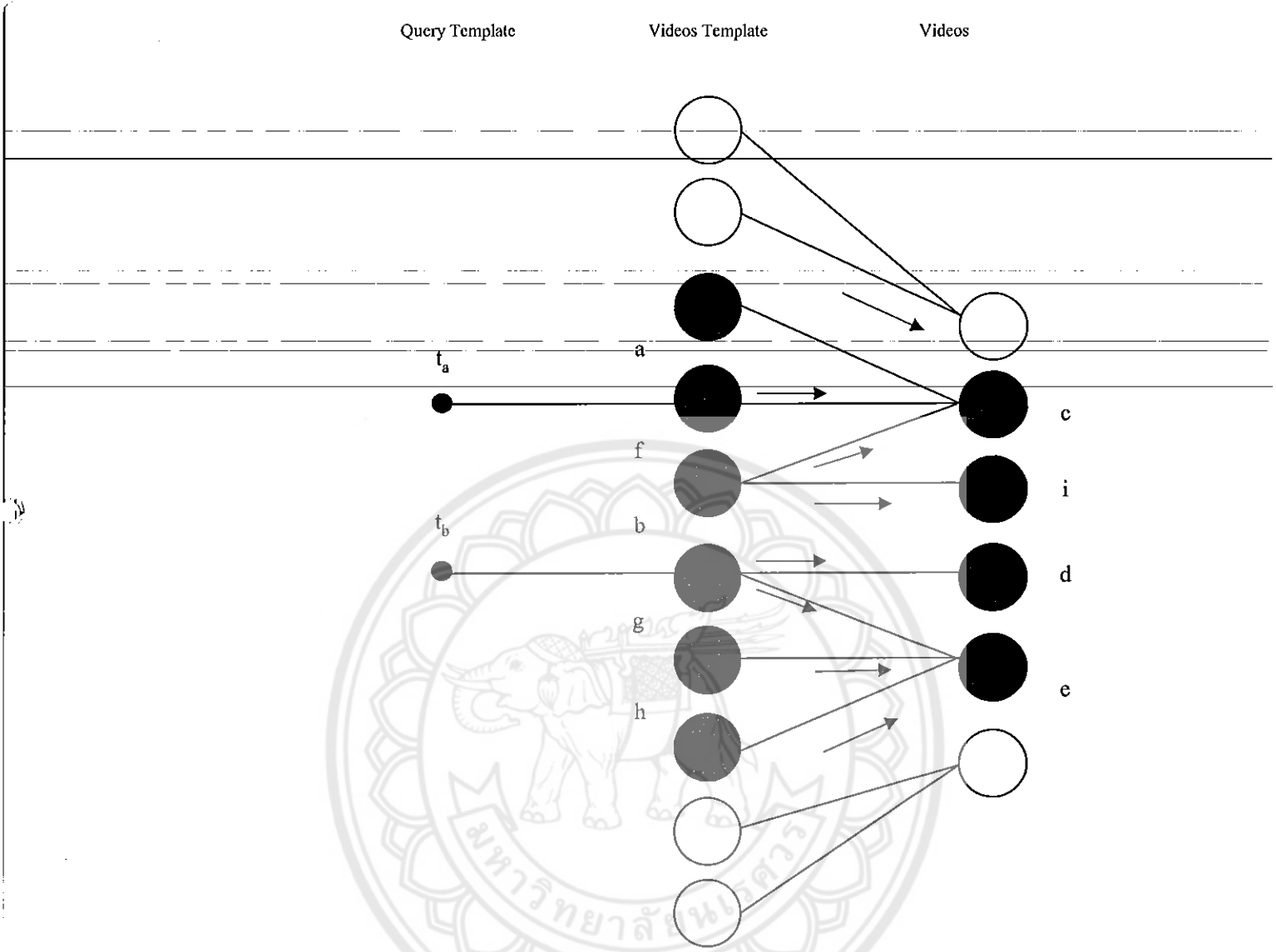
รูปที่ 2.11 ตัวอย่างการปรับตัวอัตโนมัติภายในเครือข่าย (1)

เมื่อทำการคิวรี่โดยใช้วีดิโอจากนั้นจะนำวีดิโอที่นำมาคิวรี่ไปเปรียบเทียบว่าตรงกับวีดิโอต้นแบบอันไหนเมื่อเปรียบเทียบแล้วก็จะนำไปหาว่ามีวีดิโอไหนบ้างในฐานข้อมูลที่จัดอยู่ในกลุ่มของวีดิโอต้นแบบอันนี้บ้างก็จะทำให้ได้วีดิโอออกมา โดยตรงจุดนี้จะมีการปรับปรุงค่าที่ใช้ในการคิวรี่ใหม่ โดยค่าที่ได้มาใหม่นี้เป็นค่าที่มาจากพารามิเตอร์น้ำหนักของวีดิโอที่เพิ่งได้มา



รูปที่ 2.12 ตัวอย่างการปรับตัวอัตโนมัติภายในเครือข่าย (2)

จากนั้นจะนำเวกเตอร์ที่มีค่าพารามิเตอร์น้ำหนักอันใหม่ไปเปรียบเทียบกับต้นแบบทั้งหมดอีกครั้ง จะพบว่าหลังจากการคิวรี่โดยค่าพารามิเตอร์ใหม่นี้จะตรงกับวีดิโอต้นแบบอีกหลายอัน เนื่องมาจากภาพวีดิโอที่ได้หลังจากขั้นตอนแรกนั้นอาจจะมีองค์ประกอบต่างๆ ที่ต่างหรือเพิ่มเติมจากของเดิม เช่น ภายในวีดิโอนั้นอาจจะกล่าวถึงภาพภูเขาและอาจจะมีภาพที่เป็นบุคคลปรากฏอยู่ด้วย อาจจะเป็นดารา หรือบุคคลที่ต้องการค้นหาอยู่ด้วย ทำให้ไปเปรียบเทียบตรงกับวีดิโอต้นแบบอื่นๆ ที่อาจเกี่ยวข้องกับดาราคนนั้นหรือบุคคลที่ต้องการค้นหา



รูปที่ 2.13 ตัวอย่างการปรับตัวอัตโนมัติภายในเครือข่าย (3)

หลังจากที่ได้วีดิโอต้นแบบอันใหม่แล้ว วีดิโอต้นแบบเหล่านั้นจะส่งค่าพารามิเตอร์ของตนเองออกไปค้นหาวีดิโอที่มีความใกล้เคียงกันกับวีดิโอต้นแบบนั้นๆ หรือจัดอยู่ในกลุ่มเดียวกัน ซึ่งจะทำให้ได้วีดิโอออกมาอีกหลายชุดด้วยกัน ซึ่งจะเห็นได้ว่าการค้นหาได้มีการปรับปรุงตัวมันเองให้มีประสิทธิภาพมากขึ้น เพราะจากที่ควรจะได้แค่วีดิโอเพียงสามไฟล์จากในขั้นตอนแรกและอาจจะไม่ตรงกับที่ต้องการด้วย แต่เมื่อใช้ ARFN จะทำให้ได้ผลการค้นหาที่แม่นยำและเที่ยงตรงมากขึ้น

2.3.1 ตัวอย่างแสดงการทำงานของ ARFN

จากหลักการทำงานของ ARFN ข้างต้น การที่จะได้มาซึ่งเวกเตอร์ที่มีค่าพารามิเตอร์น้ำหนักใหม่ที่มีการปรับปรุงเปลี่ยนแปลงทุกครั้งที่ทำการคิวรี่นั้นมาจากการคำนวณทางคณิตศาสตร์โดยใช้หลักการของ Similarity Measure Cosine โดยหลักการนี้จะนำมาใช้ในการเปรียบเทียบ AVI โดยจะนำวีดิโอมาเปรียบเทียบกับต้นแบบทีละตัว ซึ่งจะมีวิธีการหาค่าเวกเตอร์ที่มีค่าพารามิเตอร์น้ำหนักใหม่ดังนี้

กำหนดเวกเตอร์ของวีดิโอต่างๆ ดังนี้

$\vec{v}_n = [t_1, t_2, t_3, t_4, t_5]$

$\vec{v}_1 = [0.1, 0.2, 0, 0, 0] = \vec{v}_q$

$\vec{v}_2 = [0.2, 0.5, 0, 0, 0]$

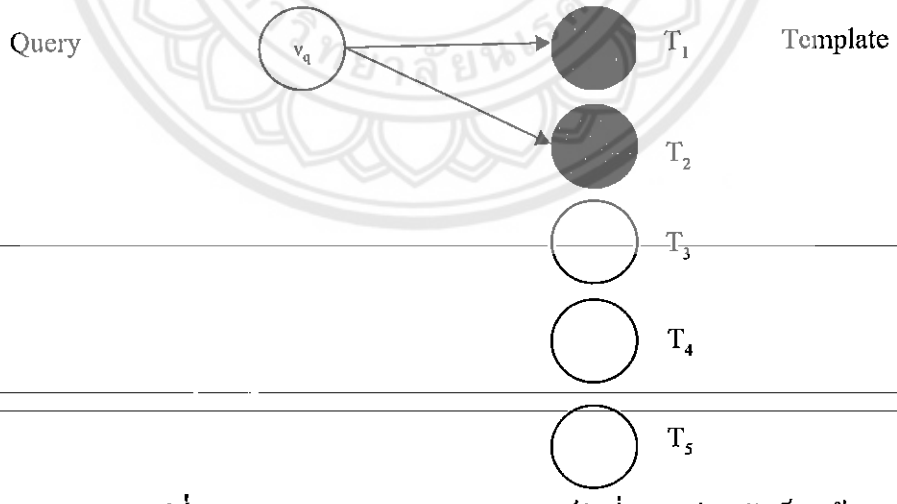
$\vec{v}_3 = [0.2, 0.5, 0.1, 0, 0]$

$\vec{v}_4 = [0, 0, 0, 0, 0]$

$\vec{v}_5 = [0, 0, 0.1, 0.2, 0]$

$\vec{v}_6 = [0, 0, 0.1, 0.3, 0]$

เมื่อ $\vec{v}_q$ คือค่าคิวรี่ที่ใช้ในการค้นหาวีดิโอ และค่าพารามิเตอร์น้ำหนัก (w) ภายในเวกเตอร์นั้นจะมีค่า $0 \leq w \leq 1$ จะเห็นว่าเวกเตอร์คิวรี่นี้จะเปรียบเทียบเจอกับต้นแบบ 2 ตัว



รูปที่ 2.14 แสดงตัวอย่างการนำเวกเตอร์คิวรี่ (v_q) เทียบกับช็อตต้นแบบ

จากนั้นจะทำการเปรียบเทียบค่าของเวกเตอร์คิวรี่ ($\vec{v}_q$) กับเวกเตอร์ของวีดิโอต่างๆ โดยใช้หลักการของ Similarity Measure Cosine จะได้ค่าดังนี้

$\vec{v}_1$ มี $(\vec{v}_q, \vec{v}_1) = 1$
$\vec{v}_2$ มี $(\vec{v}_q, \vec{v}_2) = 0.5$
$\vec{v}_3$ มี $(\vec{v}_q, \vec{v}_3) = 0.45$
$\vec{v}_4$ มี $(\vec{v}_q, \vec{v}_4) = 0$
$\vec{v}_5$ มี $(\vec{v}_q, \vec{v}_5) = 0$
$\vec{v}_6$ มี $(\vec{v}_q, \vec{v}_6) = 0$

ซึ่งตอนนี้จะสังเกตเห็นได้ว่าหลังจากเปรียบเทียบแล้วจะมีค่าเกิดขึ้นเมื่อมีการเปรียบเทียบกับ $\vec{v}_2$ และ $\vec{v}_3$ ซึ่งนั่นหมายความว่าการคิวรี่ด้วย $\vec{v}_q$ นั้นจะได้วีดิโอที่เหมือนหรือใกล้เคียงกัน 2 ไฟล์ ซึ่งก็คือ $\vec{v}_2$ และ $\vec{v}_3$

จากนั้นจะทำการปรับปรุงค่าคิวรี่นี้ใหม่หลังจากการคิวรี่ในรอบแรกแล้ว โดยจะเรียกว่าเวกเตอร์คิวรี่นี้ใหม่ว่า $\vec{v}_q^*$ โดยวิธีการหาค่าเวกเตอร์นี้จะใช้สูตรดังต่อไปนี้

$$a_r^{(t)} = \frac{l_r}{\left(\sum_{r=1}^{R} l_r^2\right)^{1/2}}$$

$$l_r = w_{qr} + \alpha \sum_{j \in Pos} a_r^{(t)} \overline{w}_{jr} - \beta \sum_{j \in Neg} a_j^{(j)} \overline{w}_{jr}$$

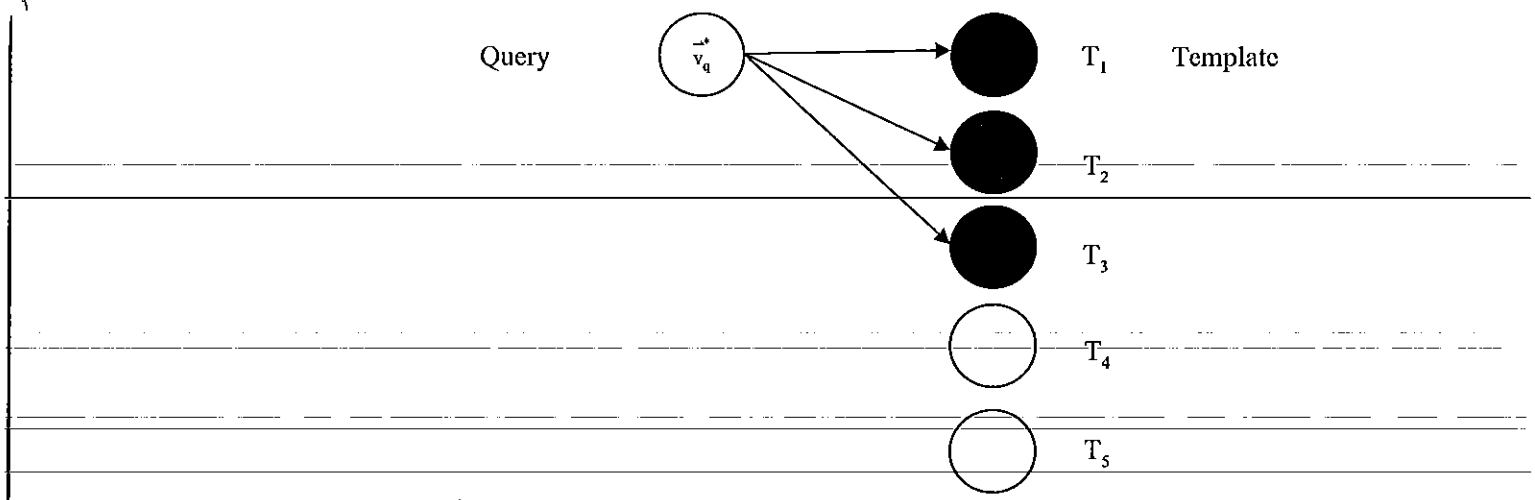
เมื่อ $0 \leq \alpha, \beta \leq 0$ และ $\alpha \gg \beta$

โดยที่ค่า $\vec{v}_q^*$ จะมีค่าดังนี้

$$\vec{v}_q^* = \left[\left(\frac{0.1+0.2+0.2}{3}\right), \left(\frac{0.2+0.5+0.5}{3}\right), \left(\frac{0+0+0.1}{3}\right), \frac{0}{3}, \frac{0}{3}\right]$$

$\vec{v}_q^* = [0.16, 0.4, \mathbf{0.03}, 0, 0]$

มีค่า 0.03 ปรากฏขึ้นที่ตำแหน่งของต้นแบบ t_3 นั่นหมายความค่าคิวรี่ใหม่นี้เปรียบเทียบตรงกันกับต้นแบบ t_3 ด้วย



รูปที่ 2.15 แสดงตัวอย่างการนำเวกเตอร์คิวรี่ที่ปรับตัวแล้ว ($\vec{v}_q^*$) เทียบกับช็อตต้นแบบ

จะเห็นว่าเวกเตอร์คิวรี่มีเปลี่ยนแปลงใหม่นี้จะเปรียบเทียบเจอกับตัวต้นแบบที่มากขึ้นกว่าเดิมคือเป็น 3 ตัว และเมื่อนำค่า $\vec{v}_q^*$ ไปทำการคิวรี่ใหม่อีกครั้งจะให้ผลดังนี้

$\vec{v}_q^*$ = [0.16, 0.4, **0.03**, 0, 0]
$\vec{v}_1$ = [0.1, 0.2, 0, 0, 0]
$\vec{v}_2$ = [0.2, 0.5, 0, 0, 0]
$\vec{v}_3$ = [0.2, 0.5, 0.1, 0, 0]
$\vec{v}_4$ = [0, 0, 0, 0, 0]
$\vec{v}_5$ = [0, 0, **0.1**, 0.2, 0]
$\vec{v}_6$ = [0, 0, **0.1**, 0.3, 0]

ทำการเปรียบเทียบค่าของเวกเตอร์คิวรี่ ($\vec{v}_q^*$) กับเวกเตอร์ของวีดิโอต่างๆ โดยใช้หลักการของ Similarity Measure Cosine จะได้ค่าดังนี้

$\vec{v}_1$ มี $sim(\vec{v}_q^*, \vec{v}_1) = 0.85$
$\vec{v}_2$ มี $sim(\vec{v}_q^*, \vec{v}_2) = 0.7$
$\vec{v}_3$ มี $sim(\vec{v}_q^*, \vec{v}_2) = 0.65$
$\vec{v}_4$ มี $sim(\vec{v}_q^*, \vec{v}_2) = 0$
$\vec{v}_5$ มี $sim(\vec{v}_q^*, \vec{v}_2) = 0.34$
$\vec{v}_6$ มี $sim(\vec{v}_q^*, \vec{v}_2) = 0.41$

ซึ่งตอนนี้จะสังเกตเห็นได้ว่าหลังจากเปรียบเทียบแล้วจะมีค่าเกิดขึ้นเมื่อมีการเปรียบเทียบกับ $\vec{v}_5$ และ $\vec{v}_6$ ซึ่งนั่นหมายความว่าการคิวรี่ด้วย $\vec{v}_q^*$ ใหม่นั้นจะได้วีดิโอที่เหมือนหรือใกล้เคียงกันเพิ่มอีก 2 ไฟล์ ซึ่งก็คือ $\vec{v}_5$ และ $\vec{v}_6$ เพิ่มขึ้นมา นั่นหมายความว่าหลังจากการใช้ ARFN แล้วทำให้การค้นหาวีดิโอได้ผลการค้นหาที่แม่นยำและเที่ยงตรงมากขึ้น

2.4 เพียร์ทูเพียร์ (Peer-to-Peer)

กลุ่มบริษัทที่ทำงานในเรื่อง เพียร์ทูเพียร์ (Peer-to-Peer) ซึ่งประกอบด้วยบริษัทขนาดใหญ่ในอุตสาหกรรม อันได้แก่ อินเทล (Intel) ซิสโก (Cisco) และ เอชพี (HP) ได้อธิบายเกี่ยวกับ เพียร์ทูเพียร์ไว้ว่า "อธิบายอย่างง่ายได้ว่า เพียร์ทูเพียร์ คือ การใช้ ทรัพยากรทางคอมพิวเตอร์ และเซอร์วิสร่วมกัน โดยการโต้ตอบกันโดยตรงระหว่างระบบ" [7] แต่ถึงอย่างนั้น การจะบอกว่าสิ่งไหนเป็นหรือไม่เป็น เพียร์ทูเพียร์ก็ยังเป็นเรื่องยาก ในการอธิบายถึงเพียร์ทูเพียร์ ได้ถูกอธิบายไว้หลายความหมาย ต่างคน ต่างความคิด ซึ่งเพียร์ทูเพียร์อาจถูกอธิบายในเรื่องของแนวคิด สภาพแวดล้อม เครื่องมือ โมเดล (Model) หรือคุณสมบัติของระบบ Clay Shirkey ผู้แต่งหนังสือ O'Reilly Network ได้อธิบาย ว่า เพียร์ทูเพียร์ คือ "คลาส (class) ของแอพพลิเคชั่น (application) ที่ใช้ทรัพยากร (หน่วยความจำ สารสนเทศ) ให้เป็นประโยชน์ บนเอดจ์ของอินเตอร์เน็ตที่สามารถใช้ได้" [8] เนื่องจาก เป็นการเข้าถึง (access) ทรัพยากรที่อยู่แบบกระจาย (Decentralized) ดังนั้นสภาพแวดล้อมการเชื่อมต่อจะไม่เสถียรและ ไม่สามารถคาดเดาไอพีแอดเดรส (IP Address)ได้ ดังนั้น เพียร์ทูเพียร์โหนด (P2P Node) จะต้องกระทำการนอกระบบ ดีเอ็นเอส (DNS) และแยกตัวออกจาก เซิร์ฟเวอร์ (server) ที่เป็นศูนย์รวม

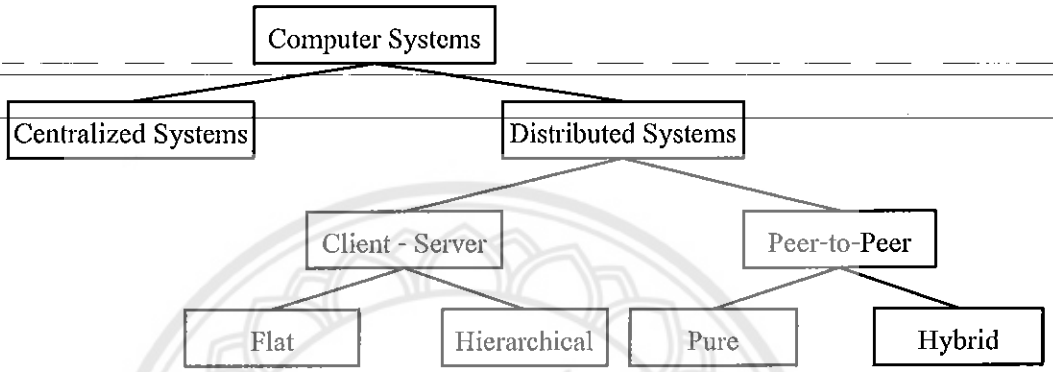
ระบบเพียร์ทูเพียร์ คือ ระบบที่เพียร์เป็นอิสระต่อกัน แต่มีการพึ่งพาสารสนเทศจากเพียร์อื่น ซึ่งเพียร์จะเชื่อมต่อเพียร์เน็ตเวิร์ก (peers network) เข้าด้วยกัน โดยที่เพียร์อาจเป็นพีดีเอ(PDA) พีซี (PC) หรืออุปกรณ์ต่างบนระบบเครือข่าย

เนื่องจากหลายเหตุการณ์ และหลายปัจจัยทำให้เกิดการเปลี่ยนแปลงระบบอินเตอร์เน็ตจากเพียร์ทูเพียร์ มาสู่สถาปัตยกรรมแบบ ไคลเอนท์-เซิร์ฟเวอร์ (Client-Server) ที่มีมากในปัจจุบัน เมื่ออินเตอร์เน็ตเริ่มมีบทบาทในทางธุรกิจ องค์กร บริษัทก็ต่างสร้าง ไฟร์วอล์ (firewall) เพื่อควบคุมการเข้าถึงข้อมูลของตน และคนส่วนใหญ่ใช้เครื่องพีซี (PC) ในการเข้าสู่อินเตอร์เน็ต ซึ่งความสามารถแตกต่างจากเครื่องเซิร์ฟเวอร์ ที่ใช้เก็บข้อมูลเป็นอย่างมากทำให้ไม่เหมาะกับลักษณะ เพียร์ทูเพียร์ นอกจากนี้แอพพลิเคชั่นที่ใช้ รวมถึงเวิลด์ไวด์เว็บ (World Wide Web) และเอฟทีพี (FTP) ก็ยังมีพื้นฐานอยู่บนสถาปัตยกรรมแบบไคลเอนท์-เซิร์ฟเวอร์ [7-9]

ในไม่กี่ปีมานี้เทคโนโลยี เพียร์ทูเพียร์ ได้กลับมามีบทบาทอย่างมากกับอินเตอร์เน็ต และข้อมูลที่ถูกเก็บแบบกระจายอีกครั้ง และโดยเฉพาะอย่างยิ่งในเรื่องของสื่อ (media) ต่างๆ [8]

2.4.1 ระบบคอมพิวเตอร์ (Computer Systems)

ระบบคอมพิวเตอร์ถูกแบ่งออกเป็นระบบย่อยๆ ดังรูปที่ 2.16 โดยที่ระบบคอมพิวเตอร์จะถูกแบ่งออกเป็น ระบบรวมศูนย์ (Centralized) และระบบกระจาย (Decentralized) ซึ่งระบบรวมศูนย์จะมีทรัพยากรณ์คอมพิวเตอร์ที่ไม่ทำงานร่วมกับอุปกรณ์อื่น ในขณะที่ระบบกระจายจะมีการติดต่อสื่อสาร และทำงานร่วมกันบนเครือข่าย [9]

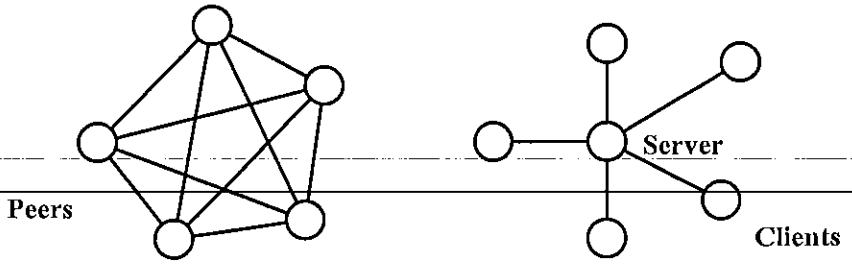


รูปที่ 2.16 แสดงระบบคอมพิวเตอร์ที่ถูกแบ่งออกเป็นแต่ละประเภท [9]

จากรูปเห็นได้ว่าระบบกระจาย ยังถูกแบ่งออกเป็น 2 โมเดล คือ ไคลเอนท์-เซิร์ฟเวอร์ และเพียร์ทูเพียร์ (ความแต่งต่างระหว่าง 2 ระบบนี้จะถูกอธิบายในหัวข้อถัดไป) นอกจากนั้น ไคลเอนท์-เซิร์ฟเวอร์ โมเดล ยังถูกแบ่งได้อีก 2 โมเดลย่อย คือ โมเดลที่ให้ ไคลเอนท์ เชื่อมต่อกับเซิร์ฟเวอร์เดียวเท่านั้นเรียกว่า แฟลท ไคลเอนท์-เซิร์ฟเวอร์ (Flat Client-Server) และโมเดล ไฮราคีคอล ไคลเอนท์-เซิร์ฟเวอร์(Hierarchical Client-Server)ที่ถูกจัดเป็นลำดับชั้น โดย เซิร์ฟเวอร์ที่อยู่ระดับใดๆ จะให้บริการกับไคลเอนท์ ที่อยู่ในชั้นที่สูงกว่า เช่นกัน เพียร์ทูเพียร์ โมเดล ถูกแบ่งออกเป็น 2 โมเดลย่อย คือ โมเดลที่ไม่ใช้เซิร์ฟเวอร์ในการเก็บ เมต้า-อินฟอร์เมชัน (meta-information) ของเพียร์ เรียกว่า เพียว เพียร์ทูเพียร์ (Pure Peer-to-Peer) และ โมเดลที่ใช้เซิร์ฟเวอร์ช่วยเก็บ เมต้า-อินฟอร์เมชัน (meta-information) และคอยให้บริการบางอย่าง เรียกว่า ไฮบริด เพียร์ทูเพียร์ (Hybrid Peer-to-Peer) [9]

2.4.2 เพียร์ทูเพียร์ กับ ไคลเอน์-เซิร์ฟเวอร์ (P2P vs. Client-Server)

เพียร์ทูเพียร์โมเดล เป็นอีกทางเลือกหนึ่งนอกจาก ไคลเอนท์-เซิร์ฟเวอร์โมเดล ที่มักใช้กับเครือข่ายในปัจจุบัน โดยที่ในไคลเอนท์-เซิร์ฟเวอร์ โมเดลจะมี เซิร์ฟเวอร์ หรือกลุ่มเซิร์ฟเวอร์ขนาดเล็กที่คอยให้บริการต่างกับไคลเอนท์ ซึ่งแสดงความแตกต่างดังรูปที่ 2.17 ซึ่งในทางกลับกัน ในเพียร์ทูเพียร์โมเดล จะไม่พึ่งพาอาศัยเซิร์ฟเวอร์ ทุกเพียร์จะมีเชื่อมต่อกันด้วยเงื่อนไขเดียวกัน แต่หากคิดว่าเครือข่ายเพียร์ทูเพียร์ที่ดีไม่ต้องใช้เซิร์ฟเวอร์เลย นั้นก็เป็นความเข้าใจที่ผิด ซึ่งจะพบเห็นได้กับเครือข่ายเพียวเพียร์ทูเพียร์ แต่เพียร์ทูเพียร์แอพพลิเคชันส่วนใหญ่มักทำงานกับเซิร์ฟเวอร์ [10]



รูปที่ 2.17 แสดงความแตกต่างระหว่างเพียร์ทูเพียร์โมเดล กับ ไคลเอนท์-เซิร์ฟเวอร์ [9]

ข้อดีของ ไคลเอนท์-เซิร์ฟเวอร์ คือ เป็นที่รู้จักกันเป็นส่วนใหญ่ ถูกนำมาใช้มากในปัจจุบัน ผลลัพธ์จากการทำงานเป็นไปตามมาตรฐาน เนื่องจากเป็นระบบรวมศูนย์จึงทำให้ง่ายต่อการปรับปรุง และง่ายต่อการควบคุมประสิทธิภาพ มีความปลอดภัย (security) และความเชื่อถือได้ แต่ในทางกลับกัน ไคลเอนท์เซิร์ฟเวอร์ มีข้อจำกัดในเรื่องของการขยายขนาดของเครือข่าย ความยืดหยุ่น และค่าใช้จ่ายที่มากกว่า เพียร์ทูเพียร์เน็ตเวิร์ก [9]

2.4.3 ประวัติของเพียร์ทูเพียร์ (History of P2P)

แอพพลิเคชั่น (application) ที่มีการใช้งานของ FTP ในสมัยก่อน ถือได้ว่าเป็น เพียร์ทูเพียร์แอพพลิเคชั่น เลยก็ว่าได้ แต่เริ่มเดิมทีแล้ว ประเภทของระบบส่วนใหญ่ที่นักวิชาการ หรือบุคคลฝ่ายเทคนิค เป็นผู้ใช้ จะแตกต่างกันไปขึ้นอยู่กับการจัดการ และการเลือกใช้ให้เหมาะสมกับงาน หลังจาก ประมาณ ค.ศ. 1980 – 1990 ที่มีคอมพิวเตอร์ส่วนบุคคลที่ถูกใช้กันอย่างแพร่หลาย ระบบคอมพิวเตอร์ส่วนใหญ่จึงเป็นแบบไคลเอนท์-เซิร์ฟเวอร์ ซึ่งง่ายต่อการบริหารและจัดการระบบ ซึ่งคอมพิวเตอร์ส่วนบุคคลที่มีความสามารถในการทำงานน้อยกว่าเฟรมเวิร์กคอมพิวเตอร์จะทำหน้าที่เป็นไคลเอนท์ เมื่อคอมพิวเตอร์ถูกพัฒนาให้มีหน่วยความจำ และประสิทธิภาพมากขึ้น ทำให้เกิดระบบกระจายขึ้น ซึ่งในตอนนั้นมีแนพสเตอร์(Napster) ซึ่งเป็นโปรแกรมที่ทำให้ผู้ใช้สามารถแชร์ไฟล์เพลงบนเครือข่ายอินเตอร์เน็ตกับผู้คนจำนวนมากในระบบเพียร์ทูเพียร์

2.4.4 คุณลักษณะของเพียร์ทูเพียร์ (Characteristics of P2P)

เพียร์ทูเพียร์ถูกนำไปใช้ด้วยจุดประสงค์บางประการหรือหลายประการ ได้แก่ การแบ่งค่าใช้จ่าย ลดค่าใช้จ่าย เพื่อทำให้สามารถขยายขนาดได้ มีความน่าเชื่อถือ การใช้ทรัพยกรตามข้อตกลงร่วมกัน การทำงานร่วมกัน เพิ่มความเป็นอิสระ ป้องกันความลับ ความยืดหยุ่น หรือการติดต่อสื่อสารแบบ แอด-ฮอค (ad-hoc) และเพื่อให้เกิดการร่วมมือกัน [9]

เมื่อเปลี่ยนจากระบบที่อ้างอิงแอดเดรสเป็นระบบเพียร์ทูเพียร์ จะต้องแก้ไขหรือทำให้เกิดสิ่งต่างๆดังต่อไปนี้ [9]

2.4.4.1 การกระจาย (Decentralization) โดยปกติแล้วเน็ตเวิร์กแอพพลิเคชั่นจะทำงานบนแบบโมเดล ไคลเอนท์-เซิร์ฟเวอร์ ที่มีสารสนเทศรวมอยู่ที่เซิร์ฟเวอร์ที่เป็นศูนย์กลาง ซึ่งในความเป็นมีปัญหาที่เกิดขึ้นกับโมเดลนี้ อาจทำให้เกิดปัญหาคอขวดเมื่อไคลเอนท์ติดต่อไปยังเซิร์ฟเวอร์

จำนวนมากในเวลาเดียวกัน ซึ่งทรัพยากรของไคลเอนท์จะไม่ถูกใช้อย่างเต็มที่ แต่เซิร์ฟเวอร์จะถูกใช้ทรัพยากรจำนวนมาก ปัญหาดังกล่าวนี้สามารถแก้ได้โดยใช้ระบบกระจาย ซึ่งแบ่งออกเป็น 2 ลักษณะคือ เพียว เพียร์ทูเพียร์ และไฮบริด เพียร์ทูเพียร์

2.4.4.2 ข้อจำกัดของขนาด (Scalability) เพียร์ทูเพียร์ช่วยแก้ไขปัญหาเรื่องขนาดของเครือข่ายได้ ซึ่งระบบรวมศูนย์จำเป็นต้องทำงานแบบเป็นจังหวะ (synchronization) และจดจำสถานะจำนวนมากซึ่งส่งผลต่อข้อจำกัดของเครือข่าย ตัวอย่างเช่น แนพสเตอร์ (Napster) ที่มีความสามารถในการให้ไคลเอนท์จำนวนมากแชร์ไฟล์เพลงแบบไม่มีข้อจำกัดในเรื่องของจำนวนของไคล์เอนท์

2.4.4.3 ค่าใช้จ่าย (Cost of ownership) ในโครงสร้างที่เป็นแบบรวมศูนย์จะทำให้ภาระค่าใช้จ่ายถูกรวมอยู่ที่เซิร์ฟเวอร์เพียงที่เดียว แต่หากใช้เพียร์ทูเพียร์จะช่วยแก้ปัญหานี้ โดยการแบ่งภาระค่าใช้จ่ายให้กับ เพียร์แต่ละเพียร์ ซึ่งไม่ทำให้เกิดภาระค่าใช้จ่ายอยู่เพียงจุดๆเดียว

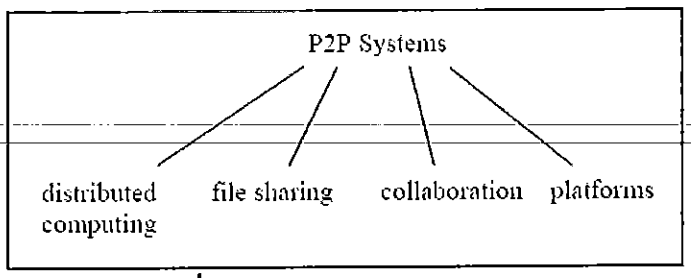
2.4.4.4 ประสิทธิภาพ (Performance) คอมพิวเตอร์ส่วนมากจะมีการกระทำแบบเสมือนจริง (virtual) โดยขึ้นอยู่กับระบบและระบบเพียร์ทูเพียร์ด้วย โดยในเชิงทฤษฎีแล้วระบบเพียร์ทูเพียร์จะมีประสิทธิภาพที่สูง เช่น หน่วยเก็บข้อมูล หน่วยประมวลผล ซึ่งทรัพยากรมีความต่างกัน

2.4.4.5 ความปลอดภัย (Security) ระบบเพียร์ทูเพียร์มักมีการใช้ระบบความปลอดภัยร่วมกัน ซึ่งเหมือนกับระบบกระจายทั่วไป แต่ว่ามีความต้องการใหม่ที่ต่างจากระบบอื่น

2.4.4.6 ความสามารถดำเนินอย่างทั่วถึง (Interoperability) ในปัจจุบันยังไม่มีมาตรฐานในการสร้างเพียร์ทูเพียร์แอพพลิเคชั่นที่ทำงานอย่างทั่วถึง แต่มีเจเอกซ์ทีเอ (JXTA) ที่สร้างสภาพแวดล้อมแบบกระจาย ซึ่งเป็นทางออกหนึ่งในการสร้างเพียร์ทูเพียร์แอพพลิเคชั่น

2.4.5 ระบบเพียร์ทูเพียร์ (P2P Systems)

ระบบเพียร์ทูเพียร์สามารถแบ่งออกเป็น 4 ประเภท โดยแบ่งจากลักษณะการทำงานร่วมกันและจุดประสงค์ในการทำงาน ดังรูปที่ 2.18



รูปที่ 2.18 แสดงระบบเพียร์ทูเพียร์

2.4.5.1 การคำนวณแบบกระจาย (distributed computing) การคำนวณแบบกระจายคือการใช้ทรัพยากรที่ยังไม่ถูกใช้งาน เช่น ซีพียู มิพส์ (CPU MIPS) พื้นที่ดิสก์ เมื่อต้องมีการคำนวณ

จำนวนมาก ระบบคำนวณแบบกระจายจะทำงานในลักษณะเพียร์ทูเพียร์ที่แบ่งงานให้กับคอมพิวเตอร์เครื่องอื่นคำนวณอย่างอิสระต่อกัน ในขณะที่มีเซิร์ฟเวอร์ที่ทำหน้าที่กระจายงานและรวบรวมผลลัพธ์จากการคำนวณของแต่ละเครื่อง ตัวอย่างแอพพลิเคชั่นที่ใช้ระบบการคำนวณแบบกระจาย เช่น Avaki และ SETI@home

2.4.5.2 การแชร์ไฟล์ (file sharing) แอพพลิเคชั่นที่ใช้ระบบการแชร์ไฟล์จะทำให้ไฟล์ของผู้ใช้เป็นประโยชน์ต่อผู้ใช้คนอื่นที่อยู่ในระบบ โดยผู้ใช้สามารถที่จะค้นหาชื่อไฟล์ และเลือกที่จะดาวน์โหลดไฟล์ที่ต้องการได้ ตัวอย่างแอพพลิเคชั่นที่มีระบบการแชร์ไฟล์ เช่น แนพสเตอร์ และฟรีเน็ต (Freenet)

2.4.5.3 คอลลอบอราชั่น (Collaboration) เพียร์ทูเพียร์แอพพลิเคชั่นประเภทนี้จะทำให้ผู้ใช้ทำงานร่วมกันอยู่บนระดับแอพพลิเคชั่น นั้นคือ ผู้ใช้หลายคนสามารถทำงานร่วมกันได้ เมื่อมีผู้ใช้คนใดเปลี่ยนแปลงงาน งานที่อยู่ที่ผู้ใช้คนอื่นก็จะถูกเปลี่ยนให้เหมือนกัน แต่การออกแบบกับระบบประเภทนี้ต้องคำนึงถึง 2 ประการ คือ ประการแรก ไม่มีเซิร์ฟเวอร์ในการส่งข้อความบอกให้กับเพียร์อื่น และควบคุมให้เป็นจังหวะเดียวกัน ประการที่สอง คือ เพียร์แต่ละเพียร์มีความสามารถที่ต่างกัน

2.4.5.4 แฟลตฟอร์ม (Platforms) คือแอพพลิเคชั่นที่ทำหน้าที่ในการสร้างฟังก์ชั่นการทำงานในระบบเพียร์ทูเพียร์ ซึ่งมี 2 ประเภท คือ ประเภทแรก เจเอกซ์ทีเอซึ่งเป็นโอเพนซอร์ส และประเภทที่สองคือ ดอทเน็ต (.NET) ซึ่งเป็นเทคโนโลยีของไมโครซอฟท์ (Microsoft)

2.4.6 สถาปัตยกรรมเพียร์ทูเพียร์ (Peer-to-Peer Architecture)

หากพิจารณาระบบอินเตอร์เน็ต จะพบว่าคอมพิวเตอร์หลายล้านเครื่องที่เชื่อมต่อกันอยู่ในเครือข่าย โดยทฤษฎีแล้วคอมพิวเตอร์จะเชื่อมต่ออยู่กับคอมพิวเตอร์เครื่องอื่น และข้อมูลก็ถูกเก็บไว้บนระบบที่สามารถเข้าถึงข้อมูลนั้นได้ โดยทั่วๆ ไป โทโพโลจี (topology) ของคอมพิวเตอร์บนอินเตอร์เน็ตเป็นแบบกลุ่มที่แยกมาเป็นโลคอลเน็ตเวิร์ก (local network) ซึ่งในแต่ละกลุ่มคอมพิวเตอร์จะมองเห็นคอมพิวเตอร์ที่อยู่ในกลุ่มเดียวกัน และในบางครั้งก็จะเชื่อมต่อกับอินเตอร์เน็ต [7]

คอมพิวเตอร์บางเครื่องเป็นเซิร์ฟเวอร์และโฮสต์ (host) เก็บข้อมูล เช่น Google ซึ่งมีเว็บเซิร์ฟเวอร์ (web server) คอยให้บริการ สำหรับเครื่องที่เปิด Google บนโลคอลเน็ตเวิร์ก ผ่านอินเตอร์เน็ตทำหน้าที่เป็นไคลเอนท์ โดยการติดต่อกันในลักษณะ ไคลเอนท์-เซิร์ฟเวอร์ มีการโต้ตอบกันในช่วงเวลาใดเวลาหนึ่งระหว่างไคลเอนท์กับเซิร์ฟเวอร์มาก และนอกจากไคลเอนท์จะเป็นผู้เปิด Google แล้วยังสามารถที่จะแชร์ไดร์ (share drive) ในกลุ่มคอมพิวเตอร์ได้อีกด้วย ในสถานการณ์นี้จะถือว่าเครื่องดังกล่าวเป็นเซิร์ฟเวอร์ที่ยอมให้ไคลเอนท์เครื่องอื่นเข้าถึงข้อมูลในไดร์ของตน

โดยมากระบบเพียร์ทูเพียร์จะแบ่งไคลเอนท์กับเซิร์ฟเวอร์ออกจากกันไม่ชัดเจน นอกจากนี้คอมพิวเตอร์สามารถเชื่อมต่อกับคอมพิวเตอร์เครื่องอื่นที่ใช้โทโพโลจีแบบวงแหวน (token-ring topology) แต่ระบบเพียร์ทูเพียร์อาจจะเป็นสถาปัตยกรรมอื่นก็ได้ [7]

ส่วนมากแล้วเพียร์จะเชื่อมต่อกับเพียร์อื่นบนอินเตอร์เน็ต โดยใช้ ทีซีพีโปรโตคอล (TCP protocol) หรือ เอชทีทีพีโปรโตคอล (HTTP protocol) โดยที่เอชทีทีพีโปรโตคอลถูกสร้างอยู่ชั้นบนของทีซีพี/ไอพี (TCP/IP) และยอมให้ติดต่อกันบนพอร์ต 80 ซึ่งทำให้เอชทีทีพีโปรโตคอลเป็นที่นิยมมากในระบบเพียร์ทูเพียร์เนื่องจากองค์กรส่วนมากจะเก็บพอร์ต 80 ไว้นอกไฟร์วอลล์ [7]

2.5 จาวา ซ็อกเก็ต (Java Socket)

ซ็อกเก็ต หมายถึง จุดเชื่อมต่อระหว่างโปรแกรมที่ทำงานอยู่บนเครือข่าย นอกจากนี้แอพพลิเคชันยังเข้าถึงเครือข่ายและส่งข้อมูลผ่านซ็อกเก็ต ซ็อกเก็ตแบ่งออกตามจุดประสงค์การใช้งานและแพลตฟอร์มได้ดังนี้

- ยูนิกซ์ โดเมน ซ็อกเก็ต (Unix Domain Sockets)
- อินเตอร์เน็ต โดเมน ซ็อกเก็ต (Internet Domain Sockets)
- เอ็นเอส โดเมน ซ็อกเก็ต (NS Domain Sockets)

จากซ็อกเก็ตทั้ง 3 ชนิดนี้ มีเพียง อินเตอร์เน็ต โดเมน ซ็อกเก็ต เท่านั้นที่สามารถใช้งานได้โดยที่ไม่ยึดติดกับแพลตฟอร์มใด ๆ ซึ่งจาวาสนับสนุนการทำงานของซ็อกเก็ตแบบ อินเตอร์เน็ต โดเมน ซ็อกเก็ตเท่านั้น [11]

2.5.1 อินเตอร์เน็ต โดเมน ซ็อกเก็ต (Internet Domain Sockets)

การทำงานหลักของอินเตอร์เน็ต โดเมน ซ็อกเก็ต เป็นการทำงานผ่านอินเตอร์เน็ต ซ็อกเก็ต (Internet Socket) เรียกทั่วไปว่า ซ็อกเก็ต (Socket) หรือ เน็ตเวิร์ค ซ็อกเก็ต (Network Socket) ซึ่งอินเตอร์เน็ต ซ็อกเก็ต คือ การติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ต้นทางและปลายทาง ซึ่งเป็นการติดต่อสื่อสารอยู่บนเครือข่ายที่ใช้ อินเตอร์เน็ต โปรโตคอล (Internet Protocol) คุณลักษณะของเน็ตเวิร์ค ซ็อกเก็ต ที่แตกต่างจากซ็อกเก็ตอื่น คือ โปรโตคอลที่สนับสนุน ซึ่งสนับสนุนโปรโตคอลต่อไปนี้

- ทีซีพี (TCP)
- ยูดีพี (UDP)
- รอวไอพี (Raw IP) [11]

2.5.1.1 ทีซีพี (TCP)

ทีซีพี ย่อมาจาก ทรานสมิชชันคอมโทรลโปรโตคอล (Transmission Control Protocol) เป็นหนึ่งในโปรโตคอลหลักของอินเตอร์เน็ตโปรโตคอล ซึ่งเป็นโปรโตคอลที่มีความน่าเชื่อถือใน

การส่งข้อมูลระหว่างผู้ส่งและผู้รับ โดยในการรับส่งข้อมูลแต่ละครั้งจะต้องทำการเชื่อมต่อระหว่างผู้ส่งและผู้รับก่อนเสมอ ทีซีพีถูกใช้กับซ็อกเก็ตในเรื่องของสตรีม ซ็อกเก็ต (Stream Socket) [11, 12]

2.5.1.2 ยูดีพี (UDP)

ยูดีพี ย่อมาจาก ยูซเซอร์ ดาต้าแกรม โปรโตคอล (User Data Protocol) เป็นหนึ่งในโปรโตคอลหลักของอินเตอร์เน็ตเช่นเดียวกับทีซีพี แต่แตกต่างที่ยูดีพีมีความน่าเชื่อถือในการรับส่งข้อมูลน้อยกว่าทีซีพี อันเนื่องจากในการรับส่งข้อมูลระหว่างผู้รับกับผู้ส่งไม่มีการสร้างการเชื่อมต่อก่อนการรับส่งข้อมูล ยูดีพีถูกใช้กับซ็อกเก็ตในเรื่องของ ดาต้าแกรม ซ็อกเก็ต (Datagram Sockets) [12]

2.5.1.3 รอวไอพี (Raw IP)

รอวไอพี ไม่ได้ถูกจัดอยู่ในรูปแบบของโปรโตคอลเหมือนกับทีซีพี และยูดีพี แต่ซ็อกเก็ตมีความสัมพันธ์กับรอว ไอพี ในเรื่องของรอว ซ็อกเก็ต (Raw Socket) โดยที่การรับส่งข้อมูลของทีซีพี และยูดีพี ซึ่งรอวซ็อกเก็ตจะทำการรับข้อมูลที่เป็นเฮดเดอร์ของข้อมูล การทำงานดังกล่าวถูกนำไปประยุกต์ใช้กับระบบปฏิบัติทั่วไป [11]

2.5.2 ซ็อกเก็ต ใน จาวา (Socket in Java)

จาวามีคลาสไลบรารีที่ถูกเตรียมไว้เพื่อตอบสนองต่อการทำงานของซ็อกเก็ตที่ได้อธิบายไว้ข้างต้น ซึ่งแพคเกจ และคลาสไลบรารีที่เกี่ยวข้องในการเข้าถึงซ็อกเก็ต และการทำงานต่างๆ มีดังนี้

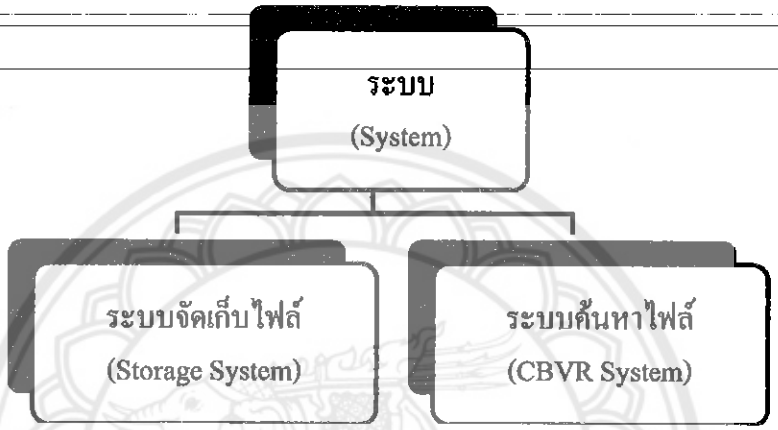
- แพคเกจ java.net
- คลาส ScrverSocket
- คลาส Socket
- คลาส MulticastSocket

บทที่ 3

ออกแบบระบบ

3.1 ระบบ (System)

ระบบถูกแบ่งออกเป็นระบบย่อยดังรูปที่ 3.1



รูปที่ 3.1 แสดงแผนผังของระบบ

จากรูปจะเห็นว่าระบบถูกแบ่งออกเป็น 2 ระบบย่อย คือ ระบบจัดเก็บไฟล์ (Storage System) และระบบค้นหาไฟล์ (CBVR System) โดยระบบจัดเก็บไฟล์จะทำหน้าที่จัดการเกี่ยวกับไฟล์ที่ต้องการแชร์ (sharing) เพื่อให้ผู้อื่นสามารถค้นหาไฟล์นั้นได้ ซึ่งไฟล์ที่จะทำการจัดเก็บนั้นต้องถูกแบ่งออกเป็นส่วนย่อยของวีดิโอ โดยการทำเซกเมนต์เทชัน (Segmentation) หลังจากนั้นจะถูกนำไปทำดัชนีเพื่อใช้ในระบบค้นหาไฟล์ ส่วนระบบค้นหาไฟล์ทำหน้าที่เกี่ยวกับการค้นหาไฟล์ที่ถูกจัดเก็บด้วยระบบจัดเก็บไฟล์ ซึ่งมีการทำดัชนีไว้แล้ว โดยใช้วีดิโอตัวอย่างในการค้นหา ซึ่งมีการปรับปรุงคุณภาพการค้นหาไฟล์โดยใช้โมเดลการปรับตัวอัตโนมัติ ทั้งสองระบบย่อยดังกล่าวนี้จะได้อธิบายอย่างละเอียดในหัวข้อถัดไป

3.2 การจัดเก็บข้อมูล

3.2.1 การทำส่วนย่อยของวีดิโอ (Video Segmentation)

ในส่วนนี้จะทำการใช้โปรแกรมสำเร็จรูป Video Editor 7.0 เข้ามาทำการแยกส่วนย่อยของไฟล์วีดิโอ โดยจะไม่ทำการพัฒนาโปรแกรมในส่วนนี้ เพราะจะไม่ได้ผลลัพธ์ที่มีประสิทธิภาพนัก ซึ่งจะต้องนำส่วนย่อยของวีดิโอที่ได้มาไปทำดัชนีในขั้นตอนต่อไป

3.2.2 การทำดัชนี (Indexing)

ขั้นตอนนี้จะนำวีดิโอที่ทำการแบ่งส่วนย่อยมาแล้วมาทำการทำดัชนีให้กับส่วนย่อยแต่ละอันโดยใช้หลักการทำดัชนีแบบ AVI ซึ่งในส่วนนี้จะนำโปรแกรม "อัลกอริทึม AVI สำหรับการทำดัชนีวีดิโอ" ซึ่งเป็นปริญญานิพนธ์ที่มีผู้ทำมาแล้ว โดยหลักๆ จะมีขั้นตอนดังต่อไปนี้

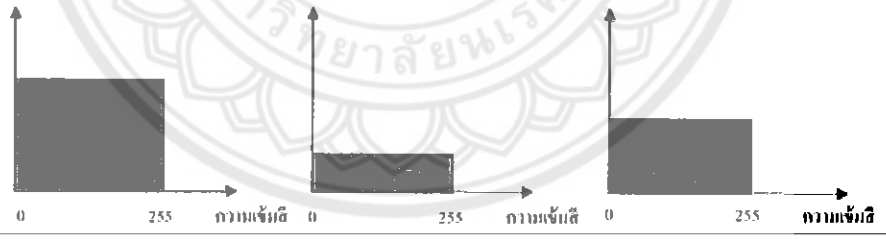
3.2.2.1 ดำเนินการแยกช็อตวีดิโอต้นแบบเป็นเฟรมรูปภาพ (Template Generation)

เนื่องจากไฟล์วีดิโอเป็นภาพเคลื่อนไหวไม่สามารถนำมาหาข้อมูลฮีสโตแกรมได้เลย จึงต้องนำเฟรมของวีดิโอที่เป็นภาพนิ่งมาใช้หาข้อมูลฮีสโตแกรม แล้วนำเฟรมที่มีความเหมือนหรือใกล้เคียงกันมาจัดกลุ่มและทำการหาตัวแทนของกลุ่มนั้นๆ ออกมา เพื่อนำไปใช้ในการเปรียบเทียบต่อไป

3.2.2.2 ทำการถอดข้อมูลภาพที่ได้ให้อยู่ในรูปข้อมูลฮีสโตแกรม

การหาค่า RGB Color Histogram จะทำการวิเคราะห์ค่าสีแต่ละ Pixel จะทำให้ได้ค่าสีที่ออกมาเป็นฮีสโตแกรม (การสะสมจำนวนของค่าเม็ดสี) ซึ่งจะมีอยู่ 3 ค่าคือ

Red หมายถึง ค่าสีแดง ที่เป็นองค์ประกอบของเฟรมนั้น
Green หมายถึง ค่าสีเขียว ที่เป็นองค์ประกอบของเฟรมนั้น
Blue หมายถึง ค่าสีน้ำเงิน ที่เป็นองค์ประกอบของเฟรมนั้น



รูปที่ 3.2 รูปแสดงค่าสีต่างๆ

จากนั้น ทำการเก็บข้อมูลฮีสโตแกรมของแต่ละเฟรม โดยระบุค่าสีต่างๆ ว่ามีค่าเม็ดสีเท่าไรที่ได้จากการวิเคราะห์ Pixel ซึ่งเริ่มตั้งแต่ 0 ถึง 255

3.2.2.3 นำข้อมูลฮีสโตแกรมเข้าสู่กระบวนการ Vector Quantization

โดยกระบวนการนี้จะเป็นการกำหนดค่าเวกเตอร์ตามฮีสโตแกรมที่ได้มาในแต่ละเฟรมของวีดิโอเพื่อนำไปเก็บลงในฐานข้อมูล ซึ่งเป็นกระบวนการทำซ้ำ ซึ่งได้เลือก 2 หลักการข้างต้นมาแก้ปัญหา กระบวนการนี้ต้องการค่า codebook เริ่มต้น $C^{(0)}$ ค่า codebook เริ่มต้นนี้ ได้มาจากกระบวนการ Splitting ในกระบวนการนี้ ค่า codevector เริ่มต้น จะเซตให้เป็นค่าเฉลี่ยของ training

sequence ทั้งหมด codevector นี้จะถูกแตกออกเป็น 2 เวกเตอร์ ในกระบวนการทำซ้ำ จะดำเนินการด้วยเวกเตอร์ 2 ตัวนี้ โดยกำหนดให้เป็นค่า codebook เริ่มต้น codevector 2 ตัวสุดท้าย จะถูกแตกออกเป็น 4 เวกเตอร์ และดำเนินการซ้ำไปเรื่อยๆ จนได้ค่าของ codevector ที่เหมาะสม กระบวนการดังกล่าวเป็นดังนี้

LBG Design Algorithm

1. รับค่า τ และกำหนดให้ $\varepsilon > 0$ ควรจะเป็นจำนวนที่มีค่าน้อยๆ ให้ N=1 และ

$$c_1^* = \frac{1}{M}\sum_{m=1}^{M} x_m \tag{3.1}$$

2. คำนวณหาค่า

$$D_{ave}^* = \frac{1}{Mk}\sum_{m=1}^{M} \| x_m - c_1^* \|^2 \tag{3.2}$$

3. Splitting : i = 1, 2, ..., N ให้

$$c_i^{(0)} = (1+\varepsilon)c_i^*,$$
$$c_{N+1}^{(0)} = (1+\varepsilon)c_i^* \tag{3.3}$$

กำหนด N = 2N

4. Iteration : ให้ $D_{ave}^{(0)} = D_{ave}^*$ กำหนด Iteration index i = 0

i. สำหรับ m = 1, 2, ..., M หาค่าที่น้อยที่สุดของ

$$\| x_m - c_n^{(i)} \|^2 \tag{3.4}$$

ทำทั้งหมด จาก n = 1, 2, ..., N ให้ n* เป็นค่า index ที่รับค่าน้อยที่สุด กำหนดให้

$$Q(x_m) = c_{n^*}^{(i)} \tag{3.5}$$

ii. สำหรับ n = 1, 2, …, N ปรับค่า codevedtor

$$c_n^{(i+1)} = \frac{\sum_{Q(x_m)=c_n^{(i)}} x_m}{\sum_{Q(x_m)=c_n^{(i)}} 1} \quad (3.6)$$

iii. กำหนดให้ i = i+1

iv. หาค่าการกระจาย

$$D_{ave}^{(i)} = \frac{1}{Mk}\sum_{m=1}^{M} \| x_m - Q(x_m) \|^2 \quad (3.7)$$

v. ถ้า $\frac{D_{ave}^{(i-1)} - D_{ave}^{(i)}}{D_{ave}^{(i-1)}} > \varepsilon$ กลับไปทำขั้นตอนที่ (i)

vi. กำหนด $D_{ave}^{*} = D_{ave}^{(i)}$ สำหรับค่า n = 1, 2, …, N กำหนดให้

$$c_n^{*} = c_n^{(i)} \quad (3.8)$$

เป็นค่า codevector ตัวสุดท้าย

5. ทำซ้ำในขั้นตอนที่ 3 และ 4 จนกระทั่งได้ตัวเลขของ codevector ที่เหมาะสม

3.2.2.4 นำค่าเวกเตอร์ที่ได้มาทำการหาดัชนีและเก็บลงในฐานข้อมูล

จากขั้นตอน Vector Quantization จะทำให้ได้จำนวนของ codevector ทั้งหมดที่ได้จากการกำหนดค่าของผู้ใช้ ซึ่งจะมีค่าเป็น 2n (2, 4, 8, 16, …, 2048) จากนั้นโปรแกรมจะทำการ split ค่า codevector ให้ได้ตามที่กำหนดหากจำนวนเฟรมมีมาก ค่า codevector ก็จะมีจำนวนมากตามไปด้วย ทำการเก็บ codevector ลงเมตริกซ์ ซึ่งจะมีขนาด n x m โดยที่ n คือ จำนวน codevector และ m คือ ค่าสี RGB ของ Histogram ซึ่งมีจำนวน 256 x 3 เท่ากับ 768 ค่า

จากนั้นนำค่า codevector ที่ได้ทั้งหมด มาทำการ Indexing โดยการนำค่า histogram ของช็อตวีดิโอแต่ละช็อตมาเปรียบเทียบกับ codevector ทุกตัวว่า histogram นั้นๆ อยู่ในกลุ่มของ codevector ใดทำจนครบทุก histogram จากนั้นทำการเก็บค่าสถิติของจำนวนสมาชิกใน codevector ของช็อตวีดิโอนั้นแล้วทำการจัดเก็บลงฐานข้อมูล

3.3 ระบบการค้นหาโดยใช้การเปรียบเทียบวีดิโอกับวีดิโอที่มีระบบปรับตัวอัตโนมัติ

3.3.1 ระบบการเปรียบเทียบวีดิโอกับวีดิโอ[1]

ในส่วนนี้จะนำส่วนของค่าดัชนีและช็อตวีดิโอแบบ AVI มาใช้ในการค้นหา เพื่อหาความคล้ายคลึงของช็อตวีดิโอที่ใช้ค้นหากับช็อตวีดิโอที่อยู่ในฐานข้อมูล จะต้องทำการนำข้อมูลของช็อตวีดิโอที่ใช้ค้นหามาเปรียบเทียบกับข้อมูลช็อตวีดิโอที่อยู่ในฐานข้อมูลแต่ละเครื่อง โดยการคำนวณหาค่าระยะห่าง (distance) ระหว่างช็อตวีดิโอต้นแบบกับช็อตวีดิโอในฐานข้อมูล ซึ่งอัลกอริทึมในการเปรียบเทียบ คือ Cosine Similarity เป็นการหาระยะห่างระหว่างจุด

$Q = (w_{1j}, w_{2j}, \ldots, w_{nj})$ และ $D_i = (w_{1j}, w_{2j}, \ldots, w_{nj})$

$$sim(Q, D_i) = \frac{\sum_i w_{Q,j} w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2} \sqrt{\sum_i w_{i,j}^2}} \qquad (3.9)$$

โดยที่ $sim(Q, D_i)$ = ระยะห่างระหว่างช็อตสองช็อต

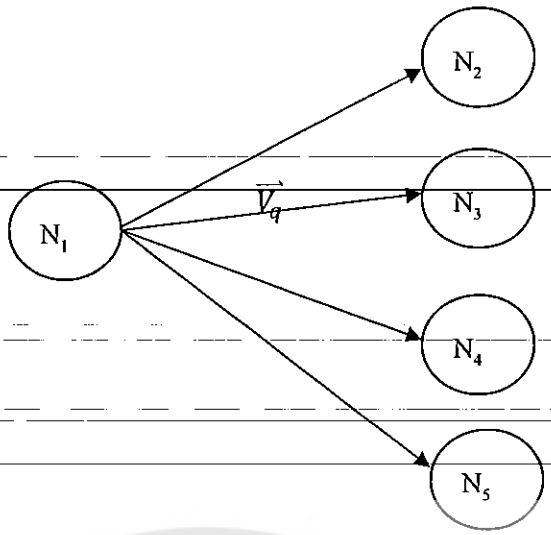
w_{Qj} = ค่าของช็อตที่ใช้ค้นหา (Q) เมื่อทำการเปรียบเทียบกับต้นแบบที่ j

w_{ij} = ค่าของช็อตที่อยู่ภายในฐานข้อมูลที่ i เมื่อทำการเปรียบเทียบกับต้นแบบที่ j

เมื่อได้ค่า $sim(d_j, d_k)$ มาจะสามารถตัดสินว่าช็อตที่ใช้ค้นหากับช็อตที่ทำการเปรียบเทียบนี้มีความคล้ายกันมาเท่าไร เช่น ถ้าค่า $sim(d_j, d_k) = 1$ จะแสดงว่าเป็นช็อตที่ใช้ค้นหาเหมือนกับช็อตที่เปรียบเทียบอยู่ แต่ถ้าค่า $sim(d_j, d_k)$ มีค่าใกล้เคียงศูนย์ จะแสดงว่าช็อตที่ใช้ค้นหาไม่คล้ายกับช็อตที่เปรียบเทียบอยู่

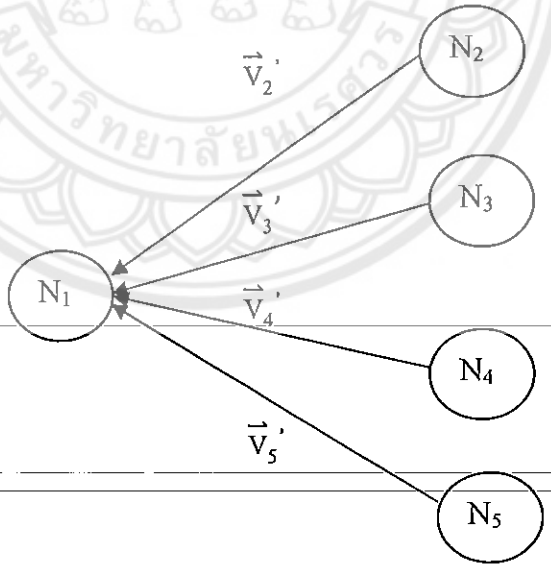
3.3.2 ระบบการปรับตัวแบบอัตโนมัติ

ระบบการปรับตัวแบบอัตโนมัติมีวิธีการทำงาน โดยจะทำการส่งเวกเตอร์ V_q จากโนดแรก (N_1) ไปยังโหนดที่อยู่ติดกันดังรูป



รูปที่ 3.3 แสดงการส่งเวกเตอร์ V_q ไปยังโนดที่ติดกับโนดที่ทำการค้นหา

เมื่อ N_2, N_3, N_4 และ N_5 ได้รับเวกเตอร์ V_q จาก N_1 แล้วจะทำการเปรียบเทียบเวกเตอร์ที่ได้รับมากับวีดิโอที่ถูกเก็บไว้ในฐานข้อมูล และทำการปรับปรุงคุณภาพค่าเวกเตอร์โดยอัตโนมัติจะได้ค่าเวกเตอร์ใหม่ขึ้นมาโดยเรียกว่าเวกเตอร์ V_q' ซึ่งเป็นเวกเตอร์สำหรับค้นหาวีดิโอที่ถูกปรับปรุงคุณภาพแล้ว N_2, N_3, N_4 และ N_5 จะทำการส่งเวกเตอร์ V_q' กลับ N_1 ดังรูป



รูปที่ 3.4 แสดงการส่งเวกเตอร์ที่ถูกปรับปรุงแล้ว ($\vec{V}_q'$) กลับไปยังโนดที่ทำการค้นหา

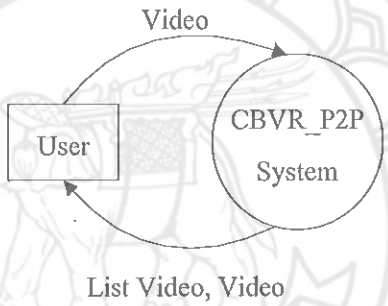
เมื่อ N_1 ได้รับค่าเวกเตอร์ที่ถูกปรับปรุงคุณภาพจาก N_2, N_3, N_4 และ N_5 จะทำการเปรียบเทียบกับค่าเวกเตอร์เดิมที่ส่งไป ถ้าเวกเตอร์ที่ถูกปรับปรุงมีคุณภาพดีกว่าจะนำเวกเตอร์ที่ถูกปรับปรุงเก็บไว้ และส่งไปยัง N_2 , N_3 , N_4 และ N_5 อีกครั้งเพื่อทำการค้นหา และปรับปรุงคุณภาพ แต่

ถ้าเวกเตอร์ที่ถูกปรับปรุงแล้วมีคุณภาพไม่ดีกว่าเวกเตอร์เดิมจะนำเวกเตอร์เดิมไปทำการค้นหาในรอบต่อไป โดยจะทำการปรับปรุงคุณภาพของเวกเตอร์ 3 รอบและทำการแสดงผลของวีดิโอที่มีค่าใกล้เคียงกับวีดิโอที่ใช้ค้นหาโดยเรียงจากค่าที่ใกล้เคียงมากไปน้อย

3.4 วิเคราะห์การทำงานของโปรแกรม

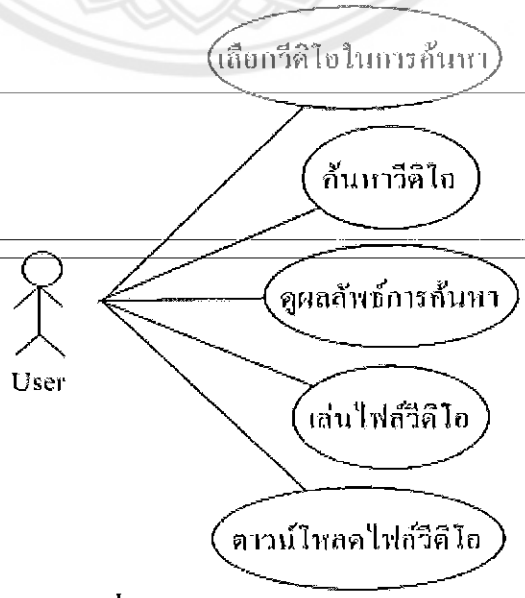
ในขั้นตอนนี้จะกล่าวถึงการวิเคราะห์การทำงานของโปรแกรมว่าทำงานอย่างไร ส่วนประกอบของโปรแกรมมีอะไรบ้างและแต่ละส่วนประกอบทำหน้าที่อะไรบ้างมีความสำคัญอย่างไร โดยที่จะแสดงเป็นแผนภาพไดอะแกรมเพื่อให้ง่ายต่อการทำความเข้าใจ

3.4.1 Context Diagram



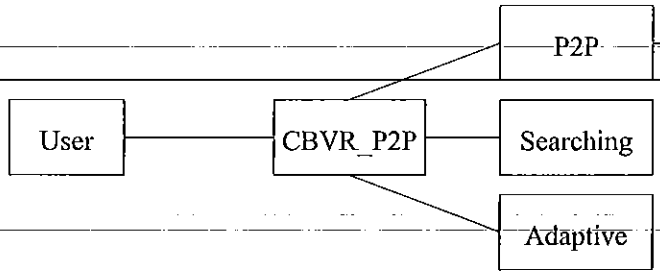
รูปที่ 3.5 แสดงแผนภาพ Context

3.4.2 Use case Diagram



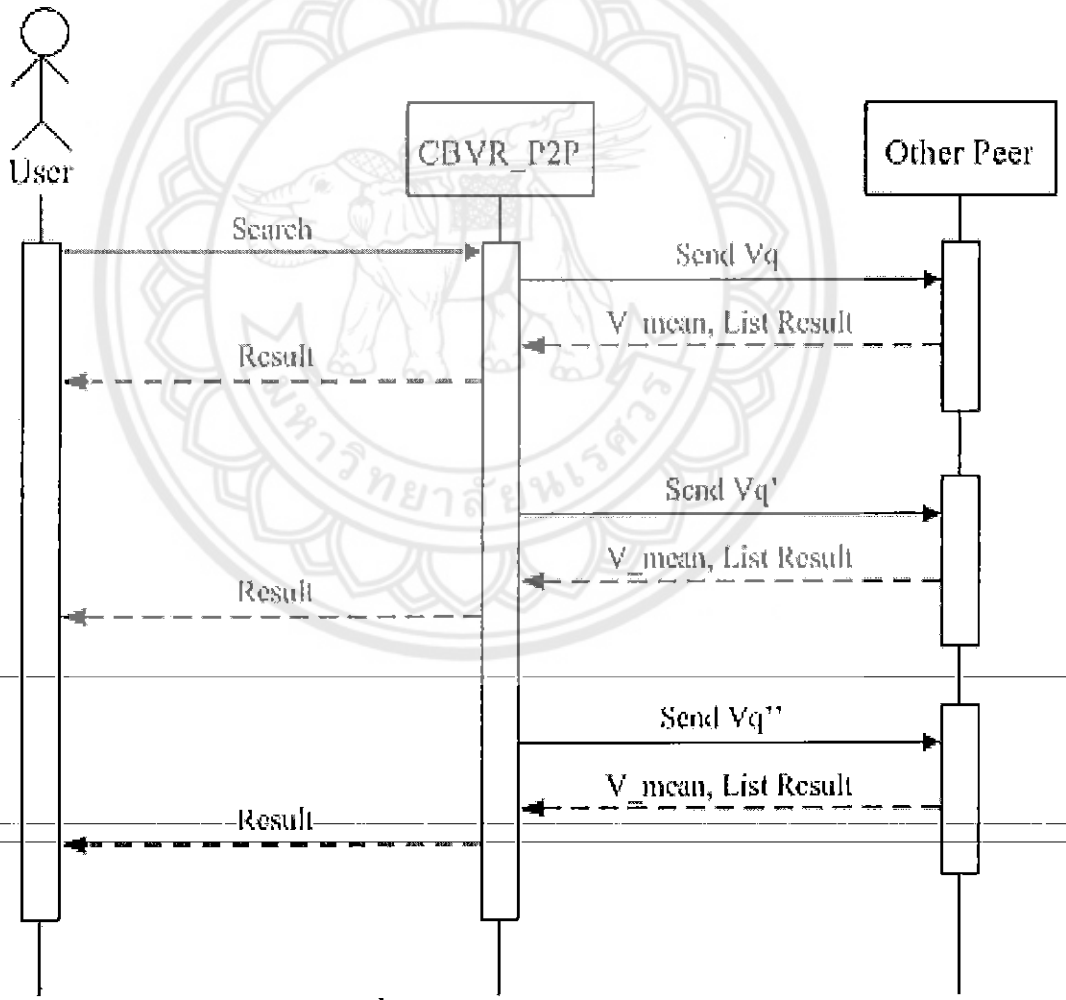
รูปที่ 3.6 แสดงแผนภาพ Use Case

3.4.3 ER Diagram



รูปที่ 3.7 แสดงแผนภาพ ER

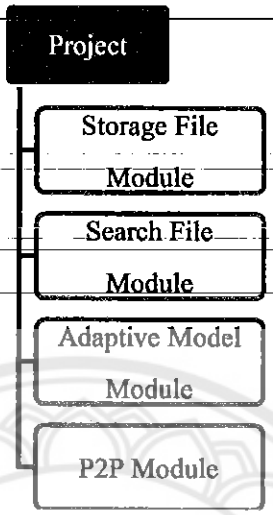
3.4.4 Sequence Diagram



รูปที่ 3.8 แสดงแผนภาพ Sequence

3.5 ออกแบบโปรแกรม (Design Program)

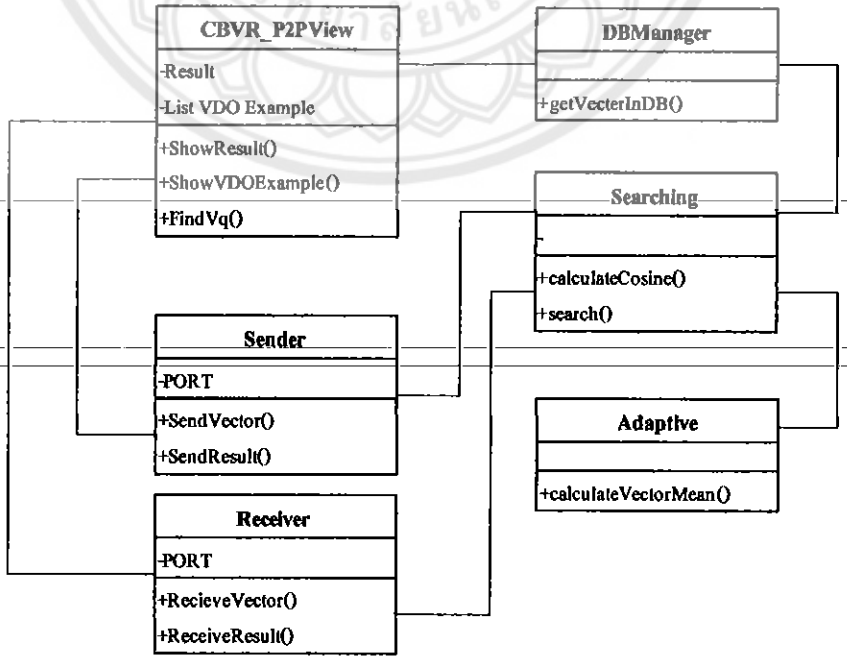
โปรแกรมถูกแบ่งออกเป็นโมดูล (Module) ย่อย เพื่อสะดวกต่อการพัฒนาดังรูปที่ 3.2



รูปที่ 3.9 แสดงโมดูลย่อยของโปรแกรม

จากรูปแสดงให้เห็นถึงโมดูลย่อยของโปรแกรมที่ถูกแบ่งออกเป็น 4 โมดูล ได้แก่ โมดูลการจัดเก็บไฟล์ (Storage File Module) โมดูลการค้นหาไฟล์ (Search File Module) โมดูลโมเดลการปรับตัว (Adaptive Model Module) และ โมดูลเพียร์ทูเพียร์ (P2P Module)

3.5.1 Class Diagram



รูปที่ 3.10 แสดงแผนภาพ Class

3.5.2 ต้นแบบรายละเอียดการทำงาน (Operation Specification Template)

ตารางที่ 3.1 ตารางต้นแบบรายละเอียดการเก็บไฟล์วีดิโอลงฐานข้อมูล

Scenario Number	1	**User Objective**	เก็บไฟล์วีดิโอลงฐานข้อมูล
Scenario Objective		เพื่อแสดงให้เห็นถึงการเก็บไฟล์วีดิโอลงฐานข้อมูล	
Source	**Step**	**Action**	**Comments**
ผู้ใช้	1	เรียกใช้ระบบการจัดเก็บไฟล์	
ระบบ	2	แสดงไฟล์วีดิโอทั้งหมดที่อยู่ในเครื่องผู้ใช้ที่สามารถเพิ่มเข้าไปในฐานข้อมูลได้	
ผู้ใช้	3	เลือกไฟล์ที่ต้องการเพิ่ม	ตรวจสอบว่าไฟล์ที่เพิ่มอยู่ในฐานข้อมูลแล้วหรือไม่
ระบบ	4	ทำการเพิ่มชื่อไฟล์ลงในฐานข้อมูล	
ระบบ	5	แบ่งวีดิโอออกเป็นส่วนย่อย	
ระบบ	6	จัดทำดัชนีให้กับไฟล์วีดิโอ	

ตารางที่ 3.2 ตารางต้นแบบรายละเอียดการลบไฟล์วีดิโอออกจากฐานข้อมูล

Scenario Number	2	**User Objective**	ลบไฟล์วีดิโอออกฐานข้อมูล
Scenario Objective		เพื่อแสดงให้เห็นถึงการลบไฟล์วีดิโอออกจากฐานข้อมูล	
Source	**Step**	**Action**	**Comments**
ผู้ใช้	1	เรียกใช้ระบบการลบไฟล์	
ระบบ	2	แสดงไฟล์วีดิโอทั้งหมดที่อยู่ในฐานข้อมูล	
ผู้ใช้	3	เลือกไฟล์ที่ต้องการลบ	
ระบบ	4	ทำการลบชื่อไฟล์ออกจากฐานข้อมูล	

ตารางที่ 3.3 ตารางต้นแบบรายละเอียดการค้นหาไฟล์วีดิโอภายในเครื่อง

Scenario Number	3	**User Objective**	ค้นหาไฟล์วีดิโอภายในเครื่อง
Scenario Objective		เพื่อแสดงให้เห็นการค้นหาไฟล์วีดิโอภายในเครื่อง	

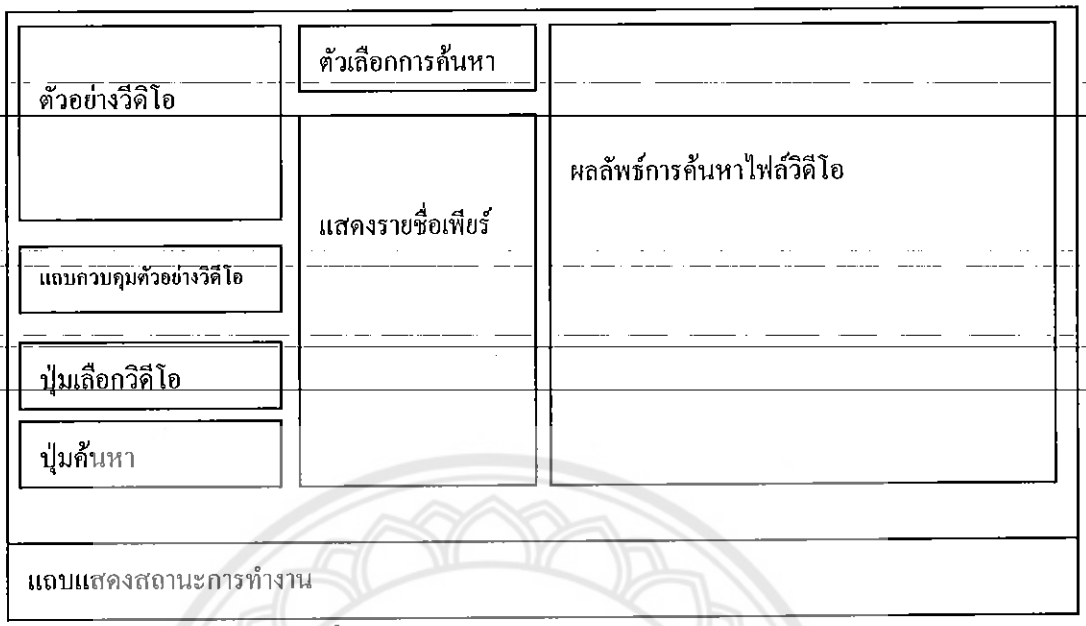
ตารางที่ 3.3 (ต่อ) ตารางต้นแบบรายละเอียดการค้นหาไฟล์วีดิโอภายในเครื่อง

Source	Step	Action	Comments
ผู้ใช้	1	เรียกใช้ระบบการค้นหาไฟล์	
ระบบ	2	รอให้ผู้ใช้ใส่วีดิโอตัวอย่างในการค้นหา	
ผู้ใช้	3	เลือกไฟล์วีดิโอในการค้นหา	
ระบบ	4	ทำการค้นหาไฟล์วีดิโอ	
ระบบ	5	ทำการปรับปรุงผลลัพธ์การค้นหาไฟล์วีดิโอ	
ระบบ	6	แสดงผลลัพธ์ที่ดีที่สุดในการค้นหาไฟล์วีดิโอ	

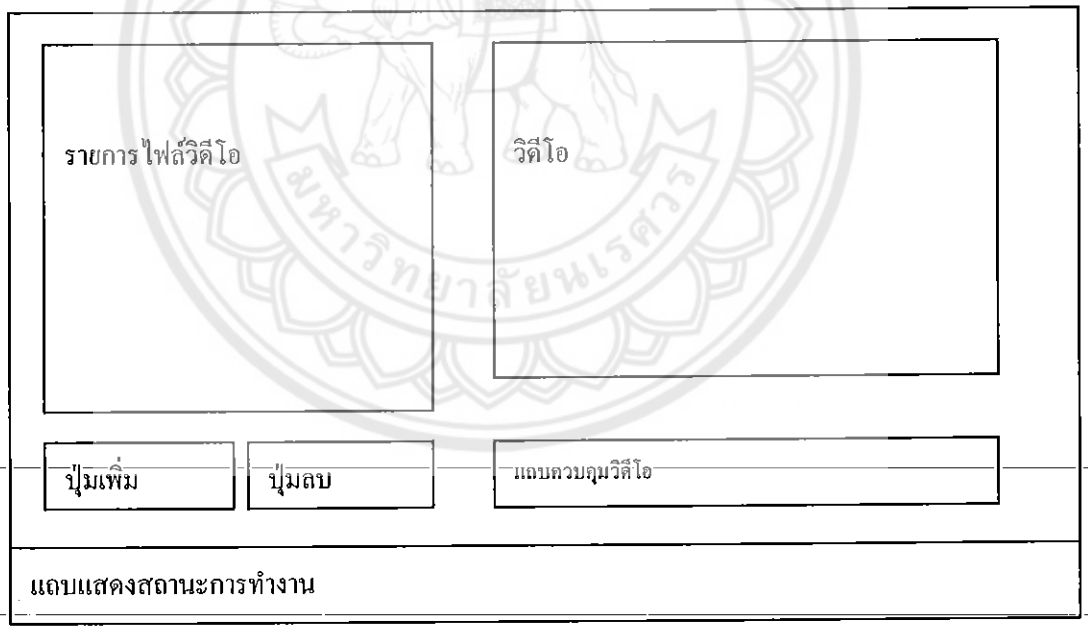
ตารางที่ 3.4 ตารางต้นแบบรายละเอียดการค้นหาไฟล์วีดิโอบนเครือข่ายเพียร์ทูเพียร์

Scenario Number	4	**User Objective**	ค้นหาไฟล์วีดิโอบนเพียร์ทูเพียร์เน็ตเวิร์ก
Scenario Objective	เพื่อแสดงให้เห็นการค้นหาไฟล์วีดิโอบนเพียร์ทูเพียร์เน็ตเวิร์ก		
Source	**Step**	**Action**	**Comments**
ผู้ใช้	1	เรียกใช้ระบบการค้นหาไฟล์	
ระบบ	2	รอให้ผู้ใช้เลือกวีดิโอตัวอย่าง	
ผู้ใช้	3	เลือกไฟล์วีดิโอในการค้นหา	
ระบบ	4	ทำการส่งคิวรี่ไปยังเพียร์อื่นที่อยู่ในเพียร์ทูเพียร์เน็ตเวิร์ก	
เพียร์อื่นๆ	5	ทำการค้นหาไฟล์วีดิโอแล้วส่งผลลัพธ์กลับมายังระบบ	
ระบบ	6	ส่งคิวรี่ใหม่เพื่อทำการปรับปรุงคุณภาพของผลลัพธ์	
เพียร์อื่นๆ	7	ทำการค้นหาไฟล์ของคิวรี่ใหม่เพื่อปรับปรุงผลลัพธ์ และส่งผลลัพธ์ใหม่กลับไปให้ระบบ	
ระบบ	8	แสดงผลลัพธ์ที่ดีที่สุดในการค้นหาไฟล์วีดิโอ	

3.5.3 รูปแบบโปรแกรม (Interface)



รูปที่ 3.11 รูปแบบหน้าต่างการค้นหาไฟล์วีดิโอ



รูปที่ 3.12 รูปแบบหน้าต่างการจัดเก็บไฟล์วีดิโอ

บทที่ 4

ผลการทดลอง

การทดลองแบ่งออกเป็น 2 ประเภทดังตารางที่ 4.1

ตารางที่ 4.1 ตารางแสดงแผนการทดลอง

ลำดับ	การทดลอง
1	**การทดลองการทำงานของโปรแกรมตั้งแต่เริ่มต้นจนจบการทำงาน** • การทดสอบการอ่านข้อมูลวีดิโอตัวอย่างจากฐานข้อมูล • การทดสอบการติดต่อระหว่างเพียร์บนเครือข่ายเพียร์ทูเพียร์ • การทดสอบการค้นหา • การทดสอบการเล่นไฟล์วีดีโอ • การทดสอบการดาวน์โหลดไฟล์วีดิโอ
2	**การทดลองหาประสิทธิภาพการค้นหา** • การเปรียบเทียบประสิทธิภาพการค้นหาโดยมีข้อมูลรวบรวมไว้อยู่ในฐานข้อมูลเดียว • การเปรียบเทียบประสิทธิภาพการค้นหาโดยมีฐานข้อมูลแบบกระจาย • การเปรียบเทียบประสิทธิภาพระหว่างการปรับตัวอัตโนมัติภายในเพียร์กับระบบเพียร์ทูเพียร์ • การเลือกค่าที่เหมาะสม - การทดลองหาค่า Upper Threshold - การทดลองหาค่า Lower Threshold - การทดลองหาค่า Beta - การทดลองหาค่า Gamma • การวัดประสิทธิภาพในการค้นหา

4.1 สิ่งที่ต้องเตรียมก่อนการทดลอง

1. โปรแกรม CBVR ที่พัฒนาโดยใช้ภาษาจาวา
2. ไฟล์ช็อตวีดิโอที่ได้จากการแยกช็อตด้วยโปรแกรม Video Editor 7.0 ซึ่งเป็นไฟล์ mpg
3. รูปภาพคีย์เฟรมของไฟล์วีดิโอทั้งหมดที่ได้จากการแยกช็อต

4. Java Media Framework 2.1.1e (JMF) เพื่อแสดงวีดิโอภายหลังการค้นหา
5. ฐานข้อมูลที่ได้จากการทำดัชนีโดยใช้หลักการทำดัชนีวีดิโอแบบปรับตัวอัตโนมัติ

4.2 การทดลองการทำงานของโปรแกรมตั้งแต่เริ่มต้นจนจบการทำงาน

4.2.1 การทดสอบการอ่านข้อมูลวีดิโอตัวอย่างจากฐานข้อมูล

การทดสอบการการอ่านข้อมูลวีดิโอตัวอย่างจากฐานข้อมูล เพื่อนำวีดิโอตัวอย่างมาแสดงผลและนำเวกเตอร์มาใช้ในการค้นหา ซึ่งได้ผลการทดสอบการทำงานดังตารางที่ 4.2

ตารางที่ 4.2 ตารางแสดงการทดสอบการอ่านข้อมูลวีดิโอตัวอย่างจากฐานข้อมูล

ลำดับ	อินพุต	ระบบ	ผลที่คาดว่าจะได้รับ	ผลที่เกิดขึ้นจริง
1	เปิดระบบ	ปกติ	โปรแกรมแสดงรายการคีย์เฟรมของวีดิโอ	โปรแกรมแสดงรายการคีย์เฟรมของวีดิโอ
2	เปิดระบบ	ปกติ (ไม่มีฐานข้อมูล)	โปรแกรมแสดงรายการคีย์เฟรมของวีดิโอว่างเปล่า	โปรแกรมแสดงรายการคีย์เฟรมของวีดิโอว่างเปล่า
3	เปิดระบบ	ปกติ (ฐานข้อมูลมีรูปแบบไม่ถูกต้องตามกำหนด)	โปรแกรมไม่สามารถทำงานต่อได้	โปรแกรมไม่สามารถทำงานต่อได้

4.2.2 การทดสอบการติดต่อระหว่างเพียร์บนเครือข่ายเพียร์ทูเพียร์

การทดสอบการติดต่อระหว่างเพียร์บนเครือข่ายเพียร์ทูเพียร์ เพื่อทดสอบการเชื่อมต่อ และรับส่งข้อมูลระหว่างเพียร์

ตารางที่ 4.3 ตารางแสดงการทดสอบการติดต่อระหว่างเพียร์บนเรือข่ายเพียร์ทูเพียร์

ลำดับ	อินพุต	ระบบ	ผลที่คาดว่าจะได้รับ	ผลที่เกิดขึ้นจริง
1	เปิดระบบ	ปกติ	โปรแกรมแสดงชื่อเพียร์ที่เปิดอยู่บนเครือข่ายเพียร์ทูเพียร์ทั้งหมด	โปรแกรมแสดงชื่อเพียร์ที่เปิดอยู่บนเครือข่ายเพียร์ทูเพียร์ทั้งหมด

ตารางที่ 4.3 (ต่อ) ตารางแสดงการทดสอบการติดต่อระหว่างเพียร์บนเรือข่ายเพียร์ทูเพียร์

ลำดับ	อินพุต	ระบบ	ผลที่คาดว่าจะได้รับ	ผลที่เกิดขึ้นจริง
2	การส่ง	ปกติ	ข้อความถูกส่งไปยังเพียร์	ข้อความถูกส่งไปยังเพียร์
	ข้อความ		ปลายทาง	ปลายทาง
3	รับ ข้อความ จาก	ปกติ	โปรแกรมสามารถรับข้อความที่ถูกต้องจากเพียร์อื่น	โปรแกรมรับข้อความที่ถูกต้องจากเพียร์อื่น
	เพียร์อื่น			

4.2.3 การทดสอบการค้นหา

การทดสอบการค้นหา เพื่อตรวจสอบการทำงานของระบบการค้นหา ซึ่งได้ผลการทดสอบดังตารางที่ 4.4

ตารางที่ 4.4 ตารางแสดงการทดสอบการค้นหา

ลำดับ	อินพุต	ระบบ	ผลที่คาดว่าจะได้รับ	ผลที่เกิดขึ้นจริง
1	วีดิโอตัวอย่างที่ถูกเลือก	ปกติ	รายการคีย์เฟรมผลลัพธ์ของวีดิโอที่ถูกค้นหา	รายการคีย์เฟรมผลลัพธ์ของวีดิโอที่ถูกค้นหา
2	ไม่มีการเลือกวีดิโอตัวอย่าง	ปกติ	โปรแกรมไม่ทำการค้นหา	โปรแกรมไม่ทำการค้นหา
3	วีดิโอตัวอย่างที่ถูกเลือก	ไม่ปกติ (เพียร์อื่นปิดการทำงานของระบบ)	โปรแกรมหยุดการค้นหา	โปรแกรมหยุดการค้นหา

4.2.4 การทดสอบการเล่นไฟล์วีดีโอ

การทดสอบการเล่นไฟล์วีดิโอ เพื่อตรวจสอบการทำงานของการเล่นไฟล์วีดิโอจากผลการค้นหาภายในเพียร์และบนเครือข่ายเพียร์ทูเพียร์

ตารางที่ 4.5 ตารางแสดงการทดสอบการเล่นไฟล์วีดิโอ

ลำดับ	อินพุต	ระบบ	ผลที่คาดว่าจะได้รับ	ผลที่เกิดขึ้นจริง
1	ผู้ใช้กดคีย์เฟรมของวีดิโอที่ต้องการ	ปกติ	โปรแกรมเล่นไฟล์วีดิโอที่ถูกเลือก	โปรแกรมเล่นไฟล์วีดิโอที่ถูกเลือก
2	ผู้ใช้กดคีย์เฟรมของวีดิโอที่ต้องการ	ไม่ปกติ (เพียร์ที่มีวีดิโอที่ถูกเลือกทำการปิดระบบ)	โปรแกรมไม่เล่นไฟล์วีดิโอ และแจ้งเตือนให้ผู้ใช้ทราบ	โปรแกรมไม่เล่นไฟล์วีดิโอ และแจ้งเตือนให้ผู้ใช้ทราบ
3	ผู้ใช้กดคีย์เฟรมของวีดิโอที่ต้องการ	ปกติ (ระบบไม่ถูกติดตั้ง JMF)	โปรแกรมไม่เล่นไฟล์วีดิโอ	โปรแกรมไม่เล่นไฟล์วีดิโอ

4.2.5 การทดสอบการดาวน์โหลดไฟล์วีดิโอ

การทดสอบการดาวน์โหลดไฟล์วีดิโอ เพื่อตรวจสอบการทำงานของการดาวน์โหลดไฟล์วีดิโอจากเพียร์อื่นบนเครือข่ายเพียร์ทูเพียร์

ตารางที่ 4.6 ตารางการทดสอบการดาวน์โหลดไฟล์วีดิโอ

ลำดับ	อินพุต	ระบบ	ผลที่คาดว่าจะได้รับ	ผลที่เกิดขึ้นจริง
1	ผู้ใช้กดคีย์เฟรมของวีดิโอที่ต้องการ	ปกติ	โปรแกรมดาวน์โหลดไฟล์วีดิโอมาเก็บไว้ในเครื่องของผู้ใช้	โปรแกรมดาวน์โหลดไฟล์วีดิโอมาเก็บไว้ในเครื่องของผู้ใช้

ตารางที่ 4.6 (ต่อ) ตารางการทดสอบการดาวน์โหลดไฟล์วีดิโอ

ลำดับ	อินพุต	ระบบ	ผลที่คาดว่าจะได้รับ	ผลที่เกิดขึ้นจริง
2	ผู้ใช้กดคีย์เฟรมของวีดิโอที่ต้องการ	ไม่ปกติ (เพียร์ที่มีวีดิโอที่ถูกเลือกทำการปิดระบบ)	โปรแกรมไม่สามารถดาวน์โหลดไฟล์วีดิโอ และแจ้งเตือนให้ผู้ใช้ทราบ	โปรแกรมไม่สามารถดาวน์โหลดไฟล์วีดิโอ และแจ้งเตือนให้ผู้ใช้ทราบ
3	ผู้ใช้กดคีย์เฟรมของวีดิโอที่ต้องการ	ปกติ (ไฟล์วีดิโอไม่มีอยู่บนเครื่องที่ต้องการดาวน์โหลด)	โปรแกรมไม่สามารถดาวน์โหลดไฟล์วีดิโอ และแจ้งเตือนให้ผู้ใช้ทราบ	โปรแกรมไม่สามารถดาวน์โหลดไฟล์วีดิโอ และแจ้งเตือนให้ผู้ใช้ทราบ

4.3 การทดลองหาประสิทธิภาพการค้นหา

ผลการทดลองจากการค้นหาไฟล์วีดิโอโดยการใช้วีดิโอตัวอย่างบนระบบการค้นหาวีดิโอบนเครือข่ายเพียร์ทูเพียร์โดยใช้โมเดลการปรับตัวอัตโนมัติใช้การสังเกตจากเนื้อหาของวีดิโอและชื่อเรื่อง แล้วใช้หลักการเปรียบเทียบผลของค่า Recall และค่า Precision ซึ่งเป็นค่าที่ได้จากการคำนวณสมการต่อไปนี้

$$\%\text{ความถูกต้อง (Recall)} = \frac{\text{จำนวนวีดิโอที่ค้นหาได้ที่มีเนื้อหาเกี่ยวข้องกับวีดิโอตัวอย่าง}}{\text{จำนวนวีดิโอที่มีเนื้อหาเกี่ยวข้องกับวีดิโอตัวอย่างทั้งหมด}} \times 100 \qquad (4.1)$$

$$\%\text{ความถูกต้อง (Precision)} = \frac{\text{จำนวนวีดิโอที่ค้นหาได้ที่มีเนื้อหาเกี่ยวข้องกับวีดิโอตัวอย่าง}}{\text{จำนวนวีดิโอผลลัพธ์ที่ค้นหาได้ทั้งหมด}} \times 100 \qquad (4.2)$$

ยกตัวอย่าง เช่น วีดิโอที่มีเนื้อเกี่ยวกับเรื่อง A มีทั้งหมด 37 ไฟล์และสามารถค้นหาได้ทั้งหมด 36 ไฟล์ดังนั้น

$$\%\ \text{Recall} = \frac{36}{37} \times 100 = 97.30\ \%$$

4.3.1 การเปรียบเทียบประสิทธิภาพการค้นหาโดยมีข้อมูลรวบรวมไว้อยู่ในฐานข้อมูลเดียวและมีการปรับตัวอัตโนมัติภายในเพียร์

ในการทดลองนี้เป็นการทดลองเพื่อหาประสิทธิภาพของการค้นหาจากฐานข้อมูลเดียว คือ จะทำการค้นหาข้อมูลไฟล์วีดิโอจากเครื่องเดียวเท่านั้นหรือการค้นหาไฟล์วีดิโอที่อยู่ภายในเครื่องนั่นเอง และมีการปรับปรุงตัวเวกเตอร์ที่ใช้ทำการค้นหาจากสมการ

$$\vec{v}_q^{*} = \alpha(v_q) + \beta \sum_{j \in Pos} a_r^{(i)} \overline{w}_{jr} - \gamma \sum_{j \in Neg} a_j^{(j)} \overline{w}_{jr} \qquad (4.3)$$

โดยที่ $\sum_{j \in Pos} \overline{v}_{jr}$ นั้นได้มาจากการนำเวกเตอร์ที่มีค่าของ Cosine Similarity ที่มากกว่า 0.9 (Threshold Value) ทั้งหมดมา และมีพารามิเตอร์ $\alpha = 1, \beta = 1, \gamma = 0$

การทดลองนี้จะทำการทดลองสุ่มวีดิโอตัวอย่างที่จะนำทำการค้นหาทั้งหมด 41 ตัวอย่าง แล้วทำการวัดค่า Recall เฉลี่ยของแต่ละรอบของการปรับปรุงค่าเวกเตอร์

จากรูปที่ 4.1 การทดลองจะเห็นได้ว่ามีจำนวนวีดิโอจากในฐานข้อมูลที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 15 ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 37 ไฟล์ เมื่อนำมาคำนวณหาค่า Recall ในรอบที่ 1 จะได้ว่า

$$\% \text{ Recall} = \frac{15}{37} \times 100 = 40.54 \%$$

จากรูปที่ 4.2 การทดลองจะเห็นได้ว่ามีจำนวนวีดิโอจากในฐานข้อมูลที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 32ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 32 ไฟล์ เมื่อนำมาคำนวณหาค่า Recall ในรอบที่ 2 จะได้ว่า

$$\% \text{ Recall} = \frac{32}{37} \times 100 = 86.49 \%$$

จากรูปที่ 4.3 การทดลองจะเห็นได้ว่ามีจำนวนวีดิโอจากในฐานข้อมูลที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 32ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 37 ไฟล์ เมื่อนำมาคำนวณหาค่า Recall ในรอบที่ 3 จะได้ว่า

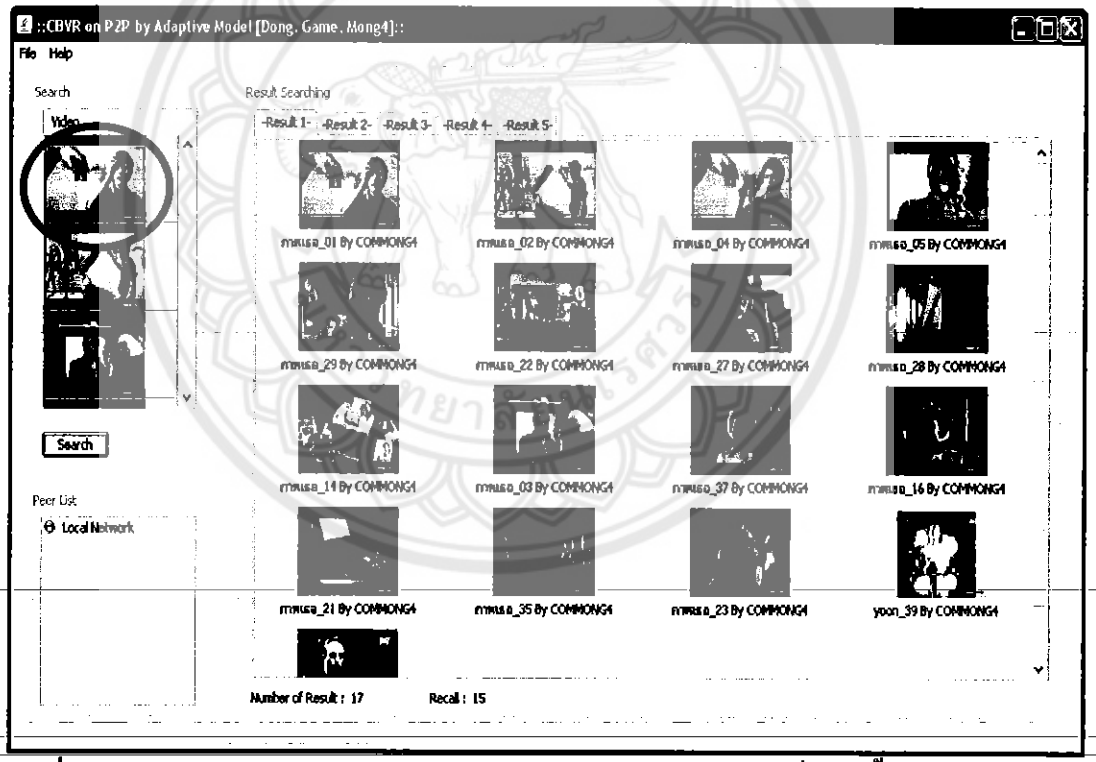
$$\% \text{ Recall} = \frac{32}{37} \times 100 = 86.49 \%$$

จากรูปที่ 4.4 การทดลองจะเห็นได้ว่ามีจำนวนวีดิโอจากในฐานข้อมูลที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 32ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 37 ไฟล์ เมื่อนำมาคำนวณหาค่า Recall ในรอบที่ 4 จะได้ว่า

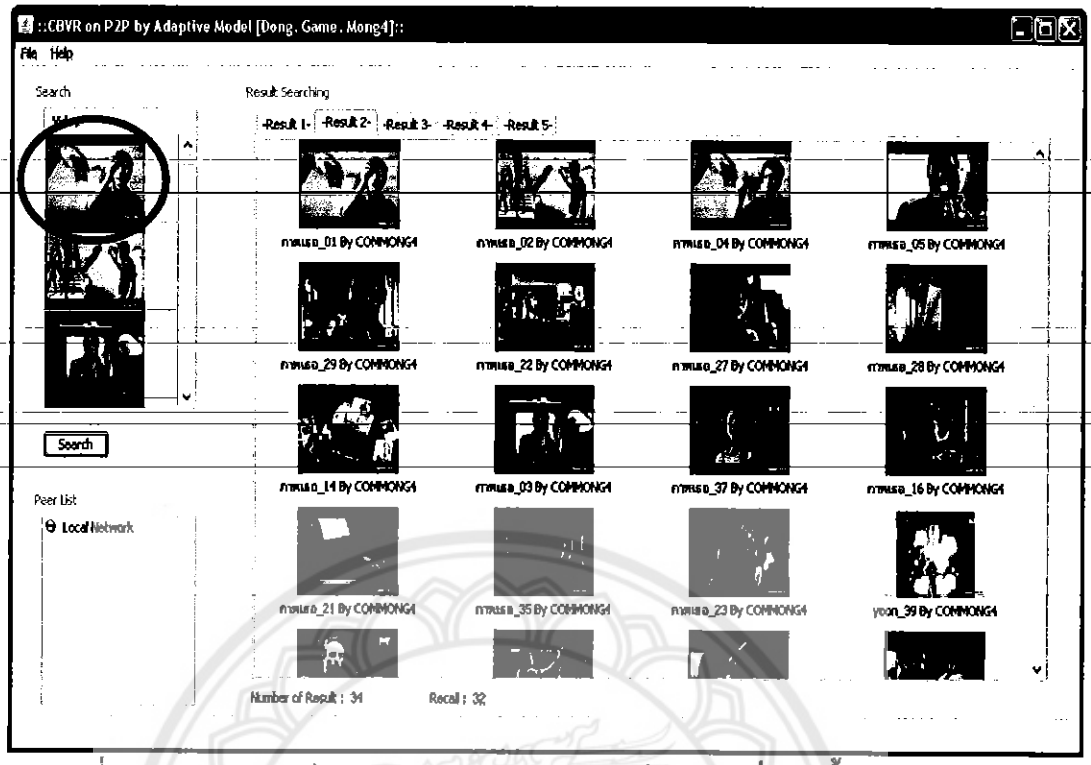
$$\% \text{ Recall} = \frac{32}{37} \times 100 = 86.49\ \%$$

จากรูปที่ 4.5 การทดลองจะเห็นได้ว่ามีจำนวนวีดิโอจากในฐานข้อมูลที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 32ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 37 ไฟล์ เมื่อนำมาคำนวณหาค่า Recall ในรอบที่ 5 จะได้ว่า

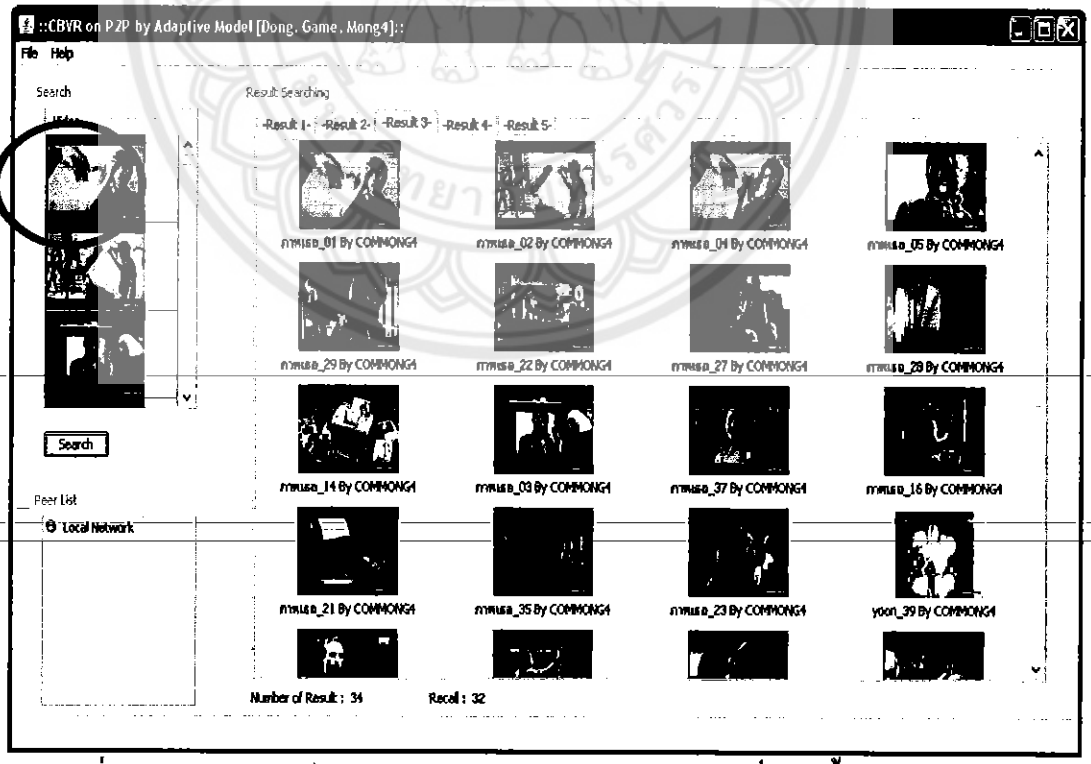
$$\% \text{ Recall} = \frac{32}{37} \times 100 = 86.49\ \%$$



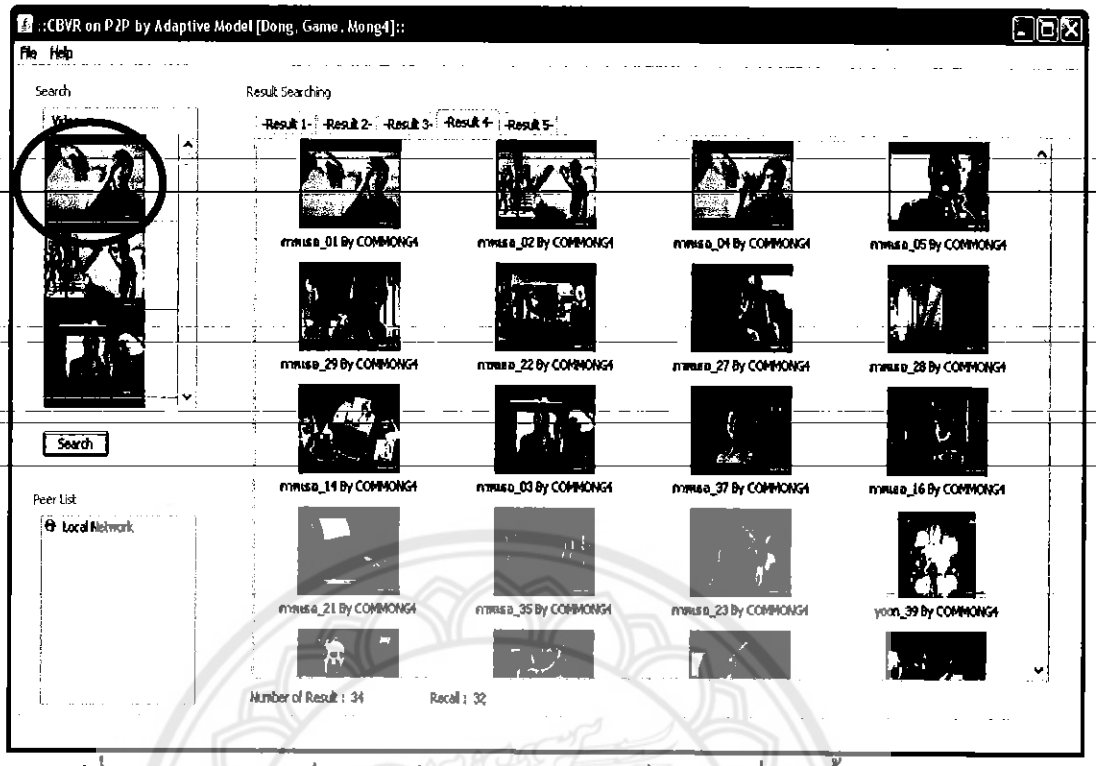
รูปที่ 4.1 แสดงผลลัพธ์จากการค้นหาจากภายในเพียร์ (Local) ในรอบที่ 1 มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 15ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 37 ไฟล์



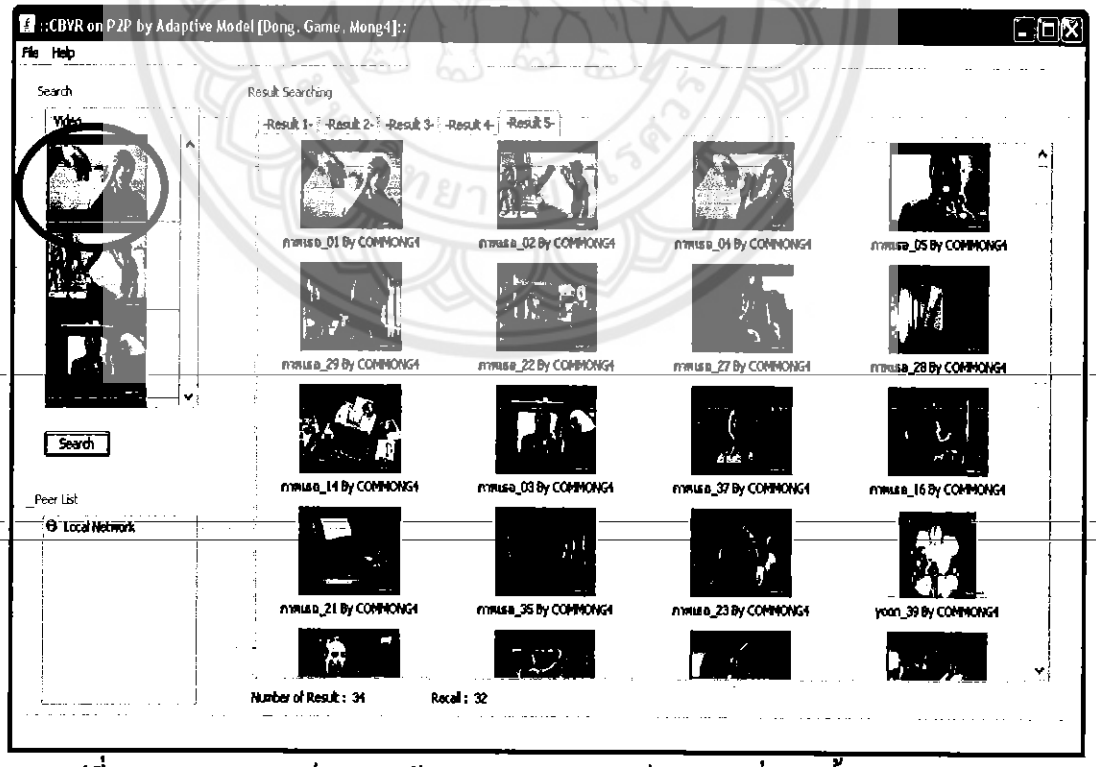
รูปที่ 4.2 แสดงผลลัพธ์จากการค้นหาจากภายในเพียร์ ในรอบที่ 2 มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 32ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 37 ไฟล์



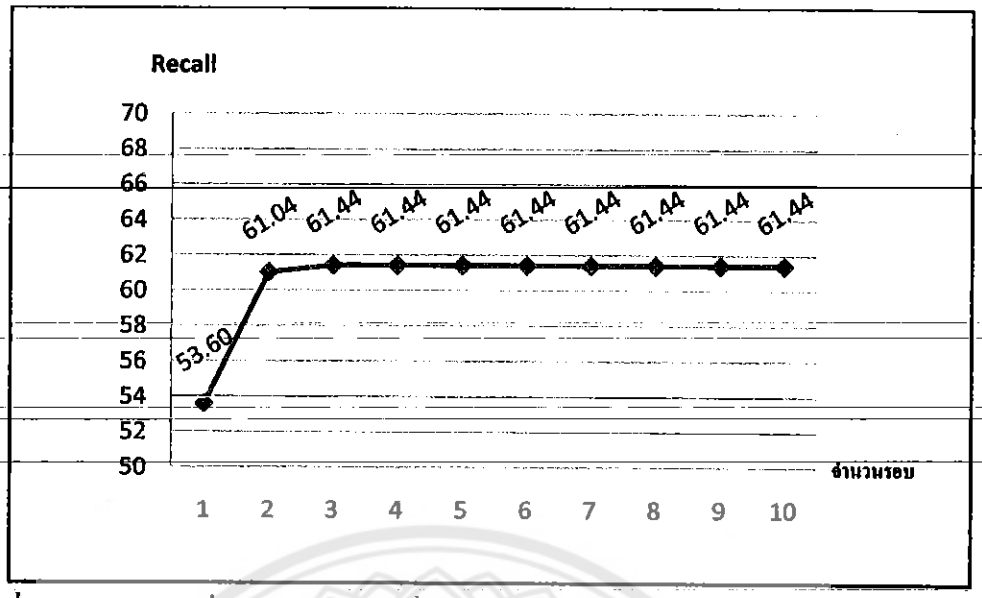
รูปที่ 4.3 แสดงผลลัพธ์จากการค้นหาจากภายในเพียร์ ในรอบที่ 3 มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 32ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 37 ไฟล์



รูปที่ 4.4 แสดงผลลัพธ์จากการค้นหาจากภายในเพียร์ ในรอบที่ 4 มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 32 ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 37 ไฟล์



รูปที่ 4.5 แสดงผลลัพธ์จากการค้นหาจากภายในเพียร์ ในรอบที่ 5 มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 32 ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 37 ไฟล์



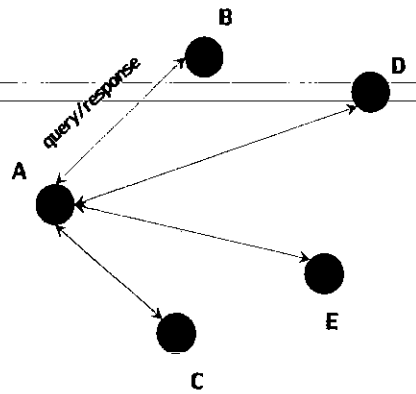
รูปที่ 4.6 แสดงค่าเฉลี่ยของ RECALL ที่ได้ในแต่ละรอบของการค้นหาภายในเพียร์จากวีดิโอตัวอย่าง

4.3.2 การเปรียบเทียบประสิทธิภาพการค้นหาโดยมีฐานข้อมูลแบบกระจาย และปรับตัวอัตโนมัติบนระบบเพียร์ทูเพียร์

ในการทดลองนี้เป็นการทดลองเพื่อหาประสิทธิภาพของการค้นหาจากหลายๆ ฐานข้อมูล คือจะทำการค้นหาข้อมูลไฟล์วีดิโอจากเครื่องลูกข่ายที่อยู่ภายในระบบซึ่งจะรวมถึงภายในเครื่องที่ทำการค้นหาด้วย และจะมีการปรับปรุงตัวเวกเตอร์ที่ใช้ทำการค้นหาตามสมการที่ (4.3)

โดยที่ $\sum_{j \in Pos} \bar{v}_{jt}$ นั้นได้มาจากการนำเวกเตอร์ที่มีค่าของ Cosine Similarity ที่มากกว่า 0.9 (Threshold Value) ทั้งหมดมา และมีพารามิเตอร์ $\alpha = 1, \beta = 1, \gamma = 0$

การทดลองนี้จะทำการทดลองสุ่มวีดิโอตัวอย่างที่จะนำทำการค้นหาทั้งหมด 41 ตัวอย่าง แล้วทำการวัดค่า Recall เฉลี่ยของแต่ละรอบของการปรับปรุงค่าเวกเตอร์



รูปที่ 4.7 แสดง peer-to-peer โมเดล

จากรูปที่ 4.8 การทดลองจะเห็นได้ว่ามีจำนวนวีดิโอจากในฐานข้อมูลที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 178ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 220 ไฟล์ เมื่อนำมาคำนวณหาค่า Recall ในรอบที่ 1 จะได้ว่า

$$\% \text{ Recall} = \frac{102}{220} \times 100 = 46.36\ \%$$

จากรูปที่ 4.9 การทดลองจะเห็นได้ว่ามีจำนวนวีดิโอจากในฐานข้อมูลที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 178ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 220 ไฟล์ เมื่อนำมาคำนวณหาค่า Recall ในรอบที่ 2 จะได้ว่า

$$\% \text{ Recall} = \frac{178}{220} \times 100 = 80.90\ \%$$

จากรูปที่ 4.10 การทดลองจะเห็นได้ว่ามีจำนวนวีดิโอจากในฐานข้อมูลที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 178ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 220 ไฟล์ เมื่อนำมาคำนวณหาค่า Recall ในรอบที่ 3 จะได้ว่า

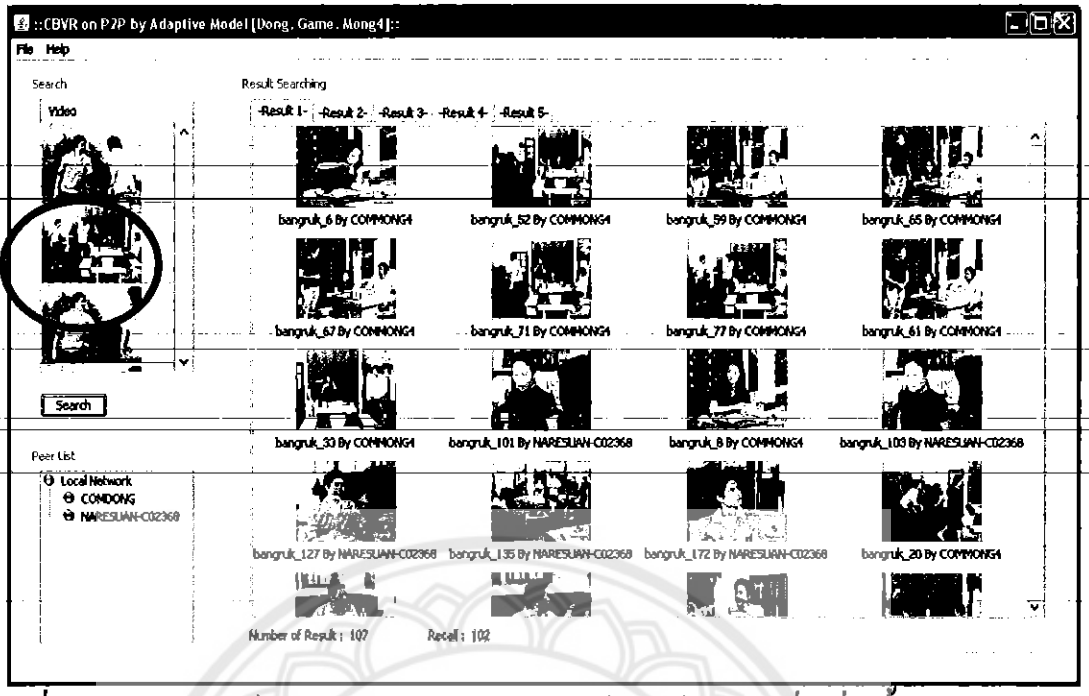
$$\% \text{ Recall} = \frac{178}{220} \times 100 = 80.90\ \%$$

จากรูปที่ 4.11 การทดลองจะเห็นได้ว่ามีจำนวนวีดิโอจากในฐานข้อมูลที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 178ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 220 ไฟล์ เมื่อนำมาคำนวณหาค่า Recall ในรอบที่ 4 จะได้ว่า

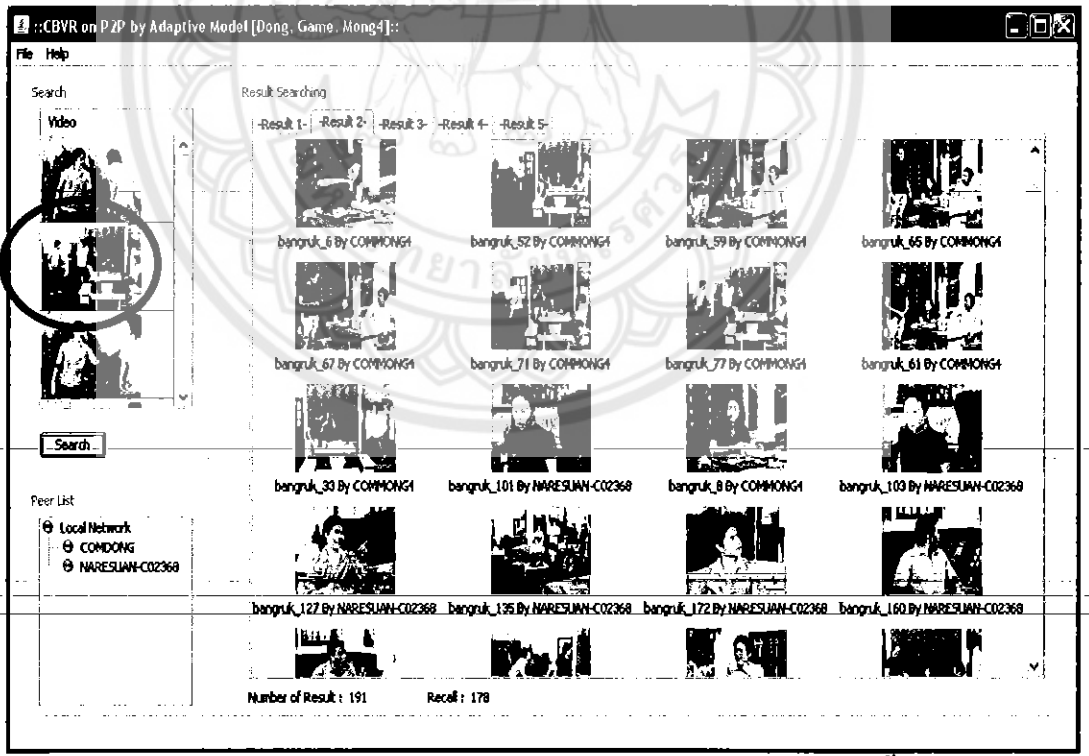
$$\% \text{ Recall} = \frac{178}{220} \times 100 = 80.90\ \%$$

จากรูปที่ 4.12 การทดลองจะเห็นได้ว่ามีจำนวนวีดิโอจากในฐานข้อมูลที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 178ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 220 ไฟล์ เมื่อนำมาคำนวณหาค่า Recall ในรอบที่ 5 จะได้ว่า

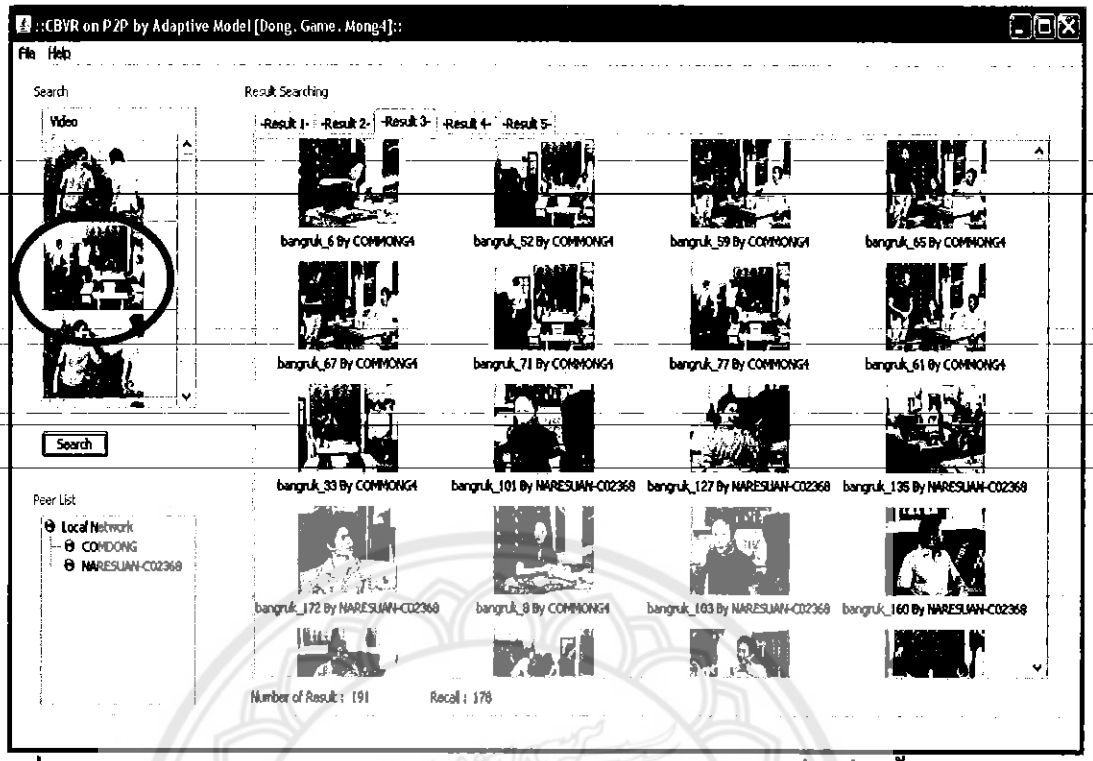
$$\% \text{ Recall} = \frac{178}{220} \times 100 = 80.90\ \%$$



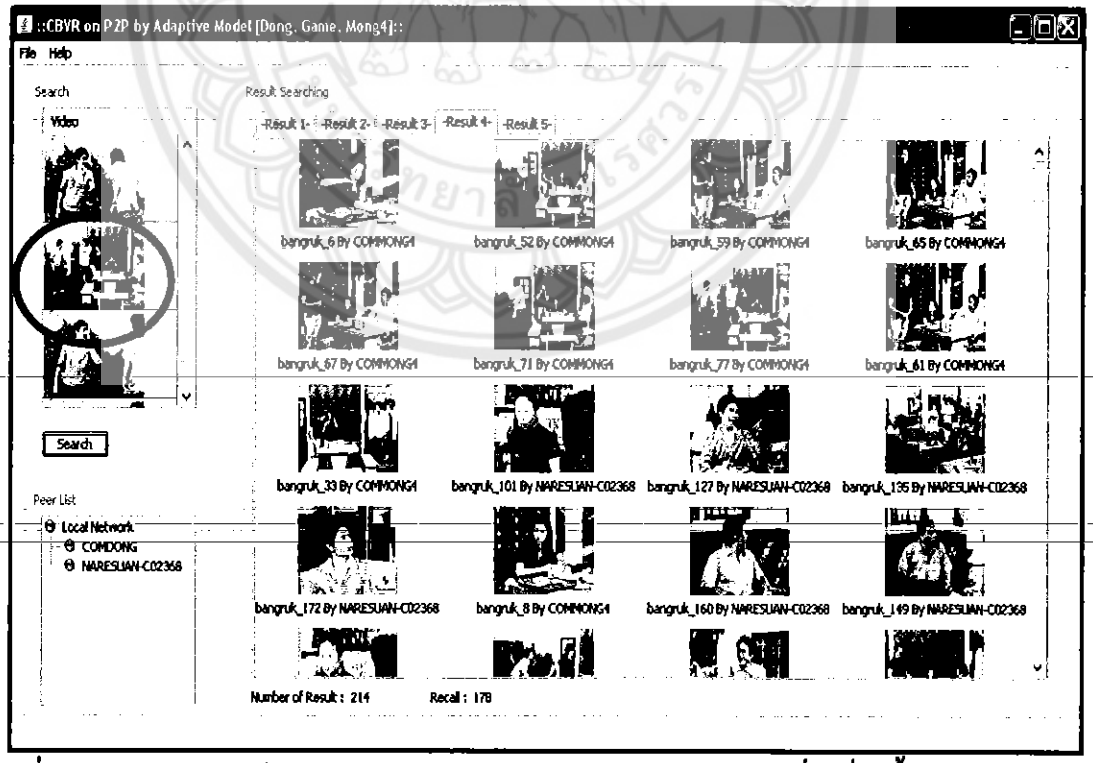
รูปที่ 4.8 แสดงผลลัพธ์จากการค้นหาบนเครือข่ายเพียร์ทูเพียร์ ในรอบที่ 1 ที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 102 ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 220 ไฟล์



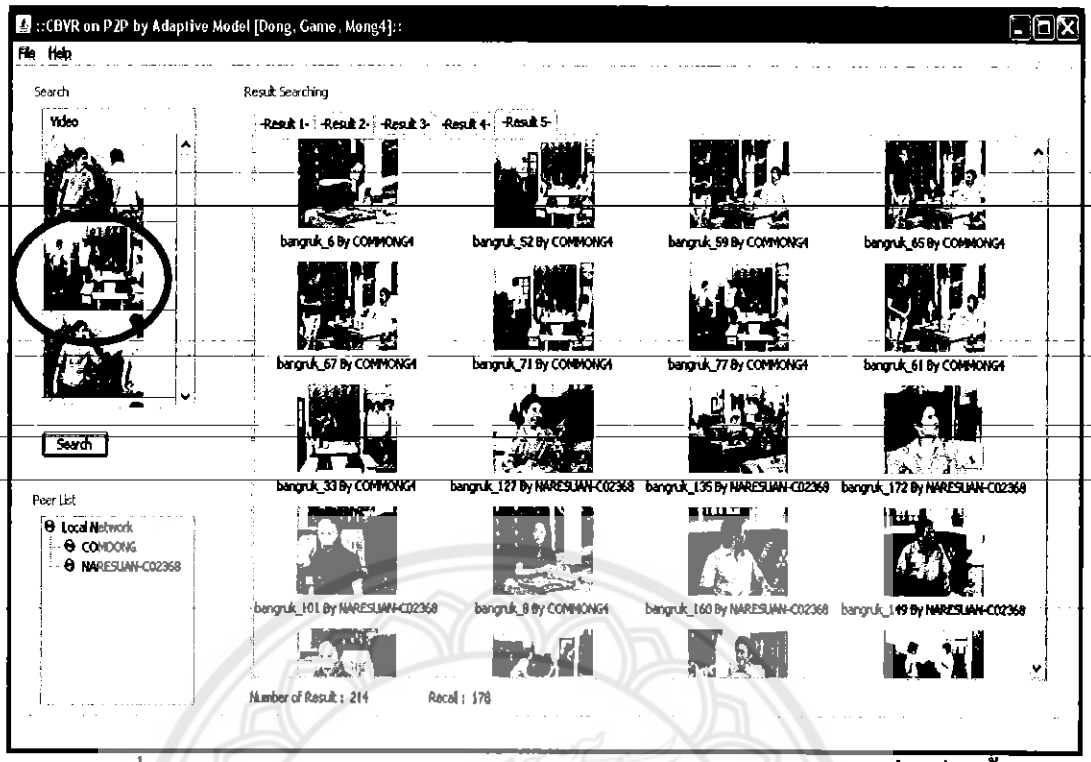
รูปที่ 4.9 แสดงผลลัพธ์จากการค้นหาบนเครือข่ายเพียร์ทูเพียร์ ในรอบที่ 2 ที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 178ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 220 ไฟล์



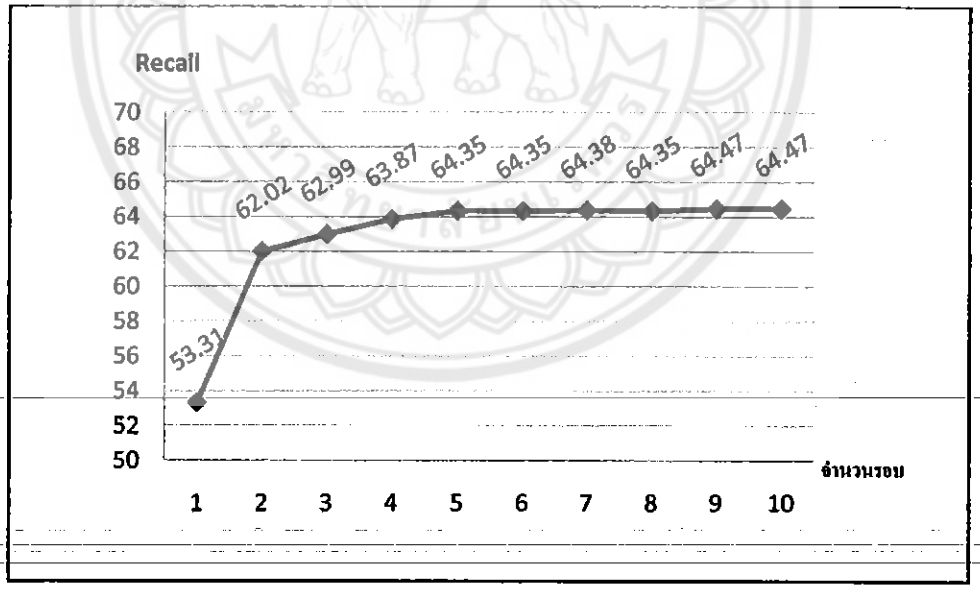
รูปที่ 4.10 แสดงผลลัพธ์จากการค้นหาบนเครือข่ายเพียร์ทูเพียร์ ในรอบที่ 3 ที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 178ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 220 ไฟล์



รูปที่ 4.11 แสดงผลลัพธ์จากการค้นหาบนเครือข่ายเพียร์ทูเพียร์ ในรอบที่ 4 ที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 178ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 220 ไฟล์

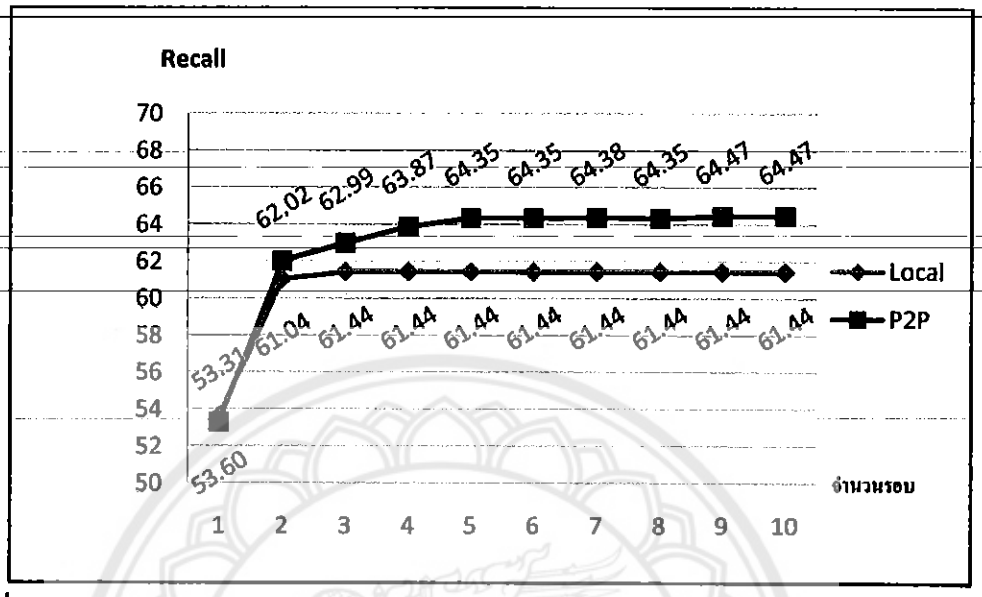


รูปที่ 4.12 แสดงผลลัพธ์จากการค้นหาบนเครือข่ายเพียร์ทูเพียร์ ในรอบที่ 5 ที่มีเนื้อหาตรงกับวีดิโอที่ใช้ค้นหาอยู่ 178 ไฟล์จากไฟล์วีดิโอที่มีเนื้อหาเดียวกัน 220 ไฟล์



รูปที่ 4.13 แสดงค่าเฉลี่ยของ Recall ที่ได้ในแต่ละรอบของการค้นหาบนระบบเพียร์ทูเพียร์จากวีดิโอตัวอย่าง

4.3.3 การเปรียบเทียบประสิทธิภาพระหว่างการปรับตัวอัตโนมัติภายในเพียร์กับการปรับตัวอัตโนมัติบนระบบเพียร์ทูเพียร์



รูปที่ 4.14 แสดงการเปรียบเทียบประสิทธิภาพระหว่างการปรับตัวอัตโนมัติภายในเพียร์กับการปรับตัวอัตโนมัติบนระบบเพียร์ทูเพียร์

จากผลการทดลองจะสังเกตเห็นว่า จำนวนของการปรับตัวนั้นจะให้ผลของประสิทธิภาพของการค้นหาที่ดีอยู่ในแค่ช่วง 3 รอบเท่านั้นหลังจากนั้นในรอบต่อๆไปประสิทธิภาพของการค้นหาจะคงที่ เพราะฉะนั้นจำนวนรอบของการปรับปรุงเวกเตอร์ที่ใช้ค้นหานั้นควรจะอยู่ที่ประมาณสามรอบก็เพียงพอที่จะให้ได้ประสิทธิภาพที่ดีได้

และจากผลของประสิทธิภาพของการค้นหาที่ได้จากทั้ง การค้นหาโดยมีข้อมูลรวบรวมไว้อยู่ในฐานข้อมูลเดียว และมีการปรับตัวอัตโนมัติภายในตัวเองกับ การค้นหาโดยมีข้อมูลแบบกระจายและปรับตัวอัตโนมัติบนระบบเพียร์ทูเพียร์ นั้นจะสังเกตเห็นว่าประสิทธิภาพการค้นหาของการค้นหาโดยมีข้อมูลแบบกระจายนั้นจะให้ผลที่ดีกว่าการค้นหาแบบฐานข้อมูลเดียว นั่นก็เพราะว่าการปรับปรุงเวกเตอร์ที่ใช้ค้นหาตามสมการที่ 4.3 ของระบบการค้นหาแบบกระจายนั้นจะปรับปรุงได้เวกเตอร์ใหม่ที่ดีกว่าของระบบการค้นหาแบบฐานข้อมูลเดียว ดังตัวอย่างต่อไปนี้

1. การค้นหาแบบฐานข้อมูลเดียว

ฐานข้อมูลมีเพียงชุดเดียวโดยที่วีดิโอที่อยู่ภายในฐานข้อมูลมีดังต่อไปนี้

$$Set = \{V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9, V_{10}\}$$

ดังนั้นหลังจากการค้นหาจะมีวีดิโอที่ค้นหาได้ทั้งหมด 5 ไฟล์ก็คือ V_2, V_3, V_5, V_6, V_7 ดังนั้น

$$Set = \{V_1, \underline{V_2}, \underline{V_3}, V_4, \underline{V_5}, \underline{V_6}, \underline{V_7}, V_8, V_9, V_{10}\}$$

จากนั้นจะสามารถหาเวกเตอร์ที่ได้จากการปรับปรุงได้จากสมการ 4.3 ดังนี้

$$\vec{v}_q^* = v_q + v_{mean}$$

$$\vec{v}_q^* = v_q + \left(\frac{v_2 + v_3 + v_5 + v_6 + v_7}{5}\right)$$

2. การค้นหาที่มีฐานข้อมูลแบบกระจาย

มีฐานข้อมูลทั้งหมด 2 ชุดโดยที่

$$Set\ 1 = \{V_1, V_2, V_3, V_4\}$$

$$Set\ 2 = \{V_5, V_6, V_7, V_8, V_9, V_{10}\}$$

ดังนั้นหลังจากการค้นหาจะมีวิดิโอที่ค้นหาได้ทั้งหมด 5 ไฟล์ก็คือ V_2, V_3 จากฐานข้อมูลชุดที่ 1 และ V_5, V_6, V_7 จากฐานข้อมูลชุดที่ 2 ดังนั้น

$$Set\ 1 = \{V_1, \underline{V_2}, \underline{V_3}, V_4\}$$

$$Set\ 2 = \{\underline{V_5}, \underline{V_6}, \underline{V_7}, V_8, V_9, V_{10}\}$$

จากนั้นจะสามารถหาเวกเตอร์ที่ได้จากการปรับปรุงได้จากสมการ 4.3 ดังนี้

$$\vec{v}_q^* = v_q + v_{mean}$$

$$\vec{v}_q^* = v_q + \left(\frac{v_2 + v_3}{2}\right) + \left(\frac{v_5 + v_6 + v_7}{3}\right)$$

$$\vec{v}_q^* = v_q + \left(\frac{2v_2 + 2v_3 + 3v_5 + 3v_6 + 3v_7}{6}\right)$$

ดังนั้นจะสังเกตเห็นว่าในการค้นหาครั้งหนึ่งของการค้นหาด้วยวิธีทั้งสองมีวีดิโอที่ค้นหาเจอทั้งหมด 5 ไฟล์เท่ากันแต่เวกเตอร์หลังการปรับปรุงของการค้นหาแบบกระจายนั้นจะได้ค่าเวกเตอร์ใหม่คือ

$$\vec{v}_q^{*} = v_q + \left(\frac{v_2 + v_3 + v_5 + v_6 + v_7}{5}\right)$$

ที่ดีกว่าเวกเตอร์ที่ได้จากการค้นหาแบบฐานข้อมูลเดียวที่ได้ค่าของเวกเตอร์ใหม่คือ

$$\vec{v}_q^{*} = v_q + \left(\frac{2v_2 + 2v_3 + 3v_5 + 3v_6 + 3v_7}{6}\right)$$

เมื่อการค้นหาแบบกระจายนั้นมีเวกเตอร์ที่ดีกว่าก็ส่งผลทำให้ประสิทธิภาพของการค้นหาหรือเปอร์เซ็นความถูกต้องนั้นมีมากขึ้นตามไปด้วย การที่มีเวกเตอร์ที่ดีกว่านั้นจะทำให้การเปรียบเทียบของ Cosine Similarity นั้นจะให้ผลของการเปรียบเทียบที่ดีกว่าเพราะว่าเวกเตอร์นำมาถูกปรับปรุงไปซึ่งอาจจะทำให้เวกเตอร์ที่ถูกปรับปรุงนั้นมีความใกล้เคียงกับเวกเตอร์ที่ใช้ค้นหาหรือแตกต่างมากขึ้นส่งผลให้การค้นหามีประสิทธิภาพมากขึ้น

4.3.4 การเลือกค่าที่เหมาะสม

ในโมเดลการปรับตัวอัตโนมัติมีการใช้ค่าตัวแปรต่างๆจากสมการที่ 4.3 ในการปรับปรุงเวกเตอร์เพื่อให้ได้ค่าเวกเตอร์ใหม่ที่ทำให้ได้ค่าเปอร์เซ็นความถูกต้องดีกว่าเดิม ซึ่งประกอบด้วยค่าต่อไปนี้

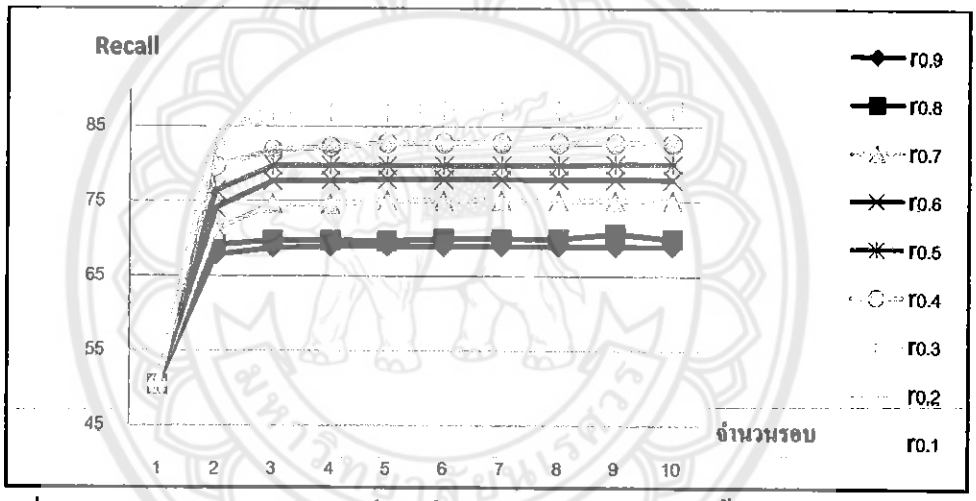
- Upper Threshold เพื่อกำหนดค่าผลลัพธ์ในด้านบวก (Positive Sample) ซึ่งหมายถึงผลลัพธ์ที่มีเวกเตอร์ใกล้เคียงกับเวกเตอร์ที่ใช้ค้นหา
- Lower Threshold เพื่อกำหนดค่าผลลัพธ์ในด้านลบ (Negative Sample) ซึ่งหมายถึงผลลัพธ์ที่มีเวกเตอร์แตกต่างกับเวกเตอร์ที่ใช้ค้นหา
- Alpha ค่าน้ำหนักของเวกเตอร์ดั้งเดิม
- Beta ค่าน้ำหนักของเวกเตอร์ค่าเฉลี่ยของผลลัพธ์ในด้านบวก
- Gamma ค่าน้ำหนักของเวกเตอร์ค่าเฉลี่ยของผลลัพธ์ในด้านลบ

ดังนั้นการทดลองต่อไปนี้จะเป็นทดลองเพื่อหาค่าที่เหมาะสมในตัวแปรต่างๆ โดยใช้วีดิโอตัวอย่าง 10 ไฟล์

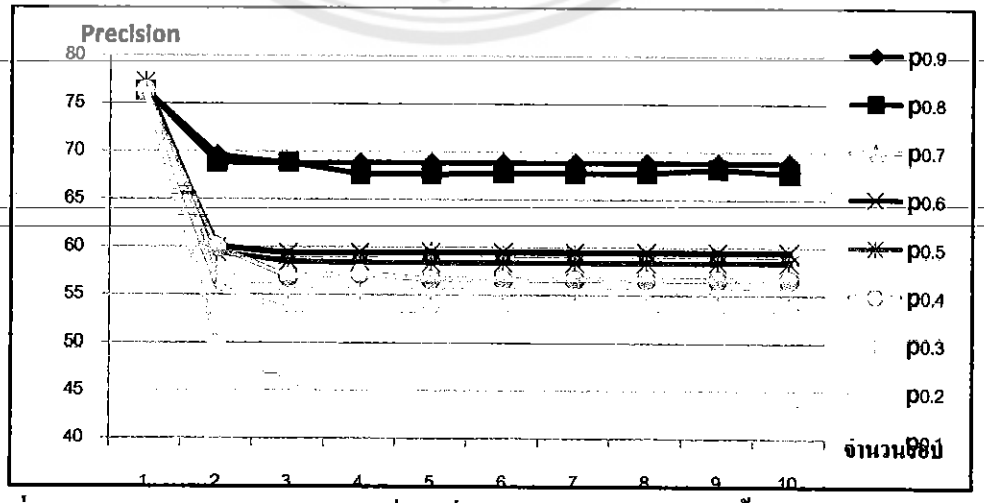
การทดลองเพื่อหาค่าพารามิเตอร์ต่างๆ ที่เหมาะสมต่อไปนี้จะเป็นการทดลองโดยการสุ่มวีดิโอตัวอย่างมาทำการค้นหาทั้งหมด 10 ตัวอย่าง แล้วทำการวัดค่าของ Recall และ Precision เฉลี่ยของแต่ละรอบของการปรับปรุงเวกเตอร์

4.3.4.1 การทดลองเปลี่ยนค่า Upper Threshold โดยให้ค่า Lower Threshold = 0 , Alpha = 1, Beta = 1, Gamma = 0

การทดลองนี้จะทำการทดลองเพื่อหาค่า Upper Threshold ที่เหมาะสมสำหรับการนำมาใช้ในแอพพลิเคชั่น โดยจะทำการเปลี่ยนค่า Upper Threshold ไปเรื่อยๆตั้งแต่ 0.9 จนถึง 0.1 แล้วทำการวัดค่า Recall และค่า Precision เพื่อนำมาหาความสัมพันธ์ระหว่างค่าทั้งสองเพื่อที่จะสามารถหาค่าหรือช่วงของค่า Upper Threshold ที่เหมาะสมที่สุดที่สามารถนำมาใช้ให้ได้ประสิทธิภาพในการค้นหาที่ดีที่สุดได้



รูปที่ 4.15 แสดงกราฟค่า Recall เมื่อเปลี่ยนค่า Upper Threshold ตั้งแต่ 0.9 , 0.8 , … , 0.1

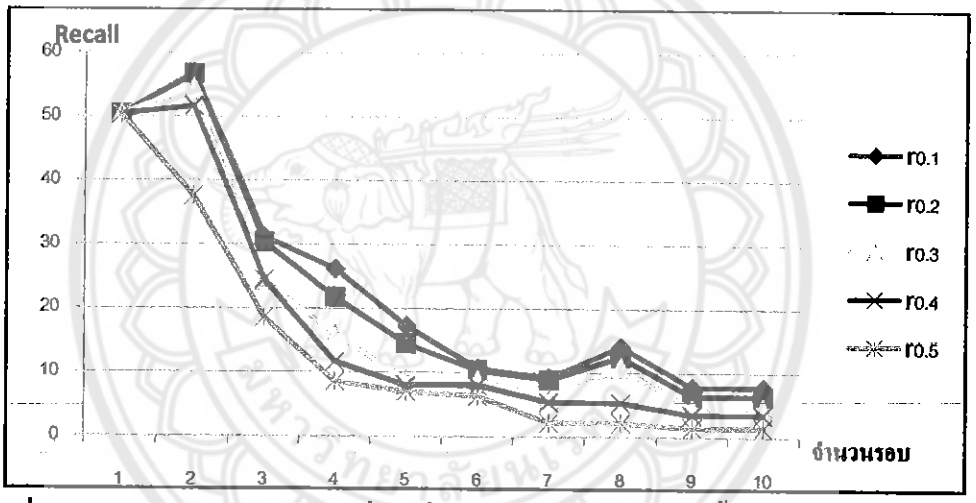


รูปที่ 4.16 แสดงกราฟค่า Precision เมื่อเปลี่ยนค่า Upper Threshold ตั้งแต่ 0.9 , 0.8 , … , 0.1

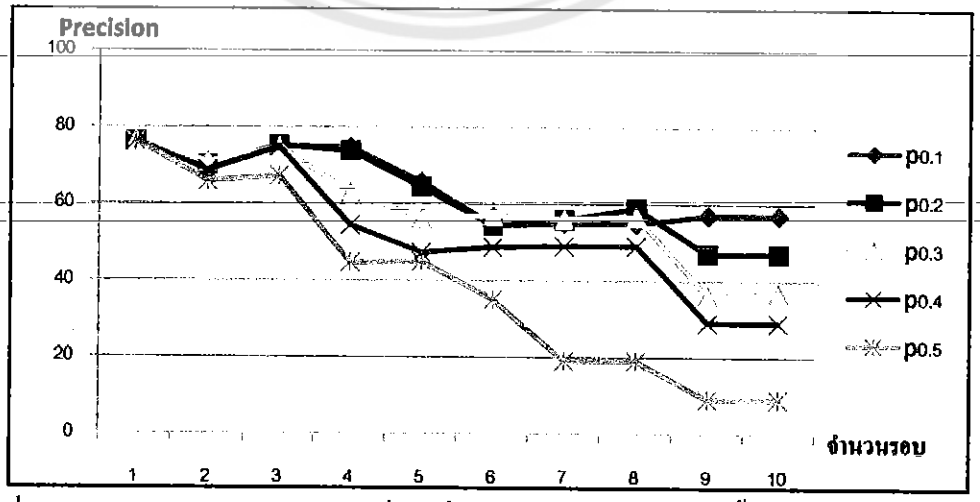
จากรูปที่ 4.15 และรูปที่ 4.16 จะเห็นได้ว่าค่า recall มีค่าเพิ่มขึ้นในแต่ละรอบการทำงานอยู่แล้ว แต่ค่า precision กลับมีค่าลดลงในแต่ละรอบการทำงาน ซึ่งมีค่าแตกต่างกันมากในรอบที่สองและรอบที่สาม ดังนั้นจึงควรเลือกค่า Upper Threshold ที่ทำให้ค่า precision ลดลงน้อยที่สุด คือช่วง Upper Threshold ระหว่าง 0.8 ถึง 0.9 เพื่อนำไปปรับปรุงค่าเวกเตอร์ที่ดีขึ้นในรอบต่อไป

4.3.4.2 การทดลองเปลี่ยนค่า Lower Threshold โดยให้ค่า Upper Threshold = 0.9, Alpha = 1, Beta = 1, Gamma = 1

การทดลองนี้จะทำการทดลองเพื่อหาค่า Lower Threshold ที่เหมาะสมสำหรับการนำมาใช้ในแอพพลิเคชั่น โดยจะทำการเปลี่ยนค่า Lower Threshold ไปเรื่อยๆตั้งแต่ 0.1 จนถึง 0.5 แล้วทำการวัดค่า Recall เพื่อที่จะสามารถหาค่าหรือช่วงของค่า Lower Threshold ที่เหมาะสมที่สุด



รูปที่ 4.17 แสดงกราฟค่า Recall เมื่อเปลี่ยนค่า Lower Threshold ตั้งแต่ 0.1 , 0.2 , … , 0.5

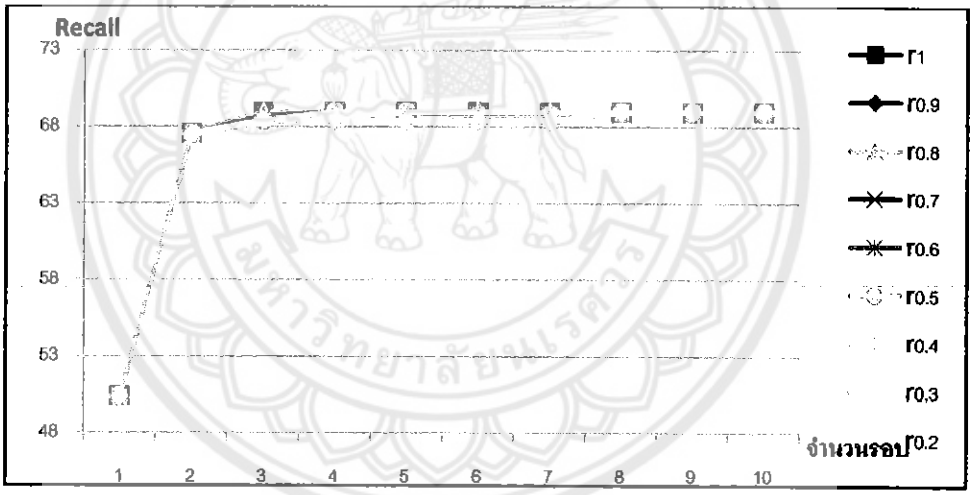


รูปที่ 4.18 แสดงกราฟค่า precision เมื่อเปลี่ยนค่า Lower Threshold ตั้งแต่ 0.1 , 0.2 , … , 0.5

จากรูปที่ 4.17 และรูปที่ 4.18 จะเห็นได้ ว่าค่า precision มีค่ามากกว่าค่า recall มากและมีอัตราการลดลงในรอบที่สองและรอบที่สามไม่มากนักซึ่งต่างจากค่า recall ที่มีค่าน้อยและมีอัตราการลดลงค่อยข้างมากดังนั้นในการเลือกค่า Lower Threshold จึงพิจารณาค่า recall เป็นสำคัญนั้นคือเลือกค่า Lower Threshold ช่วง 0.1 ถึง 0.2 ที่มีอัตราการลดลงของค่า recall น้อยที่สุดและมีค่า recall มากกว่าค่าอื่น

4.3.4.3 การทดลองเปลี่ยนค่า Beta โดยให้ค่า Upper Threshold = 0.9, Lower Threshold = 0.1, Alpha = 1, Gamma = 0

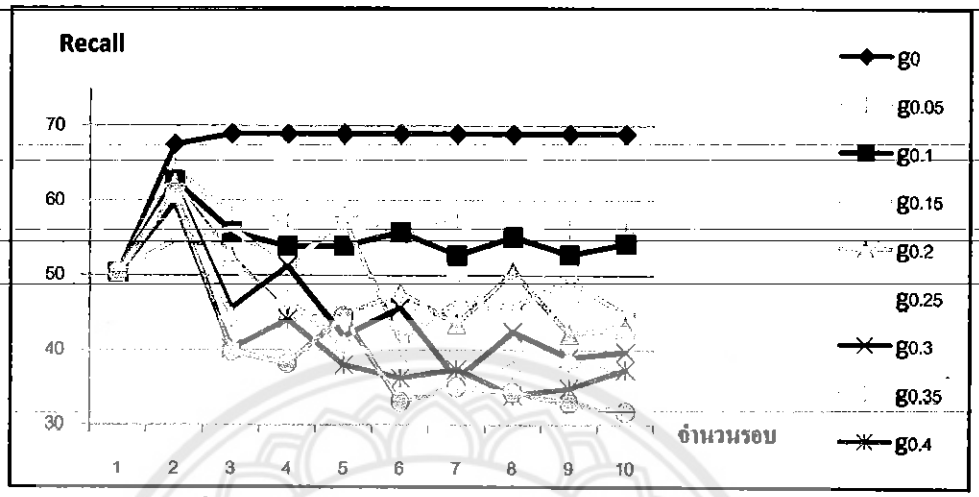
การทดลองนี้จะทำการทดลองเพื่อหาค่า Beta ที่เหมาะสมสำหรับการนำมาใช้ในแอพพลิเคชั่น โดยจะทำการเปลี่ยนค่า Beta ไปเรื่อยๆตั้งแต่ 1.0 จนถึง 0.1 แล้วทำการวัดค่า Recall เพื่อที่จะสามารถหาค่าหรือช่วงของค่า Beta ที่เหมาะสมที่สุด ซึ่งตัวแปร Beta นี้เป็นค่าน้ำหนักของค่าเฉลี่ยของเวกเตอร์ที่มีผลลัพธ์ใกล้เคียงกับวีดิโอต้นแบบ (Positive Samples) ถ้าหากต้องการให้ความสำคัญกับค่าเวกเตอร์นี้มากค่านำหนักตัวนี้จะมีมากขึ้น



รูปที่ 4.19 แสดงกราฟค่า Recall เมื่อเปลี่ยนค่า Beta ตั้งแต่ 1.0 , 0.9 , ... , 0.1

จากรูปที่ 4.19 จะเห็นว่าเมื่อเปลี่ยนค่า Beta ให้ลดลงจะทำให้ได้ค่า recall ในรอบที่สองมีค่าแตกต่างกันโดยค่า Beta ช่วง 0.7 ถึง 1 จะทำให้ได้ค่า recall ในรอบที่สองและรอบที่สามดีที่สุด

4.3.4.4 การทดลองเปลี่ยนค่า Gamma โดยให้ค่า Upper Threshold = 0.9, Lower Threshold = 0.1, Alpha = 1, Beta = 0.8



รูปที่ 4.20 แสดงค่า recall เมื่อปรับค่า Gamma เป็นค่าต่างๆ

จากผลการทดลองการเลือกค่า Beta ที่เหมาะสมนั้นจะอยู่ในช่วงตั้งแต่ 0.7 จนถึง 1 ดังนั้น เมื่อทำการทดลองหาค่า Gamma ที่เหมาะสมจึงทำการเลือกค่า Beta มาจากช่วงดังกล่าว คือเลือกค่า Beta ที่ 0.8 มาทำการทดลอง ซึ่งการทดลองที่ต้องการผลลัพธ์ที่ครบถ้วนนั้นจะต้องทำการทดลองหาค่าของ Gamma ที่เหมาะสมจากทุกๆค่าของ Beta นั่นคือค่า Beta ทั้งหมดตั้งแต่ 0.1 จนถึง 1 รวมทั้งหมด 10 ค่า แล้วจึงนำค่า Beta ของแต่ละค่ามาทำการทดลองหาค่า Gamma ที่เหมาะสมตั้งแต่ 0.1 จนถึง 0.5 ทั้งหมด 8 ค่า ดังนั้นจะได้ผลการทดลองทั้งหมด 80 ผลลัพธ์ แต่ในรายงานการทดลองนี้จะแสดงให้เห็นเฉพาะการทดลองหาค่า Gamma ที่เหมาะสมสำหรับค่า Beta ที่ 0.8 เท่านั้น

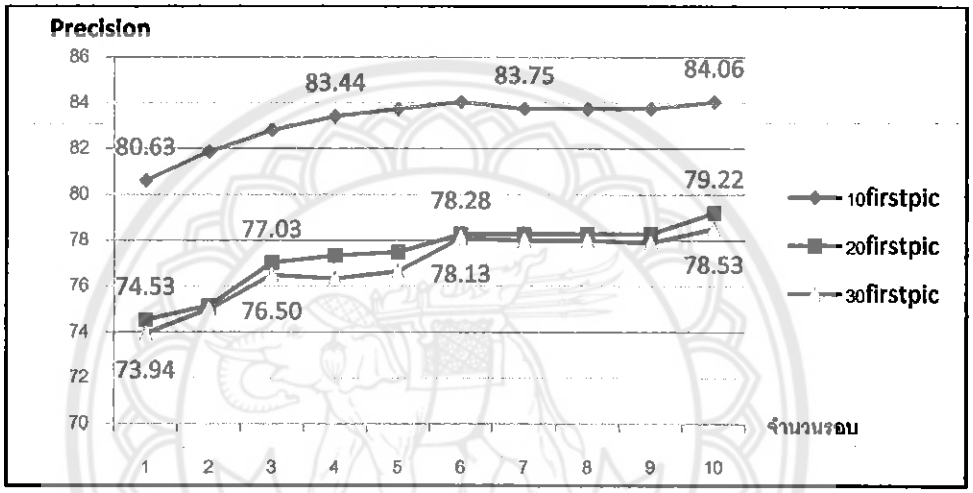
จากกราฟรูปที่ 4.20 จะเห็นว่าค่า recall จะเพิ่มขึ้นในรอบที่สองและลดลงในรอบที่ 3 ยกเว้นค่า Gamma ที่เท่ากับ 0 โดยค่า Gamma ช่วง 0 ถึง 0.1 จะไม่ทำให้ค่า recall ในรอบที่สามลดลงต่ำกว่าค่า recall ที่ได้ในการค้นหาครั้งแรก เพราะฉะนั้นค่า Gamma ที่เหมาะสมควรอยู่ในช่วง 0 ถึง 0.1

4.3.5 การวัดประสิทธิภาพในการค้นหา

หลังจากที่ได้ทดลองหาค่าตัวแปรที่เหมาะสมแล้ว ในขั้นตอนนี้จะนำค่าที่เหมาะสมให้แต่ละตัวแปรมาใช้งาน โดยใช้ค่าตัวแปรต่างๆ มีค่าดังนี้ Upper threshold เท่ากับ 0.9 Less threshold เท่ากับ 0.1 Alpha เท่ากับ 1 Beta เท่ากับ 0.8 และ Gamma เท่ากับ 0.01 เพื่อทำการทดลองวัดประสิทธิภาพของการค้นหา โดยการวัดเปอร์เซ็นความถูกต้องของการค้นหาในแต่ละรอบของการปรับปรุงเวกเตอร์เพื่อตรวจสอบว่ามีประสิทธิภาพมากน้อยเพียงใดหลังจากการใช้ค่าตัวแปรต่างๆ ที่เหมาะสมได้ผลดังนี้

ตารางที่ 4.7 ตารางแสดงค่าเฉลี่ยของเปอร์เซ็นความถูกต้องจากไฟล์วีดิโอตัวอย่างที่นำมาทดลอง จำนวน 32 ตัวอย่าง

รอบที่ / Precision	1	2	3	4	5	6	7	8	9	10
10 ภาพแรก	80.63	81.88	82.81	83.44	83.75	84.06	83.75	83.75	83.75	84.06
20 ภาพแรก	74.53	75.16	77.03	77.34	77.50	78.28	78.28	78.28	78.28	79.22
30 ภาพแรก	73.94	75.00	76.50	76.34	76.66	78.13	78.00	78.00	77.91	78.53



รูปที่ 4.21 รูปภาพแสดงกราฟค่าเฉลี่ยของเปอร์เซ็นความถูกต้องแต่ละรอบของการปรับปรุงเวกเตอร์

จากผลการทดลองเมื่อนำค่าตัวแปรที่เหมาะสมมาทำการทดลองเพื่อวัดประสิทธิภาพของการค้นหาจะสังเกตเห็นว่าผลจากการค้นหาใน 10 อันดับแรกจะได้ค่าความถูกต้องที่มากว่าผลจากการค้นหาของทั้ง 20 และ 30 อันดับ และมีแนวโน้มเพิ่มขึ้นในรอบต่อๆ ไปของการปรับปรุงค่าของเวกเตอร์ที่ใช้ค้นหา ดังนั้นจึงสรุปได้ว่าค่าของตัวแปรที่นำมาใช้ในการปรับปรุงเวกเตอร์ในส่วนของโมเดลการปรับตัวอัตโนมัตินั้นให้ผลลัพธ์ที่มีประสิทธิภาพดีขึ้น แต่แนวโน้มที่เพิ่มขึ้นจะเพิ่มมากที่สุดจนถึงรอบที่ 6 เท่านั้นหลังจากนั้นค่าความถูกต้องจะคงที่ แล้วจะมาเพิ่มขึ้นอีกในรอบที่ 10 แต่ว่าค่าที่เพิ่มขึ้นนั้นเพิ่มขึ้นไม่มากเมื่อเทียบกับรอบที่ 3 ดังนั้นจึงสรุปว่าจำนวนรอบของการปรับปรุงตัวเวกเตอร์นั้นจะทำการปรับปรุงเพียงแค่ 3 รอบเท่านั้นซึ่งถ้าหากทำการปรับปรุงมากกว่านี้จะทำให้เสียเวลาในการค้นหาเป็นอย่างมาก โดยได้ผลที่ไม่แตกต่างกันมากเท่าใดนัก

บทที่ 5
สรุปผลการดำเนินงาน

โครงงานนี้พัฒนาขึ้นเพื่อสร้างแอพพลิเคชั่นที่ใช้ค้นหาไฟล์วีดิโอที่กระจายอยู่ในฐานข้อมูลในเครื่องคอมพิวเตอร์หลายเครื่องโดยใช้วีดิโอตัวอย่าง และใช้โมเดลการปรับตัวอัตโนมัติเพื่อให้ได้ผลของการค้นหาที่มีประสิทธิภาพมากขึ้น แอพพลิเคชั่นที่พัฒนาถูกพัฒนามาจากภาษาจาวา โดยนำไลบรารีต่างๆ ของจาวามาช่วยในการพัฒนาไม่ว่าจะเป็น JMF (Java Media Framework) , Java socket , Swing Application Framework และอื่น ๆ ในส่วนของการทำดัชนีไฟล์วีดิโอต่างๆ เพื่อทำการค้นหานั้นจะใช้หลักการทำดัชนี AVI (Adaptive Video Indexing) จากวิทยานิพนธ์ "อัลกอริทึม AVI สำหรับการทำดัชนีวีดิโอ" โดยแอพพลิเคชั่นจะนำเวกเตอร์ที่ได้จากการทำดัชนี AVI ที่ได้จากวิทยานิพนธ์ดังกล่าวมาใช้ค้นหา และเปรียบเทียบวีดิโอตามหลักการ Cosine Similarity เพื่อให้ได้วีดิโอผลลัพธ์ที่ใกล้เคียงกับวีดิโอที่ใช้ค้นหามากที่สุด โดยที่เวกเตอร์ที่นำมาเปรียบเทียบนั้นจะสามารถปรับปรุงให้ได้เวกเตอร์ใหม่แบบอัตโนมัติ ซึ่งเวกเตอร์ใหม่นี้มีประสิทธิภาพในการใช้ค้นหามากขึ้น โดยประสิทธิภาพในการค้นหานั้นดูได้จากค่า recall ที่มีการพัฒนาเพิ่มขึ้นในแต่ละรอบการทำงาน

5.1 ผลการดำเนินงาน

5.1.1 ตัวแอพพลิเคชั่นสามารถนำไฟล์วีดิโอที่ต้องการมาค้นหาไฟล์วีดิโอที่ถูกเก็บแบบกระจายได้

5.1.2 ไฟล์วีดิโอที่ค้นหาได้มีความใกล้เคียงกับวีดิโอที่ใช้ค้นหา

5.1.3 ไฟล์วีดิโอที่ค้นหาได้ในแต่ละรอบมีแนวโน้มของการได้ไฟล์วีดิโอที่เกี่ยวข้องเพิ่มขึ้นเนื่องมาจากการปรับปรุงเวกเตอร์ที่ใช้ค้นหา

5.2 ปัญหาที่พบในการทำโครงงานวิจัย

5.2.1 ในการทำแอพพลิเคชั่นหลายคนจะเกิดความยุ่งยากเมื่อโปรแกรมย่อยที่ได้จากแต่ละคนมารวมกัน

5.2.2 การใช้ JMF ไม่สามารถใช้เล่นไฟล์มีเดียได้ทุกชนิด และบางครั้งมีความผิดพลาดเมื่อใช้เล่นไฟล์วีดิโอหลังจากการค้นหา

5.2.3 เนื่องจากการทำแอพพลิเคชั่นไม่ได้ถูกออกแบบไว้อย่างละเอียดจึงทำให้มีการแก้ไขโครงสร้างโปรแกรมหลายครั้ง

5.2.4 แอพพลิเคชั่นที่ทำการพัฒนาไม่สามารถดึง key frame ของไฟล์วีดิโอแต่ละไฟล์ออกมาแบบอัตโนมัติได้ ผู้ใช้ต้องทำการบันทึกภาพ key frame จากไฟล์วีดิโอแต่ละไฟล์เอง ซึ่งทำให้เสียเวลามากเมื่อมีจำนวนไฟล์วีดิโอจำนวนมาก

5.3 ข้อเสนอแนะ

5.3.1 ในการปรับปรุงค่าเวกเตอร์ตามสมการควรปรับค่าตัวแปรต่างๆ (Upper threshold , Lower threshold , Beta , Gamma) ให้อยู่ในช่วงที่เหมาะสม

5.3.2 บางค่าเวกเตอร์อาจคล้ายกันแต่ไม่ใช่เรื่องเดียวกัน จะมีผลให้ผลลัพธ์จากการค้นหามีความคลาดเคลื่อน

5.3.3 ในการเขียนโปรแกรมหลายคนควรมีการออกแบบที่ถูกต้อง เป็นแบบแผนที่แน่นอน และอาจมีการใช้โปรแกรมที่ช่วยให้การพัฒนาโปรแกรมแบบหลายคนมีความสะดวกมากขึ้น เช่น tortoise SVN เป็นต้น

5.3.4 ในการใช้ JMF ควรมีการพัฒนาให้สามารถเล่นไฟล์มีเดียได้หลายชนิดไม่ว่าจะเป็น AVI , mp4 , FLV , DAT เป็นต้น

5.3.5 ควรพัฒนาเพิ่มเติมในส่วนของการแสดงผลให้สามารถแสดง key frame ของไฟล์วีดิโอได้โดยอัตโนมัติ

5.3.6 ควรแสดงกราฟ False Positive และ False Negative เพื่อแสดงถึงความผิดพลาดในการค้นหา

5.4 แนวทางในการพัฒนาโครงงานวิจัย

5.4.1 พัฒนาแอพพลิเคชั่นให้สามารถใช้บนอินเตอร์เน็ตได้

5.4.2 พัฒนาส่วนเชื่อมต่อระหว่างส่วนการแบ่งช็อตวีดิโอ ส่วนการทำดัชนี และส่วนการค้นหา

5.4.3 พัฒนาให้แอพพลิเคชั่นมีความปลอดภัยมากขึ้น (Security)

5.4.4 พัฒนาแอพพลิเคชั่นให้สามารถปรับค่าตัวแปรต่างๆ (Upper threshold , Lower threshold , Beta , Gamma) ที่เหมาะสม โดยอัตโนมัติ

5.4.5 ปรับเปลี่ยนโมเดลการค้นหา และปรับตัวบนเครือข่ายเพียร์ทูเพียร์ในรูปแบบอื่น

เอกสารอ้างอิง

[1] J. Hill, *An Overview of Content-Based Video Retrieval*: Spring, 2008.

[2] S. P. F Idris, *Review of Image and Video Indexing Techniques*: Inc, 1997.

[3] S. P. F Idris, *Image/video indexing in the compressed domain*: Electrical and Computer Engineering in Canadian, 1996.

[4] D. o. C. S. a. Engineering, *Pattern recognition methods in image and video databases*. University Park: The Pennsylvania State University.

[5] S. R. JINGHUANG, MANDARMITRA,WEI-JINGZHUANDRAMINZABIH, *Spatial Color Indexing and Applications*. NewYork: Cornell University

[6] P. Muneesawang, L. Guan, *Multimedia Database Retrieval : A Human - Centered Approach*. New York: Springer, 2006.

[7] "Peer-to-Peer working group " in *http://www.P2Pwg.org*, 2002.

[8] J. D. Gradecki, *Mastering JXTA: Building Java Peer-to-Peer Applications*: Wiley Publishing.

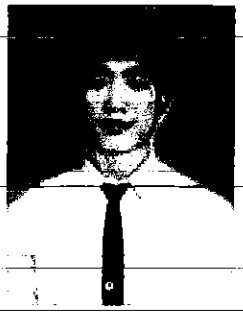
[9] D. Milojicic, Kalogeraki, V., Lukose, R., Nagaraja, K. 1, Pruyne, J., and B. Richard, Rollins, S. 2, Xu, Z. , *Peer-to-Peer Computing*. USA: HP Laboratories Palo Alto, 2002.

[10] M. Ellsworth, "The Buzz About Hive Networks: Putting Peer-to-Peer Computing to work," in *www.manyworlds.com*, 2001.

[11] A.P.Rajshekhar, "Socket Programming in Java," in *http://www.devarticles.com/c/a/Java/Socket-Programming-in-Java/*, 2007.

[12] B. A. Forouzan, *Data Communications and Networking* 4ed. New York: McGraw-Hill, 2007.

ประวัติผู้เขียนโครงงาน



ชื่อ นายกฤษฎา ศตกาญจน์

ภูมิลำเนา 45/1 หมู่ 1 ต.บุญเรือง อ.เชียงของ จ.เชียงราย 57140

ประวัติการศึกษา

- จบมัธยมจากโรงเรียนบุญเรืองวิทยาคม จังหวัดเชียงราย
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
  มหาวิทยาลัยนเรศวร

E-mail: akiracpe9@gmail.com

ชื่อ นายมงคล อักโข

ภูมิลำเนา 251/25 หมู่ 4 แขวงตลาดบางเขน เขตหลักสี่
จ.กรุงเทพมหานคร 10210

ประวัติการศึกษา

- จบมัธยมจากโรงเรียนดอนเมืองจาตุรจินดา จังหวัดกรุงเทพมหานคร
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
  มหาวิทยาลัยนเรศวร

E-mail: bnw_mong4@hotmail.com

ชื่อ นายแสวงชัย สาโรจน์

ภูมิลำเนา 69/70 ถ.ดาวดึงส์ ต.ปากน้ำโพ อ.เมือง จ.นครสวรรค์ 60000

ประวัติการศึกษา

- จบมัธยมจากโรงเรียนนครสวรรค์ จังหวัดนครสวรรค์
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
  มหาวิทยาลัยนเรศวร

E-mail: sawangchais@live.com