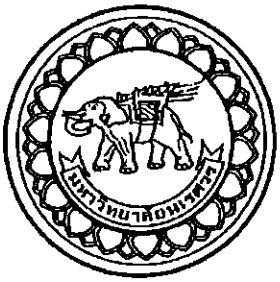


อัลกอริทึม AVI สำหรับการทำดัชนี VDO

AVI Algorithm for VDO Indexing

นางสาวณัฐกานต์ ประเสริฐสังห์ รหัส 47380020

นางสาวนริศรา สิงห์เชื้อ รหัส 47380025

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2550

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน อัลกอริทึม AVI สำหรับการทำดัชนี VDO

ผู้ดำเนินโครงงาน นางสาวณัฐกานต์ ประเสริฐสังห์ รหัส 47380020

นางสาวนริศรา สิงห์เชื้อ รหัส 47380025

อาจารย์ที่ปรึกษา ดร. ไพศาล มุณีสว่าง

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2550

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ดร. ไพศาล มุณีสว่าง)

.....กรรมการ

(ดร. พนมขวัญ ริยะมงคล)

.....กรรมการ

(ดร. ชัยรัตน์ พินทอง)

หัวข้อโครงงาน	อัลกอริทึม AVI สำหรับการทำดัชนี VDO
ผู้ดำเนินโครงงาน	นางสาวณัฐกานต์ ประเสริฐสังห์ รหัส 47380020
	นางสาวนริศรา สิงห์เชื้อ รหัส 47380025
อาจารย์ที่ปรึกษา	ดร.ไพศาล มุณีสว่าง
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2550

.....

บทคัดย่อ

โครงงานนี้ได้กล่าวถึงโปรแกรมที่มีความสามารถในการเก็บค่าดัชนีของไฟล์วิดีโอ โดยใช้หลักการของ Vector Quantization ด้วยวิธีการ LBG Design Algorithm โดยพิจารณาจากข้อมูลฮีสโตแกรมของเฟรมรูปภาพทุกภาพจากทุกช็อตวิดีโอ นำค่าที่ได้จากการ Quantize ไปใช้ในการจัดกลุ่มให้กับเฟรมรูปภาพทั้งหมด เพื่อเก็บความถี่ของจำนวนเฟรมทั้งหมดในหนึ่งกลุ่มทีละหนึ่งช็อตวิดีโอ จะทำให้ได้ดัชนีของช็อตนั้นๆ ซึ่งนั่นก็หมายถึง การเก็บดัชนีนั่นเอง โดยที่โปรแกรมสามารถทำได้ 2 รูปแบบ คือ การเก็บดัชนีแบบ One Codevector (ŋ=1) และการเก็บดัชนีแบบ Two Codevector (ŋ=2) อีกทั้งยังมีความสามารถในการค้นหาไฟล์วิดีโอ ซึ่งในที่นี้ใช้การอ้างอิงไฟล์ต้นแบบและไฟล์ที่ได้จากการค้นหาเป็นแบบช็อต โดยมีตัวเลือกในการค้นหาให้แก่ผู้ใช้ 2 แบบ คือ การค้นหาโดยใช้ดัชนีแบบ One Codevector (ŋ=1) และแบบ Two Codevector (ŋ=2) โครงงานนี้พัฒนาด้วย Visual Studio 2005 และใช้ Microsoft Access 2003 ในการจัดการฐานข้อมูล

ผลที่ได้จากการทำโครงงานนี้ คือ เราสร้างโปรแกรมคอมพิวเตอร์ที่ใช้ได้ง่าย สำหรับการเก็บดัชนีของไฟล์วิดีโอ รวมไปถึงระบบการค้นหาไฟล์วิดีโอที่มีความคล้ายหรือใกล้เคียงกันโดยอ้างอิงเป็นช็อต ซึ่งผู้ที่สนใจสามารถนำไปพัฒนาต่อเพื่อเพิ่มความสามารถให้กับโปรแกรมได้

Project Title	AVI Algorithm for VDO Indexing.
Name	Miss Nutthakan Prasertsung ID. 47380020
	Miss Narissara Singchua ID. 47380025
Project Advisor	Paisarn Muneesawang, Ph.D.
Major	Computer Engineering.
Department	Electrical and Computer Engineering.
Academic Year	2550

.....

ABSTRACT

This project studies the technique for indexing video files, employing the principle of Vector Quantization by LBG Design Algorithm. Given histogram data of every frame and every shot in a database, the program uses vector quantization to make groups of all frames, and for stores the frequency of frames in each shots indexing. There are two methods of this program: one layer style indexing and two layer style indexing. Furthermore, this program is available for searching similar video files in the database. This system was conducted by Visual studio 2005 and Microsoft Access 2003.

As a result from this project, we created a computer program that is easy to use for video indexing and searching system of similar video in shot level. Moreover, the system can be used as a prototype for future development.

กิตติกรรมประกาศ

โครงงานวิศวกรรมคอมพิวเตอร์ สำเร็จลุล่วงได้ด้วยดีเนื่องด้วยความอนุเคราะห์จากท่านอาจารย์ที่ปรึกษาโครงงานคือ ดร.ไพศาล มุณีสว่าง รวมถึงท่านคณะกรรมการอีกทั้ง 2 ท่านคือ ดร.พนมขวัญ ริยะมงคล และ ดร.ชัยรัตน์ พินทอง ที่ได้ให้แนวคิดและคำแนะนำ ตลอดจนได้สละเวลาอันมีค่าเพื่อตรวจสอบและแก้ไขข้อบกพร่องต่างๆ อีกทั้งยังสอนให้รู้จักการวางแผนการทำงาน

ในโอกาสนี้ทางคณะผู้จัดทำโครงงานจึงขอขอบพระคุณทุกๆท่านที่มีส่วนร่วมในการทำโครงงานนี้ตลอดจนผู้เขียน ผู้คิดค้นทฤษฎีต่างๆ ที่โครงงานฉบับนี้ได้นำความรู้ที่ได้มาพัฒนาระบบ ทำให้โครงงานนี้สำเร็จลุล่วงไปด้วยดี

นางสาวณัฐกานต์ ประเสริฐสังห์
นางสาวนริศรา สิงห์เชื้อ

สารบัญ

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงการ

เนื่องจากในปัจจุบันเทคโนโลยีทางด้านมัลติมีเดีย ถูกพัฒนาให้อำนวยความสะดวกกับมนุษย์และมีเพิ่มมากขึ้นอย่างเห็นได้ชัด โดยเฉพาะมัลติมีเดียด้านระบบดิจิตอลวิดีโอถือว่ามีบทบาทต่อความต้องการของมนุษย์ในยุคปัจจุบันเป็นอย่างมาก ซึ่งมีการผลิตออกสู่ท้องตลาดในหลายรูปแบบ แต่ในปัจจุบันไฟล์ประเภทความบันเทิงได้รับความนิยมอย่างแพร่หลาย ไฟล์ดังกล่าวจัดเก็บในรูปแบบที่หลากหลายแตกต่างกัน ซึ่งไฟล์นามสกุล .avi เป็นรูปแบบที่นิยมนำมาใช้ในการจัดเก็บไฟล์วิดีโอมากเป็นอันดับต้น ทางผู้จัดทำจึงได้เล็งเห็นประโยชน์ด้านอื่นๆ จากไฟล์วิดีโอดังกล่าว ที่นอกเหนือจากการนำไปรับชมและรับฟัง ก็คือ เราสามารถนำไฟล์วิดีโอดังกล่าวมาทำการเก็บดัชนีไว้เพื่อที่สามารถจะนำไปใช้ในระบบการวิเคราะห์หรือประโยชน์ในด้านอื่นๆ ต่อไปได้ แต่ที่ผู้จัดทำเกิดความสนใจที่จะทำการวิจัยคือระบบการค้นหาโดยใช้ดัชนี ซึ่งไฟล์วิดีโอหนึ่งไฟล์จะประกอบไปด้วยช็อตจำนวนหลายช็อต เราสามารถนำช็อตจากไฟล์วิดีโอดังกล่าวมาประยุกต์ใช้ในระบบการค้นหาได้ โดยการเก็บค่าดัชนีของทุกช็อตจากหลายๆ ไฟล์วิดีโอ เพื่อนำดัชนีดังกล่าวมาใช้ในการค้นหาช็อตที่มีความเหมือนหรือใกล้เคียงกัน อีกทั้งเมื่อผู้ใช้ต้องการเริ่มรับชมจากบางฉากของไฟล์วิดีโอ ที่ไม่ใช่ฉากแรกของไฟล์วิดีโอนั้น ก็สามารถค้นหาโดยใช้ดัชนีดังกล่าวได้

และจากการได้ศึกษาค้นคว้าข้อมูล การเก็บค่าดัชนีด้วยกระบวนการ LBG Vector Quantization เป็นวิธีการหาผลลัพธ์ที่มีประสิทธิภาพมากในระดับหนึ่ง เนื่องจากกระบวนการดังกล่าวมีวิธีการวิเคราะห์ค่าสีทีละพิกเซลเพื่อทำการเก็บข้อมูลฮิสโตแกรมจากทุกเฟรมในหนึ่งช็อตและในทุกๆ ไฟล์วิดีโอ จึงได้ค่าสี RGB (ข้อมูลฮิสโตแกรม) ที่ละเอียด มีประสิทธิภาพต่อการนำไปใช้ในการวิเคราะห์ต่อไปได้

1.2 จุดประสงค์ของโครงการ

1.2.1 เพื่อสร้างโปรแกรมจำลองการเก็บดัชนีและการค้นหาไฟล์วิดีโอ

1.2.2 เพื่ออำนวยความสะดวกในการค้นหาไฟล์วิดีโอ

1.2.3 เพื่อค้นคว้าและพัฒนาระบบในการค้นหาไฟล์วิดีโอจากดัชนีสี

1.2.4 เพื่อเป็นตัวเลือกอีกตัวเลือกหนึ่งในการค้นหาไฟล์วิดีโอ ให้แก่ผู้ที่สนใจ

1.3 ผลที่คาดว่าจะได้รับ

1.3.1 ได้ซอฟต์แวร์ที่อำนวยความสะดวกในการค้นหาไฟล์วิดีโอให้ง่ายขึ้น

1.3.2 ดึงความสนใจของผู้ที่ศึกษาเนื้อหาหลักการและทฤษฎีที่เกี่ยวข้องกับโครงงานให้มากขึ้น

1.3.3 ได้ทบทวนเนื้อหาที่ได้ศึกษามาแล้ว มาประยุกต์ใช้กับโครงงาน

1.3.4 เป็นแนวทางในการค้นคว้าและพัฒนาเทคนิคต่างๆ เพื่อให้ได้องค์ความรู้ใหม่ทางด้านการใช้คอมพิวเตอร์แก่นักศึกษาทั้งในระดับปริญญาตรีและผู้ที่สนใจทางด้าน Video Indexing

1.4 ขอบเขตของโครงการ

การผลิตซอฟต์แวร์ที่ใช้ในการเก็บดัชนีสีของไฟล์วิดีโอ โดยที่สามารถโหลดไฟล์วิดีโอเข้าสู่โปรแกรมได้ทีละหลายๆไฟล์ โดยไฟล์วิดีโอดังกล่าวต้องผ่านกระบวนการในการแบ่งไฟล์ออกเป็นช็อตก่อนที่จะเข้าสู่กระบวนการ Vector Quantization เพื่อทำการแบ่งกลุ่มและหาดัชนีสีจากทุกไฟล์วิดีโอที่ได้ทำการโหลดเข้าสู่โปรแกรมซึ่งจะนำไปใช้ในระบบการค้นหาต่อไป ในส่วนของระบบการค้นหาอ้างอิงไฟล์วิดีโอเป็นช็อตเช่นกัน เพื่อทำการหาช็อตที่มีความใกล้เคียงหรือเหมือนกับช็อตที่นำมาอ้างอิง

1.4.1 ผลิตซอฟต์แวร์เพื่อใช้ในการเก็บดัชนีสีของไฟล์วิดีโอหลายๆ ไฟล์ รวมถึงระบบในการค้นหาด้วย ซึ่งใช้ได้กับไฟล์วิดีโอนามสกุล .avi เท่านั้น

1.4.2 ในขั้นตอนการแบ่งไฟล์วิดีโอออกเป็นช็อต จะใช้โปรแกรมสำเร็จรูปในการแบ่งไฟล์ คือ โปรแกรม Video Editor 7.0 ซึ่งไม่ได้เขียนโปรแกรมในส่วนของการแบ่งช็อต

1.4.3 ช็อตที่ได้จากการแบ่งไฟล์วิดีโอเก็บในรูปแบบของ .avi เช่นเดียวกับไฟล์วิดีโอต้นแบบ และไฟล์ภาพนิ่งที่ได้จากการแยกออกเป็นเฟรมเก็บในรูปแบบ bitmap

1.4.4 ไฟล์รูปภาพที่ผ่านขั้นตอนการควอนไตซ์เซชั่นแล้วจะไม่สามารถโหลดภาพต้นแบบกลับมาได้

1.4.5 เพื่อใช้ในระบบการสืบค้นข้อมูลประเภทวิดีโอ

1.4.6 ในส่วนของการเก็บดัชนี มีวิธีการเก็บ 2 แบบ คือ การเก็บดัชนีแบบ one codevector และการเก็บดัชนีแบบ two codevector

1.4.7 ในระบบการสืบค้น มีวิธีการในการค้นหาอยู่ 2 แบบ คือ แบบใช้ดัชนีแบบ one codevector และแบบใช้ดัชนีแบบ two codevector

1.5 สถานที่ที่ใช้ในการดำเนินการโครงการและรวบรวมข้อมูล

1.5.1 สำนักหอสมุดมหาวิทยาลัยนเรศวร

1.5.2 หอพัก ทิตินันท์

1.6 อุปกรณ์ที่ใช้ในการดำเนินโครงการ

1.6.1 เอกสารและตำราประกอบการทำโครงการ

1.6.2 อุปกรณ์การเก็บข้อมูล, การประเมินผล และผลิตผลงานโครงการ

- ซอฟต์แวร์ที่ใช้ในการทำงาน (Video Editor 7.0, Visual Studio 2005, Microsoft Access)
- เครื่องพิมพ์พร้อมหมึกพิมพ์
- เอกสาร กระดาษ อุปกรณ์สำนักงาน

1.7 ระยะเวลาในการดำเนินงานโครงการ

ตารางที่ 1.1 ระยะเวลาในการดำเนินงานโครงการ

ขั้นที่	รายละเอียด	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	หมายเหตุ
1.	เก็บรวบรวมข้อมูล หลักการ ทฤษฎี และวิธีการที่จะนำมาใช้	←	—	→					
2.	ศึกษา หลักการ และทฤษฎีที่ได้มาเพื่อประยุกต์ใช้กับโปรแกรม		←	—	→				
3.	วิเคราะห์ข้อมูล, ออกแบบระบบ			←	—	→			
4.	เขียนโปรแกรม			←	—	—	→		
5.	ทดสอบและแก้ไขโปรแกรม					↔	↔		
6.	สรุปผล					↔	↔		
7.	จัดทำรายงาน					↔	↔		
8.	นำเสนอโครงการ						↔		

1.8 งบประมาณที่ใช้

ค่าใช้จ่ายทั้งหมดเป็นเงินทั้งหมด 2,000 บาท จำแนกออกเป็น 3 ประเภท ดังนี้

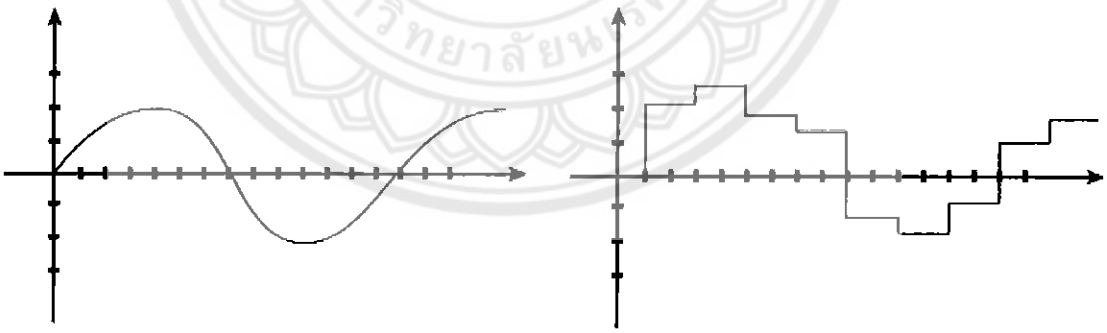
1.8.1 กระดาษที่ใช้ทำเอกสาร	500	บาท
1.8.2 หนังสือประกอบการทำโครงการ	1,300	บาท
1.8.3 อุปกรณ์และวัสดุอื่นๆ	200	บาท
รวมเป็นเงินทั้งสิ้น	2,000	บาท

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 ความรู้พื้นฐานสำหรับการประมวลผลภาพดิจิตอล [11]

Digital Image คือภาพที่เก็บอยู่ในรูปแบบของดิจิตอล ภาพที่เรามองเห็นด้วยสายตาทั่วไปนั้นเป็นภาพในลักษณะสามมิติ คือ มีมิติของความกว้าง ความยาว และความลึกหรือความสูง ส่วนภาพถ่ายที่เห็นกันอยู่ในโทรทัศน์หรือเครื่องคอมพิวเตอร์นั้น เป็นการแปลงภาพจากสามมิติมาเป็นสองมิติ โดยการแปลงสัญญาณไฟฟ้าในรูปแบบอนาล็อก ยกตัวอย่างเช่น ในกล้องวิดีโอ เซนเซอร์ที่อยู่ในกล้องจะทำการสแกนหรือวัดผลรวมความเข้มแสงที่จุดต่างๆ ไปตามแนวสแกนที่เรียกว่า Raster Scan การสแกนแบบนี้จะมีทิศทางจากบนลงล่าง และจากซ้ายไปขวา ภาพที่ได้จากการสแกนนั้นจะเป็นภาพแบบต่อเนื่อง (Continuous) ด้วยความเร็วทั่วไปที่ 24 ภาพต่อวินาที ดังแสดงในรูปที่ 2.1 เช่นเดียวกันนี้ในเครื่องรับภาพวิดีโอก็จะรับภาพที่ได้มาจากเครื่องถ่ายวิดีโอ และแสดงผลโดยเริ่มจากบนลงล่างและจากซ้ายไปขวาเช่นเดียวกัน

แต่ภาพที่ได้มาจากระบบอนาล็อกนั้นยังเป็นภาพแบบต่อเนื่อง ที่ยังไม่สามารถนำมาใช้ในการประมวลผลได้ ต้องมาทำการแปลงให้เป็นภาพเชิงตัวเลขเสียก่อนด้วยวิธีการ Digitization ซึ่งเป็นการแปลงฟังก์ชันต่อเนื่อง $f(x,y)$ ให้เป็นฟังก์ชันไม่ต่อเนื่อง $g(x,y)$ เพื่อให้นำมาประมวลผลด้วยคอมพิวเตอร์ได้ ดังแสดงในรูปที่ 2.1



ภาพแบบต่อเนื่อง (Continuous) ภาพเชิงตัวเลขจากวิธีการ Digitization

รูปที่ 2.1 การแปลงสัญญาณไฟฟ้าในรูปแบบอนาล็อก

2.1.1 ความหมายของการประมวลผลภาพดิจิตอล

การประมวลผลภาพ หมายถึง การเรียกใช้ขั้นตอนหรือกรรมวิธีใดๆ มากระทำกับภาพ โดยมีวัตถุประสงค์เพื่อปรับปรุงคุณภาพของภาพ ให้ได้ภาพใหม่ที่มีคุณสมบัติตามต้องการ เช่น ความคมชัด

หรือการประหยัดพื้นที่ในการเก็บข้อมูล หรือใช้สำหรับการประมวลผลในระดับสูง เช่นการจดจำรูปร่างลักษณะได้อย่างแม่นยำ โดยทั่วไปแล้ววัตถุประสงค์ของการประมวลผลภาพก็คือ

- Image Processing : Image in → Image out
  วิธีนี้จะใช้กระบวนการประมวลผลภาพ เพื่อทำให้ได้ภาพออกมา เช่น การตกแต่งภาพด้วยโปรแกรม Photoshop เป็นต้น
- Image Analysis : Image in → Measurements out
  วิธีนี้จะใช้กระบวนการประมวลผลภาพ เพื่อทำให้ได้ค่าการวัดออกมา เช่น การวัดขนาดในงานอุตสาหกรรม เป็นต้น
- Image Understanding : Image in → High-Level Description out
  วิธีนี้จะใช้กระบวนการประมวลผลภาพ เพื่อให้ได้ผลลัพธ์ออกมาเป็นความหมาย ตัวอย่างของ High-Level Description เช่น การจดจำตัวอักษร (Optical Character Recognition : OCR) เป็นต้น

การประมวลผลภาพด้วยคอมพิวเตอร์ สามารถทำได้โดยนำภาพที่ได้มาจากกล้องหรือ Image Source ต่างๆ ซึ่งเป็นสัญญาณอนาล็อก แล้วนำมาแปลงเป็นสัญญาณดิจิตอลที่มีลักษณะเป็นรหัสเชิงตัวเลข 0, 1 ที่สามารถใช้รูปแบบทางคณิตศาสตร์เข้ามาช่วยในการคำนวณและการประมวลผลข้อมูลภาพด้วยคอมพิวเตอร์ได้ต่อไป

การประมวลผลภาพ แบ่งได้เป็น 2 ระดับ คือ

1. การประมวลผลภาพระดับต่ำ (Low Level Image Processing)

เป็นการประมวลผลขั้นแรกสุดก่อนที่จะนำไปสู่การประมวลผลภาพระดับสูงต่อไป นั่นคือหลังจากที่เราได้ภาพมา ภาพที่ได้ก็จะประกอบด้วยองค์ประกอบต่างๆ มากมาย รวมถึงสิ่งที่เราไม่ต้องการด้วย ในที่นี้เราจะเรียกว่าสัญญาณรบกวน (Noise) ซึ่งทำให้ภาพที่ได้มีคุณภาพไม่ดี ยังไม่สามารถที่จะนำไปใช้ประมวลผลได้ ดังนั้น การประมวลผลภาพในระดับต่ำจึง ประกอบไปด้วยการกำจัดสัญญาณรบกวน การทำภาพให้ชัด (High Pass Filter) การหาขอบภาพ (Edge Detection) การแปลง Binary Image การแบ่งแยกรูปร่างวัตถุ (Image Segmentation) เป็นต้น เพื่อหาค่าตัวแปรต่างๆ มาอธิบายข้อมูลภาพ และมีวัตถุประสงค์ที่จะนำตัวแปรเหล่านี้มาใช้ในการประมวลผลภาพในระดับสูงต่อไป

2. การประมวลผลภาพระดับสูง (High Level Image Processing)

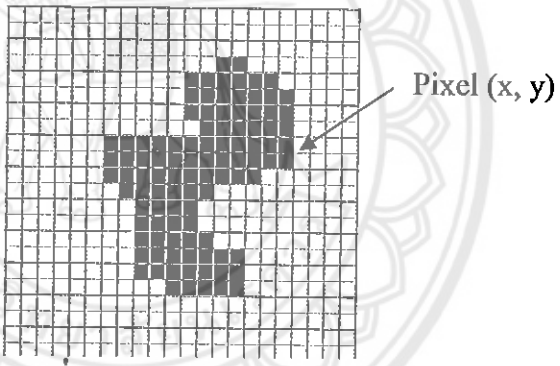
เป็นการทำให้คอมพิวเตอร์รู้จักและเข้าใจภาพนั้นได้ เช่น การจดจำใบหน้าคน หรืออาจจะเป็นการจดจำตัวอักษร เป็นต้น ความแตกต่างของการประมวลผลภาพระดับต่ำและระดับสูง คือ ข้อมูลที่นำมาใช้ในการประมวลผลระดับต่ำจะใช้ค่าความสว่าง หรือ ความเข้มของแสงโดยตรง ส่วนการรประมวลผลภาพระดับสูง ข้อมูลที่นำมาใช้ในการประมวลผลจะถูกแสดงในรูปแบบของสัญลักษณ์ โดยสัญลักษณ์เหล่านี้จะแสดงถึงสิ่งต่าง ๆ ที่อยู่ในภาพ และการใช้ตัวแปรที่ได้จากการประมวลผลภาพ

ระดับต่ำมาอธิบายถึงสัญลักษณ์เหล่านี้ การประมวลผลภาพระดับสูงนั้นส่วนใหญ่มักใช้ทฤษฎีต่างๆ เข้ามาใช้เป็นตัวช่วยในการทำงาน หรือเป็นหัวใจของโปรแกรม เช่น Fuzzy Logic, Neural Network

อย่างที่กล่าวไปแล้วว่า การประมวลผลภาพระดับสูงจำเป็นต้องใช้ข้อมูลที่ได้มาจากการประมวลผลภาพระดับต่ำ ดังนั้นจะเห็นได้ว่าการประมวลผลภาพระดับต่ำ มีความสำคัญมากสำหรับการทำให้คอมพิวเตอร์รู้จักและเข้าใจภาพได้

2.1.2 การแทนภาพด้วยข้อมูลแบบดิจิตอล

จากที่ได้กล่าวไปแล้วในตอนต้น ข้อมูลภาพแบบดิจิตอลเป็นภาพที่ถูกดัดแปลงมาจากภาพแบบต่อเนื่อง ให้อยู่ในรูปตัวเลข ด้วยวิธีการ Digitization โดยภาพ Analog Image จะถูกแบ่งให้เป็นพื้นที่สี่เหลี่ยมเล็ก ๆ ที่เรียกว่า พิกเซล (Pixel) ดังรูปที่ 2.2 โดยแต่ละพิกเซลจะใช้ (x, y) ในการระบุตำแหน่ง การแสดงข้อมูลภาพดิจิตอลสามารถอธิบายได้ด้วย เมทริกซ์ (M, N) และให้จุดต่างๆ ที่อยู่ในเมทริกซ์เป็นจุดที่พิกัด (x, y) ใด ๆ เป็นส่วนประกอบของภาพ เมื่อเราเปรียบเทียบระหว่างภาพและ Pixel Metrix ดังรูปที่ 2.2 จะเห็นว่าจุดกำเนิดของภาพจะอยู่ที่มุมล่างซ้าย แต่จุดกำเนิดของพิกเซลจะอยู่ที่มุมบนซ้าย ซึ่งจะเป็นลักษณะการประมวลผลภาพในกราฟิกของคอมพิวเตอร์



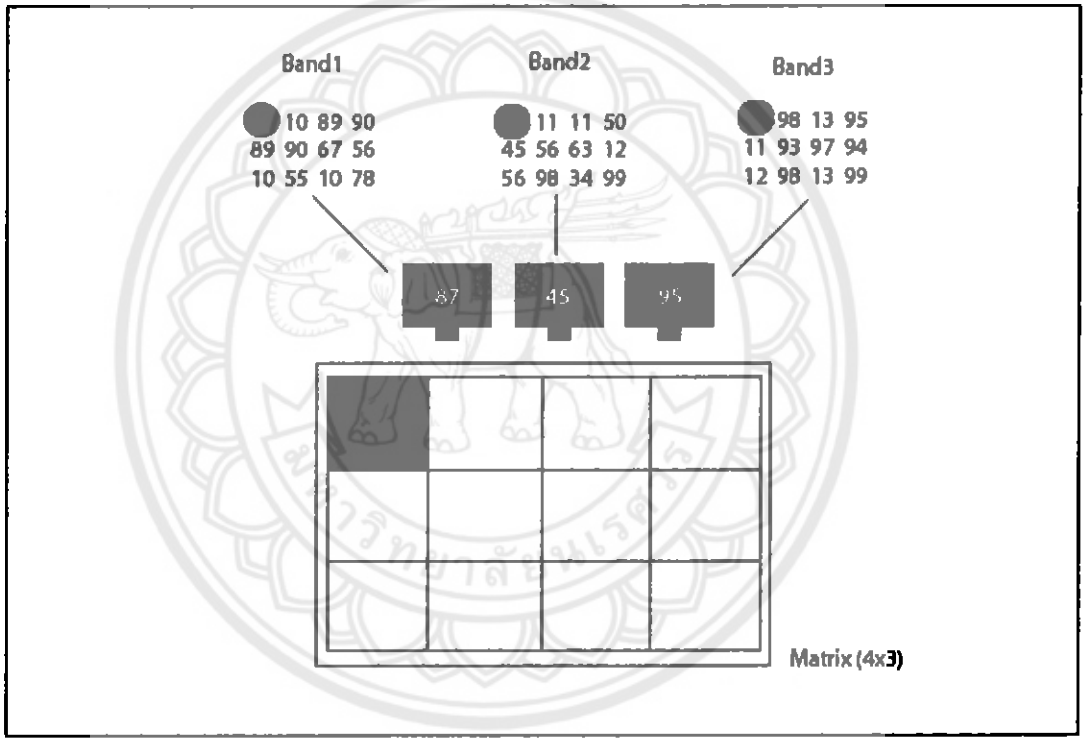
รูปที่ 2.2 แสดงค่าของพิกเซล

จากรูปที่ 2.2 ค่าของพิกเซล หรือ ฟังก์ชัน (x, y) ณ จุดใด ๆ จะแสดงได้ด้วยค่าของความเข้มแสงซึ่งอาจแบ่งได้หลายระดับ ถ้ามี 2 ระดับ ก็จะเป็นแค่ 0 กับ 1 จากรูป จุดต่าง ๆ ที่แสดงอยู่ในพิกัดนี้ก็คือ พิกเซลหรือ Picture Element ซึ่งก็คือ ความสว่างหรือค่า Luminance (L) ของภาพ ถ้าภาพนั้นเป็นภาพขาวดำ ขนาด 8 บิต จะมีค่า L เท่ากับ 28 หรือเท่ากับ 256 คือตั้งแต่ระดับ 0 จนถึง 255 บางครั้งค่าความสว่าง (L) อาจหมายถึงระดับความละเอียดของภาพ (Image Resolution) ถ้าพิกเซล เป็นภาพขาวดำจะอ่านค่าภาพดิจิตอลในรูปแบบเมทริกซ์ 2 มิติ ขนาด (M, N) ได้ดังรูปที่ 2.3

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & . & f(0,n-1) \\ f(1,0) & f(1,1) & . & f(1,n-1) \\ . & . & . & . \\ f(m-1,0) & f(m-1,1) & . & f(m-1,n-1) \end{bmatrix}_{(M*N)}$$

รูปที่ 2.3 รูปแบบเมทริกซ์ 2 มิติ ขนาด (M,N)

โดยที่ค่า $f(x,y)$ จะอยู่ในช่วง 0 ถึง 255 สมมติว่าอ่านค่าพิกเซล จากภาพหนึ่งได้ $f(x,y)$ เท่ากับ 10 แสดงว่าจุดพิกเซลนั้นมีความสว่างน้อยหรือค่อนข้างจะดำ ถ้าอ่านได้เป็น 255 แสดงว่าจุดพิกเซลนั้นมีความสว่างมาก หรือเป็นสีขาว



รูปที่ 2.4 ตัวอย่างแสดงค่าพิกเซลของเมทริกซ์ 2 มิติ ขนาด (4 x 3)

จากรูปที่ 2.4 เป็นตัวอย่างแสดงค่าพิกเซลของเมทริกซ์ 2 มิติ ขนาด (4 x 3) จะช่วยทำให้เข้าใจการแสดงค่าพิกเซลในเมทริกซ์มากขึ้น เริ่มต้นด้วยพิกัด $f(x,y)=f(0,0)$ ค่าของพิกเซลที่ได้จะเป็นการผสมสีกันระหว่างค่าของแม่สีทั้งสาม ซึ่งได้แก่ สีแดง สีเขียว และสีน้ำเงิน

2.1.3 ภาพที่นำมาใช้ในการประมวลผล

โดยทั่วไปแล้ว ภาพที่นำมาใช้ในการประมวลผลภาพดิจิตอลนั้น แบ่งออกเป็นสองประเภทหลัก ๆ คือ

- ภาพเคลื่อนไหว

ความจริงแล้วภาพเคลื่อนไหวก็คือภาพนิ่งที่นำมาแสดงต่อ ๆ กันแบบต่อเนื่อง จะต้องใช้รูปภาพอย่างน้อย 24 รูปต่อหนึ่งวินาที เนื่องจากสายตาของคนเราเมื่อนำภาพนิ่งมาฉายติดต่อกันมากกว่า 24 รูปต่อหนึ่งวินาทีแล้วก็จะมองว่าภาพนั้นเป็นภาพเคลื่อนไหว เพราะสายตาเราแยกไม่ออกเนื่องจากมีความเร็วมากเกินไป แต่หากนำภาพนิ่งมาฉายติดต่อกันน้อยกว่า 24 รูปต่อหนึ่งวินาทีแล้ว เราจะมองเห็นว่าภาพนั้นไม่ต่อเนื่อง

- ภาพนิ่ง

ภาพที่นำเข้ามาในการประมวลผลในคอมพิวเตอร์นั้น ถ้าในระบบ RGB ก็จะใช้ความเข้มแสงสี แดง เขียว และน้ำเงิน ความหมายของภาพนิ่งก็คือมีอยู่เพียงภาพเดียว (ถ้าเป็นภาพเคลื่อนไหวก็จะประกอบไปด้วยภาพนิ่งหลาย ๆ ภาพ) ภาพนิ่งที่นำมาใช้ก็มีอยู่หลาย Format ไม่ว่าจะเป็น .bmp หรือ .jpg เป็นต้น ส่วนการเลือกใช้นั้นก็แล้วแต่ความเหมาะสม ส่วนใหญ่จะเป็น .bmp เพราะไม่ต้องมาถอดรหัสก่อน เนื่องจากภาพที่เป็น .jpg นั้นมีการบีบอัดภาพให้มีขนาดเล็ก ดังนั้นหากจะนำมาใช้ก็ต้องคลายข้อมูลออกมาก่อนที่จะนำภาพไปประมวลผลต่อไป

2.2 ทฤษฎีสี [1]

สีเป็นสิ่งที่มีความสำคัญอย่างหนึ่งในการดำรงชีวิต ซึ่งมนุษย์รู้จักสามารถนำมาใช้ให้เกิดประโยชน์ในชีวิตประจำวันมาตั้งแต่สมัยดึกดำบรรพ์ในอดีตกาล มนุษย์ได้ค้นพบสีจากแหล่งต่างๆ จาก พืช สัตว์ ดิน และแร่ธาตุนานาชนิด จากการค้นพบสีต่างๆ เหล่านั้น มนุษย์ได้นำเอาสีต่างๆ มาใช้ประโยชน์อย่างกว้างขวาง โดยนำมาระบายลงไปบนสิ่งของภาชนะเครื่องใช้หรือระบายลงไปบนรูปปั้น รูปแกะสลักเพื่อให้รูปเด่นชัดขึ้นมีความเหมือนจริงมากขึ้นรวมไปถึงการใช้สีวาดลงไปบนผนังถ้ำ หน้าผา ก้อนหิน เพื่อใช้ถ่ายทอดเรื่องราวและทำให้เกิดความรู้สึกถึงพลังอำนาจที่มีอยู่เหนือสิ่งต่างๆ ทั้งปวง การใช้สีทาตามร่างกายเพื่อกระตุ้นให้เกิดความฮึกเหิม เกิดพลังอำนาจหรือใช้สีเป็นสัญลักษณ์ในการถ่ายทอดความหมายอย่างใดอย่างหนึ่ง

ในสมัยเริ่มแรก มนุษย์รู้จักใช้สีเพียงไม่กี่สี สีเหล่านั้นได้มาจากพืช สัตว์ ดิน แร่ธาตุต่างๆ รวมถึงขี้เถ้า เขม่าควันไฟ เป็นสีที่พบทั่วไปในธรรมชาติ ซึ่งได้นำมาถู ทา ต่อมาเมื่อทำการย่างเนื้อสัตว์ ไขมัน น้ำมัน ที่หยดจากการย่างลงสู่ดินทำให้ดินมีสีสันน่าสนใจ สามารถนำมาระบายลงบนวัตถุและติดแน่นทนนาน ดังนั้นไขมันนี้จึงได้ทำหน้าที่เป็นส่วนผสม (Binder) ซึ่งมีความสำคัญในฐานะเป็นสารชนิดหนึ่งที่เป็นส่วนประกอบของสี ทำหน้าที่เกาะติดผิวหน้าของวัสดุที่ถูกนำไปทาหรือระบาย นอกจากไขมันแล้วยังได้นำไข่ขาว ขี้ผึ้ง (Wax) น้ำมันลินสีด (Linseed) กาวและยางไม้ (Gum Arabic) เคซีน (Casein: ตะกอนโปรตีนจากนม) และสารพลาสติกโพลีเมอร์ (Polymer) มาใช้เป็นส่วนผสม ทำให้เกิดสีชนิดต่างๆ ขึ้นมา องค์ประกอบของสีแสดงได้ดังนี้

เนื้อสี (รงควัตถุ) + ส่วนผสม = สีชนิดต่าง ๆ

(Pigment) (Binder) Colour

ในสมัยต่อมา เมื่อมนุษย์มีวิวัฒนาการมากขึ้น จึงได้เริ่มมีการรับรู้และชื่นชมในความงาม ทางสุนทรียศาสตร์(Aesthetics) สีจึงได้ถูกนำมาใช้อย่างกว้างขวางและวิจิตรพิสดาร ซึ่งจากเดิมที่เคยใช้สีเพียงไม่กี่สีซึ่งเป็นสีตามธรรมชาติได้นำมาซึ่งการประดิษฐ์คิดค้น และผลิตสีใหม่ๆ ออกมาเป็นจำนวนมาก ทำให้เกิดการสร้างสรรค์ความงามอย่างไม่มีขีดจำกัด โดยมีการพัฒนามาเป็นระยะๆ อย่างต่อเนื่อง

2.2.1 ที่มาของสี

สีที่มนุษย์ใช้อยู่ทั่วไป ได้มาจาก

1. สสารที่มีอยู่ตามธรรมชาติและนำมาใช้โดยตรง หรือด้วยการสกัด ดัดแปลงบ้าง จากพืช สัตว์ ดิน แร่ธาตุต่าง ๆ

2. สสารที่ได้จากการสังเคราะห์ซึ่งผลิตขึ้นโดยกระบวนการทางเคมี เป็นสารเคมีที่ผลิตขึ้นเพื่อให้สามารถนำมาใช้ได้ สะดวกมากขึ้น ซึ่งเป็นสีที่เราใช้อยู่ทั่วไปในปัจจุบัน

3. แสง เป็นพลังงานชนิดเดียวที่ให้สี โดยอยู่ในรูปของรังสี (Ray) ที่มีความเข้มของแสงอยู่ในช่วงที่สายตามองเห็นได้

2.2.2 แสงสี

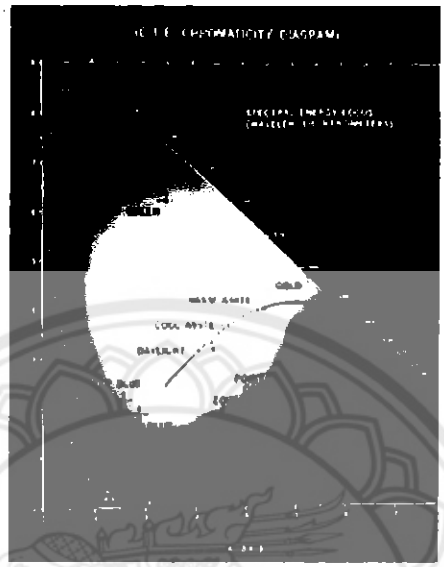
แสง เป็นพลังงานรังสี (Radiation Energy) ที่ตารับรู้และมีปฏิกิริยาตอบสนองด้วยกระบวนการวิเคราะห์แยกแยะของสมอง ตาสามารถวิเคราะห์พลังงานแสงโดยการรับรู้วัตถุ สัมพันธ์กับตำแหน่งทิศทาง ระยะทาง ความเข้มของแสง และความยาวคลื่นที่มองเห็นได้ ดังรูปที่ 2.5

รูปที่ 2.5 ความยาวคลื่นของสี

สี คือลักษณะความเข้มของแสงที่ปรากฏแก่สายตาให้เห็นเป็นสี โดยผ่านกระบวนการรับรู้ด้วยตา มองจะรับข้อมูลจากตา โดยที่ตาได้ผ่านกระบวนการวิเคราะห์ข้อมูลพลังงานแสงมาแล้ว ผ่านประสาทสัมผัสการมองเห็น ผ่านศูนย์สับเปลี่ยนในสมองไปสู่ศูนย์การมองเห็นภาพ การสร้างภาพหรือการมองเห็นก็คือ การที่ข้อมูลได้ผ่านการวิเคราะห์แยกแยะให้เรารับรู้ถึงสรรพสิ่งรอบตัว

การตรวจวัดคลื่นแสงเริ่มขึ้นใน คริสต์ศตวรรษที่ 19 ในปี 1928 ไรท์ (W.D.Wright) และ กิลด์ (J.Guild) ประสบความสำเร็จในการตรวจวัดคลื่นแสงครั้งสำคัญ และได้รับการรับรองจาก Commission Internationale de l 'Eclairage หรือ CIE ในปี 1931 โดยถือว่าเป็นการตรวจวัดมาตรฐาน

สามเหลี่ยมสี CIE เป็นภาพแสดงรูปสามเหลี่ยมเกือกม้า นำเสนอไว้ในปี 1931 โดยการวิเคราะห์สีจากแสงสเปคตรัม สัมพันธ์กับความยาวคลื่นแสง แสดงถึงแสงสีขาวท่ามกลางแสงสเปคตรัมรอบรูปเกือกม้า ดังแสดงในรูปที่ 2.6



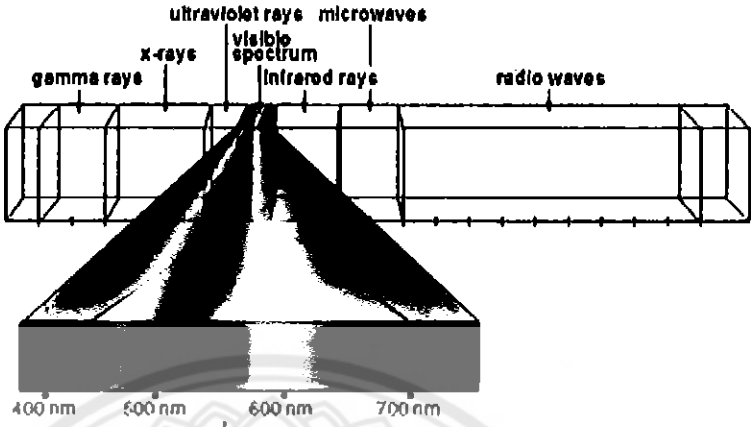
รูปที่ 2.6 สามเหลี่ยม CIE

โค้งรูปเกือกม้าดังรูปที่ 2.6 แสดงความยาวคลื่นจาก 400- 700 mu สามเหลี่ยมสี CIE สร้างขึ้นตามระบบความสัมพันธ์พิกัด X และ Y คาร์เตเซียน ในทางคณิตศาสตร์จากมุมตรงข้าม 3 มุมของรูปเกือกม้า คือสีน้ำเงินม่วงเข้มประมาณ 400 mu สีเขียวประมาณ 520 mu และสีแดงประมาณ 700 mu คือสีจากแสงที่จะนำมาผสมกันและก่อให้เกิดสีต่างๆ ขึ้น แสงสีแดงมีความยาวคลื่นสูงสุด แต่มีความถี่คลื่นต่ำสุด จะหักเหได้น้อยที่สุดและแสงสีม่วงจะมีความยาวคลื่นน้อยสุด แต่มีความถี่คลื่นสูงสุด และหักเหได้มากที่สุด

โครงสร้างของสามเหลี่ยมสี CIE นี้ ไม่ได้ขึ้นอยู่กับทฤษฎีใดทฤษฎีหนึ่ง แต่เกิดจากการทดลองค้นคว้าทางวิทยาศาสตร์ ระบบการพิมพ์อุตสาหกรรม การถ่ายภาพ ภาพยนตร์ โทรทัศน์ ได้ใช้โครงสร้างสีนี้เป็นหลักในระบบการพิมพ์ได้ใช้สีจากด้าน 3 ด้านของรูปเกือกม้าคือ สีเหลือง ฟ้า สีม่วงแดง และสีดำเป็นหลัก ส่วนในการถ่ายภาพ ภาพยนตร์ โทรทัศน์ จอคอมพิวเตอร์ ใช้สีจากมุมทั้งสาม คือ แดง เขียว น้ำเงิน เป็นหลัก

ในราวปี ค.ศ. 1666 เซอร์ ไอแซค นิวตันได้แสดงให้เห็นว่า สีคือส่วนหนึ่งในธรรมชาติของแสงอาทิตย์โดยให้ลำแสงส่องผ่านแท่งแก้วปริซึม แสงจะหักเห เพราะแท่งแก้วปริซึมความหนาแน่นมากกว่าอากาศ เมื่อลำแสงหักเหผ่านปริซึมจะปรากฏแถบสีสเปคตรัม (Spectrum) หรือที่เรียกว่า สีรุ้ง (Rainbow) คือ สีม่วง คราม น้ำเงิน เขียว เหลือง แสด แดง เมื่อแสงตกกระทบโมเลกุลของสสาร พลังงานบางส่วนจะดูดกลืนสีจากแสงบางส่วน และสะท้อนสีบางสีให้ปรากฏเห็นได้ พื้นผิววัตถุที่เรา

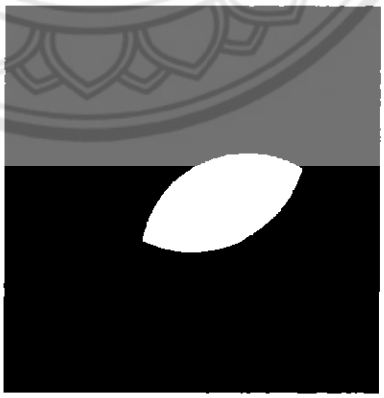
เห็นเป็นสีแดง เพราะวัตถุดูดกลืนแสงสีอื่นไว้ สะท้อนเฉพาะแสงสีแดงออกมา วัตถุสีขาวจะสะท้อนแสงสีทุกสี และวัตถุสีดำจะดูดกลืนทุกสี



รูปที่ 2.7 การหักเหของแสง

จากรูปที่ 2.7 แสดงถึงทฤษฎีการหักเหของแสงของนิวตัน และจากสามเหลี่ยมสี CIE พบว่าแสงสีเป็นพลังงานเพียงชนิดเดียวที่ปรากฏสีจากด้านทั้ง 3 ด้านของรูปสามเหลี่ยมสี CIE นักวิทยาศาสตร์ได้กำหนดแม่สีของแสงไว้ 3 สี คือ สีแดง (Red) สีเขียว (Green) และสีน้ำเงิน (Blue) ดังรูปที่ 2.8 แสงทั้งสามสี เมื่อนำมาฉายส่องรวมกัน จะทำให้เกิดสีต่างๆ ขึ้นมา คือ

แสงสีแดง + แสงสีเขียว = แสงสีเหลือง (Yellow)
แสงสีแดง + แสงสีน้ำเงิน = แสงสีแดงมาเจนตา (Magenta)
แสงสีน้ำเงิน + แสงสีเขียว = แสงสีฟ้าไซแอน (Cyan)



รูปที่ 2.8 แม่สีของแสง 3 สี คือ สีแดง สีเขียว และสีน้ำเงิน

และถ้าแสงสีทั้งสามสีฉายรวมกัน จะได้แสงสีขาวหรือไม่มีสี เราสามารถสังเกตแม่สีของแสงได้จากโทรทัศน์สีหรือจอคอมพิวเตอร์สี โดยใช้แว่นขยายส่องดูหน้าจอจะเห็นเป็นแถบสีแสงสว่าง 3 สี คือ แดง เขียว และน้ำเงิน นอกจากนี้เราจะสังเกตเห็นว่า เครื่องหมายของสถานีโทรทัศน์สีหลายๆ

ช่อง จะใช้แม่สีของแสงด้วยเช่นกัน ทฤษฎีของแสงสีนี้เป็นระบบสีที่เรียกว่า RGB (Red - Green - Blue) เราสามารถนำไปใช้ในการถ่ายทำภาพยนตร์ บันทึกภาพวิดีโอ การสร้างภาพเพื่อแสดงทางคอมพิวเตอร์ การจัดไฟแสงสีในการแสดง การจัดฉากเวที เป็นต้น

2.3 ระบบสี RGB [1]

ระบบสี RGB เป็นระบบสีของแสง ซึ่งเกิดจากการหักเหของแสงผ่านแท่งแก้วปริซึม จะเกิดแถบสีที่เรียกว่า สีรุ้ง (Spectrum) ซึ่งแยกสีตามที่สายตามองเห็นได้ 7 สี คือ แดง แสด เหลือง เขียว น้ำเงิน คราม ม่วง ซึ่งเป็นพลังงานอยู่ในรูปของรังสีที่มีช่วงคลื่นที่สายตาสามารถมองเห็นได้ แสงสีม่วงมีความถี่คลื่นสูงที่สุด คลื่นแสงที่มีความถี่สูงกว่าแสงสีม่วง เรียกว่า อุลตราไวโอเลต (Ultra Violet) และคลื่นแสงสีแดงมีความถี่คลื่นต่ำที่สุด คลื่นแสงที่ต่ำกว่าแสงสีแดงเรียกว่า อินฟราเรด (Infrared) คลื่นแสงที่มีความถี่สูงกว่าสีม่วงและต่ำกว่าสีแดงนั้น สายตาของมนุษย์ไม่สามารถรับได้และเมื่อศึกษาดูแล้ว แสงสีทั้งหมดเกิดจากแสงสี 3 สี คือ สีแดง (Red) สีน้ำเงิน (Blue) และสีเขียว (Green) ดังแสดงในรูปที่ 2.9 ทั้งสามสีถือเป็นแม่สีของแสง เมื่อนำมาฉายรวมกันจะทำให้เกิดสีใหม่อีก 3 สี คือ สีแดงมาเจนต้า สีฟ้าไซแอน และสีเหลือง และถ้าฉายแสงสีทั้งหมดรวมกันจะได้แสงสีขาว จากคุณสมบัติของแสงนี้เราได้นำมาใช้ประโยชน์ทั่วไป ในการฉายภาพยนตร์ การบันทึกภาพวิดีโอ ภาพโทรทัศน์ การสร้างภาพเพื่อการนำเสนอทางจอคอมพิวเตอร์ และการจัดแสงสีในการแสดง เป็นต้น

รูปที่ 2.9 ระบบสี RGB

2.4 โหมดสี [1]

การสร้างภาพด้วยคอมพิวเตอร์ จะต้องเกี่ยวข้องกับโหมดสี (Color Mode) ดังนั้นผู้สนใจงานกราฟิก ควรสนใจโหมดสีด้วย เพื่อให้การสร้างงานกราฟิกได้ผลตรงตามต้องการ ทั้งนี้โหมดสีบนระบบคอมพิวเตอร์ มีรายละเอียดดังนี้

โหมด Bitmap - เป็นโหมดสีที่เก็บข้อมูลของสี 1 บิตต่อพิกเซล โดยแสดงสีได้เพียงสีขาวและสีดำ ไม่มีการไล่โทนสี ลักษณะภาพที่ได้จึงมีความหยาบ แต่ได้ภาพที่มีขนาดเล็ก เหมาะสำหรับภาพลายเส้น

Indexed Color - โหมดสีที่ใช้ตารางในการเทียบสี โดยใช้ข้อมูลจำนวน 8 บิตต่อพิกเซล ทำให้แสดงสีได้ 256 สี ต่อพิกเซล ข้อเสียคือมีเลเยอร์ภาพเพียงเลเยอร์เดียว ทำให้ภาพขาดรายละเอียดไป

Grayscale - โหมดสีภาพขาวดำ ที่สามารถไล่โทนได้ 256 ระดับ

RGB Color - เป็นโหมดสีที่ใช้ Channel สีจำนวน 3 สี ได้แก่ แดง, เขียว, น้ำเงิน และแต่ละสีไล่เฉดสีได้ 256 ระดับ เมื่อรวมทั้ง 3 สี จะสามารถแสดงสีได้ถึง 16.7 ล้านสี ซึ่งเหมาะต่อการตกแต่งภาพ

Duotone - โหมดสีสำหรับภาพแบบ Monotone, Duotone, Tritone, Quotone โดยรูปภาพจะแสดงแบบ Grayscale เพียง Channel เดียว มีค่า 8 บิตต่อพิกเซล

CMYK Color - โหมดสีจาก Channel สี 4 สี คือ Cyan (ฟ้า), Magenta (บานเย็น), Yellow, Black แต่ละสีเก็บข้อมูล 8 บิต เป็นโหมดที่เหมาะสำหรับงานพิมพ์

Multichannel - โหมดสีที่มีการเก็บข้อมูลสี 8 บิตต่อพิเซล แสดงสีได้ 256 สี ใช้ในงานพิมพ์บางอย่าง

Lab Color - โหมดสีเสมือนจริง ใช้ค่า Lightness แทนความสว่าง มีค่า 0 - 100 และค่า a แทนสีเขียวถึงสีแดง และค่า b แทนสีน้ำเงิน ซึ่งมีค่า +120 ถึง -120 ใช้ในงานพิมพ์ระดับสูง

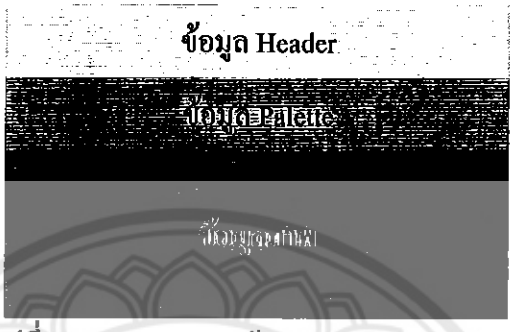
2.5 ความหมายและโครงสร้างของไฟล์ภาพชนิด Bitmap หรือ BMP [11]

ภาพแบบ Bitmap หรืออาจจะเรียกว่าภาพแบบราสเตอร์ (Raster) เป็นภาพที่เกิดจากจุดสีที่เรียกว่า Pixels ซึ่งประกอบกันเป็นรูปร่างบนพื้นที่ที่มีลักษณะเป็นเส้นตาราง (Grid) แต่ละ Pixels จะมีค่าของตำแหน่งและค่าสีของตัวเอง ภาพหนึ่งภาพ จะประกอบด้วย Pixels หลายๆ Pixels ผสมกัน แต่เนื่องจาก Pixels มีขนาดเล็กมาก จึงเห็นภาพมีความละเอียดสวยงาม ไม่หึนลักษณะของกรอบสี่เหลี่ยม จึงเป็นภาพที่เหมาะสมต่อการแสดงภาพที่มีเฉดและสีสันจำนวนมาก เช่น ภาพถ่ายหรือภาพวาด ภาพแบบ Bitmap เป็นภาพที่ขึ้นอยู่กับความละเอียดหรือความคมชัด (Resolution) ซึ่งก็คือ จำนวน Pixels ที่แน่นอนในการแสดงภาพ ดังนั้นเมื่อมีการขยายภาพ จะเกิดปัญหาคือ เห็นเป็นกรอบสี่เหลี่ยมเล็กๆ หลายๆ จุด ประกอบกัน เพราะ Grid ของภาพมีขนาดที่แน่นอนนั่นเอง และจากรูปที่ 2.10 แสดงถึงส่วนของโครงสร้างของไฟล์ BMP ที่ประกอบด้วย 3 ส่วนคือ

1. ข้อมูลเฮดเดอร์ () คือ ข้อมูลที่อยู่บริเวณส่วนหัวของไฟล์ ซึ่งประกอบไปด้วยข้อมูล ที่บอกรายละเอียดต่างๆของภาพ เช่น ความกว้างและความยาวของภาพ จำนวนสี จำนวนบิต ความละเอียด เป็นต้น

2. ข้อมูลจานสี () คือ ข้อมูลที่บอกถึงชุดของจานสี () ที่เกิดจากการผสมแม่สีทั้งสามคือ Red Green Blue มาผสมกันได้เป็นสีต่างๆ ตามจำนวนสีของภาพ เช่น รูปขนาด 4 บิต จะมี 16 สี รูปขนาด 8 บิต จะมี 256 สี เป็นต้น ซึ่งถ้ามีจำนวนสีน้อยๆ ก็จะมีการเก็บค่า Palette นี้ลงไฟล์ไปด้วย แต่ถ้าเป็นรูปประเภท 24 บิต จะไม่มีค่า Palette แต่จะใช้วิธีการเก็บค่าแม่สีทั้งสามลงไปเป็นข้อมูลแทน เพราะถ้าเก็บค่า Palette ที่มีถึง 16.7 ล้านสีลงไปด้วยจะเปลืองพื้นที่มาก ข้อแตกต่างของ BMP คือไฟล์ BMP จะเก็บค่าของ Palette = ชุดละ 4 ไบต์ คือ Red Green Blue อย่างละ 1 ไบต์

3. ข้อมูลภาพ (Data) คือ ข้อมูลสีของภาพแต่ละจุดบนจอภาพที่มาประกอบกันเป็นรูปภาพ ซึ่งค่าที่เก็บนี้จะเป็นค่าที่ใช้ในการชี้หมายเลขตาราง Palette เช่น จุดแรกมีค่าเป็น 10 ก็ไปเปิดตาราง Palette หมายเลข 10 สมมติว่าได้ความเข้มของแม่สีเป็น R = 0, G = 0, B = 100 ก็จะได้จุดนี้เป็นสีน้ำเงินซึ่งถ้าเป็นในกรณีของรูป 24 บิต จะเป็นการอ่านข้อมูลขึ้นมา 3 ค่า เป็นค่าของแม่สี RGB แล้วนำไปผสมบนจอแทน



รูปที่ 2.10 ลักษณะของข้อมูลภาพชนิด Bitmap

2.5.1 การจัดเก็บไฟล์ข้อมูลภาพชนิด BMP

การเก็บไฟล์ข้อมูลภาพชนิด BMP มีการเก็บอยู่ 2 แบบ คือ

1. แบบบีบอัดข้อมูล RLE4 เป็นการบีบอัดข้อมูลแบบ Run Length Encoder แบบ 4 บิต และ RLE 8 เป็นการบีบอัดข้อมูลแบบ Run Length Encoder แบบ 8 บิต

2. แบบไม่ได้บีบอัดข้อมูลเป็นการเก็บข้อมูลจริงๆ ซึ่งทำให้ไฟล์ภาพค่อนข้างใหญ่แต่จะทำการแสดงภาพได้รวดเร็วเพราะไม่ต้องเสียเวลาในการคลายข้อมูล

File Header

ไฟล์ภาพ BMP จะเก็บข้อมูลเป็น 4 ส่วนคือ File Header, Info Header, Color Table และ Bitmap Data ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 ตารางข้อมูลโครงสร้างของไฟล์บิตแมป

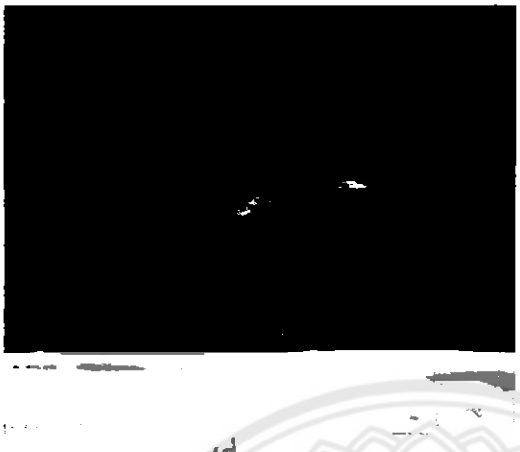
ชื่อ		ตำแหน่ง	ขนาด	รายละเอียด
File Header		1	14	เก็บรายละเอียดของไฟล์
	Signature	1	2	ต้องเป็นตัวอักษร 'BM' เสมอ
	File Size	3	4	ขนาดของไฟล์
	Reserved 1	7	2	ไม่ได้ใช้งาน มีค่าเป็น 0
	Reserved 2	9	2	ไม่ได้ใช้งาน มีค่าเป็น 0
	Bit Offset	11	4	ตำแหน่งที่เก็บข้อมูลรูปภาพ นับจากต้นไฟล์

ตารางที่ 2.1(ต่อ) ตารางข้อมูลโครงสร้างของไฟล์บิตแมป

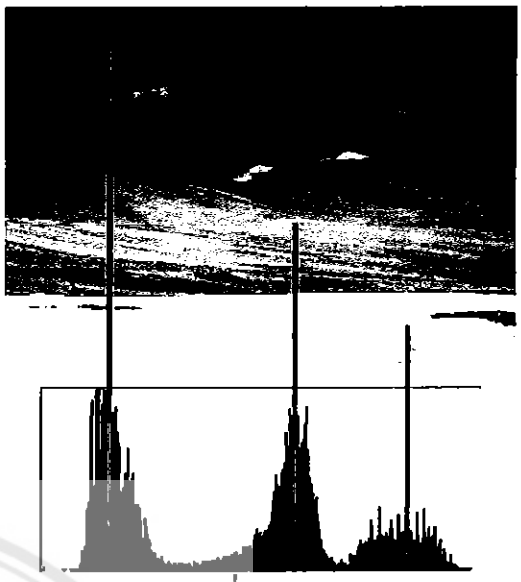
ชื่อ		ตำแหน่ง	ขนาด	รายละเอียด
Info Header		15	40	เก็บรายละเอียดของภาพ
	Size	15	40	ขนาดของ Info Header = 40
	Width	19	4	ความกว้างของภาพ
	Height	23	4	ความสูงของภาพ
	Planes	27	2	จำนวนชุดของตารางสี (=1)
	Bit Count	29	2	จำนวนบิตสีของภาพต่อ 1 Pixel สามารถมีค่าได้ดังนี้ 1 บิต = รูปขาวดำ 4 บิต = 16 สี 8 บิต = 256 สี 16 สี = 65,536 สี 24 บิต = 16 ล้านสี 32 บิต = 16 ล้านสี + ความโปร่งใสของรูป (Alpha)
	Compression	31	4	รูปแบบของการบีบอัดข้อมูลรูปภาพ 0 = BI_RGB ไม่มีการบีบอัด 1 = BI_RLE8 บีบอัดโดยการเข้ารหัสแบบ 8 Bit RLE 2 = BI_RLE4 บีบอัดโดยการเข้ารหัสแบบ 4 Bit RLE 3 = BI_BITFIELDS บีบอัดโดยการเข้ารหัสแบบ Bit Fields
	Image Size	35	4	ขนาดของภาพ ใช้ในกรณีที่มีการบีบอัดรูปภาพ ถ้าไม่มีการบีบอัดควรกำหนดเป็น 0

2.6 ทฤษฎี Histogram [2]

Histogram คือ มาตรวัดที่ใช้ในการบอกการกระจายของค่าระดับเทาในภาพทั้งภาพ โดยการนำภาพสี (RGB) ที่มีอยู่มาทำการแปลงค่าของสีภาพเป็นระดับเทา เพื่อนำมาทำการวิเคราะห์ ซึ่งการวิเคราะห์จาก Histogram นี้จะได้ผลออกมาเป็นกราฟแท่งที่บอกความสว่างในแต่ละช่วงของภาพ ตัวอย่างเช่น รูปที่ 2.11 (ก) จะเป็นภาพ RGB ปกติ เมื่อทำการเปลี่ยนเป็นภาพระดับเทา (Gray Level) และพล็อตเป็นกราฟHistogram จะกลายเป็น รูปที่ 2.11 (ข)



รูปที่ 2.11 (ก)



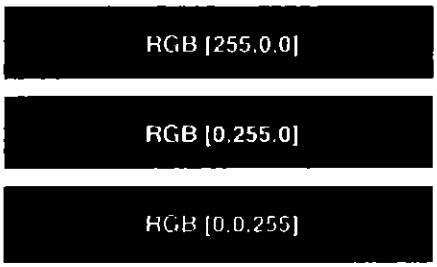
รูปที่ 2.11 (ข)

รูปที่ 2.11 แสดงฮิสโตแกรม

จากรูปที่ 2.11 (ข) จะเห็นได้ว่าเมื่อทำการพล็อตกราฟออกมาแล้ว ในบริเวณช่วงแท่งกราฟช่วงแรกจะมีปริมาณความเข้มสูงและเยอะมากเนื่องมาจากเป็นบริเวณสีของท้องฟ้าที่มีความทึบของสีฟ้าและถัดมาเป็นช่วงของภูเขาที่มีพื้นที่มาก แต่ความเข้มของระดับเทาน้อยกว่าช่วงของท้องฟ้า และสุดท้ายช่วงของพื้นที่มีระดับความสว่างและพื้นที่ค่อนข้างน้อยจึงมีปริมาณแท่งน้อยกว่าทั้งสองกลุ่ม

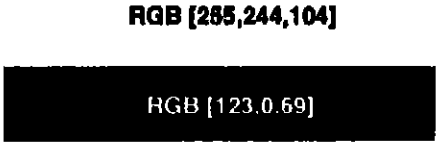
2.6.1 ข้อมูลสีฮิสโตแกรมแบบ RGB (RGB Histogram)

Histogram แบบ RGB เป็น Histogram ที่ถูกสร้างขึ้นจากภาพสีในโหมด RGB ตัว RGB Histogram จะมีวิธีการสร้างและที่มาซับซ้อนขึ้นไปอีกระดับ ในภาพแบบ 24 บิตนั้น แต่ละจุดของภาพจะประกอบไปด้วยสี 3 สี มาผสมกัน เรียกแต่ละสีว่า Channel ซึ่งก็จะมีสี แดง เขียว น้ำเงิน Red Green Blue ประกอบกันเป็นแต่ละจุดในภาพ ยกตัวอย่าง เช่น ถ้าเป็นสีแดงทั้งหมด จะมีค่าเป็น RGB (255,0,0) ส่วน RGB(0,255,0) และ RGB (0,0,255) ก็จะเป็นสีเขียวและสีน้ำเงินตามลำดับ ดังรูปที่ 2.12



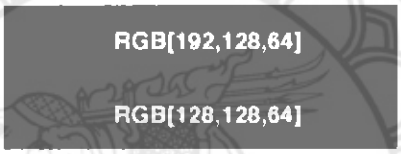
รูปที่ 2.12 แสดงค่าสีแดง RGB (255, 0, 0) สีเขียว RGB (0, 255, 0) และสีน้ำเงิน RGB (0, 0, 255)

ในความเป็นจริงแล้ว แต่ละจุดสีบนภาพ เกิดจากการผสมสีใน 3 Channel นี้เข้าด้วยกัน ทำให้เกิดเป็นสีต่าง ๆ ขึ้น เช่น สีเหลือง ดังรูป จะมีค่า RGB เท่ากับ (255,244,104) หรือสีแดงเลือดหมู จะมีค่า RGB เท่ากับ (123, 0, 69) ดังรูปที่ 2.13 โดยที่แต่ละค่าสี R G B มีช่วงแตกต่างกันได้ถึง 256 ระดับ



รูปที่ 2.13 แสดงค่าสี RGB (255,244,104) และ RGB (123, 0, 69) ตามลำดับ

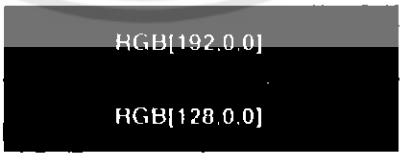
วิธีการสร้าง RGB Histogram จากภาพง่าย ๆ ที่มีแค่ 2 สีประกอบกันให้ดูตามรูปที่ 2.14



รูปที่ 2.14 แสดงค่าสี RGB (192, 128, 64) และ RGB (128, 128, 64) ตามลำดับ

ดังรูปที่ 2.14 สีด้านบน มีค่า RGB (192, 128, 64) ส่วนสีด้านล่างมีค่า RGB (128, 128, 64) ต่อไปทำการแยกสีของรูปนี้ออกเป็น 3 Channel Red Green Blue เรียงกันไปตามลำดับ

เลือกพิจารณาที่ Red Channel ก่อน โดยจะนำค่าสีแดงเท่านั้นมาใช้ สีอื่นๆ ให้มีค่าเป็น 0 ทั้งหมด สีด้านบนมีค่าสีแดงเป็น 192 ด้านล่างเป็น 128 จะได้ Red Channel ดังรูปที่ 2.15

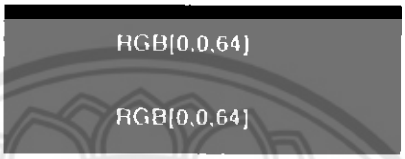


รูปที่ 2.15 แสดง Red Channel ที่ได้จากการแยกสีรูปที่ 2.14

ต่อไป พิจารณาที่ Green Channel จะมีค่าสีด้านบนและล่างเท่ากันคือ 128 ทำให้เมื่อแยก Channel มาสร้างรูปเฉพาะสีเขียวแล้วจะได้ด้านบนและด้านล่างเหมือนกัน ดังรูปที่ 2.16

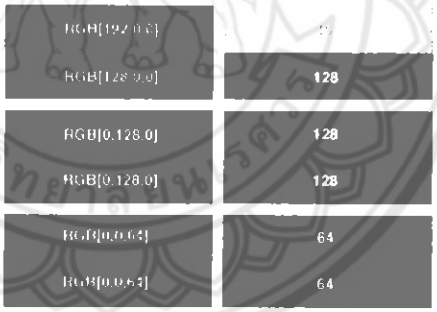
รูปที่ 2.16 แสดง Green Channel ที่ได้จากการแยกสีรูปที่ 2.14

และในการพิจารณา Blue Channel ก็เช่นกัน จะมีค่าเท่ากับ 64 ทั้งด้านบนและด้านล่าง สีของทั้งรูปก็จะออกมาสีน้ำเงินในระดับเดียวกัน ดังรูปที่ 2.16



รูปที่ 2.17 แสดง Blue Channel ที่ได้จากการแยกสีรูปที่ 2.14

จากนั้น มาพิจารณาที่ระดับความสว่างของแต่ละ Channel ที่แยกออกมา โดยใช้ระดับของสีเทามาแสดงผลแล้วนำมาเปรียบเทียบกันจะได้ผลดังรูป 2.18



รูปที่ 2.18 แสดงระดับความสว่างของแต่ละ Channel

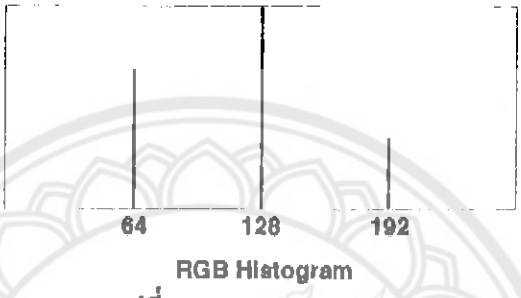
ระดับความสว่างของ Red Channel จะมีสองส่วนคือ ด้านบน จะมีค่าเป็น 192 ด้านล่าง จะมีค่าเป็น 128 ส่วนระดับความสว่างของ Green Channel และ Blue Channel จะมีระดับความสว่างเท่ากันทั้งภาพคือ 128 และ 64 ตามลำดับ

จากนั้น นำระดับความสว่างที่ได้ข้างต้นมาใช้ในการสร้าง RGB Histogram โดยจะนำระดับความสว่างเฉพาะที่เป็นสีเทาของแต่ละ Channel มาใช้ในการสร้าง RGB Histogram

- ใน Red Channel จะได้ระดับความสว่างแสดงเป็นแท่งกราฟที่ตำแหน่งค่า 192 และ 128
- ใน Green Channel จะได้ระดับความสว่างแสดงเป็นแท่งกราฟที่ตำแหน่งค่า 128
- ใน Blue Channel จะได้ระดับความสว่างแสดงเป็นกราฟแท่งที่ตำแหน่งค่า 64

จากนั้น ก็นำ Histogram ของแต่ละ Channel มารวมกันให้เป็น RGB Histogram ในการรวมกันนั้น เนื่องจาก Histogram ของแต่ละ Channel มีระดับความสูงหรือแกน y มีค่าที่ไม่เท่ากัน ดังนั้น เราต้องวัดระดับความสำคัญ โดยการนำแกน y มาเปรียบเทียบให้อยู่ในสเกลเดียวกันก่อน โดยพิจารณาจากปริมาณสีเทาของแต่ละระดับของทั้ง 3 รูป

ซึ่งจะได้พื้นที่ระดับความสว่าง (สีเทา) ของรูปที่ผ่านมาข้างต้น มีค่า 192 มีสัดส่วนอยู่ 1 ใน 3 ของ 128 และค่า 64 จะมีสัดส่วนอยู่ 2 ใน 3 ของ 128 จากนั้น ให้นำสัดส่วนความสูงของกราฟแต่ละแท่งมาสร้างเป็น RGB Histogram ดังรูปที่ 2.19



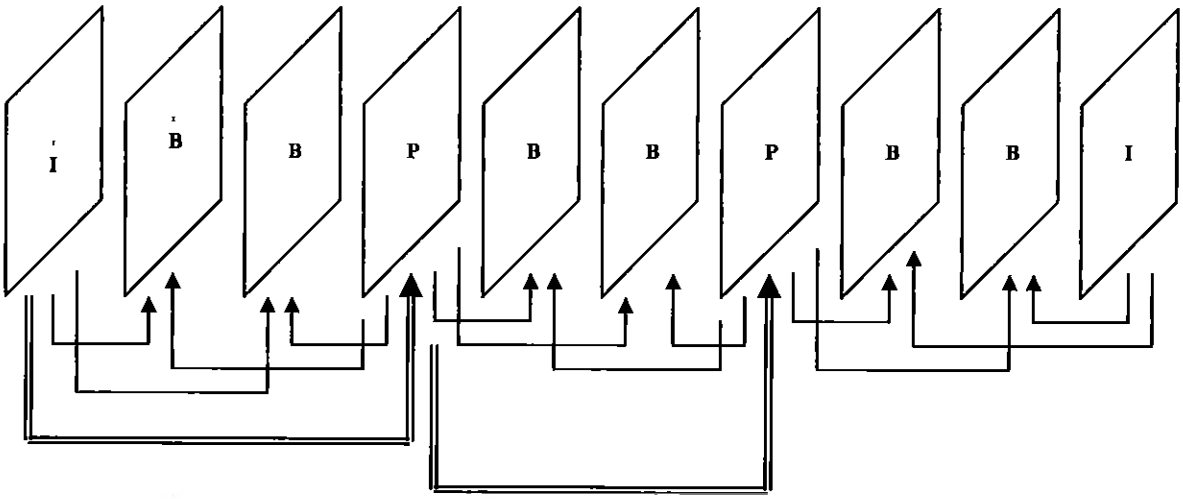
รูปที่ 2.19 RGB Histogram

2.7 ข้อมูลประเภทวิดีโอ (Video content) [11]

ภาพวีดีโอแบบดิจิตอล (Digital Video) เกิดจากการนำภาพวิดีโอที่ถ่ายด้วยกล้องถ่ายวิดีโอ ม้วนเทปวิดีโอ หรือกล้องถ่ายภาพยนตร์ นำมาบันทึกให้อยู่ในรูปแบบไฟล์ในคอมพิวเตอร์ โดยใช้ฮาร์ดแวร์ (Hardware) พิเศษที่เรียกว่า บอร์ดการจับภาพวีดีโอ (Group Based Representation) ในการจับภาพวิดีโอมาเป็นไฟล์ภาพในรูปแบบดิจิตอล

วีดีโอ (Video) มีอยู่หลายชนิดไม่ว่าจะเป็น เอวีไอ (AVI), เอ็มโอวี (MOV), เอ็มพีอีจี (MPEG) หรือจะเป็นวีดีโอแบบดิจิตอล (Digital Video) ที่ใช้ดูบนอินเทอร์เน็ต (Internet) เช่น เรียลวีดีโอ (Real Video) เป็นต้น ซึ่งก็มีลักษณะคล้ายๆ กัน ต่างกันที่คุณภาพของภาพ ความต่อเนื่องของภาพ (Playback Rate) และขนาดของไฟล์ (Compression) ที่จะมีขนาดเล็กใหญ่แตกต่างกันไป

เฟรม (Frame) ภาพยนตร์ที่บีบอัดโดยเอ็มพีอีจี-1 (MPEG1) จะมีอยู่ 3 ชนิดคือ เฟรมอินทรา (Intra-Picture : I), เฟรมพรีดิ็คทีฟ (Predictive : P) และเฟรมไบ-ไดเร็คชั่นนัล (Bi-directional : B) โดยที่ไอเฟรม (I frame) จะเป็นเฟรม (frame) สำหรับให้เฟรม (frame) อื่น ๆ อ้างอิงและเก็บข้อมูลภาพของตัวเองไว้ครบถ้วน พีเฟรม (P frame) จะอ้างอิงข้อมูลของเวคเตอร์เคลื่อนที่ (motion vector) และที่เป็นเศษเหลือ (residual) มาจากเฟรม (frame) ก่อนหน้าเท่านั้น ส่วนบีเฟรม (B frame) จะอ้างอิงมาจากทั้งพี (P) และไอ (I) ที่อยู่ก่อนหน้าและข้างหลังด้วยทำให้ลดข้อมูลที่ต้องส่งเพิ่มในบางเฟรม (frame) ลงไปได้ ดังแสดงในรูปที่ 2.20



รูปที่ 2.20 แสดงการอ้างอิงของวีดีโอ

2.8 การแยกส่วนของไฟล์วีดีโอ (Video segmentation) [3]

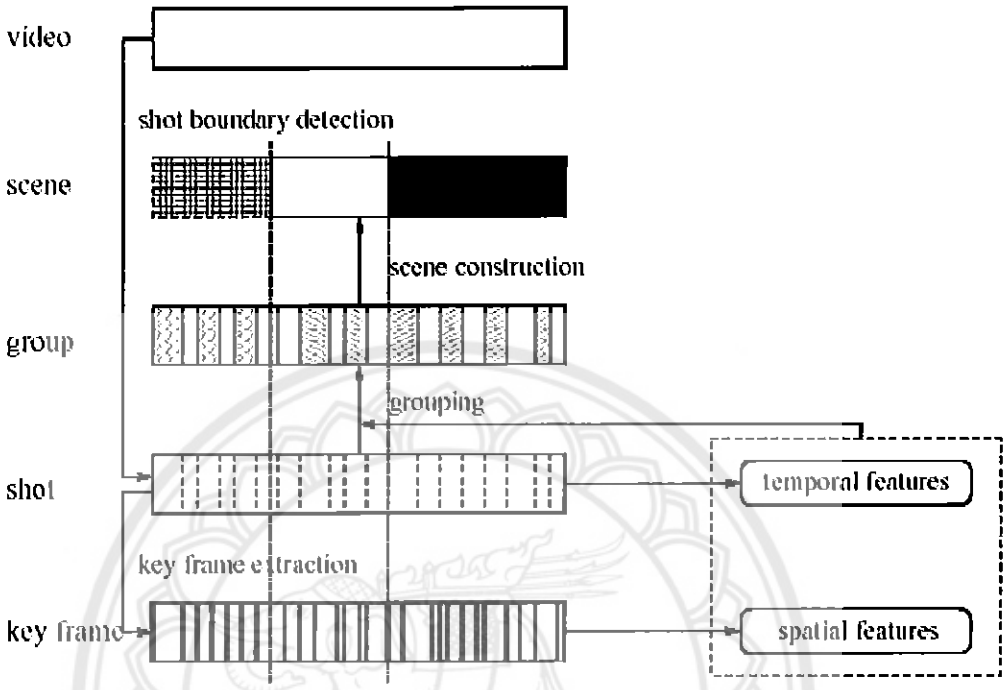
วีดีโอช็อต (Video shot): ลำดับที่ต่อเนื่องกันของเฟรม (frame) ที่ถูกอัดมาจากภาพเดี่ยว (single camera) ซึ่งเป็นการสร้างบล็อค (block) ของสายธารวีดีโอ (Video stream)

เฟรมหลัก (Key frame): เฟรม (frame) ซึ่งเป็นตัวแทนของเนื้อหาที่ทำให้เห็นเด่นชัดขึ้นของช็อต (shot) ซึ่งขึ้นอยู่กับความสลับซับซ้อนของช็อต (shot) โดยเฟรมหลัก (key frame) หนึ่ง ๆ สามารถขยายออกไปได้

วีดีโอซีน (Video scene) : ถูกนิยามเสมือนกับเป็นการรวบรวมของช็อต (shot) ที่เกี่ยวข้องกันทางสำนวน (semantically) กับช็อต (shot) ที่อยู่ใกล้กันในทางเป็นจริง ซึ่งเป็นการบอกเล่าและถ่ายทอดเรื่องราวหรือแนวคิดระดับสูง ในขณะที่ช็อต (shot) ถูกทำเครื่องหมาย (mark) โดยขอบเขตทางกายภาพ (physical boundaries) และซีน (scene) ถูกทำโดยขอบเขตทางสำนวน (semantic boundaries) ในระยะแรกๆ ในวีดีโอที่กระจายออกไปจะมีการใช้วลีในการสืบค้นการเปลี่ยนซีน (Scene change detection) สำหรับการสืบค้นขอบเขตของช็อต (Shot boundary detection) ในทาง ผิด ๆ เพื่อหลีกเลี่ยงความสับสน ในภายหลังจะใช้การสืบค้นขอบเขตของช็อต (Shot boundary detection) ในทางความหมายของการสืบค้นของขอบเขตของช็อต (shot) ทางกายภาพและใช้การสืบค้นขอบเขตของซีน (Scene boundary detection) ในทางความหมายของการสืบค้นของขอบเขตของซีน (scene) ในเชิงสำนวน (Semantic scene boundaries)

กลุ่มวีดีโอ (Video group): เป็นสิ่งที่มีตัวตนอยู่ตรงกลางระหว่างของช็อตทางกายภาพ (physical shot) กับซีนในเชิงสำนวน (semantic scenes) และให้บริการเหมือนเป็นสะพานเชื่อมระหว่างทั้งสองสิ่งนี้เข้าด้วยกัน ตัวอย่าง ช็อต (shot) ที่อยู่ใกล้กันในทางเป็นจริง (Temporally adjacent shots) หรือ ช็อตที่คล้ายคลึงอย่างชัดเจน (Visually similar shots)

โดยสรุปแล้ว ข้อมูลวีดีโอสามารถทำให้เป็นโครงสร้างแบบลำดับขั้น ซึ่งประกอบด้วย 5 ลำดับ คือ วีดีโอ (video) , ซีน (scene) , กลุ่ม (group) , ช็อต (shot) และเฟรมหลัก (key frame) ดังรูปที่ 2.21



รูปที่ 2.21 แสดงการแบ่งส่วนต่าง ๆ ของวีดีโอ

2.8.1 การวิเคราะห์วีดีโอ (Video Analysis)

การสืบค้นขอบเขตของช็อต (Shot boundary detection)

มันไม่ใช่ประสิทธิภาพทั้งหมดในการประมวลผลวีดีโอคลิป (Video clip) แต่ประโยชน์อันแรกที่ได้รับคือการแยกวีดีโอคลิป (video clip) ออกเป็นช็อต (shot) ต่างๆ และทำการประมวลผลสัญญาณที่ระดับช็อต (shot)

โดยทั่วไปการสืบค้นขอบเขตของช็อต (Shot boundary detection) แบบอัตโนมัติ สามารถที่จะแยกแยะออกเป็น 5 ประเภทคือ ใช้จุดภาพเป็นพื้นฐาน (pixel-based) , ใช้สถิติเป็นพื้นฐาน (statistic-based) , ใช้การแปลงเป็นพื้นฐาน (transform-based) , ใช้ลักษณะเด่นเป็นพื้นฐาน (feature-based) และใช้ฮิสโตแกรมเป็นพื้นฐาน (histogram-based) สำหรับการใช้จุดภาพเป็นพื้นฐาน (pixel-based) จะใช้ความเข้มที่แตกต่างกันของจุดภาพที่ฉลาด (pixel-wise) ในการทำเครื่องหมาย (mark) ขอบเขตของช็อต (shot) อย่างไรก็ตามมันมีความไวต่อเสียงค่อนข้างสูง การเอาชนะปัญหานี้คาสตุริ (Kasturi) และเจน (Jain) เสนอที่จะใช้สถิติอย่างเข้มข้น (ค่ากลางและส่วนเบี่ยงเบนมาตรฐาน) ในการวัดการสืบค้นขอบเขตของช็อต (Shot boundary detection) แต่การค้นหาเพื่อให้ประมวลผลเพื่อให้ประสบผลสำเร็จที่

เร็วกว่านั้นอาร์มาน (Arman) , ซู (Hsu) และชิว (Chiu) เสนอความเห็นที่จะให้ใช้สัมประสิทธิ์การบีบขนาดของดีซีที (DCT) ในการวัดการสืบค้นขอบเขตของช็อต (Shot boundary detection)

การสืบค้นของช็อต (shot) แบบใช้การแปลงเป็นพื้นฐาน (transform-based) นั้นใช้ประโยชน์ของเวคเตอร์เคลื่อนที่ (motion vector) ซึ่งถูกฝังไว้ในสายธารเอ็มพีอีจี (MPEG stream) เรียบร้อยแล้ว ซาบี เอท อัล (Zabih et al) ได้พูดถึงปัญหาในมุมอื่นๆ ลักษณะของขอบ (edge feature) จะถูกแยกออกจากแต่ละเฟรม (frame) เป็นอันดับแรก ขอบเขตของช็อต (shot boundaries) จะถูกสืบค้นโดยการพบการเปลี่ยนขอบอย่างทันทีทันใด เรื่อยมาจนถึงปัจจุบัน

การใช้ฮิสโตแกรมเป็นพื้นฐาน (Histogram-based) จะเป็นที่รู้จักกันมาก แทนที่จะใช้ความเข้มของจุด (pixel) ในการวัด งานวิจัยหลายๆ เล่มได้บันทึกว่าประสบผลสำเร็จดี ระหว่างความถูกต้องและความรวดเร็ว ผลงานจำนวนมากเมื่อเร็วๆ นี้ มีพื้นฐานอยู่บนการจับกลุ่มและภายหลังการกลั่นกรองซึ่งประสบความสำเร็จอันสวยงามในเรื่องความถูกต้องแน่นอนอย่างสูง โดยปราศจากผลที่ผิดพลาด

2.8.2 ตัวแทนของวีดีโอ (Video Representation)

พิจารณาแต่ละเฟรม (frame) ของวีดีโอ (video) จะเป็นวัตถุ 2 มิติ และแกนชั่วคราว (temporal axis) จะสร้างมิติที่ 3 ตัวแทนของวีดีโอ (video representation) จะเป็นการจับคู่ (mapping) จากพื้นที่ว่าง 3 ดี (3D space) ไปยังจอภาพ 2 มิติ การจับคู่ (mapping) ที่แตกต่างกันเหล่านั้น ถูกแสดงให้เป็นเทคนิคต่างๆ กันของตัวแทนของวีดีโอ (video representation)

1. ตัวแทนแบบใช้เฟรมหลักที่เรียงตามลำดับ (Sequential Key Frame Representation)

หลังจากที่ได้ช็อต (shot) และเฟรมหลัก (key frame) , ตัวแทนของวีดีโอ (video representation) อย่างง่ายๆ และที่เห็นได้ชัด คือ เฟรมหลัก (key frame) ของวีดีโอที่วางนอนเรียงกันอย่างเป็นลำดับ เทคนิคง่ายๆ อันนี้จะทำได้ดีเมื่อมีเฟรมหลัก (key frame) น้อยๆ แต่เมื่อวีดีโอคลิป (video clip) มีความยาว เทคนิคนี้จะไม่ถูกขนาดมาตรส่วน เนื่องจากมันไม่ได้จับเอาข้อมูลข่าวสารฝังเข้าไปภายในวีดีโอคลิป (video clip) ด้วย ยกเว้นเรื่องเวลา

2. ตัวแทนแบบใช้กลุ่มเป็นพื้นฐาน (Group Based Representation)

เพื่อให้บรรลุถึงความหมายของตัวแทนของวีดีโอ (video representation) เมื่อวีดีโอมีขนาดยาว ช็อต (shot) ที่มีความสัมพันธ์กันจะถูกเชื่อมเข้าไปในกลุ่ม แชงจี เอ็ท อัล (Zhange et al) แบ่งสายธารวีดีโอ (video stream) ทั้งหมดเป็นการแยกส่วนวีดีโอ (video segment) แบบทวีคูณ แต่ละส่วน (segment) ที่ถูกแบ่งออกนั้นจะบรรจุด้วยจำนวนที่เท่าๆ กันของช็อต (shot) ที่ต่อเนื่องกันตลอด ในแต่ละส่วน (segment) จะถูกแบ่งเป็นส่วนย่อย (sub–segment) ลงไปอีก ด้วยเหตุนี้จึงเป็นการสร้างตัวแทนของวีดีโอ (video representation) เป็นรูปแบบโครงสร้างต้นไม้ (tree) ซอง เอ็ท อัล (Zhong et al) ได้เสนอความคิดเห็นลำดับขั้นของวีดีโอแบบใช้การจับกลุ่มเป็นพื้นฐาน (cluster-based) ซึ่งช็อต (shot) ต่างๆ จะถูกยึดจับเป็นกลุ่มอยู่บนเนื้อหาที่มองเห็นของพวกมัน วิธีนี้จะเป็นการสร้างตัวแทนของวีดีโอ (video representation) ในรูปแบบโครงสร้างของต้นไม้ (tree) ขึ้นมาอีกครั้งหนึ่ง

3. ตัวแทนแบบใช้ฉากเป็นพื้นฐาน (Scene Based Representation)

เพื่อการนำเสนอการเข้าถึงวีดีโอที่ดีกว่าให้แก่ผู้ใช้ การสร้างตัวแทนของวีดีโอ (video representation) ในเชิงความหมาย (semantic) จึงถูกต้องการ มันไม่ใช่สิ่งแปลกที่ภาพยนตร์สมัยใหม่จะบรรจุช็อต (shot) หรือเฟรมหลัก (key frame) นับพัน เช่น หนังเรื่องเทอร์มิเนเตอร์ 2 -จั๊ดเม้นทเดย์ (Terminator 2-Judgment Day) จะมี 300 ช็อต (shot) ใน 15 นาที ของวีดีโอตอนหนึ่ง และภาพยนตร์เรื่องนี้มีความยาว 139 นาที เฟรมหลัก (key frame) ที่นำเสนอในแบบลำดับ 1 มิติ อย่างง่ายๆ สำหรับวีดีโอพื้นฐาน (หรือแม้แต่โครงสร้างแบบต้นไม้ (tree) ที่วางแผ่ออกไปในลำดับกลุ่ม (group)) ก็แทบจะไม่มีความหมายเลย ที่สำคัญกว่านั้นผู้คนจะดูวีดีโอโดยใช้ฉากในเชิงความหมาย (semantic scene) ของมันมากกว่าการใช้ช็อต (shot) หรือเฟรมหลัก (key frame) ทางกายภาพ (physical shots or key frame) ในขณะที่ช็อต (shot) กำลังสร้างบล็อค (block) ของฉากวีดีโอ (scene) ก็จะถ่ายทอดความหมายในเชิงสำนวน (semantic meaning) ไปยังผู้ชม ความไม่ต่อเนื่องของช็อต (shot) ต่างๆ ถูกเอาชนะ โดยความต่อเนื่องของฉาก (scene) การสร้างสารบัญของวีดีโอที่ระดับฉาก (scene) เป็นความสำคัญขั้นพื้นฐานสำหรับการค้นหาวีดีโอ และการเรียกคืนกลับวีดีโอ

กราฟการเปลี่ยนฉาก (Scene transition graph: STG) ของตัวแทนของวีดีโอ (video representation) ถูกเสนอความเห็นและถูกสร้างขึ้น ลำดับของวีดีโอจะถูกแบ่งเป็นตอนๆ เป็นสิ่งแรก เข้าไปในช็อต (shot) แล้วช็อต (shot) จะถูกจัดกลุ่มโดยการใช้การจัดกลุ่มที่ถูกระงับด้วยเวลา (time-constrained clustering) แล้วเอสทีจี (STG) จะถูกสร้างบนพื้นฐานของการเคลื่อนที่ของเวลา (time flow)

2.9 การทำดัชนีวีดีโอโดยวิธี AVI (Adaptive Video Indexing: AVI)

ข้อมูลของวีดีโอจะประกอบไปด้วยภาพที่ต่อเนื่องเป็นลำดับ ซึ่งวีดีโอที่มีเนื้อหาคล้ายคลึงกันโดยปกติจะบรรจุภาพที่คล้ายคลึงกัน

สมการต่อไปนี้ จะถูกใช้บ่อยเพื่อใช้ในการอธิบายว่า วีดีโอ อ้างถึงเซตความคล้ายคลึงกันของภาพได้อย่างไร

$$D_{I_x} = \{(X_i f_i) \mid X_i \in IR^p, i = 1,2,\ldots,N\} \quad (2.1)$$

D_{I_x} = ตัวอธิบายองค์ประกอบเบื้องต้น (Primary content descriptor) สำหรับวีดีโอช่วงที่ I_x

$X_{i,}$ = เวคเตอร์ที่บ่งบอกข้อมูลของแต่ละวิดีโอ ซึ่งเก็บจากค่า histogram

f_i = จำนวนเฟรม (frame) ของวีดีโอที่ i อยู่ภายใน I_x

N = จำนวนทั้งหมดของเฟรม (frame) ในวิดีโอที่มีใน Database

I_x = ช่วงของวีดีโอ ที่อาจจะเป็นช็อต (Shot), ฉาก (Scene) และคลิปเนื้อเรื่อง (Story Clips)

แบบจำลองเอวีไอ (AVI Model) ถูกนิยามโดยความน่าจะเป็นของการค้นหาแบบจำลองของเฟรม (Frame model) หรือ ตัวแทนของการทำงาน (Code vector) M_i ในวีดีโอที่ถูกนำเข้ามา ซึ่งถูกกำหนดให้เป็นสมการอย่างง่ายดังนี้

$$P(M_i) = (N \times N_1)^{-1} \sum_{j=1}^{N \times N_1} I(l_j = I_{M_i}) \qquad (2.2)$$

l_{M_i} = ป้ายชื่อ (Label) ของแบบจำลองเวคเตอร์ (Code vector) M_i

$N \times N_1$ = จำนวนรวมของป้ายชื่อ (Label) ที่ใช้สืบค้นวีดีโอที่ถูกนำเข้ามา

ซึ่งฟังก์ชัน $I(\bullet) = 1$ ถ้าอาร์กิวเม้นท์ (Argument) เป็นจริง และจะเท่ากับ 0 ถ้าเป็นกรณีอื่น

ให้ M เป็นเซตของ Code vector โดย $M = [M_1, \ldots, M_i, \ldots, M_T]$, $M_i \in IR^p$ ให้ T เป็นจำนวนของ Code Vector ทั้งหมด ซึ่งแบบจำลองเหล่านี้จะถูกสร้างขึ้นและปรับปรุงให้ดีขึ้นโดยการจัดกลุ่มเวคเตอร์ เริ่มต้นจากการหาค่าเฉลี่ยของค่า Histogram ทั้งหมดจากไฟล์วิดีโอ จะได้ค่า Code Vector 1 ตัว ที่จะนำมาใช้แบ่งต่อให้ได้ค่าเป็นไบนารี่ (1, 2, 4, 8, 16, 32, 64, …, 2^n) ซึ่งในการแบ่งแต่ละขั้นจะทำการแบ่งกลุ่มค่า Vector (Training set) ให้เป็นกลุ่มๆ โดยจะแบ่งค่า Code Vector จนกว่าจะได้ค่าที่กำหนดแล้วจึงทำดัชนีของวิดีโอช็อต โดยการนำค่า Histogram มาเป็นตัวเปรียบเทียบจัดกลุ่ม Index ซึ่งการทำ Index แบ่งเป็น 2 แบบ คือ ทำการเก็บดัชนี โดยใช้ค่าที่ใกล้ที่สุด (1 Label) และ เก็บแบบ ใช้ค่าที่ใกล้ที่สุดและค่าที่ใกล้รองลงมา (2 Labels)

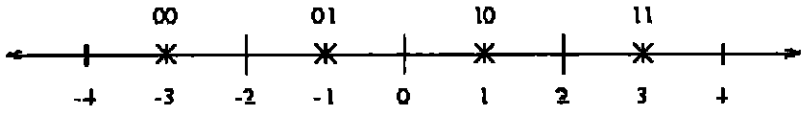
2.10 ทฤษฎี Vector Quantization [4]

2.10.1 บทนำ

Vector Quantization (VQ) คือกระบวนการบีบอัดข้อมูล บนพื้นฐานของการเข้ารหัส ซึ่งก็คือ fix-to-fix length algorithm ก่อนหน้านี้ รูปแบบของตัวประเมินค่าเวคเตอร์ (VQ) ถูกนำมาพิจารณาเพื่อนำมาใช้ในการแก้ปัญหาที่ถูกคัดค้านไว้ เนื่องมาจากความจำเป็นในการอินทิเกรตหลายๆมิติ ในปี 1980 Linde, Buzo และ Gray (LBG) ได้เสนออัลกอริทึมในการออกแบบ VQ บนพื้นฐานของ Training sequence ในการใช้ Training sequence จะหลีกเลี่ยงความจำเป็นในการอินทิเกรตหลายๆ มิติ VQ ถูกออกแบบมาเพื่อใช้ในอัลกอริทึมนี้ ซึ่งได้กล่าวไว้ในบทความของ LBG-VQ

2.10.2 เบื้องต้น

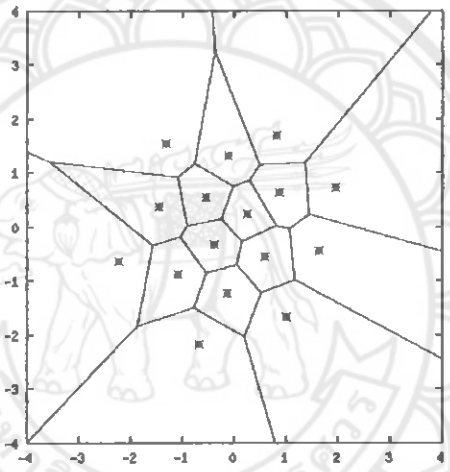
VQ เป็นการประมาณค่า ซึ่งคล้ายกับ rounding-off (จำนวนเต็มที่ใกล้ที่สุด) ตัวอย่างของ VQ 1 มิติ เป็นดังรูป



รูปที่ 2.22 VQ 1 มิติ

ในทุกๆ จำนวนที่มีค่าน้อยกว่า -2 จะถูกประมาณค่าเป็น -3 ทุกๆ จำนวนที่มีค่าอยู่ระหว่าง -2 ถึง 0 จะถูกประมาณค่าเป็น -1 ทุกๆ จำนวนที่มีค่าอยู่ระหว่าง 0 ถึง 2 จะถูกประมาณค่าเป็น +1 และทุกๆ จำนวนที่มีค่ามากกว่า 2 จะถูกประมาณค่าเป็น +3 แสดงว่าค่าที่ใช้ในการประมาณจะแทนด้วย 2 bits นั่นคือ มีอัตราส่วน 2 bits/dimension

ตัวอย่างของ VQ 2 มิติ เป็นดังรูป



รูปที่ 2.23 VQ 2 มิติ

ทุกๆ คู่ของจำนวนต่างๆ จะถูกจัดอยู่ในบริเวณที่ได้กำหนดไว้ โดยประมาณค่าด้วย รูปดาวสีแดงที่เกี่ยวข้องกับบริเวณดังกล่าว ซึ่งในที่นี้ มีทั้งหมด 16 บริเวณ และดาว 16 ดวง แต่ละดวงจะแทนด้วย 4 bits ดังนั้น 2 มิติ มี 4 bits VQ ดังนั้น อัตราส่วน ก็คือ 2 bits/dimension เช่นกัน

จากตัวอย่างข้างต้น ดาวสีแดง เรียกว่า Codevectors และบริเวณจะถูกกำหนดด้วยเส้นสีน้ำเงิน ซึ่งเรียกว่า encoding region เซตของ codevector เรียกว่า codebook และเซตของ encoding region เรียกว่า partition

2.10.3 การแก้ปัญหา (Design Problem)

VQ Design Problem กำหนดไว้ดังนี้ นำค่า vector source, distortion measure และจำนวนของ code vectors มาหา codebook (set ของดาวสีแดง) และ partition (set ของเส้นสีน้ำเงิน) จะได้ผลลัพธ์ที่เป็นตัวที่มีค่าเฉลี่ยของ distortion ที่น้อยที่สุด

ให้ Training Sequence ประกอบด้วย source vector จำนวน M ค่า

$$\tau = \{X_1, X_{2,\ldots,} X_M\} \tag{2.3}$$

Training sequence นี้ หามาจากฐานข้อมูลใหญ่ๆ อันหนึ่ง ยกตัวอย่าง เช่น ถ้าแหล่งข้อมูลคือสัญญาณจากการพูด จะได้ Training sequence จะได้มาจากการบันทึกบทสนทนาทางโทรศัพท์ M จะถูกสมมติให้มีหลายค่า ดังนั้น ลักษณะข้อมูลทางสถิติของแหล่งที่มาจะถูกแบ่งกลุ่มด้วย Training sequence สมมติให้ source vector เป็น k มิติ ยกตัวอย่างเช่น

$$X_m = \{X_{m,1}, X_{m,2,\ldots,} X_{m,k}\}, \qquad m = 1, 2, \ldots, M \tag{2.4}$$

ให้ N เป็นจำนวนของ codevectors และให้

$$C = \{c_1, c_2, \ldots, c_N\} \tag{2.5}$$

แทน codebook แต่ละ codevector มี k มิติ เช่น

$$c_n = (c_{n,1}, c_{n,2}, \ldots, c_{n,k}), \quad n = 1, 2, \ldots, N \tag{2.6}$$

ให้ S_n เป็น encoding region ซึ่งเกี่ยวข้องกับ codevector C_n และให้

$$P = \{S_1, S_2, \ldots, S_N\} \tag{2.7}$$

ใช้แทน partition ถ้า source vector X_m อยู่ใน encoding region S_n ดังนั้น การประมาณค่า (แทนด้วย Q(Xm)) คือ C_n

$$Q(X_m) = c_n, \qquad \text{if } X_m \in S_n \tag{2.8}$$

กล่าวถึง squared-error distortion measure ค่าเฉลี่ยของ distortion ได้มาจาก

$$D_{ave} = \frac{1}{Mk} \sum_{m=1}^{M} \| X_m - Q(X_m) \|^2 \tag{2.9}$$

เมื่อ $\| e \|^2 = e_1^2 + e_2^2 + ... + e_k^2$ ในการออกแบบการแก้ปัญหา ได้กำหนดไว้ดังนี้ คือ รับค่า τ และ N หาค่า C และ P จะได้ค่า D_{ave} ที่มีค่าน้อยที่สุด

2.10.4 หลักการที่ดีที่สุด (Optimality Criteria)

ถ้า C และ P เป็นวิธีการแก้ปัญหาให้กับกระบวนการหาค่าที่น้อยที่สุด นั่นคือ ต้องใช้ได้กับทั้ง 2 หลักการ ดังนี้

- **Nearest Neighbor Condition**

$$\mathbf{S_n} = \{x : \| x - c_n \|^2 \leq \| x - c_{n'} \|^2 \ \forall n' = 1,2,...,N\} \tag{2.10}$$

เงื่อนไขนี้ กล่าวว่า encoding region S_n ควรจะประกอบด้วย vector ทั้งหมดที่ใกล้กับค่า C_n มากกว่า codevector ตัวอื่นๆ สำหรับเวคเตอร์เหล่านั้น จะถูกแบ่งด้วยเส้นแบ่งเขต (เส้นสีน้ำเงิน)

- **Centroid Condition**

$$\mathbf{C_n} = \frac{\sum_{x_m \in S_n} X_m}{\sum_{x_m \in S_n} 1} \tag{2.11}$$

สมการนี้ กล่าวว่า Codevector C_n ควรจะเป็นค่าเฉลี่ยของ training vector ทั้งหมดที่อยู่ใน encoding region S_n ในการดำเนินการ ควรจะแน่ใจว่า training vector ที่มีค่าน้อยที่สุดเป็นสมาชิกของแต่ละ encoding region ดังนั้น ตัวหารของสมการข้างต้น จะไม่เป็น 0

2.10.5 ออกแบบการทำงานโดยใช้อัลกอริทึม LBG (LBG Design Algorithm)

เป็นกระบวนการทำซ้ำ ซึ่งได้เลือก 2 หลักการข้างต้นมาแก้ปัญหา กระบวนการนี้ต้องการค่า codebook เริ่มต้น $C^{(0)}$ ค่า codebook เริ่มต้นนี้ ได้มาจากกระบวนการ Splitting ในกระบวนการนี้ ค่า codevector เริ่มต้น จะเซตให้เป็นค่าเฉลี่ยของ training sequence ทั้งหมด codevector นี้ จะถูกแตกออกเป็น 2 เวคเตอร์ ในกระบวนการทำซ้ำ จะดำเนินการด้วยเวคเตอร์ 2 ตัวนี้ โดยกำหนดให้เป็นค่า codebook เริ่มต้น codevector 2 ตัวสุดท้าย จะถูกแตกออกเป็น 4 เวคเตอร์ และจะดำเนินการซ้ำไปเรื่อยๆ จนได้ค่าของ codevector ที่เหมาะสม กระบวนการดังกล่าว เป็นดังนี้

LBG Design Algorithm

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

$$c_1^* = \frac{1}{M}\sum_{m=1}^{M} x_m \tag{2.12}$$

คำนวณหาค่า

$$D_{ave}^* = \frac{1}{Mk}\sum_{m=1}^{M} \| x_m - c_1^* \|^2 \tag{2.14}$$

3. Splitting : i = 1, 2, …, N ให้

$$c_i^{(0)} = (1+\varepsilon)c_i^*, \quad c_{N+1}^{(0)} = (1+\varepsilon)c_i^* \tag{2.15}$$

กำหนด N = 2N

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

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

$$\| x_m - c_n^{(i)} \|^2 \tag{2.16}$$

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

$$Q(x_m) = c_{n^*}^{(i)} \tag{2.17}$$

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

$$c_n^{(i+1)} = \frac{\sum_{Q(x_m)=c_n^{(i)}} x_m}{\sum_{Q(x_m)=c_n^{(i)}} 1} \tag{2.18}$$

iii. กำหนดให้ i = i+1

iv. หาค่า

$$D_{ave}^{(i)} = \frac{1}{Mk}\sum_{m=1}^{M} \| x_m - Q(x_m) \|^2 \tag{2.19}$$

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

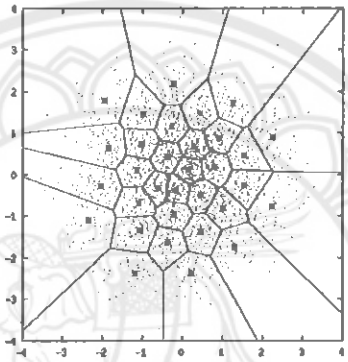
vi. กำหนด $D_{ave}^{*} = D_{ave}^{(i)}$ สำหรับค่า n = 1, 2, ..., N กำหนดให้

$$c_n^* = c_n^{(i)} \tag{2.20}$$

เป็นค่า codevector ตัวสุดท้าย

5. ทำซ้ำในขั้นตอนที่ 3 และ 4 จนกระทั่งได้ตัวเลขของ codevector ที่เหมาะสม

2.10.6 การทำเวคเตอร์ควอนไตซ์แบบ 2 มิติ (Two-Dimensional Animation)



รูปที่ 2.24 Two-Dimensional VQ

- แหล่งที่มาของรูปข้างต้น คือ memoryless Gaussian เริ่มต้นจาก ค่ากลาง 0 และความแตกต่างกันของหน่วย
- จุดเล็กๆ สีเขียว คือ training vector มี 4096 จุด
- LBG design algorithm ดำเนินการด้วย $\varepsilon = 0.001$
- กระบวนการนี้รับรองวิธีการเจาะจงพื้นที่ได้ดีที่สุด
- ขนาดของ training sequence ควรจะมีขนาดใหญ่เพียงพอ ซึ่งน่าจะมีค่าเป็น $M \geq 1000N$

2.11 การเปรียบเทียบ [12]

เมื่อต้องการหาความคล้ายคลึงของช็อตวิดีโอต้นแบบกับช็อตวิดีโอที่อยู่ในฐานข้อมูล จะต้องทำการนำข้อมูลของช็อตวิดีโอต้นแบบมาเปรียบเทียบกับข้อมูลช็อตวิดีโอในฐานข้อมูล โดยการคำนวณหาค่าระยะห่าง (Distance) ระหว่างช็อตวิดีโอต้นแบบกับช็อตวิดีโอในฐานข้อมูล ซึ่งใช้อัลกอริทึมในการเปรียบเทียบ คือ Cosine Similarity

Cosine Similarity

การหาระยะห่าง (Cosine Similarity) ระหว่างจุด P =(p_1,p_2, …,p_n) และ Q = (q_1,q_2,… ,q_n) จะได้ว่า

$$d(P,Q) = \frac{\vec{p}_i \cdot \vec{q}_i}{|\vec{p}_i||\vec{q}_i|} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^{\,2}}\sqrt{\sum_{i=1}^{n} q_i^{\,2}}} \qquad (2.21)$$

d(P,Q) = ระยะห่าง (Cosine Similarity)

p_i = เป็นค่าดัชนี (Index) ของรูปภาพต้นแบบ

q_i = เป็นค่าดัชนี (Index) ของรูปภาพในฐานข้อมูล

2.12 การจัดเรียงลำดับข้อมูล (Sorting) [13]

การเรียงลำดับข้อมูล (Sorting) คือการจัดเรียงข้อมูลให้มีการเรียงลำดับตามความต้องการ โดยจะเป็นการเรียงลำดับข้อมูลจากน้อยไปหามากหรือจากมากไปหาน้อยข้อมูลที่ทำการจัดเรียงอาจเป็นตัวเลข เช่น การเรียงลำดับคะแนนของนักเรียนจากคะแนนสูงสุดไปหาคะแนนต่ำสุดหรือข้อมูล ที่ทำการจัดเรียงอาจเป็นตัวอักษร เช่น การเรียงรายนามผู้ใช้โทรศัพท์ เป็นต้น

ประโยชน์ในการจัดเรียงข้อมูลก็คือ เพื่อให้สามารถเข้าถึงข้อมูลได้อย่างมีประสิทธิภาพ กล่าวคือ รวดเร็ว ถ้าหากมีข้อมูลมากๆ แล้วไม่ทำการเรียงลำดับข้อมูลก็จะทำให้เสียเวลาในการ ค้นหาข้อมูลนั้น เราจึงต้องมีการเรียงลำดับข้อมูลกันก่อนที่จะดึงข้อมูลมาใช้ถึงแม้จะต้องเสีย เวลาในการจัดเรียงไปบ้าง แต่การค้นหาข้อมูลจากข้อมูลที่จัดเรียงไว้แล้ว กับที่ไม่ได้จัดเรียง มี ประสิทธิภาพที่ต่างกันเป็นอย่างมาก

2.12.1 การเรียงลำดับอย่างมีประสิทธิภาพ

โดยทั่วไปการวัดประสิทธิภาพการทำงานของโปรแกรมหรือขั้นตอนวิธีพิจารณาจากเวลาและเนื้อที่ในความจำหลักที่ใช้ โปรแกรมที่เขียนขึ้นเพื่อให้โปรแกรมทำงานเร็วมักต้องใช้เนื้อที่ในความจำหลักมาก และในทางตรงกันข้ามโปรแกรมที่ใช้เนื้อที่ในความจำหลักน้อยที่สุดมักต้องใช้เวลาในการทำงานนานที่สุดด้วย ดังนั้นจึงเป็นไปได้ยากที่เราจะเขียนโปรแกรมที่ทำงานได้เร็วและใช้เนื้อที่ในความจำหลักให้น้อยที่สุด ด้วยเหตุนี้ในการทำงานจริงผู้เขียนโปรแกรมอาจจะต้องเลือกว่าโปรแกรมที่มีประสิทธิภาพนั้นต้องการให้เอนเอียงไปทางใด

หลักเกณฑ์ในการพิจารณาเพื่อเลือกวิธีการเรียงลำดับที่ดีและเหมาะสมกับระบบงาน เพื่อให้ประสิทธิภาพในการทำงานสูงสุดควรจะต้องคำนึงถึงสิ่งต่าง ๆ ดังต่อไปนี้

(1) เวลาและแรงงานที่ต้องใช้ในการเขียนโปรแกรม โดยพิจารณาว่าโปรแกรมที่เขียนจำเป็นต้องใช้เร่งด่วนขนาดไหน มีจำนวนผู้เขียนเพียงพอที่จะเขียนโปรแกรมให้เสร็จตามที่ต้องการหรือไม่ ถ้ามีความจำเป็นต้องการใช้โปรแกรมอย่างรีบด่วนและจำนวนผู้เขียนมีจำนวนน้อยไม่เพียงพอกับระบบงานที่ใหญ่มาก อาจจะเลือกวิธีการที่ไม่ซับซ้อนเพื่อให้เขียนโปรแกรมได้ง่ายเสร็จสมบูรณ์ในเวลาที่รวดเร็วขึ้น มิฉะนั้นอาจจะไม่สามารถเขียนโปรแกรมได้เสร็จเร็วตามที่กำหนดก็เป็นได้

(2) เวลาที่เครื่องคอมพิวเตอร์ต้องใช้ในการทำงานตามโปรแกรมที่เขียน ในระบบงานบางงานอาจจะต้องการให้การประมวลผลใช้เวลาน้อยที่สุด เพื่อประหยัดค่าใช้จ่าย หรือในระบบงานนั้นเป็นงานบริการที่ต้องการตอบสนองลูกค้าอย่างทันทีทันใด กรณีนี้ควรเลือกใช้วิธีการจัดเรียงลำดับที่ใช้เวลาในการประมวลผลน้อยที่สุด

(3) จำนวนเนื้อที่ในหน่วยความจำหลักมีเพียงพอหรือไม่ ถ้าเครื่องคอมพิวเตอร์ที่ใช้สำหรับงานนั้น ๆ มีขนาดหน่วยความจำหลักจำกัด เราควรเลือกใช้วิธีการเรียงลำดับแบบที่ประหยัดเนื้อที่ในหน่วยความจำ แต่ถ้ามีหน่วยความจำเหลือเฟือเราอาจจะใช้วิธีเรียงลำดับแบบใดก็ได้ที่เขียนโปรแกรมเสร็จเร็วที่สุด

อย่างไรก็ตามหากโปรแกรมเรียงลำดับที่เขียนขึ้นนั้นต้องการใช้งานเพียงครั้งเดียว เนื้อที่ในหน่วยความจำหลักมีอย่างเหลือเฟือ และเวลาที่ใช้ในการประมวลผลตามโปรแกรมนั้นมีเวลาไม่จำกัด นั่นคือทรัพยากรต่าง ๆ มีอย่างเหลือเฟือ ผู้เขียนโปรแกรมไม่จำเป็นต้องไปคิดหาวิธีการที่ดีที่สุดเพื่อทำให้การทำงานของเครื่องคอมพิวเตอร์มีประสิทธิภาพสูงสุด ซึ่งทำให้เสียเวลาและแรงงานไปโดยเปล่าประโยชน์ โดยความเป็นจริงแล้วไม่มีวิธีการเรียงลำดับแบบใดดีกว่าหรือเหนือกว่าวิธีการเรียงลำดับแบบอื่น ๆ ทั้งหมด ทั้งนี้ขึ้นอยู่กับระบบงานและเหตุการณ์เฉพาะอย่างของแต่ละระบบงานมากกว่า ที่จะเป็นตัวกำหนดว่าควรเลือกใช้วิธีการเรียงลำดับแบบใดที่เหมาะสมกับระบบงานและมีประสิทธิภาพในการทำงานสูงสุด

2.12.2 การเรียงลำดับแบบเลือก (Selection sort)

มีหลักการดังนี้ คือ การเลือกข้อมูลที่มีค่าที่มากที่สุดในแถวลำดับ และนำค่านั้นไปเปลี่ยนที่กับข้อมูลที่บรรจุ อยู่ในแถวลำดับช่องแรก ต่อมามีการเลือกค่าที่มากรองลงมา และนำค่านั้นไปแลกกับข้อมูลในแถวลำดับช่องที่สอง ขั้นตอนนี้เหมือนกันกับขั้นตอนแรก ยกเว้นจะไม่มีการเปรียบเทียบกับข้อมูลในแถวลำดับช่องแรก เพราะรู้แล้วว่าเป็นค่าที่มากที่สุด ต่อจากนั้นจะมีการดำเนินงานแบบเดิม คือ เลือกตัวที่มีค่ามากที่สุดในบรรดา ข้อมูลที่เหลือแล้วมีการสลับที่ จนกระทั่งค่าทั้งหมดที่อยู่ในแถวลำดับมีการเรียงลำดับ และจะพบว่าที่ ส่วนแรกของแถวลำดับ ประกอบด้วยค่าที่สูงสุดและที่ปลายสุดของแถวลำดับมีค่าน้อยที่สุด ตัวอย่างของการเรียงลำดับแบบเลือก แสดงในตารางที่ 2.2

ขั้นตอนการเรียงลำดับข้อมูลแบบเลือก (Selection sort) มีดังนี้

1. เริ่มจากการนำค่าแรก มาเปรียบเทียบกับค่าถัดไป ซึ่งก็คือค่าที่สองในแถวลำดับ ถ้าค่าถัดไปมีค่ามากกว่าค่าแรก จะทำการสลับที่ค่าทั้งสอง

2. ดำเนินการเช่นเดียวกับขั้นตอนที่ 1 โดยการนำค่าถัดไปมาเปรียบเทียบกับค่าตัวถัดไป (ค่าที่สาม) ดำเนินการไปเช่นนี้จนครบทุกค่าของแถวลำดับ

3. ดำเนินการเช่นเดียวกับขั้นตอนที่ 1 กับค่าที่สาม ค่าที่สี่ ไปเรื่อยๆ จนถึงค่ารองสุดท้ายของแถวลำดับ ก็จะได้แถวลำดับที่เรียงลำดับจากมากไปน้อยตามความประสงค์

ตารางที่ 2.2 แสดงการเรียงลำดับข้อมูลแบบเลือก (Selection sort)

1)	**รอบ 1**	8	20	30	3	4
		20	8	30	3	4
		30	8	20	3	4
		30	8	20	3	4
2)	**รอบ 2**	30	8	20	3	4
		30	20	8	3	4
		30	20	8	3	4
3)	**รอบ 3**	30	20	8	3	4
		30	20	8	3	4
4)	**รอบ 4**	30	20	8	3	4
	ผลลัพธ์	30	20	8	4	3

บทที่ 3
วิธีดำเนินการโครงงานวิศวกรรม

การดำเนินการโครงงาน แบ่งออกเป็น 4 ส่วน คือ

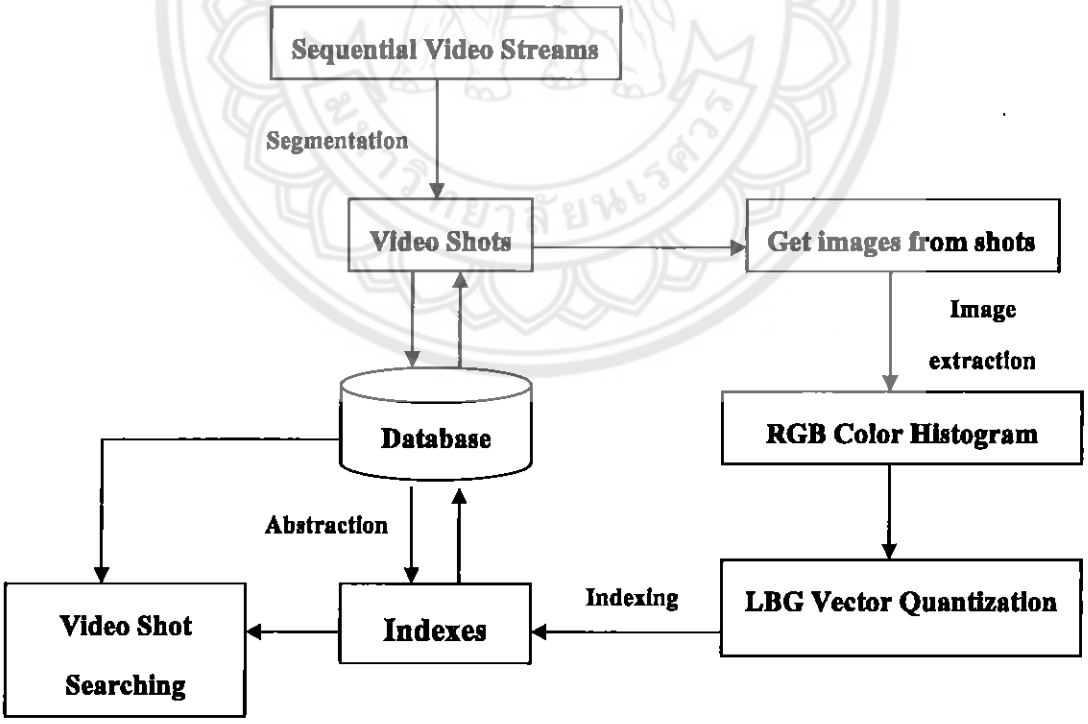
1. ส่วนของการแยกช็อตจากไฟล์วิดีโอ เพื่อนำไปใช้ในส่วนของโปรแกรม โดยในส่วนนี้จะใช้โปรแกรมสำเร็จรูป คือ Video Editor 7.0 จะไม่ทำการเขียนโปรแกรมเอง เนื่องจากอาจจะได้ผลลัพธ์ของการแยกช็อตที่ไม่ได้ประสิทธิภาพนักซึ่งอาจจส่งผลถึงส่วนของโปรแกรมและระบบการค้นหาได้

2. ส่วนของโปรแกรม โดยส่วนของโปรแกรมนี้จะพัฒนาโดยใช้โปรแกรม Visual Studio 2005

3. ส่วนของฐานข้อมูล ซึ่งจะใช้ Microsoft Access 2003 เป็นฐานข้อมูล โดยใช้ในการเก็บค่าดัชนีที่ได้จากไฟล์วิดีโอ

4. ส่วนของระบบการค้นหา ในส่วนนี้เราจะใช้ช็อตที่ได้ถูกแยกจากโปรแกรมสำเร็จรูปข้างต้นมาใช้อ้างอิงเป็นไฟล์ต้นแบบ และใช้ดัชนีจากส่วนของโปรแกรมมาใช้ในการค้นหาช็อตที่ใกล้เคียงมากที่สุด 8 อันดับแรก

โดยสามารถสรุปหลักการการทำงานได้ดังนี้



รูปที่ 3.1 แสดงขั้นตอนการทำงานของโปรแกรม

3.1 การแยกช็อต (Video Shot Segmentation)

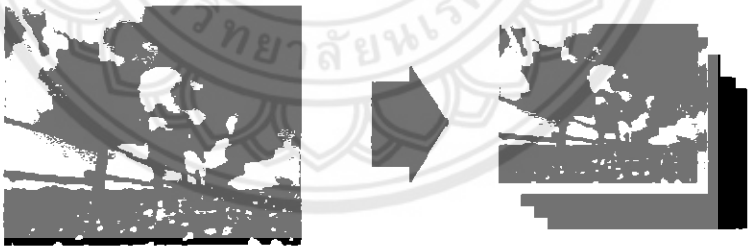
ในส่วนนี้ ผู้จัดทำใช้โปรแกรมสำเร็จรูป Video Editor 7.0 เข้ามาทำการแยกช็อต ไม่ได้อ้างอิงในส่วนของการเขียนโปรแกรม เนื่องจากหากทำการเขียนโปรแกรมในส่วนนี้เอง จะได้ผลลัพธ์ที่ไม่ได้ประสิทธิภาพนัก ซึ่งในขั้นตอนของระบบการค้นหา จะนำส่วนนี้ไปใช้ด้วยโปรแกรมจะทำการเปรียบเทียบช็อตวิดีโอจากหลายๆ ไฟล์ที่แตกต่างกัน เพื่อหาช็อตที่มีความคล้ายคลึงหรือใกล้เคียงกับช็อตต้นแบบมากที่สุด ดังนั้น ต้องทำการเลือกไฟล์วิดีโอที่แตกต่างกันประมาณ 3 ไฟล์ขึ้นไป โดยทำการแยกช็อตทีละไฟล์

3.2 การหาดัชนี (Indexing)

ในขั้นตอนการหาดัชนีแบ่งออกเป็น 4 ขั้นตอน โดยเริ่มจากให้ผู้ใช้ทำการโหลดช็อตวิดีโอที่ผ่านกระบวนการการแยกช็อตจากขั้นตอนข้างต้น เข้าสู่โปรแกรม ในที่นี้ทำการเก็บดัชนี 2 แบบ คือ แบบ one codevector และแบบ two codevector ซึ่งในแต่ละขั้นตอนมีรายละเอียดดังนี้

3.2.1 ดำเนินการแยกช็อตวิดีโอต้นแบบเป็นเฟรมรูปภาพ

ช็อตวิดีโอที่ถูกป้อนเข้าสู่โปรแกรมจะต้องนำมาทำการแยกเป็นเฟรมรูปภาพก่อน เนื่องจากไฟล์วิดีโอเป็นไฟล์ภาพเคลื่อนไหวไม่สามารถที่จะนำมาถอดข้อมูลเพื่อหาข้อมูลของฮีสโตแกรมได้ ซึ่งไฟล์วิดีโอนั้นประกอบด้วยภาพนิ่งจำนวนหลายๆ เฟรมที่เรียงต่อกัน จึงต้องทำการแยกเฟรมรูปภาพเหล่านั้นออกมา เพื่อทำการเก็บค่าเม็ดสีในแต่ละภาพ



รูปที่ 3.2 (ก) ไฟล์วิดีโอ **รูปที่ 3.2 (ข)** รูปภาพที่ถูกแยกเป็นเฟรม

รูปที่ 3.2 แสดงผลของการแยกเฟรมจากไฟล์วิดีโอ

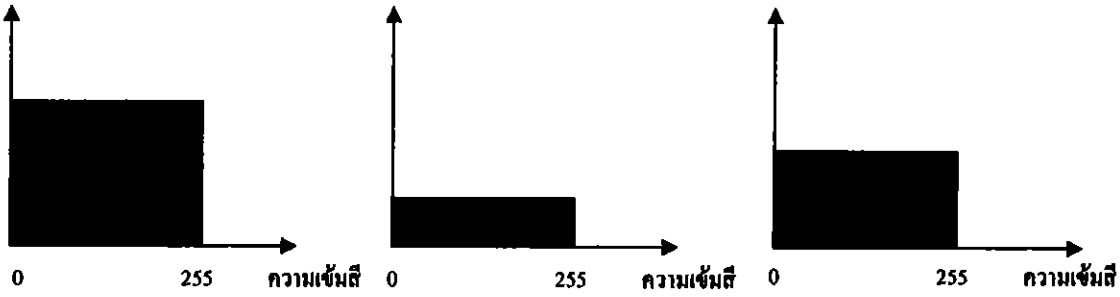
3.2.2 ทำการถอดข้อมูลภาพที่ได้ให้อยู่ในรูปข้อมูลฮีสโตแกรม

การหาค่า RGB Color Histogram จะทำการวิเคราะห์ค่าสีแต่ละ Pixel จะทำให้ได้ค่าสีที่ออกมาเป็นฮีสโตแกรม (การสะสมจำนวนของค่าเม็ดสี) ซึ่งในแต่ละรูปนั้นจะมีค่าฮีสโตแกรม 3 ค่า คือ

Red หมายถึง ค่าสีแดง ที่เป็นองค์ประกอบของภาพนั้น

Green หมายถึง ค่าสีเขียว ที่เป็นองค์ประกอบของภาพนั้น

Blue หมายถึง ค่าสีน้ำเงิน ที่เป็นองค์ประกอบของภาพนั้น



รูปที่ 3.3 วิเคราะห์หาค่าสีของแต่ละรูปภาพ

จากนั้น ทำการเก็บข้อมูลฮีสโตแกรมของแต่ละภาพ โดยระบุค่าสี R (แดง) G (เขียว) B (น้ำเงิน) ว่ามีจำนวนเม็ดสีเท่าไหร่ที่ได้จากการวิเคราะห์แต่ละ Pixel ของทั้งภาพ ซึ่งเริ่มตั้งแต่ 0 ถึง 255 เพื่อความสะดวกต่อการนำไปวิเคราะห์

3.2.3 นำข้อมูลฮีสโตแกรมเข้าสู่กระบวนการ LBG Vector Quantization

1. นำข้อมูลฮีสโตแกรมมาวิเคราะห์ ด้วยกระบวนการ LBG Vector Quantization โดยแทนจำนวนเฟรมทั้งหมด ซึ่งเรียกว่า Training Sequence ด้วย

$$\tau = \{X_1, X_2, \ldots, X_M\} \qquad (3.1)$$

โดยที่ τ แทน Training Sequence

$\mathbf{X_M}$ แทน เฟรมที่จะทำการโหลดเข้าสู่โปรแกรมด้วย X_M โดยที่ M คือจำนวนของเฟรม

ค่าภายใน X_M คือ ค่าสี R G และ B ของเฟรมนั้นโดยแทนด้วย

$$X_m = \{X_{m,1}, X_{m,2}, \ldots, X_{m,k}\} \ , m = 1, 2, .. , M \qquad (3.2)$$

, k คือ จำนวนมิติ

ยกตัวอย่างเช่น **ตัวอย่างที่ 1** ช็อต 1 ช็อตมีจำนวนเฟรมรูปภาพทั้งหมด 5 เฟรม

เฟรมที่ 1 มีค่าสี R จำนวน 256 ค่า มีค่าเท่ากับ 3, 9, 15, … , m,k

มีค่าสี G จำนวน 256 ค่า มีค่าเท่ากับ 43, 8, 209, … , m,k

มีค่าสี B จำนวน 256 ค่า มีค่าเท่ากับ 0, 23, 90, … , m,k

เฟรมที่ 2 มีค่าสี R จำนวน 256 ค่า มีค่าเท่ากับ 5, 119, 615, … , m,k

มีค่าสี G จำนวน 256 ค่า มีค่าเท่ากับ 83, 0, 28, … , m,k

มีค่าสี B จำนวน 256 ค่า มีค่าเท่ากับ 200, 19, 89, … , m,k

.

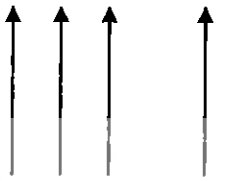
.

.

เฟรมที่ 5 มีค่าสี R จำนวน 256 ค่า มีค่าเท่ากับ 5, 119, 615, … , m,k

มีค่าสี G จำนวน 256 ค่า มีค่าเท่ากับ 83, 0, 28, … , m,k

มีค่าสี B จำนวน 256 ค่า มีค่าเท่ากับ 200, 19, 89, … , m,k



จากข้อมูลฮิสโตแกรมข้างต้นจะเขียนแทน X_m ด้วย

X_m = (ค่าสี R จากทุกเฟรมจำนวน 256 ค่า, ค่าสี G จากทุกเฟรมจำนวน 256 ค่า, ค่าสี B จากทุกเฟรมจำนวน 256 ค่า)

ดังนั้น จำนวนค่าสีทั้งหมดภายใน 1 เฟรม เท่ากับ 256 x 3 = 768 ค่า

2. ทำการหาสมาชิกภายใน codevector โดยการหาค่าเฉลี่ยของค่าสี R G และ B ของทุกๆ X_m หารด้วยจำนวนเฟรมทั้งหมดใน 1 ช็อต ซึ่งเรียกว่า codebook โดยใช้สูตร

$$c_1^* = \frac{1}{M}\sum_{m=1}^{M} x_m \qquad (3.3)$$

โดยที่ c_1^* คือ ค่า codebook

M คือ จำนวนเฟรมทั้งหมดใน 1 ช็อต

แทนสูตรข้างต้นด้วยการนำค่าของสมาชิกตัวแรกจากทุกๆ X_m มาหาผลรวม จากนั้นนำค่าของสมาชิกตัวที่สองจากทุกๆ X_m มาหาผลรวมเช่นกัน ทำการหาผลรวมของสมาชิกทีละตัวจากทุกๆ X_m จนครบสมาชิกทุกตัว แล้วแทนลงในเซตของ C_n

$$\mathbf{C = \{c_1, c_2, \dots , c_N\}} \tag{3.4}$$

โดยที่ N คือ จำนวน codebook ทั้งหมด

3. ทำการหาค่าเฉลี่ยของ Distortion โดยการนำสมาชิกในเซตของ X_m และสมาชิกในเซตของ C มาทำการเข้าสมการ

$$D^*_{ave} = \frac{1}{Mk}\sum_{m=1}^{M} \| x_m - c_1^* \|^2 \tag{3.5}$$

โดยที่ D^*_{ave} คือ ค่าเฉลี่ยของ Distortion

M คือ จำนวนเฟรมทั้งหมดใน 1 ช็อต

k คือ จำนวนสมาชิกใน X_m

4. ขั้นตอนการ Splitting เพื่อหาค่า codevector ค่าใหม่ จากสูตร

$$\begin{aligned} c_i^{(0)} &= (1+\varepsilon)c_i^*, \\ c_{N+1}^{(0)} &= (1+\varepsilon)c_i^* \end{aligned} \tag{3.6}$$

โดยที่ i = 1, 2, ..., N ในขั้นตอนนี้ ต้องทำการกำหนดค่า ϵ เพื่อใช้เป็นค่าคงที่ในการ Splitting ซึ่งจะได้ codevector ออกมาใหม่ 2 ค่า คือ $C_1^{(0)}$ และ $C_2^{(0)}$ โดยจะไม่นำ C_1^* มาพิจารณาแล้ว

5. ทำการจัดกลุ่มให้กับ X_m ทุกตัว โดยใช้ $C_1^{(0)}$ และ $C_2^{(0)}$ เป็นตัวเปรียบเทียบในการจัดกลุ่ม พิจารณาจากสูตร

$$\| x_m - c_n^{(i)} \|^2 \tag{3.7}$$

โดยเริ่มพิจารณาจาก $C_1^{(0)}$ ก่อน กับ X_m ทุกๆ เซต จากนั้นพิจารณาที่ $C_2^{(0)}$ กับ X_m ทุกๆ เซต เช่นกัน ตามลำดับ ทำการเปรียบเทียบค่าที่ได้จากการคำนวณจากสมการข้างต้นของทั้ง $C_1^{(0)}$ และ $C_2^{(0)}$ ว่าค่าที่ได้จาก $C_1^{(0)}$ หรือ $C_2^{(0)}$ มีค่าน้อยกว่ากัน ให้เลือก X_m นั้นเข้ากลุ่ม ยกตัวอย่างเช่น

กรณีที่ใช้ $C_1^{(0)}$ กับ X_1 ค่าที่ได้จาก $\| x_m - c_n^{(i)} \|^2$ มีค่าเท่ากับ 4.0408

กรณีที่ใช้ $C_2^{(0)}$ กับ X_1 ค่าที่ได้จาก $\| x_m - c_n^{(i)} \|^2$ มีค่าเท่ากับ 4.0263

จะเห็นว่า ค่าที่ได้จากการใช้ $C_2^{(0)}$ มีค่าน้อยกว่า เพราะฉะนั้น X_1 อยู่ในกลุ่มของ $C_2^{(0)}$ จากนั้นทำการจัดกลุ่มไปจนครบทุก X_m จากขั้นตอนนี้จะได้กลุ่มของ $C_1^{(0)}$ และกลุ่ม $C_2^{(0)}$ โดยมีสมาชิก คือ X_m ที่ได้ผ่านการจัดกลุ่มมาแล้ว นำค่าสมาชิก X_m ในกลุ่มของ $C_1^{(0)}$ และกลุ่ม $C_2^{(0)}$ มาทำการหา codevector ตัวใหม่ จากสูตร

$$c_n^{(i+1)} = \frac{\sum_{Q(x_m = c_n^{(i)})} x_m}{\sum_{Q(x_m = c_n^{(i)})} 1} \qquad (3.8)$$

โดยที่ $\sum_{Q(Xm)=cn(i)} Xm$ คือ ผลรวมของสมาชิกตัวที่ 1, ผลรวมของสมาชิกตัวที่ 2, ผลรวมของสมาชิกตัวที่ 3, ... ทำการหาผลรวมไปจนถึงสมาชิกตัวสุดท้ายของแต่ละเซต

$\sum_{Q(Xm)=cn(i)} 1$ คือ จำนวนสมาชิกในกลุ่มของ C นั้นๆ

จะได้ค่า codevector ตัวใหม่ นำ codevector ดังกล่าว ไปทำการหาค่า D_{ave} ใหม่ โดยพิจารณาค่า C ทีละตัวเช่นกัน และใช้ X_m ค่าเดิม จากขั้นตอนที่ 1 จากนั้นนำค่า D_{ave} ค่าใหม่มาทำการเปรียบเทียบกับค่า D_{ave} ค่าเก่า ด้วยสมการ

$$(D_{ave}^{(i-1)} - D_{ave}^{(i)}) / D_{ave}^{(i-1)} > \varepsilon \qquad (3.9)$$

หรือ

(D_{ave} ค่าเก่า - D_{ave} ค่าใหม่) / D_{ave} ค่าเก่า

หากผลลัพธ์ที่ได้มีค่ามากกว่า ϵ จะต้องกลับไปทำการหาค่า C ใหม่ เนื่องจากค่า C ตัวนี้ใช้ไม่ได้ แต่หากมีค่าน้อยว่า ϵ แสดงว่าค่า C นั้น ใช้ได้ โดยนำค่า C ที่ได้มาเก็บลงในเมตริกซ์ขนาด จำนวน codevector x 768 จากนั้นทำการหาค่า C ไปจนครบตามที่ได้กำหนดไว้ หรือจนกว่าจะหาไม่ได้

3.2.4 นำค่าเวคเตอร์ที่ได้มาทำการหาดัชนีและเก็บลงในฐานข้อมูล

จากขั้นตอน Vector Quantization เราจะได้จำนวนของ codevector ทั้งหมดที่ได้จากการกำหนดค่าของผู้ใช้ ซึ่งจะมีค่าเป็น 2^n (2, 4, 8, 16, ..., 2048) จากนั้นโปรแกรมจะทำการ split ค่า codevector ให้ได้ตามที่กำหนด หากจำนวนเฟรมมีมาก 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 ของช็อตวิดีโอแต่ละช็อตมาเปรียบเทียบกับ codevector ทุกตัวว่า histogram นั้นๆ อยู่ในกลุ่มของ codevector ใด ทำจนครบทุก histogram จากนั้นทำการเก็บค่าสถิติของจำนวนสมาชิกใน codevector ของช็อตวิดีโอนั้น แล้วทำการเก็บเก็บฐานข้อมูล

ในการเก็บดัชนีมีการเก็บอยู่ 2 รูปแบบ คือ การเก็บดัชนีแบบ One Codevector (ŋ=1) และการเก็บดัชนีแบบ Two Codevector (ŋ=2) ค่าของดัชนี คือ จำนวนสมาชิกใน codevector นั้น โดยเริ่มเก็บตั้งแต่ codevector ตัวที่ 1 ไปจนครบ แล้วทำการนับค่าของจำนวนสมาชิก เก็บลงฐานข้อมูล ดังนั้น ค่าของตำแหน่งที่ 1 ในฐานข้อมูล คือ จำนวนสมาชิกของ codevector ตัวที่ 1 นั่นเอง เช่นVideo1Shot1 มีจำนวน codevector อยู่ 10 ตัว ดังนี้

Video1Shot1 = (1, 5, 9, 20, 5, 0, 0, 0, 0, 0)

C_1 C_2 ... C_{10}

หมายความว่า Index ที่ได้นี้ เปรียบเทียบกับ codevector 10 ตัว

และค่า codevector ตัวที่ 1 มีจำนวนสมาชิก 1 ตัว

ค่า codevector ตัวที่ 2 มีจำนวนสมาชิก 5 ตัว

ค่า codevector ตัวที่ 3 มีจำนวนสมาชิก 9 ตัว

ค่า codevector ตัวที่ 4 มีจำนวนสมาชิก 20 ตัว

ค่า codevector ตัวที่ 5 มีจำนวนสมาชิก 5 ตัว

ค่า codevector ตัวที่ 6 มีจำนวนสมาชิก 0 ตัว

ค่า codevector ตัวที่ 7 มีจำนวนสมาชิก 0 ตัว

ค่า codevector ตัวที่ 8 มีจำนวนสมาชิก 0 ตัว

ค่า codevector ตัวที่ 9 มีจำนวนสมาชิก 0 ตัว

ค่า codevector ตัวที่ 10 มีจำนวนสมาชิก 0 ตัว

จะสรุปได้ว่าช็อตวิดีโอนี้มีจำนวน histogram 15 histogram หรือ 15 เฟรมนั่นเอง

โดยการเก็บค่าดัชนีแบบ Two Codevector (ŋ=2) นั้นจะเก็บจากการเรียงลำดับความใกล้เคียงของ histogram กับ codevector ในลำดับที่ 1 และ 2 มาเป็นสมาชิก ซึ่งจะทำให้ได้ความละเอียดมากขึ้น

ในส่วนของฐานข้อมูลจะเก็บทั้งหมด 3 ฟิลด์ โดยที่

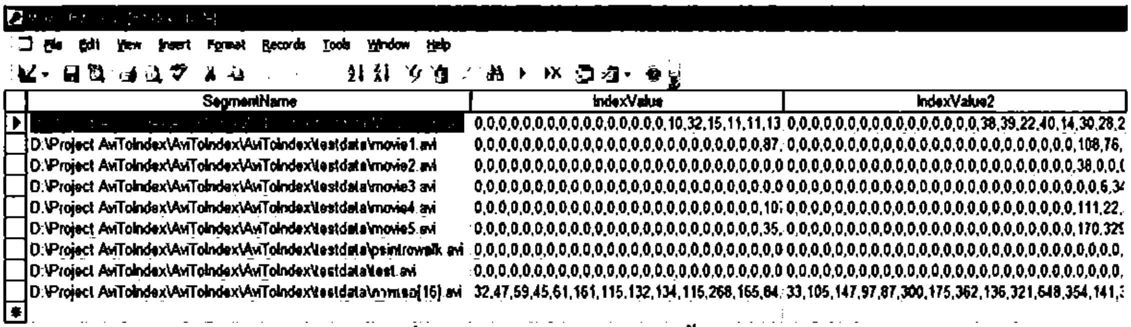
- ฟิลด์ที่ 1 เก็บชื่อ และ path ของ video
- ฟิลด์ที่ 2 เก็บค่าดัชนีแบบ One Codevector (ŋ=1)
- ฟิลด์ที่ 3 เก็บค่าดัชนีแบบ Two Codevector (ŋ=2)

3.3 การจัดการฐานข้อมูล

ในการจัดการกับฐานข้อมูลนั้น โปรแกรมจะทำการวิเคราะห์ค่าสี RGB ของทุกๆ เฟรม ในไฟล์วิดีโอทุกๆ ไฟล์ เพื่อนำมาทำการหาดัชนีสี ซึ่งดัชนีที่เก็บลงฐานข้อมูลมี 2 ประเภท คือ ดัชนีแบบ One Codevector (ŋ=1) และดัชนีแบบ Two Codevector (ŋ=2) โดยในการเก็บชื่อของช็อตวิดีโอจะอ้างอิงจาก ชื่อพาธที่เก็บช็อตวิดีโอนั้นและตามด้วยชื่อของช็อตวิดีโอ เช่น D:\Project AviToIndex\AviToIndex\AviToIndex\testdata\MOV07993.avi เป็นต้น โดยโปรแกรมในส่วนของระบบการค้นหา จะดึงข้อมูลในส่วนนี้ไปใช้ เนื่องจากในส่วนของระบบการค้นหาก็มีตัวเลือกในการค้นหาให้แก่ผู้ใช้ 2 ประเภทเช่นกัน โดยแบ่งประเภทการเลือกตามการเก็บดัชนีนั่นเอง และยังมีการเก็บฐานข้อมูลอีกส่วนหนึ่ง คือ การเก็บค่าของ codevector แต่ละตัวจากขั้นตอน Quantization

ตารางที่ 3.1 แสดงตัวอย่างการเก็บดัชนี

ชื่อช็อตวิดีโอ	ดัชนีแบบ One Codevector (ŋ=1)	ดัชนีแบบ Two Codevector (ŋ=2)
Video1shot1.avi	$a_1, a_2, a_3, \ldots\ldots\ldots\ldots, a_n$	$A_1, A_2, A_3, \ldots\ldots\ldots\ldots, A_n$
Video1shot2.avi	$b_1, b_2, b_3, \ldots\ldots\ldots\ldots, b_n$	$B_1, B_2, B_3, \ldots\ldots\ldots\ldots, B_n$
Video1shot3.avi	$c_1, c_2, c_3, \ldots\ldots\ldots\ldots, c_n$	$C_1, C_2, C_3, \ldots\ldots\ldots\ldots, C_n$
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
VideoZshotZ.avi	$z_1, z_2, z_3, \ldots\ldots\ldots\ldots, z_n$	$Z_1, Z_2, Z_3, \ldots\ldots\ldots\ldots, Z_n$

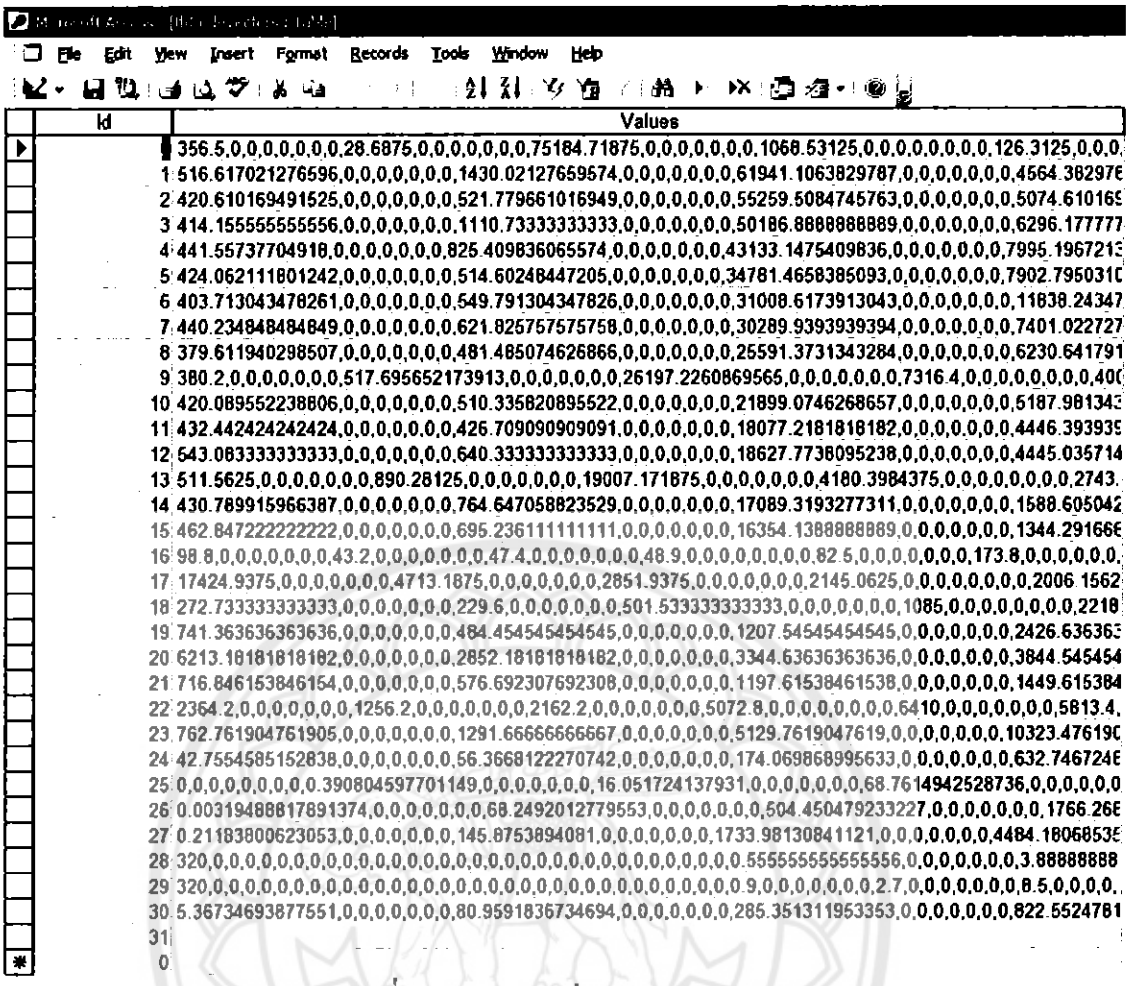


SegmentName	indexValue	IndexValue2
[illegible]	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10,32,15,11,11,13	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,38,39,22,40,14,30,28,2
D:\Project AviToIndex\AviToIndex\AviToIndex\testdata\movie1.avi	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,87,	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,108,76,
D:\Project AviToIndex\AviToIndex\AviToIndex\testdata\movie2.avi	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,38,0,0,(
D:\Project AviToIndex\AviToIndex\AviToIndex\testdata\movie3.avi	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,34
D:\Project AviToIndex\AviToIndex\AviToIndex\testdata\movie4.avi	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,111,22,
D:\Project AviToIndex\AviToIndex\AviToIndex\testdata\movie5.avi	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,35,	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,170,329
D:\Project AviToIndex\AviToIndex\AviToIndex\testdata\psintrowalk.avi	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
D:\Project AviToIndex\AviToIndex\AviToIndex\testdata\test.avi	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
D:\Project AviToIndex\AviToIndex\AviToIndex\testdata\ภาพเอ(16).avi	32,47,59,45,61,161,115,132,134,115,268,165,64	33,105,147,97,87,300,175,362,136,321,648,354,141,3

รูปที่ 3.4 แสดงการเก็บดัชนีทั้ง 2 แบบ

ตารางที่ 3.2 แสดงตัวอย่างการเก็บค่าของ codevector

ลำดับของ codevector	ค่าของ codevector
0	$a_1, a_2, a_3, \ldots\ldots, a_n$
1	$b_1, b_2, b_3, \ldots\ldots, b_n$
2	$c_1, c_2, c_3, \ldots\ldots, c_n$
.	.
.	.
.	.
.	.
.	.
.	.
.	.
.	.
.	.
.	.
n	$z_1, z_2, z_3, \ldots\ldots, z_n$



Id	Values
0	356.5,0,0,0,0,0,0,0,28.6875,0,0,0,0,0,0,0,75184.71875,0,0,0,0,0,0,0,1068.53125,0,0,0,0,0,0,0,126.3125,0,0,0,
1	516.617021276596,0,0,0,0,0,0,0,1430.02127659574,0,0,0,0,0,0,0,61941.1063829787,0,0,0,0,0,0,0,4564.382976
2	420.610169491525,0,0,0,0,0,0,0,521.779661016949,0,0,0,0,0,0,0,55259.5084745763,0,0,0,0,0,0,0,5074.610169
3	414.155555555556,0,0,0,0,0,0,0,1110.73333333333,0,0,0,0,0,0,0,50186.8888888889,0,0,0,0,0,0,0,6296.177777
4	441.55737704918,0,0,0,0,0,0,0,825.409836065574,0,0,0,0,0,0,0,43133.1475409836,0,0,0,0,0,0,0,7995.1967213
5	424.062111801242,0,0,0,0,0,0,0,514.60248447205,0,0,0,0,0,0,0,34781.4658385093,0,0,0,0,0,0,0,7902.795031(
6	403.713043478261,0,0,0,0,0,0,0,549.791304347826,0,0,0,0,0,0,0,31008.6173913043,0,0,0,0,0,0,0,11838.24347
7	440.234848484849,0,0,0,0,0,0,0,621.825757575758,0,0,0,0,0,0,0,30289.9393939394,0,0,0,0,0,0,0,7401.022727
8	379.611940298507,0,0,0,0,0,0,0,481.485074626866,0,0,0,0,0,0,0,25591.3731343284,0,0,0,0,0,0,0,6230.641791
9	380.2,0,0,0,0,0,0,0,517.695652173913,0,0,0,0,0,0,0,26197.2260869565,0,0,0,0,0,0,0,7316.4,0,0,0,0,0,0,0,400
10	420.089552238806,0,0,0,0,0,0,0,510.335820895522,0,0,0,0,0,0,0,21899.0746268657,0,0,0,0,0,0,0,5187.981343
11	432.442424242424,0,0,0,0,0,0,0,426.709090909091,0,0,0,0,0,0,0,18077.2181818182,0,0,0,0,0,0,0,4446.393939
12	643.083333333333,0,0,0,0,0,0,0,640.333333333333,0,0,0,0,0,0,0,18627.7738095238,0,0,0,0,0,0,0,4445.035714
13	511.5625,0,0,0,0,0,0,0,890.28125,0,0,0,0,0,0,0,19007.171875,0,0,0,0,0,0,0,4180.3984375,0,0,0,0,0,0,0,2743.
14	430.789915966387,0,0,0,0,0,0,0,764.647058823529,0,0,0,0,0,0,0,17089.3193277311,0,0,0,0,0,0,0,1588.605042
15	462.847222222222,0,0,0,0,0,0,0,695.236111111111,0,0,0,0,0,0,0,16354.1388888889,0,0,0,0,0,0,0,1344.291666
16	98.8,0,0,0,0,0,0,0,43.2,0,0,0,0,0,0,0,47.4,0,0,0,0,0,0,0,48.9,0,0,0,0,0,0,0,82.5,0,0,0,0,0,0,0,173.8,0,0,0,0,0,0,
17	17424.9375,0,0,0,0,0,0,0,4713.1875,0,0,0,0,0,0,0,2851.9375,0,0,0,0,0,0,0,2145.0625,0,0,0,0,0,0,0,2006.1562
18	272.733333333333,0,0,0,0,0,0,0,229.6,0,0,0,0,0,0,0,501.533333333333,0,0,0,0,0,0,0,1085,0,0,0,0,0,0,0,2218
19	741.363636363636,0,0,0,0,0,0,0,484.454545454545,0,0,0,0,0,0,0,1207.54545454545,0,0,0,0,0,0,0,2426.636363
20	6213.18181818182,0,0,0,0,0,0,0,2852.18181818182,0,0,0,0,0,0,0,3344.63636363636,0,0,0,0,0,0,0,3844.545454
21	716.846153846154,0,0,0,0,0,0,0,576.692307692308,0,0,0,0,0,0,0,1197.61538461538,0,0,0,0,0,0,0,1449.615384
22	2364.2,0,0,0,0,0,0,0,1256.2,0,0,0,0,0,0,0,2162.2,0,0,0,0,0,0,0,5072.8,0,0,0,0,0,0,0,6410,0,0,0,0,0,0,0,5813.4,
23	762.761904761905,0,0,0,0,0,0,0,1291.66666666667,0,0,0,0,0,0,0,5129.7619047619,0,0,0,0,0,0,0,10323.476190
24	42.7554585152838,0,0,0,0,0,0,0,56.3668122270742,0,0,0,0,0,0,0,174.069868995633,0,0,0,0,0,0,0,632.746724E
25	0,0,0,0,0,0,0,0,0.390804597701149,0,0,0,0,0,0,0,16.051724137931,0,0,0,0,0,0,0,68.7614942528736,0,0,0,0,0,0
26	0.00319488817891374,0,0,0,0,0,0,0,68.2492012779553,0,0,0,0,0,0,0,504.450479233227,0,0,0,0,0,0,0,1766.26E
27	0.21183800623053,0,0,0,0,0,0,0,145.8753894081,0,0,0,0,0,0,0,1733.98130841121,0,0,0,0,0,0,0,4484.1806853E
28	320,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.555555555555556,0,0,0,0,0,0,3.88888888
29	320,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.9,0,0,0,0,0,0,2.7,0,0,0,0,0,0,8.5,0,0,0,0,
30	5.36734693877551,0,0,0,0,0,0,0,80.9591836734694,0,0,0,0,0,0,0,285.351311953353,0,0,0,0,0,0,0,822.5524781
31	
*	0

รูปที่ 3.5 แสดงการเก็บค่าของ codevector

3.4 ระบบการค้นหา

ในส่วนนี้ จะนำส่วนของค่าดัชนีและช็อตวิดีโอมาใช้ในการค้นหา เพื่อหาความคล้ายคลึงของช็อตวิดีโอต้นแบบกับช็อตวิดีโอที่อยู่ในฐานข้อมูล จะต้องทำการนำข้อมูลของช็อตวิดีโอต้นแบบมาเปรียบเทียบกับข้อมูลช็อตวิดีโอในฐานข้อมูล โดยการคำนวณหาค่าระยะห่าง (Distance) ระหว่างช็อตวิดีโอต้นแบบกับช็อตวิดีโอในฐานข้อมูล ซึ่งใช้อัลกอริทึมในการเปรียบเทียบ คือ Cosine Similarity เป็นการหาระยะห่าง (Cosine Similarity) ระหว่างจุด $P=(p_1,p_2,\ldots,p_n)$ และ $Q=(q_1,q_2,\ldots,q_n)$ โดยที่จุด P ในที่นี้ คือ เซตของ X_m และจุด Q คือ เซตของ C_n ซึ่งจะได้ว่า

$$d(P,Q)=\frac{\vec{p}_i\cdot\vec{q}_i}{|\vec{p}_i||\vec{q}_i|}=\frac{\sum_{i=1}^{n}p_iq_i}{\sqrt{\sum_{i=1}^{n}p_i^{\,2}}\sqrt{\sum_{i=1}^{n}q_i^{\,2}}} \qquad (3.10)$$

โดยที่ d(P,Q) = ระยะห่าง (Cosine Similarity)

p_i = เป็นค่าดัชนี (Index) ของรูปภาพต้นแบบ

q_i = เป็นค่าดัชนี (Index) ของรูปภาพในฐานข้อมูล

ยกตัวอย่างเช่น สมมติให้ P = {2 , 9 , 4} และ Q = {6 , 8 ,1} จากสมการข้างต้น จะได้ว่า

$$d(P,Q) = \frac{(2\times6)+(9\times8)+(4\times1)}{\sqrt{(2^2+9^2+4^2}\times\sqrt{(6^2+8^2+1^2)}}$$

$$d(P,Q) = \frac{12+72+4}{\sqrt{101}\times\sqrt{101}}$$

$$d(P,Q) = \frac{88}{101}$$

$$d(P,Q) = 0.871287$$

ดังนั้น การคำนวณหาค่าระยะห่าง (Distance) ระหว่างจุด P กับจุด Q มีค่าระยะห่าง (Distance) เท่ากับ 0.871287

โดยในที่นี้ P หมายถึง X_m และ Q หมายถึง C_n ทำการแทนค่า X_m และ C_n ไปจนครบทุกตัว จะได้ค่าระยะห่าง (Distance) ของทุกตัวออกมา นำค่าระยะห่างมาทำการเรียงลำดับจากมากไปน้อย โดยใช้วิธีการเรียงลำดับแบบเลือก (Selection sort) ซึ่งในส่วนของการแสดงผลจากการค้นหา ให้นำค่าที่ได้จากการเรียงลำดับใน 8 อันดับแรก มาทำการแสดงผล

โดยแบ่งการค้นหาออกเป็น 2 ประเภท คือ

1. การค้นหาโดยใช้ดัชนีแบบ One Codevector (ŋ=1) เป็นการค้นหาโดยใช้ดัชนีในลำดับที่ 1 มาทำเปรียบเทียบกับทุกชื่อต

2. การค้นหาโดยใช้ดัชนีแบบ Two Codevector (ŋ=2) เป็นการค้นหาโดยใช้ดัชนีในลำดับที่ 1 และลำดับที่ 2 มาทำเปรียบเทียบกับทุกชื่อต

บทที่ 4
การทดสอบและวิคราะห์การทำงาน

ในบทนี้จะเป็นผลของการทำการทดลองการเก็บดัชนีและค้นหาช็อตวิดีโอจากฐานข้อมูล ผลที่ได้เป็นมาจากการสังเกตด้วยตา และใช้หลักการเปรียบเทียบผลของค่า Precision ซึ่งเป็นค่าที่ได้จากการคำนวณจากสมการดังต่อไปนี้

%ความถูกต้อง = (จำนวนช็อตวิดีโอที่ถูกเลือก / จำนวนช็อตวิดีโอทั้งหมด)

<u>ยกตัวอย่าง</u> เช่น %ความถูกต้อง = $\left(\frac{6}{8}\right)\times 100 = 75\%$

ซึ่งถ้านำค่าความถูกต้อง (Precision) ของ Query ทั้งหมด มารวมกันแล้วหาค่าเฉลี่ย ก็จะได้เป็น Precision ของ Method นั้นๆ ซึ่งเมื่อนำไปเปรียบเทียบกับ Method อื่นๆ ก็จะเห็นความแตกต่างของการค้นหาได้จากการสร้างกราฟ

4.1 สิ่งที่ต้องเตรียมก่อนทำการทดลอง

1) โปรแกรมเก็บดัชนีและค้นหาช็อตวิดีโอ ที่พัฒนามาจากโปรแกรม Visual Studio 2005

2) โปรแกรม Video Editor 7.0 เพื่อใช้ในการแยกไฟล์วิดีโอ ออกเป็นช็อต

3) ไฟล์ช็อตวิดีโอที่ได้จากการแยกช็อตด้วยโปรแกรม Video Editor 7.0 ซึ่งเป็นไฟล์ .avi ทั้งหมด

4) มีวิดีโอที่ทำการแยกช็อตวิดีโออยู่ทั้งหมด 7 เรื่อง (Beauty Shop 818 ช็อต, Elisabeth Town 785 ช็อต, บางรักซอยเก้า ตอน ลูกผู้ชายวัดกันที่ใจ 220 ช็อต, Top Gear 55 ช็อต, มิวสิควิดีโอ ภาพเธอ 37 ช็อต, รายการโทรทัศน์ จับชีพจรโลก 103 ช็อต, สารคดี ตอน Polar Bears and Global Warming 70 ช็อต) รวมมีวิดีโอทั้งหมด 2088 ช็อต

4.2 ผลการทดลอง

การทดลองการค้นหาวิดีโอมีอยู่ทั้งหมด 2 แบบด้วยกัน คือ

4.2.1 การค้นหาโดยใช้ดัชนี แบ่งเป็น

1. One Codevector (η=1)
2. Two Codevector (η=2)

4.2.2 การค้นหาโดยการแยกจำนวน Codevector แบ่งเป็น

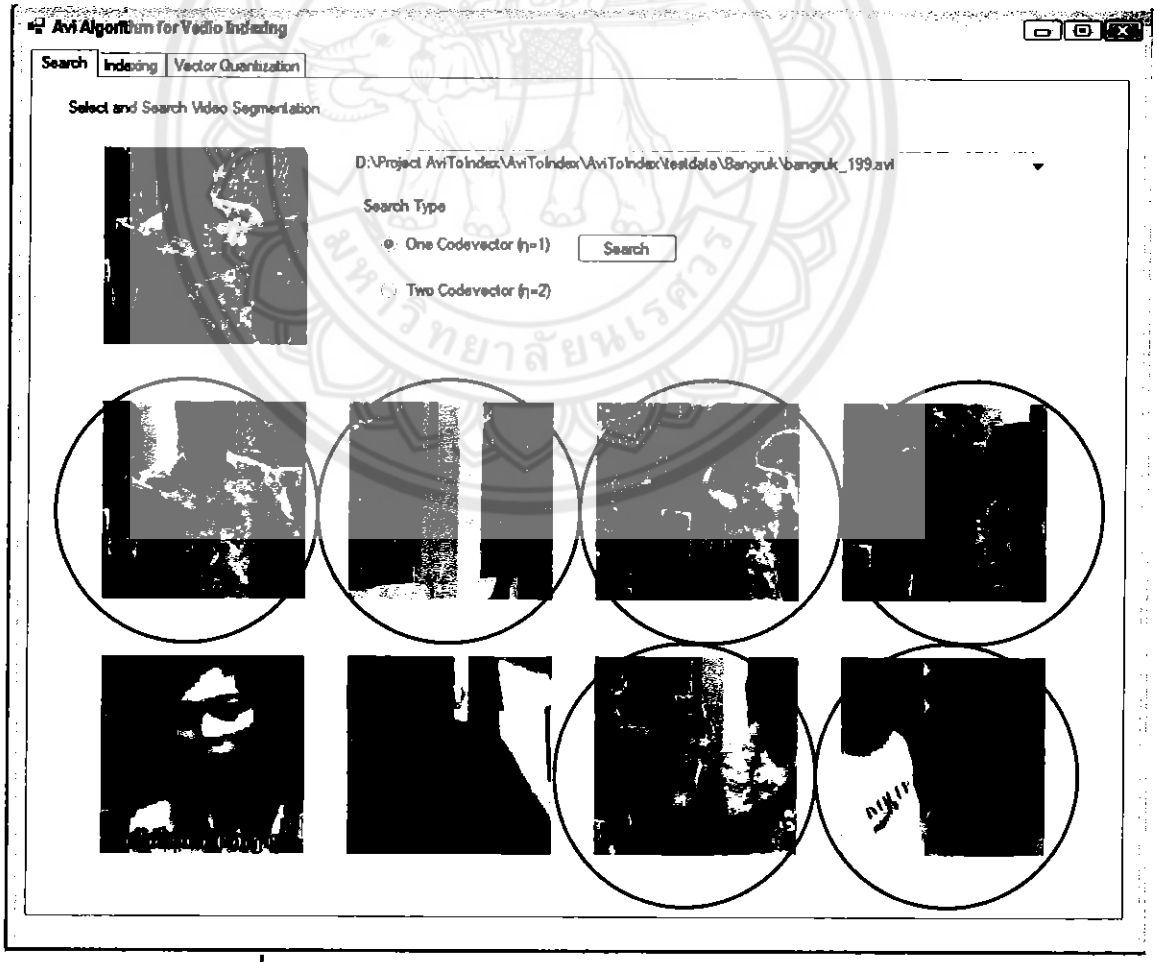
1. จำนวน Codevector 4 ตัว
2. จำนวน Codevector 16 ตัว
3. จำนวน Codevector 64 ตัว

4.2.1 การค้นหาไฟล์วิดีโอแบบใช้ดัชนี

1. การค้นหาโดยใช้ดัชนีแบบ One Codevector (ɳ=1) เปรียบเทียบกับจำนวน Codevector 64 ตัว

จากรูปการทดลองที่ 4.1 จะเห็นได้ว่ามีชื่อตวิดีโอที่มีลักษณะที่อยู่ในกลุ่มเดียวกันกับไฟล์ต้นแบบทั้งหมด 6 ไฟล์ เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

$$\% \text{ Precision} = \left(\frac{6}{8}\right) \times 100 = 75\%$$



รูปที่ 4.1 แสดงการค้นหาโดยใช้ดัชนีแบบ One Codevector (ɳ=1)

ตารางที่ 4.1 บันทึกการทดลองการค้นหาโดยใช้ดัชนีแบบ One Codevector (ŋ=1)

ลำดับ	ชื่อไฟล์วิดีโอ	จำนวนช็อตที่ถูกเลือก	ตำแหน่ง shot ที่ถูกเลือก								Precision (%)
			1	2	3	4	5	6	7	8	
1	..\\ภาพเธอ_01.avi	6	*	*	*	*	*	*			75
2	..\\ภาพเธอ_08.avi	6	*	*		*		*	*	*	75
3	..\\ภาพเธอ_16.avi	6	*	*	*		*	*	*		75
4	..\\ภาพเธอ_26.avi	5	*	*	*		*	*			62.5
5	..\\ภาพเธอ_29.avi	6	*		*		*	*	*	*	75
6	..\\ภาพเธอ_36.avi	5	*			*	*	*	*		62.5
7	..\\top gear_4.avi	5	*	*	*	*				*	62.5
8	..\\top gear_6.avi	5	*	*			*		*	*	62.5
9	..\\top gear_26.avi	6		*	*	*	*		*	*	75
10	..\\top gear_49.avi	5	*	*	*	*	*				62.5
11	..\\top gear_32.avi	4	*	*	*			*			50
12	..\\top gear_42.avi	6	*	*	*		*		*	*	75
13	..\\polar bear_2.avi	4			*	*			*	*	50
14	..\\polar bear_6.avi	8	*	*	*	*	*	*	*	*	100
15	..\\polar bear_12.avi	6	*		*		*	*	*	*	75
16	..\\polar bear_25.avi	5	*	*	*	*				*	62.5
17	..\\polar bear_33.avi	8	*	*	*	*	*	*	*	*	100
18	..\\polar bear_37.avi	4	*	*	*	*					50
19	..\\Yoon_1.avi	2	*	*							25
20	..\\Yoon_8.avi	8	*	*	*	*	*	*	*	*	100
21	..\\Yoon_10.avi	7	*	*	*		*	*	*	*	87.5
22	..\\Yoon_20.avi	4	*	*		*		*			50
23	..\\Yoon_24.avi	7	*	*	*	*	*	*		*	87.5
24	..\\Yoon_30.avi	4	*	*	*				*		50
25	..\\bangruk_2.avi	7	*	*	*		*	*	*	*	87.5
26	..\\bangruk_5.avi	5	*	*	*	*		*			62.5
27	..\\bangruk_12.avi	7	*	*	*	*	*	*	*		87.5
28	..\\bangruk_24.avi	8	*	*	*	*	*	*	*	*	100

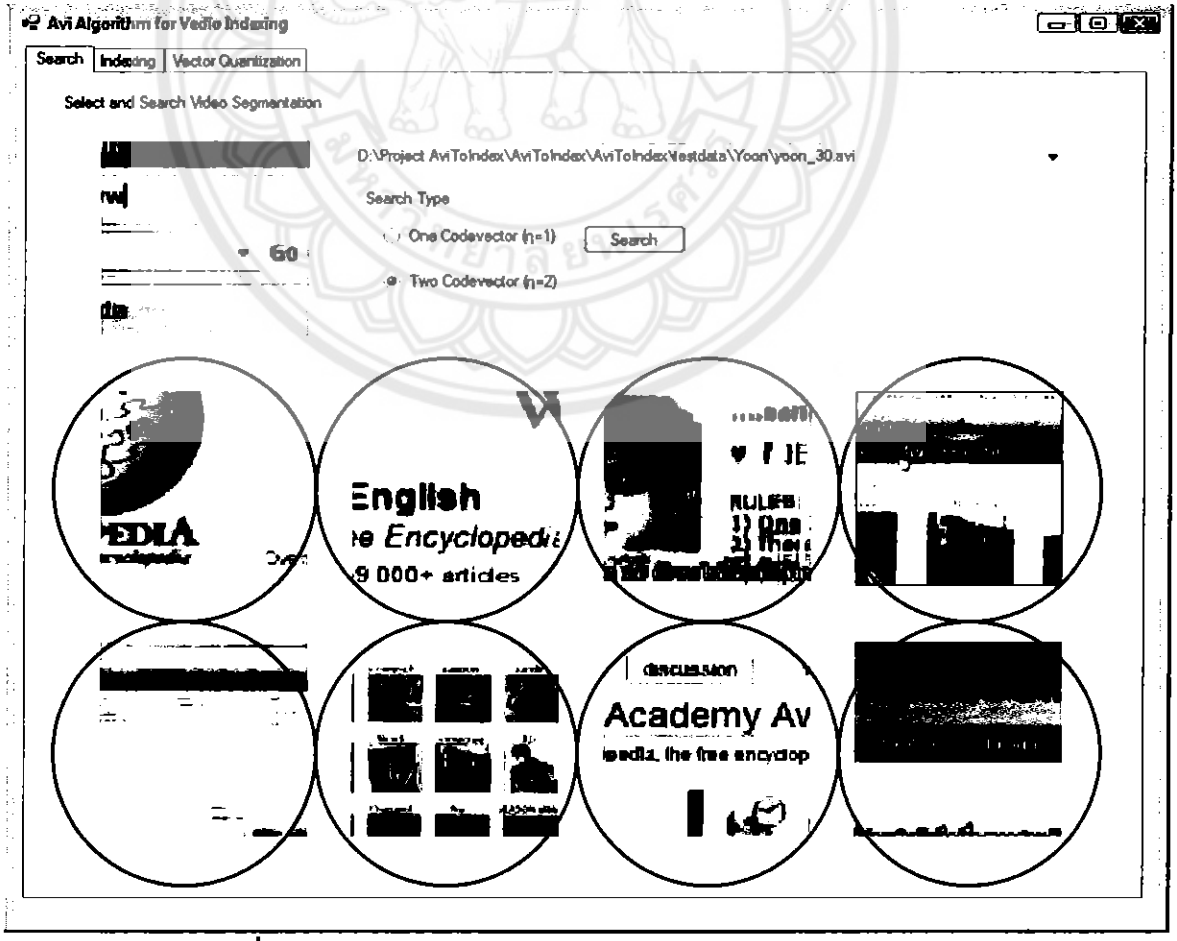
ตารางที่ 4.1(ต่อ) บันทึกการทดลองการค้นหาโดยใช้ดัชนีแบบ One Codevector (ŋ=1)

ลำดับ	ชื่อไฟล์วิดีโอ	จำนวนช็อตที่ถูกเลือก	ตำแหน่ง shot ที่ถูกเลือก								Precision (%)
29	..\\bangruk_60.avi	5	*	*		*		*	*		62.5
30	..\\bangruk_199.avi	6	*	*	*	*			*	*	75
Sum		181	28	26	24	19	18	19	19	17	2262.5
Avg		6.03									75.417

2. การค้นหาโดยใช้ดัชนีแบบ Two Codevector (ŋ=2) เปรียบเทียบกับจำนวน Codevector 64 ตัว

จากรูปการทดลองที่ 4.2 จะเห็นได้ว่ามีช็อตวิดีโอที่มีลักษณะที่อยู่ในกลุ่มเดียวกันกับไฟล์ต้นแบบทั้งหมด 8 ไฟล์ เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

$$\% \text{ Precision} = \left(\frac{8}{8}\right) \times 100 = 100\%$$



รูปที่ 4.2 แสดงการค้นหาโดยใช้ดัชนีแบบ Two Codevector (ŋ=2)

ตารางที่ 4.2 บันทึกการทดลองการค้นหาโดยใช้ดัชนีแบบ Two Codevector (ŋ=2)

ลำดับ	ชื่อไฟล์วิดีโอ	จำนวนช็อตที่ถูกเลือก	ตำแหน่ง shot ที่ถูกเลือก								Precision (%)
			1	2	3	4	5	6	7	8	
1	..\\ภาพเธอ_01.avi	5	*	*	*	*				*	62.5
2	..\\ภาพเธอ_08.avi	7	*	*	*	*	*	*	*		87.5
3	..\\ภาพเธอ_16.avi	7	*	*	*	*	*		*	*	87.5
4	..\\ภาพเธอ_26.avi	7		*	*	*	*	*	*	*	87.5
5	..\\ภาพเธอ_29.avi	2			*					*	25
6	..\\ภาพเธอ_36.avi	6	*	*		*	*	*		*	75
7	..\\top gear_4.avi	3	*		*	*					37.5
8	..\\top gear_6.avi	5	*	*	*				*	*	62.5
9	..\\top gear_26.avi	4	*		*		*			*	50
10	..\\top gear_49.avi	4		*			*		*	*	50
11	..\\top gear_32.avi	3			*	*		*			37.5
12	..\\top gear_42.avi	6	*	*	*	*	*	*			75
13	..\\polar bear_2.avi	2		*		*					25
14	..\\polar bear_6.avi	7	*	*	*	*	*	*	*		87.5
15	..\\polar bear_12.avi	7		*	*	*	*	*	*	*	87.5
16	..\\polar bear_25.avi	7	*	*	*		*	*	*	*	87.5
17	..\\polar bear_33.avi	7	*	*	*	*	*	*	*		87.5
18	..\\polar bear_37.avi	5	*	*	*	*	*				62.5
19	..\\Yoon_1.avi	2	*	*							25
20	..\\Yoon_63.avi	4	*	*	*	*					50
21	..\\Yoon_10.avi	7	*	*	*		*	*	*	*	87.5
22	..\\Yoon_20.avi	3	*	*				*			37.5
23	..\\Yoon_24.avi	4	*	*	*	*					50
24	..\\Yoon_30.avi	8	*	*	*	*	*	*	*	*	100
25	..\\bangruk_2.avi	5	*	*	*		*			*	62.5
26	..\\bangruk_5.avi	5	*		*		*	*		*	62.5
27	..\\bangruk_12.avi	6	*	*		*	*	*		*	75
28	..\\bangruk_24.avi	7	*	*	*	*	*	*		*	87.5

ตารางที่ 4.2(ต่อ) บันทึกการทดลองการค้นหาโดยใช้ดัชนีแบบ Two Codevector (ŋ=2)

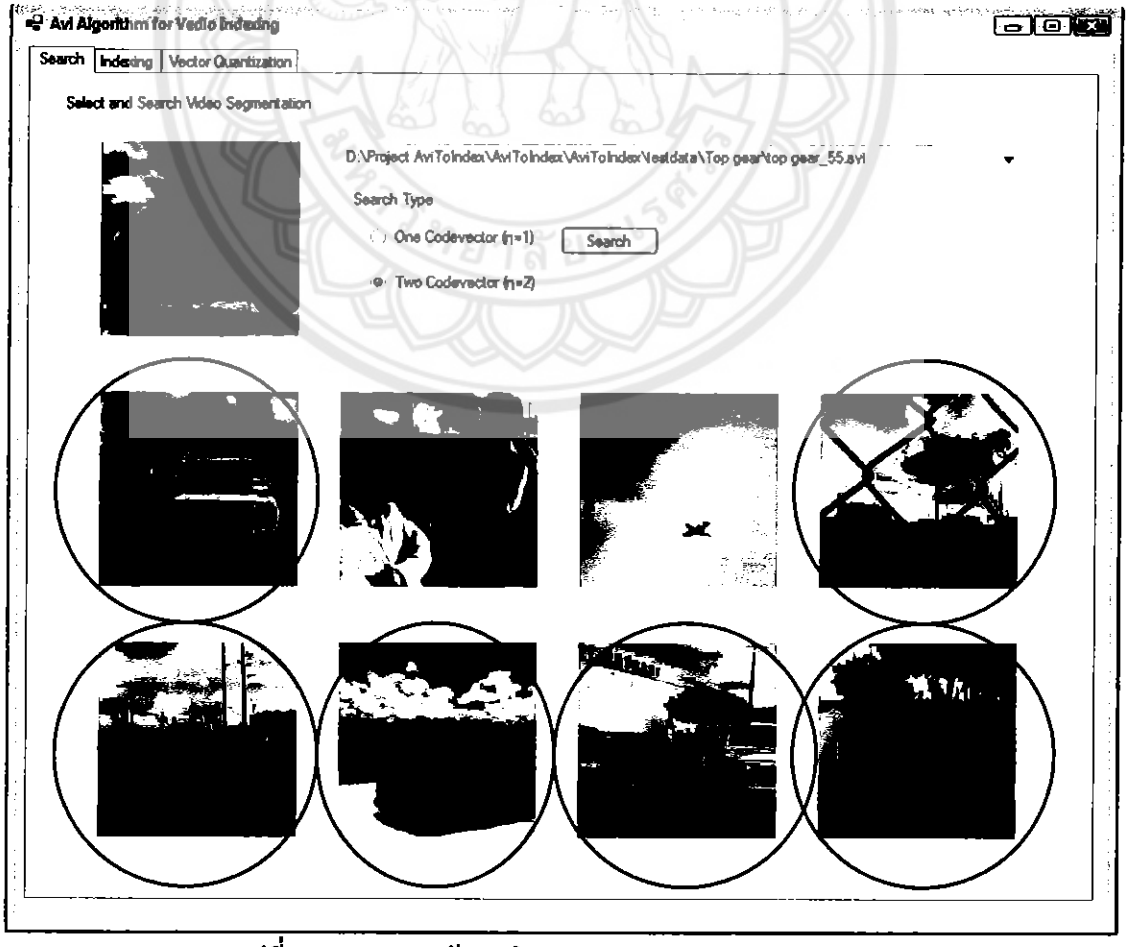
ลำดับ	ชื่อไฟล์วิดีโอ	จำนวนช็อตที่ถูกเลือก	ตำแหน่ง shot ที่ถูกเลือก								Precision (%)
29	..\\bangruk_60.avi	7	*	*	*	*	*		*	*	87.5
30	..\\bangruk_199.avi	7	*	*	*	*		*	*	*	87.5
Sum		159	24	25	24	20	19	16	13	18	1987.5
Avg		5.3									66.25

4.2.2 การค้นหาโดยการแยกจำนวน Codevector

1. การค้นหาโดยการแยกจำนวน Codevector 4 ตัว

จากรูปการทดลองที่ 4.3 จะเห็นได้ว่ามีช็อตวิดีโอที่มีลักษณะที่อยู่ในกลุ่มเดียวกันกับไฟล์ต้นแบบทั้งหมด 6 ไฟล์ เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

$$\% \text{ Precision} = \left(\frac{6}{8}\right) \times 100 = 75\%$$



รูปที่ 4.3 แสดงการค้นหาโดยการแยกจำนวน Codevector 4 ตัว

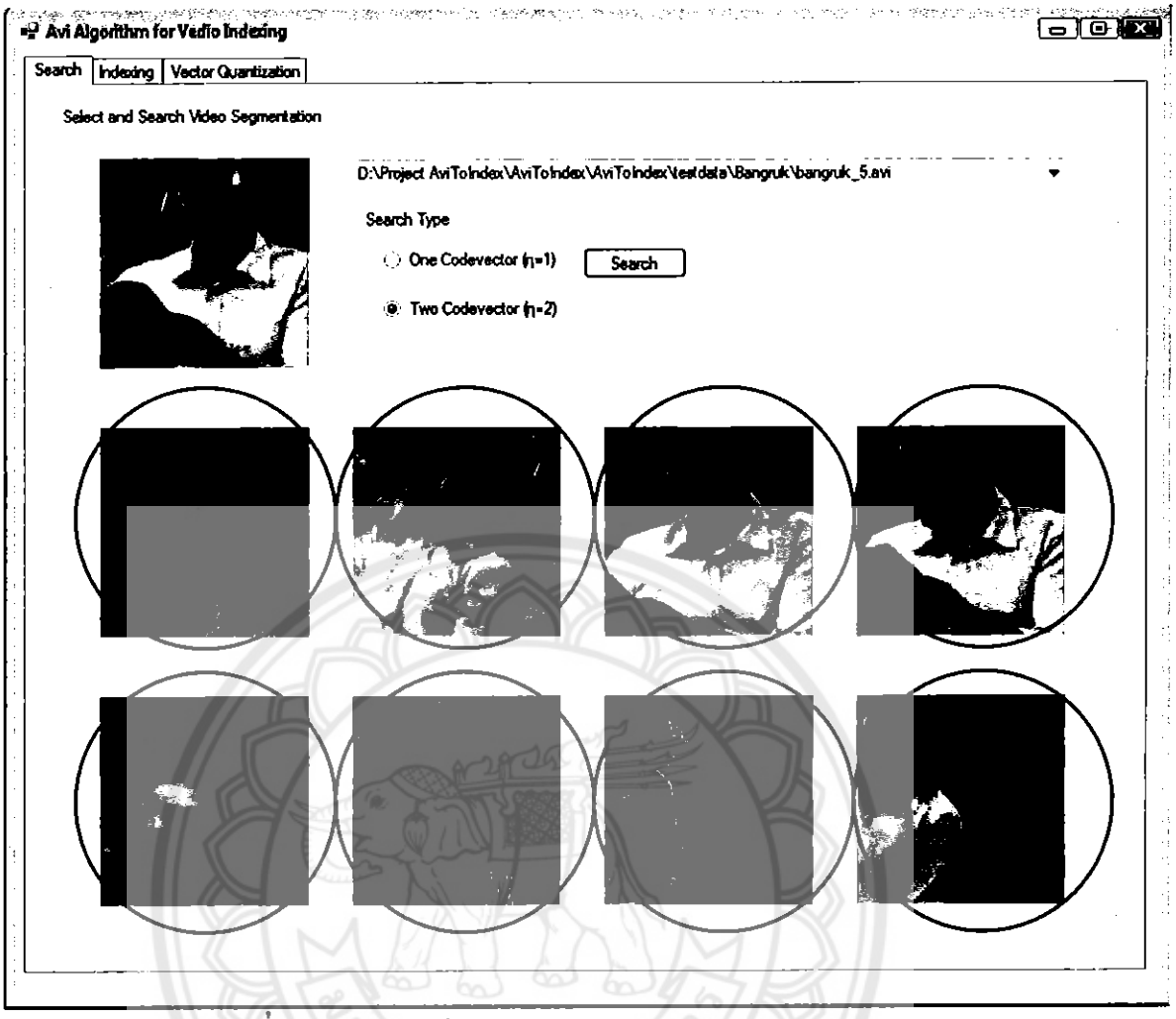
ตารางที่ 4.3 บันทึกการทดลองการค้นหาโดยการแยกจำนวน Codevector 4 ตัว

ลำดับ	ชื่อไฟล์วิดีโอ	จำนวนช็อตที่ถูกเลือก $\eta = 1$	Precision (%)	จำนวนช็อตที่ถูกเลือก $\eta = 2$	Precision (%)
1	..\\ภาพเธอ_01.avi	4	**50**	4	**50**
2	..\\ภาพเธอ_05.avi	7	**87.5**	7	**87.5**
3	..\\top gear_6.avi	4	**50**	7	**87.5**
4	..\\ top gear_55.avi	4	**50**	6	**75**
5	..\\Polar Bear_05.avi	7	**87.5**	8	**100**
6	..\\Yoon_8.avi	5	**62.5**	8	**100**
7	..\\ Yoon_31.avi	4	**50**	1	**12.5**
8	..\\bangruk_5.avi	5	**62.5**	6	**75**
9	..\\bangruk_99.avi	3	**37.5**	3	**37.5**
10	..\\bangruk_220.avi	8	**100**	5	**62.5**
Sum		51	**637.5**	55	**687.5**
Avg		5.1	**63.75**	5.5	**68.75**

2. การค้นหาโดยการแยกจำนวน Codevector 16 ตัว

จากรูปการทดลองที่ 4.4 จะเห็นได้ว่ามีช็อตวิดีโอที่มีลักษณะที่อยู่ในกลุ่มเดียวกันกับไฟล์ต้นแบบทั้งหมด 8 ไฟล์ เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

$$\% \text{ Precision} = \left(\frac{8}{8}\right) \times 100 = 100\%$$



รูปที่ 4.4 แสดงการค้นหาโดยการแยกจำนวน Codevector 16 ตัว

ตารางที่ 4.4 บันทึกการทดลองการค้นหาโดยการแยกจำนวน Codevector 16 ตัว

ลำดับ	ชื่อไฟล์วิดีโอ	จำนวนช็อตที่ถูกเลือก ŋ = 1	Precision (%)	จำนวนช็อตที่ถูกเลือก ŋ = 2	Precision (%)
1	..\\ภาพเธอ_01.avi	4	**50**	6	**75**
2	..\\ภาพเธอ_05.avi	5	**62.5**	5	**62.5**
3	..\\top gear_6.avi	8	**100**	6	**75**
4	..\\top gear_55.avi	7	**87.5**	8	**100**
5	..\\Polar Bear_05.avi	8	**100**	8	**100**
6	..\\Yoon_8.avi	7	**87.5**	8	**100**
7	..\\Yoon_31.avi	7	**87.5**	8	**100**
8	..\\bangruk_5.avi	7	**87.5**	8	**100**
9	..\\bangruk_99.avi	4	**50**	4	**50**

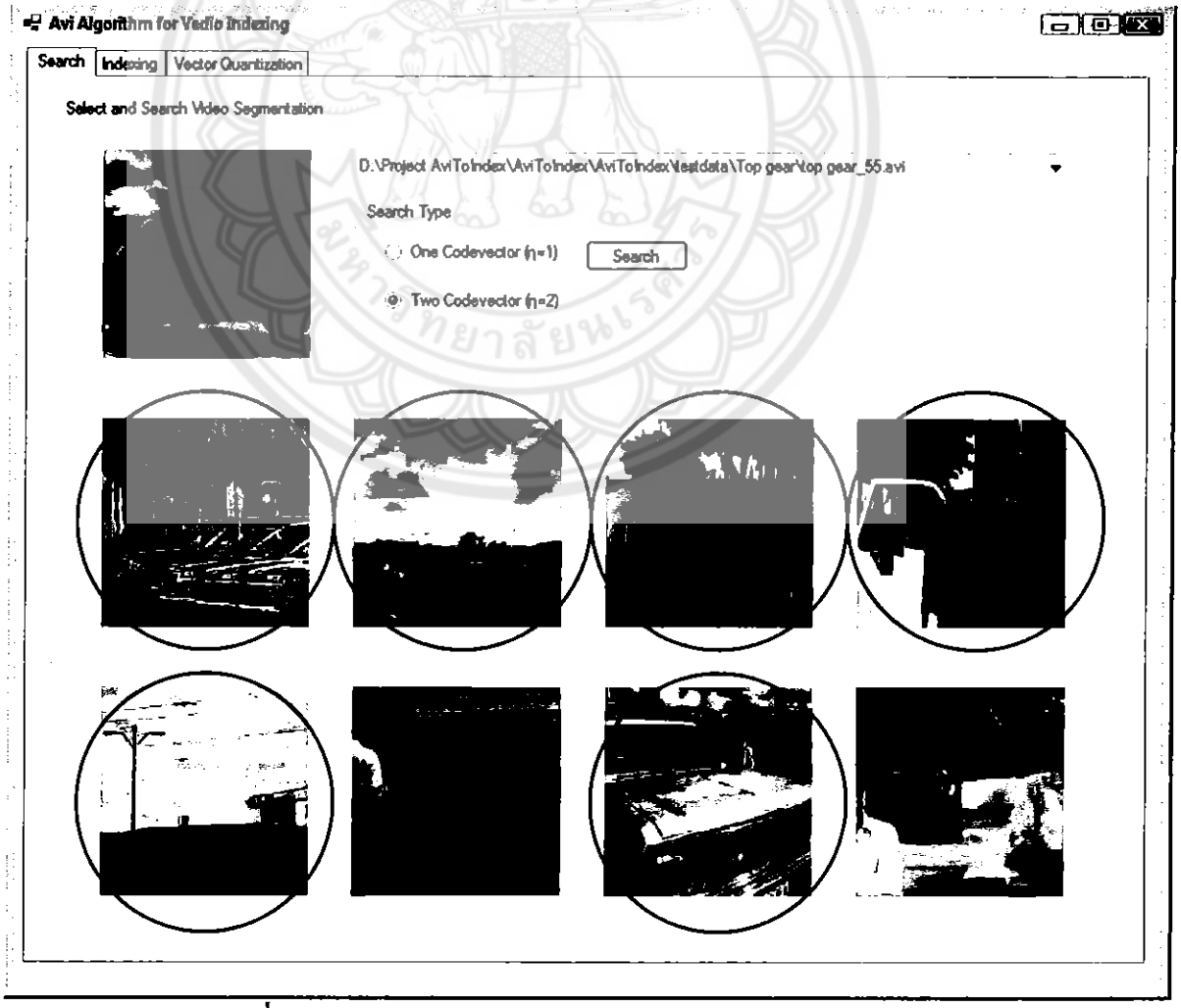
ตารางที่ 4.4(ต่อ) บันทึกการทดลองการค้นหาโดยการแยกจำนวน Codevector 16 ตัว

ลำดับ	ชื่อไฟล์วิดีโอ	จำนวนช็อตที่ถูกเลือก $\eta = 1$	Precision (%)	จำนวนช็อตที่ถูกเลือก $\eta = 2$	Precision (%)
10	..\\bangruk_220.avi	8	**100**	8	**100**
Sum		65	**812.5**	69	**817.5**
Avg		6.5	**81.25**	6.9	**81.75**

3. การค้นหาโดยการแยกจำนวน Codevector 64 ตัว

จากรูปการทดลองที่ 4.5 จะเห็นได้ว่ามีช็อตวิดีโอที่มีลักษณะที่อยู่ในกลุ่มเดียวกันกับไฟล์ต้นแบบทั้งหมด 6 ไฟล์ เมื่อนำมาคำนวณหาค่า Precision จะได้ว่า

$$\% \text{ Precision} = \left(\frac{6}{8}\right) \times 100 = 75\%$$



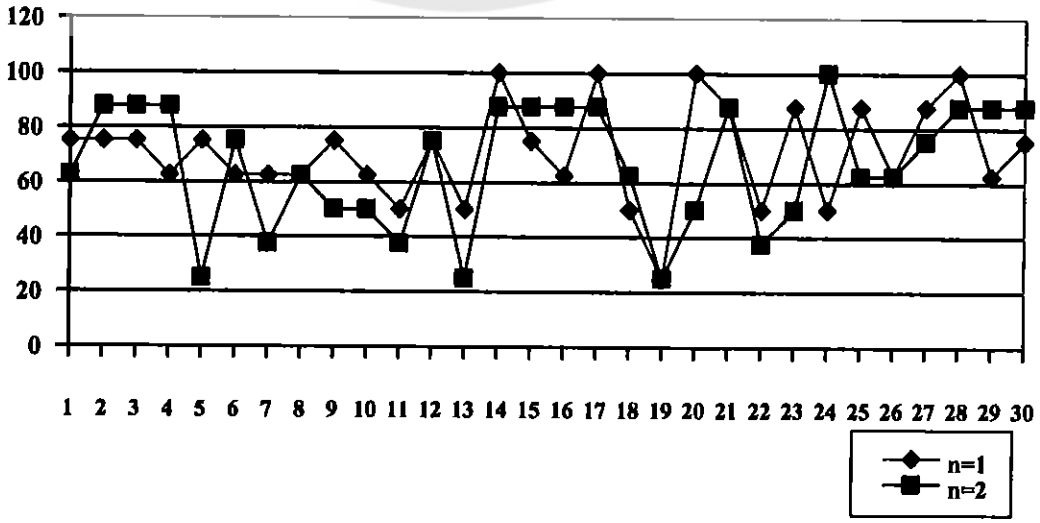
รูปที่ 4.5 แสดงการค้นหาโดยการแยกจำนวน Codevector 64 ตัว

ตารางที่ 4.5 บันทึกการทดลองการค้นหาโดยการแยกจำนวน Codevector 64 ตัว

ลำดับ	ชื่อไฟล์วิดีโอ	จำนวนช็อตที่ถูกเลือก η = 1	Precision (%)	จำนวนช็อตที่ถูกเลือก η = 2	Precision (%)
1	..\\ภาพเธอ_01.avi	4	**50**	6	**75**
2	..\\ภาพเธอ_05.avi	5	**62.5**	6	**75**
3	..\\top gear_6.avi	4	**50**	7	**87.5**
4	..\\top gear_55.avi	4	**50**	6	**75**
5	..\\Polar Bear_05.avi	8	**100**	8	**100**
6	..\\Yoon_8.avi	8	**100**	8	**100**
7	..\\Yoon_31.avi	4	**50**	7	**87.5**
8	..\\bangruk_5.avi	5	**62.5**	6	**75**
9	..\\bangruk_99.avi	6	**75**	6	**75**
10	..\\bangruk_20.avi	7	**87.5**	5	**62.5**
Sum		55	**687.5**	65	**812.5**
Avg		5.5	**68.75**	6.5	**81.25**

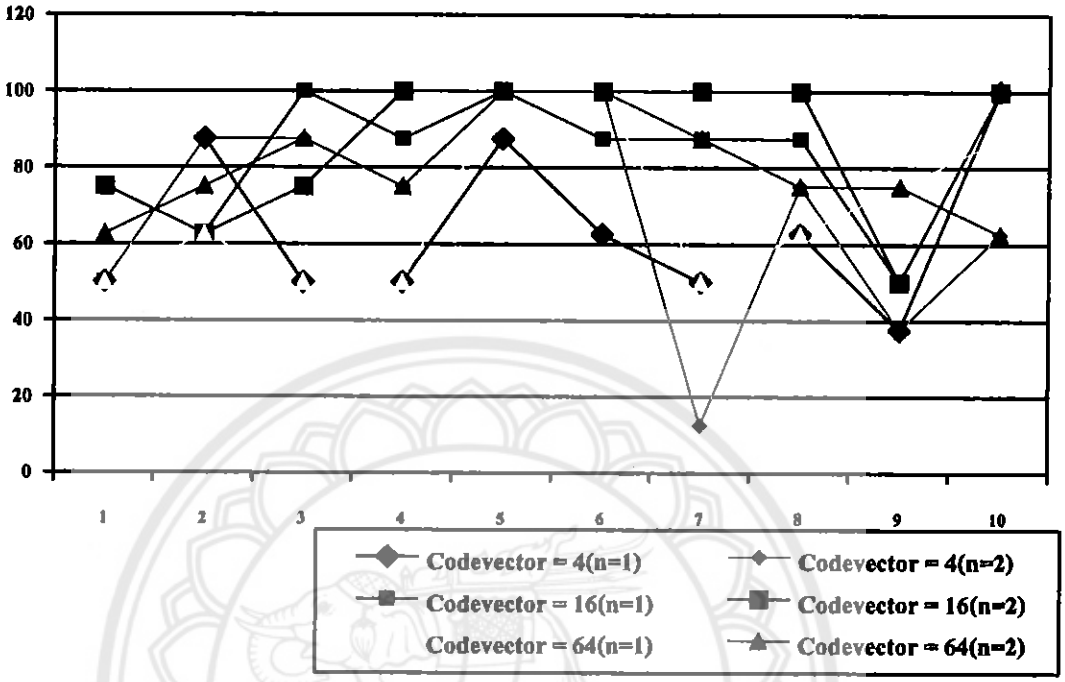
4.3 เปรียบทียบผลการทดลอง

นำผลการทดลองที่หาค่าความถูกต้อง (Precision) ของการค้นหาไฟล์วิดีโอ มาพลอตกราฟเปรียบเทียบการค้นหาทั้ง 2 วิธี จะเห็นความแตกต่างของการทดลองโดยที่เส้นกราฟที่ได้จากการค้นหาไฟล์วิดีโอในแต่ละแบบ เป็นดังรูปที่ 4.6



รูปที่ 4.6 กราฟแสดงการเปรียบเทียบการค้นหาโดยใช้ดัชนี

จากรูปที่ 4.7 จะเห็นว่าจากการค้นหา กราฟที่มีการแบ่ง Codevector มากยิ่งจะได้ความละเอียดของค่าดัชนีมากขึ้นไปด้วย จึงทำให้การค้นหาได้ละเอียดมากขึ้น



รูปที่ 4.7 กราฟแสดงการเปรียบเทียบการค้นหาโดยใช้ Codevector

บทที่ 5
สรุปผลการดำเนินงาน

โครงงานนี้พัฒนาขึ้นเพื่อความสะดวกในการค้นหาช็อตจากไฟล์วิดีโอที่ผู้ใช้ต้องการรับชม เนื่องจากไฟล์วิดีโอในปัจจุบันมีจำนวนมากขึ้น ดังนั้น โปรแกรมนี้จะมาช่วยแก้ปัญหาในระบบการค้นหา ซึ่งมีความใกล้เคียงกับความต้องการของผู้ใช้และผลที่ได้มีประสิทธิภาพในระดับหนึ่ง ในส่วนของการหาดัชนีของรูปภาพในแต่ละช็อตนั้นใช้หลักการของ LBG Vector Quantization แล้วทำการเก็บดัชนีเข้าสู่ฐานข้อมูล และในส่วนของระบบการค้นหาใช้หลักการของ Cosine Similarity เพื่อเปรียบเทียบระยะห่าง จากนั้นทำการแสดงผลช็อตที่ใกล้เคียงกับช็อตวิดีโอต้นแบบมากที่สุด 8 อันดับแรก

โครงงานนี้พัฒนาด้วย Visual Studio 2005 และใช้ Microsoft Access 2003 เป็นฐานข้อมูล

5.1 ผลการดำเนินงาน

5.1.1 ตัวโปรแกรมสามารถทำการเก็บดัชนีจากไฟล์วิดีโอหลายๆ ไฟล์ได้

5.1.2 ตัวโปรแกรมสามารถทำการค้นหาช็อตวิดีโอจากฐานข้อมูลที่มีความใกล้เคียงกับช็อตวิดีโอต้นแบบได้

5.1.3 การค้นหาโดยใช้ดัชนีแบบ Two Codevector (ไ=2) มีความละเอียดมากกว่า การค้นหาโดยใช้ดัชนีแบบ One Codevector (ไ=1)

5.1.4 การค้นหาโดยใช้จำนวน Codevector ที่แตกต่างกัน ยิ่งจำนวน Codevector มาก การค้นหาจะยิ่งมีความละเอียดมากขึ้น

5.2 ปัญหาที่พบในการทำโครงงานวิจัย

5.2.1 ในแต่ละไฟล์วิดีโอ ถ้ามีจำนวนช็อตมาก จำนวนเฟรมรูปภาพก็จะมีจำนวนมากเช่นกัน อาจจะต้องใช้เวลาในการประมวลผลเป็นเวลานาน เพราะเป็นการทำดัชนีจากไฟล์ทั้งหมดที่ได้ทำการโหลดเข้าสู่โปรแกรม ซึ่งผู้ใช้เป็นผู้กำหนดว่าจะมากหรือน้อย เวลาในการประมวลผลจึงขึ้นอยู่กับจำนวนไฟล์ที่โหลดเข้าสู่โปรแกรม

5.2.2 ในการติดต่อกับฐานข้อมูลบ่อยครั้งทำให้การประมวลผลใช้เวลานาน

5.2.3 ค่าสีที่ได้จากกระบวนการ Vector Quantization ในบางไฟล์วิดีโอ จะได้ค่าที่ใกล้เคียงกันในหลายๆเฟรม ซึ่งอาจจะมีผลกระทบต่อการนำไปใช้ในระบบการสืบค้น

5.2.4 โปรแกรมสำเร็จรูป Video Editor 7.0 ดังกล่าว ยังไม่ใช่โปรแกรมที่สามารถแยกช็อตเองได้อย่างอัตโนมัติ ผู้ใช้ต้องทำการกำหนดช็อตเอง ซึ่งก็ทำให้เสียเวลาในส่วนหนึ่ง

5.2.4 ใช้ได้กับวิดีโอไฟล์ .avi เท่านั้น

5.2.4 เวลาที่ใช้ในการหาดัชนี หากมีจำนวนเฟรมมากก็จะยิ่งใช้เวลานาน

5.3 ข้อเสนอแนะ

เนื่องจากโครงงานนี้เป็นการเริ่มการวิจัยโครงงานในขั้นแรก ถ้ามีผู้ที่ต้องการพัฒนาโครงงานนี้ต่อไป ทางผู้จัดทำโครงงานวิจัยในครั้งนี้มีข้อเสนอแนะนำดังต่อไปนี้

5.3.1 ควรใช้ทฤษฎีลักษณะพื้นฐานของภาพแบบอื่นด้วย เช่น ลักษณะพื้นผิว (Texture) รูปร่างของภาพ (Shape) และเทคนิคการตัดภาพ (Image Segmentation) ซึ่งจะทำให้การทำงานมีประสิทธิภาพ

5.3.2 เนื่องจากในขั้นตอนการทำ Vector Quantization ของภาพในแต่ละครั้ง อาจมีความคลาดเคลื่อนของค่าของแต่ละสี ค่าที่ได้อาจจะเปลี่ยนแปลงไป ซึ่งมีผลต่อการนำไปใช้ในการสืบค้น

5.3.3 การเขียนโปรแกรมทุกครั้งควรเขียนให้โปรแกรมคืนค่า Memory ด้วยทุกครั้งเพื่อป้องกันการ Error ของคอมพิวเตอร์

5.3.4 เนื่องจากโปรแกรมใช้เวลานานในการรันผล index จึงควรเขียนโปรแกรมแบบ multi-thread เพื่อแบ่งการทำงานของ CPU ให้ทำงานได้เร็วยิ่งขึ้นและประหยัด memory

5.4 แนวทางในการพัฒนาโครงงานวิจัย

5.4.1 พัฒนาให้ซอฟแวร์เก็บดัชนีโดยอัตโนมัติ

5.4.2 พัฒนาให้สามารถใช้ได้กับไฟล์วิดีโอได้หลายๆ ฟอร์แมต เช่น .MPEG, .WMA เป็นต้น

5.4.3 ควรศึกษาและพัฒนาโปรแกรมในส่วนของการแยกช็อตของไฟล์วิดีโอ โดยไม่ใช้โปรแกรมสำเร็จรูป และควรหาหลักการ วิธีการที่ได้ประสิทธิภาพมากที่สุด

เอกสารอ้างอิง

[1] "ทฤษฎีสี" [Online] : Available
http://www.prc.ac.th/newart/webart/colour01.html. 2007.

[2] "Histogram" [Online] : Available
http://terhardcore.multiply.com/journal/item/18.2007

[3] Giuseppe Patane and Marco Russo. "The Enhanced LBG Algorithm." [Online] : Available
http://ai.unime.it/~gp/publications/full/elbg.pdf

[4] "Vector Quantization" [Online] : Available
http://www.data-compression.com/vq.shtml.2007

[5] Anderberg, M. (1973). Cluster Analysis for Applications. New York: Academic.

[6] Bezdek, J., & Pal, N. (1995). Two Soft Relatives of Learning Vector Quantization. Neural Networks, 8 (5), 729-743.

[7] Chinrungrueng, C., & Sequin, C. (1995). Optimal adaptive K-Means Algorithm with Dynamic Adjustment of Learning Rate. IEEE Transaction on Neural Networks, 6 (1), 157-169.

[8] Cosman, P., Gray, R., & Vetterli, M. (1996). Vector Quantization of Image Subbands : A Survey. IEEE Transactions on Image Processing, 5 (2), 202–225.

[9] Fritzke, B. (1997). The LBG-U Method for Vector Quantization – an Improvement Over LBG Inspired from Neural Network. Neural ProcessingLetters, 5 (1), 35–45.

[10] Fukunaga, K. (1990). Introduction to Statistical Pattern Recognition (Second ed.). 24–28 Oval Road, London NW1 7DX: Academic Press Limited.

[11] สิทธิโชค ยอดระยับ. **"การเขียนโปรแกรม Digital Image Processing ด้วย Visual Basic"**. สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น). สิงหาคม 2550.

[12] "Item-based Collaborative Filtering #2" [Online] : Available
http://www.narisa.com/blog/iwat/index.php?showentry=388

[13] "การเรียงลำดับ" [Online] : Available
http://rbu.rbru.ac.th/~datastru/lesson/lesson8.htm

ประวัติผู้เขียนโครงงาน

ชื่อ นางสาวณัฐกานต์ ประเสริฐสังห์

ภูมิลำเนา 166 หมู่ 1 ตำบลหย่วน อำเภอเชียงคำ จังหวัดพะเยา 56110

ประวัติการศึกษา

- จบมัธยมจากโรงเรียนจุฬาภรณราชวิทยาลัย เชียงราย จังหวัดเชียงราย
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
  มหาวิทยาลัยนเรศวร

E-mail: chobit_s@hotmail.com

ชื่อ นางสาวนริศรา สิงห์เชื้อ

ภูมิลำเนา 14 หมู่ 2 ตำบลบ้านเป้า อำเภอเมือง จังหวัดลำปาง 52000

ประวัติการศึกษา

- จบมัธยมจากโรงเรียนเขลางค์นคร จังหวัดลำปาง
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
  มหาวิทยาลัยนเรศวร

E-mail: omeki_nb@hotmail.com