

*Republic of Iraq
Ministry of Higher Education and Scientific Research
Al-Nahrain University
College of Science*



Video Retrieval Using Different Color Models

*A Thesis Submitted to the College of Science, AL-Nahrain
University In Partial Fulfillment of the Requirements for
The Degree of Master of Science in Computer Science*

Submitted by:

Jaber Abdullah Jaber

(B.Sc. 2005)

Supervised by:

Dr. Sawsan K. Thamer

December 2009

Thu Alhija 1430

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

وَأَشْكُرُكَ بِحَسْبِ الْإِنْسَانِ

وَأَشْكُرُكَ بِحَسْبِ الْإِنْسَانِ

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

سُبْحَانَكَ اللَّهُمَّ رَبِّيَ الْأَكْبَرُ

Supervisor Certification

I certify that this thesis was prepared under our supervision at the Department of Computer Science/College of Science/Al-Nahrain University, by **Jaber Abdullah Jaber** as partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Supervisor

Signature:

Name : **Dr. Sawsan K. Thamer**

Title : **Lecturer**

Date : / / **2009**

The Head of the Department Certification

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature:

Name : **Dr. Taha S. Bashaga**

Title : **Head of the department of Computer Science,
Al-Nahrain University.**

Date : / / **2009**

Certification of the Examination Committee

We certify that we have read this thesis and as an examining committee, examined the student in its content and what is related to it, and that in our opinion it meets the standard of a thesis for the degree of Master of Science in Computer Science.

Signature:

Name: **Dr. Loay E. George**
Title : **Assistant Professor**
Date : / / **2009**
(**Chairman**)

Signature:

Name: **Dr. Dhia Abdul Hussein**
Title : **Assistant Professor**
Date : / / **2009**
(**Member**)

Signature:

Name: **Dr. Haithem A. Al-Ani**
Title : **Lecturer**
Date : / / **2009**
(**Member**)

Signature:

Name: **Dr. Sawsan K. Thamer**
Title : **Lecturer**
Date : / / **2009**
(**Supervisor**)

Approved by the Dean of College of Science, Al-Nahrain University.

Signature:

Name: **Dr. Laith Abdul Aziz Al-Ani**
Title : **Assistant Professor**
Date : / / **2009**
(**Dean of College of Science**)

DEDICATED TO MY

PARENTS...

SISTERS...

AND BROTHER...

To everyone

Taught me a letter

Jaber



Acknowledgment

I would like to express my sincere appreciation to my supervisor, Dr. Sawzan K. Thamer, for giving me the major steps to go on to explore the subject, sharing with me the ideas in my research "Video Retrieval Using Different Color Models" and discuss the points that I felt they are important.

Grateful thanks for the Head of Department of Computer Science, Dr. Taha S. Bashaga.

Also, I wish to thank the staff of Computer Science Department at Al-Nahrain University for their help.

I would like to say "thank you" to my faithful friends for supporting and giving me advises.



Abstract

With the explosion of multimedia data (videos, audios, images, and Web pages), people have no time to look at this huge data, and human attention has become a precious resource. So, a way must be found to automatically analyze, classify, summarize, discover and characterize trends in it, and to automatically flag anomalies. Many researchers have felt the need for data mining methods to deal with this amount of data.

The Multimedia Mining work includes various kinds of data like video, audio, image, etc. This work includes a set of fields, which process huge information, like clustering, recognition, classification, etc.

Video Mining is the most important kind of mining because the video became a popular easy way to deliver a message or an idea like the video chat, movies, advertisements and also a way for protection like in surveillance videos and a lot of other uses that the world is full of.

In this research, a Video Retrieval System (VRS) was implemented. This system has two phases: enrollment phase and query-matching phase. The enrollment phase constructs the videos' database. This database contains different classes that clustered according to their textural (Run Length) or shape (Moment Invariant) features with different color models (HSL or YC_bC_r). The clustering process is accomplished using K-means clustering algorithm. While, query-matching phase used for retrieving videos which are similar to an input one. Before extracting features, the input video is cut into several shots depending on the difference between successive frames histograms, then calculating the features to a chosen shot to retrieve the similar shots from the videos' database.

The performance of the system was computed using recall and precision measures. In VRS, an encouraged good result was obtained.

List of Abbreviations

Abbreviation	Meaning
AVI	Audio/Video Interleaved
ASB	Abrupt Shot Boundary
BMP	Bitmap
CBIR	Content Based Image Retrieval
CHMM	Coupled Hidden Markov Model
DB	Database
fps	frame per second
EM	Expectation Maximization
GLNU	Gray Level Non Uniformity
GLRL	Gray Level Run Length
GMM	Gaussian Mixture Model
GSB	Gradual Shot Boundary
HGRE	High Gray level Runs Emphasis
HMM	Hidden Markov Model
HSB	Hue, Saturation, Brightness
HSI	Hue, Saturation, Intensity
HSL	Hue, Saturation, Luminance
HSV	Hue, Saturation, Value
ISODATA	Iterative Self-Organization Data Analysis Technique Algorithm
ITM	Integral Template Match
LGRE	Low Gray level Runs Emphasis
LRE	Long Run Emphasis
LRHGE	Long Run High Gray level Emphasis
LRLGE	Long Run Low Gray level Emphasis
MAD	Mean Absolute Difference
RGB	Red, Green, Blue
RLN	Run Length Non uniformity
RF	Relevance Feedback
RP	Run Percentage
SBD	Shot Boundary Detection
SRE	Short Run Emphasis
SRLGE	Short Run Low Gray level Emphasis
SSE	Some of Square Error
VM	Video Mining

Table of contents

Chapter One: Overview

1.1 Introduction	1
1.2 Data Mining	1
1.3 Multimedia Data Mining	2
1.4 Video Mining	5
1.5 Literature Survey	7
1.6 Aim of Work	11
1.7 Thesis Layout	11

Chapter Two: Theoretical Background

2.1 Introduction	13
2.2 Video Mining	13
2.3 Video	14
2.3.1 Video Basic Structure	15
2.3.2 Shot Boundary Detection	16
2.3.3 Color Histogram	17
2.4 Color Models	18
2.4.1 YCbCr Color Model	20
2.4.2 HSL Color Model	21
2.5 Features Extraction	23
2.5.1 Shape Features	23
2.5.2 Texture Features	25
2.6 Down-Sampling Methods	29
2.6.1 Median Representation	29
2.6.2 Average Representation	30
2.7 Classification	31
2.8 Clustering	31
2.8.1 K-means Method Technique	32

Chapter Three: The Video Mining and Retrieval System

3.1 Introduction	36
3.2 The VRS Model	36
3.3 Video database construction module	38
3.3.1 Loading AVI video files and frames extraction	39
3.3.2 RGB to HSL Conversion	41
3.3.3 RGB to YC _b C _r Conversion	43
3.3.4 Down-Sampling	44
3.3.5 Run-length features extraction	44
3.3.6 Moment Invariants features extraction	49
3.3.7 Decision Maker	51
3.3.8 K-means clustering	52
3.4 Video Retrieval	57
3.4.1 Shot Detection	57
3.4.2 video recognition	58

Chapter Four: Tests and Results

4.1 Introduction	60
4.2 Video Test Material	60
4.3 Test Strategy	60
4.3.1 Testing the Shot Detection	61
4.3.2 Testing the Video Retrieval	64

Chapter Five: Conclusions and Future Work

5.1 Conclusions	77
5.2 Future Work	78
References	79

Appendix A



Chapter One

Overview

Chapter One

Overview

1.1 Introduction

The rapid growth of multimedia resources has generated an urgent need for techniques to process and analyze this special kind of data. Traditional machine learning methods cannot meet the special requirements in understanding the semantic meaning of and extracting knowledge from raw multimedia data. Image, video, audio data are specific multimedia data types. It is inherently hard to make the machine understand the meaning of these data by only reading pixels, frames or signals. There exists a “semantic gap” between the low level features and the high level semantic meaning. Therefore, it is necessary that human provide some guidance to the machine. [Xin06a]

People have no time to look at this data. Human attention has become the precious resource. So, ways must be found to automatically analyze, classify, summarize, discover and characterize trends in the data, and flag anomalies. This is one of the most active and exciting areas of the database research community. [Jia06]

1.2 Data Mining

Data mining is the extraction of implicit, previously unknown, and potentially useful information from data. The idea is to build computer programs that sift through databases automatically, seeking regularities or patterns. Strong patterns, if found, will likely generalize to make accurate predictions on future data. [Ian05]

Data mining has attracted a great deal of attention in the information industry and in society as a whole, due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge. The information and knowledge gained can be used for applications ranging from market analysis, fraud detection, and customer retention, to production control and science exploration. [Jia06]

1.3 Multimedia Data Mining

The word *multimedia* assumes that several data sources of different modalities are processed at the same time. Multimedia data could be semi structured (Web pages, XML documents) or unstructured (text, images, audio, and video recordings). A data mining project can deal with only one modality of data, for example, customers' audio recordings or surveillance video. [Val07]

Multimedia data mining, as the name suggests, presumably is a combination of the two emerging areas: multimedia and data mining. However, multimedia data mining is not a research area that just simply combines the research of multimedia and data mining together. Instead, the multimedia data mining research focuses on the theme of combining multimedia and data mining research together to exploit the synergy between the two areas to promote the understanding and to advance the development of the knowledge discovery in multimedia data. Consequently, multimedia data mining exhibits itself as a unique and distinct research area that synergistically relies on the state-of-the-art research in multimedia and data mining but at the same time fundamentally differs from either multimedia or data mining or a simple combination of the two areas. [Zho09]

Consequently, with the independent and advanced developments of the two areas of multimedia and data mining, with today's explosion of

the data scale and the existence of the pluralism of the data media types, it is natural to evolve into this new area called multimedia data mining. While it is presumably true that multimedia data mining is a combination of the research between multimedia and data mining, the research in multimedia data mining refers to the synergistic application of knowledge discovery theory and techniques in a multimedia database or collection. As a result, “inherited” from its two parent areas of multimedia and data mining, multimedia data mining by nature is also an interdisciplinary and multidisciplinary area; in addition to the two parent areas, multimedia data mining also relies on the research from many other areas, noticeably from mathematics, statistics, machine learning, computer vision, and pattern recognition. Figure 1.1 illustrates the relationships among these interconnected areas. [Zho09]

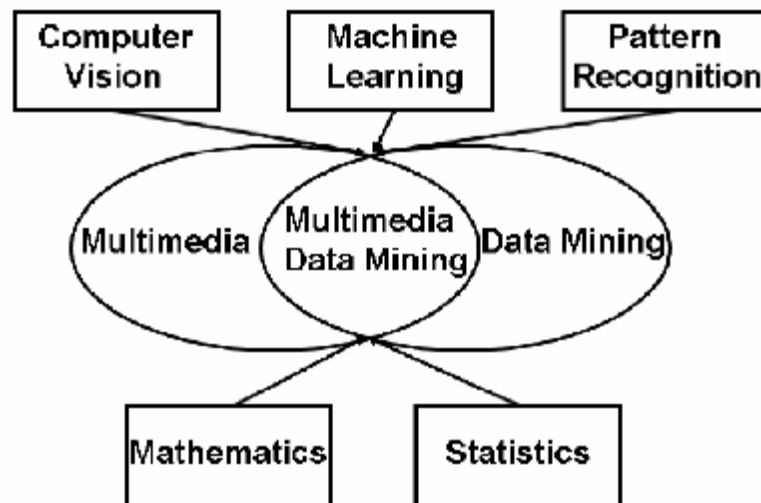


Figure 1.1: Relationships among the interconnected areas to multimedia data mining.

Advances in modern multimedia technologies have led to huge and ever-growing archives of image, audio and video in diverse application areas such as medicine, remote sensing, entertainment, education and online information services. Among all these media types, video is the

most challenging one, as it combines all other media information into a single data stream. Due to the decreasing cost of storage devices, higher transmission rates and improved compression techniques, digital video becomes available at an ever increasing rate. [Cha05]

A typical multimedia data mining system, or framework, or method always consists of three key components. Given the raw multimedia data:

1. The very first step for mining the multimedia data is to convert a specific raw data collection (or a database) into a representation in an abstract space which is called the feature space. This process is called feature extraction. Consequently, feature representation method is needed to convert the raw multimedia data to features in the feature space, before any mining activities are able to be conducted.
2. Therefore, the second key component is the knowledge representation, i.e., how to effectively represent the required knowledge to support the expected knowledge discovery activities in a multimedia database.
3. Finally, the last key component is the actual mining or learning theory and/or technique to be used for the knowledge discovery in a multimedia database. In addition to the three key components, in many multimedia data mining systems, there are user interfaces to facilitate the communications between the users and the mining systems. [Zho09]

Figure 1.2 illustrates this typical architecture of a multimedia data mining system.

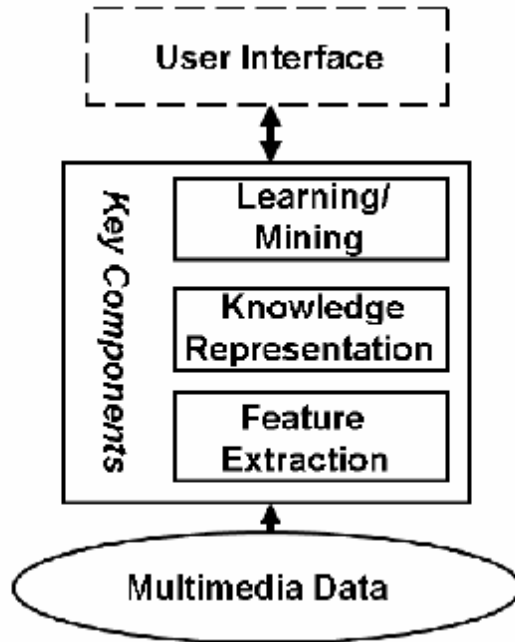


Figure 1.2: The typical architecture of a multimedia data mining system.

1.4 Video Mining

Video data are very easy to be captured and stored. The problem of how to automatically extract user needed semantic information from video data has deeply attracted researchers and enterprisers. Video understanding is just the ultimate target of computer vision. Not only the semantic information will be gotten, but also a lot of interesting patterns and knowledge from video, for there existing much spatial -temporal distributed information in video content and structure. The extracting technologies are badly needed in applications such as smart surveillance and video retrieval. Progresses in data mining have made some researchers pay attention to video data mining technologies to get the targets. [Azr03]

Video data is a kind of unstructured stream; efficient access to video is not an easy task. Generally, there are two kinds of videos: videos with some content structure and videos without any content structure. The former are videos such as movies and news where scenarios are used to convey video content. For “raw” videos like surveillance videos, they have

no scene change, therefore no content structure can be found among them. Videos with content structure information are usually edited (or post-processed) by editors (or directors), where various shots are packed back and forth to convey scenarios or context information, such as "dialog" scene. [Cha05]

The visual content, or generally content, of images and video frames can be categorized as follows: spatial, semantic, and low-level. Since video data has a time dimension, the spatio-temporal content of a video data is also considered. However, extracting spatio-temporal content requires sophisticated techniques, thus do not included in the categorization. The spatial content of an image is the relative positioning of the objects residing in the image. The semantic content is the actual meaning of the image that a user captures when he/she looks at the image. The low-level content is formed by low-level features such as color, shape, and texture. These three features are considered important underlying primitives in human visual perceptions of the real world. Various methods exist in the literature for indexing the images based on these low-level features. [Eyu02]

Video Mining (VM) has been considered as the further processing of video content. Through automatically or unsupervised mining, useful semantic information, even patterns or knowledge would be extracted from video data. The intention is according to the concepts of data mining. As viewed from its functions, VM has defined as a process which can not only automatically extract content and structure of video, features of moving objects, spatial or temporal correlations of those features, but discover patterns of video structure, objects activities, video events, etc from vast amounts of video data without little assumption of their contents. By using video mining techniques, video summarization, classification, retrieval,

abnormal events alarm and other smart video applications can be implemented. [Kex06]

1.5 Literature Survey

Many researchers have considered VM as headlines in their work; some of their published works are the following:

1. João Miguel da Costa Magalhães (2001) "**Semantic Multimedia: Mining, Fusion and Extraction**" [Joa01] The aim of this PhD work was to capture the semantic knowledge existing in multimedia content and store it for later use in a retrieval or summarization application. The problem is approached as a statistical learning problem devised in three main parts: (1) multi-modal multimedia mining: automatic features extraction and patterns discovery; (2) multi-modal fusion: multi-modal features will be combined in a probabilistic learning network and mapped into a multimedia semantic network; and (3) semantic knowledge extraction: given the high-level mapping and the domain knowledge represented in the semantic network, the most probable semantic description will be deduced.
2. Dong and Yu-Fei (2003) "**A Novel Motion-Based Representation For Video Mining**" [Don03] proposed a novel motion-based representation for video mining tasks, including a fast dominant motion extraction scheme, called *Integral Template Match* (ITM), and a set of qualitative and quantitative description schemes. It is applied to such video mining tasks as shot boundary detection, camera motion segmentation and recognition.
3. Ajay D. and Kudir A. (2004) "**Video Mining: Pattern Discovery Versus Pattern Recognition**" [Aja04] examined the significance of video mining as pattern discovery in multimedia content. The

underlying issue of pattern discovery versus pattern recognition is examined, since most past work has not drawn such a sharp distinction. The term “pattern discovery” implies a purely unsupervised approach. But in practice a mixture of unsupervised and supervised techniques will have to be used. Conventional data mining is compared with video mining and observes that a key difference is in the multi-layered semantics of multimedia content. Then significant challenges is posed and identified by video mining.

4. Tao and Yu-Fei (2005) "**Sports Video Mining with Mosaic**" [Tao05] proposed a generic approach to key event as well as structure mining for sports video analysis. Mosaic is generated for each shot as the representative image of shot content. Based on mosaic, sports video is mined by the method with prior knowledge and without prior knowledge. Without prior knowledge, the system may locate plays by discriminating those segments without essential content, such as breaks. If prior knowledge is available, the key-events in plays are detected using robust features extracted from mosaic. Experimental results have demonstrated the effectiveness and robustness of this sports video mining approach.
5. Hanning and Don (2006) "**Unusual Event Detection via Multi-camera Video Mining**" [Han06] described a framework for detecting unusual events in surveillance videos. Most surveillance systems consist of multiple video streams, but traditional event detection systems treat individual video streams independently or combine them in the feature extraction level through geometric reconstruction. This framework combines multiple video streams in the inference level, with a Coupled Hidden Markov Model (CHMM). Two-stage training is used to bootstrap a set of usual

events, and train a CHMM over the set. By thresholding the likelihood of a test segment being generated by the model, an unusual event detector is built. The performance of this detector is evaluated through qualitative and quantitative experiments on two sets of real world videos.

6. Xin and Chengcui (2006) "**An Interactive Semantic Video Mining and Retrieval Platform –Application in Transportation Surveillance Video for Incident Detection**" [Xin06a] proposed an interactive platform for semantic video mining and retrieval using Relevance Feedback (RF), a popular technique in the area of Content-Based Image Retrieval (CBIR). By tracking semantic objects in a video and then modeling spatio-temporal events based on object trajectories and object interactions, the proposed interactive learning algorithm in the platform is able to mine the spatio-temporal data extracted from the video. An iterative learning process is involved in the proposed platform, which is guided by the user's response to the retrieved results. Although the proposed video retrieval platform is intended for general use and can be tailored to many applications, the focusing was on its application in traffic surveillance video database retrieval to demonstrate the design details. The effectiveness of the algorithm is demonstrated by our experiments on real-life traffic surveillance videos.
7. Hind Ali (2006) "**Video Data Mining Using Color and Textural Features Analysis**" [Hin06] aimed to segment the video into a number of shots using three different types of algorithms, classify the video frames data into static and dynamic blocks of videos, the adopted features are statistical features (mean, standard deviation, mean absolute deviation, skewness), and textural features (energy of gradient, contrast, modification, fractal dimension H). The test

results indicated that some combinations of these features are useful to successfully recognize the video shots from each other. The results also showed that the features extracted from the statistic blocks gave higher discrimination power than the features extracted from the dynamic blocks. The k-means clustering algorithm was used to categorize the videos' shots into a number of classes. The test results indicated that this algorithm showed good stability in classifying the video shots, because most of the extracted sequences from each shot were classified as members to the same class.

8. Yi Ding and Guoliang (2007) "**Two-Layer Generative Models For Sport Video Mining**" [YiD07b] presented a two-layer generative model for sport video mining that is composed of a two-layer observation model. The first layer is the Gaussian Mixture Model (GMM) using frame wise camera motion for intra-shot analysis and the second layer is the Hidden Markov Model (HMM) involving the GMM as the mid-level observation for inter-shot analysis. A recursive model estimation method is developed for statistical inference which combines two Expectation Maximization (EM) algorithms. Specifically, the proposed generative model is used for American football play analysis where each play shot is classified into one of four classes, i.e., short plays, long plays, kicks and field goals. The experimental results show good promising classification performance.
9. Hasnaa Imad "**Content-Based Image Retrieval System using Fuzzy Rule**" [Has08] aimed to design and implement a Content-Based Image Retrieval System that based on utilizing fuzzy logic with other mechanisms. In this work, some of the fuzzy set concepts were applied to compute the membership value for each

feature in each image. The established system should be capable of interactively retrieving images that are semantically related to the user's query from a database.

1.6 Aim of work

The aim of this work is to implement an effective system for retrieving similar videos from a stored video database depending on the visual video contents according to a query input video. This video content-based retrieval system implemented using a combination between texture features and HSL color attributes that gives good results.

The implemented Video Retrieval System (VRS) used statistical methods to compute videos' features and try to use different color attributes with them.

The goal from that is to find the closer videos from the query video shot using the combination that can satisfy better retrieving results.

1.7 Thesis Layout

In addition to chapter one, the remaining parts of this thesis consists of the following chapters:

Chapter Two: (Theoretical Background)

In this chapter the Video Mining (VM) definition is introduced, what does it mean and how it works. The video definition is also mentioned with its common or basic structure and the boundaries that separate the video shots and how the last detected using the difference between frames' color histograms.

Chapter Three: (Video Retrieval System)

In this chapter, the proposed system design and implementation steps are given. The video retrieval modules are described in details.

Chapter Four: (Tests and Results)

This chapter is dedicated to present the results of the conducted tests on the established retrieval system using different AVI test videos.

Chapter Five: (Conclusions and Future Work)

Some conclusion remarks that derived from the analysis of test results are given in this chapter. Also, some suggestions for future work are listed.

Chapter Two

Theoretical Background

Chapter Two

Theoretical Background

2.1 Introduction

This chapter introduces the Video Mining (VM) definition, what does it mean and how it works. The video definition is also mentioned with its common or basic structure and the boundaries that separate the video shots and how the boundary detected using the difference between frames' color histograms.

Two different color spaces (i.e., YC_bC_r and HSL) are described and how converted from RGB color model which is the basic model. Also two of the famous ways of features extraction (shape and texture) were explained with the equations that deal with them.

The Down Sampling Methods are described with different equations to minimize the calculations that lead to a less execution time; also the implementation of the down-sampling on the images was described with figures. The k-means was described as a process for classifying the videos into number of classes.

2.2 Video Mining

Video Mining (VM) is the process of discovering the implicit and previously unknown knowledge or interesting patterns from a massive set of video data. Researches on VM are still in its infancy, it is a new research field, the concept and methods of VM need thorough research and study. [Cha05]

Work in video data mining can be divided into two categories: mining similar motion patterns and mining similar objects. The first type

of systems uses motion information to mine similar event patterns or identify peculiar events. [Ara07]

It is interesting in that the raw data in videos is expressed in a way that is not directly amenable to the use of conventional mining techniques. Novel ways of representing the raw, pixel-level data, which lend themselves to the capturing of patterns, are needed, and ways of representing the patterns themselves are required. There is a lot of variation in the details of the patterns and it is important to abstract out the key features of a behavior, perhaps by some sort of ‘coarsening’ of the record of activities. By this means, unwanted or irrelevant details and noise can be filtered out. [Bel07]

The second category systems aim to group frequently appearing objects in videos. Different appearances of the same object are defined in different parts of the video as different instances of that object. The purpose of the object mining system is to group these different instances of the same object. The problem is difficult because the object can appear in different ways in different parts of the video due to different imaging conditions, lightening conditions, back grounds and occlusions. [Ara07]

2.3 Video

Video structure is difficult to understand; some videos still have their contents structure. Generally, there are two kinds of videos: videos with some content structure and videos without any content structure. The former are videos such as movies and news where scenarios are used to convey video content. For “raw” videos, like surveillance videos, they have no scene change, therefore no content structure can be found among them. Videos with content structure information are usually edited (or post-processed) by editors (or directors), where various shots are packed back and forth to convey scenarios or context information, such as

“dialog” scene. Even though, other videos have not such a contents structure as “dialog” scene and news videos, but many videos are a broken sequence of frames recorded from a single camera motion (shot) that can also express the contents structure of the video. [Cha05]

2.3.1 Video Basic Structure

In content-based video analysis, video can be analyzed in five structured levels as shown in Figure 2.1. For the nature of the analysis involved in these five structured levels, the domain dependency tends to increase and the computability/analysis accuracy tends to decrease toward the higher structured levels. Specifically, in frame level analysis, low level features such as color, texture, shape, motion and audio are generally used and the analysis requires no or minimum domain knowledge. At this level, many Shot Boundary Detection (SBD) methods have been proposed to segment video into shots, each of which can then be represented by one or a few key frames from the shot. [YiJ06]

A frame is a static image and minimum logic unit of video. A shot is an uninterrupted segment of video frame sequence with static or continuous camera motion, while a scene is a series of shots that are coherent from the narrative point of view. These shots are either shot in the same place or they share similar thematic content. [Cha05]

Based on the detected shots and the clustered or segmented scenes, one or more key frames can be extracted. Afterwards, image features such as color, shape and texture are used to index these key frames. In addition, high-level representations such as regions, objects, and motion can also be used to infer semantic events of interest and help summarize the content of the whole video sequence. Finally, videos can be browsed and retrieved based on the similarity between the features or events of the query video sequence and the video sequences in the database. [YiJ06]

A *video basic structure mining* is defined as the process of discovering fundamental logic structure from preprocessed video program, adopts data mining method such as classification, clustering and association rule etc. [Cha05]

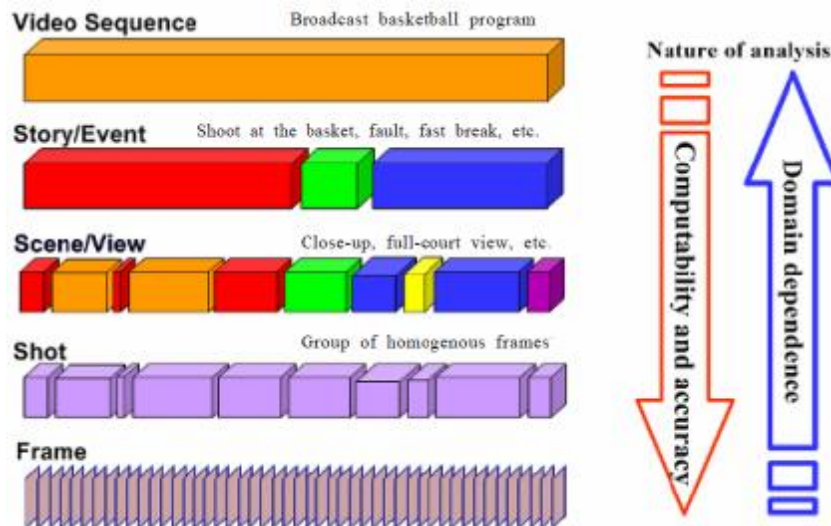


Figure 2.1: Video Basic Structure

2.3.2 Shot Boundary Detection

Shot Boundary Detection (SBD) is commonly the first step in the process of indexing, characterizing and retrieving of video. A shot is a temporal sequence of frames generated and recorded continuously by a single camera act, which usually depicts a continuous action without significant content changes.

To form a video sequence, shots are joined together during video sorting and post editing with either abrupt cuts or gradual visual effects according to the nature of the scene changes or story sequences. There are generally two types of shot transitions: Abrupt Shot Boundary (ASB), also known as shot cut or hard cut, where the change of video content occurs over a single frame; and Gradual Shot Boundary (GSB), such as

fade in, fade out, dissolve and wipe, where the change takes place gradually over a short sequence of frames.

The main technique of SBD is to detect the change between each consecutive frame pair. If the change exceeds a predetermined threshold, a shot boundary is assumed. In the literature, the change is usually measured by using such low-level video features as pixel intensities, color histograms, edges and compressed video attributes. [YiJ06]

The color histogram distance has been widely used as a measure of (dis)similarity between images for the purposes of object recognition, content-based retrieval, and temporal video segmentation. A histogram is first computed for each image in the sequence and the distance between successive histograms is used as a measure of local activity. As shown in equation 2.1:

$$D(a, b) = \sum_{i=1}^B |a_i - b_i|, \dots \dots \dots (2.1)$$

Where a and b are histograms of successive frames, and B the number of histogram bins. This metric (D) has been shown to perform well for temporal video segmentation and is equivalent (for normalized histograms) to the histogram intersection metric commonly used in retrieval literature. [Nun00]

2.3.3 Color Histogram

The color histogram serves as an effective representation of the color content of an image if the color pattern is unique compared with the rest of the data set. The color histogram is easy to compute and effective in characterizing both the global and local distribution of colors in an image. In addition, it is robust to translation and rotation about the view axis and changes only slowly with the scale, occlusion and viewing angle.

2.4 Color Models (spaces)

Color is perceived by humans as a combination of tri-stimuli R (red), G (green), and B (blue) who are usually called three primary colors. From R, G, B representation, other kinds of color representations (spaces) can be derived by using either linear or nonlinear transformations. Several color spaces, such as RGB, HIS, CIE $L^*u^*v^*$ are utilized in color image segmentation, but none of them can dominate the others for all kinds of color images. Selecting the best color space still is one of the difficulties in color image segmentation.

Red, green, and blue components can be represented by the brightness values of the scene obtained through three separate filters (red, green, and blue filters) based on the following equations:

$$R = \int_{\lambda} E(\lambda) S_R(\lambda) d\lambda \dots\dots\dots (2.2)$$

$$G = \int_{\lambda} E(\lambda) S_G(\lambda) d\lambda \dots\dots\dots (2.3)$$

$$B = \int_{\lambda} E(\lambda) S_B(\lambda) d\lambda \dots\dots\dots (2.4)$$

Where S_R , S_G , S_B are the color filters on the incoming light or radiance $E(\lambda)$, and λ is the wavelength. [Che01]

Each pixel of the image can be represented as a point in a 3D color space. Commonly used color space for image retrieval include *RGB*, *Munsell*, *CIE $L^*a^*b^*$* , *CIE $L^*u^*v^*$* , *HSV* (or *HSL*, *HSB*), and *opponent color* space. [Lon01]

The RGB color space can be geometrically represented in 3-dimensional cube as in figure 2.2. The coordinates of each point inside the cube represented the values of red, green, blue components, respectively.

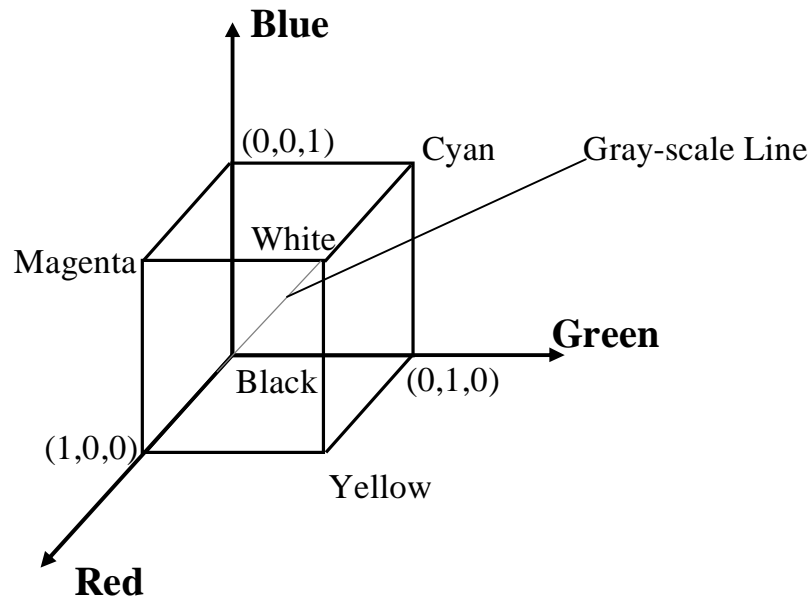


Figure 2.2: RGB Color Cube

The laws of colorimetry are: (1) any color can be created by these three colors and the combination of the three colors is unique; (2) if two colors are equivalent, they will be again equivalent after multiplying or dividing the three components by the same number; (3) the luminance of a mixture of colors is equal to the sum of luminance of each color. The tri-stimulus values that served as the color basis are: 425.8nm for blue, 546.1nm for green, and 700.0nm for red. Any color can be expressed by these three color bases.

RGB is the most commonly used model for the television system and pictures acquired by digital cameras. Video monitors display color images by modulating the intensity of the three primary colors (red, green, and blue) at each pixel of the image. RGB is suitable for color display, but not good for color scene segmentation and analysis because of the high correlation among the R, G, and B components. By high correlation, it is meant that if the intensity changes, all the three components will be changed accordingly. Also, the measurement of a color in RGB space does not represent color differences in a uniform

scale; hence, it is impossible to evaluate the similarity of two colors from their distance in RGB space. [Che01]

2.4.1 YC_bC_r Color Model (Transformation)

The Y, C_b, and C_r components are scaled and shifted versions of the analog Y, U, and V components, where the scaling and shifting operations are introduced so that the resulting components take value in the range of (0,255).

The transformation matrix is presented for deriving this coordinate from digital RGB coordinate. Assuming that the RGB values are in the range of (0-255), the YC_bC_r values are related to RGB values by:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.586 & 0.115 \\ -0.168 & -0.33 & 0.498 \\ 0.498 & -0.417 & 0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}, \dots\dots\dots (2.5)$$

In the YC_bC_r coordinate, Y reflects the luminance and is scaled to have a range of (16...235); C_b and C_r are scaled versions of color differences B - Y and R - Y, respectively. The scaling and shifting is designed so that they have a range of (16...240). The maximum value of C_r corresponds to red (C_r = 240 or R = 255, G = B = 0), whereas the minimum value yields cyan (C_r = 16 or R = 0, G = B = 255). Similarly, the maximum and minimum values of C_b correspond to blue (C_b = 240 or R = G = 0, B = 255) and yellow (C_b = 16 or R = G = 255, B = 0).

The spatial sampling rate introduced previously refers to the luminance component, Y. For the chrominance components, C_b and C_r, usually only half of the sampling rate is used; this leads to half number of pixels in each line, but the same number of lines per frame. This is known as the 4:2:2 format implying there are 2 C_b samples and 2 C_r samples for

every 4 Y samples. To further reduce the required data rate also the 4:1:1 format defined, in which the chrominance components are sub sampled along each line by a factor of 4, i.e., there are 1 C_b sample and 1 C_r sample for every 4 Y samples. This sampling method, however, yields very asymmetric resolutions in the horizontal and vertical directions. Another sampling format is therefore developed, which sub samples the C_b and C_r components by half in both the horizontal and vertical directions. In this format, there are also 1 C_b sample and 1 C_r sample for every 4 Y samples. But to avoid the confusion with the previously defined 4:1:1 format, this format is designated as 4:2:0. As shown in figure 2.3. [Wan02]

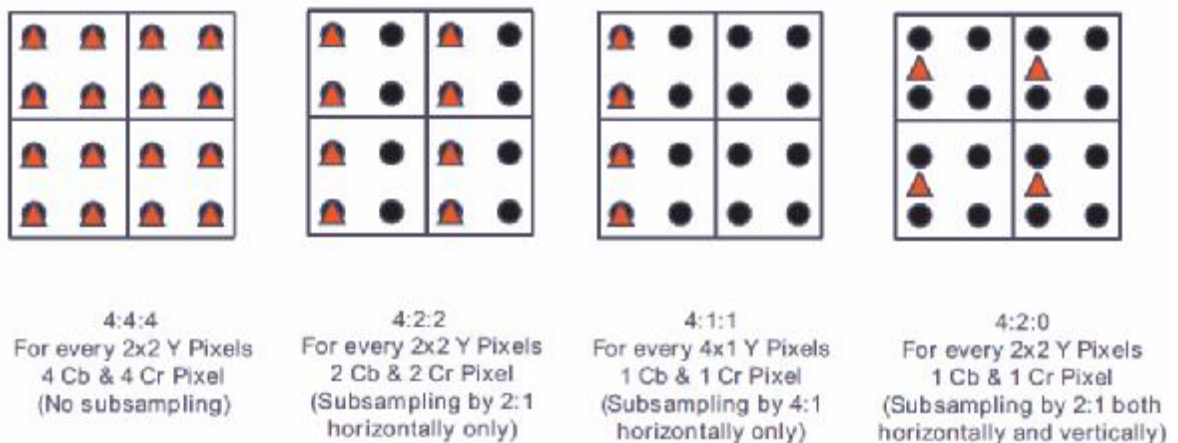


Figure 2.3: The formats of the YCbCr

2.4.2 HSL Color Model (Transformation)

HSL and HSV are two related representations of points in an RGB color model that attempt to describe perceptual color relationships more accurately than RGB, while remaining computationally simple. *HSL* stands for hue, saturation and lightness, while *HSV* stands for hue, saturation and value.

HSI and HSB are alternative names for these concepts, often substituting *intensity* and *brightness* for lightness and/or value; their

definitions are less standardized, but they are typically interpreted to be synonymous with HSL and HSV. Figure (2.4) shows the representation of the HSL and its relation to the HSV.

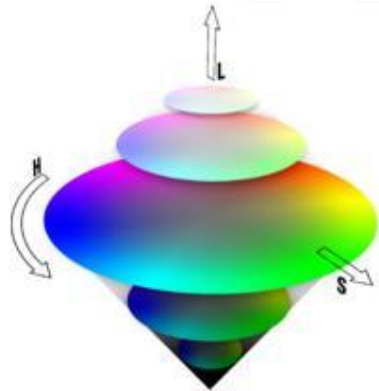


Figure 2.4 : HSL arranged as a double cone

The two representations are similar in purpose, but differ somewhat in approach. Both are mathematically cylindrical, but while HSV (Hue, Saturation, Value) can be thought of conceptually as an inverted cone of colors (with a black point at the bottom, and fully-saturated colors around a circle at the top), HSL conceptually represents a double-cone or sphere (with white at the top, black at the bottom, and the fully-saturated colors around the edge of a horizontal cross-section with middle gray at its center). Note that while “hue” in HSL and HSV refers to the same attribute, their definitions of “saturation” differ dramatically. [HSL09]

HSL color representation gained using the following equations:

$$h = \begin{cases} 0, & \text{if } \max = \min \\ (60^\circ \times \frac{g-b}{\max - \min} + 360^\circ) \bmod 360^\circ, & \text{if } \max = r \\ 60^\circ \times \frac{b-r}{\max - \min} + 120^\circ, & \text{if } \max = g \\ 60^\circ \times \frac{r-g}{\max - \min} + 240^\circ, & \text{if } \max = b \end{cases} \dots (2.6)$$

$$s = \begin{cases} 0, & \text{if } \max = \min \\ \frac{\max - \min}{\max + \min} = \frac{\max - \min}{2l}, & \text{if } l \leq \frac{1}{2} \\ \frac{\max - \min}{2 - (\max + \min)} = \frac{\max - \min}{2 - 2l}, & \text{if } l > \frac{1}{2} \end{cases}, \dots\dots\dots (2.7)$$

$$l = \frac{1}{2}(\max + \min), \dots\dots\dots (2.8)$$

Where, the *min* and *max* stands for the maximum and the minimum values of the R, G, and B color representations.

2.5 Features Extraction

Features extraction is a process that extracts a set of new features from the original features through some functional mapping. [Wes05] Feature extraction deals with how the specific traits of content information can be identified and extracted from the content-rich data. Feature description specifies how those features can be described and organized for efficient retrieval processing. Lastly, proximity evaluation provides the specification in which similarities among contents can be measured based on their features. Two types of features will be taken, *shape* and *texture* features: [Lin01]

2.5.1 Shape Features

Shape features of objects or regions have been used in many content-based image retrieval systems. Compared with color and texture features, shape features are usually described after images have been segmented into regions or objects. Since robust and accurate image segmentation is difficult to achieve, the use of shape features for image retrieval has been limited to special applications where objects or regions are readily available, and as example the *moment invariants* is taken. [Lon01]

Moment Invariants

Classical shape representation uses a set of *moment invariants*. If the object R is represented as a binary image, then the central moments of order $p+q$ for the shape of object R are defined as:

$$m_{p,q} = \sum_{(x,y) \in R} (x - x_c)^p (y - y_c)^q, \dots \quad (2.9)$$

Where (x_c, y_c) is the center of object. This central moment can be normalized to be scale invariant:

$$h_{p,q} = \frac{m_{p,q}}{m_{0,0}^g}, g = \frac{p+q+2}{2}, \dots \quad (2.10)$$

These normalized central moments have been used to develop a set of seven compound spatial moments that are invariant in the continuous image domain to translation, rotation, and scale change. The *invariant moments* are defined below: [Wil01, Lon01].

$$f_1 = m_{2,0} + m_{0,2}, \dots \quad (2.11)$$

$$f_2 = (m_{2,0} - m_{0,2})^2 + 4m_{1,1}^2, \dots \quad (2.12)$$

$$f_3 = (m_{3,0} - 3m_{1,2})^2 + (m_{0,3} - 3m_{2,1})^2, \dots \quad (2.13)$$

$$f_4 = (m_{3,0} + m_{1,2})^2 + (m_{0,3} + m_{2,1})^2, \dots \quad (2.14)$$

$$f_5 = (m_{3,0} - 3m_{1,2})(m_{3,0} + m_{1,2})[(m_{3,0} + m_{1,2})^2 - 3(m_{0,3} + m_{2,1})^2] \\ + (m_{0,3} - 3m_{2,1})(m_{0,3} + m_{2,1})[(m_{0,3} + m_{2,1})^2 - 3(m_{3,0} + m_{1,2})^2], \quad (2.15)$$

$$f_6 = (m_{2,0} - m_{0,2})[(m_{3,0} + m_{1,2})^2 - (m_{0,3} + m_{2,1})^2] \\ + 4m_{1,1}(m_{3,0} + m_{1,2})(m_{0,3} - m_{2,1}), \dots \quad (2.16)$$

$$f_7 = (3m_{2,1} - m_{0,3})(m_{3,0} + m_{1,2})[(m_{3,0} + m_{1,2})^2 - 3(m_{0,3} + m_{2,1})^2],$$

$$\dots \quad (2.17)$$

2.5.2 Texture Features

Texture is very difficult to define. This difficulty is demonstrated by the number of different texture definitions attempted by vision researchers. It is simply defined as the relationships between gray levels in neighboring pixels that contribute to the overall appearance of an image. It can also be viewed as a global pattern arising from the repetition, either deterministically or randomly, of local sub-patterns. [YiJ06]

In statistical texture analysis, texture features are computed from the statistical distribution of observed combinations of intensities at specified positions relative to each other in the image. According to the number of intensity points (pixels) in each combination, statistics are classified into first-order, second-order and higher-order statistics. [Fri95]

The *Gray Level Run Length* (GLRL) method is a way of extracting higher order statistical texture features.

The Gray Level Run Length

A gray level run is a set of consecutive pixels having the same gray level value. The length of the run is the number of pixels in the run. Run length features encode textural information related to the number of times each gray level, for example, “1,” appears in the image by itself, the number of times it appears in pairs, and so on. Take for example the image:

$$I = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 1 & 0 & 0 \\ 3 & 2 & 3 & 3 \\ 3 & 2 & 2 & 2 \end{bmatrix}$$

With four possible levels of gray (N=4.) For each of the four directions (0°, 45°, 90°, and 135°) we define the corresponding run length

matrix Q_{RL} . Its (i, j) element gives the number of times a gray level $i-1$, $i=1, \dots, N_g$, appears in the image with run length j , $j = 1, 2, \dots, N_r$. This is an $N_g \times N_r$ array, where N_r is the largest possible run length in the image.

Obtained for 0° :

$$Q_{RL}(0^\circ) = \begin{bmatrix} 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

The first element of the first row of the matrix is the number of times gray level “0” appears by itself (0 for our example), the second element is the number of times it appears in pairs (2 in the example), and so on. The second row provides the same information for gray level “1” and so on.

For the 45° direction:

$$Q_{RL}(45^\circ) = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{bmatrix}$$

Based on the preceding definition of the run length matrix, the following features are defined. [Ser03]

- **Short run emphasis:**

$$SRE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \left(\frac{Q_{RL}(i, j)}{j^2} \right)}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}, \dots \dots \dots (2.18)$$

By dividing each run length value by the square of its length, short run lengths are emphasized, The denominator is the total number of runs in the image.

- **Long run emphasis:**

$$LRE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} (Q_{RL}(i, j)j^2)}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}, \dots \dots \dots (2.19)$$

Here each run length value multiplied by the square of its length, in order to give higher weight to the long runs.

• **Gray level non uniformity:**

$$GLNU = \frac{\sum_{i=1}^{N_g} \left[\sum_{j=1}^{N_r} Q_{RL}(i, j) \right]^2}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}, \dots \dots \dots (2.20)$$

High run length values will contribute most to this feature. The GLN feature will have its lowest value if the runs are evenly distributed over all gray levels.

• **Run length non uniformity:**

$$RLN = \frac{\sum_{j=1}^{N_r} \left[\sum_{i=1}^{N_g} Q_{RL}(i, j) \right]^2}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}, \dots \dots \dots (2.21)$$

The RLN feature will have its lowest value if the runs are evenly distributed over all run lengths.

• **Run percentage:**

$$RP = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}{L}, \dots \dots \dots (2.22)$$

This feature is the ratio between the total number of observed runs in the image and the total number of possible runs if all runs had a length of one. [Fri95]

• **Low gray level runs emphasis:**

$$LGRE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j) / i^2}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}, \dots \quad (2.23)$$

This feature Measures the distribution of low gray level values. The LGRE is expected large for the image with low gray level values.

• **High gray level runs emphasis:**

$$HGRE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} i^2 Q_{RL}(i, j)}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}, \dots \quad (2.24)$$

This feature Measures the distribution of high gray level values. The HGRE is expected large for the image with high gray level values.

[Don04]

• **Short run low gray-level emphasis:**

$$SRLGE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j) / (i^2 * j^2)}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}, \dots \quad (2.25)$$

This feature Measures the joint distribution of short runs and low gray level values. The SRLGE is expected large for the image with many short runs and lower gray level values.

• **Short run high gray-level emphasis:**

$$SRHGE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j) * i^2 / j^2}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}, \dots \quad (2.26)$$

This feature Measures the joint distribution of short runs and high gray level values. The SRHGE is expected large for the image with many short runs and high gray level values. [Fri95]

• **Long run low gray-level emphasis:**

$$LRLGE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j) * j^2 / i^2}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}, \dots \dots \dots (2.27)$$

This feature Measures the joint distribution of long runs and low gray level values. The LRLGE is expected large for the image with many long runs and low gray level values.

• **Long run high gray-level emphasis:**

$$LRHGE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j) * i^2 * j^2}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} Q_{RL}(i, j)}, \dots \dots \dots (2.28)$$

This feature Measures the joint distribution of long runs and high gray level values. The LRHGE is expected large for images with many long runs and high gray level values. [Don04]

2.6 Down Sampling Methods

Down sampling is a process used for minification only. It may be used to create thumbnail representations of an image. The basic idea behind down sampling process is to represent a block of adjacent pixels with one pixel. The type of down-sampling method depends on the speed and quality requirements. The most popular down-sampling methods are [Cran97]:

2.6.1 Median Representation

Median representation replaces a block of pixels with its median value; see Figure (2.5) where an nxn window is passed over the image. For each down sampled block, its pixels values are read and put into an

array, and then sorted in ascending order according to their values. The middle value is then used to represent that block. This method requires much computation time due to the number of comparisons needed to sort the block of pixels.

2.6.2 Average Representation

Average representation also uses the $n \times n$ window (see Figure 2.6). Each block of pixels is represented by the average of all pixels values. This is not as slow as median representation.

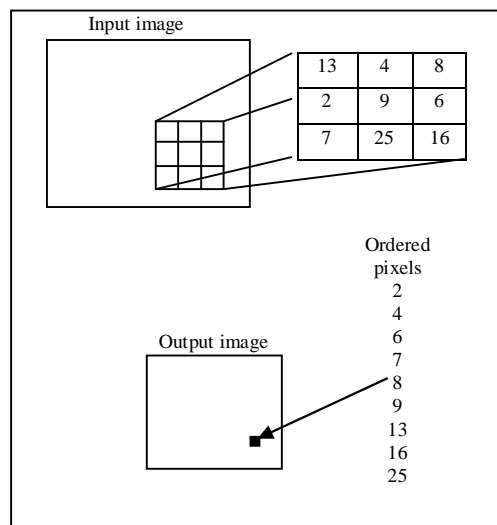


Figure (2.5) Minification by median representation

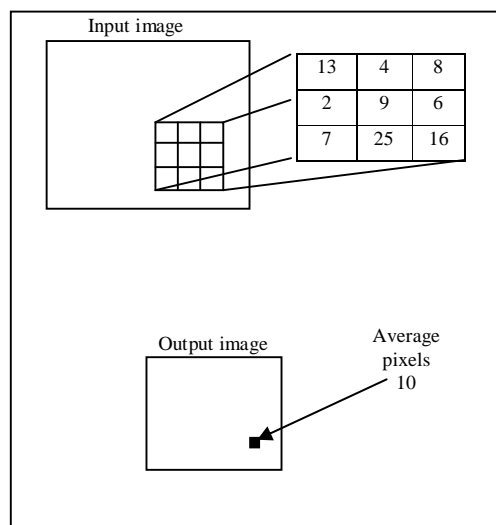


Figure (2.6) Minification by average representation

2.7 Classification

The task of classification is to learn a function that maps data to the available set of classes. A classifier learns from a training set containing a large number of already mapped objects for each class. The training objects are considered to be labeled with the name of the class they belong to. Classification is also called *supervised learning* because it is directed by these labeled objects.

Classification is an extensively researched topic in data mining and machine learning. The main hurdle to leveraging the existing classification methods is that these assume record data with a fixed number of attributes. In general, people dealing with sequence data use to convert the data into non-sequential data and then apply traditional classification algorithm. [Dav08]

2.8 Clustering

Data mining can quite often be defined as a useful hidden knowledge extraction process from a huge database. Basically, two types of techniques, supervised and unsupervised, are using to extract this knowledge. When the problem is not predefined, then the researcher always chooses the unsupervised technique to solve their problems. A good number of unsupervised techniques introduced by researchers and are free for use. K-means algorithm is an old unsupervised technique but still it is a popular technique. The job of unsupervised technique is called clustering. [Dav08]

Clustering, defined broadly, is the grouping of similar objects. More specifically, clustering is the unsupervised classification of *patterns* into groups based upon similarity, where a pattern is a representation of features or observations made on an object. Upon first glance, the problem of clustering is quite similar to that of classification. It should be

noted, however, that the key difference between the two is the *unsupervised* nature of clustering. Traditional supervised classification involves a special input training set or predefined set of classes into which objects are placed, whereas clustering attempts to derive meaningful classes solely from the data. [Mec06]

Clustering is often a critical component of the data mining or knowledge discovery process. Data mining tasks frequently involve large amounts of data, and clustering methods may be employed during the data transformation stage of the knowledge discovery process doing so effectively abstracts or compresses the data and allows the subsequent data mining algorithms to treat each data cluster as a single datum. Clustering may also be utilized to aid the practitioner in visualizing and interpreting data mining results, possibly revealing previously unknown tendencies in the data. [Mec06]

Clustering techniques apply when there is no class to be predicted but rather when the instances are to be divided into natural groups. These clusters presumably reflect some mechanism at work in the domain from which instances are drawn, a mechanism that causes some instances to bear a stronger resemblance to each other than they do to the remaining instances. Clustering naturally requires different techniques to the classification and association learning methods. [Ian05] And one of these techniques is the *K-means*.

2.8.1 K-means Method Technique

The *k*-means algorithm is the best known partitional clustering algorithm. It is perhaps also the most widely used among all clustering algorithms due to its simplicity and efficiency. Given a set of data points and the required number of *k* clusters (*k* is specified by the user), this

algorithm iteratively partitions the data into k clusters based on a distance function. [Bin07]

The aim of the *k-means* (which also goes by the names of the *c-means* or *iterative relocation* or *basic ISODATA*) algorithm is to partition the data into k clusters so that the within-group sum of squares is minimized. The simplest form of the *k-means* algorithm is based on alternating two procedures. The first is one of assignment of objects to groups. An object is usually assigned to the group to whose mean it is closest in the Euclidean sense. The second procedure is the calculation of new group means based on the assignments. The process terminates when no movement of an object to another group will reduce the within-group sum of squares. [And02]

The *k-Means* or ISODATA clustering algorithm is the most popular example of an algorithm that performs iterative adjustment of c (k in the original algorithm version) cluster centroids. It has a distinct application from the tree clustering methods since it is intended to yield a clustering solution for an arbitrarily large number of patterns and for a previously defined number of clusters. The choice of the number of clusters can be made by performing tree clustering either in a smaller set of data or in a reduced dimension space obtained for instance by multidimensional scaling. [Mar01]

K-means Algorithm

Let the set of data points (or instances) D be

$$\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\},$$

Where $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ir})$ is a vector in a real-valued space $X \subseteq \mathbb{R}^r$, and r is the number of attributes in the data (or the number of dimensions of the *data space*). The *k-means* algorithm partitions the given data into k clusters. Each cluster has a cluster **center**, which is also

called the cluster **centroid**. The centroid, usually used to represent the cluster, is simply the mean of all the data points in the cluster, which gives the name to the algorithm, i.e., since there are k clusters, thus k means. Figure gives the *k-means* clustering algorithm. [Bin07]

```

Algorithm k-means( $k, D$ )
1  choose  $k$  data points as the initial centroids (cluster centers)
2  repeat
3    for each data point  $\mathbf{x} \in D$  do
4      compute the distance from  $\mathbf{x}$  to each centroid;
5      assign  $\mathbf{x}$  to the closest centroid      // a centroid represents a cluster
6    endfor
7    re-compute the centroid using the current cluster memberships
8  until the stopping criterion is met

```

Figure 2.7: The *K-means* algorithm

In the first step, the algorithm randomly selects K data points to be the seeds. MacQueen's algorithm simply takes the first K records. In cases where the records have some meaningful order, it may be desirable to choose widely spaced records, or a random selection of records. Each of the seeds is an embryonic cluster with only one element. [Mic04]

It then computes the distance between each seed centroid and every data point. Each data point is assigned to the centroid that is closest to it. A centroid and its data points therefore represent a cluster. Once all the data points in the data are assigned, the centroid for each cluster is re-computed using the data points in the current cluster. This process repeats until a stopping criterion is met. The stopping (or convergence) criterion can be any one of the following:

1. no (or minimum) re-assignments of data points to different clusters.
2. no (or minimum) change of centroids.
3. minimum decrease in the **sum of squared error** (SSE),

$$SSE = \sum_{j=1}^k \sum_{x \in C_j} dist(x, m_j)^2, \dots \dots \dots (2.38)$$

Where k is the number of required clusters, C_j is the j th cluster, m_j is the centroid of cluster C_j (the mean vector of all the data points in C_j), and $dist(x, m_j)$ is the distance between data point x and centroid m_j .

And figure (2.8) show three iterations of the K-means applied on a set of data points in a 2-dimensional space. [Bin07]

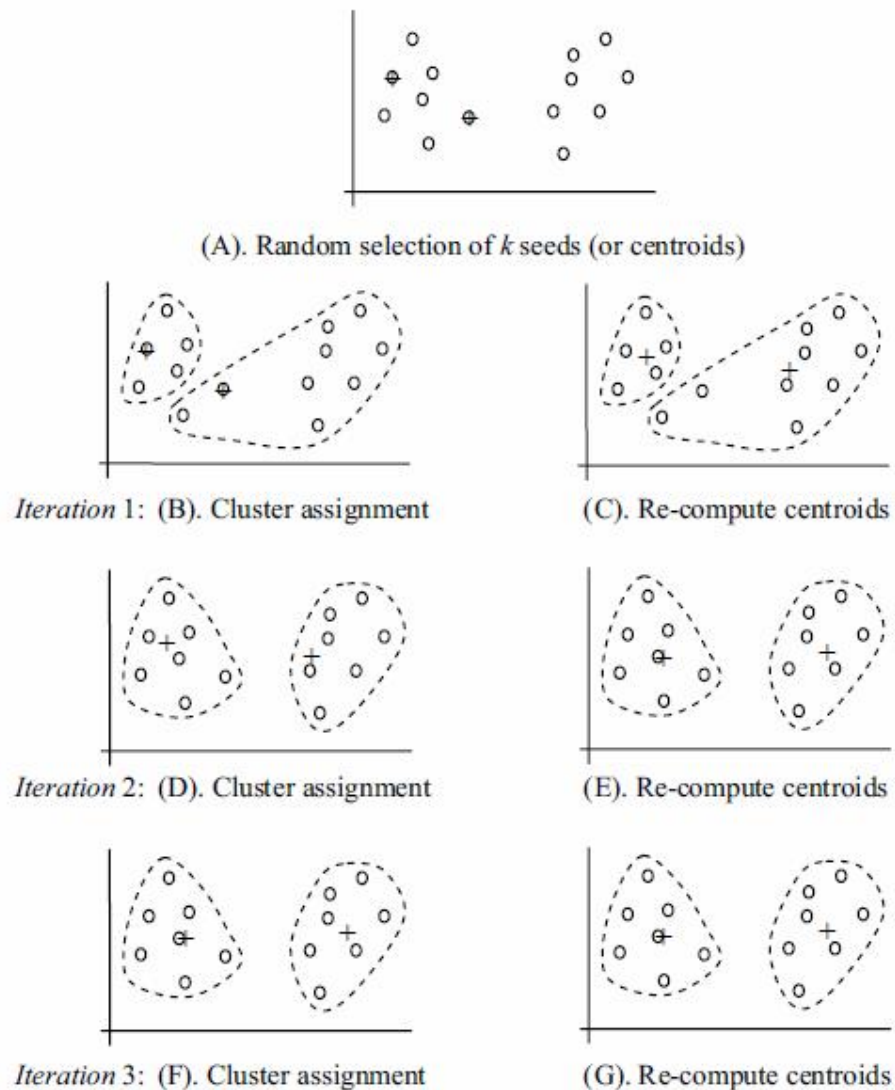


Figure 2.8: The working of the *K-means* algorithm

Chapter Three

Video Retrieval System

Chapter Three

Video Retrieval System

3.1 Introduction

This chapter presents an implementation of the Video Retrieval System (VRS), and a detailed description for each part. This system contains in addition to the main steps for performing system task, a set of steps which helps in reducing calculation time (such as Down-Sampling, and Quantization).

VRS has two phases. These are: video database construction (enrollment phase) and video retrieval (query-matching phase). The main purpose of enrollments phase is to extract features from the input videos (collection of video shots) and store these features into files as a database of different classes which is used later in retrieval operation. The input for the query-matching phase is a video file that contains different shots. After selecting one of these shots it will be separated for extracting the features from it. Then, these features will be compared to the centroids of the classes in the constructed video DB. The files in the nearest class will be shown as results of retrieving operation.

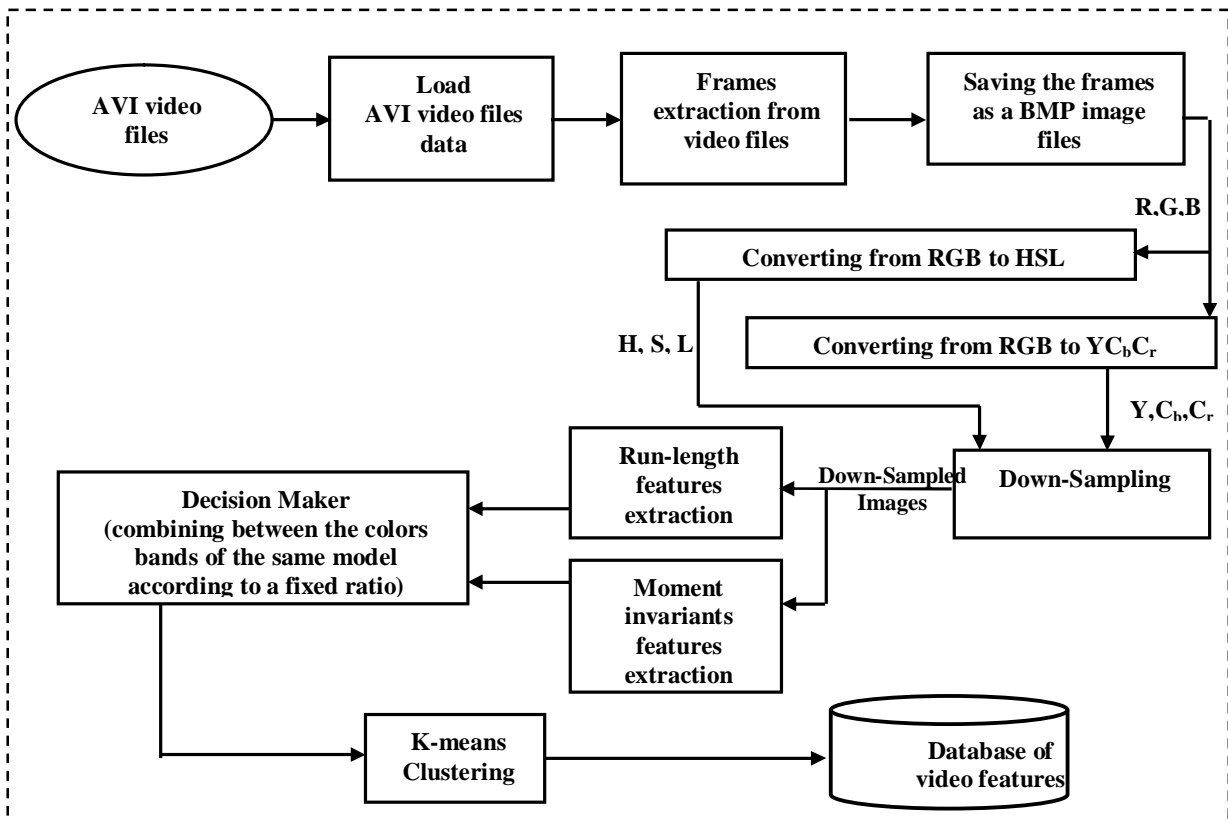
The structure of the established system and the functionality of its modules will be discussed in details in the next sections.

3.2 The VRS Module

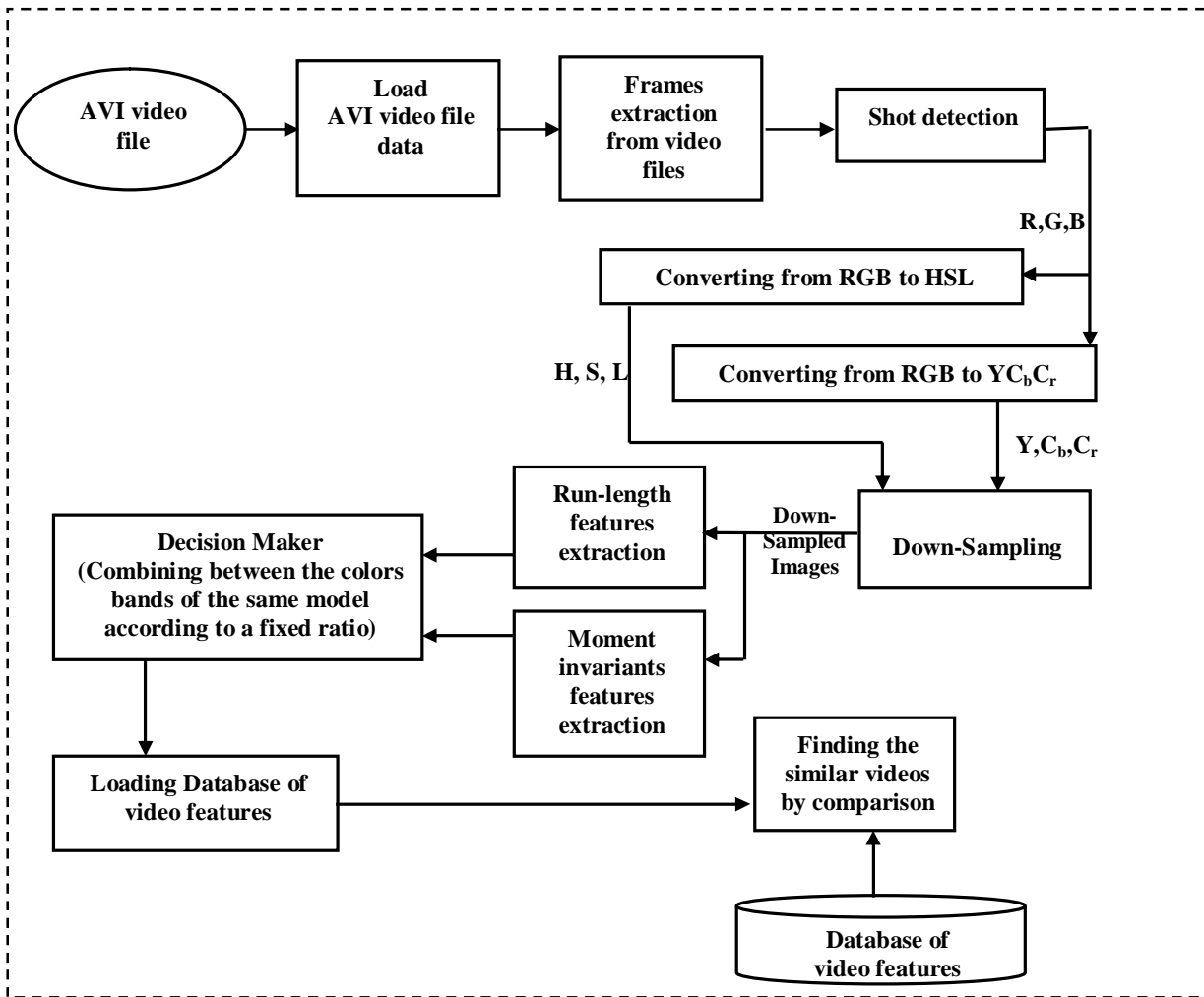
The general structure of the VRS is illustrated in figure (3.1). It consists of two basic modules: video database construction and video retrieval modules. The input to the first module is a collection of

uncompressed AVI (Audio\Video Interleaved) videos' shots. The data of this collection of AVI videos' features is passed through a set of operations to produce a database of videos' features; these features will be used in the second module to retrieve the similar AVI videos of an input uncompressed AVI video shot.

Each of the two modules went through a sequence of related operations that accomplish the system's task.



a. Videos database construction
Figure 3.1: The System Model



b. Video retrieval
Figure 3.1: The System Model

3.3 Video Database Construction module

As shown in figure (3.1a), the video database construction sub modules are: loading the AVI video files, then its output will be directed to frames extraction module, color conversion module, down-sampling module, features extraction module, and as a final step, the K-means module that classify the videos' shots into a set of classes according to the extracted features. Each one will be described in the next sections.

3.3.1 Loading AVI video files and frames extraction Modules

The input to loading module is a collection of AVI video files that had been collected from video different websites and TV networks reports and documentaries.

By using an external library like (*avifil32.dll* - Dynamic Link Library which exists in different versions of Microsoft Windows in *system32* folder that holds all the libraries of the operating system) makes it easier to load the video file and extract frames from it because it had all the functions that deal with the AVI file for example:

- *AVIFileOpen*: opens the AVI file as binary to get the file header and check the file type signature.
- *AVIFileInfo*: gets information about the opened AVI file (like the number of frames, height, width, ...,etc).
- *AVIStreamGetFrame*: gets the frames from the opened AVI file after having the information of the beginning and the end of the frames from another included function.

BMP (Bitmap) image files would be created depending on the information gained from the above mentioned functions. Also the contents of the image files which are consist of the extracted frame and a standard BMP header image file. The Created BMP images saved in a temporary folder created in the application path. All the mentioned operations were illustrated in Algorithm (3.1).

Algorithm (3.1) Convert AVI file**Goal:** Convert AVI file to BMP files in a temporary folder**Input:**

AVI file.

Output:

BMP images. //frames.

noofframes. // number of frames.

width. //width of the frame.

Hgt. //height of the frame.

Step1: Call AVIFileInit // opens AVIFile library "avifil32.dll" which exists in windows

Set Res←AVIFileOpen(pAVIFile, szFile, OF_READ, &0).

Step2: Check If Res is not equal to 0 Then

Output Unable to open this file

Else

Set hr←AVIFileInfo(pAVIFile, fileInfo, 108)

Check If hr is not equal to 0 Then

Output Error Unable to open AVI file info

End If

Set H←fileInfo.dwHeight

Set W←fileInfo.dwWidth

Set Res ← AVIFileGetStream (pAVIFile, pAVIStream, streamtypeVIDEO, 0)

Set firstframe←AVIStreamStart(pAVIStream)

Check If firstframe is equal to -1 Then

Output Error

Else

Set numFrames←AVIStreamLength(pAVIStream)

Check If numFrames is equal to -1 Then

Output Error

Else

Set Totalframes←numFrames

Set pGetFrameObj←AVIStreamGetFrameOpen(pAVIStream, bih) //force function to return 24bit DIBS

Check If pGetFrameObj is equal to 0 Then

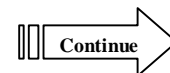
SetGetFrameObj←AVIStreamGetFrameOpen(pAVIStream, ByVal AVIGETFRAMEF_BESTDISPLAYFMT)

Check If pGetFrameObj is equal to 0 Then

Output Error No suitable decompressor found for this video stream

End If

End If



```

Set dib ← New cdib
For all i Do {where firstframe ≤ i ≤ (numFrames - 1) + firstframe}
Set pDIB ← AVIStreamGetFrame(pGetFrameObj, i)
Check If dib.CreateFromPackedDIBPointer(pDIB) Then
On Error Resume Next
Call MakeDirectory App.path & "\temporary"
Call dib.WriteToFile(App.path & "\temporary\" & i &
".bmp")
End If
End For
End If
End If
End If

```

3.3.2 RGB to HSL Conversion

The RGB frames are converted to H, S, and L bands using the equations (2.6), (2.7), and (2.8) respectively. S and L bands are different from H, where H is an angle; therefore an additional step is appended in the RGB to HSL algorithm to control the H value from increasing more than 360° , so that the color will be in the safe range (0° - 360°).

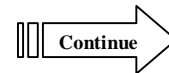
The mapping operation is illustrated in algorithm (3.2):

Algorithm (3.2) Color Transformation RGB to HSL**Goal:** Convert images from RGB to HSL color transformation**Input:***Pics* //Array of Images in RGB color model.**Output:***Ycc* //Array of Images in HSL color model.

```

Step1: For all i Do {where  $FirstImage \leq i \leq LastImage$ }
    For all j Do {where  $0 \leq j \leq Hm$ } // Hm stands for Height -1
        For all k Do {where  $0 \leq k \leq Wm$ } // Wm stands for Width -1
            Set  $R \leftarrow Pics(i).R(k, j)/256$ 
            Set  $G \leftarrow Pics(i).G(k, j)/256$ 
            Set  $B \leftarrow Pics(i).B(K, J)/256$ 
            Check If  $R \geq G$  Then
                Set  $Max \leftarrow R$ 
            Else
                Set  $Max \leftarrow G$ 
            End If
            Check If  $Max < B$  Then
                Set  $Max \leftarrow B$ 
            End If
            Check If  $R \leq G$  Then
                Set  $Min \leftarrow R$ 
            Else
                Set  $Min \leftarrow G$ 
            End If
            Check If  $Min > B$  Then
                Set  $Min \leftarrow B$ 
            End If
            Set  $L \leftarrow (Max + Min)/2$ 
            Check If Max is equal to Min Then
                Initialize H and S by 0 //Achromatic
            Else
                Set  $D \leftarrow Max - Min$ 
                Check If  $L > 0.5$  Then
                    Set  $S \leftarrow D / (2 - Max - Min)$ 
                Else
                    Set  $S \leftarrow D / (Max + Min)$ 
                End If
                Check If Max is equal to R Then
                    Set  $H \leftarrow (G - B) / D$ 
                Check If  $G < B$  Then
                    Increase H by 6

```



```

        End If
    Else If Max is equal to G Then
        Set  $H \leftarrow (B-R)/D+2$ 
    Else If Max is equal to B Then
        Set  $H \leftarrow (R-G)/D+4$ 
    End If
    Set  $H \leftarrow H/6$ 
End If
Check If  $(H*60) > 180$  Then
    Set  $Y_{cc}(i, k, j).Y \leftarrow 360-H*60$ 
Else
    Set  $Y_{cc}(i, k, j).Y \leftarrow H*60$ 
End If
Set  $Y_{cc}(i, k, j).C_r \leftarrow L*256$ 
Set  $Y_{cc}(i, k, j).C_b \leftarrow S*256$ 
End For
End For
End For

```

3.3.3 RGB to $Y C_b C_r$ Conversion

This module is used to convert the RGB frames to $Y C_b C_r$ according to equation (2.5), as shown in Algorithm (3.3):

Algorithm (3.3) Color Transformation RGB to $Y C_b C_r$

Goal: Convert images from RGB to $Y C_b C_r$ color transformation

Input: Pics //Array of Images in RGB color model.

Output: Ycc //Array of Images in $Y C_b C_r$ color model.

Step1: Initialize x by 0

```

    For all i Do {where FirstImage  $\leq i \leq$  LastImage}
        For all j Do {where  $0 \leq j \leq H_m$ }
            For all k Do {where  $0 \leq k \leq W_m$ }
                 $Y_{cc}(x, k, j).Y = 0.2989 * Pics(i).R(k, j) + 0.5866 * Pics(i).G(k, j) + 0.1145 * Pics(i).B(k, j).$ 
                 $Y_{cc}(x, k, j).C_b = -0.168 * Pics(i).R(k, j) - 0.33 * Pics(i).G(k, j) + 0.498 * Pics(i).B(k, j) + 128$ 
                 $Y_{cc}(x, k, j).C_r = 0.498 * Pics(i).R(k, j) - 0.417 * Pics(i).G(k, j) - 0.081 * Pics(i).B(k, j) + 128$ 
            End For
        End For
        Increase x by 1
    End For

```


3.3.4 Down-Sampling

The goal of this operation is to take the average representation as clarified in figure (2.6) and minifying the images.

Algorithm (3.4) Down-Sampling Algorithm

Goal: Applying Down-Sampling algorithm on one band

Input: Level // Down-Sampling Level

The three bands //Y, Cb, and Cr

Wm, Hm //Width and Height of the image

Output:

D_Sample //Y, Cb, and Cr or H, S, and L after applying Down-Sampling algorithm on them

Step1: Set wtemp←Wm

Set htemp←Hm

Step2: For all iL Do {where $1 \leq iL \leq \text{Level}$ }

Set W2←wtemp/2

Set H2←htemp/2

Step3: For all i Do {where $0 \leq i \leq \text{LastImage} - \text{FirstImage}$ }

For all iy Do {where $0 \leq iy \leq H2$ }

Set yy←2*iy

Set yp←yy+1

For all ix Do {where $0 \leq ix \leq W2$ }

Set xx←2*ix

Set Xp←xx+1

Set D_Sample (i, ix, iy).Y←(Ycc(i, xx, yy).Y+Ycc(i, xx, yp).Y+Ycc(i, Xp, yy).Y+Ycc(i, Xp, yp).Y)/4

Set D_Sample (i, ix, iy).Cb←(Ycc(i, xx, yy).Cb+Ycc(i, xx, yp).Cb+Ycc(i, Xp, yy).Cb+Ycc(i, Xp, yp).Cb)/4

Set D_Sample (i, ix, iy).Cr←(Ycc(i, xx, yy).Cr+Ycc(i, xx, yp).Cr+Ycc(i, Xp, yy).Cr+Ycc(i, Xp, yp).Cr)/4

End For

End For

End For

Set wtemp←W2

Set htemp←H2

End For

3.3.5 Run-length features extraction

The goal of this step is to extract texture features from the LL sub-band from the Down-Sampling as mentioned before.

The output is an array of 11 features for each frame in the video, then the mean absolute difference is calculated for all of the frames to the same video to produce a feature vector for each video file.

The Creation of the R array which contains the runs in all of the video frames then directed to the Run Length features calculation algorithm, this creation illustrated in algorithm (3.5). This algorithm written with pseudo code description. For complete details, see algorithm (A.1) appendix A.

Algorithm (3.5) Generate RunLength Matrix

Goal: Generate the R() (Run-Length) array that contains all the runs for a video

Input: QuantGray()// array of the quantized the input color band
Theta//the angle of the runs

Output:
R()//Run-Length array that contains all the runs in an input video

Step1: Set Glevelm ← Glevel – 1

Call Hightm ← H2 – 1

Call Widm ← W2 – 1

Check If Theta is equal to 0 **Then**

Compute all the run lengths from the QuantGry array when Theta angle value equals 0 then store the runs in R array

Else If Theta is equal to 45 **Then**

Compute all the run lengths from the QuantGry array when Theta angle value equals 45 then store the runs in R array and for one of the cases $W2 > H2$ or $H2 > W2$

Else If Theta is equal to 90 **Then**

Compute all the run lengths from the QuantGry array when Theta angle value equals 90 then store the runs in R array

Else If Theta is equal to 135 **Then**

Compute all the run lengths from the QuantGry array when Theta angle value equals 45 then store the runs in R array and for one of the cases $W2 > H2$ or $H2 > W2$

The equations that compute the Run Length features are (2.18) to (2.28) will be illustrated in algorithm (3.6). For complete details, see algorithm (A.2) in Appendix A.

Algorithm (3.6) Compute Run Length Features

Goal: Calculate the Run-Length features

Input: $R()$ //Run-Length array that contains all of the runs for the input video
 Theta The angle of the runs

Output:
 $RLfeats()$ //the array of Run-Length features

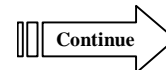
Step1: Set $R_{higt} \leftarrow G_{level} - 1$
 Check If Theta is equal to 0 Then
 Set $R_{wid} \leftarrow W_2$
 Else If Theta is equal to 45 Then
 Check If $W_2 > H_2$ Then
 Set $R_{wid} \leftarrow H_2$
 Else
 Set $R_{wid} \leftarrow W_2$
 End If
 Else If Theta is equal to 90 Then
 Set $R_{wid} \leftarrow H_2$
 Else If Theta is equal to 135 Then
 Check If $W_2 > H_2$ Then
 Set $R_{wid} \leftarrow H_2$
 Else
 Set $R_{wid} \leftarrow W_2$
 End If
 End If

Step2: Initialize Sum to 0
 For all i Do {Where $0 \leq i \leq R_{higt}$ }
 For all J Do {Where $1 \leq J \leq R_{wid}$ }
 Set $Sum \leftarrow Sum + R(i, J)$
 End For
 End For

Step3: Initialize Sum1 to 0
 Compute the Short Run Emphasis feature
 Set $RLFeats.SRE \leftarrow Sum1 / Sum$

Step4: Initialize Sum1 to 0
 Compute The Long Run Emphasis feature
 Set $RLFeats.LRE \leftarrow Sum1 / Sum$

Step5: Initialize Sum1 to 0
 Compute the Gray Level Non-Uniformity feature
 Set $RLFeats.GLNU \leftarrow Sum1 / Sum$



Step6: Initialize Sum1 to 0
Compute the Run Length Non-Uniformity Feature
Set RLFeats.RLNU \leftarrow Sum1/Sum

Step7: Compute the Run Percentage feature
Set RLFeats.RP \leftarrow Sum/Rwid*Rhigt

Step8: Initialize Sum1 to 0
Compute the Low Gray Level Run Emphasis feature
Set RLFeats.LGRE \leftarrow Sum1/Sum

Step9: Initialize Sum1 to 0
Compute High Gray Level Run Emphasis feature
Set RLFeats.HGRE \leftarrow Sum1/Sum

Step10: Initialize Sum1 \leftarrow 0
Compute Short Run Low Gray Level Emphasis feature
Set RLFeats.SRLGE \leftarrow Sum1/Sum

Step11: Initialize Sum1 \leftarrow 0
Compute Short Run High Gray Level Emphasis feature
Set RLFeats.SRHGE \leftarrow Sum1/Sum

Step12: Initialize Sum1 to 0
Compute Low Run Low Gray Level Emphasis feature
Set RLFeats.LRLGE \leftarrow Sum1/Sum

Step13: Initialize Sum1 to 0
Compute Low Run High Gray Level Emphasis feature
Set RLFeats.LRHGE \leftarrow Sum1/Sum

After computing the Down-Sampled image, the quantization color band array (*Temp_Mean_Runlength*) should be computed for D_Sample, as shown in Algorithm (3.7). For complete details, see algorithm (A.3)

Algorithm (3.7) Quanti

Goal: quantizing the input D_Sample array from Down_Sampling algorithm

Input $D_Sample()$ //array of that is resulted from the Down_Sampling algorithm

$LastImage$ //the last image in the input video shot

$FirstImage$ //the first image in the input video shot

$RLFeats$ //the array of the run-length features

Output:

Fll //text file that contains the output results

$Temp_Mean_Runlength()$ //array contains the output quantized features

Step1: Set $Glevel \leftarrow 64$

For all i **Do** {Where $0 \leq i \leq LastImage - FirstImage$ }

 Set $Fll \leftarrow i \& "LL_y"$

For all K **Do** {Where $0 \leq K \leq W2$ }

For all J **Do** {Where $0 \leq J \leq H2$ }

 Set $QuantGry(K, J) \leftarrow (D_Sample(i, K, J).y + 256) * 64 \text{ Div } 512$

End For

End For

 Call Run_Length

 Write $RLFeats$ array to File of name Fll

 Compute the $Temp_Mean_Runlength$ array for band Y from array

$RLFeats$ that resulted from Run_Length

 Set $Fll \leftarrow i \& "LL_cb"$

For all K **Do** {Where $0 \leq K \leq W2$ }

For all J **Do** {Where $0 \leq J \leq H2$ }

 Set $QuantGry(K, J) \leftarrow (D_Sample(i, K, J).C_b + 256) * 64 \text{ Div } 512$

End For

End For

 Call Run_Length

 Write $RLFeats$ array to File of name Fll

 Compute the $Temp_Mean_Runlength$ array for band C_b from array

$RLFeats$ that resulted from Run_Length

 Set $Fll \leftarrow i \& "LL_cr"$

For all K **Do** {Where $0 \leq K \leq W2$ }

For all J **Do** {Where $0 \leq J \leq H2$ }

 Set $QuantGry(K, J) \leftarrow (D_Sample(i, K, J).C_r + 255) * 64 \text{ Div } 512$

End For

End For

 Call Run_Length

 Write $RLFeats$ array to File of name Fll

 Compute the $Temp_Mean_Runlength$ array for band C_r from array

$RLFeats$ that resulted from Run_Length

End For

Step2: Set $i \leftarrow LastImage - FirstImage$

 Divide $Temp_Mean_Runlength$ by i to obtain the mean value to the three bands Y , C_b , and C_r

3.3.6 Moment invariants features extraction

The goal of this step is to extract Moment Invariants features; this will be accomplished using equations (2.11) to (2.17) sequentially which are implemented in algorithm (3.8).

Algorithm (3.8) Moment Invariants

Goal: extracting Moment invariants features

Input: *D_Sample()*//array of that is resulted from the Down-Sampling algorithm

LastImage//last image of the input video shot

FirstImage//first image of the input video shot

Output:

Mmnt_1()//array of extracted images' first moment invariants

Mmnt_2()//array of extracted images' second moment invariants

Mmnt_3()//array of extracted images' third moment invariants

Mmnt_4()//array of extracted images' forth moment invariants

Mmnt_5()//array of extracted images' fifth moment invariants

Mmnt_6()//array of extracted images' sixth moment invariants

Mmnt_7()//array of extracted images' seventh moment invariants

Temp_Mean_Moment()//array that contains the Moment invariants features

Step1: Initialize *yy* to 1

For all *i* Do {Where $0 \leq i \leq \text{LastImage} - \text{FirstImage}$ }

Call *mew* 0, 0, *i*, *s2*

Step1: Call *mew* 3, 0, *i*, *s1*//Computes the central Moment for $p=3, q=0$

Set $n30(0).y \leftarrow s1(0).y/s2(0).y^{yy}$

Set $n30(0).Cb \leftarrow s1(0).Cb/s2(0).Cb^{yy}$

Set $n30(0).Cr \leftarrow s1(0).Cr/s2(0).Cr^{yy}$

Repeat Step1 for the functions *mew* (0,3), *mew* (2,1), *mew* (1,2), *mew* (1,1), *mew* (0,2), and *mew*(2,0) to produce *n03*, *n21*, *n12*, *n11*, *n02*, and *n20*

Step2: Set $Mmnt_1(i, 0).y \leftarrow n20(0).y + n02(0).y$

Set $TT_y \leftarrow n20(0).y - n02(0).y$

Set $Mmnt_2(i, 0).y \leftarrow TT_y * TT_y + 4 * n11(0).y * n11(0).y$

Set $TT_y1 \leftarrow n30(0).y - 3 * n12(0).y$

Set $TT_y2 \leftarrow 3 * n21(0).y - n03(0).y$

Set $Mmnt_3(i, 0).y \leftarrow TT_y1 * TT_y1 + TT_y2 * TT_y2$

Set $TT_y4 \leftarrow n30(0).y + n12(0).y$

Set $TT_y5 \leftarrow n21(0).y + n03(0).y$

Set $Mmnt_4(i, 0).y \leftarrow TT_y4 * TT_y4 + TT_y5 * TT_y5$

Set $TT_y6 \leftarrow TT_y4 * TT_y4 - 3 * TT_y5 * TT_y5$

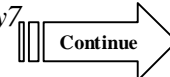
Set $TT_y7 \leftarrow TT_y2 * TT_y5 * 3 * TT_y4 * TT_y4 - TT_y5 * TT_y5$

Set $Mmnt_5(i, 0).y \leftarrow TT_y1 * TT_y4 * TT_y6 + TT_y7$

Set $Mmnt_6(i, 0).y \leftarrow TT_y * TT_y4 * TT_y4 - TT_y5$

$* TT_y5 + 4 * n11(0).y * TT_y4 * TT_y5$

Set $Mmnt_7(i, 0).y \leftarrow TT_y2 * TT_y4 * TT_y6 + TT_y7$



Repeat Step2 for the color sub-bands C_b and C_r instead of the color band Y

Step3: Write $Mmnt_1(i, 0).y$, $Mmnt_2(i, 0).y$, $Mmnt_3(i, 0).y$, $Mmnt_4(i, 0).y$, $Mmnt_5(i, 0).y$, $Mmnt_6(i, 0).y$, and $Mmnt_7(i, 0).y$ **to File**

Repeat Step3 for the color sub-bands C_b and C_r instead of the color band Y

Step4: Set $Temp_Mean_Moment(Count_i).Mmnt_1.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_1.y + Mmnt_1(i, 0).y$

Set $Temp_Mean_Moment(Count_i).Mmnt_2.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_2.y + Mmnt_2(i, 0).y$

Set $Temp_Mean_Moment(Count_i).Mmnt_3.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_3.y + Mmnt_3(i, 0).y$

Set $Temp_Mean_Moment(Count_i).Mmnt_4.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_4.y + Mmnt_4(i, 0).y$

Set $Temp_Mean_Moment(Count_i).Mmnt_5.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_5.y + Mmnt_5(i, 0).y$

Set $Temp_Mean_Moment(Count_i).Mmnt_6.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_6.y + Mmnt_6(i, 0).y$

Set $Temp_Mean_Moment(Count_i).Mmnt_7.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_7.y + Mmnt_7(i, 0).y$

Repeat Step4 for the color sub-bands C_b and C_r instead of the color band Y

End For

Set $i \leftarrow LastImage - FirstImage$

Step5: Set $Temp_Mean_Moment(Count_i).Mmnt_1.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_1.y / i$

Set $Temp_Mean_Moment(Count_i).Mmnt_2.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_2.y / i$

Set $Temp_Mean_Moment(Count_i).Mmnt_3.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_3.y / i$

Set $Temp_Mean_Moment(Count_i).Mmnt_4.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_4.y / i$

Set $Temp_Mean_Moment(Count_i).Mmnt_5.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_5.y / i$

Set $Temp_Mean_Moment(Count_i).Mmnt_6.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_6.y / i$

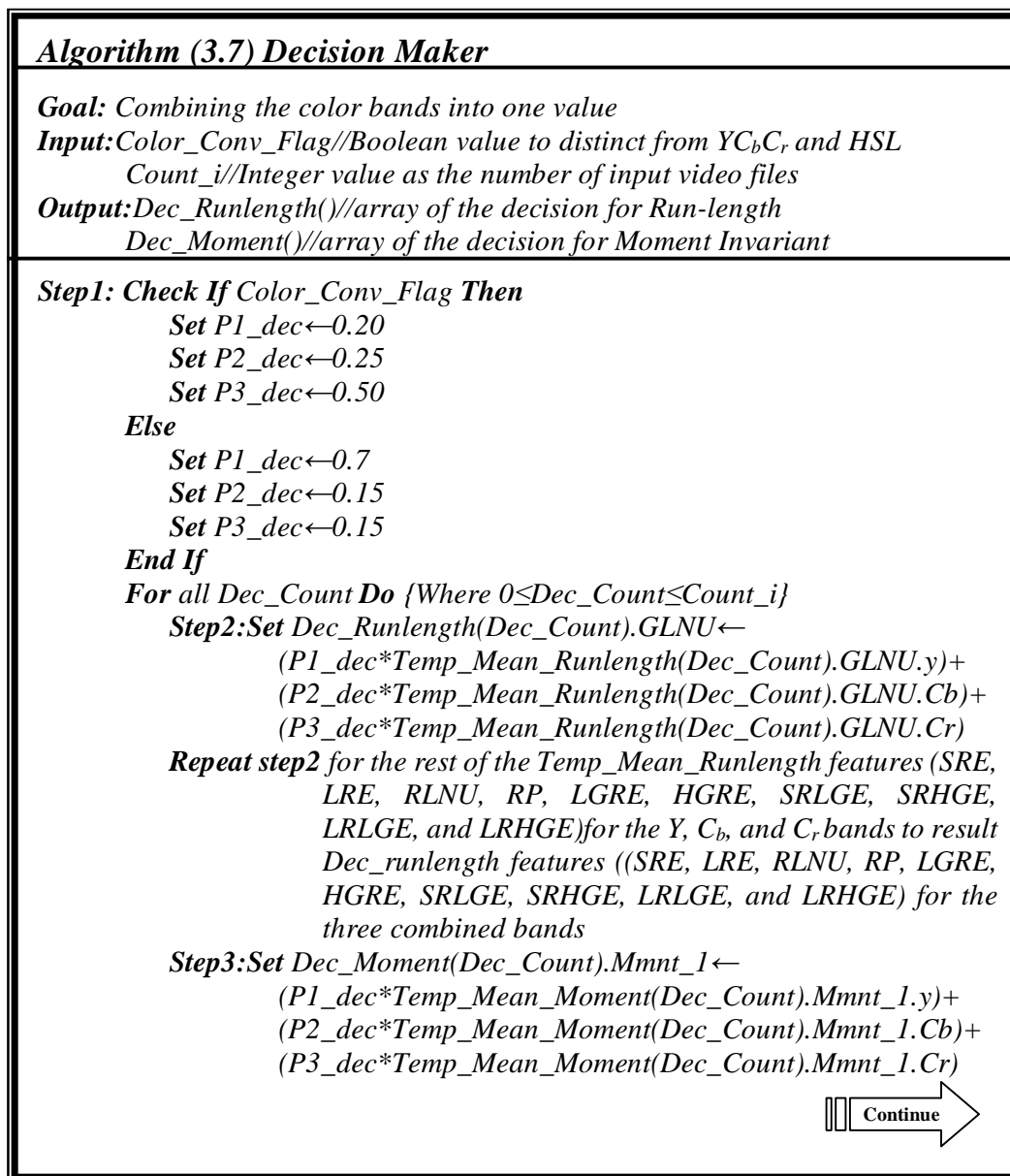
Set $Temp_Mean_Moment(Count_i).Mmnt_7.y \leftarrow Temp_Mean_Moment(Count_i).Mmnt_7.y / i$

Repeat Step5 for the color sub-bands C_b and C_r instead of the color band Y

3.3.7 Decision Maker

The goal from this step is to combine the resulted color bands of the features from Run-length and Moment invariant algorithms according to some tested and standard ratios that will make it easier to cluster the videos into specified classes.

The standard ratios for Y, C_b, and C_r are 0.70: 0.15: 0.15 respectively and the tested ratios were used for H, S, and L is 0.20: 0.25: 0.55 respectively. The Decision Maker module is illustrated in algorithm (3.7).

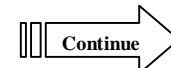


Repeat step3 for the rest of Temp_Mean_Moment features (Mmnt_2, Mmnt_3, Mmnt_4, Mmnt_5, Mmnt_6, and Mmnt_7) for the Y, C_b, and C_r bands to result Dec_moment features (Mmnt_2, Mmnt_3, Mmnt_4, Mmnt_5, Mmnt_6, and Mmnt_7) for the three combined bands

End For

3.3.8 K-means clustering

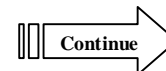
The aim of using K-means method is to cluster the input videos to build the videos shots DB according to the output values of the Decision Maker algorithm, and according to the values of Run-length and Moment invariant features in two separate algorithms (3.8) and (3.9) respectively.

Algorithm (3.8) K-means Runlength**Goal:** Clustering the input videos according to Run-length features**Input:***NoOfClasses* //Number of Classes*Count_i* //integer value refer to the number of input videos*Dec_Runlength()*//array of the combined values resulted from Decision Maker algorithm**Output:** *Count_Temp_Centroid_R()* //array of the number of files in each class*Centroid_R()* //array of the classes centroids*Classes_R()* //array of the belonging of files to which class**Step1: Check If** $NoOfClasses \geq Count_i$ **Then****Output Error****Else****For all** *Counter_Kmeans_R* **Do** {Where $0 \leq Counter_Kmeans_R \leq NoOfClasses$ }**Set** *Centroid_R(Counter_Kmeans_R)* ← *Dec_Runlength(Counter_Kmeans_R)***End For****Initialize** *Dif_Check_R* to 1**Do Until** *Dif_Check_R* is equal to 0**Initialize** *Dif_Check_R* to 0**For all** *Counter_Kmeans_R* **Do** {Where $0 \leq Counter_Kmeans_R \leq NoOfClasses$ }**Set** *Old_Centroid_R(Counter_Kmeans_R)* ← *Centroid_R(Counter_Kmeans_R)***End For****For all** *Counter_Kmeans_R* **Do** {Where $0 \leq Counter_Kmeans_R \leq Count_i$ }**For all** *Temp_Count_R* **Do** {Where $0 \leq Temp_Count_R \leq NoOfClasses$ }**Step2: Set** *Sum_Dif_R(Temp_Count_R)* ← *Sum_Dif_R(Temp_Count_R) + |Centroid_R(Temp_Count_R).GLNU - Dec_Runlength(Counter_Kmeans_R).GLNU|***Repeat Step2** for the rest of the *Centroid_R* and *Dec_Runlength* values (*SRE, LRE, RLNU, RP, LGRE, HGRE, SRLGE, SRHGE, LRLGE, and LRHGE*)**Set** *Sum_Dif_R(Temp_Count_R)* ← *Sum_Dif_R(Temp_Count_R) / 11***End For****Set** *Min_Dif_R* ← *Sum_Dif_R(0)***Set** *Min_Dif_Flag_R* ← *False***For all** *Temp_Count_R* **Do** {Where $0 \leq Temp_Count_R \leq NoOfClasses$ }**Check If** *Sum_Dif_R(Temp_Count_R) ≤ Min_Dif_R* **Then**

```

    Set Min_Dif_R ← Sum_Dif_R(Temp_Count_R)
    Increase Count_Temp_Centroid_R(Temp_Count_R) by 1
    Step3: Set Temp_Centroid_R(Temp_Count_R).GLNU ←
        Temp_Centroid_R(Temp_Count_R).GLNU +
        Dec_Runlength(Counter_Kmeans_R).GLNU
    Repeat Step3 for the rest of the Temp_Centroid_R and
        Dec_Runlength values (SRE, LRE, RLNU,
        RP, LGRE, HGRE, SRLGE, SRHGE,
        LRLGE, and LRHGE)
    Set Classes_R(Counter_Kmeans_R) ← Temp_Count_R
    Set Min_Dif_Flag_R ← True
    End If
    End For
    End For
    For all Temp_Count_R Do {Where  $0 \leq \text{Temp\_Count\_R} \leq$ 
        NoOfClasses}
        Step4: Set Centroid_R(Temp_Count_R).GLNU ←
            Temp_Centroid_R(Temp_Count_R).GLNU /
            Count_Temp_Centroid_R(Temp_Count_R)
        Repeat Step4 for the rest of the Temp_Centroid_R and
            Centroid_R values (SRE, LRE, RLNU, RP,
            LGRE, HGRE, SRLGE, SRHGE, LRLGE, and
            LRHGE)
    End For
    For all Counter_Kmeans_R Do {Where  $0 \leq \text{Counter\_Kmeans\_R} \leq$ 
        NoOfClasses}
        Step5: Set Dif_Check_R ← Dif_Check_R +
            |Old_Centroid_R(Counter_Kmeans_R).GLNU -
            Centroid_R(Counter_Kmeans_R).GLNU|
        Repeat Step5 for the rest of the Centroid_R and
            Old_Centroid_R values (SRE, LRE, RLNU, RP,
            LGRE, HGRE, SRLGE, SRHGE, LRLGE, and
            LRHGE)
        Set Dif_Check_R ← Dif_Check_R / 11
    End For
    End Do
    Write NoOfClasses, Count_i, Centroid_R, Classes_R, and
        Count_Temp_Centroid_R to File
    End If
    For all Temp_Count_R Do {Where  $0 \leq \text{Temp\_Count\_R} \leq \text{Count\_i}$ }
        Write Classes_R(Temp_Count_R), and
            File_Db(Temp_Count_R).filename to File
    End For

```

Algorithm (3.9) K-means Moment**Goal:** Clustering the input videos according to Moment Invariant features**Input:***NoOfClasses* //Number of Classes*Count_i* //integer value refer to the number of input videos*Dec_Runlength()*//array of the combined values resulted from Decision Maker algorithm**Output:** *Count_Temp_Centroid_M* //array of the number of files in each class*Centroid_M* //array of the classes centroids*Classes_M()* //array of the belonging of files to which class**Step1: Check If** $NoOfClasses \geq Count_i$ **Then****Output Error****Else****For all** *Counter_Kmeans* **Do** {Where $0 \leq Counter_Kmeans \leq NoOfClasses$ }**Set** *Centroid(Counter_Kmeans)* ← *Dec_Moment(Counter_Kmeans)***End For****Initialize** *Dif_Check* to 1**Do Until** *Dif_Check* equal to 0**Initialize** *Dif_Check* to 0**For all** *Counter_Kmeans* **Do** {Where $0 \leq Counter_Kmeans \leq NoOfClasses$ }**SetOld_Centroid(Counter_Kmeans)** ← *Centroid(Counter_Kmeans)***End For****For all** *Counter_Kmeans* **Do** {Where $0 \leq Counter_Kmeans \leq Count_i$ }**For all** *Temp_Count* **Do** {Where $0 \leq Temp_Count \leq NoOfClasses$ }**Set** *Min_Dif_Flag* ← False**Step2: Set** *Sum_Dif(Temp_Count)* ←*Sum_Dif(Temp_Count) + |Centroid(Temp_Count).**Mmnt_1 - Dec_Moment(Counter_Kmeans). Mmnt_1/***Repeat Step2** for the rest of the *Centroid* and *Dec_Moment* values (*Mmnt_2*, *Mmnt_3*, *Mmnt_4*, *Mmnt_5*, *Mmnt_6*, and *Mmnt_7*)**End For****Set** *Min_Dif* ← *Sum_Dif(0)***For all** *Temp_Count* **Do** {Where $1 \leq Temp_Count \leq NoOfClasses$ }**Check If** *Sum_Dif(Temp_Count)* < *Min_Dif* **Then****Set** *Min_Dif* ← *Sum_Dif(Temp_Count)***Increase** *Count_Temp_Centroid(Temp_Count)* by 1**Step3: Set** *Temp_Centroid(Temp_Count).Mmnt_1* ←*Temp_Centroid(Temp_Count).Mmnt_1 +**Dec_Moment(Counter_Kmeans).Mmnt_1***Repeat Step3** for the rest of the *Temp_Centroid* and *Dec_Moment* values (*Mmnt_2*, *Mmnt_3*, *Mmnt_4*, *Mmnt_5*, *Mmnt_6*, and *Mmnt_7*)

```

        Set Min_Dif_Flag ← True
    End If
End For
Check If Not (Min_Dif_Flag) Then
    Initialize Classes(Counter_Kmeans) to 0
    Increase Count_Temp_Centroid(0) by 1
    Step4: Set Temp_Centroid(0).Mmnt_1 ←
        Temp_Centroid(0).Mmnt_1 +
        Dec_Moment(Counter_Kmeans).Mmnt_1
    Repeat Step4 for the rest of the Temp_Centroid and
        Dec_Moment values (Mmnt_2, Mmnt_3,
        Mmnt_4, Mmnt_5, Mmnt_6, and Mmnt_7)
    End If
End For
For all Temp_Count Do {Where 0 ≤ Temp_Count ≤ NoOfClasses}
    Step5: Set Centroid(Temp_Count).Mmnt_1 ←
        Temp_Centroid(Temp_Count).Mmnt_1 /
        Count_Temp_Centroid(Temp_Count)
    Repeat Step5 for the rest of the Centroid and
        Count_Temp_Centroid values (Mmnt_2,
        Mmnt_3, Mmnt_4, Mmnt_5, Mmnt_6, and
        Mmnt_7)

    End For
For all Counter_Kmeans Do {Where 0 ≤ Counter_Kmeans ≤
    NoOfClasses}
    Step6: Set Dif_Check ← Dif_Check +
        |Old_Centroid(Counter_Kmeans).Mmnt_1 -
        Centroid(Counter_Kmeans).Mmnt_1|
    Repeat Step6 for the rest of the Old_Centroid and Centroid
        values (Mmnt_2, Mmnt_3, Mmnt_4, Mmnt_5,
        Mmnt_6, and Mmnt_7)

    End For
End Do
Write NoOfClasses, Count_i, Centroid, Classes, and
    Count_Temp_Centroid to File
For all Temp_Count Do {Where 0 ≤ Temp_Count ≤ Count_i}
    Write Classes(Temp_Count), File_Db(Temp_Count).filename to
        File
    End For
End For
End If

```

The final step will be saving the output of the K-means algorithm into a Binary file for Run-length and another one for Moment invariant so that to be used in the retrieval phase.

3.4 Video Retrieval

In this phase, Many steps are common with the offline phase, but it is applied on a single input video shot and these common steps are: AVI loading, frames extraction, saving frames as BMP images, RGB to $YCbCr$ or HSL color conversions, Down-Sampling, Features extraction (Run-length or Moment Invariant), and Decision Maker.

The other steps of the retrieval phase are (Shot Detection, and Video Recognition).

3.4.1 Shot Detection

The goal of this step is to detect the boundaries of the shots according to the difference between the histograms of the sequential images in the same video. The difference is calculated according to equation (2.1).

The calculation of one image histogram is illustrated in algorithm (3.10).

Algorithm (3.10) Histogram
Goal: Calculate the histogram of each picture
Input: <i>Pics()</i> //array of R,G, and B bands for the input image
Output: <i>G()</i> //array of the histogram of the input image
<pre> Step1: For all i2 Do {Where $0 \leq i2 \leq Totalframes$} For all y2 Do {Where $0 \leq y2 \leq Hm$} For all x2 Do {Where $0 \leq x2 \leq Wm$} Set tempr ← (<i>Pics</i>(i2).R(x2,y2)+<i>Pics</i>(i2).G(x2,y2)+ <i>Pics</i>(i2).B(x2,y2)) Div 3 Increase G(i2, tempr) by 1 End For End For End For </pre>

After applying algorithm (3.10) on all of the videos' images, Frame Subtraction algorithm is applied on the calculated histogram to find the shot boundaries. Frame Subtraction is illustrated in algorithm (3.11).

Algorithm (3.11) Frames Subtraction**Goal:** find the Cut boundaries (scene change) of the input video**Input:***G()*//the histogram array**Output:***Scene()*//array of the indexes of the boundaries**Step1:** For all $x3$ Do {Where $1 \leq x3 \leq \text{Totalframes}$ } For all $y3$ Do {Where $0 \leq y3 \leq 255$ } Set $\text{temp}(x3) \leftarrow (\text{temp}(x3) + (G(x3 - 1, y3) - G(x3, y3)) * (G(x3 - 1, y3) - G(x3, y3)))$

End For

 Set $\text{meang} \leftarrow \text{meang} + \text{temp}(x3)$ Write $\text{temp}(x3)$ to File

End For

Step2: Set $\text{meang} \leftarrow \text{meang} \text{ Div } (\text{Totalframes} - 1)$ Initialize $\text{Scene}(0)$ to 0 Initialize Scene_in to 0**Step3:** For all $x3$ Do {Where $1 \leq x3 \leq \text{Totalframes}$ } Check If $\text{temp}(x3) \geq \text{meang} * 30$ Then Increase Scene_in by 1 Write Scene_in to File Set $\text{Scene}(\text{Scene_in}) \leftarrow x3$ Write $\text{Scene}(\text{Scene_in})$ to File

Else

 Write $x3$ to File

End If

End For

3.4.2 Video Recognition

The closest class found by subtracting the features vector from the centroids that represent the classes of the videos' shots and find the closest class to the input video then retrieving the videos' shots of this class after sorting them in descending order according to their closeness to the input one, then viewing the matched videos.

The Insertion Sort was used as a sorting algorithm for sorting videos' shots in descendent order according to the difference between the input file and classes' videos shots as illustrated in algorithm (3.12).

Algorithm (3.12) Insertion Sort

Goal: Applying insertion sort to the videos of the same class according to the input video

Input:

Ret_Array_Size//the size of the unsorted input array

Output:

Unsorted_Ret()//Array that contains sorted elements

Step1: Check If $Ret_Array_Size \geq 2$ Then

For all $Count_Ret_i$ Do {Where $1 \leq Count_Ret_i \leq Ret_Array_Size$ }

Set *Sort_Flag* ← True

Set *Count_Ret_j* ← *Count_Ret_i*

Do Until (*Count_Ret_j* < 1) And *Sort_Flag* is False

Check If *Unsorted_Ret*(*Count_Ret_j*).Diff >

Unsorted_Ret(*Count_Ret_j* - 1).Diff **Then**

Set *Temp_Sort* ← *Unsorted_Ret*(*Count_Ret_j*)

Set *Unsorted_Ret*(*Count_Ret_j*) ←

Unsorted_Ret(*Count_Ret_j* - 1)

Set *Unsorted_Ret*(*Count_Ret_j* - 1) ← *Temp_Sort*

Decrease *Count_Ret_j* by 1

Else

Set *Sort_Flag* ← False

End If

End Do

End For

End If



Chapter Four

Tests and Results

Chapter Four

Tests and Results

4.1 Introduction

This chapter is devoted to present and discuss the results of the conducted tests to study the retrieval performance of the suggested VMRS. This chapter illustrates the results of two sets of results. The first set represents the retrieval tests that have been done on features extracted from two different color components, and run length matrices. In the second set, the retrieval tests have been done on the same two different color components, but with shape features (Moments Invariant). Two accuracy measures (i.e., precision and recall) have been used to assess the performance of the VRS.

4.2 Video Test Material

AVI videos were used for constructing the video database, these files were collected from two resources: Internet web sites (Youtube, Metacafe, and Kooora), and scientific TV channels reports (Discovery, and BBC). The database contains 7 classes with 7 to 10 video shots for each class. The videos were of size 320×240 , with different fps (frame per second) to each shot (15, 24 or 25 fps), different duration for each shot (3 to 10 seconds), and with bit depth 24 bpp (bit per pixel).

4.3 Test Strategy

Several tests were applied to evaluate VMRS performance. Two types of tests were adopted. They are Soft testing (where the tested AVI file from the database) and Hard testing (where the tested AVI file is completely different from the stored ones).

The test operation consists of two steps. The first one is to choose a shot from the set of shots which will be extracted from the input AVI file.

The chosen shot will be an input to the next step to receive the similar videos' shots of the saved video shots DB.

After that, the performance measured will be computed (Precision and Recall) to find the results' accuracy.

4.3.1 Testing the Shot Detection

This step dedicated to show the test results of the query video which consists of more at least two shots so that the detector will reveal obvious results. The query Desert video is shown in figure 4.1, which contains 6 video shots with different number of frames for each shot, and the whole number of the frames was 573 frames. The shot detection step took 22.390 sec to complete the whole desert video and produce results.



Figure 4.1: Shot detection result for a Desert video

Figure (4.2) shows the MAD between the histograms of successive frames for the same Desert query video in figure (4.1).

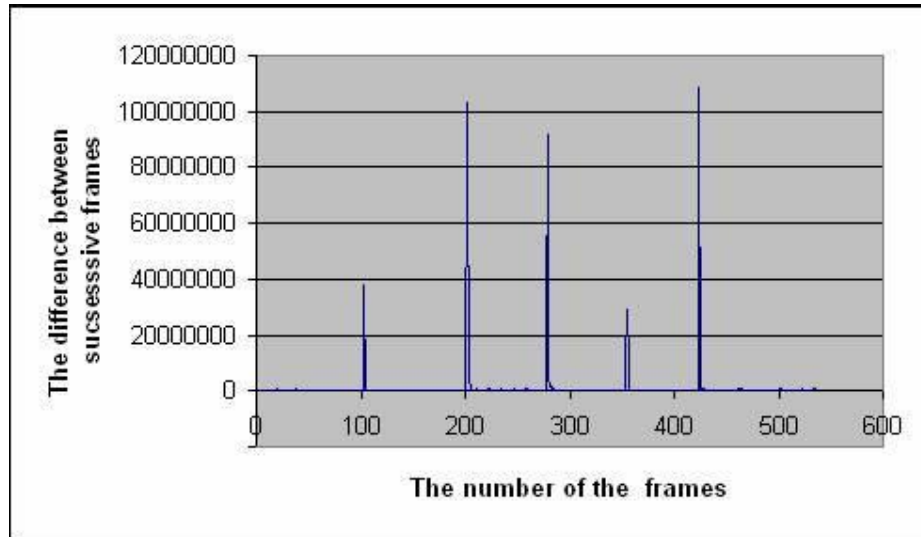


Figure 4.2: MAD value for detecting the video shots

Figure (4.1) shows the first picture of each shot of the query video, and any chosen one of these frames will be an input to classification step, so that depending on this picture, the similar video shots will be viewed.

The peaks in the charts of histograms differences refer to the cut change that is found in the video stream, where shots are detected from it.

Another two different query video examples was tested to prove the system accuracy in detecting the changes of the shots. The first one is for a football video with length of 222 frames that took (13.562) sec processing time, and the results are shown in Figure (4.3).



Figure 4.3: Shot detection applied on a football video

The difference between the histograms of the successive frames of the football video example will be illustrated in figure (4.5).

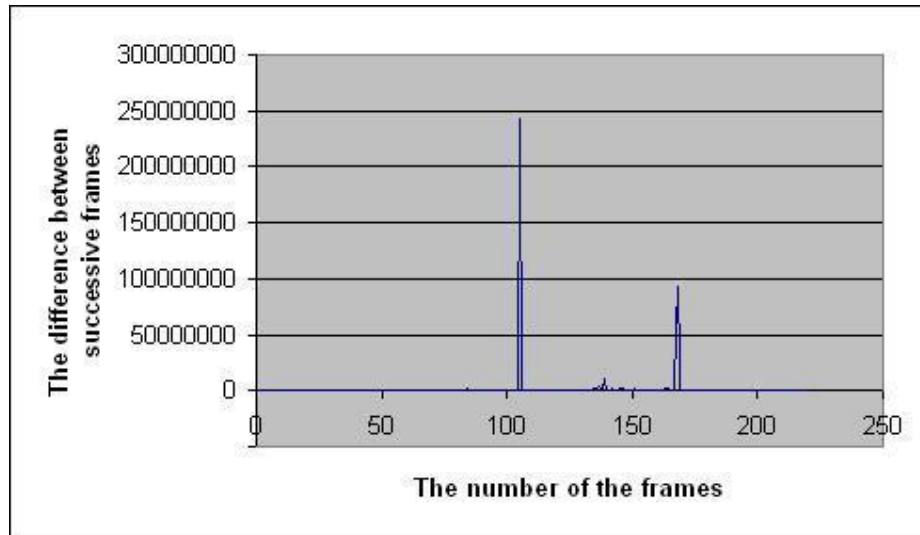


Figure 4.5: The results of the shot detection after applied on the football video sample

The second video example that shows the mountains from different places is the second test video which contains 634 frames and took 23.56 sec and shown in figure (4.6):

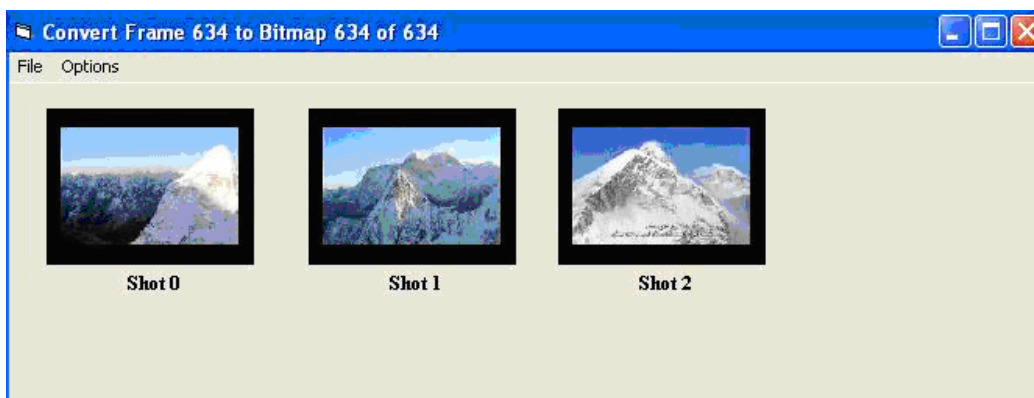


Figure 4.6: Shot detection applied on a Nature video that contains mountains

The difference between the histograms of the successive frames of the mountains query video will be illustrated in figure (4.7).

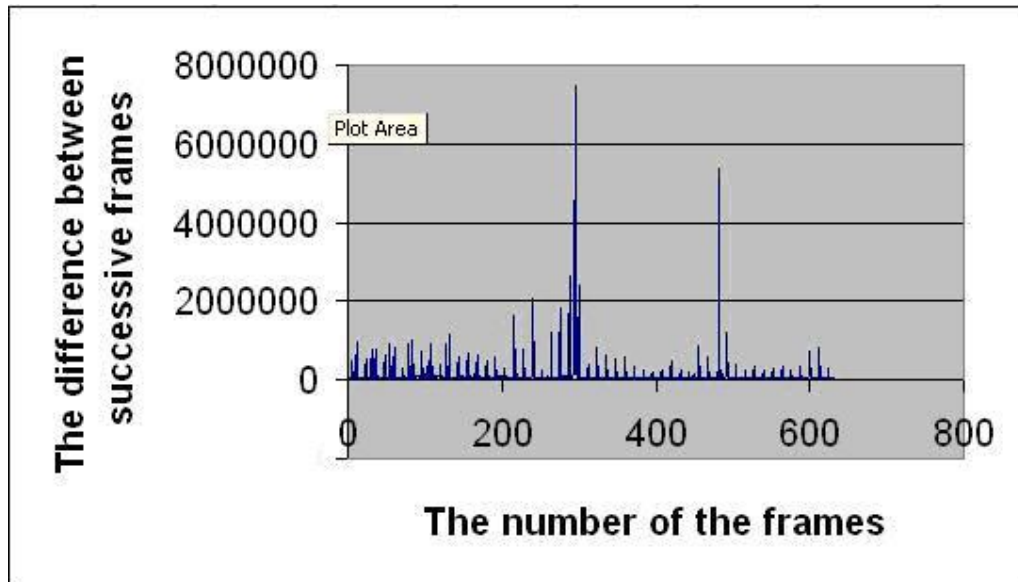


Figure 4.7: The results of the shot detection after applied on the mountains video sample

4.3.2 Testing the Video Retrieval (online phase)

The features calculation of the selected video's shot from the previous step will have four possibilities. The results of each combination will be compared according to the average performance measures (Precision and Recall). This will be declared in the next section.

A. Using the HSL color conversion with the Run length features:

The time in the retrieval results for the query video which is found in the retrieval results form (window) divided to three parts: loading database, converting the shot to features (color conversion, Haar Wavelet, and the Run length features extraction) for a shot that contains 146 frames, and the retrieval part.

For example, in figure (4.8) the time was divided into (0.0156) sec, (10.328) sec, and (0.015) sec respectively.

The precision and recall measures illustrated in the following equations:

$$\text{Precision} = \frac{\{\text{Relevant Video Shots}\} \cap \{\text{Retrieved Video Shots}\}}{\{\text{Retrieved Video Shots}\}}$$

$$\text{Recall} = \frac{\{\text{Relevant Video Shots}\} \cap \{\text{Retrieved Video Shots}\}}{\{\text{Relevant Video Shots}\}}$$

These measurements applied on the four possibilities but after showing the retrieval results.

Figure (4.8) shows the results of the HSL-Run Length combination.

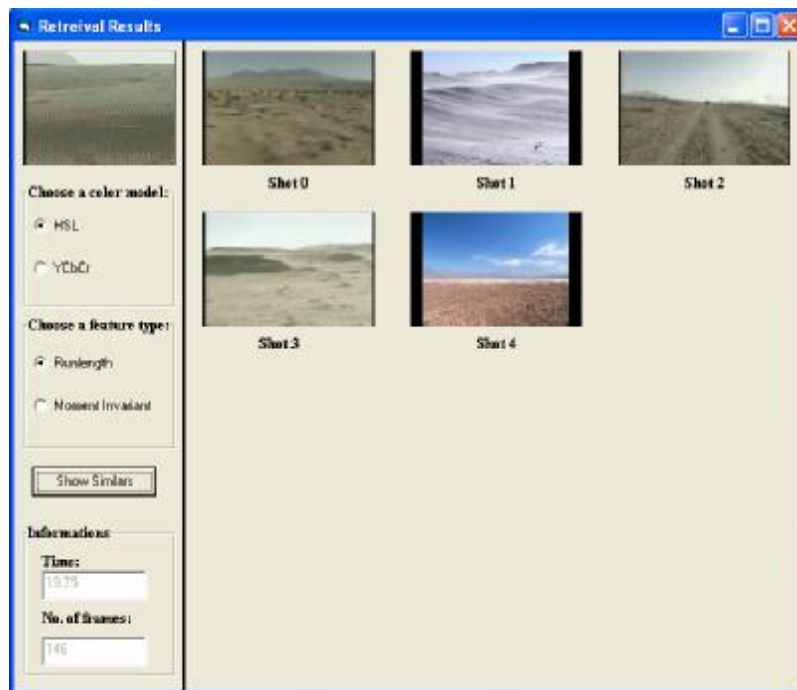


Figure 4.8: The retrieval results of the HSL and Run Length combination applied on the Desert query video

Another tested shot were taken from the football video that shown in figure (4.3), and the result illustrated in figure (4.9).

In the Retrieval Results form the video on the left part represents the shot chose from the shot detection step to find similar videos from the database as shown in the right part of the form.

Figure (4.8) is an example of this Retrieval Results representation of Desert retrieved videos.

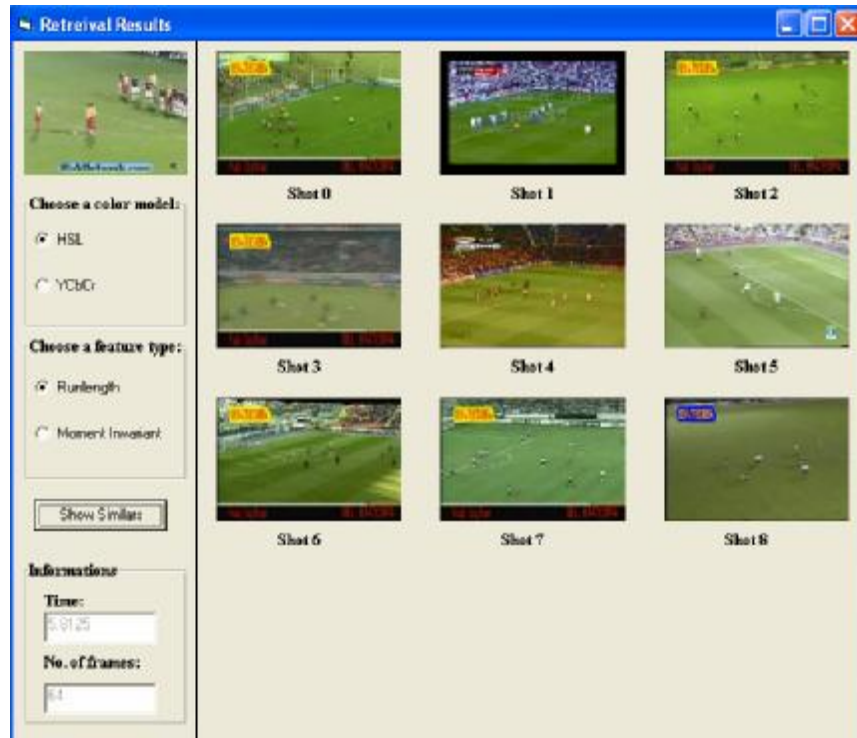


Figure 4.9: The retrieval results of the HSL and Run Length combination on Football query video

The last example will be the mountains video which is mentioned in the shot detection section. The HSL and Run length combination tested for this video as shown in figure (4.10) with the retrieval results.

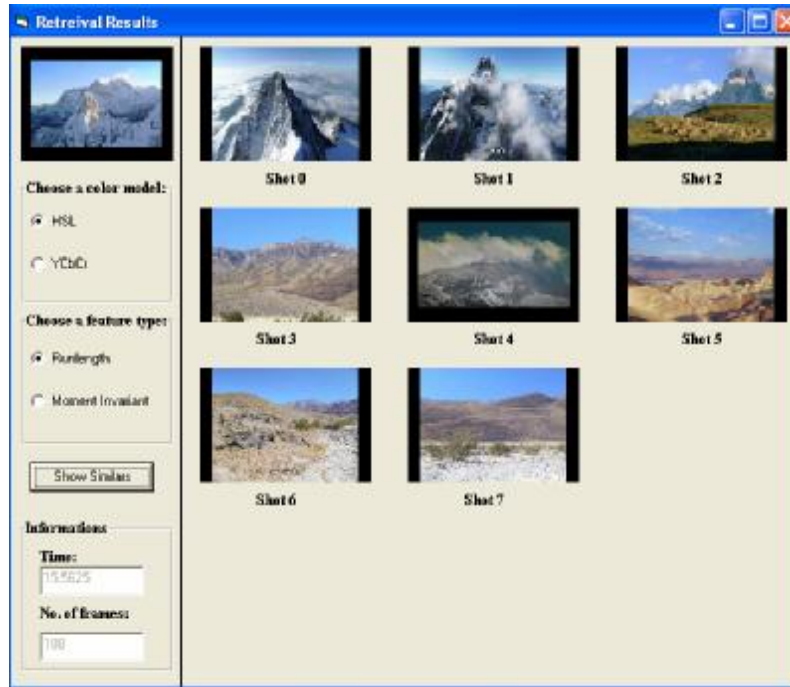


Figure 4.10: The retrieval results of the HSL and Run Length combination on Mountains query video

B. Using the HSL color conversion with the Moment Invariant features:

This combination took (0.015) sec for the database loading and the retrieval part, and the converting of the selected shot to Moment Invariant features took (9.365) sec for a video shot contains 146 frames.

Figure (4.11) shows the retrieval results, the time and number of frames of the desert query video shot.

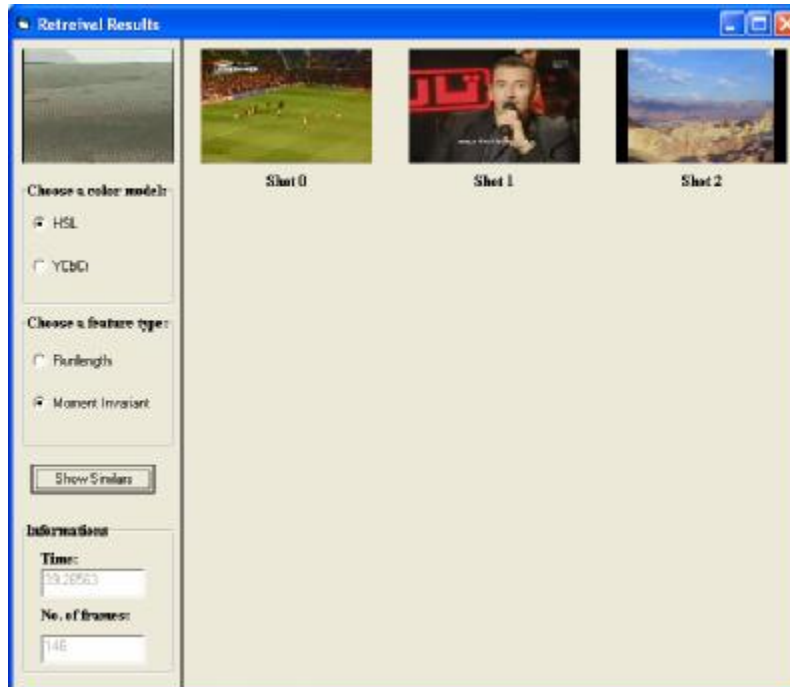


Figure 4.11: The retrieval results of the HSL and Moment invariant combination applied on the Desert query video

Also applied on the football video and the mountains video results a set of retrieved shot as shown in figures (4.12) and (4.13) respectively.

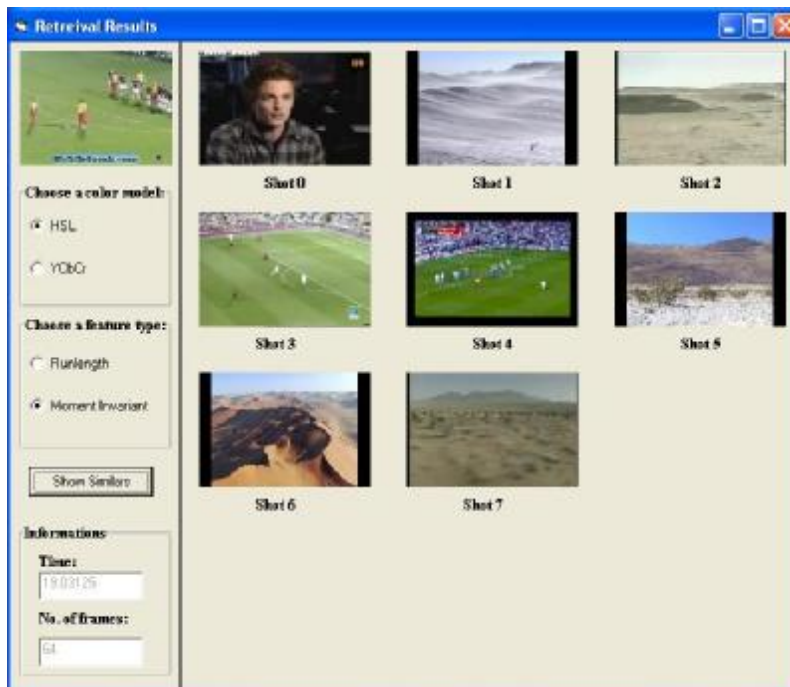


Figure 4.12: The retrieval results of the HSL and Moment invariant combination applied on the Football query video

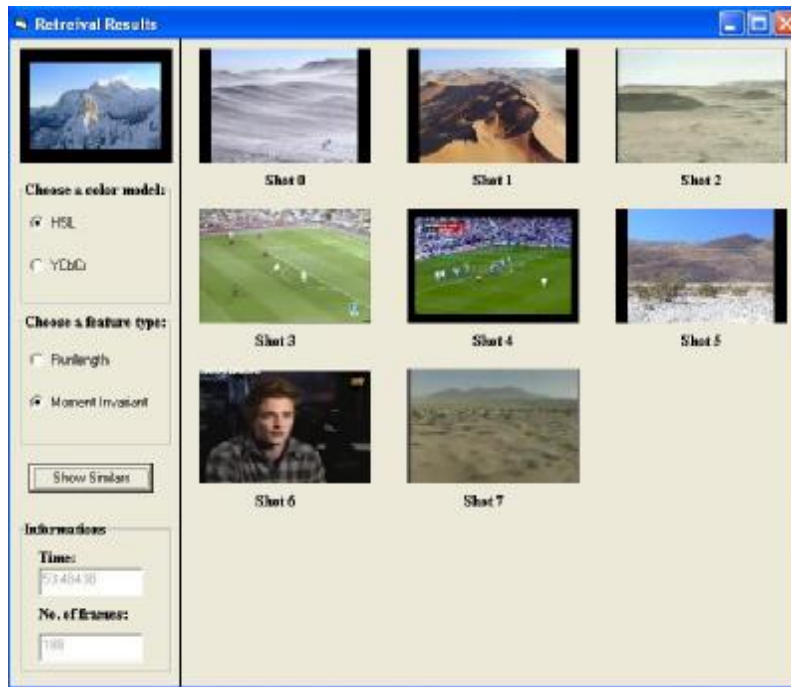


Figure 4.13: The retrieval results of the HSL and Moment invariant combination applied on the Mountains query video

C. Using the $YCbCr$ color conversion with the Run Length features:

This combination took (0.015) sec for the database loading and the retrieval part, and the converting of the selected shot to Run Length features took (19.719) sec for a video shot contains 146 frames.

Figure (4.14) shows the retrieval results, the time and number of frames of the query shot:

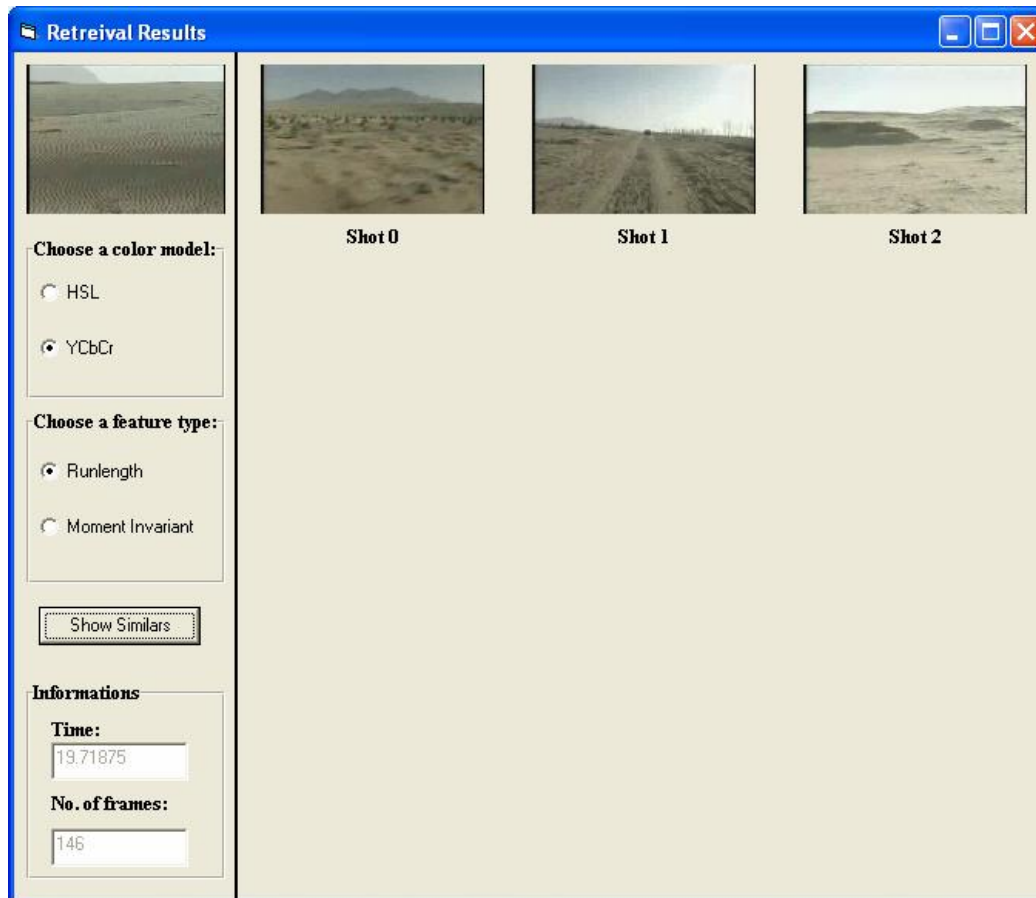


Figure 4.14: The retrieval results of the $YCbCr$ and Run Length combination applied on the desert query video

The $YCbCr$ and Run length combination will be applied on the football and mountains videos as shown in figures (4.15) and figure (4.16) respectively:



Figure 4.15: The retrieval results of the $YCbCr$ and Run Length combination applied on the Football query video

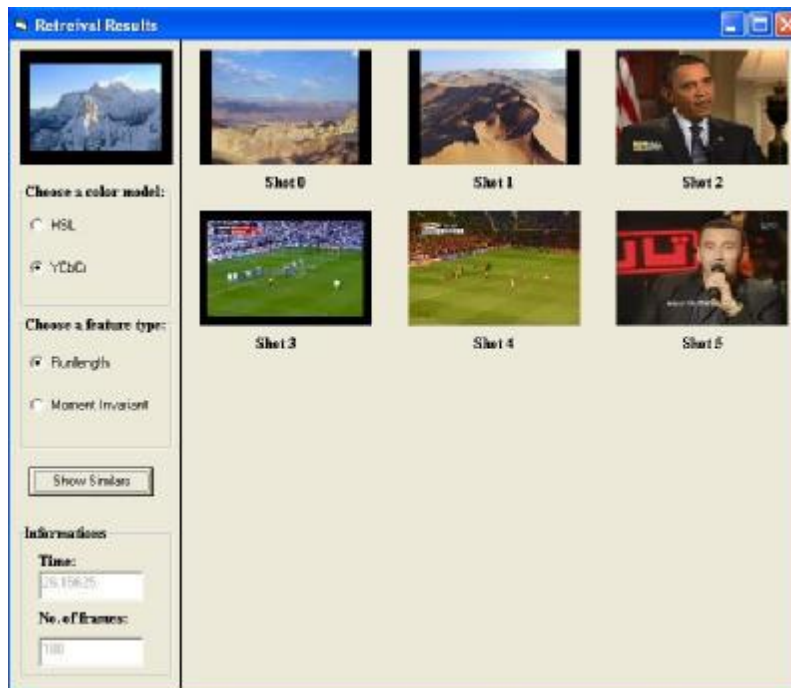


Figure 4.16: The retrieval results of the $YCbCr$ and Run Length combination applied on the Mountains query video

D. Using the YC_bC_r color conversion with the Moment Invariant features:

This combination took (0.015) sec for the database loading and the retrieval part, and the converting of the selected shot to Moment Invariant features took (39.391) sec for a video shot contains 146 frames.

Figure (4.17) shows the retrieval results, the time and number of frames of the desert query shot.

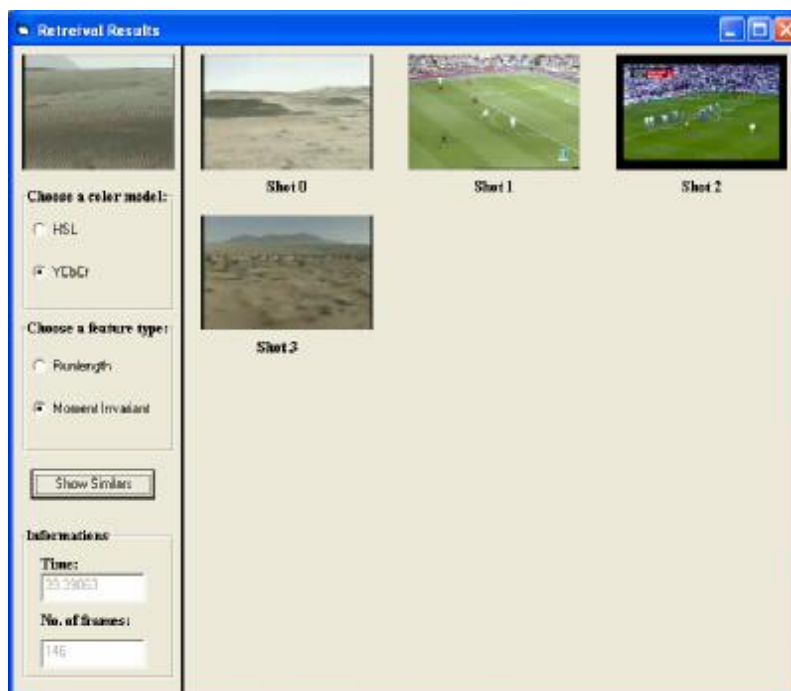


Figure 4.17: The retrieval results of the YC_bC_r and Moment Invariant combination applied on the Desert query video

The YC_bC_r and Moment Invariant combination where applied on the football and mountains videos as shown in figures (4.18) and figure (4.19), respectively.

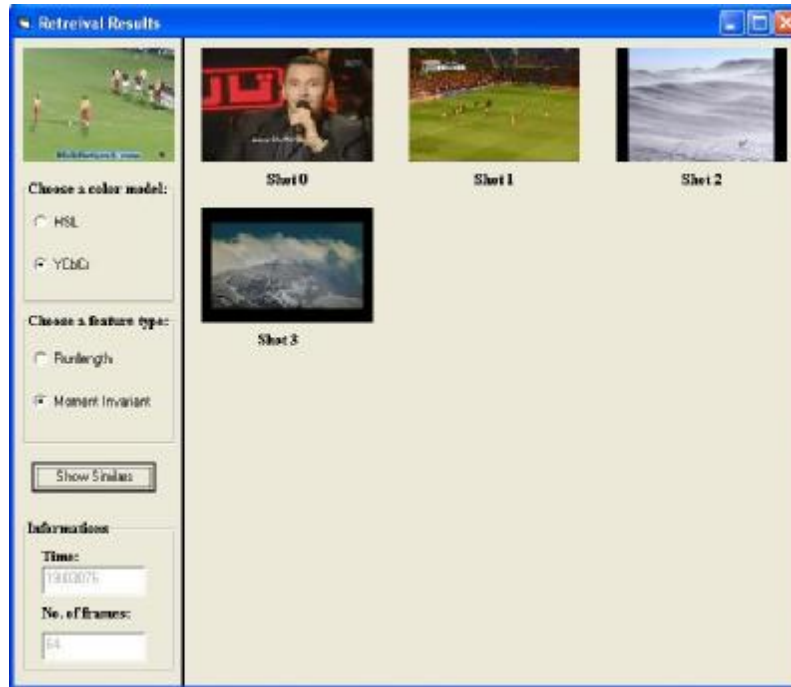


Figure 4.18: The retrieval results of the $YCbCr$ and Moment Invariant combination applied on the Football query video

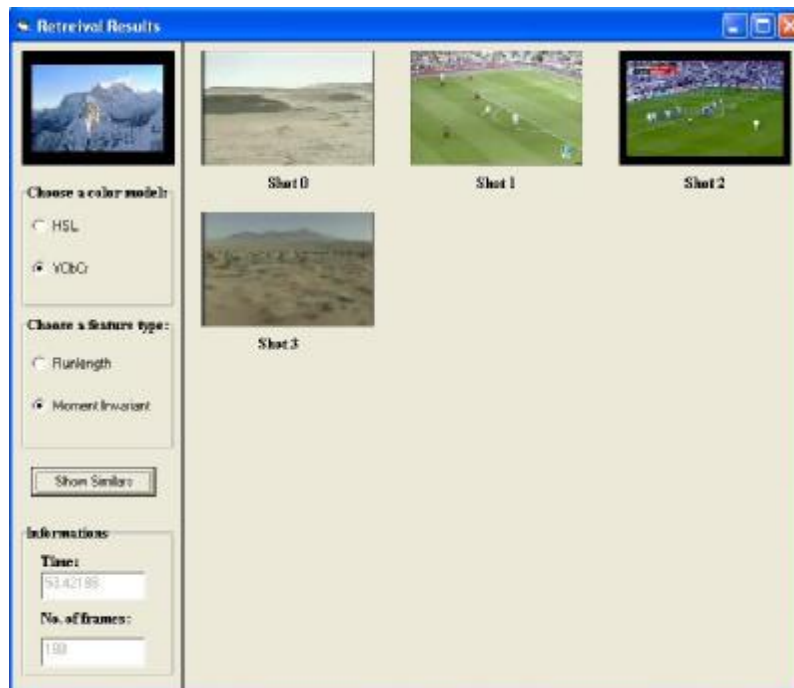


Figure 4.19: The retrieval results of the $YCbCr$ and Moment Invariant combination applied on the Mountains query video

The average performance measures of the four mentioned combinations for the desert video shot is illustrated in Figure (4.20) and assessed by the precision and the recall measures:

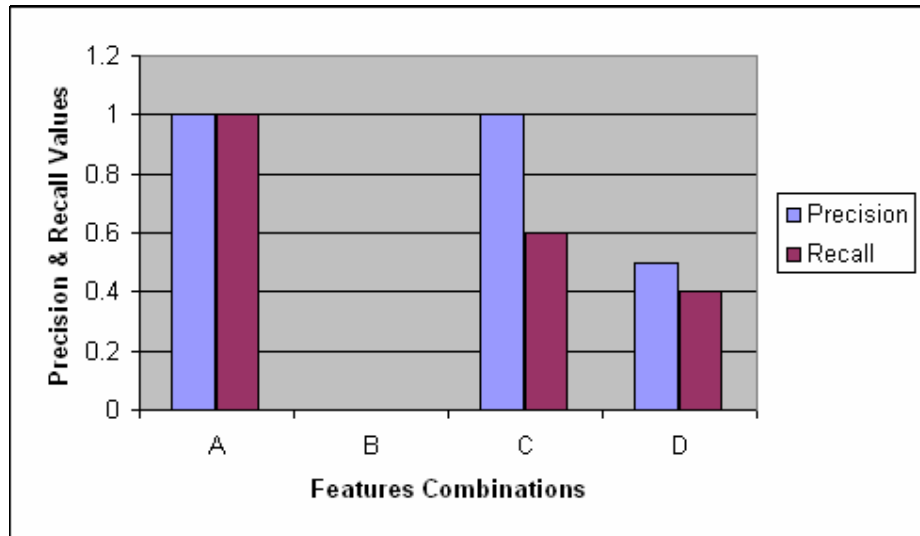


Figure 4.20: The average performance assessed by Recall and Precision Measures for the Desert query Video

From figure (4.20), it can be concluded that combination A is the best result in retrieving similar videos according to the performance measures.

And for the football and mountains query videos, the Recall and precision measures are shown in figures (4.21) and (4.22) respectively:

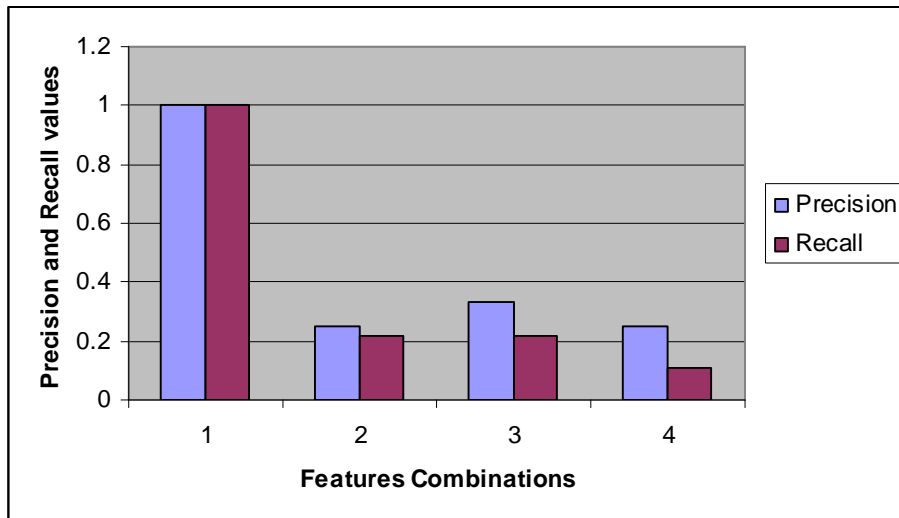


Figure 4.21: The average performance assessed by Recall and Precision Measures for the Football query Video

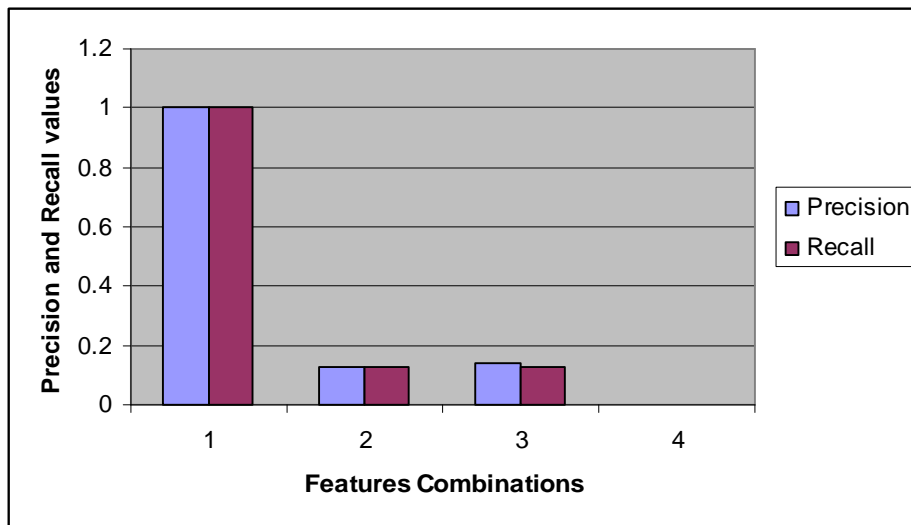


Figure 4.22: The average performance assessed by Recall and Precision Measures for the Mountains query Video

The figures (4.8) to (4.19) represent the results of Hard testing.

For the following figures, different samples from the videos database will be taken, and the results of the retrieval will show that the first video shot is the same as retrieved query one, and the minimum difference between them is 0.

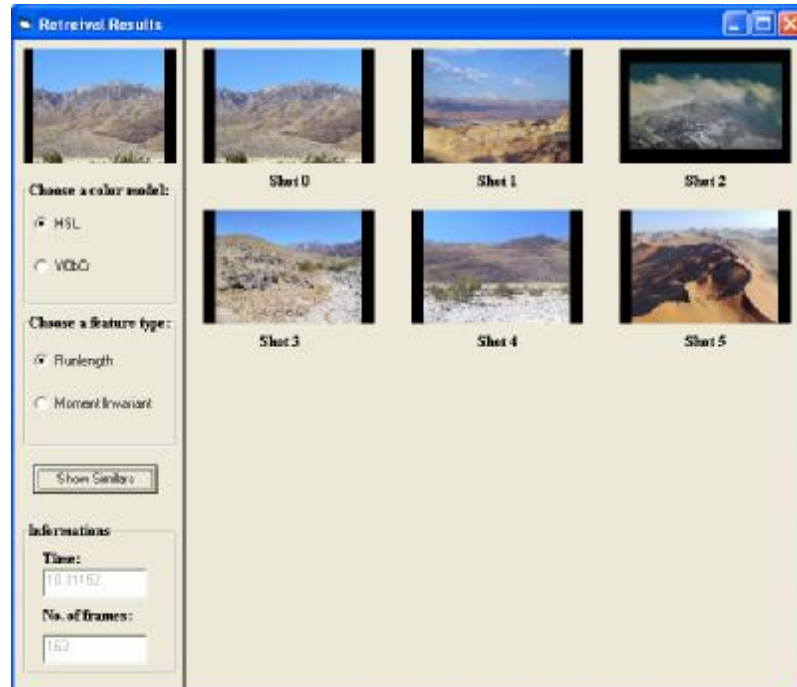


Figure 4.23: The retrieved (shot 0) is the same as the query Mountain video

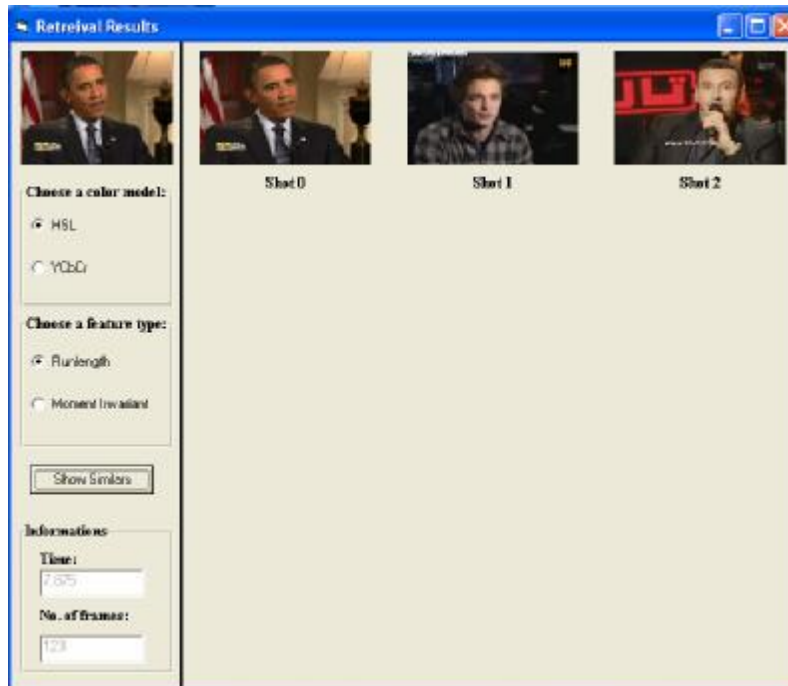


Figure 4.23: The retrieved (shot 0) is the same as the query Interview video

Chapter Five

Conclusions and Future Work

Chapter Five

Conclusions and Future Work

5.1 Conclusion

From the test results conducted to investigate the performance of the VMRS, the following remarks were derived:

1. The shot detection was implemented to detect the Cut video shot changes using the MAD on the histograms of the successive frames; the results are good for this type of video shots.
2. The variation in number of frames in the video does not affect the features decision because the mean values of the features, over all frames belong to the same shot, are adopted.
3. The performance measures (precision and recall) indicates that the used HSL color model and the Run Length features are the best combination, and the other three combinations were not good in classifying the shots.
4. The Decision Making module that combines between the colors bands according to standard ratios for the YC_bC_r and tested ratios for the HSL indicates that the tested results of the HSL that was close to the standard (0.20, 0.25, and 0.55) successes in the test and brought good retrieval results.
5. The delay time that is more than 10 second as average is mainly due to the calculations of the query shot.

5.2 Future Work

1. Using the codec files to extract the frames from the compressed videos so that the system can be managed to operate on the videos directly.
2. Combine between the Audio or the text and the video to accomplish better classification and best retrieval results
3. Intelligent techniques can be added such as Fussy to enhance retrieval results.



References

References

[Aja04]

Ajay Divukurun, Kudir A. Peker, Shih-Fu Chung, Regunuthun Radhakrishnun, and Lexing Xie, “**Video Mining: Pattern Discovery Versus Pattern Recognition**”, International Conference on Image Processing (ICIP), pp 2379-2382, IEEE, 2004.

[And02]

Andrew R. Webb, “**Statistical Pattern Recognition**”, Butterworth Heinemann, John Wiley & Sons Ltd, [Second Edition], 2002.

[Ara07]

Arasanathan A. and Nishan C. “**A Novel Video Mining System**”, Department of Electrical and Electronic Engineering, University of Bristol, UK, pp I-185- I-188, IEEE, 2007.

[Azr03]

Azriel Rosenfeld, David Doermann, and Daniel Dementhon, “**VIDEO MINING**”, Kluwer Academic Publishers, 2003.

[Bel07]

D.A. Bell, A. Beck, P. Miller, Q.X. Wu, and A. Herrera, “**Video Mining - Learning Patterns of Behavior via an Intelligent Image Analysis System**”, School of Electronics and Electrical Engineering, and Computer Science, Queen’s University, Belfast, BT7 1NN, N.I., UK, pp 460-464, IEEE, 2007.

[Bin07]

Bing Liu, “**Web Data Mining**”, Springer-Verlag Berlin Heidelberg, 2007.

[Cha05]

Chang, et al, “**A Framework for Video Structure Mining**”, National Natural Science Foundation of China (NSFC), pp 1524-128, IEEE, 2005.

[Che01]

H. D. Cheng et al, “**Color Image Segmentation: Advances & Prospects**”, Dept. of Computer Science, Utah State University Logan, pp(2-43), 2001.

[Dav03]

David Salomon, “**Data Compression**”, Third Edition, Springer, 2003.

[Dav08]

David Taniar, “**Data Mining and Knowledge Discovery Technologies**”, IGI Publishing, 2008.

[Don03]

Dong-Jun hn, et al, “**A Novel Motion-Based Representation For Video Mining**”, IEEE, pp(I11 – 469- Ill – 472), 2003.

[Don04]

Dong-Hui Xu et al, “**Run-Length Encoding For Volumetric Texture**”, IEEE, 2004.

[Eri06]

Eric Li et al, “**Towards the Parallelization of Shot Detection - a Typical Video Mining Application Study**”, Intelligent Multimedia Processing Laboratory, School of Computer Science, Telecommunications, and Information Systems, DePaul University, Chicago, Illinois, USA, 2006.

[Eyu02]

Eyüp Sabri Konak, “**A Content-Based Image Retrieval System For Texture And Color Queries**”, MSc., Bilkent, Computer Engineering and the Institute of Engineering and Science, 2002.

[Fen03]

D. Feng et al, “**Multimedia Information Retrieval and Management**”, Springer-Verlag Berlin Heidelberg, 2003.

[Fri95]

Fri. Al., “**Statistical Texture Measures Computed from Gray Level Run Length Matrices**”, Image Processing Laboratory, Department of Informatics, University of Oslo”, pp (1-6), 1995.

[Han06]

Han. Zh. and Don Ki., “**Unusual Event Detection via Multi-camera Video Mining**”, pp(1-6), IEEE, The 18th International Conference on Pattern Recognition (ICPR'06), 2006.

[Has08]

Hasnaa Imad "**Content-Based Image Retrieval System using Fuzzy Rule**", M. Sc. Thesis, Al-Nahrain University , College of Science, 2008.

[Hin06]

Hind Ali,"**Video Data Mining Using Color and Textural Features Analysis**", M. Sc. Thesis, Al-Nahrain University , College of Science,2006.

[HSL09]

"**HSL and HSV**", Internet Page, http://en.wikipedia.org/wiki/HSL_and_HSV, 2009.

[Ian05]

Ian H., Eibe Fr., “**Data Mining**”, Diane Cerra, , [Second Edition], 2005.

[Jia06]

Jiawei Han and Micheline Kamber, “**Data Mining**”, Diane Cerra, [Second Edition], 2006.

[Joa01]

João Miguel da Costa Magalhães, “**Semantic Multimedia: Mining, Fusion and Extraction**” , IEEE Multimedia, pp (1-10), 2001.

[KeX06]

Ke-Xue Dai et al, “**A Probabilistic Model For Surveillance Video Mining**”, Natural Science Foundation of China (NSFC), pp(1144-1148), 2006.

[Kex06]

Kexue Da., et al, “**Video Mining: Concepts, Approaches and Applications**”, Natural Science Foundation of China (NSFC), pp (477-480), 2006.

[Lin01]

Ling Guan et al, “**Multimedia Image and Video Processing**”, Mage and Video Compression for Multimedia Engineering, 2001.

[Loa06]

Dr. Loay E. Go. and Dhiah E. Al-Sh., “**Reference Video Frames Determination of Interframe Coding**” , Baghdad University, pp (2-9), 2006.

[Lon01]

F. Long, H. Zhang and D. D. Feng, "**FUNDAMENTALS OF CONTENT-BASED IMAGE RETRIEVAL**", IEEE Multimedia, 2001.

[Mar01]

J. P. Marques de Sa, “**Pattern Recognition**”, Springer, 2001.

[Mec06]

Mechal W.Berry, Murray Browne, "**Lecture Notes in Data Mining**", World Scientific , 2006.

[Mic04]

Mic. J.A. Be. and Gor. S. Lin., “**Data Mining Techniques For Marketing Sales, and Customer Relationship Management**”, [Second Edition], Wiley Publishing Inc., 2004.

[Nun00]

Nuno Va., “**Statistical Models of Video Structure for Content Analysis and Characterization**”, IEEE, and Andrew Lippman, pp (2-17), Vol. 9, 2000.

[Pav07]

Pavan K. Turaga et al , “**From Videos to Verbs: Mining Videos for Activities using a Cascade of Dynamical Systems**”, Department of Electrical and Computer Engineering and Center for Automation Research, UMIACS University of Maryland, College Park, 2007.

[Pio04]

Pio. Po.and Agn. Li., “**The Haar–Wavelet Transform in Digital Image Processing: Its Status and Achievements**”, Machine GRAPHICS & VISION, vol.13, pp (79-98), 2004.

[Raf92]

Raf. C. Go. And Ric. E. Wo., “**Digital Image Processing**” ,Prentice Hall, Second Edition, 1992.

[Ser03]

Ser.Th. and Kon. Ko, “**Pattern Recognition**”, Second Edition, Elsevier, 2003.

[Tao05]

Tao Mei, “**Sports Video Mining with Mosaic**”, IEEE, Proceedings of the 11th International Multimedia Modeling Conference, 2005.

[Val07]

Valery A Petrushin and Latifur Khan (Eds). “**Multimedia Data Mining and Knowledge Discovery**”, British Library Cataloguing, 2007.

[Wan02]

Y. Wang, J. Ostermann, and Y. Zhang, “**Video Pprocessing and Communications**”, Prentice-Hall, 2002.

[Wen06]

Wenlong Li et al, “**Performance Analysis of Java Concurrent Programming: A Case Study of Video Mining System**”, IEEE Xplore, 2006.

[Wes05]

Wes. Chu and Tsa. Yo. Lin, “**Foundations and Advances in Data Mining**”, pp (4-340), Springer, Vol.180, 2005.

[Wil01]

William K. Pratt et al, “**Digital Image Processing**”, John Wiley & Sons, Third Edition, 2001.

[Xin06a]

Xin.Ch., Chen.Zh., “**An Interactive Semantic Video Mining and Retrieval Platform Application in Transportation Surveillance Video for Incident Detection**”, IEEE, Proceedings of the Sixth International Conference on Data Mining (ICDM'06), 2006.

[Xin06b]

Xin C. An., “**Automatic Intravital Video Mining of Rolling and Adhering Leukocytes**”, IEEE, Proceedings of the 5th International Conference on Machine Learning and Applications (ICMLA'06), 2006.

[YiD07a]

Yi Di. and Guo. Fan, “**Segmental Hidden Markov Models for View-based Sport Video Analysis**”, National Science Foundation (NSF), 2007.

[YiD07b]

Yi Ding et al, “**Two-Layer Generative Models For Sport Video Minig**”, , IEEE, pp(1731-1734), 2007.

[YiJ06]

Yu-Jin Zhang, “**Advances in Image and Video Segmentation**”, IRM Press (an imprint of Idea Group Inc.), 2006.

[Zho09]

Zho. Zh and Ruo. Zh., “**Multimedia Data Mining**”, Taylor & Francis Group, LLC, 2009.

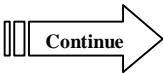


Appendix

Appendix A

A.1 Generate the Run Length Matrix

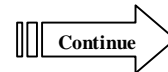
The generation of this matrix will be illustrated in details in algorithm (A.1):

<p>Algorithm (A.1) Generate RunLength Array</p>
<p>Goal: Generate the R()(run-length) array that contains all the runs in the image</p>
<p>Input: Theta//the angle of the runs</p>
<p>Output: R()//Run-length array that is generated by algorithm (3.6)</p>
<p>Step1: Set $Glevelm \leftarrow Glevel - 1$ Call $Higtm \leftarrow H2 - 1$ Call $Widm \leftarrow W2 - 1$ Check If Theta is equal to 0 Then For all y Do {Where $0 \leq y \leq Higtm$} Initialize length to 1 Set Level \leftarrow QuantGry(0, y) For all x Do {Where $1 \leq x \leq Widm$} Check If Level is equal to QuantGry(x, y) Then Increase length by 1 Else Increase R(Level, length) by 1 Initialize length to 1 Set Level \leftarrow QuantGry(x, y) End If End For Increase R(Level, length) by 1 End For Else If Theta is equal to 45 Then Check If $W2 > H2$ Then</p>
<p style="text-align: right;"></p>

```

Initialize J to 0
Initialize JJ to 0
Set Factor ← Widm - Higtm
For all x Do {Where 0 ≤ x ≤ Widm}
    Initialize length to 1
    Set Level ← QuantGry(x, 0)
    For all y Do {Where 0 ≤ y ≤ Higtm - J}
        Check If Level is equal to QuantGry(x + y, y) Then
            Increase length by 1
        Else
            Increase R(Level, length) by 1
            Initialize length to 1
            Set Level ← QuantGry(x + y, y)
        End If
    End For
    Check If JJ is equal to Factor Then
        Set J ← |x - JJ + 1|
    Else
        Increase JJ by 1
    End If
    Increase R(Level, length) by 1
End For
Set Level ← QuantGry(Widm, 0)
Increase R(Level, 1) by 1
Set Factor ← Widm - (Widm - Higtm)
For all y Do {Where 1 ≤ y ≤ Higtm}
    Initialize length to 1
    Set Level ← QuantGry(0, y)
    For all x Do {Where 1 ≤ x ≤ Factor - y}
        Check If Level is equal to QuantGry(x, y + x) Then
            Increase length by 1
        Else
            Increase R(Level, length) by 1
            Initialize length to 1
            Set Level ← QuantGry(x, y + x)
        End If
    End For
    Increase R(Level, length) by 1
End For
Set Level ← QuantGry(0, Higtm)
Increase R(Level, 1) by 1
End If
Check If H2 > W2 Then
    Set Factor ← Higtm - Widm
    Initialize J to 0
    Initialize JJ to 0
    For all y Do {where 0 ≤ y ≤ Higtm}
        Initialize length to 1

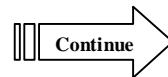
```



```

Set Level←QuantGry(0, y)
For all x Do {Where  $1 \leq x \leq \text{Widm} - J$ }
  Check If Level is equal to QuantGry(x, y + x) Then
    Increase length by 1
  Else
    Increase R(Level, length) by 1
    Initialize length to 1
    Set Level←QuantGry(x, y + x)
  End If
End For
Check If JJ is equal to Factor - 1 Then
  Set J←|y - JJ|
Else
  Increase JJ by 1
End If
Increase R(Level, length) by 1
End For
Set Level←QuantGry(0, Hightm)
Increase R(Level, 1) by 1
Set Factor←Hightm - Widm
For all x Do {Where  $1 \leq x \leq \text{Widm}$ }
  Initialize length to 1
  Set Level←QuantGry(x, 0)
  For all y Do {Where  $1 \leq y \leq \text{Factor} - x$ }
    Check If Level is equal to QuantGry(x + y, y) Then
      Increase length by 1
    Else
      Increase R(Level, length) by 1
      Initialize length to 1
      Set Level←QuantGry(x + y, y)
    End If
  End For
  Increase R(Level, length) by 1
End For
Set Level←QuantGry(Widm, 0)
Increase R(Level, 1) by 1
End If
Else If Theta is equal to 90 Then
  For all x Do {Where  $0 \leq x \leq \text{Widm}$ }
    Initialize length to 1
    Set Level←QuantGry(x, 0)
    For all y Do {Where  $1 \leq y \leq \text{Hightm}$ }
      Check If Level is equal to QuantGry(x, y) Then
        Increase length by 1
      Else
        Increase R(Level, length) by 1
        Initialize length to 1
        Set Level←QuantGry(x, y)

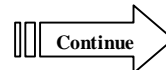
```




```

    End If
  End For
  Increase R(Level, length) by 1
End For
Else If Theta is equal to 135 Then
  Check If W2 > H2 Then
    Initialize J to 0
    Initialize JJ to 0
    Set Factor ← Widm - Higtm
    For all x Do {Where 1 ≤ x ≤ Widm}
      Initialize length to 1
      Set Level ← QuantGry(W2 - x, 0)
      For all y Do {Where 1 ≤ y ≤ Higtm - J}
        Check If Level is equal to QuantGry(W2 - x - y, y) Then
          Increase length by 1
        Else
          Increase R(Level, length) by 1
          Initialize length to 1
          Set Level ← QuantGry(W2 - x - y, y)
        End If
      End For
      Check If JJ is equal to Factor Then
        Set J ← |x - JJ|
      Else
        Increase JJ by 1
      End If
      Increase R(Level, length) by 1
    End For
    Set Level ← QuantGry(0, 0)
    Increase R(Level, 1) by 1
    Set Factor ← Widm - (Widm - Higtm)
    For all y Do {Where 1 ≤ y ≤ Higtm}
      Initialize length to 1
      Set Level ← QuantGry(Widm, y)
      For all x Do {Where 1 ≤ x ≤ Factor - y}
        Check If Level is equal to QuantGry(Widm - x, y + x) Then
          Increase length by 1
        Else
          Increase R(Level, length) by 1
          Initialize length to 1
          Set Level ← QuantGry(Widm - x, y + x)
        End If
      End For
      Increase R(Level, length) by 1
    End For
    Set Level ← QuantGry(Widm, Higtm)
    Increase R(Level, 1) By 1
  End If

```



```

        Set Level←QuantGry(x - y, y)
    End If
End For
    Increase R(Level, length) by 1
End For
Set Level←QuantGry(0, 0)
Increase R(Level, 1) by 1
Set Factor←Higtm - Widm
Initialize J to 0
Initialize JJ to 0
For all y Do {Where  $0 \leq y \leq \text{Higtm}$ }
    Initialize length to 1
    Set Level←QuantGry(Widm, y)
    For all x Do {Where  $1 \leq x \leq \text{Factor} - J$ }
        Check If Level is equal to QuantGry(Widm - x, y + x) Then
            Increase length by 1
        Else
            Increase R(Level, length) by 1
            Initialize length to 1
            Set Level←QuantGry(Widm - x, y + x)
        End If
    End For
    Check If JJ is equal to Factor - 1 Then
        Set J←|y - JJ|
    Else
        Increase JJ by 1
    End If
    Increase R(Level, length) by 1
End For
Set Level←QuantGry(Widm, Higtm)
Increase R(Level, 1) by 1
End If
End If
Check If H2 > W2 Then
    For all x Do {Where  $\text{Widm} \geq x \geq 1$ }
        Initialize length to 1
        Set Level←QuantGry(x, 0)
        For all y Do {Where  $1 \leq y \leq x$ }
            Check If Level is equal to QuantGry(x - y, y) Then
                Increase length by 1
            Else
                Increase R(Level, length) by 1
                Initialize length by 1
            End If
        End For
    End For
End If

```

A.2 Compute the Run Length features

This step is devoted to extract the Run Length features from the R array and the theta given angles shown in figure (A.2).

Algorithm (A.2) Compute RunLength Features

Goal: Calculate the Run-length features

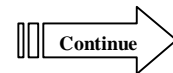
Input: R()*//array of the runs of an input video*
Theta*//the angle of the runs*

Output:
RLfeats()*//Run-length array that is generated be algorithm (3.6)*

Step1: Set Rhigt \leftarrow Glevel - 1
Check If Theta is equal to 0 Then
Set Rwid \leftarrow W2
Else If Theta is equal to 45 Then
Check If W2 > H2 Then
Set Rwid \leftarrow H2
Else
Set Rwid \leftarrow W2
End If
Else If Theta is equal to 90 Then
Set Rwid \leftarrow H2
Else If Theta is equal to 135 Then
Check If W2 > H2 Then
Set Rwid \leftarrow H2
Else
Set Rwid \leftarrow W2
End If
End If

Step2: Initialize Sum to 0
For all i Do {Where $0 \leq i \leq \text{Rhigt}$ }
For all J Do {Where $1 \leq J \leq \text{Rwid}$ }
Set Sum \leftarrow Sum + R(i, J)
End For
End For

Step3: Initialize Sum1 to 0
For all i Do {Where $0 \leq i \leq \text{Rhigt}$ }
For all J Do {Where $1 \leq J \leq \text{Rwid}$ }
Set temp \leftarrow J²
Set Sum1 \leftarrow Sum1 + R(i, J)/temp
End For
End For
Set RLFeats.SRE \leftarrow Sum1 / Sum



Step4: Initialize Sum1 to 0
For all i Do {Where $0 \leq i \leq Rhigt$ }
 For all J Do {Where $1 \leq J \leq Rwid$ }
 Set temp $\leftarrow J^2$
 Set Sum1 $\leftarrow Sum1 + R(i, J) * temp$
 End For
End For
Set RLFeats.LRE $\leftarrow Sum1 / Sum$

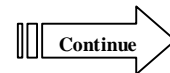
Step5: Initialize Sum1 to 0
For all i Do {Where $0 \leq i \leq Rhigt$ }
 Initialize Sum2 $\leftarrow 0$
 For all J Do {Where $1 \leq J \leq Rwid$ }
 Set Sum2 $\leftarrow Sum2 + R(i, J)^2$
 End For
 Set Sum1 $\leftarrow Sum1 + Sum2$
End For
Set RLFeats.GLNU $\leftarrow Sum1 / Sum$

Step6: Initialize Sum1 to 0
For all J Do {Where $1 \leq J \leq Rwid$ }
 Initialize Sum2 $\leftarrow 0$
 Set Sum2 $\leftarrow Sum2 + R(i, J)^2$
End For
 Set Sum1 $\leftarrow Sum1 + Sum2$
End For
Set RLFeats.RLNU $\leftarrow Sum1 / Sum$

Step7: Set RLFeats.RP $\leftarrow Sum / Rwid * Rhigt$
Initialize Sum1 $\leftarrow 0$
For all i Do {Where $1 \leq i \leq Rhigt$ }
 For all J Do {Where $1 \leq J \leq Rwid$ }
 Set Sum1 $\leftarrow Sum1 + R(i, J) / i^2$
 End For
End For
Set RLFeats.LGRE $\leftarrow Sum1 / Sum$

Step8: Initialize Sum1 to 0
For all i Do {Where $0 \leq i \leq Rhigt$ }
 For all J Do {Where $1 \leq J \leq Rwid$ }
 Set Sum1 $\leftarrow Sum1 + R(i, J) * i^2$
 End For
End For
Set RLFeats.HGRE $\leftarrow Sum1 / Sum$

Step9: Initialize Sum1 $\leftarrow 0$
For all i Do {Where $1 \leq i \leq Rhigt$ }
 For all J Do {Where $1 \leq J \leq Rwid$ }
 Set Sum1 $\leftarrow Sum1 + R(i, J) / (i^2 * J^2)$
 End For
End For
Set RLFeats.SRLGE $\leftarrow Sum1 / Sum$



```

Step10: Initialize Sum1 ← 0
  For all i Do {Where  $1 \leq i \leq Rhigt$ }
    For all J Do {Where  $1 \leq J \leq Rwid$ }
      Set Sum1 ← Sum1 + R(i, J) *  $i^2 / J^2$ 
    End For
  End For
  Set RLFeats.SRHGE ← Sum1/Sum

Step11: Initialize Sum1 to 0
  For all i Do {Where  $0 \leq i \leq Rhigt$ }
    For all J Do {Where  $1 \leq J \leq Rwid$ }
      Set Sum1 ← Sum1 + R(i, J) *  $J^2 / i^2$ 
    End For
  End For
  Set RLFeats.LRLGE ← Sum1/Sum

Step12: Initialize Sum1 to 0
  For all i Do {Where  $1 \leq i \leq Rhigt$ }
    For all J Do {Where  $1 \leq J \leq Rwid$ }
      Set Sum1 ← Sum1 + R(i, J) *  $J^2 * i^2$ 
    End For
  End For
  Set RLFeats.LRHGE ← Sum1/Sum

```

A.3 The generation of the Quantized Run Length features array

The input to this step is the RLfeats (array of the Run Length features) and the Down-Sampled images. This step illustrated in figure (A.3).

Algorithm (A.3) Quanti

Goal: quantizing the input LL array from Haar algorithm

Input: D-Sample()/array of that is resulted from the Down-Sampling algorithm

LastImage//the last image in the input video shot

FirstImage//the first image in the input video shot

RLFeats//the array of the run-length features

Output:

Fll//text file that contains the output results

Temp_Mean_Runlength()/array contains the output quantized features

Step1: Set Glevel←64

For all i Do {Where $0 \leq i \leq \text{LastImage} - \text{FirstImage}$ }

Set Fll←i&"_LL_y"

For all K Do {Where $0 \leq K \leq W2$ }

For all J Do {Where $0 \leq J \leq H2$ }

Set QuantGry(K, J)←(Haar_LL(i, K, J).y + 255) * 63 Div 510

End For

End For

Call Run_Length

Write Fll, RLFeats.SRE*100, RLFeats.LRE*100,

RLFeats.GLNU*100, RLFeats.RLNU*100, RLFeats.RP*100,

RLFeats.LGRE*100, RLFeats.HGRE*100, RLFeats.SRLGE*100,

RLFeats.SRHGE*100, RLFeats.LRLGE*100,

and RLFeats.LRHGE*100 to File

Set Temp_Mean_Runlength(Count_i).GLNU.y←

Temp_Mean_Runlength(Count_i).GLNU.y+RLFeats.GLNU

Set Temp_Mean_Runlength(Count_i).HGRE.y←

Temp_Mean_Runlength(Count_i).HGRE.y+RLFeats.HGRE

Set Temp_Mean_Runlength(Count_i).LGRE.y←

Temp_Mean_Runlength(Count_i).LGRE.y+RLFeats.LGRE

Set Temp_Mean_Runlength(Count_i).LRE.y←

Temp_Mean_Runlength(Count_i).LRE.y+RLFeats.LRE

Set Temp_Mean_Runlength(Count_i).LRHGE.y←

Temp_Mean_Runlength(Count_i).LRHGE.y+RLFeats.LRHGE

Set Temp_Mean_Runlength(Count_i).LRLGE.y←

Temp_Mean_Runlength(Count_i).LRLGE.y+RLFeats.LRLGE

Set Temp_Mean_Runlength(Count_i).RLNU.y←

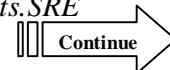
Temp_Mean_Runlength(Count_i).RLNU.y+RLFeats.RLNU

Set Temp_Mean_Runlength(Count_i).RP.y←

Temp_Mean_Runlength(Count_i).RP.y+RLFeats.RP

Set Temp_Mean_Runlength(Count_i).SRE.y←

Temp_Mean_Runlength(Count_i).SRE.y+RLFeats.SRE

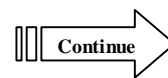


```

Set Temp_Mean_Runlength(Count_i).SRHGE.y←
    Temp_Mean_Runlength(Count_i).SRHGE.y+RLFeats.SRHGE
Set Temp_Mean_Runlength(Count_i).SRLGE.y←
    Temp_Mean_Runlength(Count_i).SRLGE.y+RLFeats.SRLGE
Set Fll←i&"_LL_cb"
For all K Do {Where 0≤K≤W2}
    For all J Do {Where 0≤J≤H2}
        Set QuantGry(K, J)←(Haar_LL(i, K, J).Cb+255)*63 Div 510
    End For
End For
Call Run_Length
Write Fll, RLFeats.SRE*100, RLFeats.LRE*100,
    RLFeats.GLNU*100, RLFeats.RLNU*100, RLFeats.RP*100,
    RLFeats.LGRE*100, RLFeats.HGRE*100, RLFeats.SRLGE*100,
    RLFeats.SRHGE*100, RLFeats.LRLGE*100,
    and RLFeats.LRHGE*100 to File
Set Temp_Mean_Runlength(Count_i).GLNU.Cb←
    Temp_Mean_Runlength(Count_i).GLNU.Cb+RLFeats.GLNU
Set Temp_Mean_Runlength(Count_i).HGRE.Cb←
    Temp_Mean_Runlength(Count_i).HGRE.Cb+RLFeats.HGRE
Set Temp_Mean_Runlength(Count_i).LGRE.Cb←
    Temp_Mean_Runlength(Count_i).LGRE.Cb+RLFeats.LGRE
Set Temp_Mean_Runlength(Count_i).LRE.Cb←
    Temp_Mean_Runlength(Count_i).LRE.Cb+RLFeats.LRE
Set Temp_Mean_Runlength(Count_i).LRHGE.Cb←
    Temp_Mean_Runlength(Count_i).LRHGE.Cb+RLFeats.LRHGE
Set Temp_Mean_Runlength(Count_i).LRLGE.Cb←
    Temp_Mean_Runlength(Count_i).LRLGE.Cb+RLFeats.LRLGE
Set Temp_Mean_Runlength(Count_i).RLNU.Cb←
    Temp_Mean_Runlength(Count_i).RLNU.Cb+RLFeats.RLNU
Set Temp_Mean_Runlength(Count_i).RP.Cb←
    Temp_Mean_Runlength(Count_i).RP.Cb+RLFeats.RP
Set Temp_Mean_Runlength(Count_i).SRE.Cb←
    Temp_Mean_Runlength(Count_i).SRE.Cb+RLFeats.SRE
Set Temp_Mean_Runlength(Count_i).SRHGE.Cb←
    Temp_Mean_Runlength(Count_i).SRHGE.Cb+RLFeats.SRHGE
Set Temp_Mean_Runlength(Count_i).SRLGE.Cb←
    Temp_Mean_Runlength(Count_i).SRLGE.Cb+RLFeats.SRLGE

Set Fll←i&"_LL_cr"
For all K Do {Where 0≤K≤W2}
    For all J Do {Where 0≤J≤H2}
        Set QuantGry(K, J)←(Haar_LL(i, K, J).Cr+255)*63 Div 510
    End For
End For
Call Run_Length

```

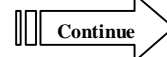


Write *Fl*, *RLFeats.SRE*100*, *RLFeats.LRE*100*,
*RLFeats.GLNU*100*, *RLFeats.RLNU*100*, *RLFeats.RP*100*,
*RLFeats.LGRE*100*, *RLFeats.HGRE*100*, *RLFeats.SRLGE*100*,
*RLFeats.SRHGE*100*, *RLFeats.LRLGE*100*,
*and RLFeats.LRHGE*100 to File*
Set *Temp_Mean_Runlength(Count_i).GLNU.Cr*←
Temp_Mean_Runlength(Count_i).GLNU.Cr+RLFeats.GLNU
Set *Temp_Mean_Runlength(Count_i).HGRE.Cr*←
Temp_Mean_Runlength(Count_i).HGRE.Cr+RLFeats.HGRE
Set *Temp_Mean_Runlength(Count_i).LGRE.Cr*←
Temp_Mean_Runlength(Count_i).LGRE.Cr+RLFeats.LGRE
Set *Temp_Mean_Runlength(Count_i).LRE.Cr*←
Temp_Mean_Runlength(Count_i).LRE.Cr+RLFeats.LRE
Set *Temp_Mean_Runlength(Count_i).LRHGE.Cr*←
Temp_Mean_Runlength(Count_i).LRHGE.Cr+RLFeats.LRHGE
Set *Temp_Mean_Runlength(Count_i).LRLGE.Cr*←
Temp_Mean_Runlength(Count_i).LRLGE.Cr+RLFeats.LRLGE
Set *Temp_Mean_Runlength(Count_i).RLNU.Cr*←
Temp_Mean_Runlength(Count_i).RLNU.Cr+RLFeats.RLNU
Set *Temp_Mean_Runlength(Count_i).RP.Cr*←
Temp_Mean_Runlength(Count_i).RP.Cr+RLFeats.RP
Set *Temp_Mean_Runlength(Count_i).SRE.Cr*←
Temp_Mean_Runlength(Count_i).SRE.Cr+RLFeats.SRE
Set *Temp_Mean_Runlength(Count_i).SRHGE.Cr*←
Temp_Mean_Runlength(Count_i).SRHGE.Cr+RLFeats.SRHGE
Set *Temp_Mean_Runlength(Count_i).SRLGE.Cr*←
Temp_Mean_Runlength(Count_i).SRLGE.Cr+RLFeats.SRLGE

End For

Step2: *Set i*←*LastImage – FirstImage*

Set *Temp_Mean_Runlength(Count_i).GLNU.y*←
Temp_Mean_Runlength(Count_i).GLNU.y / i
Set *Temp_Mean_Runlength(Count_i).HGRE.y*←
Temp_Mean_Runlength(Count_i).HGRE.y / i
Set *Temp_Mean_Runlength(Count_i).LGRE.y*←
Temp_Mean_Runlength(Count_i).LGRE.y / i
Set *Temp_Mean_Runlength(Count_i).LRE.y*←
Temp_Mean_Runlength(Count_i).LRE.y / i
Set *Temp_Mean_Runlength(Count_i).LRHGE.y*←
Temp_Mean_Runlength(Count_i).LRHGE.y / i
Set *Temp_Mean_Runlength(Count_i).LRLGE.y*←
Temp_Mean_Runlength(Count_i).LRLGE.y / i
Set *Temp_Mean_Runlength(Count_i).RLNU.y*←
Temp_Mean_Runlength(Count_i).RLNU.y / i
Set *Temp_Mean_Runlength(Count_i).RP.y*←
Temp_Mean_Runlength(Count_i).RP.y / i
Set *Temp_Mean_Runlength(Count_i).SRE.y*←
Temp_Mean_Runlength(Count_i).SRE.y / i



Set Temp_Mean_Runlength(Count_i).SRHGE.y←
Temp_Mean_Runlength(Count_i).SRHGE.y/i

Set Temp_Mean_Runlength(Count_i).SRLGE.y←
Temp_Mean_Runlength(Count_i).SRLGE.y/i

Set Temp_Mean_Runlength(Count_i).GLNU.Cb←
Temp_Mean_Runlength(Count_i).GLNU.Cb / i

Set Temp_Mean_Runlength(Count_i).HGRE.Cb←
Temp_Mean_Runlength(Count_i).HGRE.Cb / i

Set Temp_Mean_Runlength(Count_i).LGRE.Cb←
Temp_Mean_Runlength(Count_i).LGRE.Cb / i

Set Temp_Mean_Runlength(Count_i).LRE.Cb←
Temp_Mean_Runlength(Count_i).LRE.Cb / i

Set Temp_Mean_Runlength(Count_i).LRHGE.Cb←
Temp_Mean_Runlength(Count_i).LRHGE.Cb / i

Set Temp_Mean_Runlength(Count_i).LRLGE.Cb←
Temp_Mean_Runlength(Count_i).LRLGE.Cb / i

Set Temp_Mean_Runlength(Count_i).RLNU.Cb←
Temp_Mean_Runlength(Count_i).RLNU.Cb / i

Set Temp_Mean_Runlength(Count_i).RP.Cb←
Temp_Mean_Runlength(Count_i).RP.Cb / i

Set Temp_Mean_Runlength(Count_i).SRE.Cb←
Temp_Mean_Runlength(Count_i).SRE.Cb / i

Set Temp_Mean_Runlength(Count_i).SRHGE.Cb←
Temp_Mean_Runlength(Count_i).SRHGE.Cb / i

Set Temp_Mean_Runlength(Count_i).SRLGE.Cb←
Temp_Mean_Runlength(Count_i).SRLGE.Cb / i

Set Temp_Mean_Runlength(Count_i).GLNU.Cr←
Temp_Mean_Runlength(Count_i).GLNU.Cr / i

Set Temp_Mean_Runlength(Count_i).HGRE.Cr←
Temp_Mean_Runlength(Count_i).HGRE.Cr / i

Set Temp_Mean_Runlength(Count_i).LGRE.Cr←
Temp_Mean_Runlength(Count_i).LGRE.Cr / i

Set Temp_Mean_Runlength(Count_i).LRE.Cr←
Temp_Mean_Runlength(Count_i).LRE.Cr / i

Set Temp_Mean_Runlength(Count_i).LRHGE.Cr←
Temp_Mean_Runlength(Count_i).LRHGE.Cr / i

Set Temp_Mean_Runlength(Count_i).LRLGE.Cr←
Temp_Mean_Runlength(Count_i).LRLGE.Cr / i

Set Temp_Mean_Runlength(Count_i).RLNU.Cr←
Temp_Mean_Runlength(Count_i).RLNU.Cr / i

Set Temp_Mean_Runlength(Count_i).RP.Cr←
Temp_Mean_Runlength(Count_i).RP.Cr / i

Set Temp_Mean_Runlength(Count_i).SRE.Cr←
Temp_Mean_Runlength(Count_i).SRE.Cr / i

Set Temp_Mean_Runlength(Count_i).SRHGE.Cr←
Temp_Mean_Runlength(Count_i).SRHGE.Cr / i

Set Temp_Mean_Runlength(Count_i).SRLGE.Cr←
Temp_Mean_Runlength(Count_i).SRLGE.Cr / i

الخلاصة

مع الزيادة والتضخم الحاصل في المعلومات المتعددة الوسائط (فيديو، صوت، صور، صفحات شبكة الانترنت)، والناس ليس لديهم الوقت الكافي للنظر لهذه المعلومات، واهتمامات البشر أصبحت من الأشياء الثمينة. لذا، أصبح من الضروري إيجاد وسيلة لتحليل، تصنيف، تلخيص، إيجاد وتشخيص الميول فيها، وتعيين الأمور الخارجة عن القياس بشكل أوتوماتيكي. عدة باحثين أحسوا بالحاجة إلى طرق تعدين البيانات للتعامل مع هذه الكمية من المعلومات. تعدين البيانات المتعددة الوسائط يتضمن أنواع متعددة من البيانات كالفديو، الصوت، والصور، الخ. هذا العمل يتضمن عدة مجالات التي تعالج مجموعة كبيرة من المعلومات مثل التعرف، والتمييز، التصنيف، الخ.

تعدين الفيديو هو أكثر الأنواع أهمية بالنسبة للتعدين لان الفيديو أصبح وسيله سهلة ومتداولة بكثرة لإيصال رسالة أو فكرة ما عن طريق دردشة الفيديو، الأفلام، والدعايات كما أنها تعتبر من احد طرق الحماية مثل أفلام المراقبة والكثير من الاستخدامات التي تملأ العالم. في هذا البحث، تم بناء نظام تعدين واسترجاع بيانات الفيديو. وهذا النظام يتضمن مرحلتين: مرحلة التسجيل ومرحلة مطابقة الاستفسار. المرحلة التسجيل تبني قاعدة بيانات الفيديو. قاعدة البيانات تحتوي على أصناف مختلفة تعقدت وفقا للخصائص النسيجية أو الشكلية مع نماذج لونية مختلفة. عملية التعرف أنجزت باستخدام خوارزمية (K-means). بينما، المرحلة مطابقة الاستفسار استثمرت لاسترجاع الفيديوهات المشابهة للفيديو المدخل. قبل استخراج الخصائص، يقطع الفيديو المدخل إلى عدة لقطات اعتمادا على الفرق بين المدرج الاحصائي للصور المتتالية، ثم تحسب الخصائص للقطعة المختارة من أجل استرجاع اللقطات المشابهة من قاعدة بيانات الفيديو.

أداء النظام تم تقييمه باستخدام مقاييس (recall and precession). وهذا النظام أعطى نتائج جيدة ومشجعة.



جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

استرجاع الفيديو باستخدام أنواع مختلفة من نماذج اللون

رسالة

مقدمه إلى كلية العلوم في جامعة النهرين كجزء من
متطلبات نيل درجة الماجستير في علوم الحاسبات

من قبل

جابر عبدالله جابر

(بكالوريوس جامعة النهرين 2005)

إشراف

د. سوسن كمال ثامر