CONTENT BASED RETRIEVAL AND INDEXING FOR MULTIMEDIA APPLICATIONS

by

Omkar V. Sardessai (P.R NO:201104286)
Preetesh P. Kunde (P.R NO:201104815)
Sarvesh S. Shirodker (P.R NO:201008302)
Shripad A. Bhat (P.R NO:201008267)

A project report submitted
in partial fulfillment of the requirements
for the degree of

Bachelor of Engineering
in
Electronics and Telecommunication Engineering

Goa University

Under the guidance of

Mr. Samarth Borkar
Assistant Professor
Department of Electronics and Telecommunication Engineering,

Department of Electronics and Telecommunication Engineering
Goa College of Engineering
June 2014

Project Approval Sheet

This is to certify that

Omkar Sardessai (P.R NO:201104286)
Preetesh Kunde (P.R NO:201104815)
Sarvesh Shirodker (P.R NO:201008302)
Shripad Bhat (P.R NO:201008267)

Have been admitted to the candidacy of Bachelors degree of Electronics and Telecommunication Engineering in 2013-2014 and they have undertaken the dissertation entitled **CONTENT BASED RETRIEVAL AND INDEXING FOR MULTIMEDIA APPLICATIONS** which is approved for the degree of BE (Electronics and Telecommunication) under Goa University as it is found satisfactory.

Internal Guide
Mr. Samarth Borkar
Assistant Professor
Electronics and Telecommunication Engineering Department
Goa College Of Engineering

Dr. Ameeta Amonkar
Professor & Head of Department
Electronics and Telecommunication Department
Goa College of Engineering
Farmagudi, Ponda-Goa

Dr. V.N.Shet
Principal
Goa College of Engineering
Farmagudi, Ponda-Goa

Place: Goa College of Engineering,Farmagudi
Date: /June/2014

CERTIFICATE

This is to certify that the project

CONTENT BASED RETRIEVAL AND INDEXING FOR MULTIMEDIA APPLICATIONS

SUBMITTED BY

Omkar Sardessai (P.R NO:201104286)
Preetesh Kunde (P.R NO:201104815)
Sarvesh Shirodker (P.R NO:201008302)
Shripad Bhat (P.R NO:201008267)

has been successfully completed and defended in the year 2013-14 as a partial fulfillment of the requirement for the Degree of BACHELOR OF ENGINEERING in Electronics and Telecommunication Engineering, at Goa College of Engineering, Farmagudi, Ponda-Goa

_____ _____

(Internal Examiner) (External Examiner)

Place:
Date:

Acknowledgment

We wish to express our most sincere gratitude and appreciation to our project guide, **Prof. Samarth B. Borkar**. for his guidance, patience, positive attitude and encouragement throughout the development of this project. We have been extremely lucky to have a guide who cared so much about our work, was very systematic, practical and who responded to our questions and queries so promptly.

Completing this work would have been all the more difficult were it not for the support and guidance provided by **Prof. Abhishek Gudekar**, of Smt. Parvatibai Chowgule College of Arts and science. He provided us with a boost to start off with our project by providing us with the knowledge of some of the basic concepts required in this project.

Even though she was not technically involved in helping out with our project, our Head of Department, **Dr. Ameeta Amonkar**, was a constant source of positivity and encouragement which gave us more energy to work towards the fulfilment of our project.

In the erstwhile stages of our project, we required a path, a mental layout in order to commence our project and for this, a peaceful and positive atmosphere was essential, where we could discuss our plans and ideas pertaining to the project and this place was the **Krishnadas Shama State Central Library, Panaji**. We would also like to thank its staff for providing us with all the source of information we required to start off with the project.

We would also like to wish our deepest thanks to **Dr. V. N. Shet**, Principal of Goa College of Engineering for his help and support.

Last, but not the least, we would like to thank our family members, especially our parents, well wishers, friends and the All Mighty, without them we would have never made it this far in life. They have given us the moral support and encouragement towards the completion of this project. Without them, this work would not have taken shape.

ABSTRACT

With the advent of computer technology, a large amount of data is generated and stored. The data is stored in various ways such as in the form of text, image, videos, etc. As the stored data is increasing enormously over the years, efficient retrieval of the data has become difficult. Video sharing on the web is constantly on the rise due to the fact that mobile phones equipped with cameras has increased over the years and number of people accessing internet has increased manifold and video is one of the most preferred type of data owing to its content richness.

Content-based retrieval of video data has been still an active research area. The efficient retrieval of video data is proven difficult because of the unstructured way videos are stored. The code for segmenting, annotating and indexing of videos is done using OpenCV Python. In this project, we have come up with a tool for easy and efficient viewing of videos using OpenCV Python.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Background

There has been a tremendous growth in the usage of digital data. The most wide spread and common among the digital data is digital video which has become an essential part of our lives. Even if there are many tools to retrieve and process the digital data, it is a little cumbersome, less effective and less efficient. In the present day scenario the technologies that allow easy capture and sharing of digitized video have been rapidly developing, for example internet enabled mobile phones equipped with digital camera. To date, handheld devices and the Internet have become a common method to create and transport video documents. As a result, there has been a huge increase in the utilization of video .So as to keep pace with the growth of video data; enhancing the current solutions for Content Based Video Retrieval (CBVR) is an important task.

Several content-based video retrieval (CBVR) systems have been proposed in the past, but they still suffer from the following challenging problems [8]:

(a) Semantic gap

(b) Semantic video concept modelling

(c) Semantic video classification

(d) Concept oriented video database indexing and access [8]

So as to meet the demands of a user for video retrieval, numerous techniques have been proposed to bridge the semantic gap between the features that can be extracted fully without human intervention and the richness of user queries in video retrieval. One of them is domain-specific approach which exploits the typical characteristics of a particular video genre to design the most effective tools for the content extraction and indexing, and the associated browsing and retrieval.
In most Content Based Video Retrieval Systems (CBVR) the entire system could be broadly divided into three parts which are:

(a)Video Segmentation [6, 9]

(b) Video Annotation [29, 31, 34, 35, 37]

(c) Video Indexing [8, 14, 18, 31]

Depending on the effectiveness of each of the part mentioned above, the efficiency and accuracy of the retrieval systems is dependent.

1.2 Objectives

Digital video is an essential component of new media applications. It requires special technical support in processing, communication, and storage. This work presents technologies for video segmentation, indexing, annotation and browsing, in order to support various multimedia viewing.

In order to enable users to view the multimedia, two types of functionality is required, key content browsing and search. The former allows users to efficiently browse through or search for key content of the video without decoding and viewing the entire video stream. The key content refers to the key frames in video sequences, prominent video objects [16].

1.3 Need for the project

In the present scenario, video is the richest source of information available to the users. The greatest problem associated with the video, is organization and efficient retrieval of the content from the videos.

It is very difficult for the users to find the relevant information from the video due to the unstructured way videos are stored. Hence there is need for an efficient content based retrieval system.

1.4 Scope for project

The entire project can be accomplished in four stages i.e. video segmentation, annotation, indexing and database creation. In video segmentation, video is classified into frames, which is the smallest part of video.

Annotation is done by giving specific labels to the features in the frames. Indexing is a method that can correlate the features from different frames. Database is created to store all the processed multimedia files.

1.5 Implementation of Project

The project can be used for efficient viewing of offline videos. It helps an user to search within a video. It can be implemented in many videos such as those of conferences, sports, Subjective Lecture Videos.

Chapter 2

LITERATURE SURVEY

2.1 Video Segmentation

As shown in fig. 2.1. Video is made up of scenes. Each scene is made up of shots. The start and cut of a camera is called a shot. Each shot is then further divided into frames. A frame is the basic component of a video.

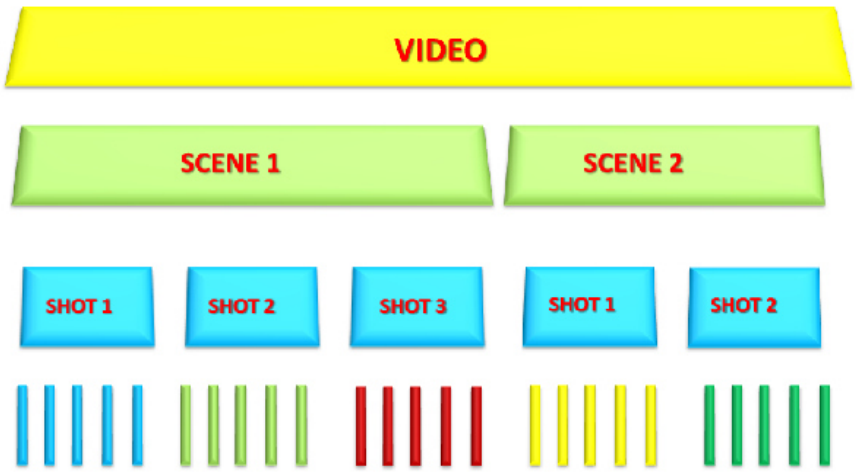

Figure 2.1: Hierarchical Division of Video

- **Frames** - They are the smallest unit of video which appear in form of images.
- **Shot** –Frames recorded in one camera operation form a shot.
- **Scene** –One or several related shots are combined in a scene.
- **Sequence** –A series of related scenes forms a sequence.
- **Video** – A video is composed of different story units such as shots, scenes, and sequences arranged according to some logical structure (defined by the screen play).

2.1.1 Temporal Segmentation

Temporal segmentation of video, as the name suggests, deals with partitioning video in time into shots and scenes. This segmentation is done in a way so as to put together

logically related frames. Temporal segmentation for shots is performed by detecting the transitions between the shots [9, 20]. The transition is characterized by a cut between the shots. The transition is either sudden or gradual and detection of the transition in former is easier by comparing the consecutive frames of video. If the transition is gradual, it is difficult to detect the transition as compared to sudden transition. One of the simplest algorithms used for temporal segmentation is Pixel wise comparison. In this algorithm, successive frames are compared. If the difference between the pixels is greater than the threshold, then transition is detected.

$$D(i, i+1) = \frac{\sum x \sum y \, |Pi(x, y) - Pi+1(x, y)|}{(XY)} \quad \text{..... eq. 1}$$

D (i, i+1) is difference between pixels of ith and (i+1)th frame.
X and Y are the dimensions of the frame.
Pi (x, y) corresponds to intensity value of pixel of ith frame at coordinates (x, y).

A cut is detected whenever the difference between the pixels of frames is greater than the threshold. The main disadvantage associated with this algorithm is that it fails when there is a large change in a small part or small change in a large part of the frame. A false cut is detected in the above cases. A small improvement over this method could be obtained by counting the number of pixels that undergo change between successive frames. If the change in the number of pixels is greater than threshold, then a cut is detected.

Shot Boundary Detection (SBD) is process of automatically detecting boundaries of shots of a video and research in this field has matured over the years. The methods used for the purpose of Shot Boundary detection have been described in [1, 2, 3]. There has been a study of seven years of TRECVid activity been presented [3]. The most predominant methods used are:

(1) Colour histogram.

(2) Edge Information.

(3) Luminance Values.

Combinations of these methods were also used and Color histogram along with edge information proved to be better in most cases. An average of 79 percent for precision and recall was obtained and no more significant improvement was achieved further. A local keypoint matching algorithm is presented to detect the shot changes using a so-called color context histogram (CCH) feature computed around Harris corner points [4, 5, 6]. Hannes, in his paper, describes another method to achieve the same and describes it as, shot boundary detection using Self Organising Maps (SOMs)[6]. A two-pass, graph-based shot boundary detection algorithm, inspired from the graph based image segmentation algorithm proposed by Felzenszwalb and Huttenlocher is used for the purpose of Shot Boundary detection[8]. Segmentation can even be achieved by constructing an undirected graph

$$G = (V, E) \text{..... eq.2}$$

from the video frames; each frame F is a vertex v and edge weights are computed as the distances between the frames,

$$w(vi, vj) = d(Fi, Fj) \text{ [7]..... eq.3}$$

Edge weights are the distances between the pixels. In the first pass of this algorithm abrupt transitions are detected, while in the second pass gradual transitions are detected.

2.2 Video Annotation

Video annotation is a marked-up comment made to information present in a video. It is also termed as metadata, tags, comment, etc added to a video [29]. It is the method by which information present in a video is described so that retrieval of the information present in video could be made possible.

The availability of semantically annotated video assets constitutes a critical prerequisite for the realisation of efficient content based retrieval systems pertaining to realistic user needs. In the early years of the research on Video Annotation, manual annotation techniques were developed. In case of manual annotation a person is required to annotate the videos or parts of the video. This is an extremely tedious and inaccurate task to achieve. Hence there is research being conducted so as to automate this task of annotation of videos.

Metadata is attached to the video for the purpose of video annotation. The metadata could be of various types as listed below.

1. **Content independent metadata**:Metadata associated with the video but not directly describing it. For example, Name of the Author, Date, Location, etc

2. **Content dependent metadata**:It refers to the low level and intermediate level features in the video. The low level features that can be used for annotation are color, texture, edge, motion, etc.

3. **Content descriptive metadata**:It refers to content semantics. It is related to the content of the video, the entities or objects, events, emotions and meaning of scenes.

2.2.1 Video Annotation Techniques

Various techniques for annotation of videos have been described in [29].

1. **Free text descriptions**: Free textual descriptions could be added to video. There is no pre-defined structure for the annotation [30]. For example, user can add description about the video while uploading a video on YouTube. Any combination of words or sentences can be used. Such type of annotation helps in accessing the video. Since no structuring exists, annotation is an easy task but, efficient retrieval techniques must be used.

2. **Based on the text in video**: The textual information that exists in images and video sequences are called collateral text. For example, the text of news, documentary

programs, movies and even newspaper film reviews [31]. Textual data is a source of highly semantic information and thus, if available, would allow the filtering and searching of video data by users in a more intuitive and natural way. Text embedded video; especially captions provide brief and important content information, such as the name of players or speakers, the title, location, date of an event, etc. [32].

3. **Based on machine learning**: From the video low-level features can be extracted. Various machine learning techniques such as support vector machine (SVM), Bayesian networks, Clustering, similarity and metric learning can be used. Different approaches for video annotation based on machine learning have been discussed [33, 34, 35].

4. **Based on Rule Learning**: Visual features can be extracted from the video. These low-level features can be used for annotation but gap exists between the information that can be extracted automatically from visual data and the interpretation that the same data has for a user in a given situation: the semantic gap. Rules are made to infer a set of high-level concepts from low-level descriptors. Jordon et al. introduced a rule-based approach for the generation of inference rules using fuzzy logic [36]. One of the proposed systems can automatically annotate the video shots based on the pre-annotated data set[37].

5. **Based on Graph**: Graph-based learning is a semi-supervised method. Graph with labeled and unlabeled vertices are used. These vertices are samples; the edges reflect the similarities between sample pairs. A function is estimated on the graph based on a label smoothness assumption. These methods have already been success Fully applied in image and video content analysis [38, 39].

6. **Based on ontology**: Explicit specification of a conceptualization can defined as Ontology. Classification of different aspects of life into hierarchical categories is achieved in Ontology. Ontology consists of entities and their relationships, which may be organized as classes and subclasses, each class may also consist of one or more instances. For example, it can be found that a room is a subclass of the class house. Jin-Woo Jeong Hyun-Ki Hong, and Dong-Ho Lee, have presented an automatic video annotation technique which makes use of ontology to facilitate video retrieval and sharing process in smart TV environment[40]. A framework for ontology enriched semantic annotation of CCTV video is proposed. Visual and text semantics are linked with appropriate keywords provided by domain experts[41]. These visual semantics are annotated by keywords of CCTV ontology.

2.3 Video Indexing

Video indexing is one of the most important parts of the content based video retrieval systems. It can be defined as the process of extracting from the video data, the temporal location of the feature and its value.

These approaches can be categorized based on the two main levels of video content: low-level (perceptual) features and high-level (semantic) annotation. The main benefits of low-level feature-based indexing techniques are:

1. They can be fully automated using feature extraction techniques, such as image and sound analysis [29].

2. Users can use similarity search using certain feature characteristics such as the shape and color of the objects on a frame or the volume of the soundtrack.

However, feature-based indexing tends to ignore the semantics contents, whereas users mostly want to search video based on the semantic rather than the low-level characteristics. There are elements beyond perceptual level, which can make feature based-indexing very tedious and inaccurate. For example, users cannot always describe the characteristics of certain objects they want to view for each query. The main benefit of high-level semantic-based indexing is the support of more natural, powerful and flexible ways of querying.

For example, users can browse a video based on the semantic hierarchy concepts like topical classification and they can search particular video according to the keywords. However, this type of indexing is often achieved through manual intervention as the process of mapping low-level features to semantic concepts is not straight forward due to semantic gaps.

Manual semantic annotation should be minimized as it can be very time-consuming, biased and incomplete. Feature-based video indexing techniques can be categorized based on the features and segments extracted.

2.3.1 Segment-based Indexing Techniques

During the process of indexing texts, a document is divided into smaller components such as sections, paragraphs, sentences, phrases, words, letters and numerals, and thereby indices can be built on these components.

Using a similar approach, video can also be decomposed into a hierarchy similar to the storyboards in filmmaking. Storyboard indexing for hierarchical video browsing is shown in fig. 2.2, which can be described as follows.

A video can contain stories of a birthday party, a vacation and a wedding. Each of the stories contains a set of scenes, for example a wedding story contains the church blessing and the wedding party scenes. Each scene is then further partitioned into shots, for instance the wedding party scene can be made up of shots of the special guests of the party and the excitement of the brides.

Each shot is comprised of a sequence of individual frames. Thus, in this indexing framework, we have defined that a frame is a single image/picture, a shot is a sequence of frames with similar characteristics, a scene is a sequence of shots that correspond to a semantic content, and a story is a sequence of scenes that reveals a single semantic story. A variation of this hierarchy is presented in which a video document usually has one or more purposes such as entertainment and information.

To achieve its purpose, a video document contains a genre of sports which contains several sub-genres such as soccer, basketball and tennis. The logical units of sports video are play and break which contain named events such as goal and foul.

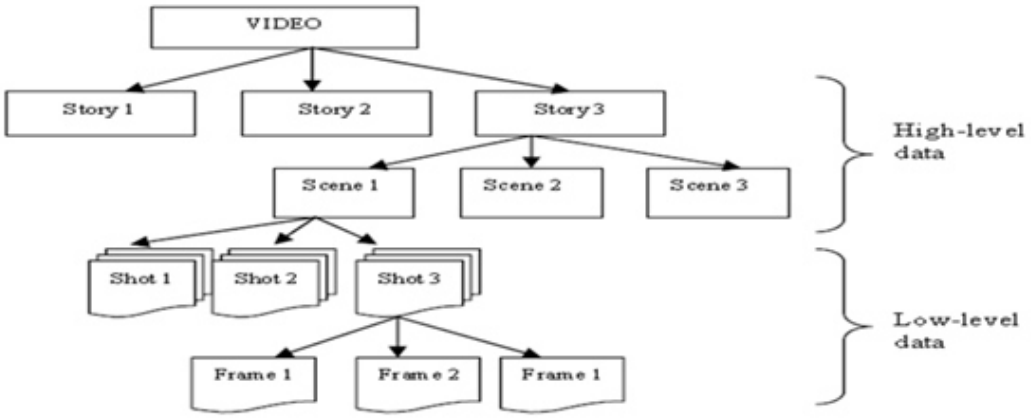

Figure 2.2: Segment-based Indexing Techniques

2.3.2 Object-based Indexing Techniques

Object-based video indexing aims to distinguish particular objects throughout video sequence to capture content changes. In particular, a video scene is defined with a complex collection of objects, the location and physical attributes of each object and the relationship between them. Objects extraction process is more complex than extracting low-level features such as color, texture and volume. However, the process on video can be considered easier as compared to on image because an object region usually moves as a whole within a sequence of video frames.

Moreover, object extraction can be easily done when video is compressed using MPEG-4 object-based coding standard [10]. An MPEG-4 video session (VS) is a group of one or more specific video objects (VOs). Each VO contains video object layers VOL(s) that consists of an ordered sequence of video object plane VOP(s). A VOP is a semantic object containing shape and motion information in a scene.. Object tracking is commonly done by detecting moving edges.

However, objects extraction is generally very difficult when objects in a frame are many, very small and move very fast thereby causing blur. For this reason, it is to be noted that objects-based indexing does not appear to be very effective in team-based sport videos such as basketball, soccer, and American football as the objects, such as players and ball, are many and move constantly. On the other hand, object-based video indexing techniques can be more effective for Sport videos that involve fewer objects such as tennis and sumo where users can query videos based on their favourite player or object movement.

2.3.3 Event-based Indexing Techniques

By tracking objects activities, events in video segments can be detected [33, 50, 51, 52]. Event based video indexing aims to detect interesting events automatically from raw video track. However, there is yet a clear definition for event itself for video indexing.

Event can be generally defined as the relations between appearing objects in a time interval that may occur before or after the other event. Event can also be defined as long-term temporal objects which are characterized by spatio - temporal features at multiple temporal scales, usually over tens or hundreds of frames as shown in fig. 2.3.

An event includes

(a) temporal textures such as flowing water: indefinite spatial and temporal type[1]

(b) activities such as person walking: temporally periodic but spatially restricted and

(c) isolated motion events such as smiling: no repeat either in space or in time.

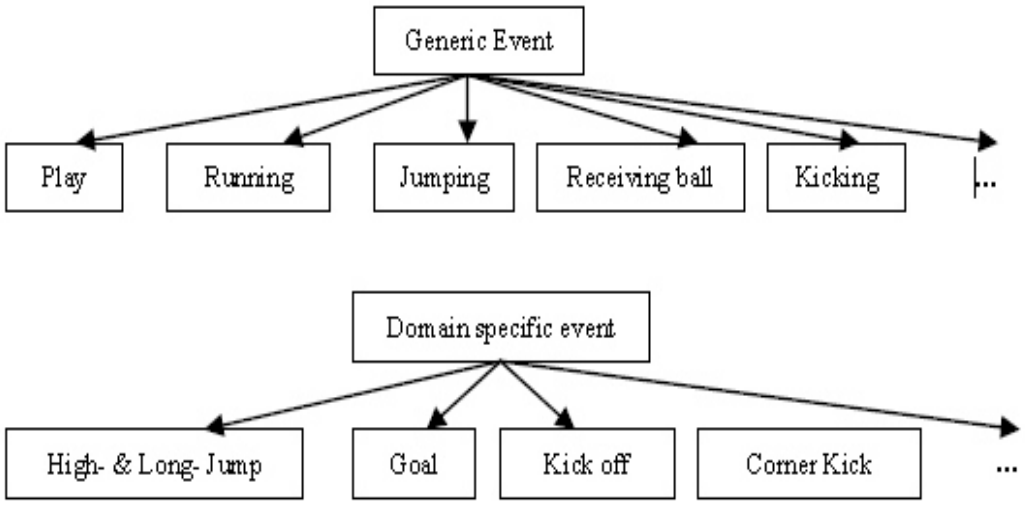

Figure 2.3: Generic and domain specific events example

Feature-based Indexing Technique	Benefits	Limitations
Segments-based (indexed by key frames)	Can be easily automated. Segments reveal temporal structure well (e.g. in hierarchy). Supports queries by image similarity. Enable intuitive browsing through visual key frames that represent the whole shot.	Key frame is often not enough to represent information in a shot. Motion and temporal information is not represented. Hard to determine the best keyframe both manually and automatically especially, for application/user specific contexts.
Object-based	By identifying objects throughout scene, and tracking the motion, content changes can be identified from a video sequence. Easier when video is compressed using MPEG-4 object based coding standard.	Object detection becomes difficult in certain situations, such as complex background, small sized objects and large number of overlapping objects such as soccer ball.
Event-based	Event is often how viewers remember video content easily Event is a feature and can be considered to be close to semantic	Event detection is generally difficult. Often has to be named using domain specific terms to make more sense to users.

Table 2.1: Comparison of various techniques

Enable intuitive browsing through visual key frames that represent the whole shot.

2.3.4 Annotation based video indexing

Text data management, such as Information Retrieval (IR) techniques have matured and successfully supported by traditional DBMSs [41, 42, 46]. Thus, another alternative for managing video is to annotate the semantics of video segments using key words or free texts. Thus, user queries can be managed using standard query language, such as SQL and browsing can be based on hierarchical topic (or subject) classification.

However, the major limitation of this approach is the fact that it would be extremely tedious and ineffective to manually annotate every segments of video. On the other hand, the process of mapping low-level video features into high-level semantic concepts is not straight forward. There are also some major drawbacks which can already be expected from Annotation-based indexing includes:

1. Keywords/free text selection is subjective and often depends on application and domain requirements.

2. An image often worth a thousand words. This means that words are often not able to fully describe a single frame therefore it is expected that words will be extremely insufficient to describe a video segment.

3. When users do not know how to explain what they want using words, it is often the case that they would like to query based on a similar image or sound. Similarly in browsing a video document, users often find that visual key frames representation are more helpful and interesting compared to pure texts.

Despite these limitations, this approach still needs to be explored as annotations can still be the closest representation of video semantic content. Moreover, a video application such as sport and news can share many keywords; therefore, a glossary of keywords can be constructed to assist a more uniform indexing and making queries easier. The followings sub-sections describe subject classification and MPEG-7 semantic graph as the examples of annotation-based indexing.

Chapter 3

IMPLEMENTATION

3.1 Introduction

The software platform used for this project is OpenCV Python. There are two major parts of the software used for the project. They are:

3.1.1 OpenCV

The major advantage of using OpenCV over conventional Image Processing softwares such as MATLAB is its ability to provide real time operation. The libraries of the OpenCV are programmed using C++ language so the execution time is very less as compared to MATLAB. This was one of the primary reasons for choosing OpenCV for the implementation of the project.

3.1.2 Python

Python is a widely used general-purpose, high-level programming language. Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

3.1.3 PyEnchant

PyEnchant is a spellchecking library for Python, based on the excellent Enchant library. It is used for checking the spellings of the query provided by the user and provide suggestions in case of misspelled words.

3.1.4 EasyGUI

EasyGUI is the module which has been used to design the user interface which is required for the project.

3.1.5 Py2exe

Py2exe is a Python Distutils extension which converts Python scripts into executable Windows programs, able to run without requiring a Python installation. This software has been used in the project because installation of python and OpenCV and other modules required for the project is a cumbersome for a user who wishes to use the software created by us in the project. Using py2exe we have created executable files which can be easily used by the user.

3.2 Structure of Implementation

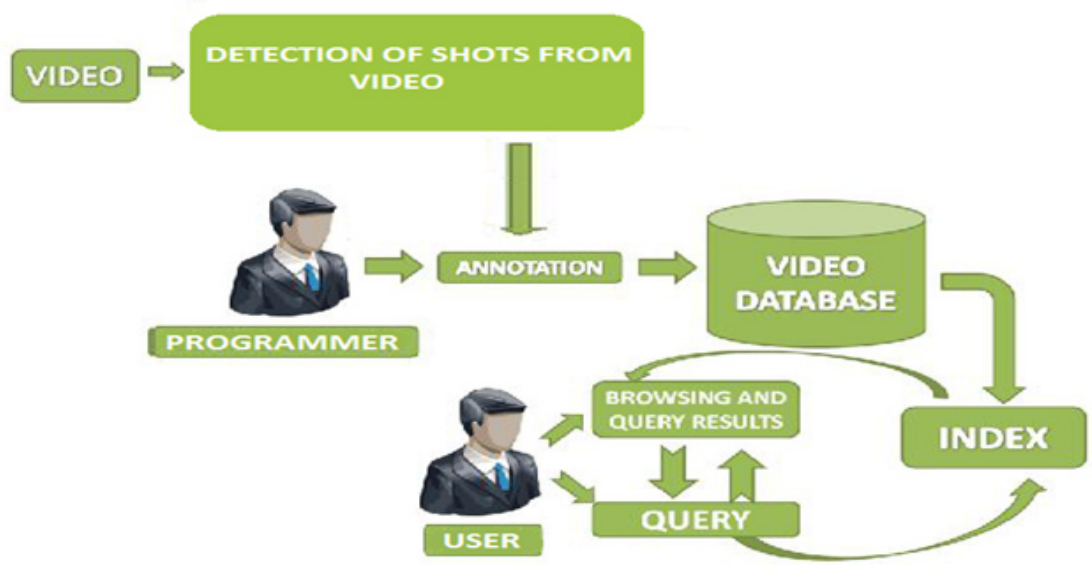

Figure 3.1: Basic block diagram of Content Based Video Retrieval.

The basic block diagram of the Content Based Video Retrieval system is as shown in fig. 3.1. For the purpose of retrieving the content from the video, it is required to annotate the video. For the purpose of annotating the video, firstly the shots present in the video are detected. Annotation represents the information present in a particular shot. These annotations are stored in a database.

User is asked to enter a Query. A query could be any information relating to the video which the user wants to seek. With the help of Video Indexing technique, relevant content is retrieved by comparing the entered query with the stored annotations.

3.2.1 Video Annotation

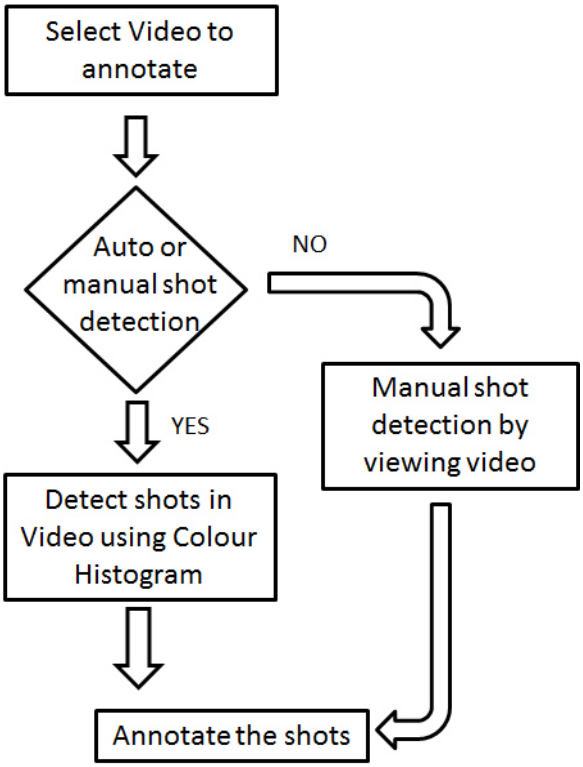

Figure 3.2: Flow chart for Annotation of video.

As shown in fig. 3.2, the programmer selects a video to be annotated. The prior step to annotate any video is to detect the shots present in it and this can be achieved using either manual or automatic shot detection. The former can be performed using color histogram and the later can be latter can be carried out manually by the programmer with a hit of specific key.

Selection of Video

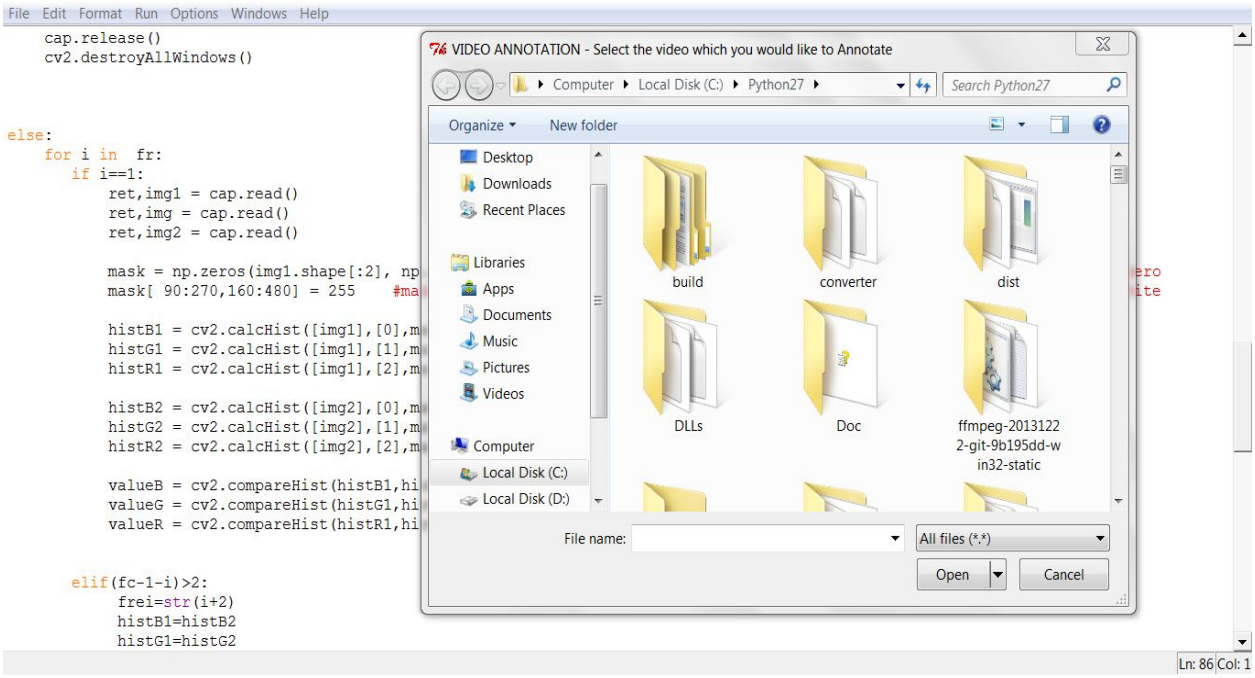

Figure 3.3: Window for selecting videos.

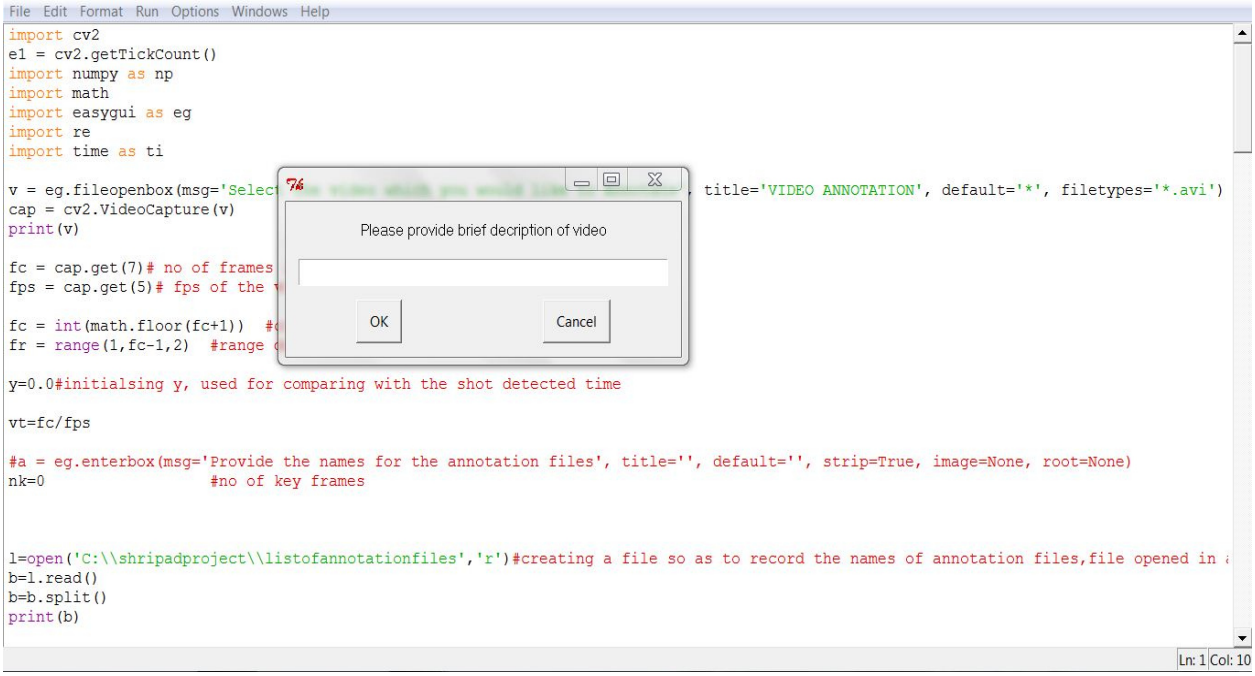

Figure 3.4: Window for providing short description of the video

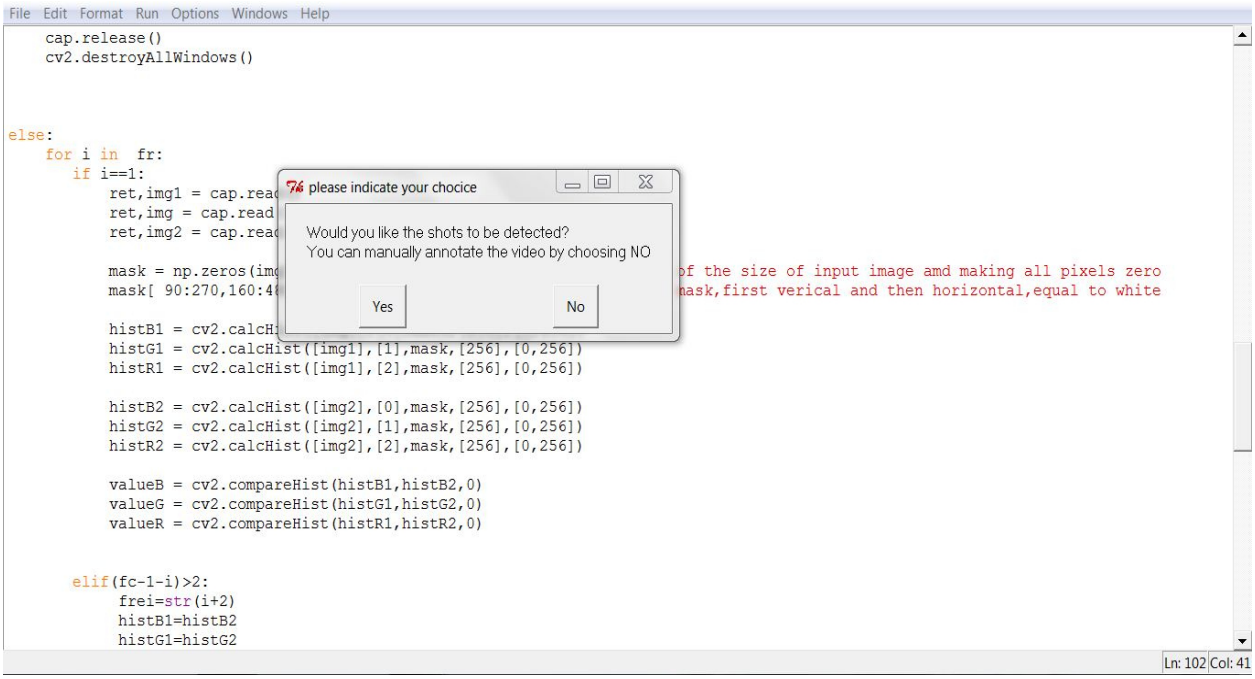

Figure 3.5: Window for selecting automatic or manual detection of shots.

After selecting the video by the programmer to annotate as shown in fig. 3.3, next, a dialogue box appears for providing short description to the video as shown in fig. 3.4.

Detecting and Annotating shots in a video
Depending on the choice entered by user, shot detection can be either automatic or manual as shown in fig. 3.5.
(i) Automatic shot detection.
Each frame is taken and the Colour Histogram of each frame is calculated. For better and accurate results, colour histogram for all the three different types of pixels, Red, Green and Blue (R, G, B) is calculated.

Colour Histogram is a representation of the distribution of colours in an image. It is plot of intensity of pixels versus number of pixels corresponding to those intensities.

The histograms of two consecutive frames are calculated and then the correlation of the two is found. There can be two cases.

1.Two frames are completely dissimilar.
2.Two frames are similar.
Case (i) As seen from fig. 3.6 and fig. 3.7, the histogram of two dissimilar frame is completely different. When the two histograms are correlated, the value of their correlation is very much less than unity.

Dissimilar frames

Figure 3.6: 1.Desert(847 X 1017 pixels) 2. Tulips (847 X 1017 pixels).

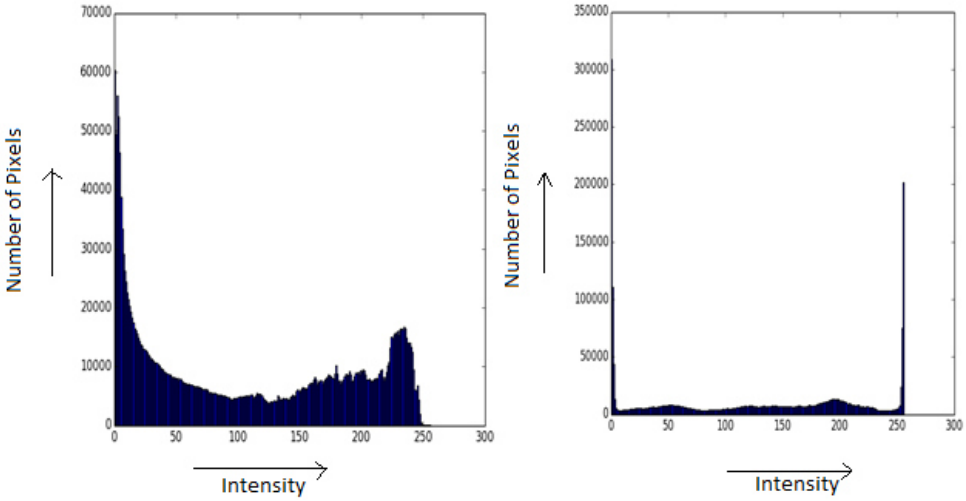

Figure 3.7: Histogram of two Dissimilar frames.

Case(ii) As seen from fig. 3.8 and fig. 3.9, If the two frames are similar, then their histograms are almost identical, so the correlation is almost equal to 1.

Similar frames

Figure 3.8: 1.First frame(847 X 1017 pixels) 2. Second frame(847 X 1017 pixels).

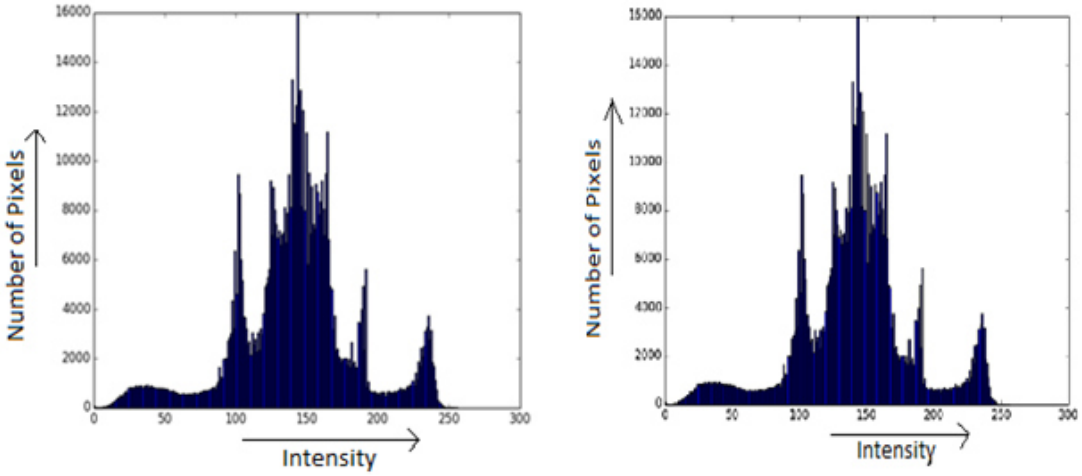

Figure 3.9: Histogram of two similar frames.

$$d(H1, H2) = \frac{\sum_1 (H_1(I) - \bar{H}_1)(H_2(I) - \bar{H}_2)}{\sqrt{\sum_1 (H_1(I) - \bar{H}_1)^2 \sum_1 (H_2(I) - \bar{H}_2)^2}} \quad \text{.....eq. 5}$$

$$\text{where } \bar{H}_k = \frac{1}{N} \sum H_k(J)$$

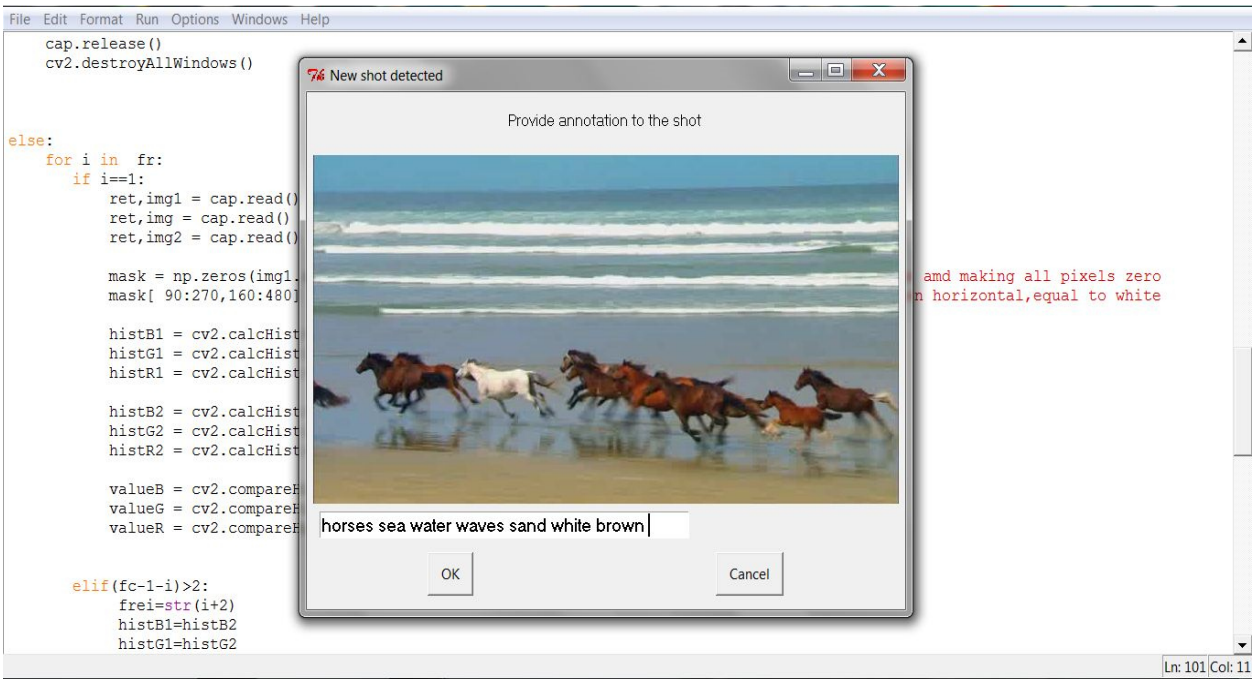

Figure 3.10: Window for providing annotations during automatic shot detection.

The programmer is informed of a new shot automatically detected using a dailogue box as shown in fig. 3.10.

(ii) Manual Shot detection.
Incase if the user choses manual annotation, the user will be able to manually pause a playing video whenever he wishes to and can provide the annotations at that instant of time as seen in fig. 3.11.

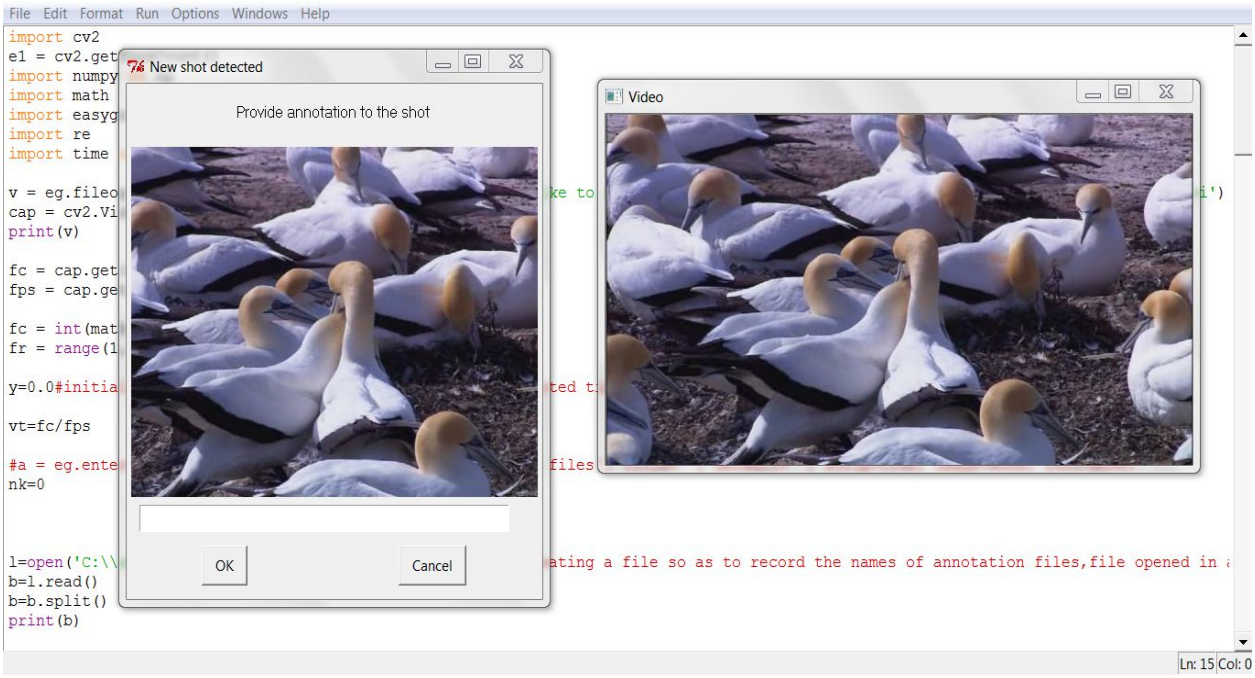

Figure 3.11: Window for providing annotation during manual shot detection

When two frames are similar, their histograms are similar and the correlation is much less than unity.

When two dissimilar histograms are correlated, the correlation is found very much less than unity and hence a shot is detected. When a new shot is detected, the counter storing the number of shots is incremented by 1. This is carried on till the last two frame are processed. For increasing the speed of processing, histogram of not consecutive, but alternate (odd) frames are calculated and correlated.

When a new shot is detected, the programmer is asked to give annotations to each of the newly detected shots. At the same time, the time of beginning of the shot is recorded and stored in time file. The counter is incremented. All the annotations are then stored in annotation file and the programmer is notified about the completion of the process through a message box as shown in fig. 3.12.

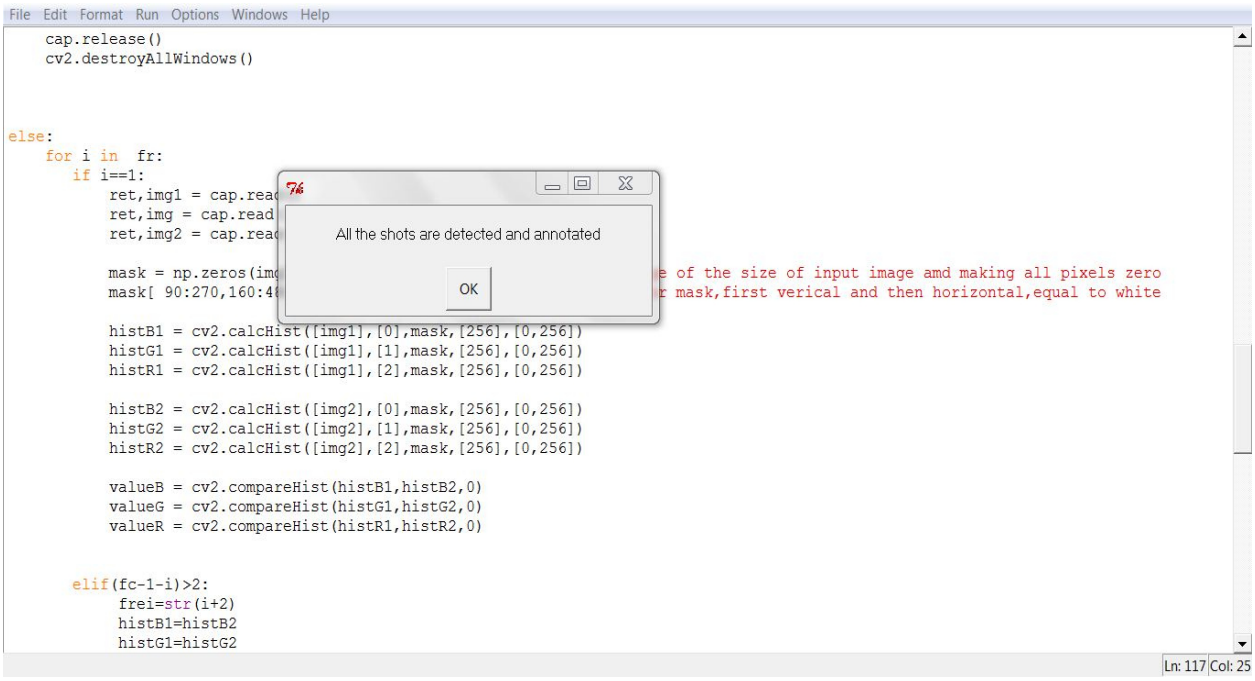

Figure 3.12: Window to inform that all the shots are detected and annotated.

Chapter 4

RESULTS AND ANALYSIS

4.1 Analysis of shot detection

Various video files were taken and the code was implemented on them. The results for the first four videos is as shown. Each frame is extracted and depending on the number of frames extracted and the length of the video, the fps of the video can be calculated. The various features of a frame such as the resolution can also be obtained.

1. **Name of video**: chennai.avi
   Resolution: 640 x 360
   Length of the video: 2min 1sec (121 sec)

Trial No.	Time for detection of all shots(in sec)
1	13.3762
2	13.2
3	14.577

Table 4.1: Average time to detect shots in Chennai.avi

Average time to detect shots: 13.6977sec which is 11.32% of total video time.

2. **Name of video**: wildlife.avi
   Resolution: 640 x 360
   Length of the video: 30sec

Trial No.	Time for detection of all shots(in sec)
1	3.9966
2	4.1452
3	4.0938

Table 4.2: Average time to detect shots in wildlife.avi

Average time to detect shots: 4.0786sec which is 13.56% of total video time.

3. **Name of video**: dip.avi
   Resolution: 640 x 360
   Length of the video: 54min3sec(3243sec)

Trial No.	Time for detection of all shots(in sec)
1	320
2	319

Table 4.3: Average time to detect shots in dip.avi

Average time to detect shots: 319sec which is 9.86% of total video time.

4. **Name of video**: quadcopter.avi
   Resolution: 640 x 360
   Length of the video: 175ec

Trial No.	Time for detection of all shots(in sec)
1	24.53
2	22.34
3	20.34

Table 4.4: Average time to detect shots in quadcopter.avi

Average time to detect shots: 20.43sec which is 11.4% of total video time.

After implementing algorithm for the above four videos, average time to detect shots is 11.535% of total video time.

4.2 Retrieval of content from the videos

A Graphical User Interface is created (GUI), which asks the user to enter the query. The query is read and ten stored. The stored query is then compared with the stored annotations. Then, further the time corresponding to the matched annotations is retrieved and video starts from that particular time.

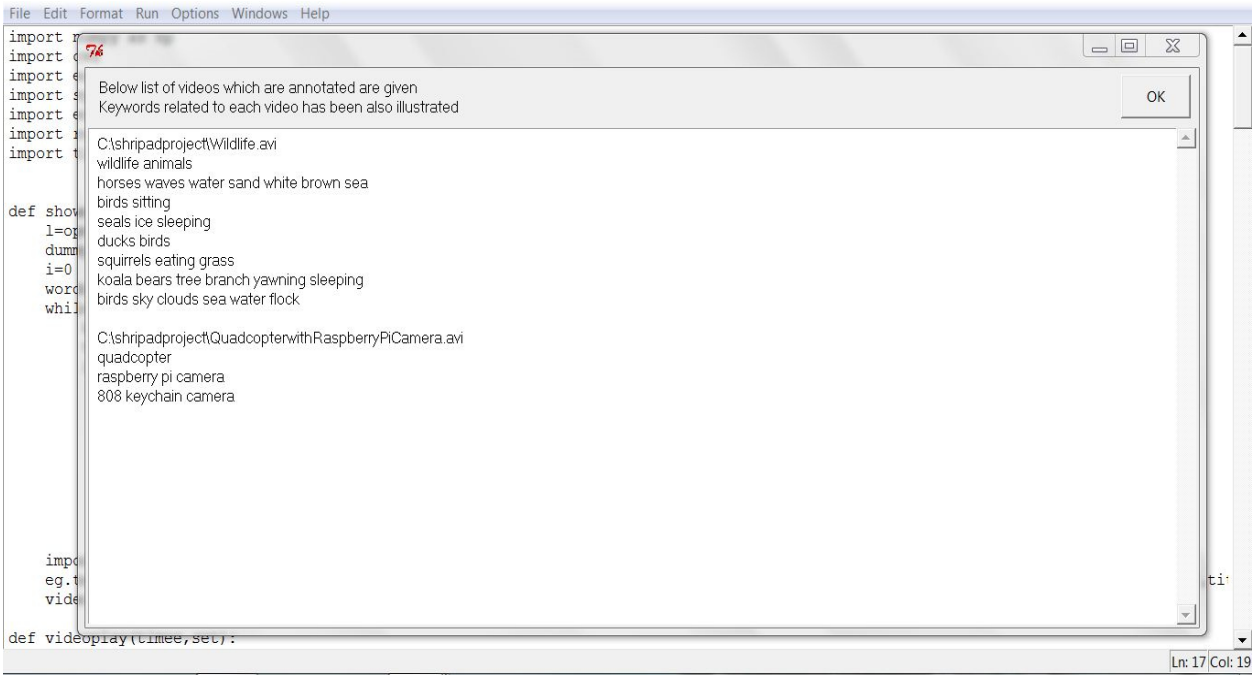

Figure 4.1: Window displaying existing keywords to the user.

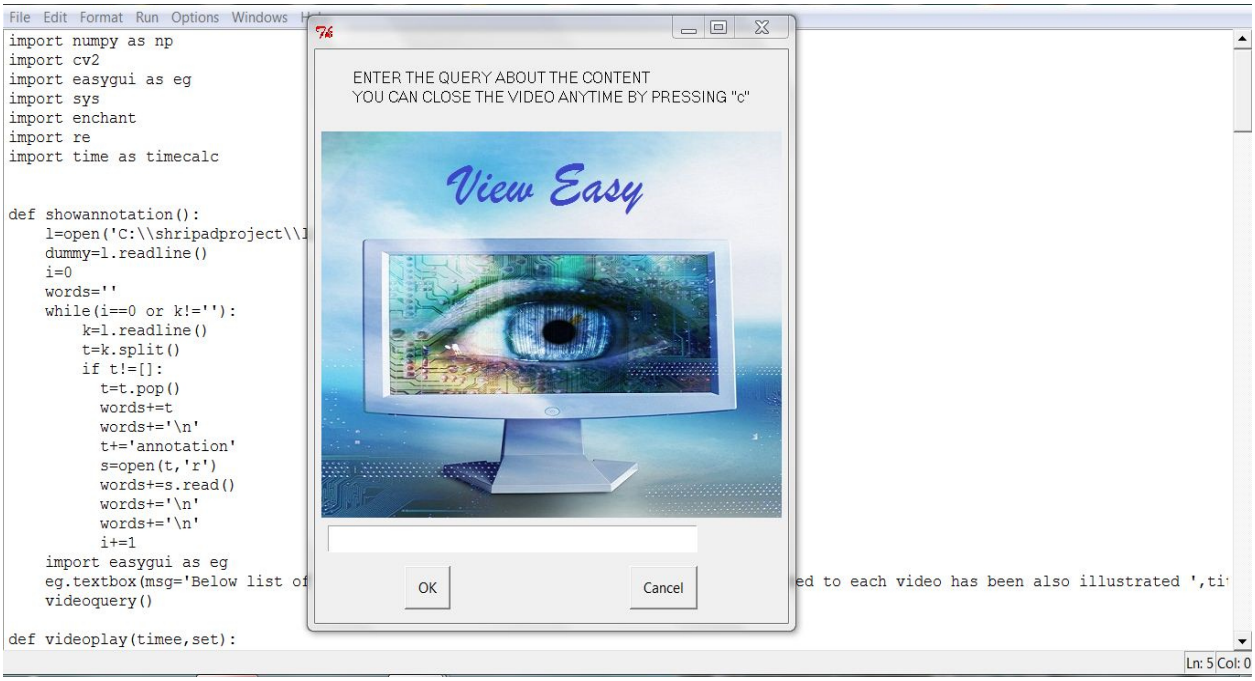

Figure 4.2: Window displaying GUI.

In Fig.4.1, the keywords which are present as annotations to all the present videos are displayed. By referring to these keywords, the user can input his query.

In Fig.4.2, a GUI is shown in which the user can enter the query for finding the intended video content present within the video. There can be three cases.

1.No relevant Content is present.
2.Content is present, but user makes spelling errors in the query.
3.Query typed is correct, but there are multiple results.

1.No relevant contents present

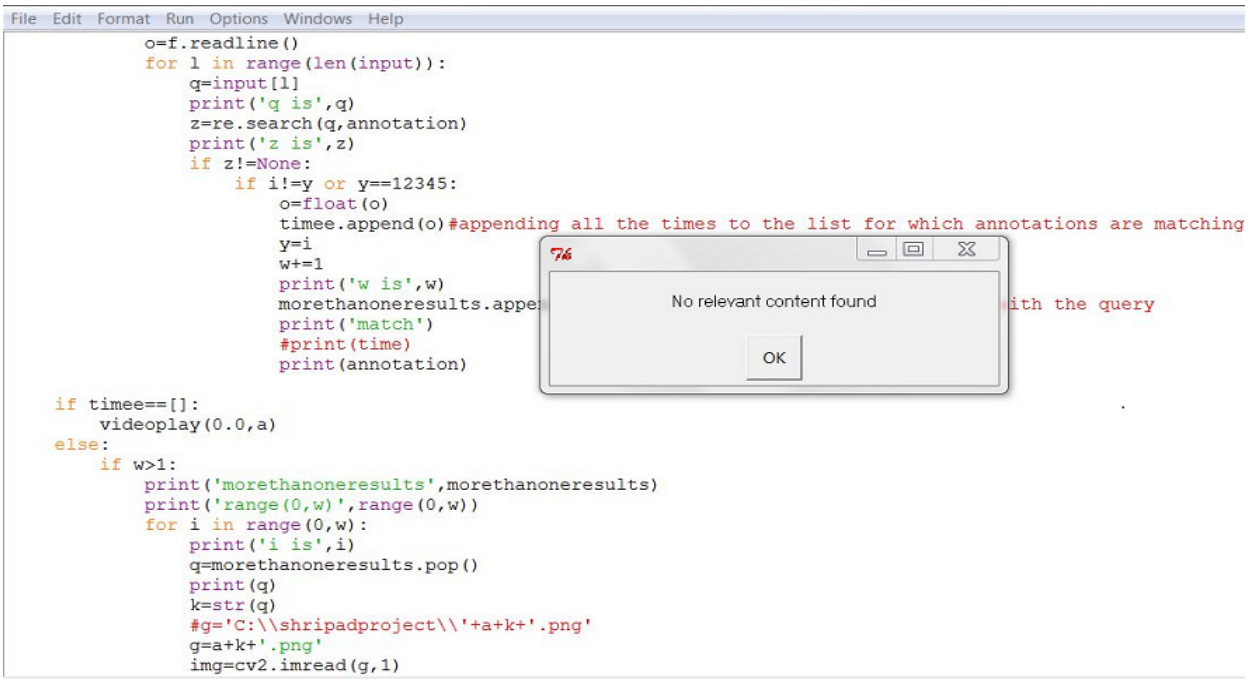

Figure 4.3: Window displaying no relevant content found.

When user types in a query which is not present in the video, the program replies to the user with message No relevant contents found as shown in fig. 4.3.

2.Content is present, but user makes spelling errors in the query

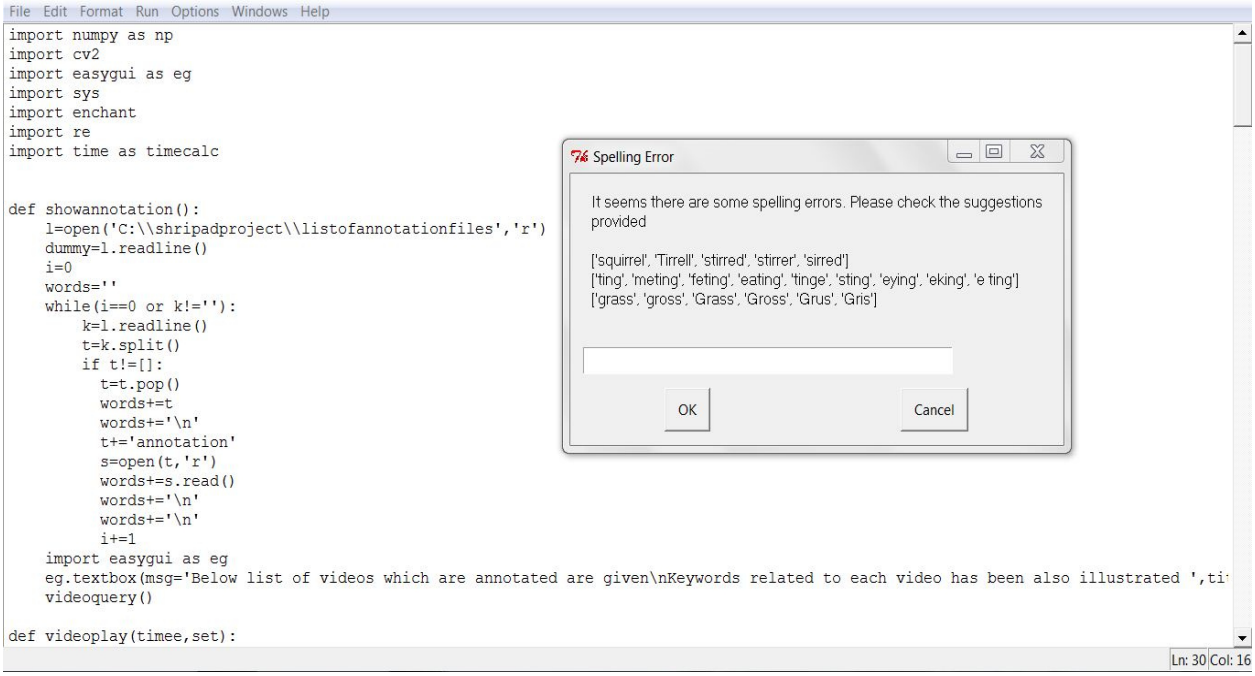

Figure 4.4: Window displaying spelling errors and providing suggestion.

If the user types in a query with one or more errors, the program compares the typed query with the default dictionary present in OpenCV Python and informs the user of spelling error and provides the user with all the possible words, those are closest to the typed query as shown in fig. 4.4.

3. Query typed is correct and relevant
When query typed is correct and relevant, there can arise two situations.

a. Single Result present.
In this case, the program will respond to the query and start the video from the starting of the frame corresponding to the shot.

b. Multiple results present.

(i) Multiple videos present
The user enters a query in the GUI. In case if the entered query matches with the stored annotation pertaining to two different existing videos, then the user will be intimated and will be provided with a tab to choose between the two different videos as seen in fig. 4.5.

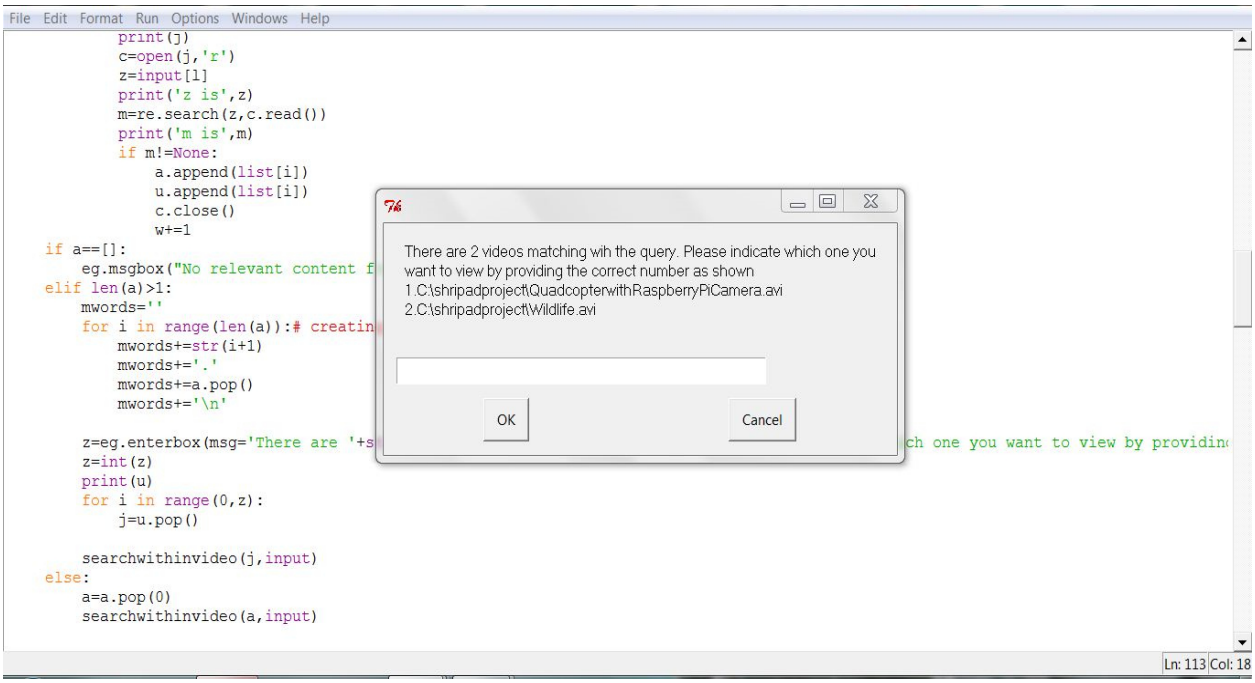

Figure 4.5: Multiple video results for a query.

(ii) Multiple results within a video

In case of multiple results, the program will show the user all the possible windows of the shots which the user might be willing to view. The user is further provided with a tab to enter that window or shot which he wishes to view and the video is played from that particular frame as shown in fig. 4.6.

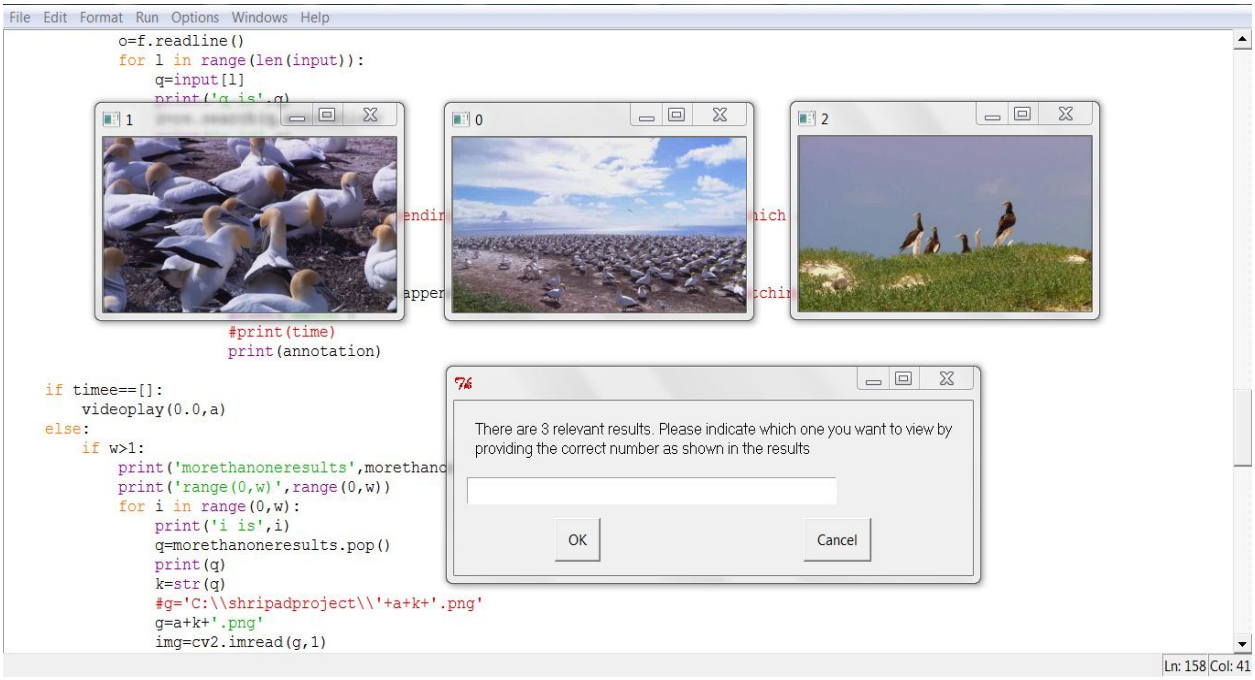

Figure 4.6: Multiple results within a video for a query.

Chapter 5

CONCLUSION

In this project, we have reviewed recent developments in content-based video indexing and retrieval and have tried to implement our own framework for efficient video retrieval. In this work, we have studied various techniques used in content based video retrieval systems. Some of the most efficient and popular tasks used in CBVR include, video segmentation, video annotation, video indexing and creation of a video database.

OpenCV computer vision library has been used for the implementation of the image processing part of the framework. Python has been used for achieving the task of video annotation and video indexing.

After considering various techniques for video segmentation, we can conclude that color histogram is one of the efficient methods to detect shots. Manual annotation is used to describe the content present in the detected shots.

Finally we arrive at a conclusion that, the recent work done on segmentation and database has matured, but the major challenge and research has to be performed in bridging the semantic gap through efficient annotation and indexing techniques.
After implementing the algorithm for a set four videos, average time to detect shots is 11.535% of total video time.

5.1 Future Work

In this project we have implemented a novel method of structuring a video with the help of Video Segmentation and Annotation.

There could be further advancement to this approach as follows:-

1. In this project, video annotation was performed manually. Attempts can be made to automate the process of Video Annotation with the help of Optical Character Recognition (O.C.R.).The audio present in the video can also be used for automatic annotation by parallel use of speech to text conversion softwares.

2. The main focus of this project has been on the visual content present in video(s). In future, a frame could be designed in order to retrieve the audio content along the visual content and both can be played simultaneously.

3. The framework and the software created through this project can be used onlyfor offline videos. The same can be extended on the platforms of existing video sharing websites and can be used for online videos.

APPENDIX

A. Glossary

1. **AVI**
Audio Video Interleave.AVI can play in main stream media player such as Windows Media Player.It is a derivative of the Resource Innterchange File Format (RIFF), which divides a file's data into blocks or chunks. By using RIFF format, the audio - visual contained in the blocks can be encoded or decoded by software called codec.

2. **JPEG**
Joint Photographic Experts Group.JPEG is file format used for pictures and images. It is used by digtal cameras and for storing and transfering images. It is a universal format. JPEG files can be easily opened and processed.

3. **OpenCV**
OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day. Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc.

4. **PyEnchant**
PyEnchant is a spellchecking library for Python, based on the excellent Enchant library. It combines all the functionality of the underlying Enchant library with the flexibility of Python and a nice "Pythonic" object-oriented interface. It also aims to provide some higher-level functionality than is available in the C API. In addition to the above software, Windows video codec pack needs to be installed, so that videos could be played with the help of OpenCV Python. The software installed is windows 7.codec.pack.v4.0.8.setup.exe. It has been downloaded from the site: download.cnet.com.

5. **Py2exe**
Py2exe is a Python Distutils extension which converts Python scripts into executable Windows programs, able to run without requiring a Python installation. This software has been used in the project because installation of python and OpenCV and other modules required for the project is a cumbersome for a user who wishes to use the software created by us in the project. Using py2exe we have created executable files which can be easily used by the user. The software is downloaded from www.py2exe.org and has to be copied to the folder C:/Python27/lib/site - packages.

Installation of the Software

Python 2.7 and OpenCV 2.4.5 has been installed and used for the project. The installation procedure mentioned here is for Windows platform.

Below Python packages are to be downloaded and installed to their default locations.
1. Python-2.7.x.(http://python.org/ftp/python/2.7.5/python-2.7.5.msi)

2. Numpy.(http://sourceforge.net/)

3. Matplotlib.(http://sourceforge.net/)

4. Install all packages into their default locations.

5. Python will be installed to C:/Python27/.

6. After installation, open Python IDLE.

7. Enter import numpy and make sure Numpy is working fine.

8. Download latest OpenCV release from sourceforge site and extract it.

9. Go to opencv/build/python/2.7 folder.

10. Copy cv2.pyd to C:/Python27/lib/site-packages.

11. Open Python IDLE and type following codes in Python terminal.

```
import cv2
```

If you don't get any errors, OpenCV-Python is installed successfully.

In addition to the above installation, two more modules are used in the implementation of the project.

B. Research Publications

A Paper titled **Overview of Existing Content Based Video Retrieval sys tems,**published in the International Journal for Advanced Engineering and Global Technology, Volume 2, Issue 2, February 2014, ISSN 2309-4893.

Presented and published a paper titled **Content Based Video Information Retrieval using Color Histogram**, at **National Conference on Electronics Technologies (NCET), 2014**, held at Goa College of Engineering

C. Achievements

At a technical festival, GENESIS V 5.0, organised by Computer Science Department,at Gogte Institute of Technology, Belgaum - Karnataka, won **First Place** for Project Demo on 4th of April, 2014.

Secured **First Place** for Paper Presentation at XPLODE' 14, organised by Mining Department, at Goa College of Engineering on 28th of March, 2014.

Secured **Second Place** for Paper Presentation at REVAMP' 14, at Reva Institute of Technology and Management, Bangaluru, on 3rd of April, 2014.

Secured **Second Place** for Paper Presentation at SPARX 7, organised by Department of Electrical and Electronics Engineering, at Goa College of Engineering, on 24th of March, 2014.

Secured **Third Place** for Project Presentation at REVAMP' 14, at Reva Institute of Technology and Management, Bangaluru, on 3rd of April, 2014.

D. Codes

Code shown below is used for detection of the shots and annotation of the shots. All the videos have to be annotatedfor further retrieval of content.

```
import cv2# import the OpenCV module for image processing operations

importnumpy as np# import Numpy for array operations

import math# importmath module for mathematical operations

importeasygui as eg# EasyGuI module for designing GUI

import re# Module used for regular expression matching operations

import enchant# Module used for spell check and correction


def spellcheck(input):# Function for spell checking and correction

        x = input.split()# split the sentence into words

xwords=''# create an empty string

        s=0

        d = enchant.Dict("en_US")# object to access the enchantdictionary

for i in range(len(x)):# for all the words present in input string

            y = d.check(x[i])# check if spelling of the word is correct

if y == False and (d.suggest(x[i])!=[]):

# if there isspelling mistake present and if there aresuggestions for the mistake

                n=str(d.suggest(x[i]))# suggestions provided for the wrong words

xwords+=n#add the suggested words to empty string

                s=1#to indicate that wrong word has suggestions

xwords+='\n'# create a new line
```

```
if y == False:# if there is spelling mistake
if s==1:#if there are suggestions available
                    u='It seems there are some spelling errors. Please
check the suggestions provided\n\n'
else:# if there are no suggestions
                    u='It seems there are some spelling errors.\n\n'
xwords = str(xwords)# convert list to string
input = eg.enterbox(msg=u+xwords, title='Spelling
Error', default='', strip=True, image=None,
root=None)
# A dialog box informs the user about spelling mistakes, provides suggestions
for misspelled words and asks the user to input correct words
spellcheck(input)# function  call for repeatedly checking for spelling errors
else:# if there are no spelling errors
return(input)# return the input sentence


v = eg.fileopenbox(msg='Select the video which you would like to
Annotate', title='VIDEO ANNOTATION', default='*', filetypes='*.avi')
# A dialog box asks the user to select the video to be annotated
cap = cv2.VideoCapture(v)# creating videocapture object for the selected video
fc = cap.get(7)# fc is equal to number of frames in the video
fps = cap.get(5)# fps of the video
```

```
fc = int(math.floor(fc+1)) #Converting float to integer since range does not accept float

fr = range(1,fc-1,2) #Creating an array of odd numbers till fc-1

y=0.0#Initializing y, used for comparing with the shot detected time

vt=fc/fps# Total length of the video in seconds


l=open('C:\\shripadproject\\listofannotationfiles','r')

#creating a file so as to record the names of videos which have been annotated,opened in read mode so as to read the existing content if any

b=l.read()# read all the content content present in the file

b=b.split() # separate the names of all the videos


c=0

for i in range(len(b)):

if v==b[i]:# Checking whether the selected video has been already  annotated

        c=1# c=1 if video is already annotated

l.close()# close the read file

l=open('C:\\shripadproject\\listofannotationfiles','a+')

# Opening the file in append mode to record the name of video to be annotated

if c==0:# if the selected video has not been annotated previously

l.write('\n')# Move to a new line on the file

l.write(v)# Write the name of the video in the file

l.close()# Close the file
```

```
Time=v+'time'# Location of the time file

Annotation =v+'annotation'# Location of the annotation file

f = open(Time, 'w')   # Open the time file recording annotation time

g = open(Annotation, 'w') #Open the annotation file recording annotation

des = eg.enterbox(msg='Please provide brief description of video',
title='', default='', strip=True, image=None, root=None)

# A dialog box asks the user to provide brief description of video

f.write(des)#Writing brief description of the video in time file

f.write('\n')# Move to a new line

g.write(des)#Writing brief description of the video in annotation file

g.write('\n')# Move to new line


b=eg.ynbox(msg='Would you like the shots to be detected?\n You can
manually annotate the video by choosing NO', title='please indicate
your choice')

# Dialog box asks the user if shot detection has to be done automatically or manually by the user


if not b:#if the user wants to manually detect the  shots

frameno=0#Initializing the counter to count the frames

nk=0#Initialization of number of key frames
```

```
while(cap.isOpened()):# Checking whether video can be opened
ret,frame = cap.read()# read the frame of video
```
sequentially

```
cv2.imshow('Video',frame)# Display the frame
frameno=frameno+1# Increment the frame counter
if cv2.waitKey(int(fps))& 0xFF == ord('p'):
```
Wait for the time corresponding to frame rate of the video after every frame and stop the video if key ‘p’ is pressed. User can annotate the required part of video by pausing the video by pressing key’p’.

```
ifnk>0:# if not the first key frame
f.write('\n')# Move to new line
g.write('\n')# Move to new line

nk=nk+1# Increment key frame counter
nk=str(nk)# Convert integer to string
                                    z=v+nk+'.png'# location for storing the key frame
nk=int(nk)# convert string to integer
cv2.imwrite(z,frame)# Save the key frame
```

```
annotate = eg.enterbox(msg='Provide
annotation to the shot ', title='New shot
detected ', default='', strip=True,
image=z, root=None)
```
Dialog box asks the user to annotate the displayed keyframe

```
spellcheck(annotate)
```
check the spelling of the annotation provided and correct it by calling spellcheck function

```
                                x=str(frameno/fps)
```
#Time corresponding to the annotation

```
f.write(x)
```
Write the time in Time file

```
g.write(annotate)
```
Store annotation in annotation file

```
ifframeno==fc-1:
```
if the last frame is reached

```
break
```
Stop the video

```
ti.sleep(0.5)
```
Delay of 0.5 seconds

```
cap.release()
```
#Release the video capture object

```
cv2.destroyAllWindows()
```
Close the video display window

```
else:# If shots are to be detected automatically
for i in  fr:
if i==1:# If the first frame of video
ret,img1 = cap.read()# Store first frame in img1
ret,img = cap.read()# Dummy reading
ret,img2 = cap.read()# Store Third frame in img2


mask = np.zeros(img1.shape[:2], np.uint8)
#making mask image of the size of input image img1
mask[ 90:270,160:480] = 255
# The central part of the frame of resolution 640x360 is
kept and rest is masked


            histB1 = cv2.calcHist([img1],[0],mask,[256],[0,256])
            histG1 = cv2.calcHist([img1],[1],mask,[256],[0,256])
            histR1 = cv2.calcHist([img1],[2],mask,[256],[0,256])
# Histogram computation of img1 for three color component Blue Green and Red


histB2 = cv2.calcHist([img2],[0],mask,[256],[0,256])
            histG2 = cv2.calcHist([img2],[1],mask,[256],[0,256])
            histR2 = cv2.calcHist([img2],[2],mask,[256],[0,256])
# Histogram computation of img2 for three color components Blue Green and Red
```

```
valueB = cv2.compareHist(histB1,histB2,0)
valueG = cv2.compareHist(histG1,histG2,0)
valueR = cv2.compareHist(histR1,histR2,0)
```
#comparing the histogram of img1 and img2 with theirrespective color components.
The value returned is a correlation between the two histograms. The maximum value is equal to one and possible if same image is compared with itself.

```
elif(fc-1-i)>2:# if not the first frame and not second last or last frame
            histB1=histB2
            histG1=histG2
            histR1=histR2
```
histogram of first image is same as histogram of secondimage in the previous iteration of the for loop

```
ret,img = cap.read()# dummy reading
ret,img2 = cap.read()

            histB2 = cv2.calcHist([img2],[0],mask,[256],[0,256])
            histG2 = cv2.calcHist([img2],[1],mask,[256],[0,256])
            histR2 = cv2.calcHist([img2],[2],mask,[256],[0,256])




valueB = cv2.compareHist(histB1,histB2,0)
```

```
valueG = cv2.compareHist(histG1,histG2,0)
valueR = cv2.compareHist(histR1,histR2,0)

value = (valueB*valueB+valueG*valueG+valueR*valueR)/3
```
Computation of average value of histogram correlation betweenthe two images

```
frei=float(i+2)
        x=(frei)/fps
```
Calculating the time corresponding to a particular frame

```
if value < 0.64:
```
#Detect a new shot if histogram correlation value computed is less than 80%. Square of correlation is 64%

```
if (x-y)>1.0 or x<1.0:
```
#Iftime between consecutive detected shots shouldbe greater than one second or the video time is less than one second

```
if (vt-x)>1.0:
```
No shot detection within last one second of video

```
                                y=x
                                x=str(x)
```

```
ifnk>0: #if the shot detected is not the first
f.write('\n')
g.write('\n')

nk=nk+1# Increment the key frame counter
nk=str(nk)# convert integer to string
                                    z=v+nk+'.png'# Location to store the key frame
cv2.imwrite(z,img2)# Save the key frame
nk=int(nk)# convert string to integer

annotate = eg.enterbox(msg='Provide
annotation to the shot ', title='New shot
detected ', default='', strip=True,
image=z, root=None)
 # Dialog box asks the user to annotate detected shot and displays
the key frame
spellcheck(annotate)
# spellcheck and correction of the annotation
f.write(x)# write the time in time file
g.write(annotate)# write annotation in annotation file
cap.release()# Release the videocapture object
f.close()# Close the time file
g.close()# Close the annotation file


eg.msgbox("All the shots are detected and annotated")
```

A dialog box displays message that all the shots are annotated

```
n=open(v+'Number', 'w')# File created in write mode to store the no. of  keyframes

k=str(nk+1)# convert integer to string so that it can be written to file

n.write(nk)# write to the file

n.close()# close the file
```

Following Code is for the retrieval of the content from the video. The user can search for the required content by providing a query and then correct video content is played if a match is found.

```
import cv2# import the OpenCV module for image processing operations
importnumpy as np# import Numpy for array operations
import math# importmath module for mathematical operations
importeasygui as eg# EasyGuI module for designing GUI
import re # Module used for regular expression matching operations
import enchant# Module used for spell check and correction

defshowannotation():# Display Keywords related to already annotated videos to the user

l=open('C:\\shripadproject\\listofannotationfiles','r')
# Open the listofannotations file to obtain  list of videos annotated
dummy=l.readline()# Dummy reading
    i=0
words=''# Initialize empty string
while(i==0 or k!=''):# loop until all the lines in the file are read
        k=l.readline()# read a line from the file
        t=k.split()# split the line into words
```

```
if t!=[]:# If the read line is not empty
          t=t.pop()
words+=t
words+='\n'
t+='annotation'#location of the annotation file
          s=open(t,'r')# opening annotation files of all the annotated videos
words+=s.read()# read the respective annotation file and concatenate it to words
words+='\n'
words+='\n'
          i+=1


importeasygui as eg# Importing EasyGuI module for creating GUI


eg.textbox(msg='Below list of videos which are annotated are
given\nKeywords related to each video has been also illustrated
',title='',text=words,codebox=0)
# Display The keywords related to the videos which user can use while typing the query


showannotation()# Call the videoquery() function
```

```
defvideoplay(timee,set):
```

function used to play the video, has two parameters

1. Location of the video 2.Time from where video has to played

```
import math
import time as ti
```

importing the math and time module

```
    t=float(timee)
```

convert time from string to float

```
set=set.split()
```

to get rid of '\n' in the location of video

```
set=set.pop()
```

Pop the location of video from the list

```
cap = cv2.VideoCapture(set)
```

Create videocapture object for the video

```
fc = cap.get(7)
```

No. of frames in the video

```
fc = int(math.floor(fc+1))
```

#converting float to integer

```
fps = cap.get(5)
```

FPS of the video

```
cap.set(0,t*1000)
```

Set the time, from where the video has to be played. Time isspecified in milliseconds hence multiplied by 1000

```
    f= (t/fps)*fc
```

Frame no. corresponding to the time from where video has to played

```
    f=int(f)# convert from float to integer
framespersec =int(fps)# convert from float to integer
while(cap.isOpened()):# Loop if video can be opened
ret,frame = cap.read()# Read the frames sequentially
cv2.imshow('Video',frame)# Display the frame
                              f=f+1# Increment frame counter

if cv2.waitKey(framespersec)& 0xFF == ord('c')
or f == fc-1 :
```

Wait for time equal to FPS of video. Stop if the last frame or if key

'c' is pressed

```
Break# close the video
ti.sleep(0.5)# Delay of 0.5 seconds
cap.release()# Release videocapture object
cv2.destroyAllWindows()# Close the display window

    u=eg.ynbox(msg='Would like to continue searching for the
videos???',title='Please indicate your choice')
```

Ask the user if he/she wishes to continue searching for videos or exit

```
if u:# If user wishes to continue searching for videos
```

```
showannotation()# Call the showannotation() function



defvideoindexing(input):
```

Function to find out which video matches with the user queryand ‘input’is the query provided by user

```
import math
import time as ti
import re


input=input.split()# split the user query into words
    a=[]# Initialize an empty list
    u=[]# Initialize an empty list


    l=open('C:\\shripadproject\\listofannotationfiles','a+')
list=l.read()
list=list.split()
```

Read the location of all the videos annotated and separate each one

```
    w=0# Initialize the counter to count no. of matches


for i in range(len(list)):
```

to find out the videos which match with the query

```
for l in range(len(input)):#for word wise matching of query
            j=list[i]+'annotation'# Location of annotation file
            c=open(j,'r')# Open each annotation file in read mode
            z=input[l]
            m=re.search(z,c.read())
```

Match the user query with the annotation

```
if m!=None:# if a match is found
a.append(list[i])
```

Append the location of matching videos to the list

```
u.append(list[i])
```

Append the location of matching videos to the list

```
c.close()# Close the annotation file
w+=1# Increment video match counter

if a==[]:# if No match Found
eg.msgbox("No relevant content found")# Display the message to user

eliflen(a)>1:# If more than one videos are matching with the query
mwords=''
```

```
for i in range(len(a)):# creating a string to display the list of matching videos
mwords+=str(i+1)# Add No. corresponding to each video
mwords+='.'
mwords+=a.pop()
mwords+='\n'

        z=eg.enterbox(msg='There are '+str(w)+' videos matching with
the query. Please indicate which one you want to view by
providing the correct number as shown\n'+mwords, title='',
default='', strip=True, image=None, root=None)
# A dialog box user asks the user which video he/she wants to see. The dialog box displays
the matching videos. The user can enter the No. corresponding to the video he/she wants to
view

        z=int(z)
for i in range(0,z):
            j=u.pop()
# Used to pop the location of the required video from the list depending on the input from the
user

searchwithinvideo(j,input)# function to search within video
```

```
else:# If only one matching video is found
        a=a.pop(0)# Pop the location of video
searchwithinvideo(a,input)# function to search within video


defsearchwithinvideo(a,input):
```

Function used to search within video and has two parameters

1. Location of the video where match is found

2. Input query of the user

```
    a=str(a)# Conversion from list to string
    n=open(a+'Number', 'r')# Open file containing the number of detected shots
nk=n.readline()#reading no. of key frames
    f=open(a+'time','r')# Open the time file associated with video
    t=f.readline()# Dummy read
    g=open(a+'annotation','r')# Open the annotation file associated with video
    s=g.readline()# Dummy read


nk=int(nk)# convert string to integer
fn = range(1,nk)# array from 1 to nk


timee=[]
```

list of times for which annotation and input can match, usefulin case there are more than one match within the video

```
w=0# Counter for more than one matching within video
morethanoneresults=[]
# List to hold the no. of matching shot
    y=12345 # Initializing the variable for comparison


for i in fn:
annotation=g.readline()# Read the annotation file line by line
            o=f.readline()# Read the time file line by line
for l in range(len(input)):
                q=input[l]
                z=re.search(q,annotation)# matching input query and annotation


if z!=None:# If match is found
if i!=y or y==12345:# To avoid duplication of match
                          o=float(o)# Convert string to float
timee.append(o)
#appending all the times to the list for which annotations are
matching
                          y=i
w+=1
morethanoneresults.append(i)
#to record which shots are matching with the query
```

```
iftimee==[]:
videoplay(0.0,a)
```
Display the video from start if nothing matching within the video

```
else:
if w>1:
```
If more than one match within a video

```
for i in range(0,w):
                q=morethanoneresults.pop()
                k=str(q)
                g=a+k+'.png'
```
Location of key frames

```
img=cv2.imread(g,1)
```
Read the keyframes corresponding to matching shots

```
res = cv2.resize(img,None,fx=0.5, fy=0.5,
interpolation = cv2.INTER_AREA)
```
Resize the image for display

```
                b=str(i)
cv2.imshow(b,res)
```
Display the keyframes of matching shots

```
            w=str(w)
```
Convert integer to string

```
inm = eg.enterbox(msg='There are '+w+' relevant results.
Please indicate which one you want to view by providing
the correct number as shown in the results', title='',
default='', strip=True, image=None, root=None)
```
A dialog box asks the user which result he wants to view, after displaying key frames of matching shots

```
inm =int(inm)
```
Convert from string to integer

```
for i in range(0,inm+1):
timeepop=timee.pop()
```
For obtaining the time matching with the input provided by the user

```
cv2.destroyAllWindows()
```
Close the image displaying windows

```
videoplay(timeepop,a)
```
Call the videoplay function to play the video from the time retrieved above

```
else:
```
If only one result within the video

```
videoplay(timee.pop(0),a)
```

```
defvideoquery():# Function used to obtain the query from the user
input = eg.enterbox(msg='ENTER THE QUERY ABOUT THE CONTENT\nYOU
CAN CLOSE THE VIDEO ANYTIME BY PRESSING "c"', title=' ',
default='', strip=True, image='C:\shripadproject\Computer.jpg',
root=None)
```
Ask the user for a query for the required video content

```
if input == None:# If the user does not provide input query
eg.msgbox("Please provide an input if you wish to continue")
```
A dialog box displays the above message
```
input = eg.enterbox(msg='ENTER THE QUERY ABOUT THE CONTENT\nYOU
CAN CLOSE THE VIDEO ANYTIME BY PRESSING "c"', title=' ',
default='', strip=True, image='C:\shripadproject\Computer.jpg',
root=None)

else:# If the user provides a query
import string# Import the string module for string operations
input=string.lower(input)
```
converting all uppercase letters to lowercase letters in the query
```
spellcheck(input)# Call the spellcheck for spellcheck and correction of user query
```

```
def spellcheck(input):# Function for spell checking and correction
        x = input.split()# split the sentence into words
xwords=''# create an empty string
        s=0
        d = enchant.Dict("en_US")# object to access the enchant dictionary
for i in range(len(x)):# for all the words present in input string
            y = d.check(x[i])# check if spelling of the word is correct
if y == False and (d.suggest(x[i])!=[]):
# if there isspelling mistake present and if there aresuggestions for the mistake
                n=str(d.suggest(x[i]))# suggestions provided for the wrong words
xwords+=n#add the suggested words to empty string
                s=1#to indicate that wrong word has suggestions
xwords+='\n'# create a new line

if y == False:# if there is spelling mistake
if s==1:#if there are suggestions available
                    u='It seems there are some spelling errors. Please
check the suggestions provided\n\n'
else:# if there are no suggestions
                    u='It seems there are some spelling errors.\n\n'
xwords = str(xwords)# convert list to string


input = eg.enterbox(msg=u+xwords, title='Spelling
```

```
Error', default='', strip=True, image=None,
root=None)
```

A dialog box informs the user about spelling mistakes, provides suggestions for misspelled words and asks the user to input correct words

`spellcheck(input)` # function call for repeatedly checking for spelling errors

`else:` # if there are no spelling errors

`return(input)` # return the input sentence

`showannotation()` # Function call and start of the program execution

References

[1] Shripad A. Bhat, Omkar V. Sardessai, Preetesh P. Kunde and Sarvesh S. Shirodker, Overview of Existing Content Based Video Retrieval Systems $International Journal Of Advanced Engineering and Global Technology Vol.2\ Issue.2\ February 2014$.

[2] J. Yuan, H. Wang, L. Xiao, W. Zheng, J. Li, F. Lin, and B. Zhang, A Formal Study of Shot Boundary Detection, IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 2, pp. 168 -186, 2007.

[3]$V.T.Chasanis, A.C.Likas, and N.P.Galatsanos, Scene Detection in Videos Using Shot Clustering and Sequence Alignment, IEEE Transactions on Multimedia, vol.11, no.1, pp.89100, 2009$.

[4] A. F. Smeaton, P. Over, and A. R. Doherty, Video Shot Boundary Detection: Seven Years of TRECVid Activity, Computer Vision Image Understanding, vol. 114, no. 4, pp. 411418, 2010.

[5]$C.R.Huang, H.P.Lee, and C.S.Chen, Shot Change Detection via Local Keypoint Matching, IEEE - Transactions on Multimedia, vol.10, no.6, pp.10971108, 2008$.

[6] C. Huang, C. Chen, and P. Chung, Contrast Context HistogramAn Efficient Discriminating Local Descriptor for Object Recognition and Image Matching, Pattern Recognition, vol. 41, no. 10, pp. 30713077, 2008.

[7]$C.Harris and M.Stephens, A Combined Corner and Edge detector in Proceedings of the Fourth Alvey vision Conference, pp.147 - 151, 1988$.

[8] Hannes Muurinen, Video Segmentation and Shot Boundary Detection Using Self-Organizing Maps, masters Thesis, Helsinki university of technology.

[9]$Muhammet Bastan, Bil Video-7 : Video parsing, Indexing and Retrieval, Dissertation for doctor of philosophy Bilkent University$.

[10] Jose Lezama, Karteek Alahari, Josef Sivic and Ivan Laptev, Track to the Future: Spatio-temporal Video Segmentation with Long-range Motion Cues Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2011) .

[11]$R.Narasimha, A.Savakis, R.M.Rao, and R.DeQueiroz, Keyframe extraction using MPEG-7 motion descriptors, in Proc. Asilomar Conf. Signals, Syst. Comput., Nov.2003, vol.2, pp.15751579$

[12] T. Wang, Y. Wu, and L. Chen, An approach to video key-frame extraction based on rough set, in Proc. Int. Conf. Multimedia Ubiquitous Eng.,2007.

[13]$M.Guironnet, D.Pellerin, N.Guyader, and P.Ladret, Video summarization based on camera motion and a subjective evaluation method, EURASIP J. Image Video Process., vol.2007, pp.112, 2007$.

[14] B. Fauvet, P. Bouthemy, P. Gros, and F. Spindler, A geometrical keyframe selection

method exploiting dominant motion estimation in video, in Proc. Int. Conf. Image Video Retrieval, Jul. 2004, pp. 419427.

[15]$WeimingHu, NianhuaXie, LiLi, XianglinZeng, andStephenMaybank,$
$ASurveyonVisualContent-BasedVideoIndexingandRetrieval, IEEE$
$TransactionsonSystems, man, andcybernetics, vol.41, no.6, november2011.$

[16] H. J. Zhang, J. Wu, D. Zhong, and S. W. Smoliar, An integrated system for content-based video retrieval and browsing, Pattern Recognit., vol. 30, no. 4, pp. 643658, 1997.

[17]$X.-D.Zhang, T.-Y.Liu, K.-T.Lo, andJ.Feng, Dynamicselectionandeffective$
$compressionofkeyframesforvideoabstraction, PatternRecognition.Lett.vol.24, no.910,$
$pp.15231532, Jun.2003.$

[18] B. T. Truong and S. Venkatesh, Video abstraction: A systematic review and classification, ACM Trans. Multimedia Comput. Commun. Appl., vol. 3, no. 1, art. 3, pp. 137, Feb. 2007.

[19]$H.S.Chang, S.Sull, andS.U.Lee, Efficientvideoindexingschemeforcontent-based$
$retrieval, IEEETrans.CircuitsSyst.VideoTechnol., vol.9, no.8, pp.12691279, Dec.1999.$

[20] A. M. Ferman and A. M. Tekalp, Two-stage hierarchical video summary extraction to match low-level user browsing preferences, IEEE Trans.Multimedia, vol. 5, no. 2, pp. 244256, Jun. 2003.

[21]$Z.H.Sun, K.B.Jia, andH.X.Chen, Videokeyframeextractionbasedonspatial-$
$temporalcolordistribution, inProc.Int.Conf.Intell.Inform.HidingMultimedia$
$SignalProcess., 2008, p.196-199.$

[22] A. Girgensohn and J. Boreczky, Time-constrained keyframe selectiontechnique, Multimedia Tools Appl., vol. 11, no. 3, pp. 347358, 2000.

[23]$X.D.Yu, L.Wang, Q.Tian, andP.Xue, Multilevelvideorepresentationwith$
$applicationtokeyframeextraction, inProc.Int.MultimediaModelling$
$Conf., 2004, pp.117123.$

[24] D. Gibson, N. Campbell, and B. Thomas, Visual abstraction of wildlife footage using Gaussian mixture models and the minimum description length criterion, in Proc. IEEE Int. Conf. Pattern Recog., Dec. 2002, vol. 2, pp. 814817.

[25]$J.CalicandE.Izquierdo, Efficientkey-frameextractionandvideoanalysis,$
$inProc.Int.Conf.Inf.Technol.: CodingComput., Apr.2002, pp.2833.$

[26] H.-W. Kang and X.-S. Hua, To learn representativeness of video frames, in Proc. ACM Int. Conf. Multimedia, Singapore, 2005, pp. 423 426.

[27]$J.CalicandB.Thomas, Spatialanalysisinkey-frameextractionusingvideo$
$segmentation, inProc.WorkshopImageAnal.MultimediaInteractiveServices,$
$Lisbon, Portugal, Apr.2004.$

[28] X. M. Song and G. L. Fan, Joint key-frame extraction and object segmentation for

content-based video analysis, IEEE Trans. Circuits Syst. Video Technol., vol. 16, no. 7, pp. 904914, Jul. 2006.

[29]$C.KimandJ.Hwang, Object-basedvideoabstractionusingclusteranalysis$, $inProc.IEEEInt.Conf.ImageProcess., Oct.2001, vol.2, pp.657660.$

[30] Khushboo Khurana1, M. B. Chandak, Study of Various Video Annotation Techniques, International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 1, January 2013 .

[31]$S.Abburu, MultiLevelSemanticExtractionforCricketVideobyTextProcessing$, $InternationalJournalofEngineeringScienceandTechnology, Vol.2(10), pp.5377-5384, 2010.$

[32] A. Salway and E. Tomadaki, Temporal Information in Collateral Texts for Indexing Video, Procs. LREC Workshop on Annotation Standards for Temporal Information in Natural Language, pp. 36-43, 2002.

[33]$D.Palma, J.Ascenso, F.Pereira, AutomaticTextExtractioninDigitalVideobased onMotionAnalysis, Int.Conf.onImageAnalysisandRecognition(ICIAR2004), Porto, 2004.$

[34] J. LU, Y. Tian, Y. Li, Y. Zhang, Z. Lu, A Framework for Video Event Detection Using Weighted SVM Classifiers, Artificial Intelligence and Computational Intelligence, AICI '09 International Conference, Vol. No.4, pp.255 259, 2009.

[35]$C.Yang, M.Dong, Region-basedImageAnnotationusingAsymmetricalSupport VectorMachine-basedMultiple-InstanceLearning, InProc.IEEEComputerSociety ConferenceonComputerVisionandPatternRecognition, Volume2, June17-22, 2006.$

[36] S. Barrat, S. Tabbone, Classification and automatic annotation extension of images using Bayesian network, in S+SSPR, 2008.

[37]$R.S.Jardon, S.Chaudhury, andK.K.Biswas, Genericvideoclassification: An evolutionarylearningbasedfuzzytheoreticapproach, inProc.IndianConf.Computer VisionGraphicsandImageProcessing, pp.7991, 2002.$

[38] A. Dorado, J. Calic, E. Izquierdo, A Rule-Based Video Annotation System, IEEE Transactions on Circuits and Systems for Video Technology , Vol. 14, No. 5, May 2004.

[39]$J.R.He, M.J.Li, H.J.Zhang, H.H.Tong, andC.S.Zhang, Manifoldrankingbased imageretrieval, inProc.ACMMultimedia, NewYork, NY, pp.916, 2004.$

[40] H. Tong, J. R. He, M. J. Li, C. S. Zhang, and W. Y. Ma, Graphbased multi-modality learning, in Proc. ACM Multimedia, Singapore, pp. 862871, 2005.

[41]$J.Jeong, H.Hong, andD.Lee, Ontology-basedAutomaticVideoAnnotationTechnique InSmartTVEnvironment, IEEETransactiononconsumerElectronics, Vol.57, No.4, November2011.$

[42] A. K. Elgmagarmid, H. Jiang, A. A. Helal, A. Joshi, and M. Admed, Video database systems : issues, products, and applications Boston: Kluwer Academic Publishers, 1997.

[43]$G.J.Lu, Multimediadatabasemanagementsystems.Boston; London :: ArtechHouse$, 1999.

[44] R. Tusch, H. Kosch, and L. Bszrmnyi, VIDEX: an integrated generic video indexing approach, presented at The eighth ACM international conference on Multimedia,
Marina del Rey, California, United States, 2000.

[45]$C.Djeraba, Content-basedmultimediaindexingandretrieval, Multimedia, IEEE, vol.9, pp.18-22, 2002$.

[46] H. J. Zhang, Content-based video browsing and retrieval, in Handbook of Internet and multimedia systems and applications, B. Furht, Ed.: CRC press LLC, 1999.

[47]$E.OomotoandK.Tanaka, VideoDatabaseSystems-RecentTrendsinResearchandDevelopmentActivities, inTheHandbookofMultimediaInformationManagement, R.J.a.R.M.WilliamI.Grosky, Ed.UpperSaddleRiver, NJ : PrenticeHall, 1997, pp.405448$.

[48] D. Ponceleon, S. Srinivasan, A. Amir, D. Petkovic, and D. Diklic, Key to effective video retieval: Effective cataloging and browsing, presented at IEEE Internation Workshop on Content-based image and video databases, Bombay, India, 1998.

[49]$C.KimandJ.-N.Hwang, Fastandautomaticvideoobjectsegmentationandtrackingforcontent-basedapplications, CircuitsandSystemsforVideoTechnology, IEEE Transactionson, vol.12, pp.122-129, 2002$.

[50] C. Yajima, Y. Nakanishi, and K. Tanaka, Querying video data by spatiotemporal relationships of moving object traces, Presented at 6th IFIP Working Conference on Visual Database Systems, Brisbane, 2002.

[51]$M.Teraguchi, K.Masumitsu, T.Echigo, S.Sekiguchi, andM.Etoh, Rapidgenerationofevent-basedindexesforpersonalizedvideodigests, presentedatPatternRecognition, 2002.Proceedings.16thInternationalConferenceon, 2002$.

[52] B. Li and M. Ibrahim Sezan, Event detection and summarization in sports video, presented at Content-Based Access of Image and Video Libraries, 2001. (CBAIVL 2001). IEEE Workshop on, Sharp Labs. of America, Camas, WA, USA, 2001.

[53]$A.KokaramandP.Delacourt, Anewglobalmotionestimationalgorithmanditsapplicationtoretrievalinsportsevents, presentedatMultimediaSignalProcessing, 2001IEEEFourthWorkshopon, 2001$.

[54] C. Wu, Y.-F. Ma, H.-J. Zhang, and Y.-Z. Zhong, Events recognition by semantic inference for sports video, presented at Multimedia and Expo, 2002. Proceedings. 2002 IEEE International Conference on, 2002.

[55]$N.Babaguchi, Y.Kawai, andT.Kitahashi, Eventbasedindexingofbroadcastedsportsvideobyintermodalcollaboration, Multimedia, IEEETransactionson, vol.4, pp.68-75, 2002$.

[56] Yang Yang, Zheng-Jun Zha, Heng Tao Shen and Tat-Seng Chua, Robust Semantic Video Indexing by Harvesting Web Images, S. Li et al. (Eds.): MMM 2013, Part I, LNCS 7732, pp. 7080, 2013.

[57] Heng Tao Shen, Xiao fang Zhou, Zi Huang Jie, Shao Xiangmin Zhou, UQLIPS: A Real-time Near-duplicate Video Clip DetectionSystem School of Information Technology and Electrical Engineering, The University of Queensland.