Chapter I

Introduction

1.1 Overview

Nowadays, digital multimedia like cameras, video recorders, and digicams has become an integral part of people's lives. Be it their availability or cost or even the quality of images or videos they capture, they are all at their optimum best. This advancement has resulted in them being used in day-to-day lives wontedly.

Due to the continuous sublime raise in the digital multimedia, a lot of digital content is widely available. Furthermore, the improvements in digital content distribution have led to easy video recording. As a result, videos act as a utility and try to facilitate our lives even further. Videos can be used in various places like e-learning, gaming, training, conferencing, documentaries, movies, marketing and sales tapes, aerials, albums, surveillance etc.

Amongst the various applications of digital videos, surveillance is of a great significance. The need for surveillance has increased manifold due to increase in the demand for security. They provide a medium for storing information, usually of people for the purpose of influencing, managing, directing or protecting them. CCTV cameras can be found at banks, airports, railway stations, public places, buildings, traffic signals etc. Thus, surveillance results in very large amount of videos and in turn very large amount of data which needs to be processed.

The rapid growth in the usage of videos has led to the need to process huge chunks of visual data. Processing these very huge chunks of data demands plenty of resources like time, man-power, hardware storage, etc. So as to solve this problem, various solutions have been provided in the literature. Video summarization is one amongst them. It allows the user to navigate through and retrieve only the interesting sequences of the video. It helps in efficient storage, quick browsing and retrieval of large collection of video data without losing important aspects. Video summarization can be defined as a non-linear content-based sampling algorithm, which provides a compact representation of a given video.

Specifically, any video summarization technique offers the following requirement, "*I have just an hour to watch this video, tell me what to watch and where to watch*". It automatically compiles the most salient and informative portion of the video for the users, by automatically scanning through the video stream, clustering the related scenes and removing the temporally redundant contents.

The proposed and implemented project is a video summarization system. It uses various clustering techniques for summarization. Unlike earlier techniques, the average processing time taken by this system is roughly equal to the original length of the video. It also compares and analyses three well-known clustering techniques.

1.2 Motivation

Video has been around for a long time. Producing styles have evolved over the years, distribution channels have emerged, interactivity has blossomed, and technology has changed the face of videos over the years. The old cliché says "*A picture is worth a thousand words*". Now that picture is no longer a picture, but a video. Here are some facts according to Going Social: The State of Video in 2013 that support the paradigm shift from "*A picture is worth a thousand words*" to "*A moving picture is worth a million people*":

- ✓ Forty billion videos are streamed in U.S. each month
- ✓ Seventy five people watch videos online each month
- ✓ $6.3 billion will be spent on video ads in 2015

These are not just numbers, they help us grasp the magnitude of why video is more than just a fantastic story telling medium. With the increasing importance of videos in day to-day life, it is necessary to develop efficient and flexible video processing techniques.

Surveillance videos are gaining popularity since early 1990's. There has been a significant expansion of CCTV surveillance around the world. The globalised trends of late modernity have accelerated this growth. Few facts regarding surveillance videos across the world are as follows:

- ✓ Beijing, London, Chicago, Houston and New York are the top five cities with the largest surveillance networks

- ✓ About 400,000 surveillance cameras have been installed in Beijing and 70,000 have been added since the end of 2010, according to the Beijing Security and Protection Industry Association (BSPIA)
- ✓ UK has 1% of world's population but 20% of its CCTV cameras
- ✓ India and China have the highest CCTV camera markets in Asia

Surveillance videos are usually very long. They exhibit redundancy at a very large scale. Due to various time and storage constraints summarizing such videos becomes a necessity than an accessory. The main motive of video summarization is to value time and make the video as short and crisp as possible within a smaller time span, albeit retaining the important aspects of the video.

Video summarization does not restrict itself to surveillance videos. Imagine viewing wedding videos. Most people would not want to go through these long recordings. A video summary in such cases could provide a condensed and viewer-friendly recap.

Most of the existing summarization techniques have a processing time which is at least 1.5 times the original video length. Despite reducing the video length, these systems, thus, fail to fulfil its main motive of reducing the effective time. The importance of videos in the current era, the size of videos and the flaws in the existing video summarization techniques motivated this project.

1.3 Problem Formulation

There are two main techniques for video summarization: Key-frame based and video skimming. In key frame based techniques, a subset of representative frames that contain significant visual content are selected and then played as video summaries. These representative frames are extracted using techniques like clustering. In video skimming, the original video is segmented into various parts of shorter duration; interesting segments are then chosen and joined to form a summary.

Both the techniques have their own advantages and disadvantages. Key-frame based summarization selects the most valuable frames, but does not produce a continuous video. On the other hand, video skimming promises continuity but is not efficient in terms of choosing the segments. This system implements the best of both techniques under one roof. It uses the idea of clustering from key frame summarization and uses the idea of temporal segmentation from video skimming.

The working of this video summarization system mimics the human brain. This analogy is explained as follows: Man starts scanning the video as-and-when it begins. He keeps in mind all those video clippings which he had watched in this process. Whenever he sees a new part of the video he tries to compare it with what was already seen. If the key features of both the videos are the same, then he concludes that this portion of the video was redundant and he could have had omitted watching the same. Subconsciously, he creates scenes in his mind, which technically can be thought of as clusters. Thus, the basic problem for video summarizer is to develop this cognitive model. Apart from this, issues like complexity, responsiveness and accuracy exist. Effective time must not exceed twice the original video length. The system must take videos of both structured and unstructured nature.

1.4 Overview of Proposed and Implemented Solution

As discussed in section 1.3, the system mimics the working of human brain. The proposed and implemented work is nothing but a clustering agent. Clustering is chosen as it is one of the most common and effective video summarization mechanisms. The clustering agent perceives the inputs of the video, which happens to be its environment. It creates clusters and assigns segments to these clusters. Valuable segments are chosen from these clusters in a round robin fashion.

The crisp idea of the project is as follows: Given a video, the video summarizer scans through the main video stream and generates the segments temporally. The first frame of each segment is considered to represent it. The histograms of all these representative frames are plotted. Based on these histograms, value vectors for all frames are computed. These vectors, or the dataset *D*, are the inputs to the clustering algorithms. The various algorithms used in this project are- K-means, KFCG and FCM. These algorithms return the clustered data set. Frames are chosen from the clusters in round robin manner and corresponding segments are added to the summary video file.

The main novelty with this approach is that it is simple and provides consistent output videos. All ideas of learning, reasoning, understanding and responding are taken from data mining and various artificial intelligence techniques. It is due to this reason that the system promises to perform partly-online summarizing rather than a batch approach.

1.5 Scope of the Project

The scope of this project is implicit. Its application area ranges from surveillance to ordinary videos.

The system can be profoundly used in the surveillance set-up. Assuming that we have been provided with a 2 hour video and we want to spot a person who passed by just for a minute. It is extremely unfruitful to watch the entire length to just find a thing which is not even one-hundredth of the input time. The system solves this problem by reducing the video length effectively. Thus, the system would be of great use to investigation officers.

It can also be used to reduce the lengths of boring monotonous videos. Various recordings of functions like weddings can be made crisp with the help of this system.

Sports videos and documentaries are no exception. The system can summarize them accurately. The system retains the original story-line of these videos to a great extent.

These are just some applications of the system. The scope of the system is not limited. It can be used wherever a video is involved.

Chapter II

Review of Literature

Due to the immense improvements and advances in the Digital world, video summarization can be thought of as the need of the hour. Marat Fayzullin proposed a model of video summarization based on three parameters: Priority (of frames), Continuity (of the summary) and non-repetition (of the summary). As per the CPR model, an optimal summary is the one that maximizes an objective function based on these three parameters. [1] All the existing summarization techniques follow this model.

A video summary represents the abstract view of the original video. It is simply a highlight of the original sequence which is the concatenation of selected video segments or key frames. [2] Essentially, video summarization can be of two types: Static or Dynamic. Static video summaries are composed of a set of key frames extracted from the original video sequence. On the other hand, Dynamic summaries are composed of a set of shots and are produced by taking into account the similarity or domain-specific relationships among all video shots. Static video summaries can be created for originally structured videos like movies; whereas Dynamic video summaries are more appropriate for unstructured random videos like surveillance videos. The need for summarization is more for unstructured data as they are more profoundly dilated in our day to day lives. Hence, it is necessary to work for and develop dynamic summarization more closely.

Most of the static summarization techniques use a key frame based approach. A key frame is a frame that represents the content of a logical unit. [3] A key frame must be as much representative as possible. Key frames based summarization was further classified using sampling, scene segmentation and shot segmentation. [4]

Research was going on for having different approaches to key frame summarization. One such included extracting key frames which represent the most important contents of the video. [5] Such approaches use the basic idea of frame difference.

A popular technique in key frame summarization is to compute frame differences based on some criteria and then discard the frames whose difference with the adjacent frames are less than a certain threshold. Various low level features have been applied for this purpose including colour histograms, frame correlations, etc. [6] The frame difference based methods are intuitive and simple in nature. These properties make them suitable for real-time applications. However, for extracting a particular key frame, these techniques consider only sufficient content change between consecutive frames. Therefore, they do not completely represent the video preceding it. Moreover, the key frame based approach does not give smooth and continuous summaries.

In any dynamic summarization, the problem under consideration is the representation of the various features. A feature vector in this case represents an interesting point. The use of such vectors makes the problem more manageable while providing increased robustness to noise and pose variation. Spatio- temporal cuboids [8 9] are used for this behaviour recognition and feature representation. They describe the local temporal patch around the detected interest points. In other words, Spatio-temporal cuboids provide a recognition algorithm for interesting points. After detecting the interesting points, the point of concern is to represent them as a vector. Various features have to be extracted like visual appeal, motion, sound, etc. The approach using histograms has led to state-of-art in the feature representation of an interesting point. Histogram of Gradient [10] is used for representing an object; whereas Histogram of Optical Flow [11] is used for representing motion. Based on these approaches, Zhao [12] proposed a dynamic model for unstructured videos.

So as to make the key frame based techniques effective, we encapsulate the idea of both static and dynamic summarization under one roof. This not only makes it easy to implement but also makes the system almost in par with any dynamic technique.

The proposed and implemented system implements key features of Zhao model in the normal key-frame based set-up. It uses clustering for summarization [13]. Clustering analysis has been studied in other areas like data mining, machine learning and statistics. Different clustering techniques have been proposed with different capabilities and different computational requirements. Most clustering techniques share their need for user-specified arguments and prior knowledge to produce best results.

Clustering is multi-disciplinary [14 15]. Clustering has always been used in statistics and science. It was introduced in pattern recognition framework in 1973 by Duda and Hart. Image segmentation clustering algorithms can be applied to image segmentation [16] and computer vision [17]. Clustering is also used for data compression in image processing; this is also known as vector quantization. It can also be used for data modelling.

In video summarization, visually similar frames are clustered into one group using the Euclidean distance measure. When clusters are formed, a fraction of the frames that has given a larger distance metric is retrieved from each group to form a sequence making up the desired output [18]. Clustering ensures that video summary represents the most unique frames of the input video and gives equal attention to preserving continuity of the summarized video.

Clustering algorithms can be classified into two categories: partitional and hierarchical [19]. Partitional clustering attempts to directly decompose the data sets into a set of disjoint clusters. The criterion function that the clustering algorithm tries to minimize may emphasize the local structure of the data, as by assigning clusters to peaks in the probability density function, or the global structure. Typically the global criteria involve minimizing some measure of dissimilarity in the samples within each cluster, while maximizing the dissimilarity of different clusters. K-Means and FCM are some examples of partitional clustering techniques. K-Means algorithm starts by taking *k* centroids, one for each cluster and then associates each point in the given data set to the nearest centroid [20]. FCM algorithm works by assigning membership to each data point corresponding to each cluster on the basis of distance between the cluster centre and data point; more the data is nearer to the cluster centre more is its membership towards that particular cluster [21].

Hierarchical clustering proceeds successively by either merging smaller clusters into larger ones or by splitting larger clusters. The clustering methods differ in the rule by which it is decided which two small clusters are merged or which large cluster is split. KFCG [22] is an example of top-down hierarchical clustering. In KFCG algorithm, initially centroid is computed. Two vectors are generated by adding proportionate error to the centroid. Euclidean distances of all the points are computed with respect to the vectors. Two clusters are formed based on the proximity to the vectors. This process continues for many iterations.

This system implements and analyses K-Means, FCM and KFCG clustering methods.

Chapter III

System Analysis

3.1 Functional Requirements

The system requires the user to browse and select the required video intended to be summarized. Having selected the video, the GUI provides the user an option amongst the three clustering methods- K-means, KFCG and FCM – to be used for summarization. Having selected the method, the processing begins after which the user can view the summary. A progress bar shows the status of summarization.

On the other hand, once the user selects the process button, the system agent scans through the input video, obtains temporal data, extracts the frames and stores them for further processing. It then processes the frames stored, using the method prompted by the user and stores the frames, used for summary construction. Once the user opts to view the summary from the GUI, a summary video is constructed from the former frames. The summary is displayed and the frames stored are thereby deleted.

Based on these, the functional requirements of the various subsystems are as described in sections 3.1.1 and 3.1.2.

3.1.1 Input System Requirements

The user must essentially do the following for the summarization process:

i. Upload a video of an appropriate quality with the minimum signal-to-noise ratio
ii. The video must have at least some redundancy, else the effect of summarization would be minimal
iii. The input video must be in AVI or MP4 format
iv. The input video needs to be unstructured for optimum results

3.1.2 Processing and Output System Requirements

i. Individual frames extracted, need to be saved in a directory, so that their histograms can be obtained and further processing- as prompted by user- can be done
ii. The system must have sufficient space to store the frames which are being extracted and processed

3.2 Non Functional Requirements

A non-functional requirement specifies the criteria that can be used to judge the operation of a system, rather than specific behaviours. The various non-functional requirements of the video summarization system are:

i. Performance: Requires a high speed processor so that the summarization can be done as fast as possible.
ii. Latency: Processes the videos such that the length of the summarized videos is less
iii. Accuracy: The output summary must contain all the important aspects of the original video

3.3 Specific Requirements

The specific requirements highlight the various software and hardware requirements of the system.

Software requirements:

i. MATLAB R2013a and ahead.
ii. Windows 7 or ahead

Hardware requirements:

i. 3.4 GHZ Intel Core i3 processor or ahead.
ii. Minimum 4 GB of RAM

3.4 Use Case Diagram and Description

Based on the functional requirements and system description, the use case diagram for the summarization system is constructed. Tables 3.1-3.5 give the description of the various use cases used in Figure 3.1

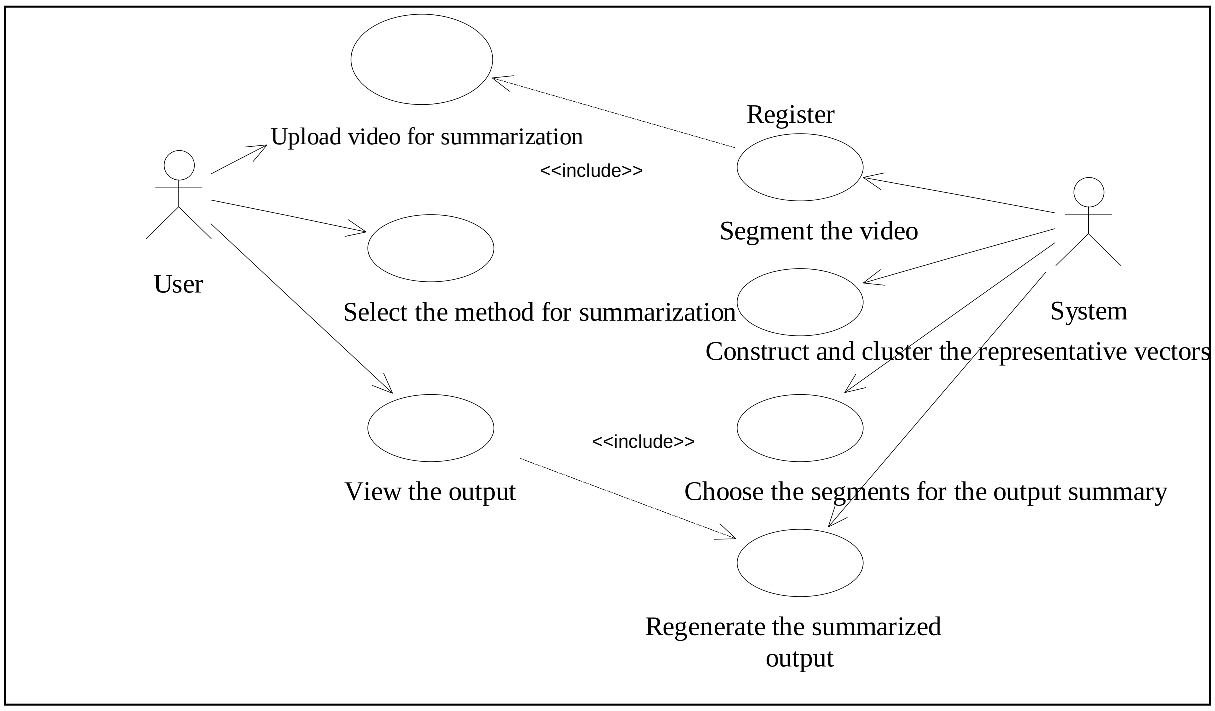

Register
Upload video for summarization
<<include>>
User
Segment the video
Select the method for summarization
System
Construct and cluster the representative vectors
<<include>>
View the output
Choose the segments for the output summary
Regenerate the summarized output

Fig. 3.1 Use Case Diagram of the video summarization system

Table 3.1 Description of the use case- Select the method for summarization

Use case name	Select the method for summarization
Actor	System
Brief description	The system implements three clustering techniques namely: K-Means, KFCG and FCM. The user choses the method in which he is interested
Preconditions	Video should be uploaded
Post-conditions	Appropriate clustering technique is chosen
Flow of Events	1. The user accesses the system and uploads the video 2. The user then choses the technique needed by him

Table 3.2 Description of the use case-Segment the video

Use case name	Segment the video
Actor	System
Brief description	This module scans the entire unstructured video and makes temporal segments
Preconditions	Video should be uploaded and should be unstructured
Post-conditions	Make temporal segments
Flow of Events	1. The user accesses the system and uploads the video 2. The video is then scanned and temporal segments are made

Table 3.3 Description of the use case-Construct and cluster the representative vectors

Use case name	Construct and cluster the representative vectors
Actor	System
Brief description	First frame from each segment is considered to be the representative frame for that segment. The histogram vectors for these frames are constructed. These vectors are then clustered using the selected technique
Preconditions	The entire video should be scanned and temporal segments should be made
Flow of Events	1. The uploaded video is scanned 2. Based on the frame rate, the temporal segments are made 3. First frame of each segment is taken 4. Histograms for these frames are computed 5. Each Histogram is represented as a vector 6. These vectors are clustered using the chosen technique

Table 3.4 Description of the use case - Choose the segments for output summary

Use case name	Choose the segments for output summary
Actor	System
Brief description	Representative frames from each cluster are chosen in a round robin fashion. The corresponding segments to these frames are then extracted
Preconditions	Value vectors of representative frames must be clustered using some appropriate technique
Post-conditions	Get the segments to be present in the output summary
Flow of Events	1. The frames are clustered 2. Frames from these clusters are chosen in a round robin fashion 3. Corresponding segments are then extracted

Table 3.5 Description of the use case - Regenerate the Summarized output

Use case name	Regenerate the summarized output
Actor	System
Brief description	This module plays the sequences in the chosen segments as the output summary
Preconditions	Video must be segmented and the segments must be clustered
Post-conditions	Generate the output
Flow of Events	1. The frames are clustered 2. Few frames are chosen and the corresponding segments are extracted 3. These segments are then played as the output summary

Chapter IV

Analysis Modelling

4.1 Data Flow Diagram

The various levels of DFD of the video summarization system are given in Figures 4.1-4.6. The data dictionary for the same are given in Tables 4.1-4.6

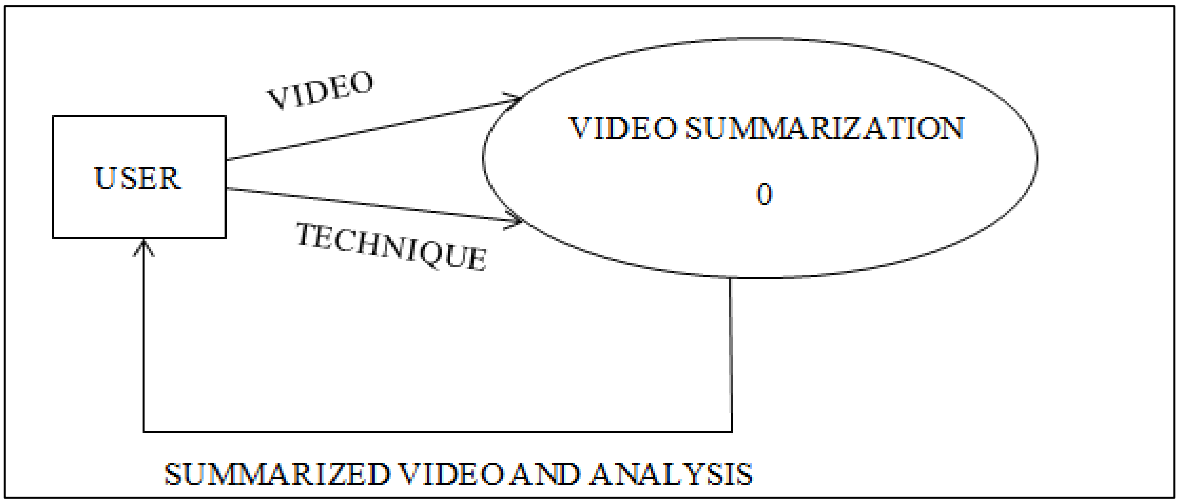


Fig. 4.1 DFD Level 0: Context Diagram for Video Summarization

Table 4.1 Data Dictionary for DFD Level 0

Data	Description	Data Type
Video	The video to be summarized is browsed and given to the system	AVI MP4
Technique	The user choses the technique for summarizing the video	String
Summarized Video and Analysis	The input video is processed and the summarized video is given to the user. The user can also compare the various clustering techniques for summarizing the given video	AVI MP4

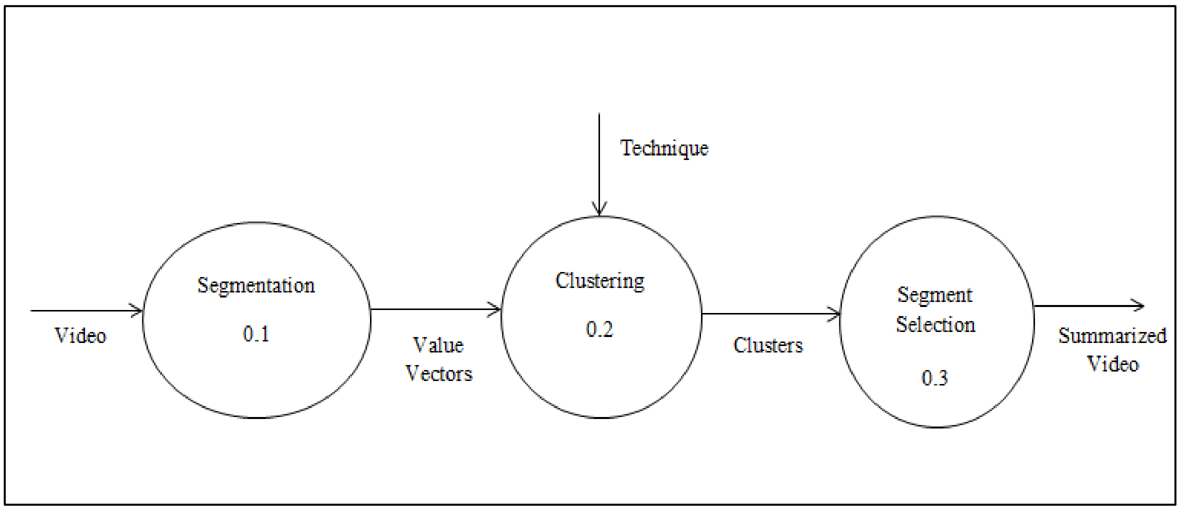


Fig. 4.2 DFD Level 1

Table 4.2 Data Dictionary for DFD Level 1

Data	Description	Data Type
Video	The video to be summarized is browsed and given to the system	AVI MP4
Value Vectors	Every Segment is represented using a value vector	Unsigned Integer
Technique	The user gets to choose the method for clustering: K-Means, KFCG, FCM	String
Clusters	Similar video segments fall under one cluster	Cell Array
Summarized Video	The input video is processed and the summarized video is given to the user	AVI MP4

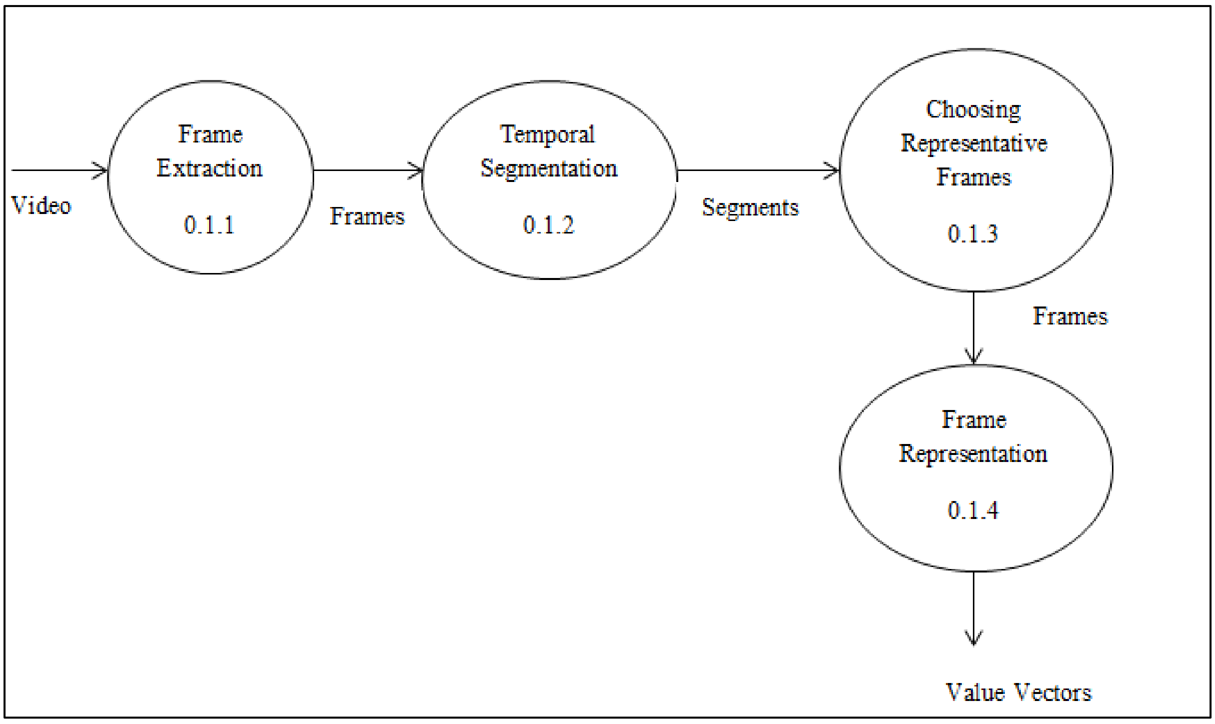

Video
Frame Extraction 0.1.1
Frames
Temporal Segmentation 0.1.2
Segments
Choosing Representative Frames 0.1.3
Frames
Frame Representation 0.1.4
Value Vectors

Fig. 4.3 DFD Level 2 for segmentation

Table 4.3 Data Dictionary for DFD Level 2 for Segmentation

Data	**Description**	**Data Type**
Video	The video to be summarized is browsed and given to the system	AVI MP4
Frames	Videos can be thought of as a set of images which are moved at a constant rate. Frames are these constituent images of the video	JPEG
Segments	Group of frames form a segment	Array of Images
Frames	Every segment is represented by a frame. For convenience, we assume the first frame of every segment to represent that segment	JPEG
Value Vectors	Every frame can be represented as a vector. This vector is known as the value vector	Unsigned Integer

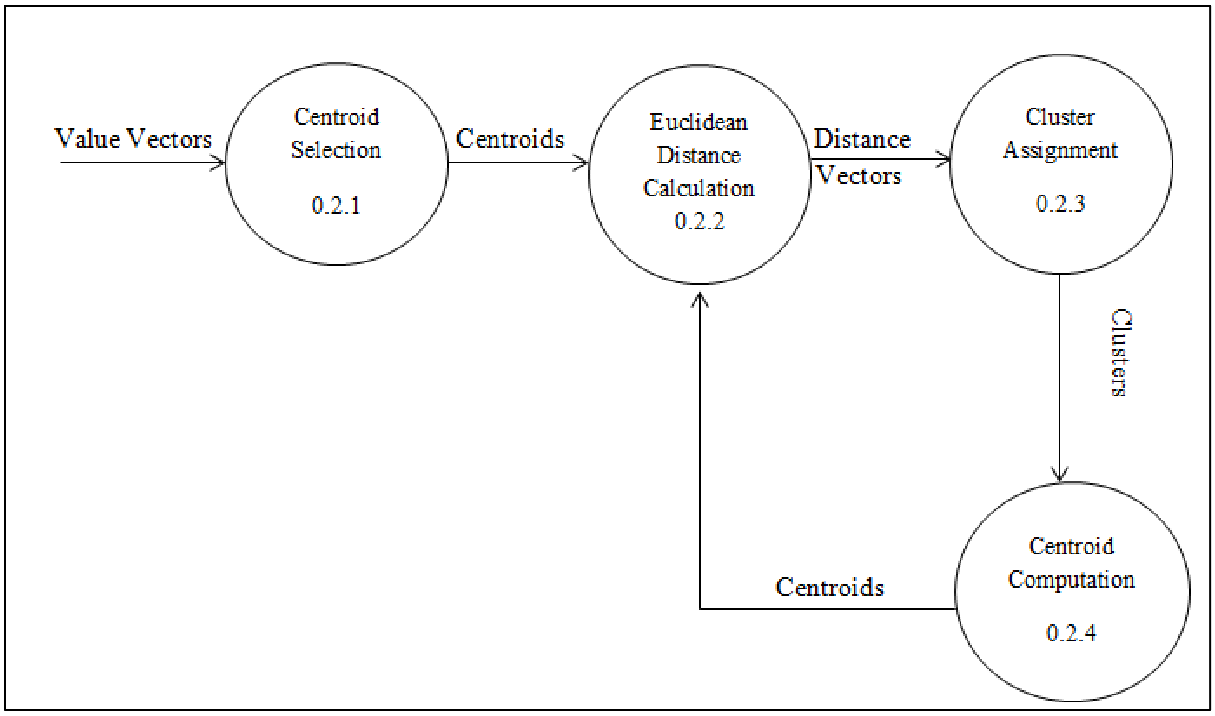


Fig. 4.4 DFD Level 2 for K-Means Clustering

Table 4.4 Data Dictionary for DFD Level 2 for K-Means Clustering

Data	**Description**	**Data Type**
Value Vectors	They are the representative vectors of the frame	Unsigned Integer
Centroids	Video can be thought of as a set of value vectors. Amongst these vectors, we choose some vectors as the centroids	Integer Array
Distance Vectors	Distance of every vector with the centroid is computed	Integer Array
Clusters	Similar video segments fall under one cluster. Lesser the distance vector, more similar are the vectors and thus, more the probability of falling under one cluster	Cell Array
Centroids	Centroids of the new clusters are again calculated. This continues for some fixed number of iterations	Integer Array

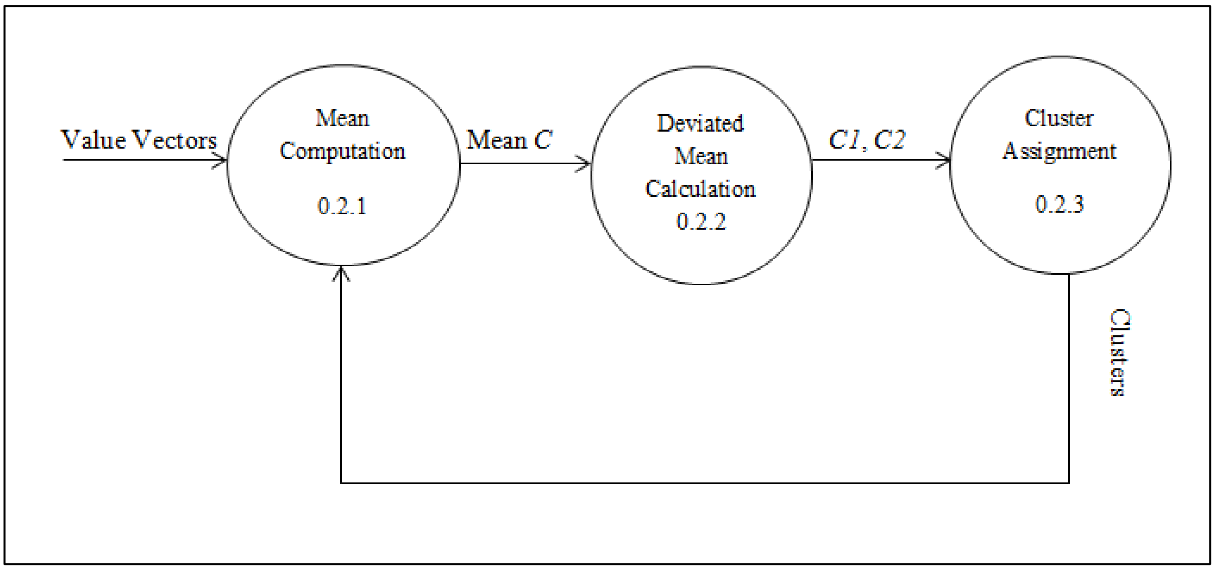


Fig. 4.5 DFD Level 2 for KFCG Clustering

Table 4.5 Data Dictionary for DFD Level 2 for KFCG Clustering

Data	**Description**	**Data Type**
Value Vectors	They are the representative vectors of the frame	Unsigned Integer
Mean *C*	The average of the cluster under consideration is computed. Initially, we assume the entire set of value vectors as one cluster	Double
C1, *C2*	These two values are generated due to deviations in the original mean $C1 = 1.1 \text{ x } C$ $C2 = 0.9 \text{ x } C$	Double
Clusters	Similar video segments fall under one cluster. Here, the distances are computed with respect to *C1* and *C2*	Cell Array

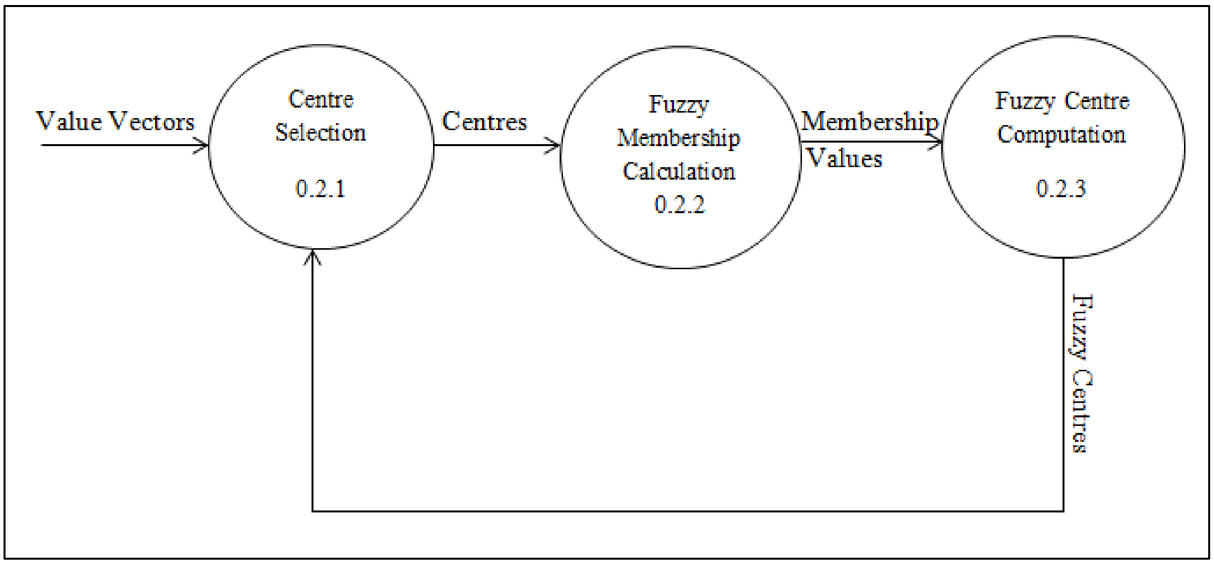


Fig. 4.6 DFD Level 2 for FCM Clustering

Table 4.6 Data Dictionary for DFD Level 2 for FCM Clustering

Data	**Description**	**Data Type**
Value Vectors	They are the representative vectors of the frame	Unsigned Integer
Centres	Random centres from the initial set is chosen	Integer Array
Membership Values	Membership values of every cluster-element pair is computed	Integer Matrix
Fuzzy Centres	Based on the membership values, new fuzzy centres are computed	Integer Array

The Level 0 DFD gives the basic structure of the system. The user gives the video and his desired clustering technique. The video summarization algorithm processes these inputs and gives back the summarized video as the output.

The Level 1 DFD gives the various stages of the summarization process. The video is first segmented temporally. Feature vectors are used to represent these segments. The feature vectors are clustered. Segments from these clusters are chosen randomly.

The Level 2 DFD for segmentation explains the process of segmentation in detail. Frames are extracted from the video. Group of frames form a segment. The first frame of every segment represents the segment. Histogram is made for the representative frames. With the help of these histograms, value vectors are calculated.

4.2 Activity Diagram

The activity diagram gives the complete flow of the project from the system point of view. The activity diagram for the video summarization system is given in Figure 4.7. This diagram can be described as follows: the user first browses for the video and choses the technique for clustering. Video summarizer first segments the video and evaluates vectors for every segment. It then clusters these vectors based on the technique chosen by the user. Representative frames are chosen from these clusters in a round robin manner. The segments corresponding to the chosen frames are extracted. The sequence of these segments is the output summarized video.

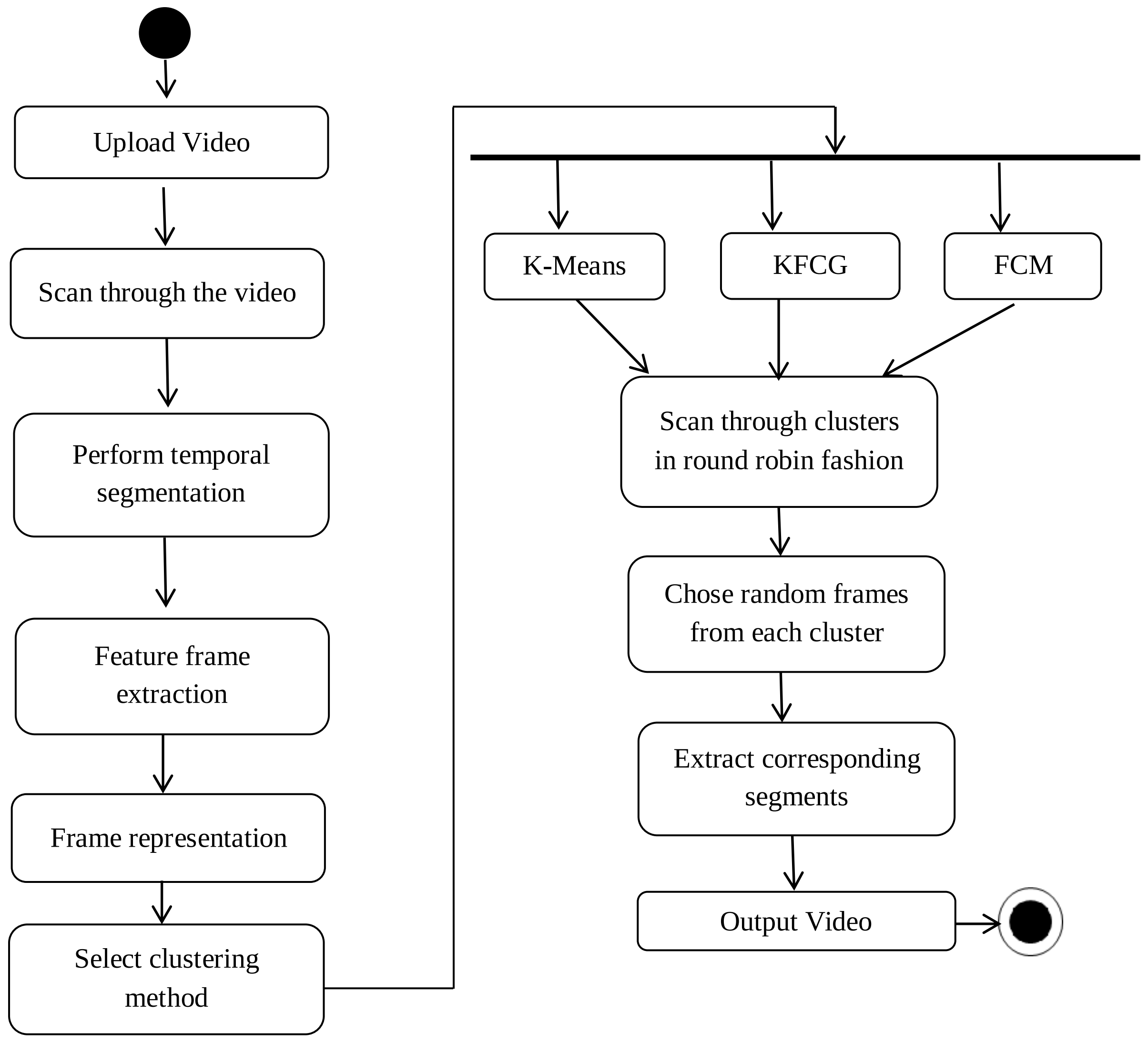
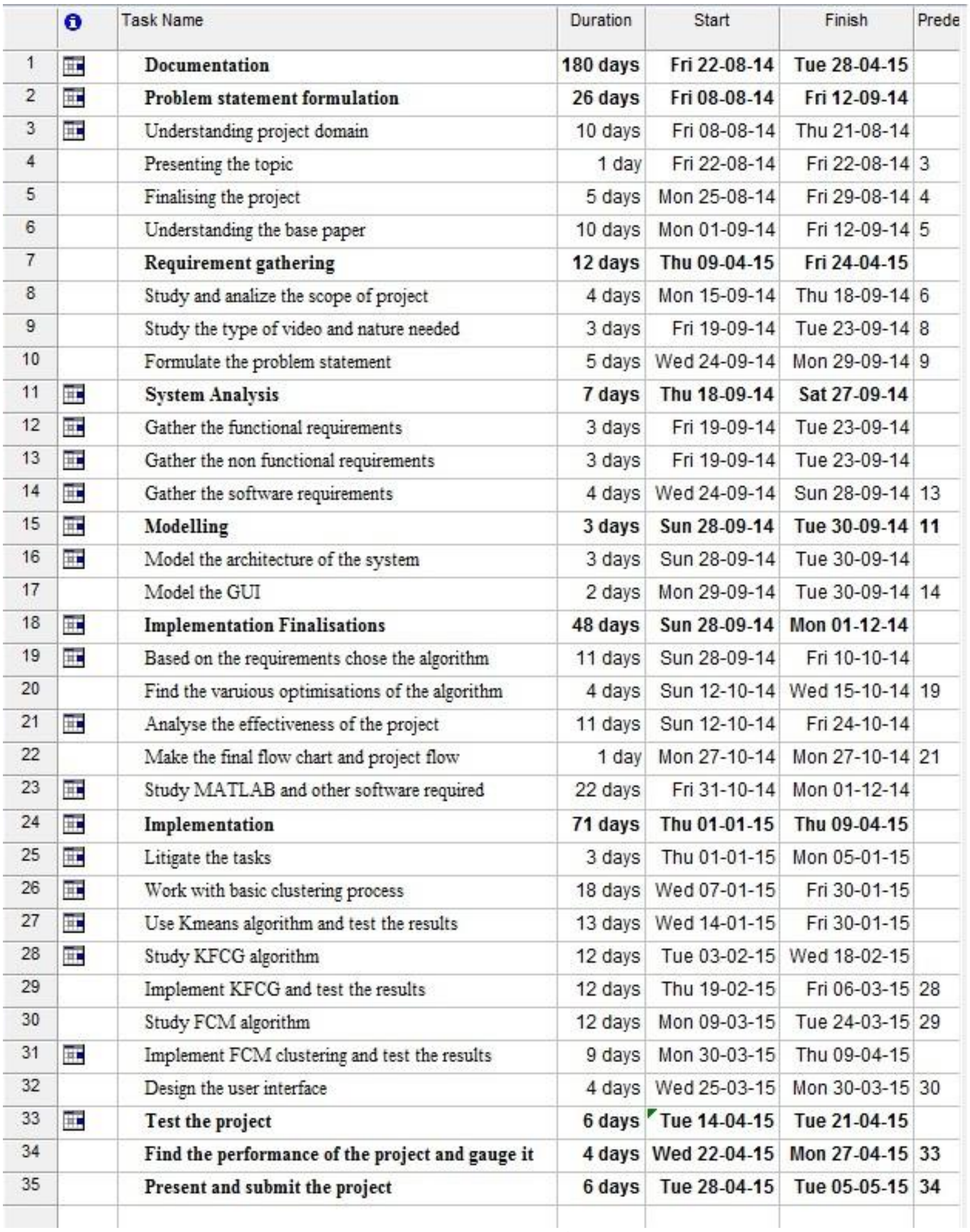

Upload Video
Scan through the video
Perform temporal segmentation
Feature frame extraction
Frame representation
Select clustering method
K-Means
KFCG
FCM
Scan through clusters in round robin fashion
Chose random frames from each cluster
Extract corresponding segments
Output Video

Fig. 4.7 Activity Diagram for the Video Summarization System

4.3 Timeline Chart

Project timeline is a chart showing progress on different project activities on calendar time scale. The timeline chart in Figure 4.8 gives the scheduled plan for this project in the year 2014-2015.



	ℹ	Task Name	Duration	Start	Finish	Prede
1	▦	**Documentation**	**180 days**	**Fri 22-08-14**	**Tue 28-04-15**	
2	▦	**Problem statement formulation**	**26 days**	**Fri 08-08-14**	**Fri 12-09-14**	
3	▦	Understanding project domain	10 days	Fri 08-08-14	Thu 21-08-14	
4		Presenting the topic	1 day	Fri 22-08-14	Fri 22-08-14	3
5		Finalising the project	5 days	Mon 25-08-14	Fri 29-08-14	4
6		Understanding the base paper	10 days	Mon 01-09-14	Fri 12-09-14	5
7		**Requirement gathering**	**12 days**	**Thu 09-04-15**	**Fri 24-04-15**	
8		Study and analize the scope of project	4 days	Mon 15-09-14	Thu 18-09-14	6
9		Study the type of video and nature needed	3 days	Fri 19-09-14	Tue 23-09-14	8
10		Formulate the problem statement	5 days	Wed 24-09-14	Mon 29-09-14	9
11	▦	**System Analysis**	**7 days**	**Thu 18-09-14**	**Sat 27-09-14**	
12	▦	Gather the functional requirements	3 days	Fri 19-09-14	Tue 23-09-14	
13	▦	Gather the non functional requirements	3 days	Fri 19-09-14	Tue 23-09-14	
14	▦	Gather the software requirements	4 days	Wed 24-09-14	Sun 28-09-14	13
15	▦	**Modelling**	**3 days**	**Sun 28-09-14**	**Tue 30-09-14**	**11**
16	▦	Model the architecture of the system	3 days	Sun 28-09-14	Tue 30-09-14	
17		Model the GUI	2 days	Mon 29-09-14	Tue 30-09-14	14
18	▦	**Implementation Finalisations**	**48 days**	**Sun 28-09-14**	**Mon 01-12-14**	
19	▦	Based on the requirements chose the algorithm	11 days	Sun 28-09-14	Fri 10-10-14	
20		Find the varuious optimisations of the algorithm	4 days	Sun 12-10-14	Wed 15-10-14	19
21	▦	Analyse the effectiveness of the project	11 days	Sun 12-10-14	Fri 24-10-14	
22		Make the final flow chart and project flow	1 day	Mon 27-10-14	Mon 27-10-14	21
23	▦	Study MATLAB and other software required	22 days	Fri 31-10-14	Mon 01-12-14	
24	▦	**Implementation**	**71 days**	**Thu 01-01-15**	**Thu 09-04-15**	
25	▦	Litigate the tasks	3 days	Thu 01-01-15	Mon 05-01-15	
26	▦	Work with basic clustering process	18 days	Wed 07-01-15	Fri 30-01-15	
27	▦	Use Kmeans algorithm and test the results	13 days	Wed 14-01-15	Fri 30-01-15	
28	▦	Study KFCG algorithm	12 days	Tue 03-02-15	Wed 18-02-15	
29		Implement KFCG and test the results	12 days	Thu 19-02-15	Fri 06-03-15	28
30		Study FCM algorithm	12 days	Mon 09-03-15	Tue 24-03-15	29
31	▦	Implement FCM clustering and test the results	9 days	Mon 30-03-15	Thu 09-04-15	
32		Design the user interface	4 days	Wed 25-03-15	Mon 30-03-15	30
33	▦	**Test the project**	**6 days**	**Tue 14-04-15**	**Tue 21-04-15**	
34		**Find the performance of the project and gauge it**	**4 days**	**Wed 22-04-15**	**Mon 27-04-15**	**33**
35		**Present and submit the project**	**6 days**	**Tue 28-04-15**	**Tue 05-05-15**	**34**

Fig. 4.8 Timeline Scheduling for the project

The various phases as described in the project time line are:

i. Problem Statement Formulation and Requirements Gathering
ii. System Analysis
iii. Modelling
iv. Ground work and Implementation
v. Testing

A Gantt chart provides a graphical illustration of a schedule that helps to plan, coordinate, and track specific tasks in a project. The Gantt chart for the timeline scheduling in Figure 4.8 is given in Figure 4.9.

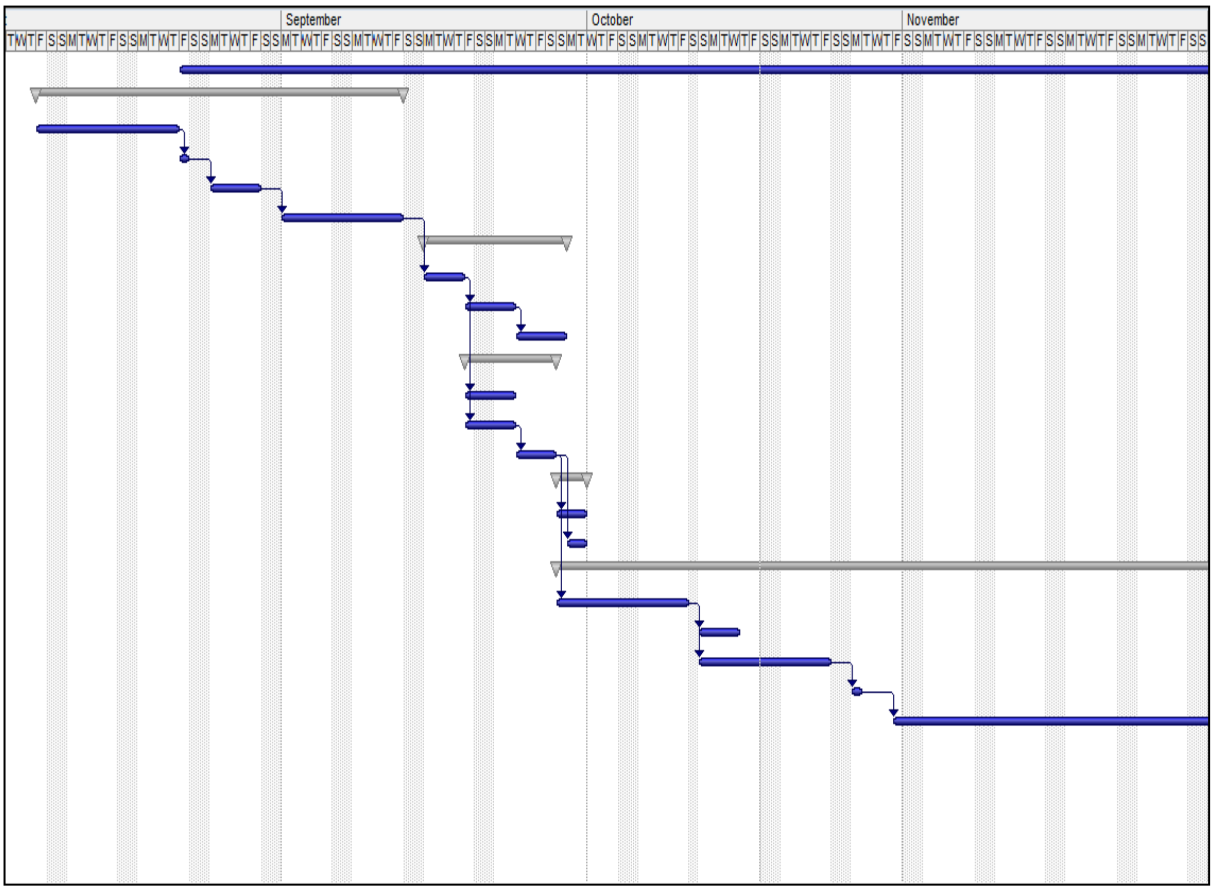


Fig. 4.9 Gantt chart for the project

Chapter V

Design

5.1 System Description

The system starts scanning the video. Temporal segmentation is performed while scanning. The length of a segment, k varies as per the video length. It is assumed that the first frame of every segment represents that segment; thus it is considered as the representative frame. The vectors for the representative frames are known as value vectors. Value vectors can be simply thought of as very big numbers. Value vectors are computed as follows:

$$value = \Sigma\, i \times histogram\,(i) \qquad (1)$$

histogram (i) denotes the intensity value in the histogram of the i^{th} intensity pixel. The value of i varies from 0 to 255. For the sake of complexity, the frames are converted to grey scale before finding the histogram. This is so because if left in the RGB colour system, three different histograms for R, G and B would have had to be computed. The value vector increases as the image tends to be brighter. Thus, as per the system, brighter objects tend to show distinct important features.

Every segment is, thus, represented by a value vector. Hence, the process of clustering segments simply results to the process of clustering the value vectors. These vectors are clustered using any of the clustering techniques (K-Means, KFCG and FCM). Each algorithm makes eight different clusters. The user gets to choose the clustering technique as per his/ her interest.

The output summary will always have 8 segments. Making the number of output segments will not make the video length constant as the size of each segment is different for different videos. Each cluster is moved across in a round robin manner. The highest valued segment is always chosen from the cluster. The summarised video is simply the collection of these segments.

5.2 Description of the Architecture

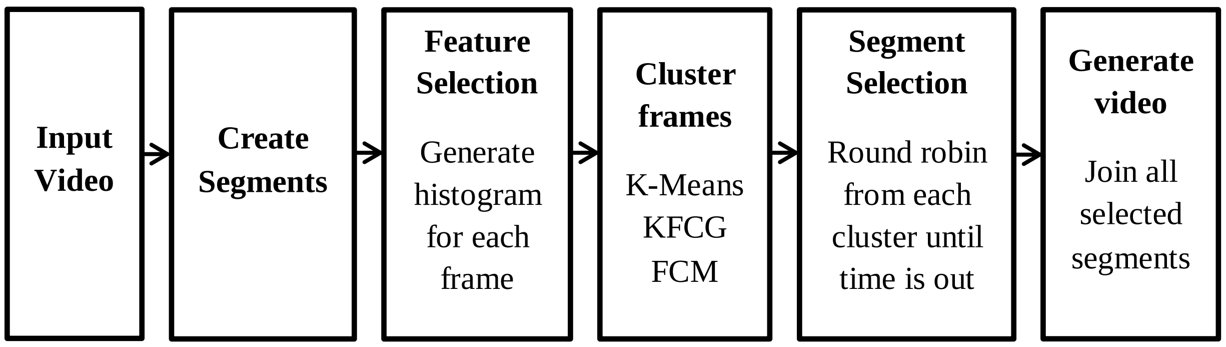

Input Video
Create Segments
Feature Selection
Generate histogram for each frame
Cluster frames
K-Means
KFCG
FCM
Segment Selection
Round robin from each cluster until time is out
Generate video
Join all selected segments

Fig. 5.1 Block Diagram of the video summarization system

The various phases in the Figure 5.1 are explained as follows.

1. Input Video

The input video is taken from the user. Frames are extracted from this video. The video should be in AVI or MP4 format

2. Create Segments

Group of frames form a segment. Let *t* denote the initial video length. Let *k* denote the time duration of each segment in seconds. Let *f* denote the frame rate of the video.

$$k = 2 \times round\left(\frac{t}{60}\right) \tag{2}$$

Initially, the video duration is in seconds. This duration is then converted from seconds to minutes by dividing by 60. The obtained duration is rounded off to the nearest integer. It must be noted that the value of *k* cannot be 1. This is so because a 1 frame segment is equivalent to no segmentation at all. Thus, the rounded integer is multiplied by 2 so as to make the minimum segment length as 2.

Let *t* be less than 30 seconds.

$$\begin{gathered} \frac{t}{60} < 0.5 \\ round\left(\frac{t}{60}\right) = 0 \\ k = 0 \end{gathered} \tag{3}$$

The segment length will be zero if the original video length is less than 30 seconds. Thus, we restrict the system to take videos of length of at least 30 seconds. For videos of length lesser than 30 seconds, the system would give errors.

Number of frames in a segment, n, can then be computed easily.

$$n = k \times f \tag{4}$$

3. Feature Selection

The representative frames are chosen. For simplicity, the first frame of each segment is chosen as the representative frame. These frames are then represented by the value vectors. Value vectors can be computed by using Equation 1.

4. Cluster Frames

Data set to be clustered is the set of value vectors of the representative frames. The data set is clustered according to the technique chosen by the user.

5. Segment Selection

The number of output segments is kept as 6. Thus, the clusters are traversed in a round robin manner for 6 iterations. In a single iteration, only one cluster is considered and only one vector is chosen from the cluster. So as to optimise the results, the highest valued value vector is always chosen from the clusters.

Segments corresponding to the value vectors are extracted. These segments form the sequences of the summarized video.

6. Generate Video

The output segments are then played as a video.

5.3 User Interface Design

Before implementation, it is necessary to design the final interface of the system. The menu structure and navigation flow must also be developed in order to avoid poor interface design. The basic interface of the video summarization system is given in Figure 5.2

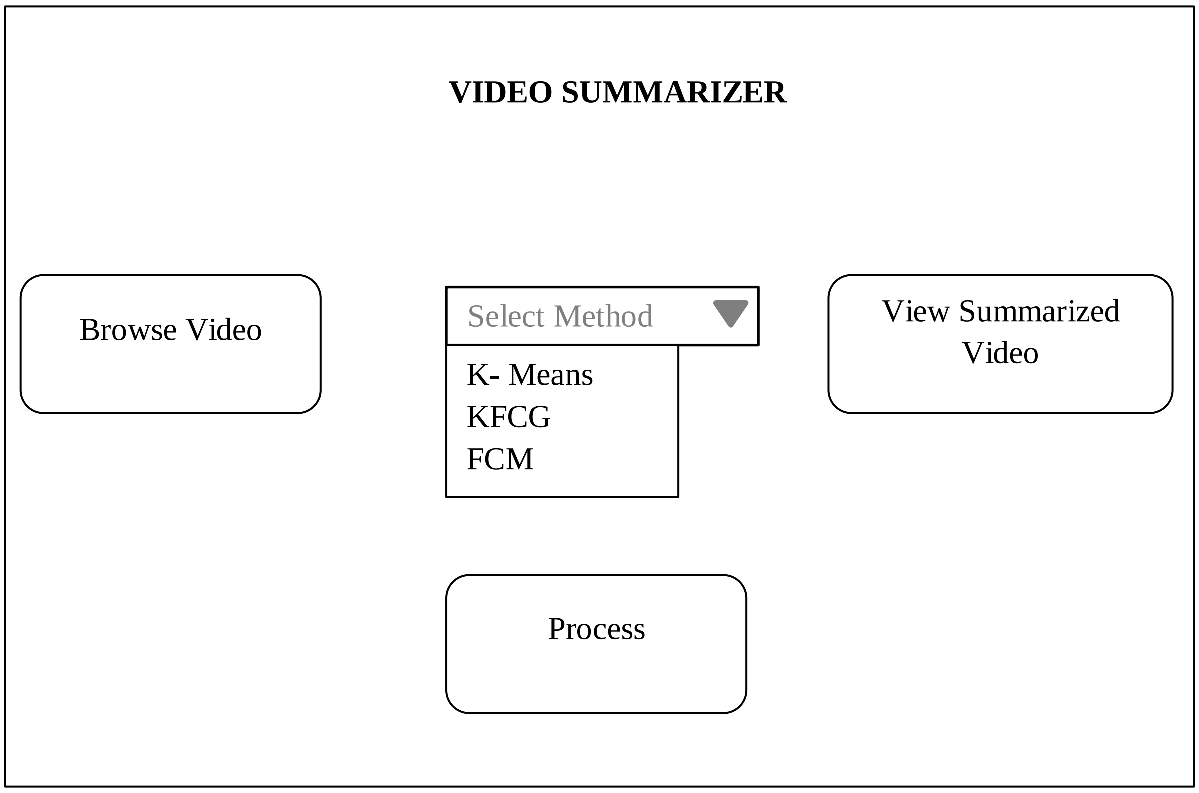


Fig. 5.2 Main Interface of the Video Summarizer

The tentative functions of the buttons used in Figure 5.2 is given below:

Browse Video: User browses for the video to be summarized. It provides a simple menu structure

Select Method: The user selects the clustering technique

Process: User clicks this button for summarizing the original video. A process bar must be present to show the user the status of processing. Figure 5.3 represents the process bar design for this system

View Summarized Video: The output summary should be played. The analysis of the processing time should also be displayed. Figure 5.4 gives a rough design of the analysis

The process bar for the video summarization system is given in Figure 5.3. This process bar must be displayed as soon as the user presses the process button and must be closed as and when the processing of the video is finished.

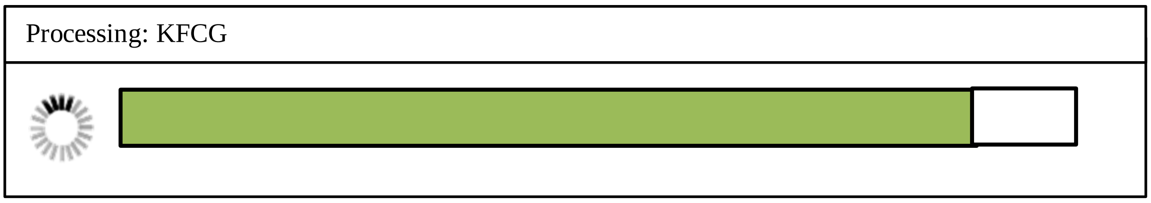
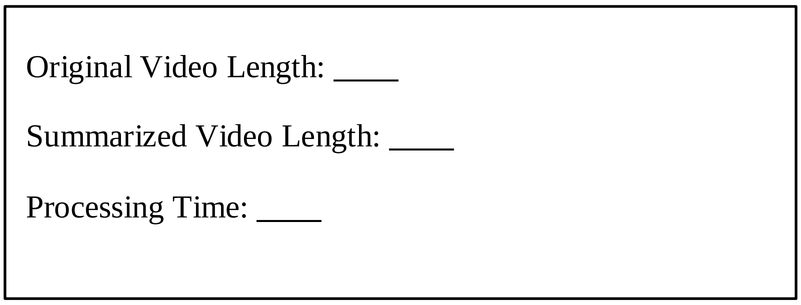


Fig. 5.3 Process Bar

The layout of the analysis chart is given in Figure 5.4. This chart must be comprehensive and should give the quantitative values of all the major parameters involved.


Original Video Length: ____

Summarized Video Length: ____

Processing Time: ____

Fig. 5.4 Design of analysis to be displayed after summarizing

The system must also show all the representative frames of the initial video and the summarized video. They can be displayed during processing

Chapter VI

Implementation

As discussed in Section 5.2, the video summarization system goes through the following phases: input acquisition, segmentation, feature selection, clustering, segment selection and video generation.

The video summarization system is implemented in MATLAB. MATLAB takes video only in AVI or MP4 format. Thus, the input acquisition phase restricts the video to be in these two formats only. MATLAB provides various functions for getting the properties of the video. These functions are used widely in the implementation of this system.

After getting the input video, the system segments the video. The segment length is determined by Equation 2. The first frame is chosen as the representative frame. The representative frames are stored in a dictionary.

The representative frames are converted into grey scale for easy processing. The histograms of these frames are computed in the feature selection phase. MATLAB provides an in-built function named “imhist” for determining the histogram of a given image. Value vectors for the representative frames are computed using Equation 1.

The value vectors form the data set for the clustering phase. MATLAB provides built-in functions for K-Means and FCM clustering techniques.

- The kmeans function takes a pre-defined number of clusters and the data set as its input parameters. The kmeans function returns an n-by-1 vector containing the cluster indices of each point

- The fcm function is the MATLAB function for FCM clustering. The input arguments of this function are: the data set to be clustered and the number of clusters. This function returns complex arguments as output. The output arguments of this function are:

- Centre: Matrix of final cluster centres where each row provides the centre coordinates
- U: Fuzzy Membership Matrix
- Obj_fcn: Values of the objective function during iterations

The segment selection phase iterates through the clusters in round robin fashion. In every iteration, the highest valued vector is chosen. The “checkvalue” function given in Section 6.6 maps the chosen representative vector to its corresponding segment.

In the last phase, the chosen segments are ordered as per the input video ordering and are written into a video file.

In this chapter, Sections 6.1-6.7 gives the MATLAB codes of the important routines.

MATLAB was chosen due to the following reasons:

- The graphical output is optimized for interaction. You can plot your data very easily, and then change colours, sizes, scales, etc. by using the graphical interactive tools
- Vectorized operations can be performed easily
- Gives pre-defined functions for K-Means and FCM
- Provides a very simple and easy-to-understand syntax
- Allows object oriented programming
- Provides a very-large database of built-in algorithms for image processing and computer vision applications
- Allows to call external libraries, such as OpenCV
- Provides easy environment for working with videos

6.1 VideoMain

- Takes input video from user
- Determines the segment size
- Temporally Segments the video
- Extracts the representative frames from each segment

```
global ExPath d l k framerate1 numFrames1 numFramesWritten11;

%Extracting & Saving of frames from a Video file through Matlab Code%
d.setValue(0.1);
l=6;

% assigning the name of sample avi file to a variable
filename1 = ExPath;

%reading a video file
mov = VideoReader(filename1);
framerate1=mov.FrameRate;
k1=round(mov.duration/60);
k=k1*2;

% Defining Output folder as 'snaps'
opFolder = 'C:\VideoSumm\snaps';

%if  not existing
if ~exist(opFolder, 'dir')
%make directory & execute as indicated in opfolder variable
mkdir(opFolder);
end

%getting no of frames
numFrames1= mov.NumberOfFrames;

%setting current status of number of frames written to zero
numFramesWritten11 = 0;
t=1;

%for loop to traverse & process from frame '1' to 'last' frames
for m = 1 :k*framerate1 :numFrames1
```

```
    numFramesWritten11 = numFramesWritten11 + 1;
    currFrame = read(mov, m);    %reading individual frames
    opBaseFileName = sprintf('%d.png', t);
    opFullFileName = fullfile(opFolder, opBaseFileName);
    imwrite(currFrame, opFullFileName, 'png');   %saving as 'png' file
    t=t+1;
end

d.setValue(0.3);
VideoSum;
```

6.2 VideoSumKMeans

- Plots histogram of the representative frames
- Computes value vectors based on the histograms
- Clusters the vectors based on K-Means Algorithm
- Selects highest valued vectors from the clusters in a round-robin fashion
- Extracts entire segments of the value vectors and generates output video

```
global d numFramesWritten11 FileName framerate1 processName imageNames1 k;

n=numFramesWritten11-1;
fd=FrameDef.empty;
dataset=zeros(n,1);
for i=1:n
    %accessing first frames of each segment
    name=strcat(num2str(i),'.png');
    name=strcat('C:\VideoSumm\snaps\',name);
    im1=imread(name);

    %histograms
    im1=rgb2gray(im1);
    h=imhist(im1); value=0;
    for t=1:256
```

```
        value=value+t*h(t);
    end;
    dataset(i)=value;
    fd(i)=FrameDef(name,value,0);
  end;

d.setValue(0.4);
[idx,C] = kmeans(dataset,k);
clust=zeros(n,k);
final=zeros(l,1);

for j=1:k
    for o=1:n
        if idx(o)==j
    clust(o,j)=dataset(o,1);
        end
    end
end

d.setValue(0.5);
rr=1; ss=1;
clust=sort(clust,'descend');
display(clust);
temp=1;
while true
    if(ss>l)
        break;
    end
    if (clust(temp,rr)~=0)
       pos=checkValue(clust(temp,rr),n,fd); %find the value in object array
       if(fd(pos).flag==0)                  %flag to avoid repeatability of frames
        final(ss)=clust(temp,rr);
        fd(pos).flag=1;
        rr=mod(rr,k)+1;
        ss=ss+1;
        temp=1;
       else
           temp=temp+1;
       end;
    else
       temp=temp+1;
       continue;
    end
end

finalpos=zeros(1,l);
for i=1:l
    finalpos(i)=checkValue(final(i),n,fd);
end
sortedpos=sort(finalpos);

%displaying the sorted segments in a window
figure('units','normalized','outerposition',[0 0 1 1],'Name','selected Segments');
 for n = 1:length(sortedpos)
 f= fullfile(workingDir,'snaps', imageNames1{sortedpos(n)});         % it will specify images
names with full path and extension
 our_images = imread(f);                 % Read images
 subplot(10,10,n), imshow(our_images);              % Show all images
 title(strcat('Segment ',num2str(sortedpos(n))));
 end

d.setValue(0.6);
opFolder = 'C:\VideoSumm\snaps2';
    %if  not existing
    if ~exist(opFolder, 'dir')
    %make directory & execute as indicated in opfolder variable
    mkdir(opFolder);
   ttt=1;
 for i=1:l
   count=((sortedpos(i)-1)*k*framerate1)+1;
   count2=(count+k*framerate1);
   for m = count:count2
      %numFramesWritten11 = numFramesWritten11 + 1;
      currFrame = read(mov, m);    %reading individual frames

      opBaseFileName = sprintf('%d.png', ttt);
      opFullFileName = fullfile(opFolder, opBaseFileName);
      imwrite(currFrame, opFullFileName, 'png');   %saving as 'png' file
      ttt=ttt+1;
   end
end
d.setValue(0.9);

%storing snaps2 folder images in imagesNames to create a video
workingDir = 'C:\VideoSumm';
imageNames = dir(fullfile(workingDir,'snaps2','*.png'));
imageNames = {imageNames.name};

%sorting the imagesNames according to numeric names of images
for i = 1:length(imageNames)
out(i) = cellfun(@(x)str2double(regexp(x,'\d*\.\d*','match')),imageNames(i));
end
out=sort(out);
for i = 1:length(imageNames)
   imageNames(i)=cellstr(strcat(num2str(out(i)),'.png'));
end

 %assign name to output file
FileName=strcat(FileName,'_summarize.avi');
outputVideo = VideoWriter(fullfile(workingDir,FileName));
outputVideo.FrameRate = framerate1;
```

```
open(outputVideo)

for ii = 1:length(imageNames)
   img = imread(fullfile(workingDir,'snaps2',imageNames{ii}));
   writeVideo(outputVideo,img)
end

d.setValue(1);
close(outputVideo)
d.dispose();
msgbox('Processing Finished');

%remove directories
rmdir('C:\VideoSumm\snaps','s');
rmdir('C:\VideoSumm\snaps2','s');
processName='K means';
analysisOfProcess;
```

6.3 KFCG Function

- Takes the value vector data set and number of iterations as parameters
- Applies KFCG algorithm on the data set
- Returns the clusters in matrix form

```
function clust=kfcg(D,itr)
%global numFramesWritten11;
D=transpose(D);
Y=cell(power(2,itr),1);
A=cell(power(2,itr+1)-1,1);A{1}=D(1,:); l=1;

for i=0:itr-1
    for j=1:power(2,i)
        l=l+1; n1=l;
        p=floor(l/2);
        C1=1.1*mean(A{p});
        l=l+1;
        C2=0.9*mean(A{p}); n2=l;
        m=1; n=1;
        for k=1:length(A{p})
            b=A{p,1}(1,k);
            if abs(b-C1)>abs(b-C2)
                A{n1,1}(1,n)=b; n=n+1;
            else
                A{n2,1}(1,m)=b; m=m+1;
            end
        end
    end
end

q=power(2,itr);
for j=1:q
    Y{q-j+1}=A{l}; l=l-1;
end

Y = Y(~cellfun(@isempty, Y)); col=numel(Y);
for i=1:col
    M(i)=length(Y{i});
end
row=max(M)-1; clust=zeros(row,col);
for i=1:col
    for j=1:row
     if j<=length(Y{i})
         clust(j,i)=Y{i,1}(1,j);
     end
    end
end
```

6.4 VideoSumFCM

- Plots histogram of the representative frames and computes value vectors based on the histograms
- Finds the membership values of the value vector and clusters
- Assigns vectors to various clusters based on the membership values
- Selects highest valued vectors from the clusters in a round-robin fashion
- Extracts entire segments of the value vectors and generates output video

```
global d numFramesWritten11 FileName framerate1 mov processName imageNames1;

n=numFramesWritten11-1;
fd=FrameDef.empty;
dataset=zeros(n,1);
for i=1:n
   %accessing first frames of each segment
   name=strcat(num2str(i),'.png');
   name=strcat('C:\VideoSumm\snaps\',name);
   im1=imread(name);

   %histograms
   im1=rgb2gray(im1);
   h=imhist(im1);    value=0;
   for t=1:256
      value=value+t*h(t);
   end
   dataset(i)=value;
   fd(i)=FrameDef(name,value,0);
end;

d.setValue(0.4);
opts = [nan;nan;nan;0];
[center,U,obj_fcn] = fcm(dataset,k,opts);
```

```
idx=zeros(n,1);
for i=1:n
  [a,idx(i)]=max(U(:,i));
end
clust=zeros(n,k);
final=zeros(l,1);

for j=1:k
   for o=1:n
      if idx(o)==j
   clust(o,j)=dataset(o,1);
      end
   end
end

d.setValue(0.5);
rr=1;
ss=1;
clust=sort(clust,'descend');
temp=1;
while true
   if(ss>l)
      break;
   end
   if (clust(temp,rr)~=0)
     pos=checkValue(clust(temp,rr),n,fd); %find the value in object array
     if(fd(pos).flag==0)                 %flag to avoid repeatability of frames
      final(ss)=clust(temp,rr);
      fd(pos).flag=1;
      rr=mod(rr,k)+1;
      ss=ss+1;
      temp=1;
     else
```

```
            temp=temp+1;
        end;
    else
        temp=temp+1;
        continue;
    end
end

finalpos=zeros(1,l);
for i=1:l
    finalpos(i)=checkValue(final(i),n,fd);
end
sortedpos=sort(finalpos);

%displaying the sorted segments in a window
figure('units','normalized','outerposition',[0 0 1 1],'Name','selected Segments');
 for n = 1:length(sortedpos)
 f= fullfile(workingDir,'snaps', imageNames1{sortedpos(n)});        % it will specify images
names with full path and extension
 our_images = imread(f);                % Read images
 subplot(10,10,n), imshow(our_images);              % Show all images
 title(strcat('Segment ',num2str(sortedpos(n))));
 end

d.setValue(0.6);
opFolder = 'C:\VideoSumm\snaps2';
    if ~exist(opFolder, 'dir')
    %make directory & execute as indicated in opfolder variable
    mkdir(opFolder);
    end

ttt=1;
for i=1:l
    count=((sortedpos(i)-1)*k*framerate1)+1;
```

```
   count2=(count+k*framerate1);
   for m = count:count2
      %numFramesWritten11 = numFramesWritten11 + 1;
      currFrame = read(mov, m);    %reading individual frames
      opBaseFileName = sprintf('%d.png', ttt);
      opFullFileName = fullfile(opFolder, opBaseFileName);
      imwrite(currFrame, opFullFileName, 'png');   %saving as 'png' file
      ttt=ttt+1;
   end
end
d.setValue(0.9);

%storing snaps2 folder images in imagesNames to create a video
workingDir = 'C:\VideoSumm';
imageNames = dir(fullfile(workingDir,'snaps2','*.png'));
imageNames = {imageNames.name};

%sorting the imagesNames according to numeric names of images
for i = 1:length(imageNames)
out(i) = cellfun(@(x)str2double(regexp(x,'\d*\.\d*','match')),imageNames(i));
end
out=sort(out);
for i = 1:length(imageNames)
   imageNames(i)=cellstr(strcat(num2str(out(i)),'.png'));
end

%assign name to output file
FileName=strcat(FileName,'_summarize.avi');
outputVideo = VideoWriter(fullfile(workingDir,FileName));
outputVideo.FrameRate = framerate1;
open(outputVideo)

for ii = 1:length(imageNames)
  img = imread(fullfile(workingDir,'snaps2',imageNames{ii}));
  writeVideo(outputVideo,img)
end

d.setValue(1);
close(outputVideo)
d.dispose(); msgbox('Processing Finished');
%remove directories
rmdir('C:\VideoSumm\snaps','s'); rmdir('C:\VideoSumm\snaps2','s');
processName='FCM';
analysisOfProcess;
```

6.5 FrameDef Class

- Defines the properties of the representative frame
- Gives name to the representative frame
- Stores the value vector of the frame in hist
- Flag is a Boolean variable which checks if the current frame is included in the output or not
- Function fobj initialises the values of the class data

```
classdef FrameDef
   properties
   name
   hist
   flag
   end
   methods
      function fobg=FrameDef(n,h,f)
         fobg.name=n; fobg.hist=h; fobg.flag=f;
      end
   end
end
```

6.6 checkValue Function

Given the value vector as parameter, it returns the corresponding segment number

```
function pos=checkValue(value,n,f)
pos=0;
for i=1:n
   if f(i).hist==value
      pos=i; break;
   end
end
```

6.7 AnalysisOfProcesses

Generates the final analysis chart

```
global FileName mov processName
toc; f = figure('Visible','off');
FileName=strcat('C:\VideoSumm\',FileName);
mov2=VideoReader(FileName);
a=strcat('Processing time :',num2str(toc));
b=strcat('Orginal Length :',num2str(mov.duration));
c=strcat('Summmarized Length :',num2str(mov2.duration));
S=sprintf('Process : %s\n1.%s\n2.%s\n3.%s',processName,a,b,c);
eh = uicontrol('Style','text',...
   'Position',[200 200 200 100],...
   'String',S);
set(f,'Visible','on');
```

Chapter VII

Testing

Testing is the investigation conducted to provide stakeholders with information about the quality of the product or service under test. It provides an objective, independent view of the software to allow the users to understand and appreciate the risks of software system implementation.

There are generally four levels of testing: unit testing, integration testing, system testing and user acceptance testing. They are grouped by where they are added in the software system development process or by the level of specificity of the test. The hierarchical levels of testing are shown in Figure 7.1.

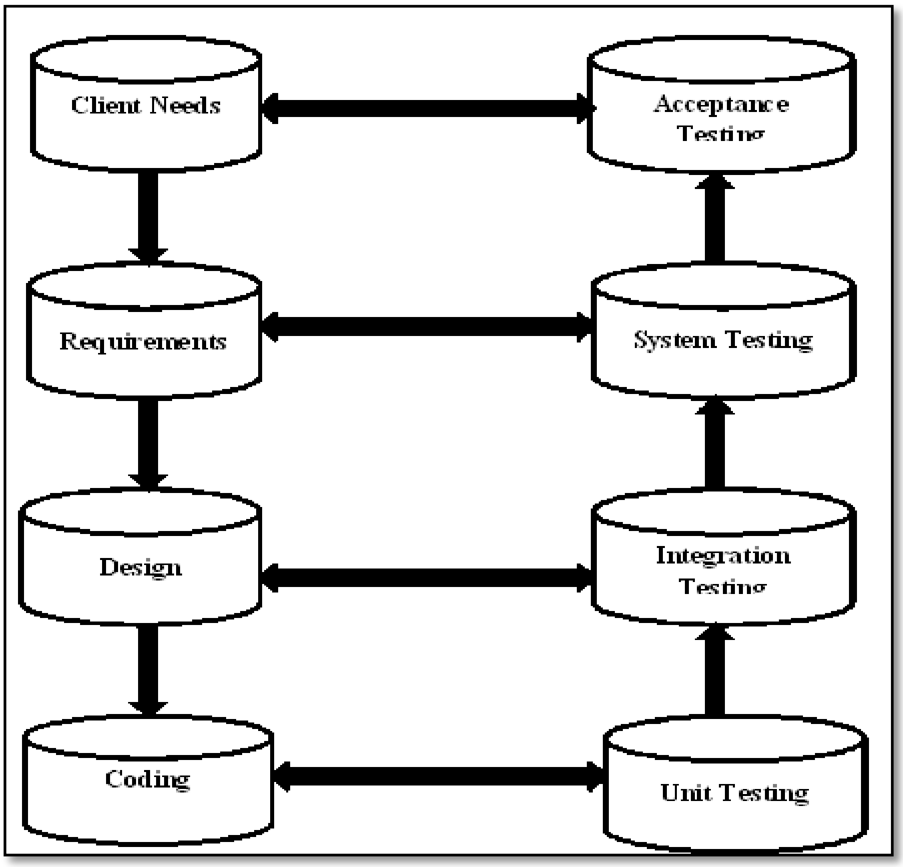


Fig. 7.1 Hierarchical Levels of Testing

Sections 7.1-7.3 describe the various testing levels for the video summarization system.

7.1 Unit Testing

Unit testing is a testing level in which the smallest parts of an application, called units, are individually and independently scrutinized for proper operation. The objective of this testing is to verify the correctness of individual units.

The various important units of the video summarization system are: KFCG clustering unit, FCM clustering unit and K-Means clustering unit. These units are tested with the help of a simple calling function. There needs to be three cases for testing these modules:

Case 1: Data set contains values which are spread over the spectrum. In this case, both closely spaced and far spaced items are included

Case 2: Closely spaced data items

Case 3: Far spaced data items

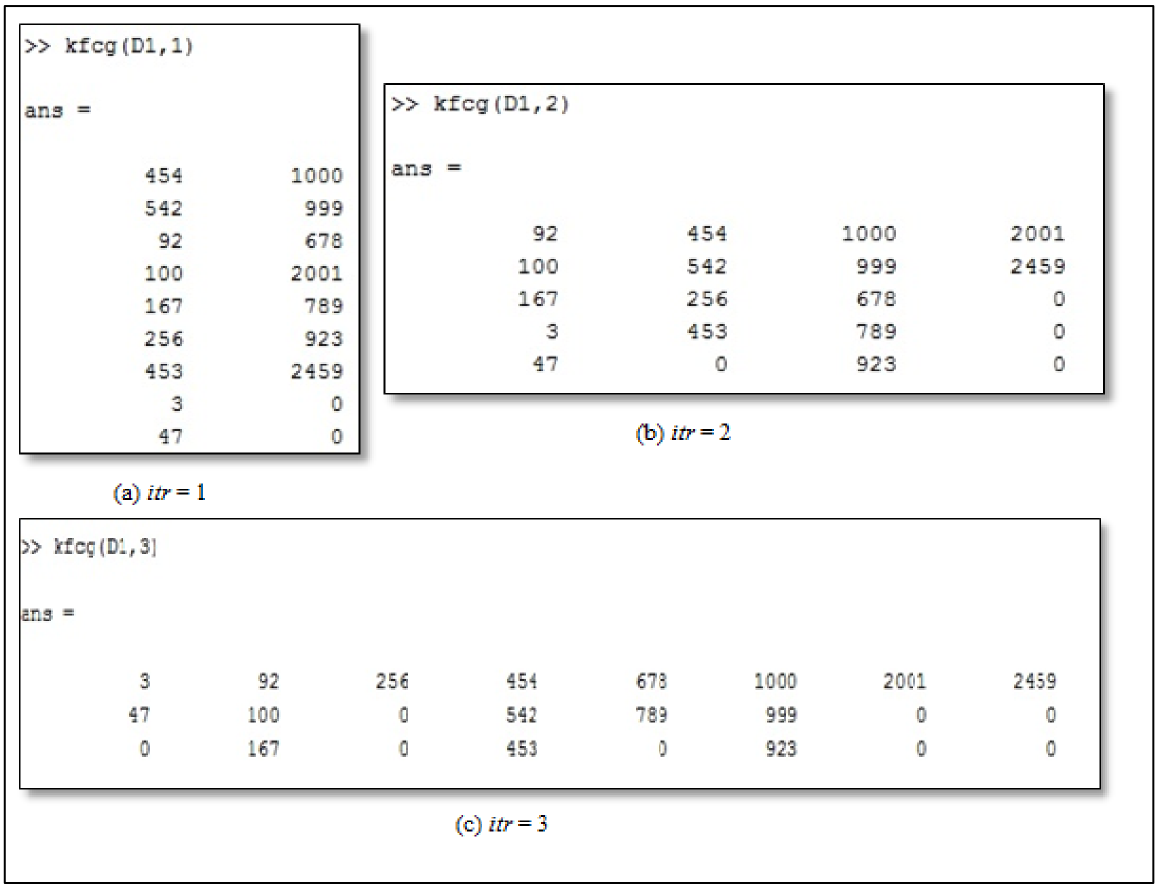
Based on these cases, the following data sets are taken:
D1 = {1000, 999, 678, 454, 2001, 789, 542, 92, 100, 167, 256, 453, 923, 3, 47, 89, 2459}
D2 = {1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1012, 1013}
D3 = {5, 100, 500, 1000, 2500, 6000, 10000, 12500}

Sections 7.1.1 and 7.1.2 test the units of the video summarization system. All the units are tested for D1, D2 and D3. Testing of FCM module is not shown explicitly as even it is an in-built function like K-Means.

7.1.1 KFCG Module

The kfcg function takes the data set, *D* and the number of iterations, *itr* as parameters. This function always forms 2^{itr} clusters. For all the three data sets, we test for three iterations: 1, 2 and 3. For 1 iteration, two clusters should be formed; for 2 iterations, four should be formed and for 3 iterations, eight should be formed. The clusters should be represented in a matrix form where the column indicates the cluster and row indicates the data item.

Figures 7.2 -7.4 shows the various testing of kfcg function.



```
>> kfcg(D1,1)

ans =

   454    1000
   542     999
    92     678
   100    2001
   167     789
   256     923
   453    2459
     3       0
    47       0
```

(a) *itr* = 1

```
>> kfcg(D1,2)

ans =

    92     454    1000    2001
   100     542     999    2459
   167     256     678       0
     3     453     789       0
    47       0     923       0
```

(b) *itr* = 2

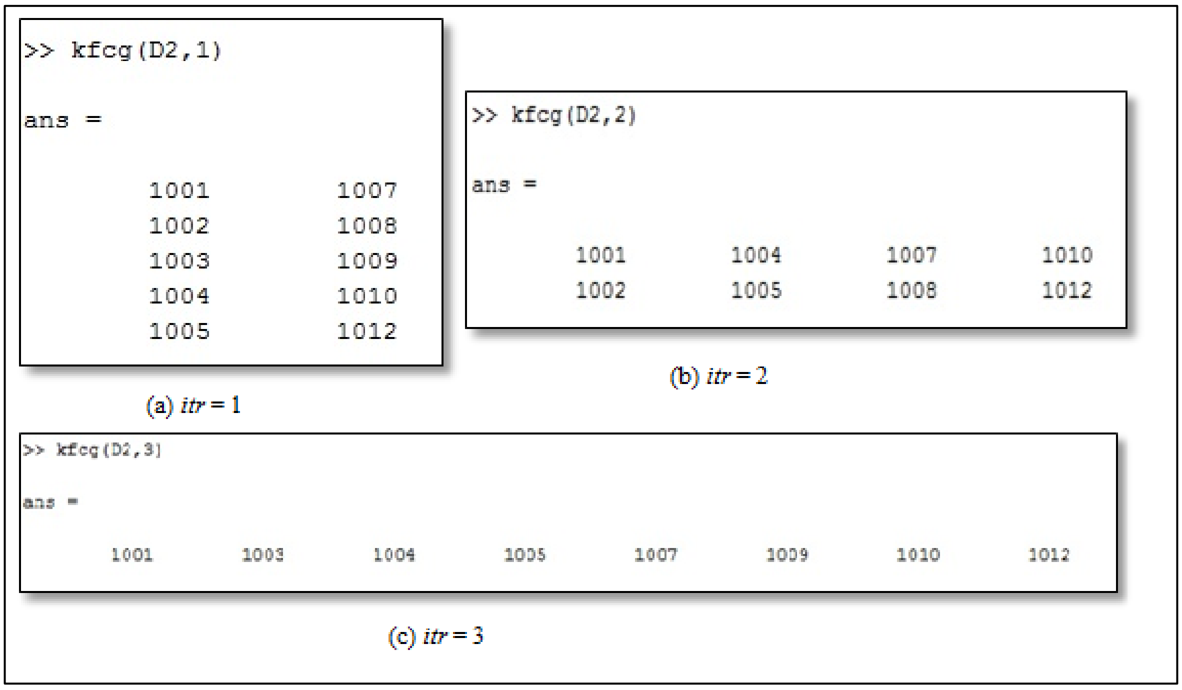
```
>> kfcg(D1,3)

ans =

     3      92     256     454     678    1000    2001    2459
    47     100       0     542     789     999       0       0
     0     167       0     453       0     923       0       0
```

(c) *itr* = 3

Fig. 7.2 KFCG testing for D1



```
>> kfcg(D2,1)

ans =

   1001    1007
   1002    1008
   1003    1009
   1004    1010
   1005    1012
```

(a) *itr* = 1

```
>> kfcg(D2,2)

ans =

   1001    1004    1007    1010
   1002    1005    1008    1012
```

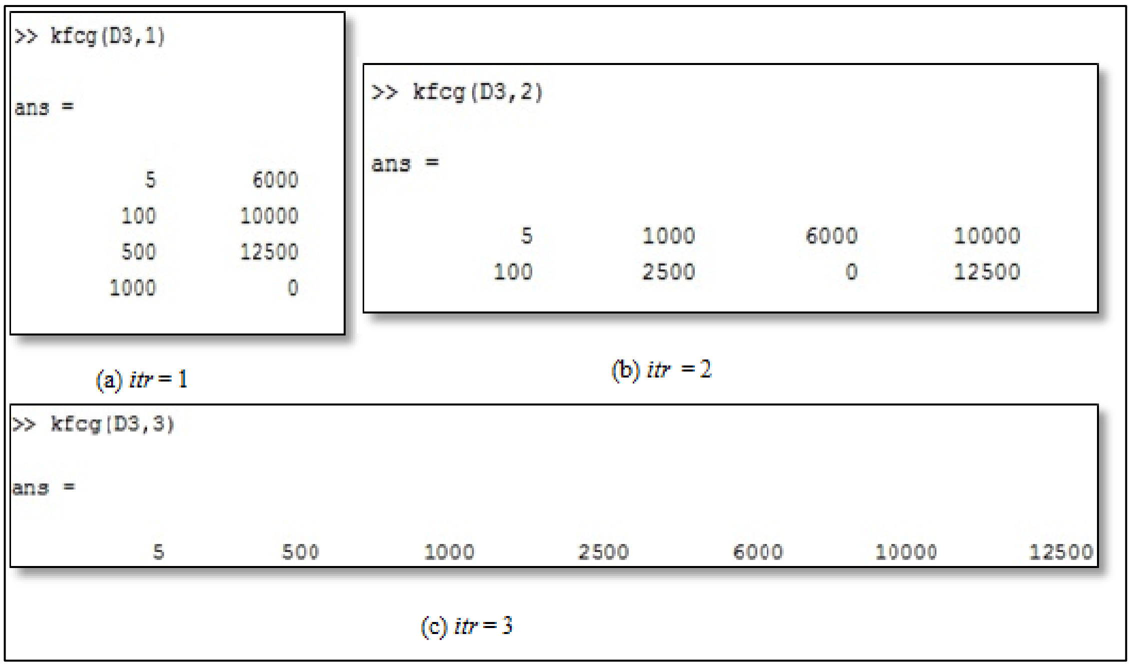
(b) *itr* = 2

```
>> kfcg(D2,3)

ans =
   1001    1003    1004    1005    1007    1009    1010    1012
```

(c) *itr* = 3

Fig. 7.3 KFCG testing for D2



```
>> kfcg(D3,1)

ans =

     5        6000
   100       10000
   500       12500
  1000           0
```

(a) *itr* = 1

```
>> kfcg(D3,2)

ans =

     5        1000        6000       10000
   100        2500           0       12500
```

(b) *itr* = 2

```
>> kfcg(D3,3)

ans =

     5         500        1000        2500        6000       10000       12500
```

(c) *itr* = 3

Fig. 7.4. KFCG testing for D3

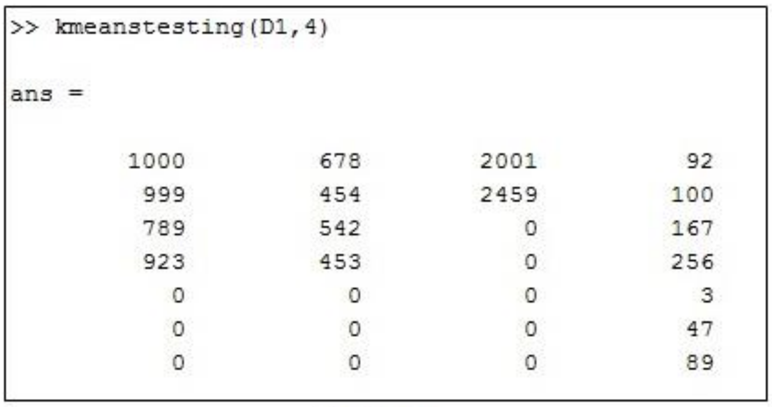
By comparing the actual outputs and manually computed outputs, it is evident that the kfcg function is correct.

7.1.2 K-Means Module

MATLAB provides an in-built function for K-Means. This function takes data set and the number of clusters as input; and returns a row vector containing the cluster indices. A function is developed to display the data items in clusters. This function returns a 2 x 2 matrix, where the column corresponds to the cluster and row to the data item.

The testing of K-Means is also done on the data sets D1, D2 and D3. It is done only for two cluster values. The implementation would be correct if obtained output and manual calculations match.

Figures 7.5-7.7 show the testing of kmeans module for the video summarization system.

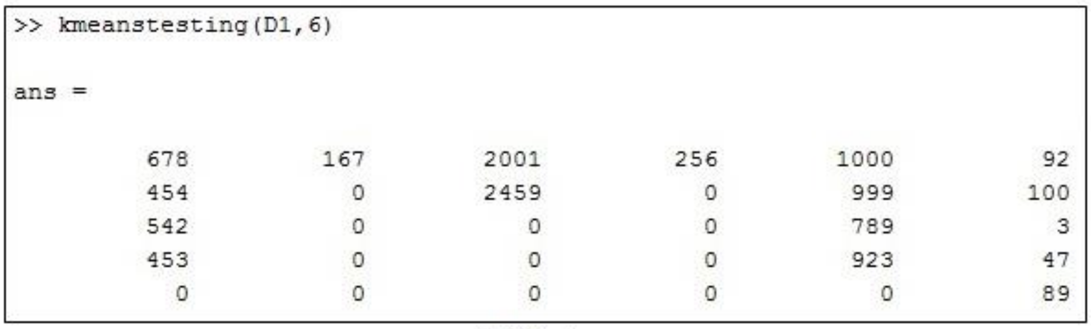


```
>> kmeanstesting(D1,4)

ans =

        1000         678        2001          92
         999         454        2459         100
         789         542           0         167
         923         453           0         256
           0           0           0           3
           0           0           0          47
           0           0           0          89
```

(a) 4 clusters



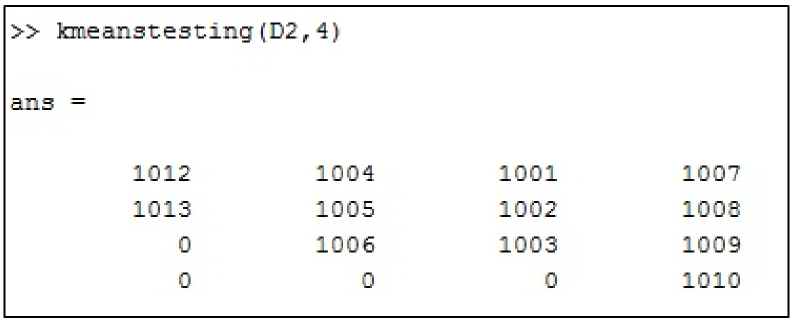
```
>> kmeanstesting(D1,6)

ans =

         678         167        2001         256        1000          92
         454           0        2459           0         999         100
         542           0           0           0         789           3
         453           0           0           0         923          47
           0           0           0           0           0          89
```

(b) 6 clusters

Fig. 7.5 K-Means testing for D1

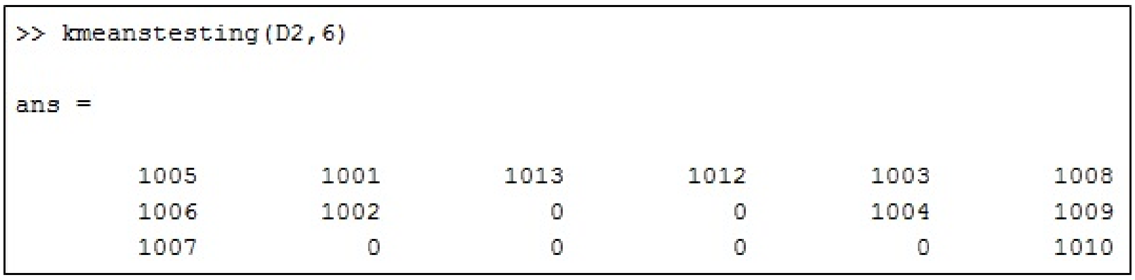


```
>> kmeanstesting(D2,4)

ans =

        1012        1004        1001        1007
        1013        1005        1002        1008
           0        1006        1003        1009
           0           0           0        1010
```

(a) 4 clusters



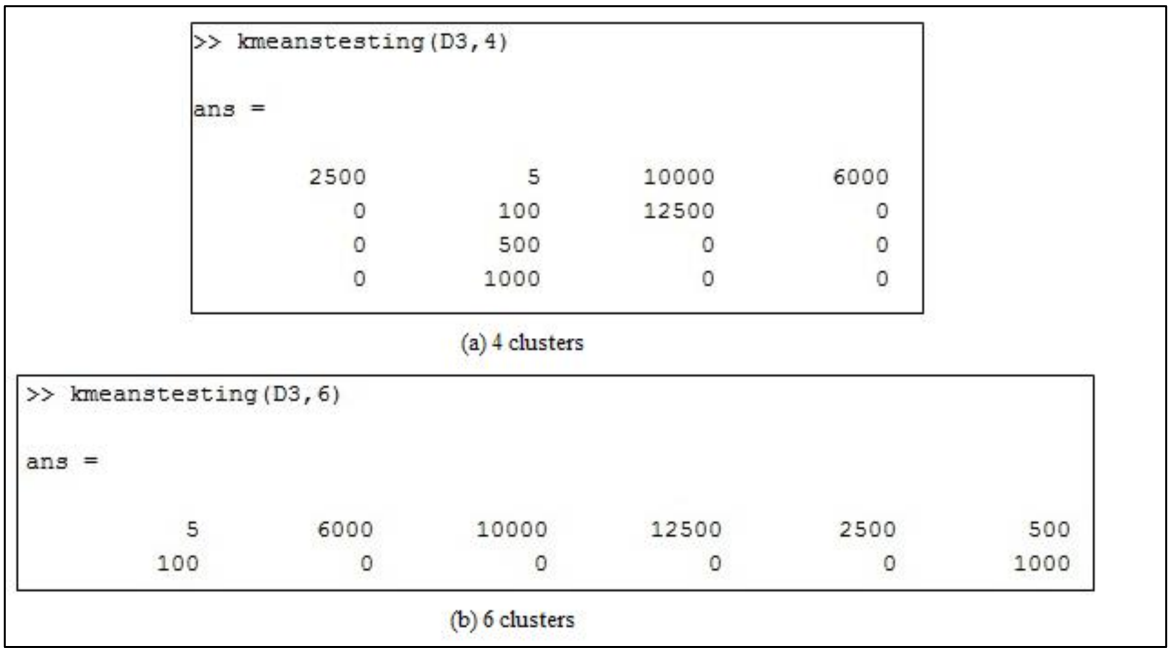
```
>> kmeanstesting(D2,6)

ans =

        1005        1001        1013        1012        1003        1008
        1006        1002           0           0        1004        1009
        1007           0           0           0           0        1010
```

(b) 6 clusters

Fig. 7.6 K-Means testing for D2



```
>> kmeanstesting(D3,4)

ans =

        2500           5       10000        6000
           0         100       12500           0
           0         500           0           0
           0        1000           0           0
```

(a) 4 clusters

```
>> kmeanstesting(D3,6)

ans =

           5        6000       10000       12500        2500         500
         100           0           0           0           0        1000
```

(b) 6 clusters

Fig. 7.7 K-Means testing for D3

From all the three tests, it is evident that K-Means function is also implemented correctly.

7.2 Integration Testing

In integration testing, the entire system is tested as a whole. Figures 7.8-7.15 show the snapshots of the final integrated system.

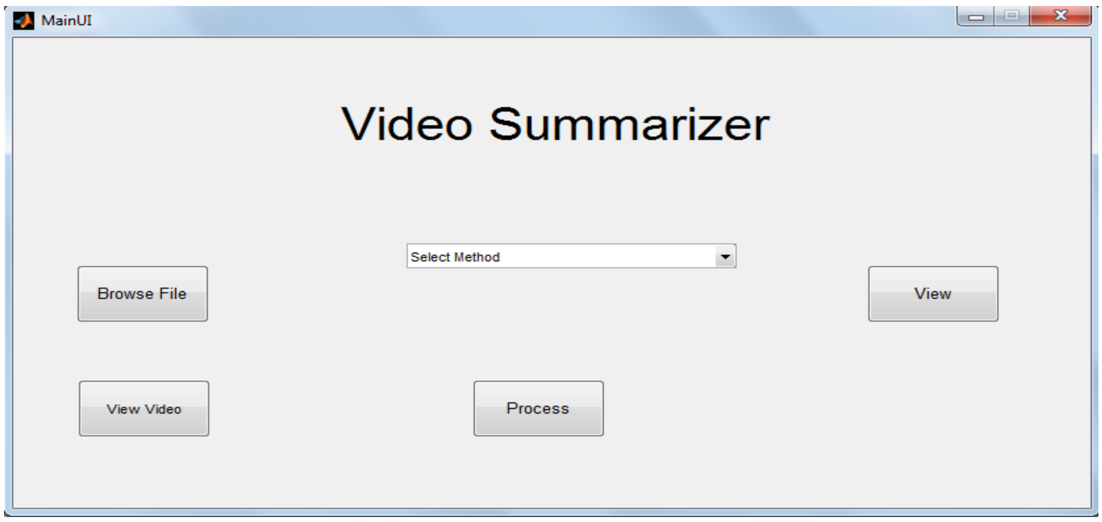


Fig. 7.8 Main Interface of the Video Summarizer

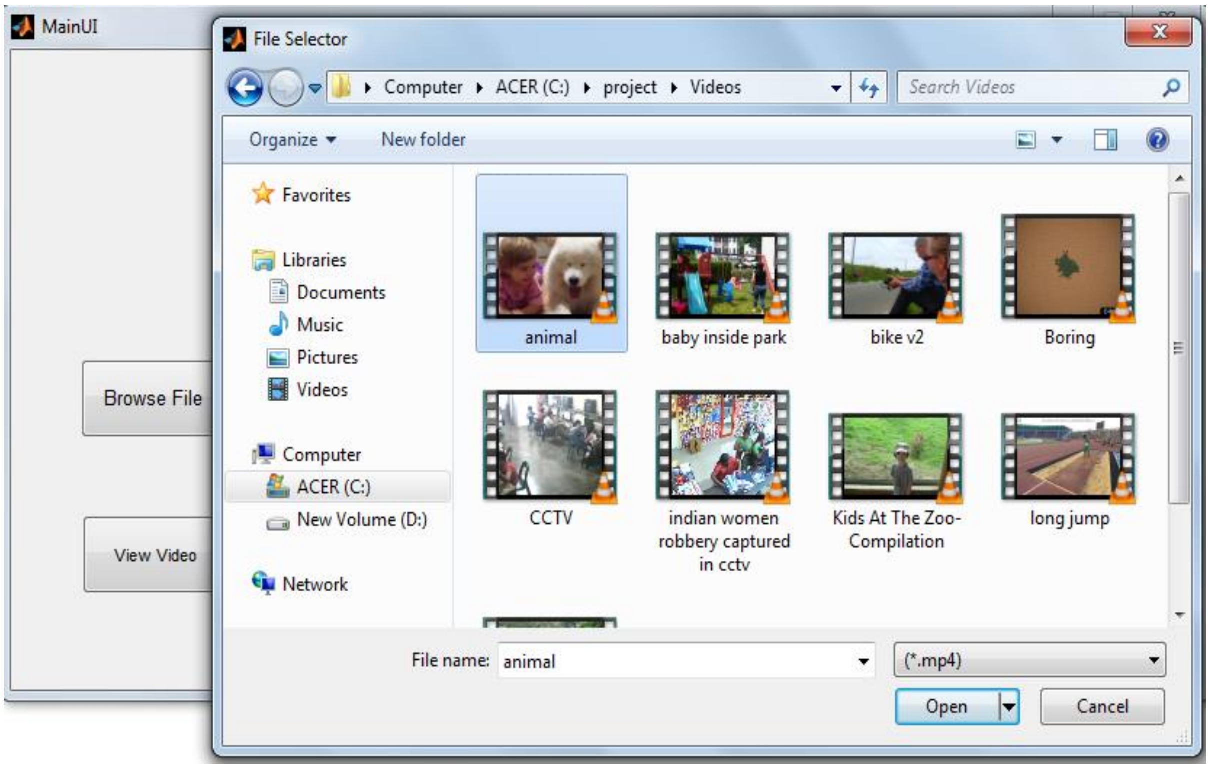


Fig. 7.9 Browsing the video

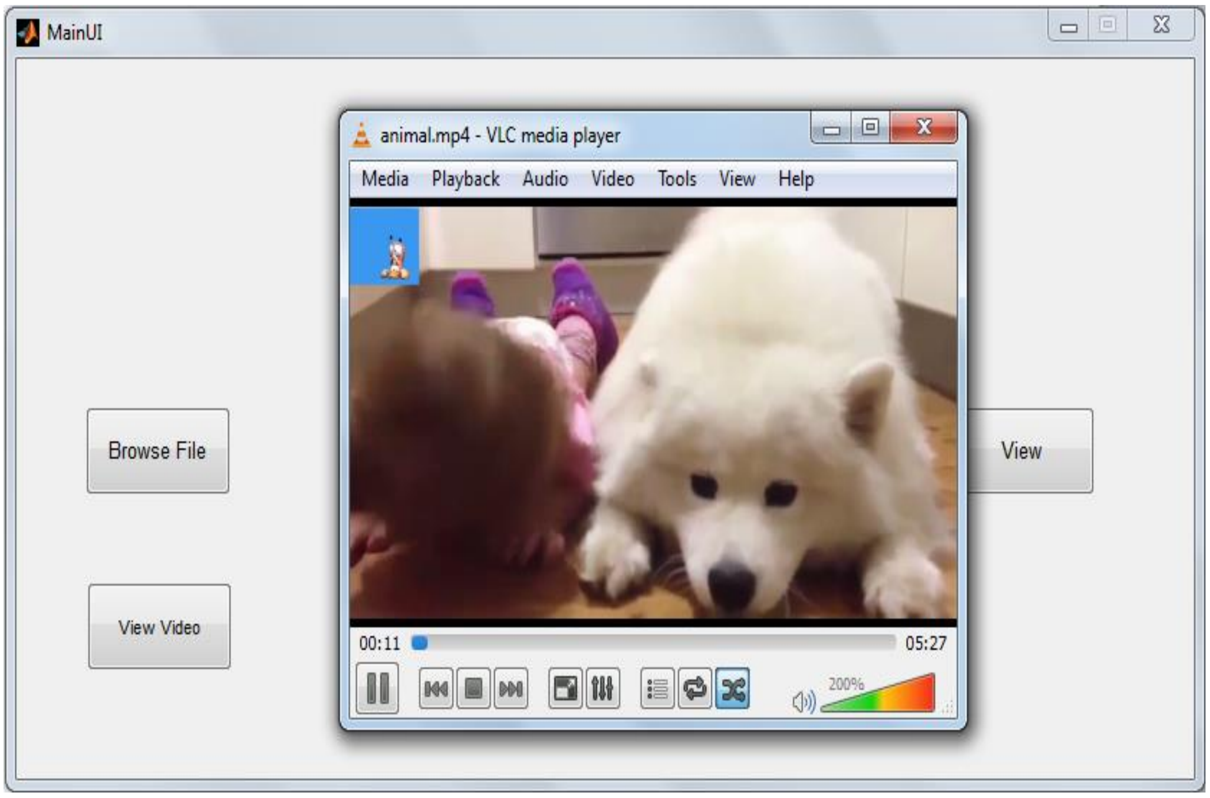


Fig. 7.10 Viewing the original video

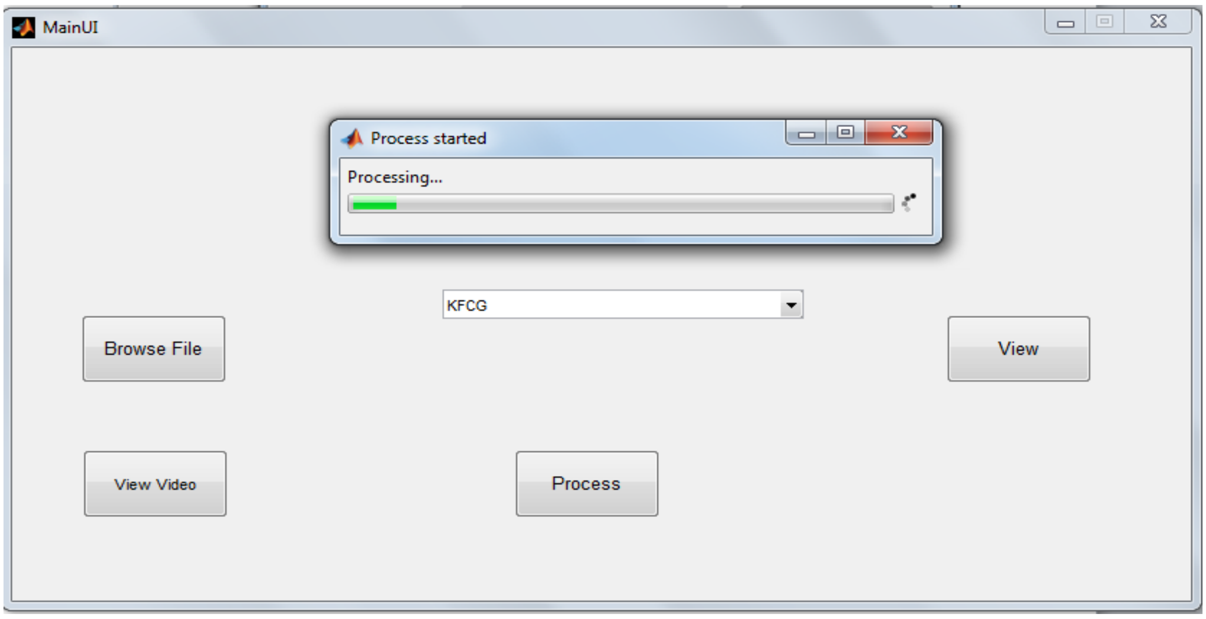


Fig. 7.11 Processing the original video

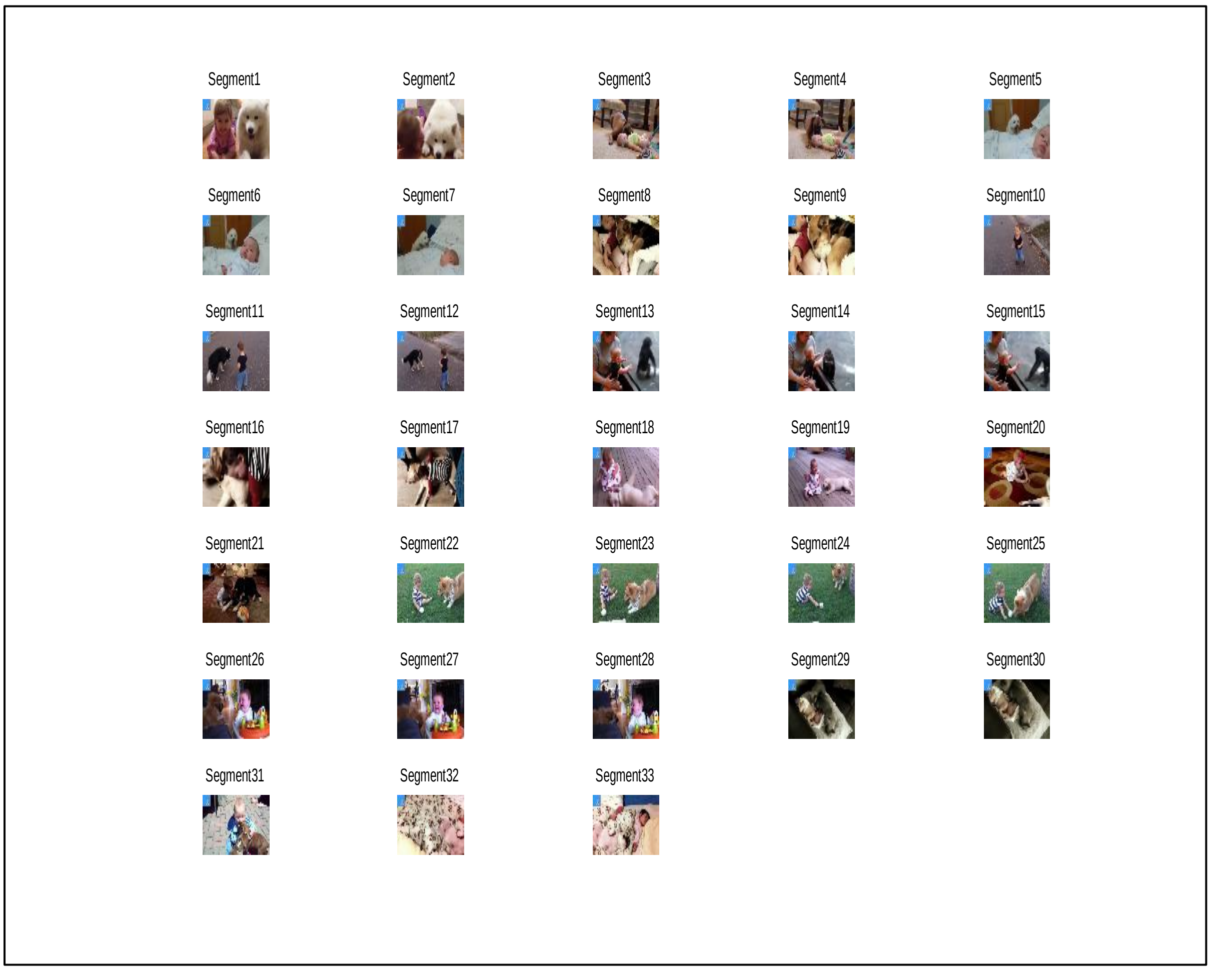


Fig. 7.12 Representative frames of all segments

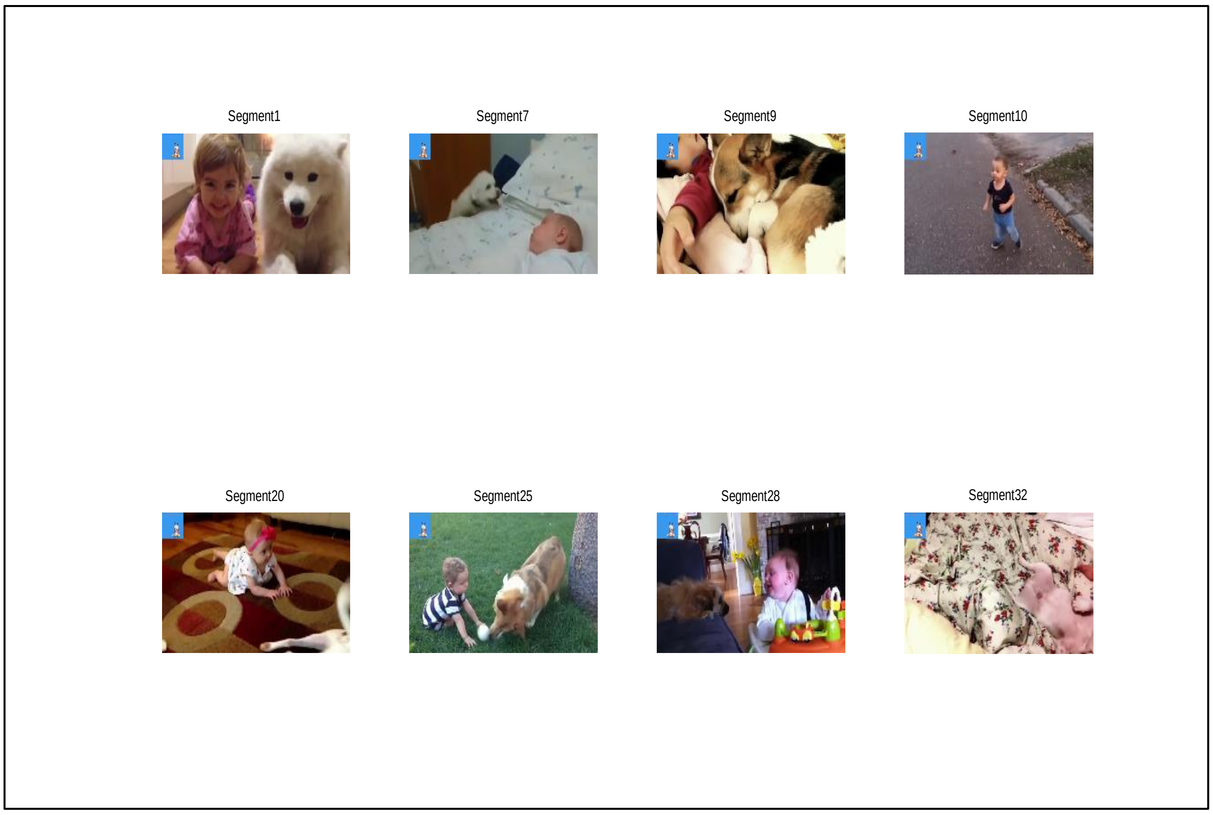


Fig. 7.13 Representative frames of selected segments

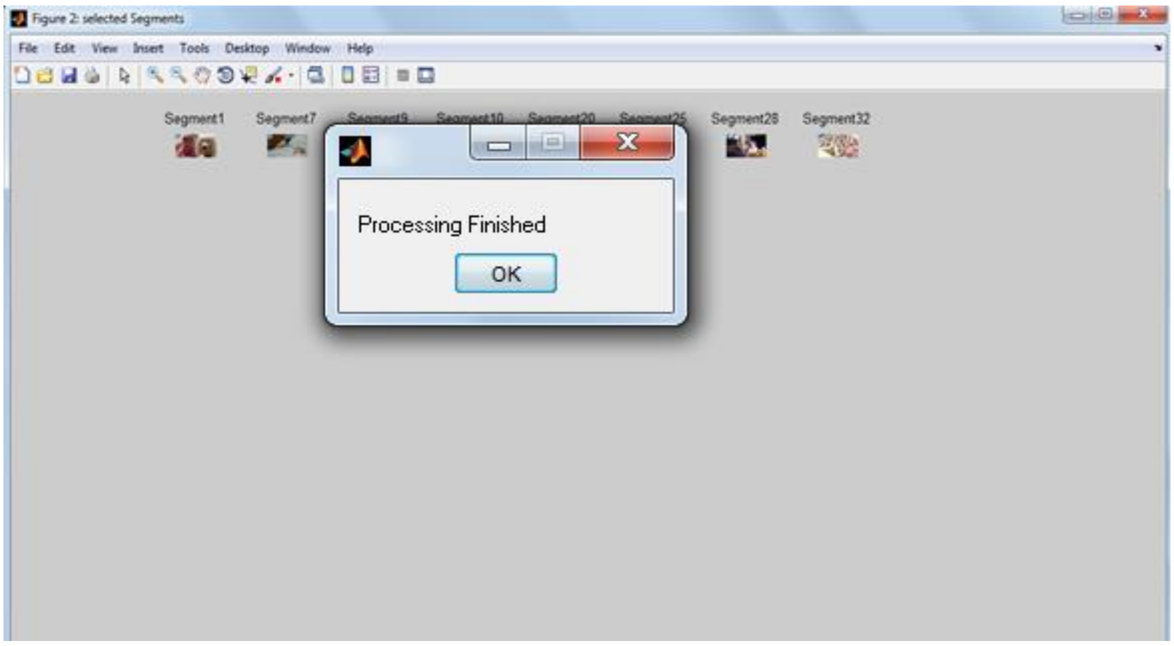
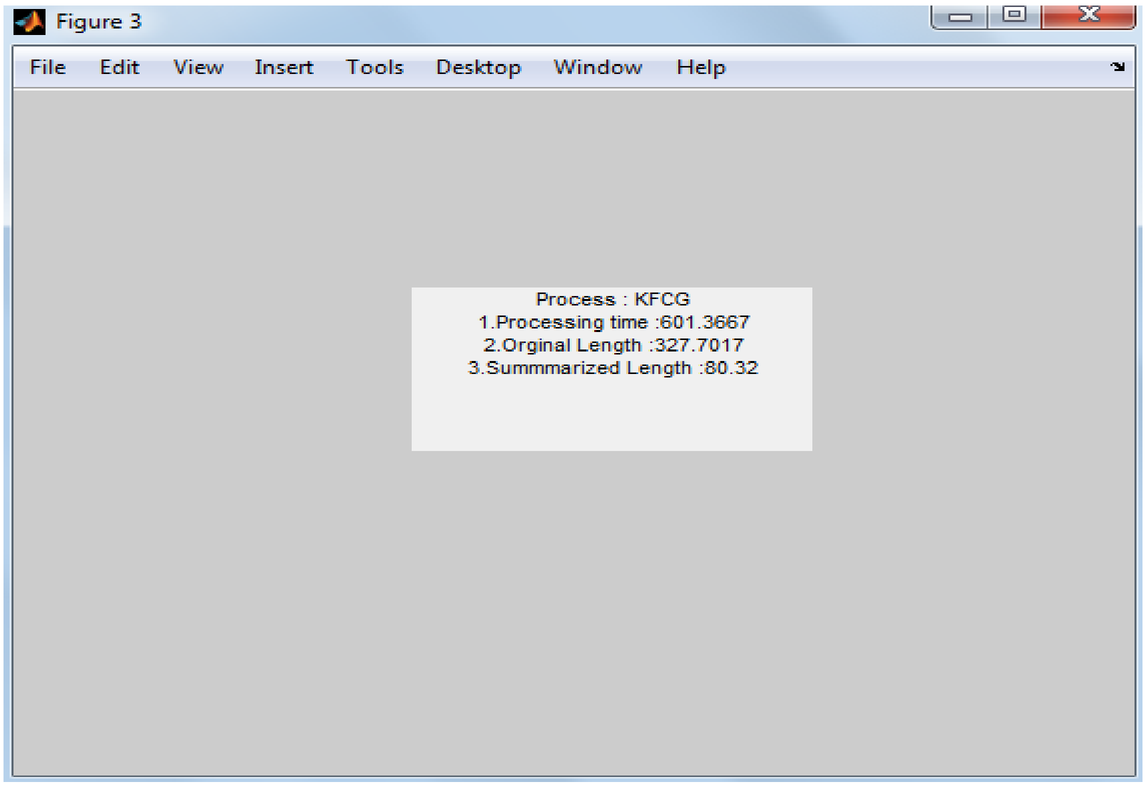


Figure 2: selected Segments
File Edit View Insert Tools Desktop Window Help
Segment1
Segment7
Segment9
Segment10
Segment20
Segment25
Segment28
Segment32
OK

Processing Finished

Fig. 7.14 After the completion of processing



```
Process : KFCG
1.Processing time :601.3667
2.Orginal Length :327.7017
3.Summmarized Length :80.32
```

Fig 7.15 Analysis chart

After integration testing, it is evident that the video summarization system is completely integrated. It was found that all the modules coordinate with each other efficiently.

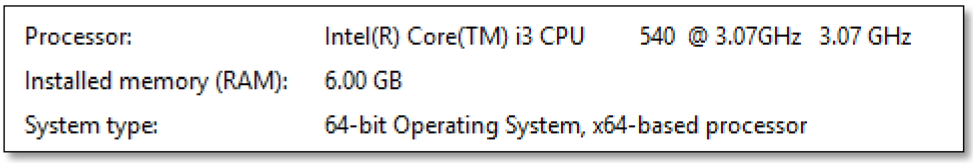
7.3 System Testing

In this type of testing, we understand the goals and requirements of the system and then develop test cases for various test scenarios and use cases.

For the video summarization system, processing and accuracy are the most important non-functional requirements of the user. Thus, the system testing for the video summarizer would aim to optimize these requirements.

In the video summarization system, the number of output segments and the number of clusters for each technique differs. Thus, the different combinations of these two variables would be considered as the different scenarios for test cases. The scenario giving the optimum results for each technique would be chosen.

Sections 7.3.1-7.3.3 perform the testing for K-Means, KFCG and FCM clustering respectively. The processing environment for all the three sections is given in Figure 7.16



Processor:	Intel(R) Core(TM) i3 CPU 540 @ 3.07GHz 3.07 GHz
Installed memory (RAM):	6.00 GB
System type:	64-bit Operating System, x64-based processor

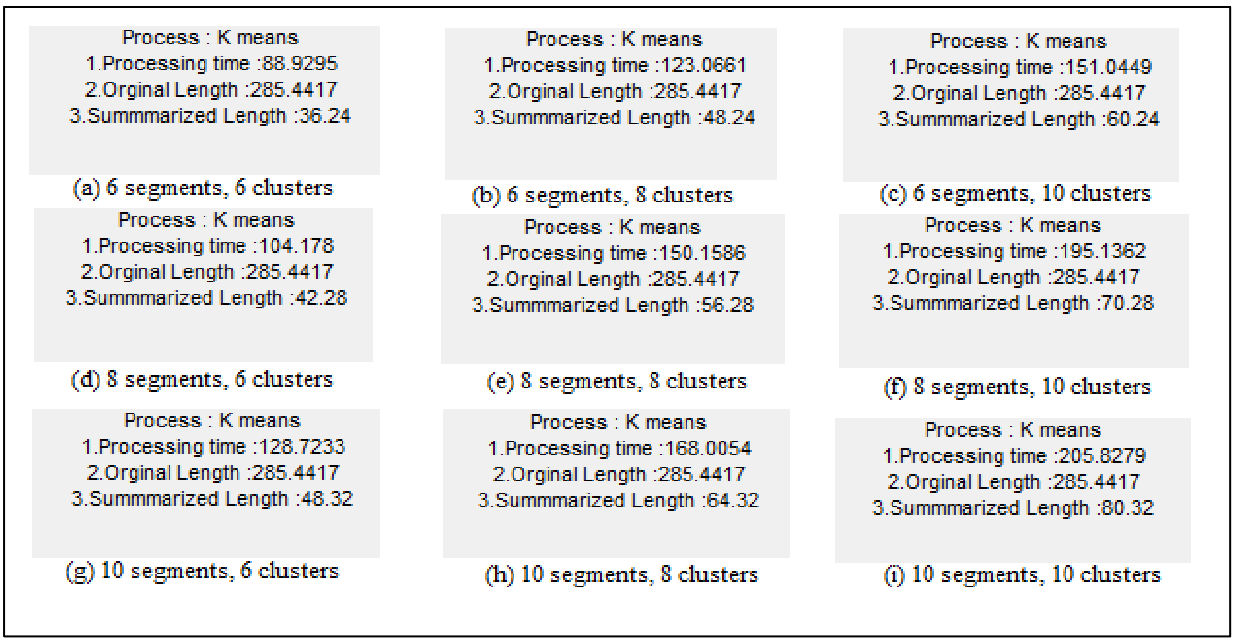
Fig. 7.16 Processing Environment for System Testing

The specification of the input video for all the three sections is as given:

- Video: v2.mp4
- Duration: 4:45 min
- FPS: 25
- Size: 24.2MB
- Converted to '.avi'

7.3.1 K-Means Test Scenarios

Three test cases are considered. Each case corresponds to the number of output segments. Each case again is divided into three sub-cases for 6, 8 and 10 clusters. Figure 7.17 shows all the test case scenarios.



```
Process : K means
1.Processing time :88.9295
2.Orginal Length :285.4417
3.Summmarized Length :36.24
```
(a) 6 segments, 6 clusters

```
Process : K means
1.Processing time :123.0661
2.Orginal Length :285.4417
3.Summmarized Length :48.24
```
(b) 6 segments, 8 clusters

```
Process : K means
1.Processing time :151.0449
2.Orginal Length :285.4417
3.Summmarized Length :60.24
```
(c) 6 segments, 10 clusters

```
Process : K means
1.Processing time :104.178
2.Orginal Length :285.4417
3.Summmarized Length :42.28
```
(d) 8 segments, 6 clusters

```
Process : K means
1.Processing time :150.1586
2.Orginal Length :285.4417
3.Summmarized Length :56.28
```
(e) 8 segments, 8 clusters

```
Process : K means
1.Processing time :195.1362
2.Orginal Length :285.4417
3.Summmarized Length :70.28
```
(f) 8 segments, 10 clusters

```
Process : K means
1.Processing time :128.7233
2.Orginal Length :285.4417
3.Summmarized Length :48.32
```
(g) 10 segments, 6 clusters

```
Process : K means
1.Processing time :168.0054
2.Orginal Length :285.4417
3.Summmarized Length :64.32
```
(h) 10 segments, 8 clusters

```
Process : K means
1.Processing time :205.8279
2.Orginal Length :285.4417
3.Summmarized Length :80.32
```
(i) 10 segments, 10 clusters

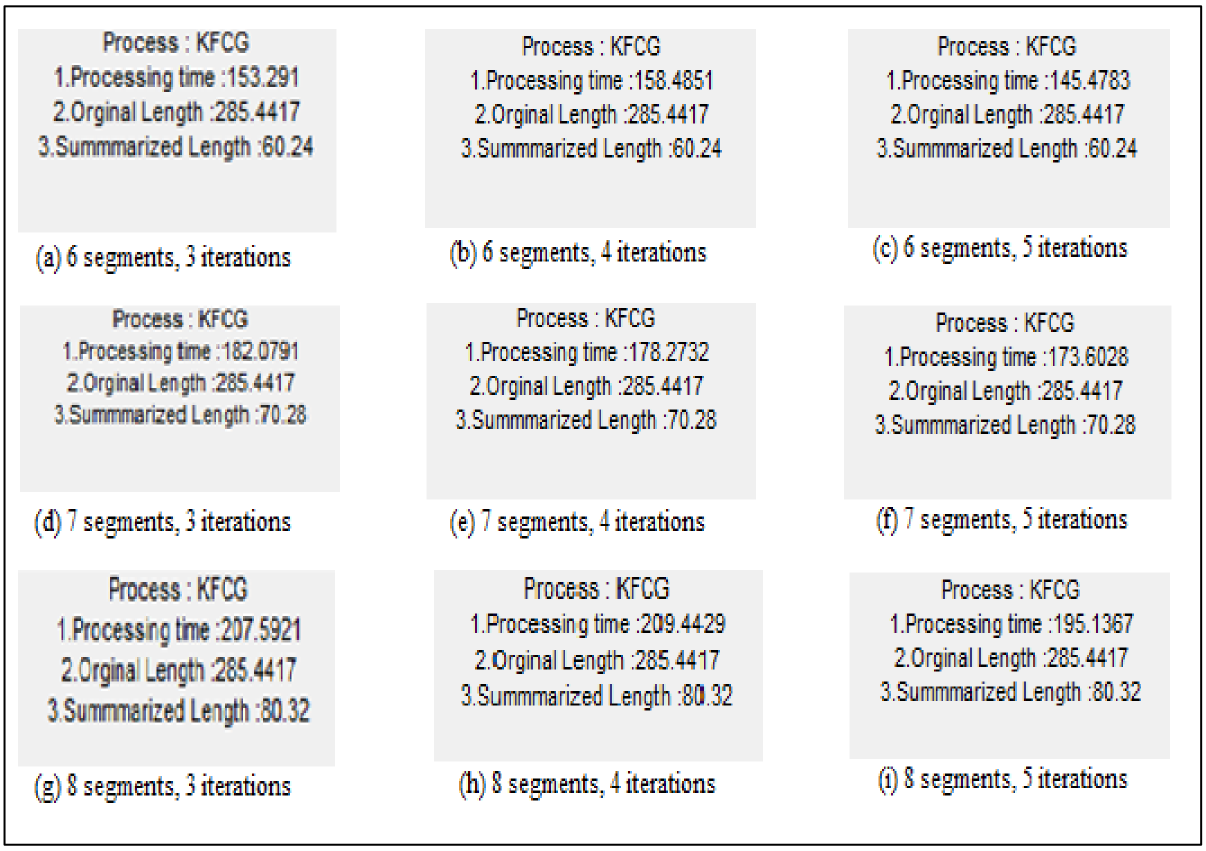
Fig. 7.17 Test case scenarios for K-Means clustering technique

The analysis for the above test case scenarios is as follows:

- For the technique with 6 output segments, the one with 6 clusters was found to be the fastest. But the 8 cluster method had very good accuracy as compared to both 6 and 10 cluster methods. Thus, it can be concluded that 8 is the optimum number of clusters for 6 output segments
- For the technique with 8 output segments, the one with 6 clusters was found to be the fastest. Even in this case, 8 clusters provided a great accuracy
- The observation for 10 output segments is similar to that of other cases
- Although 6 output segments was the fastest, there exists a speed v/s accuracy trade-off. Due to this, the 8 cluster-8 segment combination is finalised

7.3.2 KFCG Test Scenarios

The test case scenario for KFCG includes using different number of output segments and different number of iterations. Figure 7.18 gives the complete overview of the test cases.



Process : KFCG
1.Processing time :153.291
2.Orginal Length :285.4417
3.Summmarized Length :60.24

(a) 6 segments, 3 iterations

Process : KFCG
1.Processing time :158.4851
2.Orginal Length :285.4417
3.Summmarized Length :60.24

(b) 6 segments, 4 iterations

Process : KFCG
1.Processing time :145.4783
2.Orginal Length :285.4417
3.Summmarized Length :60.24

(c) 6 segments, 5 iterations

Process : KFCG
1.Processing time :182.0791
2.Orginal Length :285.4417
3.Summmarized Length :70.28

(d) 7 segments, 3 iterations

Process : KFCG
1.Processing time :178.2732
2.Orginal Length :285.4417
3.Summmarized Length :70.28

(e) 7 segments, 4 iterations

Process : KFCG
1.Processing time :173.6028
2.Orginal Length :285.4417
3.Summmarized Length :70.28

(f) 7 segments, 5 iterations

Process : KFCG
1.Processing time :207.5921
2.Orginal Length :285.4417
3.Summarized Length :80.32

(g) 8 segments, 3 iterations

Process : KFCG
1.Processing time :209.4429
2.Orginal Length :285.4417
3.Summmarized Length :80.32

(h) 8 segments, 4 iterations

Process : KFCG
1.Processing time :195.1367
2.Orginal Length :285.4417
3.Summmarized Length :80.32

(i) 8 segments, 5 iterations

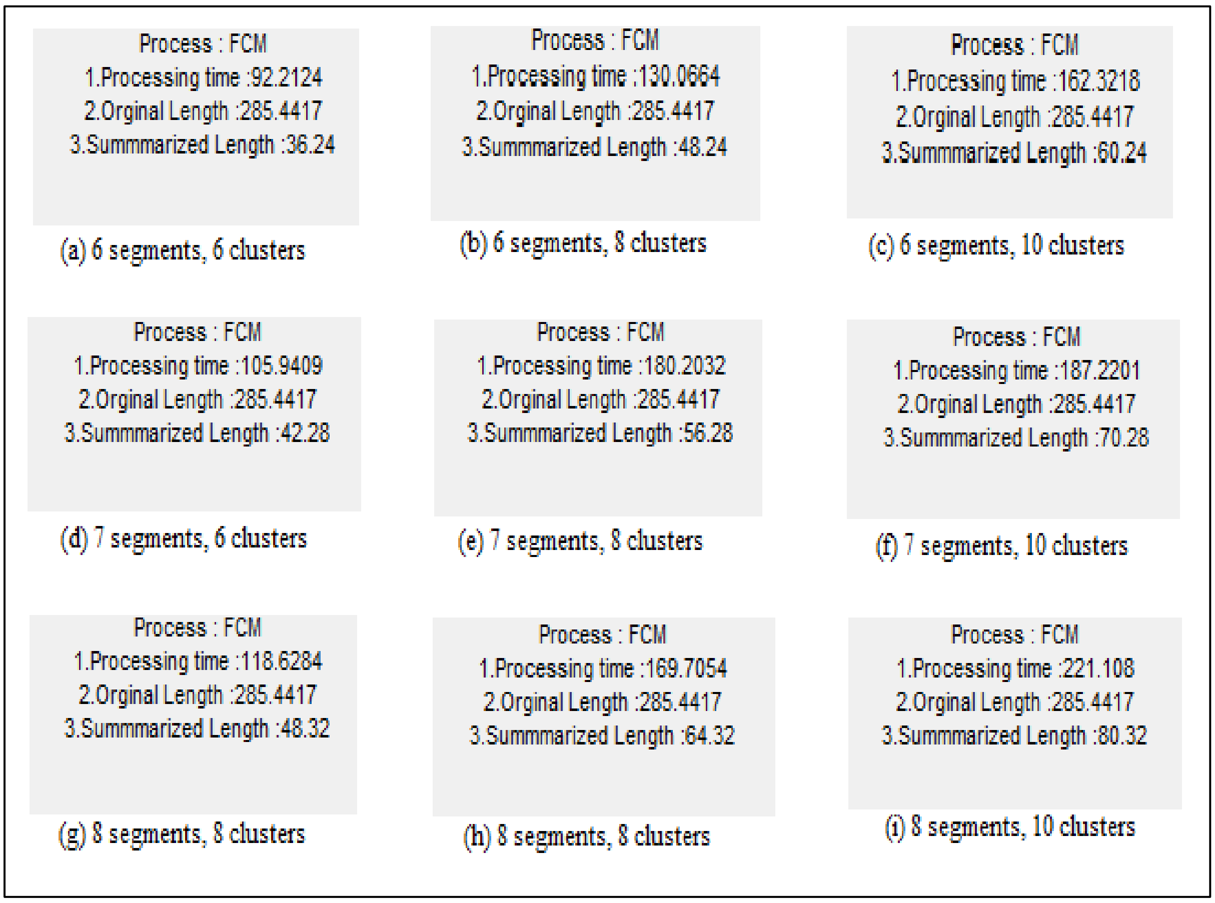
Fig. 7.18 Test case scenarios for KFCG clustering technique

The analysis for the test scenarios of KFCG clustering technique is as follows:

- For case 1, the one with 3 iterations was the fastest; but the one with 5 was the most accurate. As a result of which, we choose 4 iterations in order to compensate for the trade-off
- For case 2, the one with 3 iterations was both fast and accurate
- For case 3, the one with 4 iterations was the most accurate
- Keeping in mind all the various aspects, it can be inferred that the KFCG clustering algorithm with 7 output segments and 3 iterations is the best

7.3.3 FCM test Scenarios

FCM has a test scenario similar to K-Means. Figure 7.19 gives the various test scenarios for FCM clustering.



Process : FCM
1.Processing time :92.2124
2.Orginal Length :285.4417
3.Summmarized Length :36.24

(a) 6 segments, 6 clusters

Process : FCM
1.Processing time :130.0664
2.Orginal Length :285.4417
3.Summmarized Length :48.24

(b) 6 segments, 8 clusters

Process : FCM
1.Processing time :162.3218
2.Orginal Length :285.4417
3.Summmarized Length :60.24

(c) 6 segments, 10 clusters

Process : FCM
1.Processing time :105.9409
2.Orginal Length :285.4417
3.Summmarized Length :42.28

(d) 7 segments, 6 clusters

Process : FCM
1.Processing time :180.2032
2.Orginal Length :285.4417
3.Summmarized Length :56.28

(e) 7 segments, 8 clusters

Process : FCM
1.Processing time :187.2201
2.Orginal Length :285.4417
3.Summmarized Length :70.28

(f) 7 segments, 10 clusters

Process : FCM
1.Processing time :118.6284
2.Orginal Length :285.4417
3.Summmarized Length :48.32

(g) 8 segments, 8 clusters

Process : FCM
1.Processing time :169.7054
2.Orginal Length :285.4417
3.Summmarized Length :64.32

(h) 8 segments, 8 clusters

Process : FCM
1.Processing time :221.108
2.Orginal Length :285.4417
3.Summmarized Length :80.32

(i) 8 segments, 10 clusters

Fig. 7.19 Test case scenarios for FCM clustering technique

The analysis for FCM clustering is as follows:

- For all the three cases, the one with 6 clusters is the fastest and the one with 10 clusters is the most accurate
- So as to compensate for the trade-off of speed-accuracy, we choose 8 output segments and 8 clusters for the FCM technique

From the system testing, it is evident that all the non-functional requirements are aptly met by the video summarization system.

Chapter VIII

Results and Analysis

There are different types of videos in real life: some have lots of motion, some with lots of redundancy, some with a story, and so on. For proper analysis, five different categories of videos are taken. These categories are:

1. Animations

Documentaries may or may not have a story. But all animations do have a story. The main motive of summarizing such videos is to keep the original story intact in the summarized video.

2. Surveillance

It is also known as CCTV footages. Such videos are highly redundant, yet may contain important aspects like theft. Such videos when summarized must be extremely small, but still contain the highlighted aspect.

3. Documentaries

Documentaries are not as redundant as surveillance videos, but many a times they turn out to be boring. Unlike surveillance videos, they come along with audio. Audio is not considered while summarizing them.

4. Sports

Sports videos contain lots of motion. But having lots of motion does not mean that they have lots of meaningful content. While summarizing such videos, we will have to make sure that continuity is not affected.

5. Day-to-Day

They are the highly heterogeneous of the lot. Their characteristics are highly unpredictable. The only parameter which is considered for them is the processing time. Parents recording the play time with kids can be an example for the same

The minimum length of video was 10 seconds whereas the maximum video length was 11 minutes. About 25 videos were tested and 10 of them were analysed.

8.1 Analysis Based on Processing Time

Table 8.1 Observations

Category	**Video Name**	**Video Length**	**Processing Time**		
			K-Means	**KFCG**	**FCM**
Animations	Ben Franklin Effect	1:05	1:01	1:10	1:00
	Dino Killer	1:05	0:50	0:46	0:50
	Minecraft Eating	1:05	0:34	0:25	0:26
	Boring	1:48	3:06	3:00	2:55
	Flower of life	11:05	8:01	8:42	8:30
Surveillance	Robbery	1:43	1:32	1:15	1:16
	Animal	5:27	5:11	6:13	6:46
Documentaries	Why sitting is bad for you	5:04	4:54	4:50	5:00
	What makes a hero	4:33	2:00	1:48	1:57
Sports	V1	0:30	0:35	0:36	0:36
	Long jump	3:34	3:20	3:12	3:02
Day-to-Day	Baby in park	3:00	2:20	2:15	2:05
	Kids at zoo	5:45	3:45	3:20	3:15
	Pre-School Event	1:40	1:35	1:32	1:32

30% Compression Raton was achieved for all these videos. Table 8.1 gives the processing time taken for 10 videos in minutes.

Table 8.2 Ratio of Processing Time and Video Length

	K-Means	KFCG	FCM
Animations	0.936	0.924	0.896
Surveillance	0.922	0.935	0.989
Documentaries	0.703	0.675	0.707
Sports	1.051	1.048	1.025
Day-to-Day	0.793	0.749	0.726

Table 8.2 gives the average ratios of processing time and video lengths for all five categories. Figure 8.1 was computed based on Table 8.2

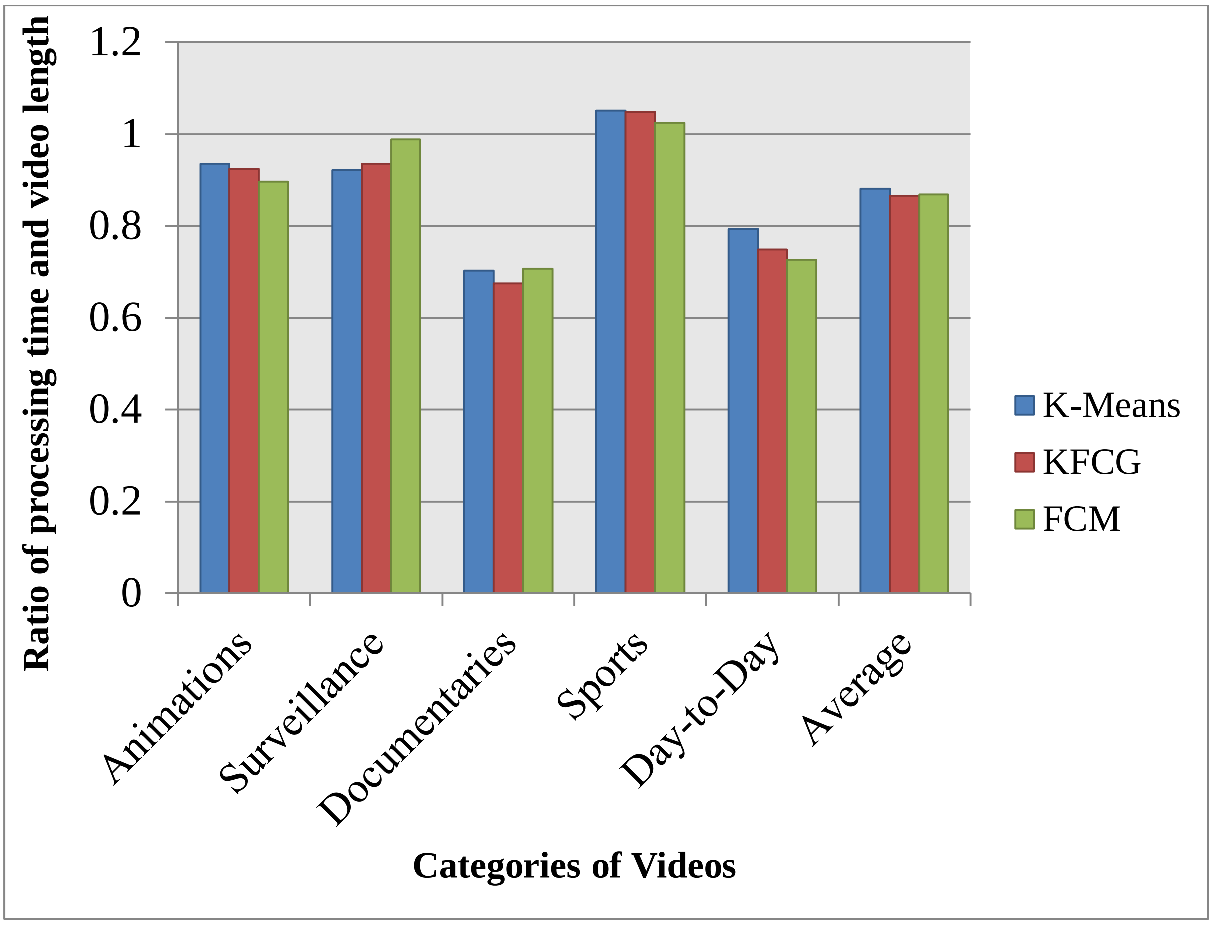


Fig. 8.1 Analysis graph for the video summarization system based on processing time

The time taken by all the clustering processes is nearly the same. K-Means takes 0.881 times the original length to process; KFCG takes 0.8662 times the original length to process; and FCM takes 0.8686 times the original video length to process. For most of the videos, KFCG was found to be the fastest.

K-Means is the slowest technique. The processing time is at least 0.8 times the original length. KFCG is faster than K-Means. FCM is also as fast as KFCG.

K-Means is found to be very effective for videos of shorter length. For videos with lengths less than 2 minutes, K-Means is the fastest. KFCG is the fastest for moderate length videos. FCM is the fastest for videos with large lengths.

8.2 Analysis based on Accuracy

Let the number of frames detected correctly be f_c. Let the total number of frames in the initial video be f_t. Then, the accuracy, A in Equation 5.

$$A = \frac{f_c}{f_t} \tag{5}$$

The video is first manually analysed. User manually checks the video and decides the frames which he feels must be a part of the output. These frames are then compared with the actual output frames.

Accuracy is analysed only for videos with some highlighted aspects or stories. CCTV summarization is accurate if the output summary consists of the highlighted aspects. Documentaries are summarized accurately if the order of important events is maintained. Animations are accurate if the stories are kept intact.

The accuracy for all the three techniques was nearly the same around 90%. CCTV videos were accurate in all the cases. Animations were accurately summarized using FCM. KFCG produced few deviations while summarizing animations. Documentaries were best summarized using KFCG. K-Means performed moderately in all the three cases.

Chapter IX

Conclusion and Future Work

The video generation in today’s era goes to enormous level. But watching such huge sequences of videos for understanding the core of it, is very time consuming. So this project work of video summarization helps in generating the summary videos faster. Summaries were generated using the clustering methods- Kmeans, KFCG and FCM. The speeds and accuracy of various clustering methods were compared. KFCG was found out to be the fastest and K-Means was the most accurate.

K-Means was faster for smaller videos as it takes the centroid faster. KFCG was slow as for small videos as it always have to construct a code book. FCM takes lesser time for bigger videos as it uses the concept of membership functions which can be computed faster

Accuracy in K-Means is higher as it is more or less similar to manual clustering. Accuracy of KFCG and FCM remains the same.

The current version, summarizes the video without taking into consideration the audio aspect of the movie. Future work would be summarizing the video that also includes audio.

The summarization speed can be increased further by making use of the GPU (Graphical Processing Unit). As a GPU consists of 100s of cores, which is far more than the cores present in a CPU, the expected result should be produced in far lesser time period. For specific applications, like video surveillance system, we can design a system which consists of a GPU. It would rely on the GPU only for its computations.

In the future, CBIR techniques could also be added for retrieving specific parts of the video. Clustering techniques are not that efficient as at every juncture of time only a selected number of previous frames are considered and not all. Dictionary can be used to overcome this drawback.

APPENDIX-A TERMINOLOGIES

1. **Segment**

Given an unstructured video, the system first starts with temporal segmentation i.e. breaking original video into segments. This is a bottom up approach as the video segments are formed based on similarity approach rather than boundary detection. Short temporal length of segments ensures consistency within each segment. Segments act as basic logical units in this system; they are analogous to key frames in static summarization methods.

2. **Value Vectors**

Every frame is represented by a feature vector. This feature vector is known as value vector. This is so because it computes the histogram values at each pixel.

3. **Representative Frame**

Every segment is represented by a single frame. This frame is known as the representative frame.

4. **Clustering**

Clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups.

5. **Effective Time**

Effective time is the sum total of processing time and video viewing time

APPENDIX-B ALGORITHMS

SUMMARIZATION ALGORITHM

i. Split the input file into time segments of *k* seconds

ii. Take the first frame of each segment. Let this frame be representative of the segment. We assign it $x_0...x_n$

iii. Compute the histograms from $x_0...x_n$ and assign it to $y_0...y_n$

iv. Cluster the histograms $y_0...y_n$ into *k* groups

v. Iterate through the *k* groups in round robin fashion and select a segment randomly from a cluster, add it to list *l* until the number of desired segments are chosen.

vi. Join list *l* of segments together to generate a video summary

K-MEANS CLUSTERING ALGORITHM

i. Initialize the center of the clusters

ii. Attribute the closest cluster to each data point

iii. Set the position of each cluster to the mean of all data points belonging to that cluster

iv. Repeat steps 2-3 until convergence

KFCG CLUSTERING ALGORITHM

i. Compute the centroid *C* of the vector set

ii. Add and subtract error vector to generate the vectors *C1* and *C2* as given in Equation B.1

$$C1 = C \times (1 + \alpha)$$
$$C2 = C \times (1 - \alpha) \quad \text{(B.1)}$$

iii. Compute distance between all training vectors belonging to this cluster and the vectors *C1* and *C2* and split the cluster into two based on the proximity

iv. Compute the centroid for the clusters obtained in step 3

v. Repeat steps 1-4 until required number of iterations are done

FUZZY C-MEANS CLUSTERING ALGORITHM

X = {x_1, x_2, x_3 ..., x_n}: Set of data points

V = {v_1, v_2, v_3 ..., v_c}: Set of centres

i. Randomly select *c* cluster centers

ii. Calculate the fuzzy membership μ_{ij} using Equation B.2

$$\mu_{ij} = 1 \Big/ \sum_{k=1}^{c} (d_{ij}/d_{ik})^{(2/m-1)} \quad \text{(B.2)}$$

iii. Compute the fuzzy centers v_j using Equation B.3

$$v_j = \frac{\sum_{i=1}^{n} \mu_{ij}^{m} x}{\sum_{i=1}^{n} \mu_{ij}^{m}} \quad \forall j = 1,2 \ldots . c \quad \text{(B.3)}$$

iv. Repeat steps 2-3 until the termination criterion is satisfied: $J < \beta$. The objective function J is computed using Equation B.4

$$J(U,V) = \sum_{i=1}^{n} \sum_{j=1}^{c} \mu_{ij}^{m} \, ||x_i - v_j||^2 \quad \text{(B.4)}$$

Bibliography

[1] Marat Fayzullin et. Al "The CPR Model for Summarizing Video" *Multimedia tools and Applications June* 2005

[2] RaviKansagara et. Al "A Study on Video Summarization Techniques" *IJIRCCE* Feb 2014

[3] Dirfaux F. "Key Frame Selection to represent a video" *IEEE* 2000

[4] Sabbar W et. Al "Video Summarization using shot segmentation and local motion estimation" *INTECH* Sept 2012

[5] Naveeb Ezaz "Adaptive key frame extraction for video summarization using aggregation mechanism" *Elsevier* 2012

[6] R.M. Jiang, A.H. Sadka, D. Crooks; "Advances in video summarization and skimming" *Springer* 2009

[7] Samy Bengio et. Al "Group Sparse Coding" *NIPS* 2009

[8] Piotr Dollar et. Al "Behaviour Recognition via Sparse Spatio-Temporal Features" *VS-PETS* 2005

[9] I. Lapdev "On space time interest points" *IJCV* 2005

[10] Navneet Dallal; Bill Triggs "Histogram of Oriented Gradient for Object Detection" *CVPR* 2009

[11] Rizwan Choudhary "Histogram of Oriented Optical Flow and Binet Cauchy Kernels on Non Linear Dynamic Systems for the Recognition of Human Actions" *CVPR* 2010

[12] Bin Zhao; Eric P. King "Quasi Real Time Summarization for Consumer Videos" *CVPR* June 2014

[13] Tommy Chheng "Video Summarization using Clustering" Department of Computer Science, University of California, *Citeseer* 2007

[14] Pavel Berkhin "Survey of Clustering Data Mining Techniques" pp. 25-71, 2002

[15] C C Aggarwal; C K Reddy “Data Clustering: Algorithms and Applications” CRC Press, Chapman & Hall Book

[16] KS Chuang; HL TZeng; S Chen “Fuzzy C-Means clustering with spatial information for image segmentation” *Elsevier* 2006

[17] H Frigui; R Krishnapuram “A robust competitive clustering algorithm with applications in computer vision” *IEEE* 1999

[18] Sony A; Ajith K; Thomas K; Thomas T “Video Summarization by clustering using Euclidean distance” *IEEE* July 2011

[19] Clustering Methods
Last modified on Feb 5, 2006 [Online]
Available at: http://users.ics.aalto.fi/sami/thesis/node9.html

[20] A tutorial on clustering algorithms: K-Means Clustering
Last modified on September 18, 2003 [Online]
Available at: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html

[21] Data clustering algorithms: Fuzzy c-means clustering algorithm
Last modified on April 21, 2015 [Online]
Available at: https://sites.google.com/site/dataclusteringalgorithms/fuzzy-c-means-clustering-algorithm

[22] Dr. H. B Kekre; Tanuja K. Sarode “New Clustering Algorithm for Vector Quantization using rotation of error vector” *IJCSIS*, vol 7, no. 3, 2010