CLUSTERING BASED VIDEO SUMMARIZATION

Submitted in partial fulfillment of the requirements for the degree of

B. E. Computer Engineering

By

Aditya Hegde	**25**
Srinidhi Raghavan	**60**

Supervisor:

Dr. Kavita Sonawane
Associate Professor

Dr. Tanuja K. Sarode
Assistant Professor

Department of Computer Engineering
St. Francis Institute of Technology
(Engineering College)

University of Mumbai
2014-2015

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 advent has resulted in them being used in day-to-day lives more commonly.

With this advent in the digital multimedia, a lot of digital content is widely available. Apart from this, the advances in digital content distribution have led to easy video recording. Be it in highly visual movies or news or documentaries or sports or anywhere and everywhere, videos act as a utility and try to facilitate our lives even further. Moreover, the need for surveillance has increased manifold due to increase in the demand for security. Thousands of video cameras can be found at banks, airports, railway stations, public places, buildings etc. resulting in large amount of information which is difficult to process in real time.

This rapid growth in such media has thus led to the need to process huge chunks of visual data. But despite the numerous advantages, there is a pit fall for the same advent. 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 plays an important part in this regard as it helps the user to navigate through and retrieve a large sequence of videos. It helps in efficient storage, quick browsing and retrieval of large collection of video data without losing important aspects.

The proposed and implemented system aims at comparing Video Summarization through three well known techniques. These techniques take a processing time which is just slightly greater than or equal to the original length of the 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 remove uninteresting contents.

1.2. Problem Formulation

Clustering is the most common and effective video summarization mechanism. In this system, we compare some of these clustering techniques. The system takes videos of both structured (movies) and unstructured (CCTV) nature.

Given a video, the video summarization scans through the main video stream and generates the segments temporally. The first frame of each segment is considered to represent the entire segment. 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 (Kekre's Fast Codebook Generation Algorithm for vector quantization) and Fuzzy C-Means. These algorithms return the clustered data set. Frames are chosen from the clusters in round robin manner and added to the summary video file.

1.3. Motivation

With the proliferation of automatic video acquisition devices and the development of storage technologies, digital libraries increasingly have large collections of digital videos. Consequently, it is an imperative task to effectively process these videos for users to easily browse and search them. Video summarization emerges as an effective technique for indexing and quick access of video content. By analysing the content of these videos, video summarization techniques use drastically reduced data to represent the original videos.

Due to various time and storage constraints, summarising videos becomes a necessity rather than an accessory. There are three main categories of techniques for video summarization: Key-frame based, video skimming and Image based semantic summarization. In key frame based techniques, a subset of representative frames that contain significant visual content are selected as video summaries. These representative frames are extracted using techniques like segmentation, clustering, etc. Segmentation has its advantage of giving summaries which are continuous in nature. On the other hand, clustering techniques aims at selecting the most valuable frames. In this system, we aim at implementing all these aspects under one roof.

As a typical example, let us consider a 4 hour CCTV footage which has to be carefully looked through for spotting a theft. Such long videos become too irritating to watch and often tend to induce boredom as the time passes. This is where the process of summarizing comes into effect. The main motive of video summarizing techniques is to value time and make it as short and crisp as possible within a smaller time span, albeit retaining the important aspects of the video. 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 (processing time + summarized video length).

These major flaws in the existing technologies motivated this project. This system imparts the knowledge of clustering for solving this issue. Here the system uses efficient clustering algorithms to remove the redundant aspects from the video. The system aims at optimizing the time taken to process any video.

1.4. Proposed and Implemented Solution

The video summarization process 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.

The system mimics this working of human brain. The proposed and implemented work is nothing but a clustering agent. It perceives the inputs of the video, which happens to be its environment. It creates clusters and assigns frames to these clusters. The redundant frames are not included in the output. It builds an initial dictionary of representative frames, and then clusters these representative frames. Highly valuable frames are chosen from these clusters in a round robin fashion. Corresponding segments of these frames forms the sequences of the output summary.

The main novelty with this approach is that the advantages of various summarization techniques are implemented under one roof. 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

Videos are used anywhere and everywhere. Let us consider a surveillance scenario. 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. This system might reduce this discrepancy by reducing the video effectively.

It can thus be found that the applications of this system are vivid- ranging from home-made videos to surveillance to even movies.

Keeping in mind the project tenure and the available resources, we restrict to simply demonstrate its effect on simple shorter home-made videos.

Chapter II

Literature Survey

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 system implements key features of Zhao model in the normal key-frame based set-up. It also tries to compare the processing time for summarization by various clustering techniques.

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 various methods- K-means, KFCG (Kekre's Fast Codebook Generation) and FCM (Fuzzy C-means clustering) – 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 in order to have a comfortable 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 mp4 format.

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. Nature of Video: The video needs to be unstructured for optimum results

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, we develop the use case diagram for this Quasi Real Time System is as given in Fig.3.1

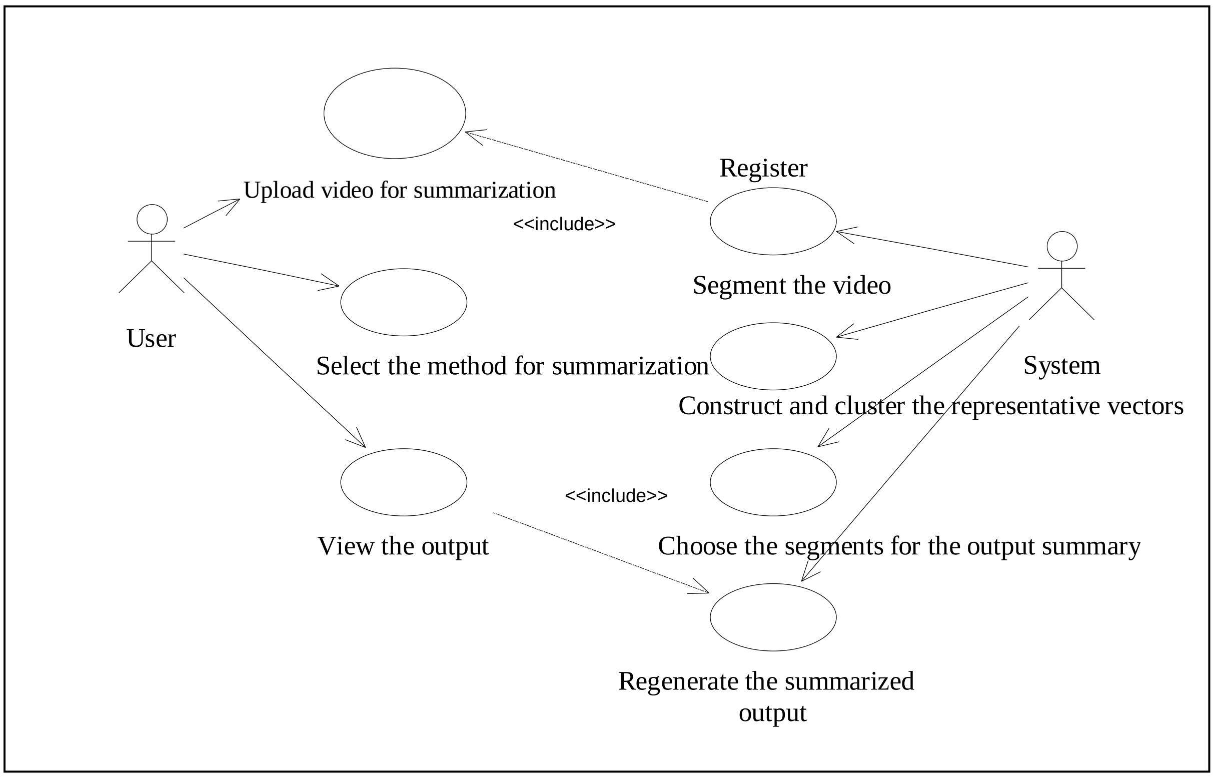

Upload video for summarization
Register
<<include>>
Segment the video
User
Select the method for summarization
System
Construct and cluster the representative vectors
<<include>>
View the output
Choose the segments for the output summary
Regenerate the summarized output

Fig.3.1. Use Case Diagram of the proposed system

Some of the use cases are as described below:

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 in
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

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.

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

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 (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

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

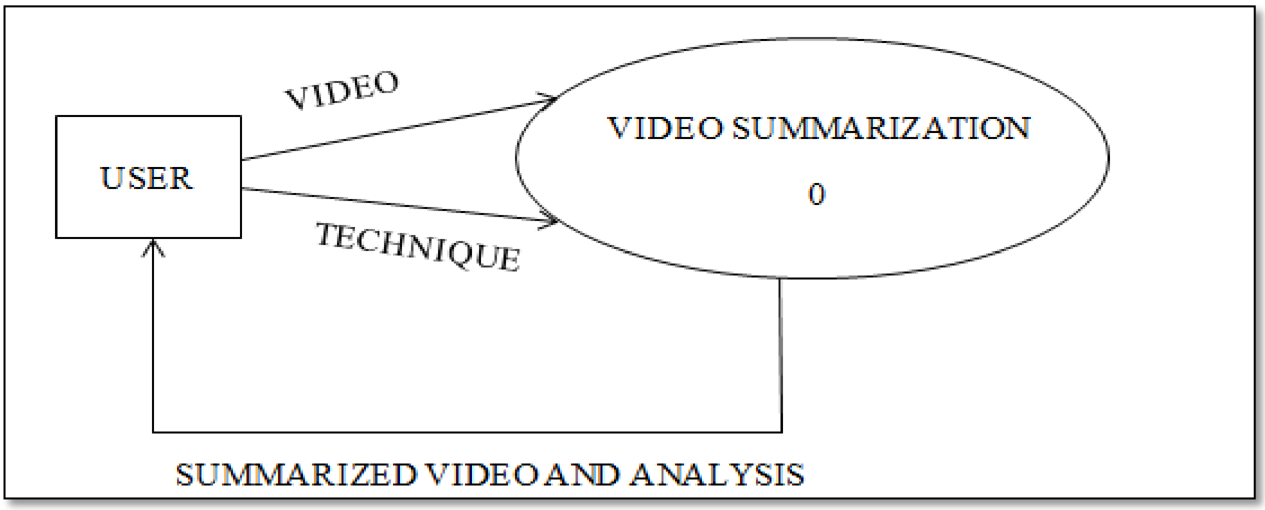


Fig 4.1: DFD Level 0: Context Diagram for Video Summarization

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 his desired technique for summarizing the input 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

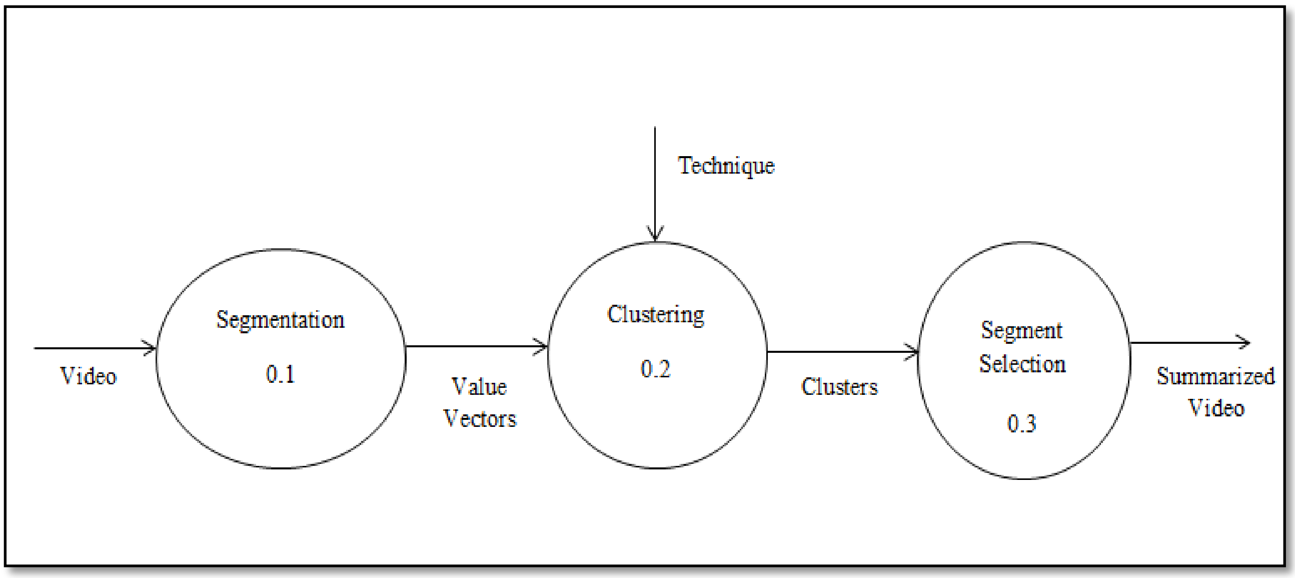


Fig 4.2: DFD Level 1

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

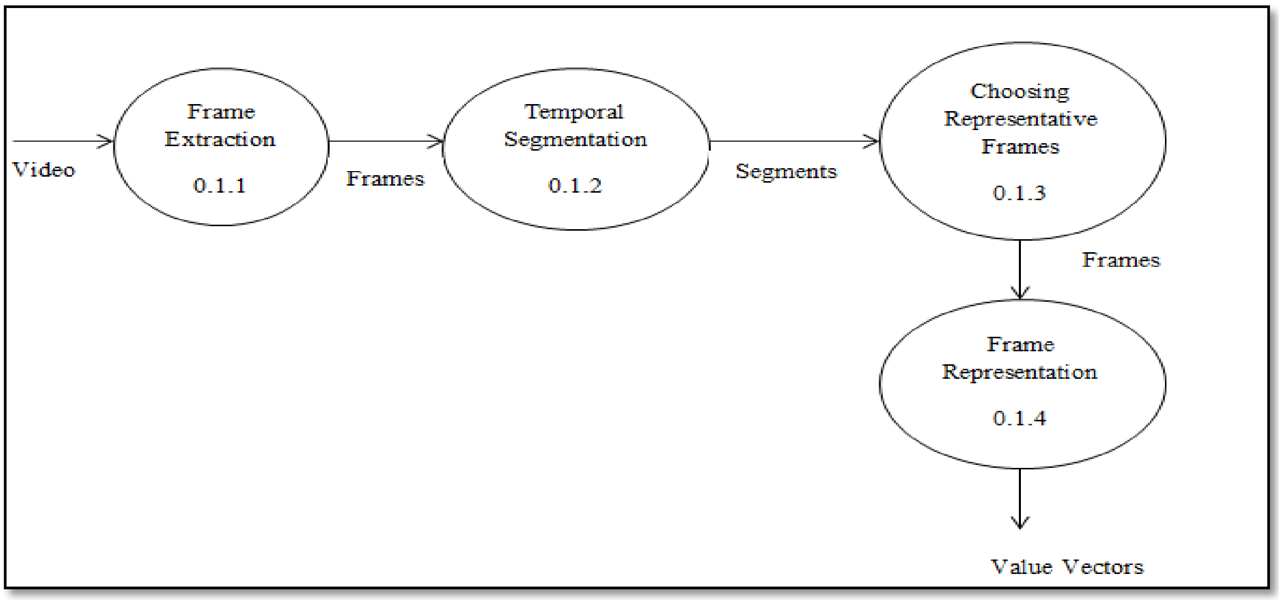


Fig 4.3: DFD Level 2 for segmentation

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 (Can be thought of as small videos)
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

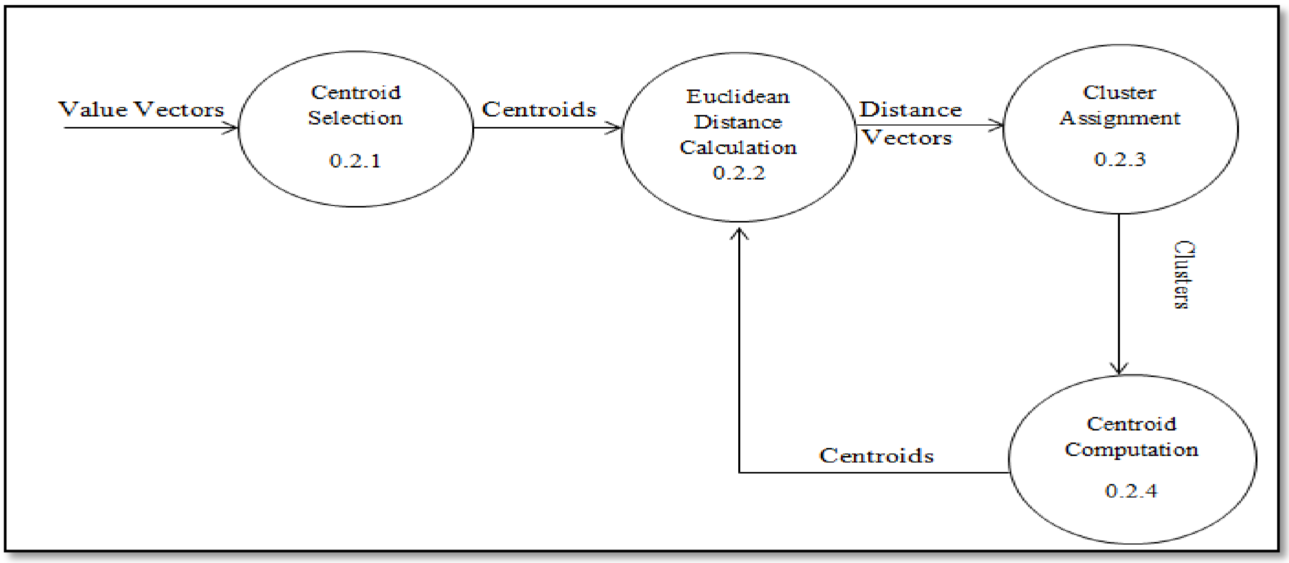


Fig 4.4: DFD Level 2 for K-Means Clustering

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

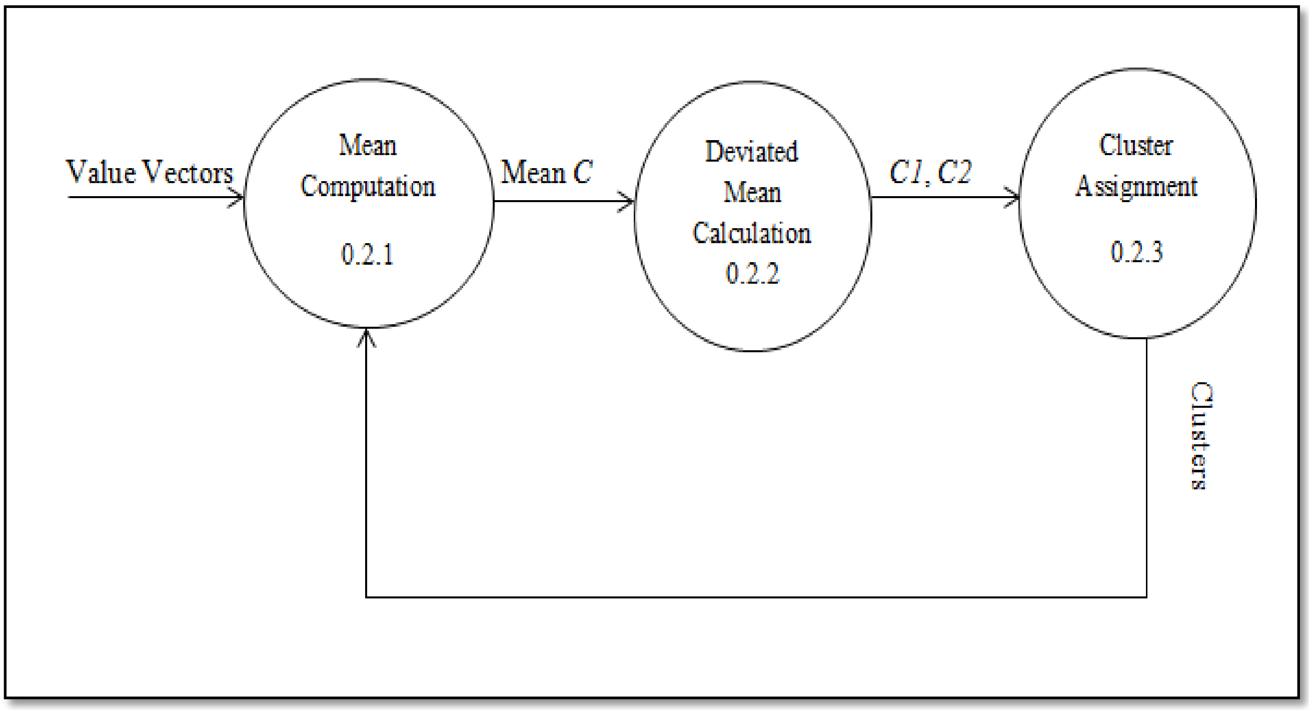


Fig 4.5: DFD Level 2 for KFCG Clustering

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 $C1 = 1.1 \times C$ $C2 = 0.9 \times C$	Double
Clusters	Similar video segments fall under one cluster. Here, the distances are computed with respect to *C1* and *C2*	Cell Array

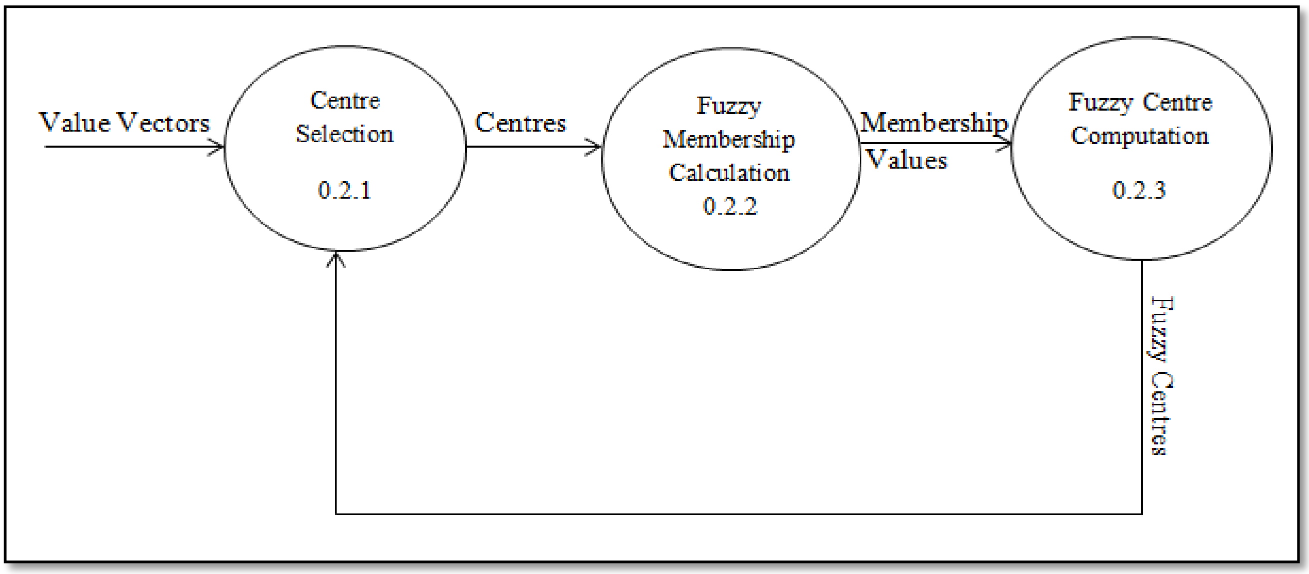


Fig 4.6: DFD Level 2 for FCM Clustering

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 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. The clustered segments are then traversed in a round-robin fashion.

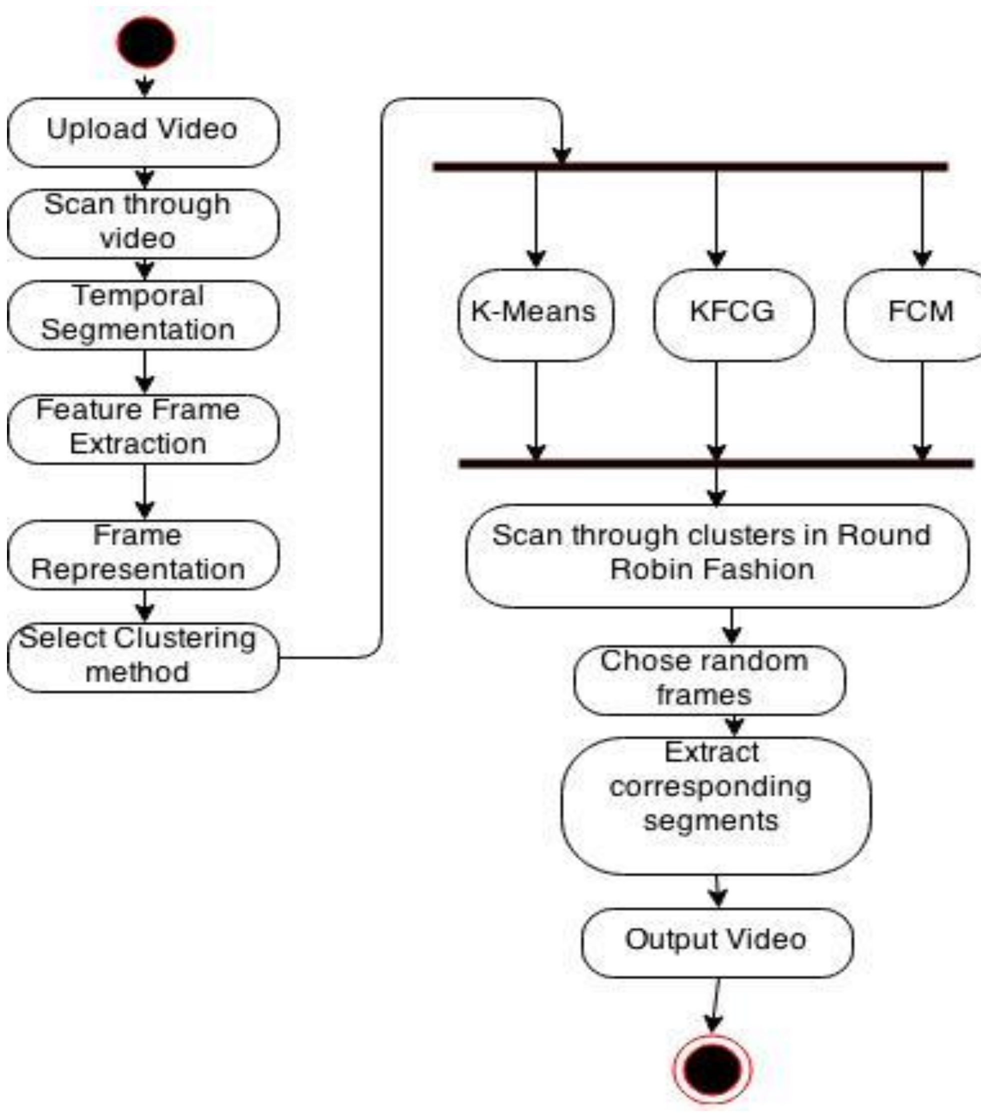
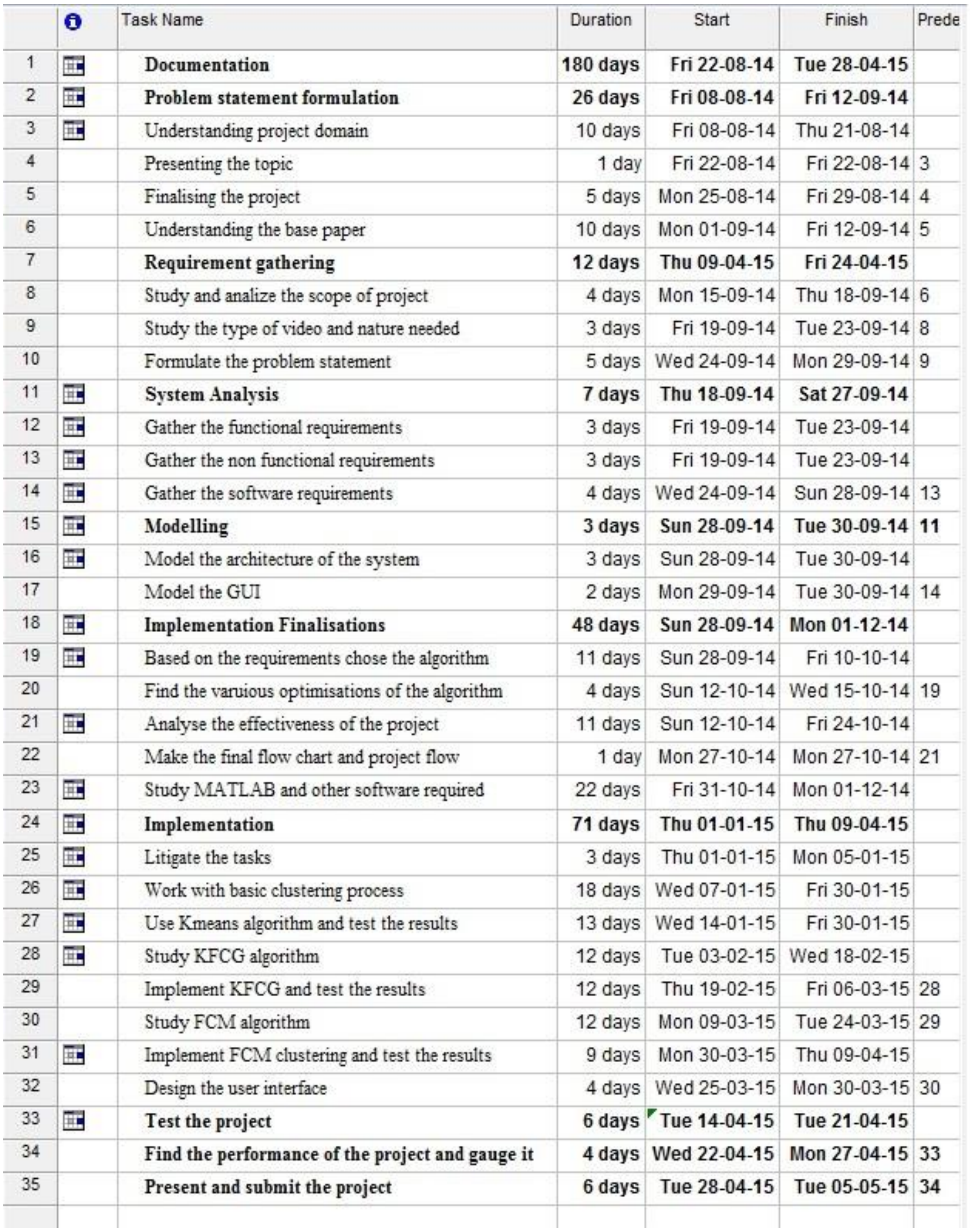


Fig 4.7: Activity Diagram for the Video Summarization System

4.3 Timeline Chart

The timeline chart gives our plan on this project in the year 2014-2015. The first phase of the project is completed. The second phase involves implementation and testing.



	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 time line are:

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

The Gantt chart for the timeline scheduling in Fig.4.8 is given in Fig.4.9.

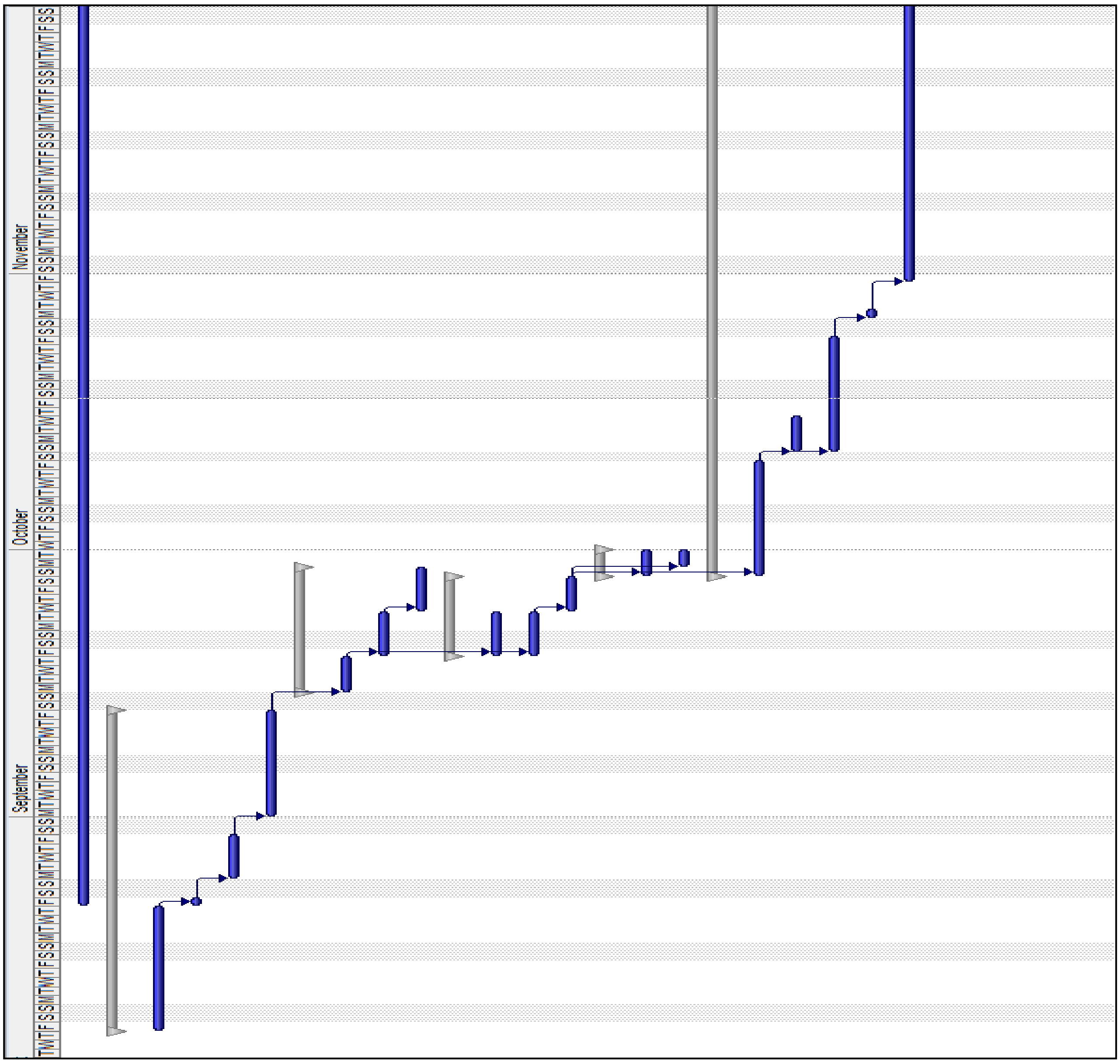


Fig 4.9: Gantt chart for the project

Chapter V

Design

5.1. System Description

The system starts scanning the video. Temporal segmentation is performed while scanning. The length of a segment 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:

$$\text{Value} = \Sigma\, i \text{ x histogram } (i)$$

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 vectors 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 Fuzzy C-Means). 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

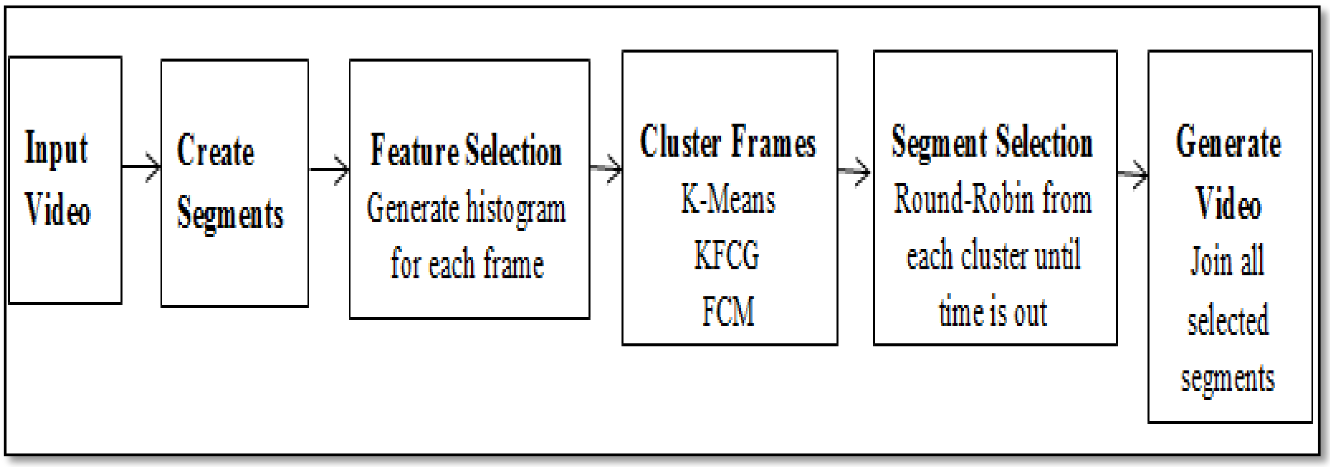

Input Video
Create Segments
Feature Selection
Generate histogram for each frame
Cluster Frames
K-Means
KFCG
FCM
Segment Selection
Round-Robin from each cluster until time is out
Generate Video
Join all selected segments

Fig 5.1 Block Diagram of the System

The various blocks in the Fig 5.1 are explained as follows.

1. Input Video

The input video is taken from the user. Frames are extracted from this video.

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. Let *f* denote the frame rate of the video.

$$k = 2 \text{ x floor } \left(\frac{t}{60}\right)$$

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

$$n = k \text{ x } f$$

3. Feature Selection

The representative frames are chosen. These frames are then represented by the value vectors

4. Cluster Frames

The value vectors are clustered according to the technique chosen by the user.

5. Segment Selection

Each cluster is traversed in a round-robin manner. The highest valued segment is always chosen from these clusters.

There are eight clusters. Thus, for unbiased computations we keep the number of output segments as 8.

6. Generate Video

The output segments are then played as a video.

5.3. User Interface

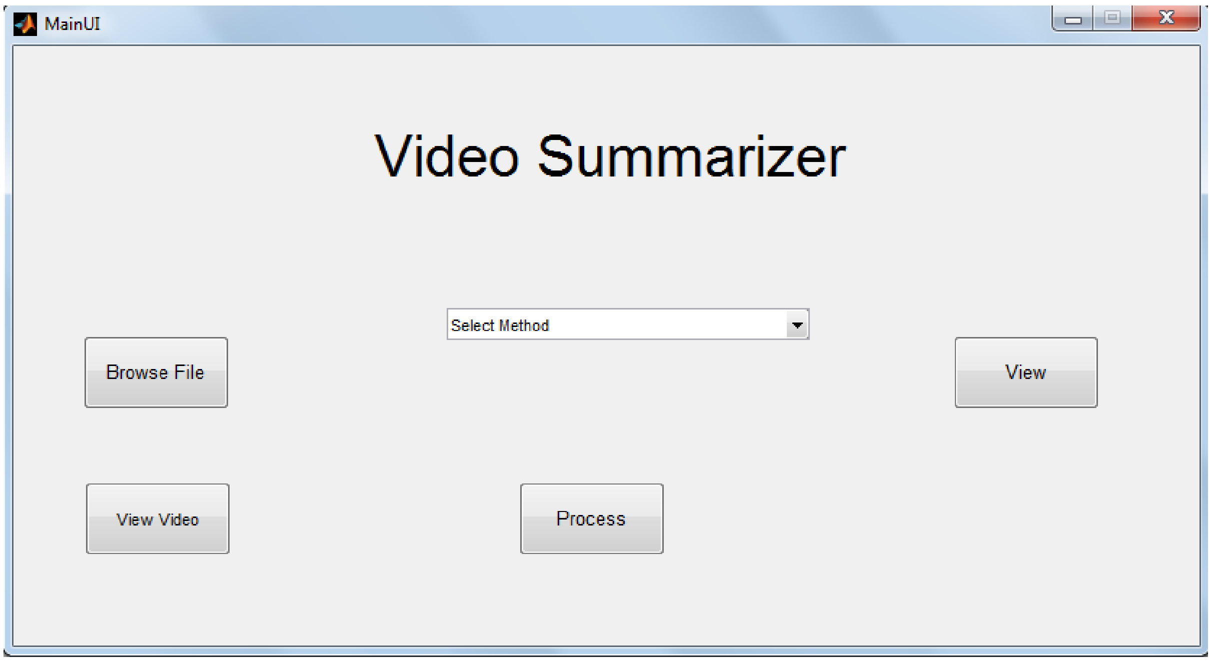


Fig 5.2 Main Interface of the Video Summarizer

Browse File: User browses for the video to be summarized

View Video: The original video is viewed by the user

Select Method: The user selects the clustering technique

Process: User clicks this button for summarizing the original video

View: The output summary can be viewed

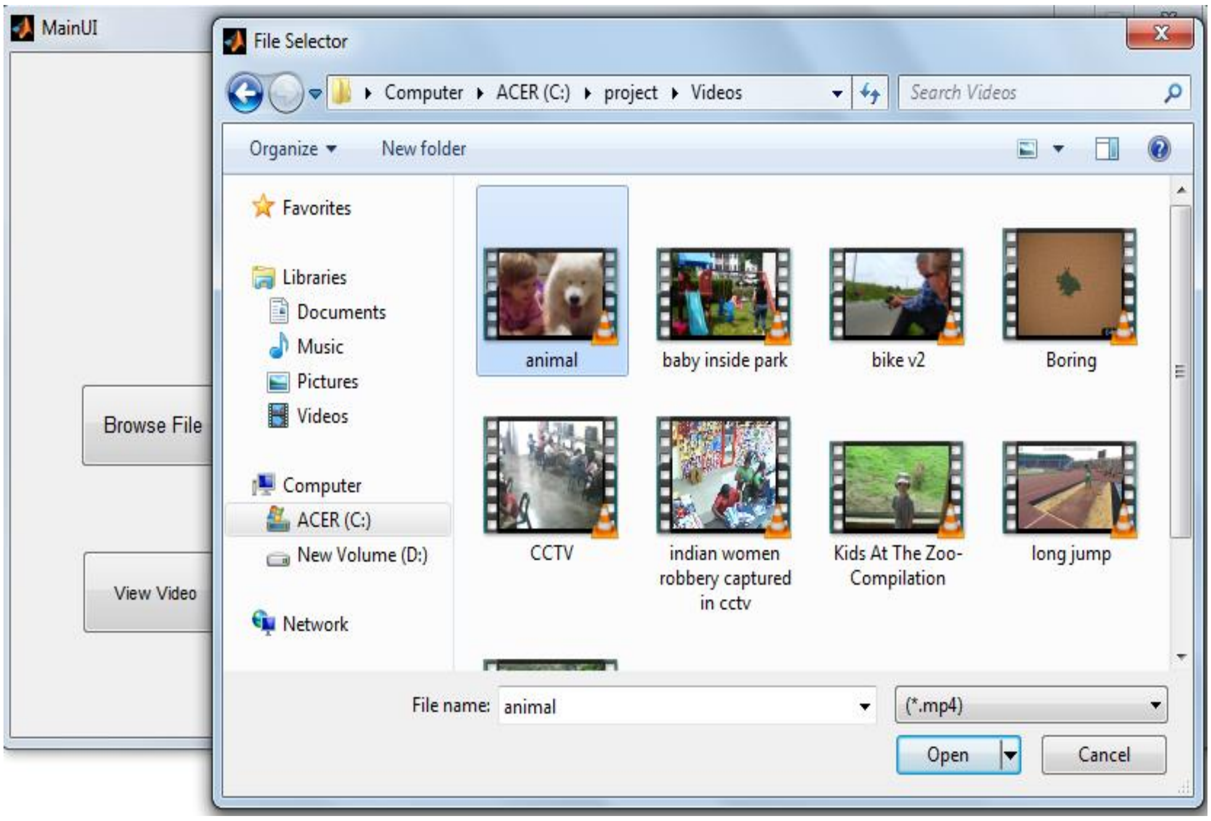


Fig 5.3 Browsing the video

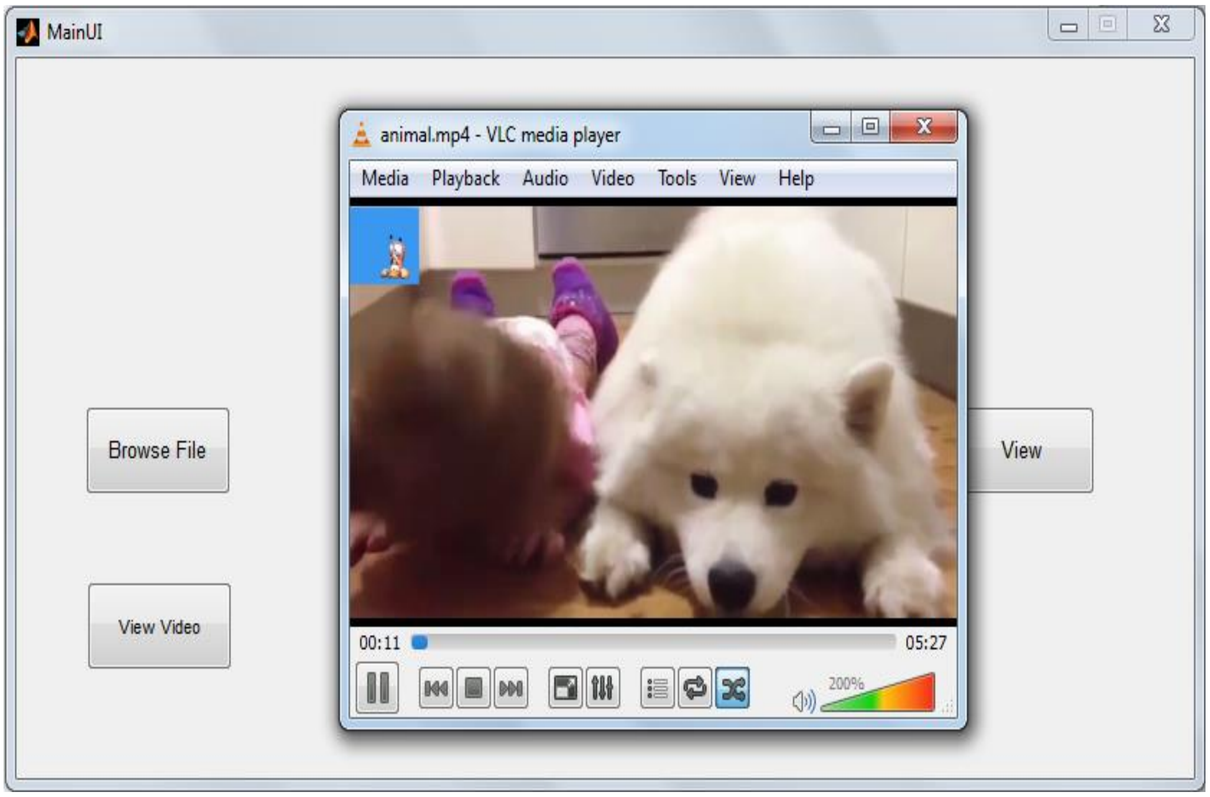


Fig 5.4 Viewing the original video

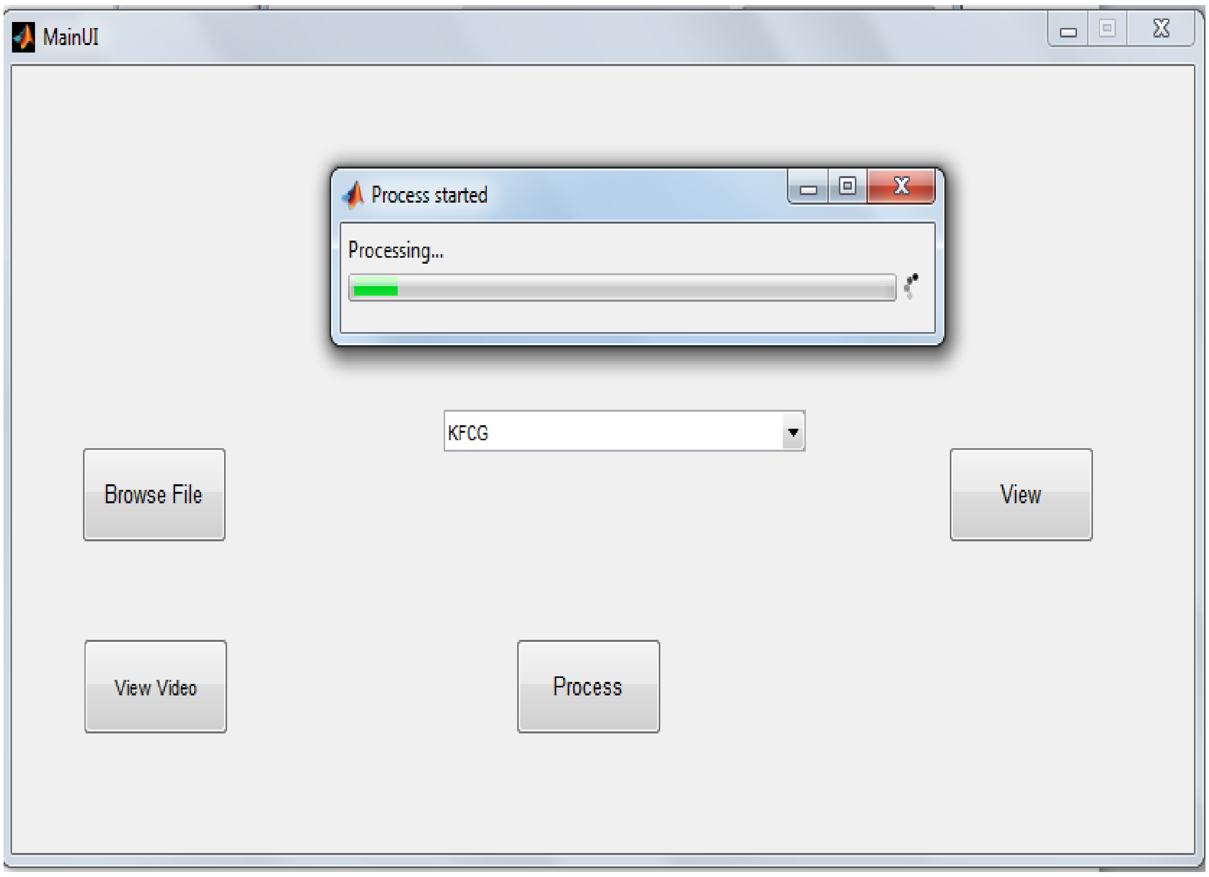


Fig 5.5 Processing the original video

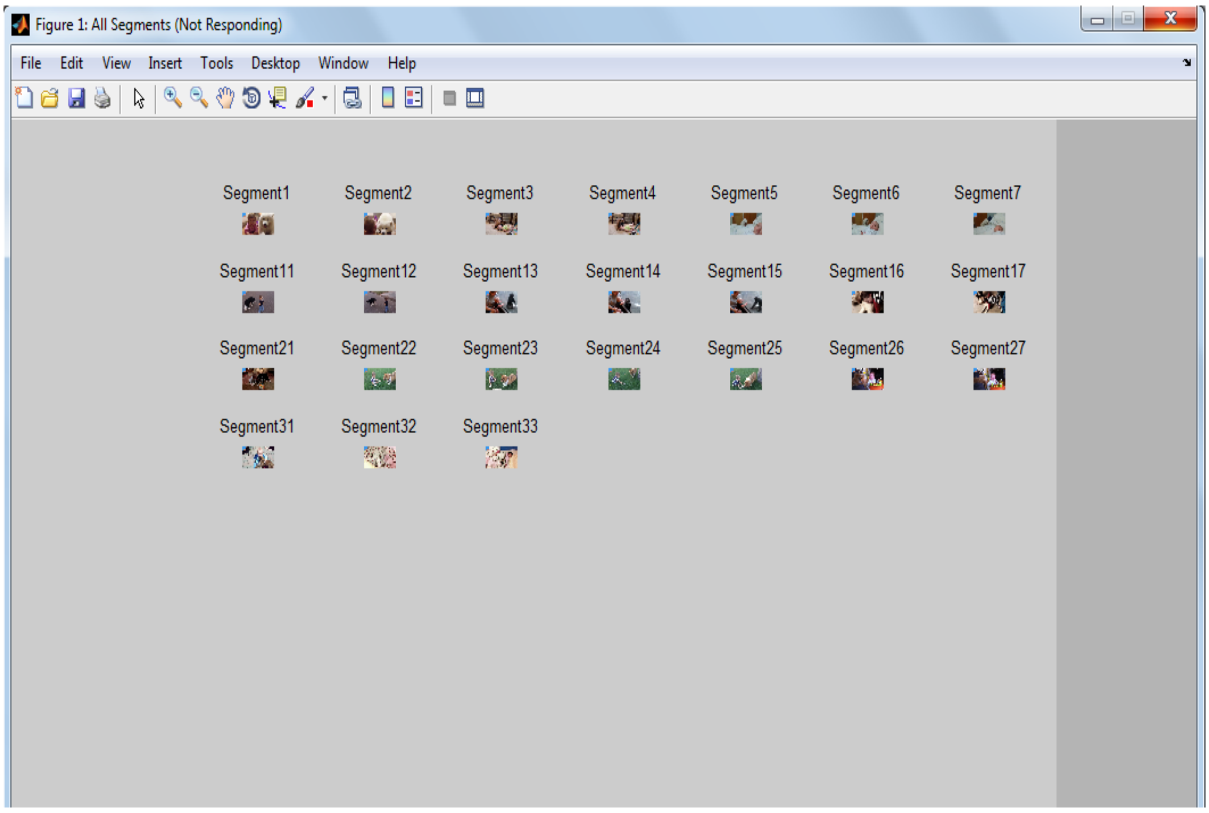


Fig 5.6 Representative frames of all the segments

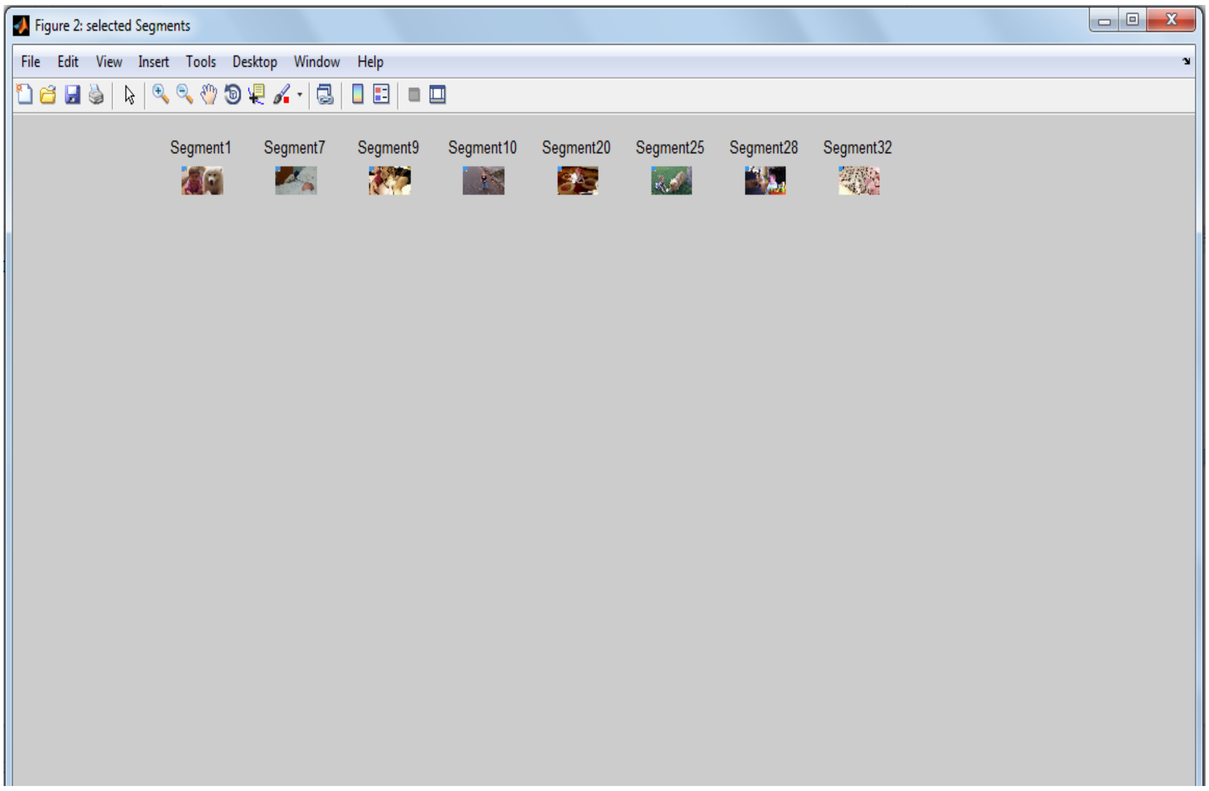


Figure 2: selected Segments
File Edit View Insert Tools Desktop Window Help
Segment1
Segment7
Segment9
Segment10
Segment20
Segment25
Segment28
Segment32

Fig 5.7 Representative frames of the selected segments

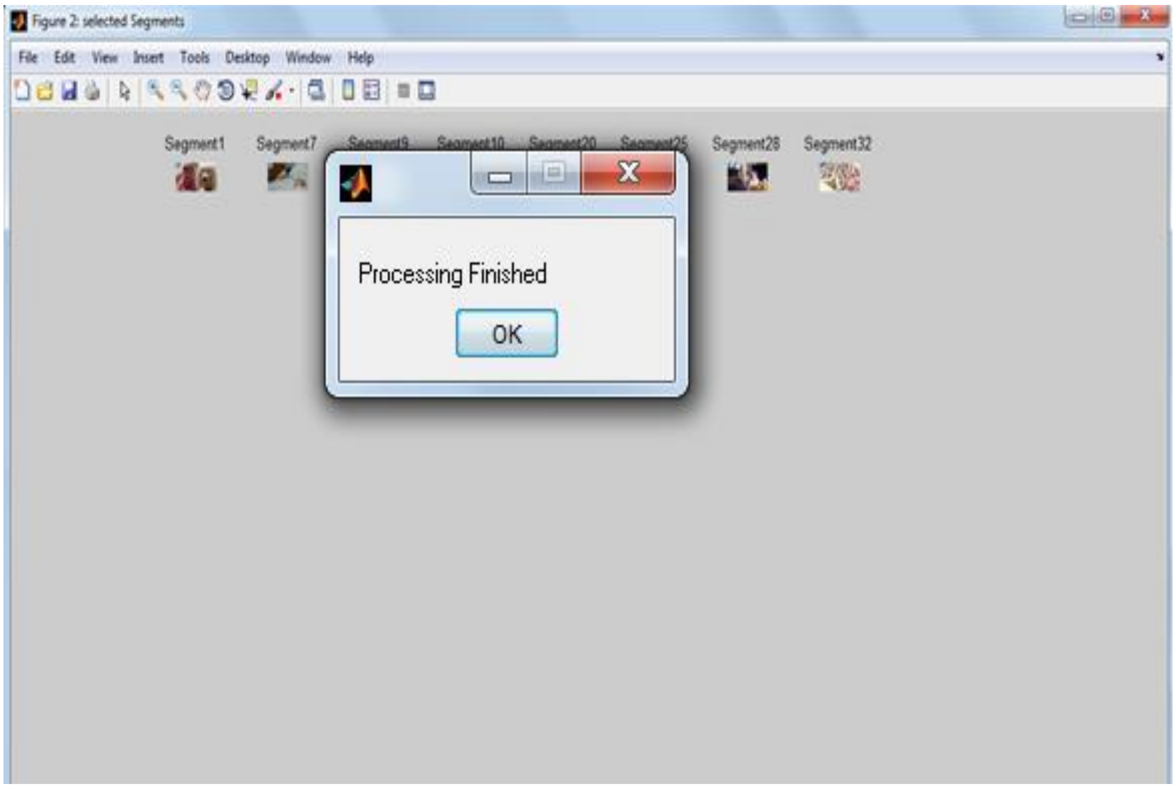
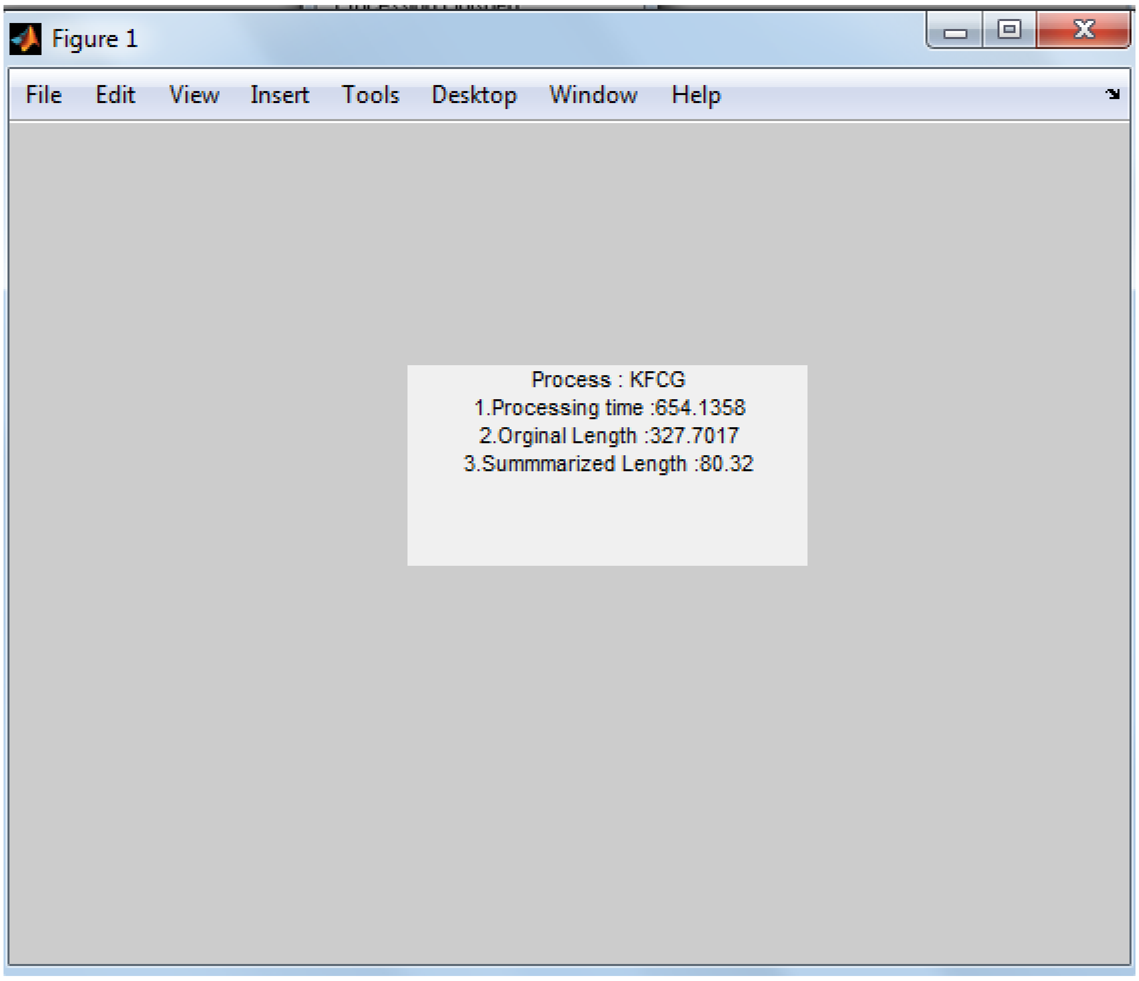


Figure 2: selected Segments
File Edit View Insert Tools Desktop Window Help
Segment1
Segment7
Segment9
Segment10
Segment20
Segment25
Segment28
Segment32
OK

Processing Finished

Fig 5.8 After the Completion of Processing



Process : KFCG
1.Processing time :654.1358
2.Orginal Length :327.7017
3.Summmarized Length :80.32

Fig 5.9 Processing Analysis

Chapter VI

Implementation

6.1 MainUI

```
function varargout = MainUI(varargin)
tic;
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MainUI_OpeningFcn, ...
                   'gui_OutputFcn',  @MainUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function MainUI_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = MainUI_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function Browse_Callback(hObject, eventdata, handles)
global ExPath FileName;
[FileName,FilePath ]= uigetfile({'*.mp4';'*.avi'},'File Selector');
ExPath = fullfile(FilePath, FileName);
len=length(FileName);
FileName=FileName(1:len-4);
```

```
function Process_Callback(hObject, eventdata, handles)
global d;
global method;
if strcmp(method,'Select Method')==1
   msgbox('Select Method');
else
d =com.mathworks.mlwidgets.dialog.ProgressBarDialog.createProgressBar('Started', []);
d.setValue(0);                     % default = 0
d.setProgressStatusLabel('Processing...');  % default = 'Please Wait'
d.setSpinnerVisible(true);              % default = true
d.setCancelButtonVisible(false);
d.setCircularProgressBar(false);
d.setVisible(true);                    % default = false
end
if strcmp(method,'Kmeans')==1
   VideoMainKmeans;
elseif strcmp(method,'KFCG')==1
   VideoMainKFCG;
elseif strcmp(method,'FCM')==1
   VideoMainFCM;
end

function Download_Callback(hObject, eventdata, handles)
global FileName;
winopen(FileName);

function popupmenu1_Callback(hObject, eventdata, handles)
global method;
contents = cellstr(get(hObject,'String'));% returns popupmenu1 contents as cell array
method=contents{get(hObject,'Value')};% returns selected item from popupmenu1

function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function pushbutton4_Callback(hObject, eventdata, handles)
global ExPath;
winopen(ExPath);
```

6.2 VideoMain

```
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;
```

6.3 VideoMainKMeans

```
global ExPath d l k framerate1 numFrames1 numFramesWritten11 imageNames1;

%Extracting & Saving of frames from a Video file through Matlab Code%
d.setValue(0.1);
l=8;

% 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
workingDir = 'C:\VideoSumm';
imageNames1 = dir(fullfile(workingDir,'snaps','*.png'));
imageNames1 = {imageNames1.name};

%sorting the imagesNames according to numeric names of images
for i = 1:length(imageNames1)
out(i) = cellfun(@(x)str2double(regexp(x,'\d*\.\d*','match')),imageNames1(i));
end
out=sort(out);

for i = 1:length(imageNames1)
   imageNames1(i)=cellstr(strcat(num2str(out(i)),'.png'));
end

figure('units','normalized','outerposition',[0 0 1 1],'Name','All Segments');
 for n = 1:length(imageNames1)
 f= fullfile(workingDir,'snaps', imageNames1{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(n)));
 end


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

6.4 VideoSumKMeans

```
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);
    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='K means';
analysisOfProcess;
```

6.5 VideoMainKFCG

```
global ExPath d l k framerate1 numFrames1 numFramesWritten11 mov imageNames1;
```

```
%Extracting & Saving of frames from a Video file through Matlab Code%
d.setValue(0.1);
l=8;

% 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
workingDir = 'C:\VideoSumm';
imageNames1 = dir(fullfile(workingDir,'snaps','*.png'));
imageNames1 = {imageNames1.name};

%sorting the imagesNames according to numeric names of images
```

```
for i = 1:length(imageNames1)
out(i) = cellfun(@(x)str2double(regexp(x,'\d*\.\d*','match')),imageNames1(i));
end
out=sort(out);

for i = 1:length(imageNames1)
   imageNames1(i)=cellstr(strcat(num2str(out(i)),'.png'));
end

figure('units','normalized','outerposition',[0 0 1 1],'Name','All Segments');
 for n = 1:length(imageNames1)
 f= fullfile(workingDir,'snaps', imageNames1{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(n)));
 end

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

6.6 VideoSumKFCG

```
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);
clust=kfcg(dataset,3);
final=zeros(l,1);

[height, width] = size(clust);

k2=width;

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,8)+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);
    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='KFCG';
analysisOfProcess;
```

6.7 KFCG Function

```
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
```

6.8 VideoMainFCM

```
global ExPath d l k framerate1 numFrames1 numFramesWritten11 mov imageNames1;

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

```
l=8;

% 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
workingDir = 'C:\VideoSumm';
imageNames1 = dir(fullfile(workingDir,'snaps','*.png'));
imageNames1 = {imageNames1.name};

%sorting the imagesNames according to numeric names of images
for i = 1:length(imageNames1)
out(i) = cellfun(@(x)str2double(regexp(x,'\d*\.\d*','match')),imageNames1(i));
end
```

```
out=sort(out);

for i = 1:length(imageNames1)
    imageNames1(i)=cellstr(strcat(num2str(out(i)),'.png'));
end

figure('units','normalized','outerposition',[0 0 1 1],'Name','All Segments');
 for n = 1:length(imageNames1)
 f= fullfile(workingDir,'snaps', imageNames1{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(n)));
 end

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

6.9 VideoSumFCM

```
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  not existing
    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;
```

6.10 FrameDef Class

```
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
```

6.11 checkValue Function

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

6.12 AnalysisOfProcesses

```
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

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, we have taken five different categories of videos. These categories are:

1. CCTV

It is also known as video surveillance. 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.

2. Short Documentaries

Documentaries are not as redundant as CCTV, 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.

3. 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.

4. 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.

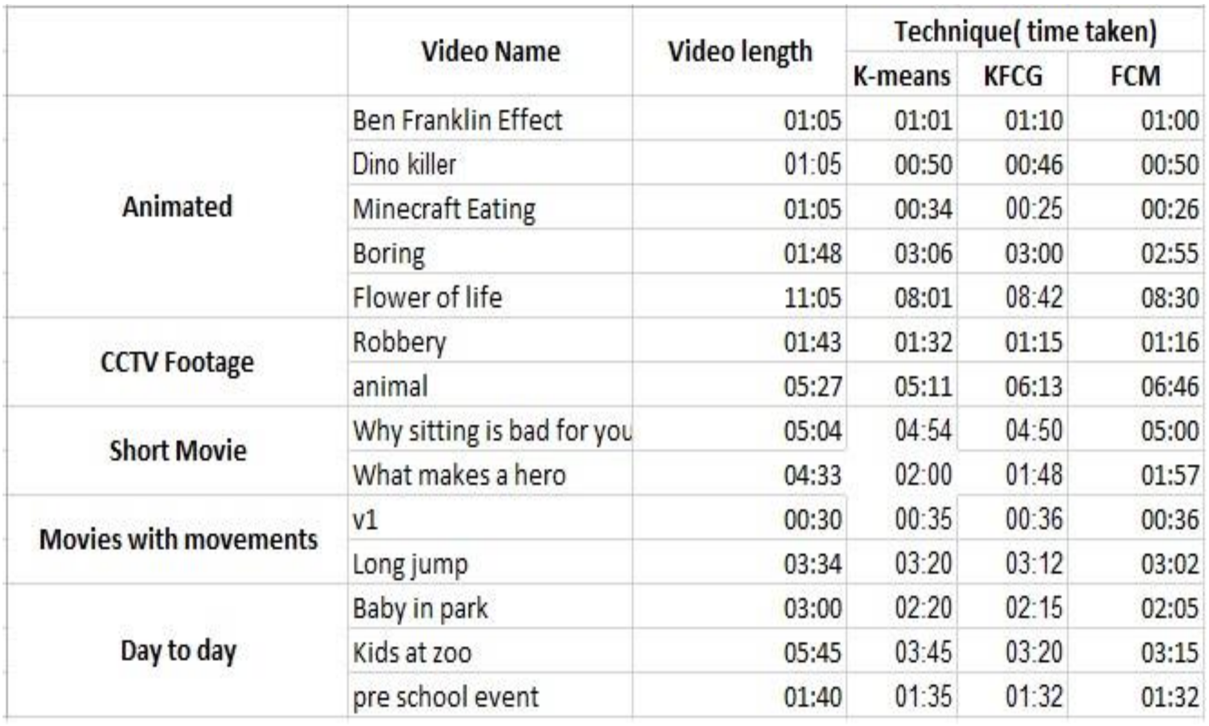
5. Sports

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.

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.

7.1. Analysis Based on Processing Time

Table 7.1 Observations



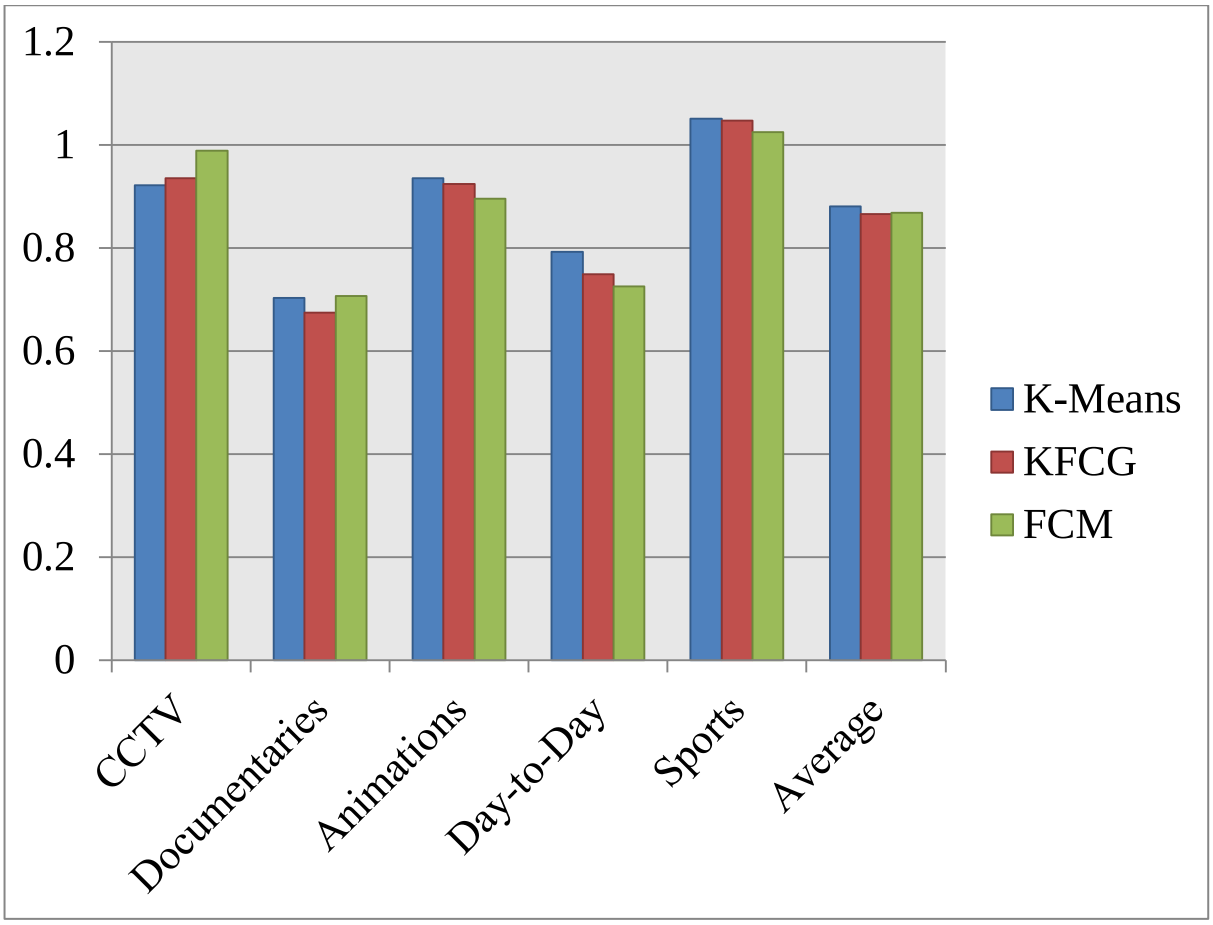
	Video Name	Video length	Technique(time taken)		
			K-means	KFCG	FCM
Animated	Ben Franklin Effect	01:05	01:01	01:10	01:00
	Dino killer	01:05	00:50	00:46	00:50
	Minecraft Eating	01:05	00:34	00:25	00:26
	Boring	01:48	03:06	03:00	02:55
	Flower of life	11:05	08:01	08:42	08:30
CCTV Footage	Robbery	01:43	01:32	01:15	01:16
	animal	05:27	05:11	06:13	06:46
Short Movie	Why sitting is bad for you	05:04	04:54	04:50	05:00
	What makes a hero	04:33	02:00	01:48	01:57
Movies with movements	v1	00:30	00:35	00:36	00:36
	Long jump	03:34	03:20	03:12	03:02
Day to day	Baby in park	03:00	02:20	02:15	02:05
	Kids at zoo	05:45	03:45	03:20	03:15
	pre school event	01:40	01:35	01:32	01:32

30% Compression Raton was achieved for all these videos. Table 7.1 gives the processing time taken for 10 videos in minutes.

Table 7.2 Ratio of Processing Time and Video Length

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

Table 7.2 gives the average ratios of processing time and video lengths for all five categories. Fig 7.1 was computed based on Table 7.2



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 KFCG. 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.

7.2. Analysis based on Accuracy

Accuracy, A, is given as follows

$$A = \frac{Number\ of\ frames\ detected\ correctly\ in\ the\ output}{Total\ Number\ of\ Frames}$$

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 VIII

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 (Kekre's Fast Codebook generation) and Fuzzy C means. 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.

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

APPENDIX 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.

APPENDIX 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*

$$C1 = C\,(1 + \alpha)$$
$$C2 = C\,(1 - \alpha)$$

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

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

iii. Compute the fuzzy centers v_j using

$$v_j = (\sum_{i=1}^{n} (\mu_{ij})^m x_i) / (\sum_{i=1}^{n} (\mu_{ij})^m), \forall j = 1, 2, ...c$$

iv. Repeat steps 2-3 until the termination criterion is satisfied: $J < \beta$

$$J(U,V) = \sum_{i=1}^{n} \sum_{j=1}^{c} (\mu_{ij})^m \left\| x_i - v_j \right\|^2$$