"Video Shot Detection and Summarization"

A BACHELOR'S MINI PROJECT

submitted in partial fulfillment

*of the requirements for the completion of the **6th** semester*

of the

UNDER GRADUATE PROGRAM

in

INFORMATION TECHNOLOGY
(B.Tech in IT)

Submitted by

Prateek Porwal(IIT2010035)
Shahbaz Khan(IIT2010039)
Rakesh Kumar(IIT2010076)

Under the Guidance of:

Prof.Uma Shankar Tiwary

IIIT-Allahabad

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
ALLAHABAD-211012

May, 2013

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this project entitled "**Video Shot Detection and Summarization**", submitted in the partial fulfillment of the completion of the semester 6th of Bachelor of Technology (B.Tech) program, in Information Technology at Indian Institute of Information Technology, Allahabad, is an authentic record of my original work carried out under the guidance of **Prof.U.S Tiwary** due acknowledgements have been made in the text of the project to all other material used. This semester work was done in full compliance with the requirements and constraints of the prescribed curriculum.

Place: Allahabad
Date:

Name of the students
Prateek Porwal
Shahbaz Khan
Rakesh Kumar

CERTIFICATE FROM SUPERVISOR

I/We do hereby recommend that the mini project report prepared under my/our supervision by Prateek Porwal, Shahbaz Khan and Rakesh Kumar titled "**Video Shot Detection and Summarization**" be **accepted** in the partial fulfillment of the requirements of the completion of 6th semester of Bachelor of Technology in Information Technology **for Examination**

Date:
Place: Allahabad

Guide's name
Prof.U.S.Tiwary, IIITA

Committee for Evaluation of the Thesis

_____ _____

_____ _____

ACKNOWLEDGEMENTS

The author would like to express his sincere gratitude to **Prof.U.S.Tiwary**

Place: Allahabad
Date:

Name of the students

Prateek Porwal(IIT2010035)
Shahbaz Khan(IIT2010039)
Rakesh Kumar(IIT2010076)

B Tech 3rd Year, IIITA

ABSTRACT

Video shot detection is the processor-intensive task of splitting a video into continuous Shots with hard or soft cuts as the boundaries. After shots are detected those shots are summarized based on key-frames extraction techniques. Those detected shots have been widely used in video summarization for video analysis, indexing and browsing. Video summarization has been categorized based on two approaches namely: static video summarization and motion-based video summarization. Many algorithms have been proposed for detecting video shot boundaries and classifying shots and shot transition types. This report presents a brief description of some of the important algorithms for video shot detection like histogram difference, statistical difference, pixel difference, edge change ratio, adaptive threshold, compression differences, principal co-ordinate system and some summarization techniques like sampling-based Key-frame extraction and shot-based key-frame extraction, color-based approach, motion-based approach and segment-based key-frame extraction.

We have **implemented** the **Histogram Difference** technique for video shot detection. And for summarizing the video, we have used two techniques, moving-images and still-images with fixed threshold.

Table of Contents

1. INTRODUCTION

1.1 Currently existing technologies

Shot detection

Shot Detection in a video sequence is a process of identifying visual discontinuities along the time domain. During this process, it is required to extract visual features that measure the degree of similarity between frames in a given shot. Shot is defined as a sequence of frames captured from a single camera operation. The shot detection approaches are classified based on the features and methods used to solve the problem and other approaches are described.

Histogram differences

Histograms are the most common method used for comparing images. And this technique can be extended to find the image whose histogram varies significantly from the previous image histogram. Thus we can detect shot boundaries. The histogram method computes 64 bin gray level histograms of the two images and Euclidean distance measure is used to find the histogram difference. If this distance between the two histograms is above a threshold, a shot boundary is assumed.

1. **Region based Histogram differences**

In this method each image is divided into 16 blocks in a 4x4 pattern. For every image, a 64-bin gray-scale histogram is computed for each region. A Euclidean distance measure is used to find the difference between the region histograms of two consecutive images. If the distance is above a threshold, the region count for that

image is incremented. If the region count is more than some predefined threshold, a shot boundary is assumed.

1.2 Analysis of previous research in this area

1. **Statistical differences**

In this method, each image is divided into 16 blocks. Then, for each block the mean and the standard deviation is found. A Euclidean distance measure is used to find the mean and standard deviation differences between the corresponding blocks of two consecutive images. Similar to the previous method, two thresholds are defined. If the region count is more than the second threshold, a shot boundary is assumed.

2. **Pixel differences**

This is the simplest method for determining shot boundaries. The difference between corresponding pixels of two consecutive images is computed. If the difference is greater than some threshold, than a shot boundary is assumed. All the above approaches can be improved to give good results by using other color spaces like RGB, HSV and YIQ instead of grayscale. And besides Euclidean distance, Chi-square distance measure can be used.

3. **Adaptive threshold**

The above approaches use static thresholds, instead Yusoff use adaptive threshold setting, by using statistics of the dissimilarity measure within a sliding window. Three different models, constant covariance model, proportional variance model and Dugad models are used for setting the threshold. Adaptive thresholding gives a better rate of detection shots than the static threshold method.

4. Principal Co-ordinate System

This is a new approach given by Alper Yilmaz. An eigen space decomposition of the RGB color space is used to describe the frames in a more descriptive coordinate system. The method is based on calculation of principal coordinate system from spatial information determined from RGB color space by applying eigen space decomposition. In this method covariance matrix of the color space is computed, then the eigenvectors are determined for that matrix and the maximum eigen valued eigenvector is used to determine the shot boundaries. This approach is well suited for real time surveillance

5. Compression Differences

Little et al used differences in the size of JPEG compressed frames to detect shot boundaries as a supplement to a manual indexing system. Arman, Hsu, and Chiu found shot boundaries by comparing a small number of connected regions. They used differences in the discrete cosine transform (DCT) coefficients of JPEG compressed frames as their measure of frame similarity, thus avoiding the need to decompress the frames. A further speedup was obtained by sampling the frames temporally and using a form of binary search to find the actual boundary. Potential boundaries were checked using a color histogram difference method.

6. Edge Change Ratio

The Edge Change Ratio Algorithm takes two consecutive frames F and F’ as input. The algorithm starts with converting the frames into grayscale frames first. Then edges in the frames are identified, thus converting the frames into Edge-Frames E and E’. Now, frames are dilated and then inverted. After all the above procedure, we will finally get the frames E and E’. In these frames, edge pixels are counted for both of them along with entering and exiting pixels. Entering edge pixels is the edge pixel that is present in the current frame E and is farther away than r in next frame. Similarly, the exiting edge pixel is the edge pixel that is present in the next frame E’ and is farther away than r in

the current frame E. So the similarity between frames E and E' is defined as maximum of fraction entering_pixels/edge_pixels and exiting_pixels/edge_pixels.

Video summary

These are the different approaches of video summary. One of the approaches in a video summary is to select keyframes (a set of salient images) from the original video sequence. Based on the way the key frames are constructed, there are 3 classes of keyframe extraction:

1. Sampling-based Key-frame Extraction

Most of earlier work in video summarization chooses to select key-frames by randomly or uniformly sampling the video frames from the original sequence at certain time intervals, which was applied in the Video Magnifier, MiniVideo system. Although this is probably the simplest way to extract key-frames, the drawback is that such an arrangement may cause some short yet important segments to have no representative frames while longer segments could have multiple frames with similar content, thus failing to capture and represent the actual video content.

2. Shot-based key-frame Extraction

More sophisticated work tends to extract key-frames by adapting to the dynamic video content. Since a shot is defined as a video segment within a continuous capture period, a natural and straightforward way of key-frame extraction is to use the first frame of each shot as its key-frame. However, while being sufficient for stationary shots, one Key-frame per shot does not provide an acceptable representation of dynamic visual content; therefore multiple key-frames need to be extracted by adapting to the

underlying semantic content. However, since computer vision still remains to be a very difficult research challenge, most of existing work chooses to interpret the content by employing some low-level visual features such as color and motion, instead of performing a tough semantic understanding. Based on the features that the video possess, the approaches are classified into 4 different classes: color-based approach, motion-based approach, mosaic-based approach.

3. Color-based Approach

The method proposed by Zhang , the key-frames are extracted in a sequential fashion for each shot. Particularly, the first frame within the shot is always chosen as the first keyframe, and then the color-histogram difference between the subsequent frames and the latest keyframe is computed. Once the difference exceeds a certain threshold, a new keyframe will be declared. A similar work is also reported by Yeung and Liu. One possible problem with above extraction methods is that there is a probability that the first frame is a part of transition effect at the shot boundary, thus strongly reducing its representative quality. In Huang from University of Illinois at Urbana-Champaig propose to extract the keyframes using an unsupervised clustering scheme. Basically, all video frames within a shot are first clustered into certain number of clusters based on the color-histogram similarity comparison where a threshold is predefined to control the density of each cluster. Next, all the clusters that are big enough are considered as the key clusters and a representative frame closest to the cluster centoid is extracted from each of them. Ferman and Tekalo reported a similar work in this. Because the color histogram is invariant to image orientations and robust to background noises, color-based keyframe extraction algorithms have been widely used. However, most of these works are heavily threshold-dependent, and cannot well capture the underlying dynamics when there is lots of camera or object motion.

4. Motion-based Approach

Motion-based approaches are relatively better suited for controlling the number of frames based on temporal dynamics in the scene. In general, pixel-based image differences or optical flow computation are commonly used in this approach. In Wolfs work, the optical flow for each frame is first computed, and then a simple motion metric is computed. Finally by analyzing the metric as a function of time, the frames at the local minima of motion are selected as the keyframes. A domain specific keyframe selection method is proposed where a summary is generated for video-taped presentations. Toklu and Liou from Siemens Corporate Research reported their work where 3 different operation levels are suggested based on the available machine resources: at the lowest level, pixel-based frame differences are computed to generate the temporal activity curve since it requires minimal resources; at level 2, color histogram-based frame differences are computed to extract color activity segments, and at level 3 sophisticated camera motion analysis is carried out to estimate the camera parameters and detect the motion activity segments. Keyframes are then selected from each segment and necessary elimination is applied to obtain the final result.

5. Mosaic-based Approach

A limitation of above approaches is that it is not always possible to select the keyframes that can represent the entire video content well. For example, given a camera panning/tilting sequence, even if multiple keyframes are selected, the underlying dynamics still couldn't be well captured. In this case, the mosaic-based approach can be employed to generate a synthesized panoramic image that can represent the entire content in an intuitive manner. Mosaics, also known as salient stills, video sprites or video layers, are usually generated in the following 2 steps:

1) Fitting a global motion model to the motion between each pair of successive frames;

2) Compositing the images into a single panoramic image by warping the images with the estimated camera parameters.

6. Segment-based Key-frame Extraction

One major drawback of using one or more key-frames for each shot is that it does not scale up for long videos since scrolling through hundreds of images is still time-consuming, tedious and ineffective. Therefore, recently more and more people begin to work on higher-level video unit, called as video segment. It could be a scene, an event, or even the entire sequence. In this context, the segment-based key-frame set will surely become more concise than the shot- based keyframe set. Uchihashi first cluster all video frames into a predefined number of clusters, and then the entire video is segmented by determining to which cluster the frames of a contiguous segment belong. Next an importance measure is computed for each segment based on its length and rarity, and all segments with their importance lower than a certain threshold will be discarded. The frame that is closest to the center of each qualified segment is then extracted as the representative keyframe, with the image size proportional to its importance index. Finally, a frame-packing algorithm is proposed to efficiently pack the extracted frames into a pictorial summary.

1.3 Problem definition and scope

Due to increased usage of videos online and offline in various fields of work and industry, need for Video Processing has become a thing of utmost importance. Among various Video Processing Techniques, Shot Detection has a very important role to play. With so many videos to watch, and that too lengthy ones, it became tedious process to look complete videos to know about their content. So Shot Detection is one such approach that eases this task by dividing a video into its shots.

Now these shots can be used to build up index for the Video so that user can jump to particular point where a new scene is about to start or they can be used to build up the summary of the video that summarizes the video content in short.

Shot Detection: Given a video V consisting of n shots, we have to find the beginning and end of each shot Also known as shot boundary detection or transition detection. It is fundamental to any kind of video analysis and video application since segmentation of a video into its basic components: the shots.

Those detected video shots are summarized further for video processing indexing and browsing.

1.4 Formulation of the present problem

Due to increased usage of videos online and offline in various fields of work and industry, need for Video Processing has become a thing of utmost importance. Among various Video Processing Techniques, Shot Detection has a very important role to play. With so many videos to watch, and that too lengthy ones, it became tedious process to look complete videos to know about their content. So Shot Detection is one such approach that eases this task by dividing a video into its shots.

Now these shots can be used to build up index for the Video so that user can jump to particular point where a new scene is about to start or they can be used to build up the summary of the video that summarizes the video content in short.

Shot Detection: Given a video V consisting of n shots, we have to find the beginning and end of each shot Also known as shot boundary detection or transition detection. It is fundamental to any kind of video analysis and video application since segmentation of a video into its basic components: the shots.

Those detected video shots are summarized further for video processing indexing and browsing.

1.5 Organization of the thesis

Our thesis of project development comprises of two parts:-

Video Shot detection: - We studied several shot detection algorithms for the development of our project. Out of them we implemented the histogram difference because this technique is far better than other techniques in terms of detection of shots accurately and relative errors.

Video Summarization: - Shot detection technique is further utilized for video summarization for indexing, browsing and video processing. After shots are detected correctly, key-frames of shots are combined together in a proper sequence in order to create the summary of the entire original video using key-frame based extraction technique.

2. DESCRIPTION OF THE HARDWARE AND SOFTWARE USED

2.1 Hardware

- Hardware specification:
- 64 bit CPU processor
- GB RAM

2.2 Software

Software specification:

- Windows operating system 64 bit
- Open CV 2.3.0
- CodeBlocks 10.0.5

3. THEORITOCAL TOOLS – ANALYSIS AND DEVELOPMENT

Some of the applications of shot detection are:

i. Managing video databases for indexing, browsing, searching and summarization.
ii. Digital Video Library as a networked Internet application allowing for Storage, cataloging, retrieval and uni-casting video sequences.
iii. Scene, event or object detection.

Video Summarization: Video summary is an abstract of a longer video document in a shorter period of time. It is a method of presenting the content of a video in a condensed manner without changing the original video.

Terminologies:

Hard cuts: A cut is an instantaneous transition from one scene to the next. There are no transitional frames between 2 shots.

Fades: A fade is a gradual transition between a scene and a constant image (fade-out) or between a constant image and a scene (fade-in).

Dissolves: A dissolve is a gradual transition from one scene to another, in which the first scene fades out and the second scene fades in.

Wipe: another common scene break is a wipe, in which a line moves across the screen, with the new scene appearing behind the line.

Schema of Cut Detection:

Calculate a time series of discontinuity feature values f(n) for each frame. Suppose we use function d(x,y) to measure the dissimilarity between frame x and y. The discontinuity feature value for frame n is f(n)=d(n- 1 ,n).

Pick the cuts position from f(n) based on some threshold techniques.

Histogram Difference Algorithm:

1. Take two consecutive frames into consideration.
2. Convert the BGR frames into HSV format.
3. Find out histograms of the two frames.
4. Find out the histogram differences for the two frames with a proposed method. In our case we have used Bhattacharya Distance as a metric for difference calculation. Bhattacharya Distance between two histograms is given as

   Fig.1

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_1 \bar{H}_2 N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}}$$

   This distance varies from [0, 1]. 0 when frames are identical and 1 when frames are least similar.
5. If the distance between two frames is greater than some threshold, which can be global (fixed) or adaptive, declare it as a cut and the beginning of the new shot.

Thresholding

Thresholding in Histogram Difference Algorithm is used to identify if there is a cut or not by comparing the difference between histograms of two consecutive frames with some value. This value is called **threshold.** Threshold for a video can be defined in two ways as below:

1. Global threshold

   A hard cut is declared each time the discontinuity value f(n) surpasses a global thresholds.

2. Adaptive threshold

   A hard cut is detected based on the difference of the current feature values f(n) from its local neighborhood. Generally this kind of method has 2 criteria for a hard cut declaration:

   - F(n) takes the maximum value inside the neighborhood

   -The difference between f(n) and its neighbors' feature values is bigger than a given threshold.

In our case, we have taken global threshold of 0.3 into consideration. Means whenever histogram difference value exceeds 0.3 bar, it will declare it as a cut and new shot will start.

Summarization

For summarization, we have taken Shot Based KeyFrame Extraction into consideration. Since a shot is defined as a video segment within a continuous capture period, a natural and straightforward way of key-frame extraction is to use the first frame of each shot as its key-frame. However, while being sufficient for stationary shots, one Key-frame per shot does not provide an acceptable representation of dynamic visual content; therefore multiple key-frames need to be extracted.

After keyframes extraction again there are two ways in which summarization can be done.

1. **Moving Images**

   In this case, starting few frames of detected shots are taken into summary. But there is a drawback to this. In case of videos with large number of shots which are of similar type, summary video can be as large as original video which gives no advantage of summarization. So to avoid this situation, we have used another threshold for selecting shots from which summary would be taken. So now if the new shot passes the criteria to be selected for the summary, then only frames from that shot would be taken into account. Now from these selected shots, few frames from the beginning are extracted and used in summary video.

2. **Still Images**

In this case summarization is done by using only the starting frames of the shots. Means a single will be extracted from each shot and put into the summary. We have done this method by making each starting frame to be on screen for near about 2secs.

4 .DEVELOPMENT OF SOFTWARE

Development of the software is done in various phases:

- OpenCV and CodeBlocks installation and Linking
- Video Shot Detection Implementation
- Video Summarization Implementation
- Merging Shot Detection with Video Summarization

4.1 OpenCV and CodeBlocks installation and Linking

The very thing that is required to start up with the project is the OpenCV libraries as we are working with the OpenCV for Video and Image Processing.

Next thing that is required is a Compiler to compiler our programs using OpenCV libraries. So we downloaded and installed OpenCV and CodeBlocks Compiler on our system.

Next task was to link the CodeBlocks compiler with the OpenCV library files that will be used to compile the program. So path to the OpenCV libraries were given to the Compiler.

4.2 Video Shot Detection Implementation.

Now we are to start with our implementation part after setting up the required environment. Shot Detection is implemented as follows:

- Take the Original video as input.
- Process the video to extract individual frames from the video for processing.
- Convert the frames which are in BGR (Blue Green Red) format to HSV (Hue Saturation Value) format.
- Plot histogram for every two consecutive HSV frames.
- Calculate histogram difference for the two consecutive frames.
- If diff > 0.3 (threshold), detect new shot.

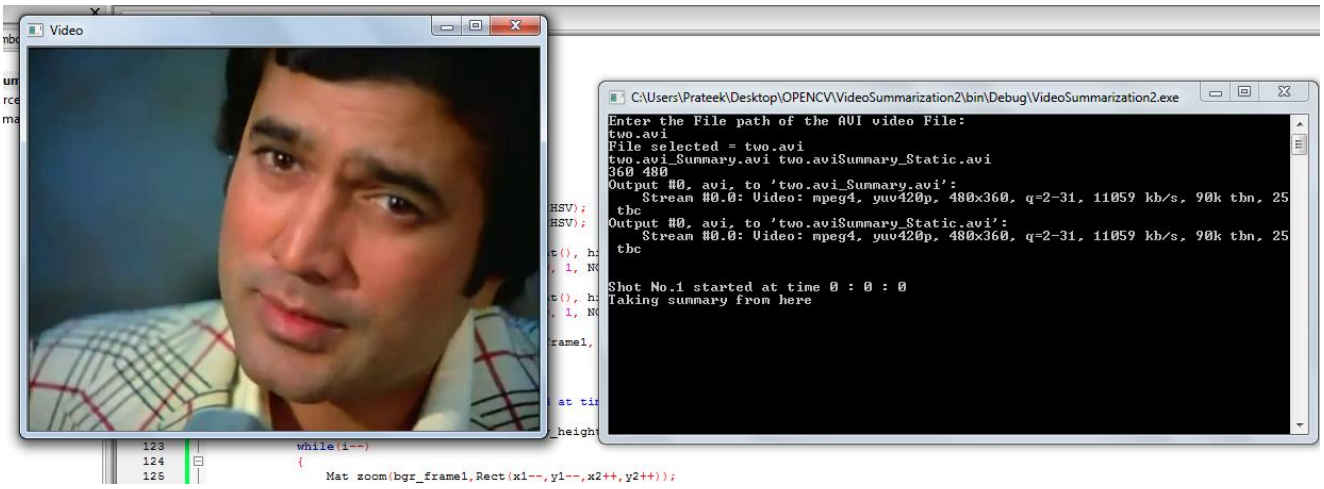

Fig 4.2.1

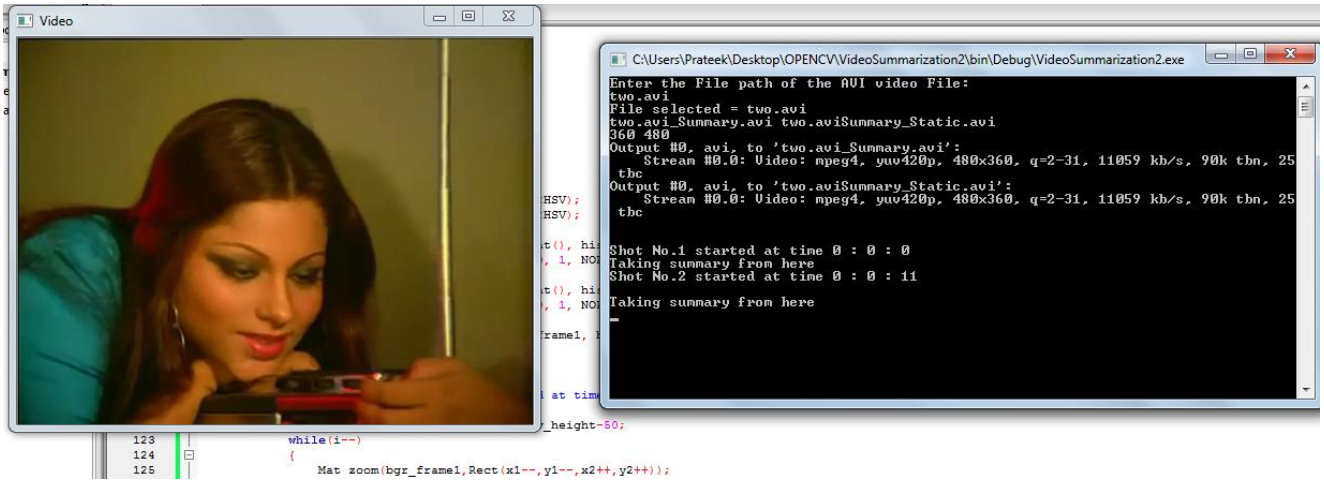

Fig 4.2.2

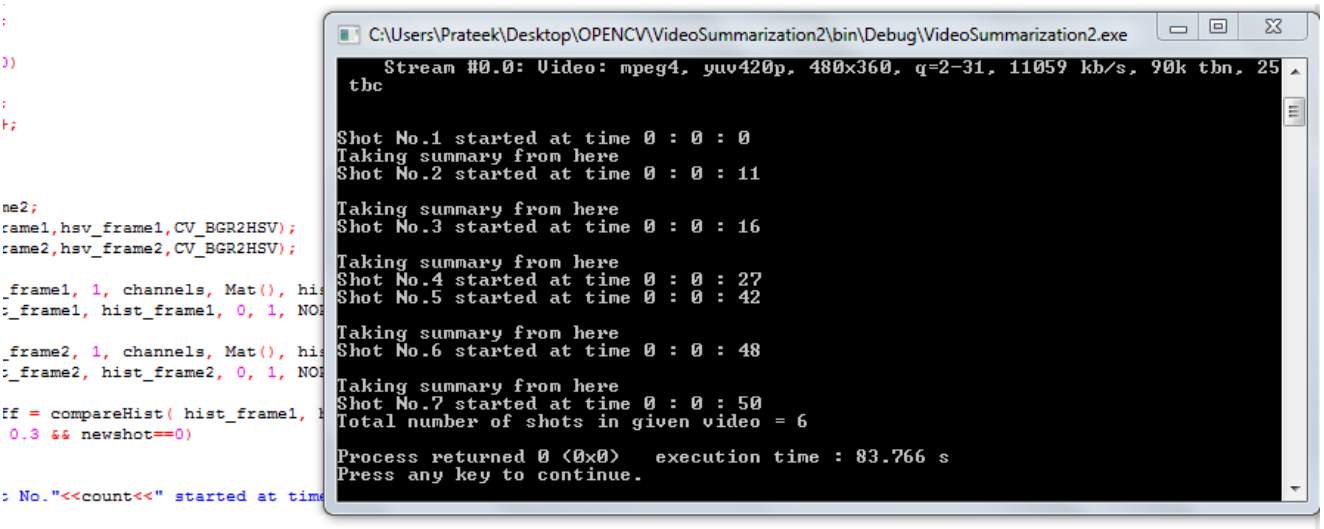

Fig 4.2.3

4.3 Video Summarization Implementation

Now since we are able to get the number of shots and their boundaries with the above algorithm, we can now apply Video Summarization algorithm on it. So summarization for Moving-Images works as follows:

- For every shot edges detected by the criteria difference > 0.3
- Check if difference > 0.45
- If yes, take the starting 3secs from the shot into the summary

And for Summarization with Still-Images method, follow the steps:

- For every shot edges detected by the criteria difference > 0.3
- Take the first frame of the shot
- Put it into the summary video. Hold it there for 2secs zooming out

Fig 4.3.1

4.4 Merging Shot Detection and Summarization

Now since we have both parts developed with us, its time to merge them now into a single program. So the complete program code goes like this:

- Take the Original video as input.
- Process the video to extract individual frames from the video for processing.
- Convert the frames which are in BGR (Blue Green Red) format to HSV (Hue Saturation Value) format.
- Plot histogram for every two consecutive HSV frames.
- Calculate histogram difference for the two consecutive frames.
- If diff > 0.3 (threshold), detect new shot.

  Take the still-image summary from first frame of the shot.

  If diff > 0.45 (threshold for shot select)

  Take summary from here taking first few frames.
- Print number of shots
- Output both summary videos

5. TESTING AND ANALYSIS

5.1 Testing Measures:

We have following three measures for detecting the quality of the shot detection algorithm:

Recall

Recall is the probability that an actual cut in the video will be detected. It is given by:

$$V = \frac{C}{C + M}$$

Precision

Precision is the probability that a cut detected by the algorithm is actually a cut.

$$P = \frac{C}{C + F}$$

F1

F1 is a combined measure for Recall and Precision. It will give us a higher value only when both recall and precision values are high.

$$F1 = \frac{2 * P * V}{P + V}$$

In the above formulas,

- C stands for number of correctly detected cuts.
- M stands for number of missed cuts.
- And F stands for number of false detected cuts

5.2 Program Results

Following table shows the summary of the results that we got for Shot Detection Implementation:

Video File	Video Length	Actual Shots	Shots Detected	Shots Missed	False Detection
one_dissolve.avi	0:30	7	6	1	0
two.avi	0:52	6	6	0	0
three_quick	1:03	19	14	5	0
four.avi	0:54	7	7	0	0
five_full.avi	5:11	42	40	3	1
blacknwhite.avi	0:24	7	5	2	0
MeraNaamJoker	10:29	108	98	14	4

And following table shows the summary of Summarization Method that we used with our program:

Video File	Video Length(mm:ss)	StaticSummaryLength(mm:ss)	MovingSummaryLength(mm:ss
one_dissolve.avi	0:30	0:12	0:11
two.avi	0:52	0:12	0:17
three_quick	1:03	0:23	0:36
four.avi	0:54	0:12	0:11
five_full.avi	5:11	1:20	2:06
blacknwhite.avi	0:24	0:10	0:07
MeraNaamJoker	10:29	2:55	4:07

5.3 Output Precision

Total number of actual shots = 196

Total number of shots detected (C) = 176

Total number of shots missed (M) = 25

Total number of false detections (F) = 5

Recall (V) = C / (C + M) = 176/ (176+25) = 0.875

Precision P = C / (C + F) = 176 / (176+5) = 0.972

Overall accuracy = (2 * P * V)/ (P + V) = 0.920

6. CONCLUSIONS

Shot Detection Algorithm using Histogram Difference is an easy to implement algorithm which works best with when encountered with hard cuts. But it has low potential in front soft cuts like fades, dissolves, wipes etc.

Although cuts like fades can be handled to some level in Histogram Difference by making some modifications and keeping some restrictions, efficiency obtained is still not up to the mark. So in case of soft cuts, better use more efficient algorithms like Edge Change Ratio.

For Video Summarization part, output was quite satisfactory as it reduced videos by more than half of their length. Also the output was explanatory giving suitable summary about the video. Still there are few better techniques than one we adopted and give a bit better results.

6 RECOMMENDATIONS AND FUTURE WORK

For future work in Shot Detection, Histogram Algorithm can be applied with the adaptive threshold instead of global threshold by keeping threshold value equal to the mean of histogram intensities in a particular shot. But this requires a pre-computation step for the before analyzing video. Hence can't be done online.

In case of Summarization, other techniques for taking keyframes can also be considered some of which includes clustering, genetic algorithm etc.

Appendix - Explanation of the Source Code

```
//////////////////Headers/////////////////////
#include <iostream>
#include <cv.h>
#include <highgui.h>
using namespace std;
using namespace cv;

int main()
{
   //////////////////Taking Video File in Input
   cout<<"Enter the File path of the AVI video File:\n";
   char filename[100],filename2[100];
   cin>>filename;
   VideoCapture input(filename);
   if(!input.isOpened())
   {
      cout<<"Error Opening video File";
      return -1;
   }
   else
   cout<<"File selected = "<<filename<<endl;


   ////////////////Generating Summary Video Names
```

```
strcpy(filename2,filename);
strcat(filename,"_Summary.avi");
strcat(filename2,"Summary_Static.avi");
cout<<filename<<" "<<filename2<<endl;



//////////Getting properties of the input Video
int v_height = input.get(CV_CAP_PROP_FRAME_HEIGHT);
int v_width = input.get(CV_CAP_PROP_FRAME_WIDTH);
cout<<v_height<<" "<<v_width<<endl;



//////////Declaring frames to store images
Mat bgr_frame1,bgr_frame2,hsv_frame1,hsv_frame2,dst;
input>>bgr_frame1; //input frame from video
namedWindow("Video", CV_WINDOW_AUTOSIZE); //window where to show the output



/// Declaring parameters for histogram calculation and difference
int h_bins = 50; int s_bins = 60;
int histSize[] = { h_bins, s_bins };
int fps = input.get(CV_CAP_PROP_FPS);
int ex = static_cast<int>(input.get(CV_CAP_PROP_FOURCC));
Size S = Size((int) input.get(CV_CAP_PROP_FRAME_WIDTH),    // Acquire input size
          (int) input.get(CV_CAP_PROP_FRAME_HEIGHT));
```

```
    // hue varies from 0 to 256, saturation from 0 to 180
    float h_ranges[] = { 0, 256 };
    float s_ranges[] = { 0, 180 };
    dst.create(v_height,v_width,ex);    //Dst object creation for zoom out effect

    const float* ranges[] = { h_ranges, s_ranges };

    // Use the o-th and 1-st channels
    int channels[] = { 0, 1 };
    MatND hist_frame1;
    MatND hist_frame2;
    int count=1;
    int newshot=2*fps, sum = 3*fps;



    ////////////////////Variables to store time
    int hours=0,mins=0,secs=0;



    ////////////Creating object for writing frames to video
    VideoWriter summary;
    summary.open(filename,ex,fps,S,true); //moving image summary object
    if(!summary.isOpened())
    {
        cout<<"\nProblem in opening output file. Video Summarization wont take place but Shot detection Will Work\n";
```

```
    }
    VideoWriter summary1;
    summary1.open(filename2,ex,fps,S,true); //still image summary object
    if(!summary1.isOpened())
    {
        cout<<"\nProblem in opening output file. Video Summarization wont take place but Shot detection Will Work\n";
    }



    cout<<"\n\n";
    cout<<"Shot No."<<count<<" started at time "<<hours<<" : "<<mins<<" : "<<secs<<"\n";
    cout<<"Taking summary from here\n";



    //extracting and processing frames
    int fcount=0;
    while(true)
    {


        fcount++;
        if(bgr_frame1.empty())
        break;
        if(newshot>0)
        newshot--;
        if(sum>0)
        {
            sum--;
```

```
        summary<<bgr_frame1; //writing frame to moving summary object
    }

    /////////keeping trach of time
    if(fcount>fps)
    {
        secs++;
        if(secs>=60)
        {
            secs=0;
            mins++;
        }
        if(mins>=60)
        {
            mins=0;
            hours++;
        }
        fcount=0;
    }
    input>>bgr_frame2;

    ///////BGR to HSV conversion
    cvtColor(bgr_frame1,hsv_frame1,CV_BGR2HSV);
    cvtColor(bgr_frame2,hsv_frame2,CV_BGR2HSV);
```

```
    //////////Histogram Calculation
    calcHist( &hsv_frame1, 1, channels, Mat(), hist_frame1, 2, histSize, ranges, true, false );
    normalize( hist_frame1, hist_frame1, 0, 1, NORM_MINMAX, -1, Mat() );

    calcHist( &hsv_frame2, 1, channels, Mat(), hist_frame2, 2, histSize, ranges, true, false );
    normalize( hist_frame2, hist_frame2, 0, 1, NORM_MINMAX, -1, Mat() );



    ////////////Histogram Difference Calculation
    double hist_diff = compareHist( hist_frame1, hist_frame2, 3 );



    //////threshold condition checking
    if(hist_diff > 0.3 && newshot==0)
    {
        count++;
        cout<<"Shot No."<<count<<" started at time "<<hours<<" : "<<mins<<" : "<<secs<<"\n";
        int i=50;
        int x1=50,y1=50,x2=v_width-50,y2=v_height-50;
        while(i--)
        {
            Mat zoom(bgr_frame1,Rect(x1--,y1--,x2++,y2++));
            resize(zoom,zoom,dst.size());
            summary1<<zoom; // writing frame to static summary object
            cvWaitKey(60);
        }
```

```
        newshot=2*fps;
    }

    ////////threshold checking for taking summary
    if(hist_diff > 0.45 && sum==0)
    {
        //summary <<bgr_frame1;
        cout<<"\nTaking summary from here\n";
        sum = 4*fps;
    }


    ///////output image being processed
    imshow("Video",bgr_frame1);
    bgr_frame1 = bgr_frame2.clone();

    if(waitKey(30) >= 0)
    continue;

  }

  ///////print the total number of shots in the video
  cout<<"Total number of shots in given video = "<<count-1<<endl;

  return 0;
}
```

REFERENCES

[1]M. Mills, “A magnifier tool for video data,” Proceedings of ACM Human Computer Interface, pp. 93–98, May 1992.

[2] Y. Taniguchi, “An intuitive and efficient access interface to real time in-coming video based on automatic indexing,” Proceedings of ACM Multimedia, pp. 25–33, November 1995.

[3] K. Otsuji, Y. Tonomura, and Y. ohba, “Video browsing using brightness data,” in Proceedings of International Society for Optical Engineering, vol. 1606, 1991, pp. 980–985.

[4] S. W. Smoliar and H. J. Zhang, “Content-based video indexing and retrieval,” IEEE transactions on Multimedia, pp. 62–72, 1994.

[5] H. J. Zhang, C. Y. Low, and S. W. Smoliar, “Video parsing and browsing using compressed data,” IEEE transactions on Multimedia Tools and Applications, vol. 1, pp. 89–111, 1995.

[6] Y.Pritch, A.Rav-Acha, and S. Peleg, “Non-chronological video synopsis and indexing,” IEEE Transactions on Pattern Analysis and Machine Intelligent, 2008.

[7] http://en.wikipedia.org/wiki/Shot_transition_detection

[8] Yael Pritch, Sarit Ratovitch, Avishai Hendel and Shmuel Peleg Clustered Synopsis of Surveillance video, The Hebrew University of Jerusalem, Israel 2009

[9] F. Chen, M. Cooper, and J. Adcock, “Video summarization preserving dynamic content,” in ACM TRECVID Video Summarization Workshop (TVS), 2007

[10] X. Yang and Z. Wei, “A novel video summarization algorithm,” in International Conference on Multimedia Technology (ICMT), 2011