Simple Implementations of Video Segmentation, Key Frame Extraction and Browsing

Kai Wu

***Abstract*—This report investigates the topic of video structure analysis. It includes shot boundary detection, key frames extraction and video browsing. Five different shot boundary detection algorithms are implemented and compared using videos including abrupt cuts and gradual shot transitions. These method belongs to two categories: pixel-to-pixel algorithms and histogram-based algorithms. It can be shown that all five methods work perfectly when there are only hard cuts in videos. When there are slow motion or gradual transitions in the videos, the performances of the algorithms decrease significantly. For each detected shot, a key frame is extracted for convenient browsing of the video content. The report starts with an overview of the areas of shot boundary detection, key frame extraction and video browsing. Then five proposed algorithms for shot boundaries are discussed in details. This follows by the implementation details of the system. Then different algorithms are applied to the test videos to test their performances. We conclude that all five methods work perfectly on the condition where only hard cuts exists in the videos. When there are slow motion or gradual shot transitions, the performance of the algorithms decrease significantly.**

***Index Terms*—Shot boundary detection, key frame extraction, video browsing.**

I. Introduction

The increased availability and usage of digital video has created a need for automated video content analysis techniques. Video analysis is commonly started with shot boundary detection. A shot is a consecutive sequence of frames that have strong content correlations. Therefore, shot is considered as the fundamental element of a video sequence and the building block for higher level semantic annotation and retrieval task. Generally, shot boundaries are classified as cuts in which the transition between successive shots is abrupt and gradual transitions which include dissolve, fade in, wipe, etc., stretching over a number of frames. Cut detection is easier than gradual transition detection [1], [8].

There are great redundancies among the frames in the same shot, if presented in its raw format, is rather unwieldy and costly. Therefore, certain frames that best reflect the shot contents are selected as key frames to succinctly represent the shot and allow rich user interaction.

Video browsing is a topic related closely to video content retrieval. According to [2], the video browsing application can be divided into 3 groups: applications that use video-player-like interaction, video retrieval applications, and browsing solutions based on video surrogates.

Kai Wu is with the Electric and Computer Engineering Department, University of British Columbia, Vancouver, BC, Canaca.(e-mail: imkaywu@gmail.com)

II. Background

A. Shot Boundary Detection

Shot boundary detection involves identifying the frame(s) where a transition takes place from one shot to another. The major techniques that have been used in this area are pixel differences, histogram comparisons, edge differences, statistical differences, and learning-based detection [1].

Pixel Differences

The simplest way to detect if two frames are significantly different is to count the number of pixels that change in value more than some threshold. It can achieve good results by selecting a threshold tailored to the input sequence. This method is sensitive to illumination, camera motion, and is slow. It's also impractical to manually adjusting the threshold.

Histograms

Histograms are the most common method used to detect shot boundaries. The simplest histogram method computes gray level or color histograms of the two images. If the bin-wise difference between the two histograms is above a threshold, a shot boundary is detected.

The color histogram-based shot boundary detection algorithm is one of the most reliable variants of histogram-based detection algorithm, refer to [7] for implementation. Its basic idea is that color content does not change rapidly within but across shots. Thus, hard cuts and other shot-lasting transitions can be detected as single peaks in the time series of the differences between color histograms of contiguous frames.

Edges Change Ratio

The number and position of edges in the edge detected images are compared. The edge change ratio (ECR) is defined as follows: Let σ_n be the number of edge pixels in frame n, X_n^{in} and X_{n-1}^{out} the number of entering and exiting edge pixels in frames n and $n-1$, respectively. The percentage of edges that enter and exit between the two frames was computed using formula as follows [4]:

$$ECR_n = \max\left(\frac{X_n^{in}}{\sigma_n}, \frac{X_{n-1}^{out}}{\sigma_{n-1}}\right)$$

Shot boundaries were detected by looking for large edge change ratio. Dissolves and fades were identified by looking at the relative values of the entering and exiting edge percentages. This method is more accurate at detecting cuts than histograms and much less sensitive to motion than chromatic scaling.

Statistical Differences

Statistical methods are motivated by the idea of breaking the images into regions and comparing statistical measures of the pixels in those regions. This method is reasonably tolerant of noise, but is slow due to the complexity of the statistical formulas. It also generates many false positives.

Learning-Based Approach

The statistical learning-based approach regards shot boundary detection as a classification task in which frames are classified as shot change or not shot change.

Supervised learning-based classifiers: The most commonly used supervised classifiers for shot boundary detection are SVM and Adaboost. The merits of the supervised-learning approaches are that there is no need to set the thresholds and different types of features can be combined to improve the detection accuracy. The limitation is their heavy reliance on a well-chosen training set containing both positive and negative examples.

Unsupervised learning-based classifiers: There are two categories: frame similarity-based and frame-based. The frame similarity-based algorithm cluster the measurements of similarity between pairs of frames into two clusters: the cluster with lower values of the similarities corresponds to shot boundaries and the other cluster with higher values of similarities corresponds to non-boundaries. The frame-based algorithms treat each shot as a cluster of frames that have similar visual content. The merit of clustering-based approaches is that the training dataset is not needed. The limitation is that the temporal information is missing and inefficiency in recognizing gradual transitions.

B. Key Frames Extraction

The extracted key frames should contain as much salient content of the shot as possible and avoid as much redundancy as possible. The number of frames should be related to the length of the shot. If the shot is shorter than a second, the middle frame was chosen and if the shot is longer, a key frame for each second was chosen.

The ideal method would be compare each frame to every other frame in the same shot and select the frame with the least difference from other frames in terms of a similarity measure [12]. However, this requires extensive computation and is not practical. Another approach is to select the first frame of the shot as the key frame. It's efficient, but not necessarily sufficient, as there can exist salient changes within a shot due to camera or object motion.

According to [6], current approaches to extract key frames are classified into six categories: sequential comparison-based, global comparison-based, reference frame-based, clustering based, curve simplification-based, and object/event-based.

Sufficient Comparison Between Frames

This method proceeds sequentially and only requires knowledge about the video sequence up until the current temporal position. Frames are compared to a previously extracted key frame until the visual content of one frame significantly differs from previously extracted key frames. This frame is selected as the next key frame.

Global Comparison Between Frames

The algorithm determines the shot boundaries by optimizing a predefined objective function that depends on the application. The function has one of the four forms.

1) *Even temporal variance:* The shot segment, each represented by a key frame, has equal temporal variance.
2) *Maximum coverage:* Maximize the representation coverage of each key frame.
3) *Minimum correlation:* Minimize the sum of correlations between key frames, making key frames as uncorrelated with each other as possible.
4) *Minimum reconstruction error*: Minimize the sum of the differences between each frame and its corresponding predicted frame reconstructed from the set of key frames using interpolation.

Reference Frame

These algorithms generate a reference frame and then extract key frames by comparing the frames in the shot with the reference frame.

Clustering

These algorithms cluster frames and then choose frames closet to the cluster centers as the key frames.

Curve simplification

These algorithms represent each frame in a shot as a point in the feature space. The points are linked in the sequential order to form a trajectory curve and then searched to find a set of points which best represent the shape of the curve.

Object/Event:

These algorithms jointly consider key frame extraction and object/event detection in order to ensure that the extracted key frames contain information about the objects or events.

C. Video Browsing

Video-Player-Like Interaction

Common video players use simple interaction as a means to navigate through the content of a video. Many efforts have been made to extend the simple video-based interaction model with a

more powerful means for content-based search and browsing. In [2], one study provided several additional features to the enhanced video browser: Time compression function, table of content, and shot seek feature, etc. Though this approach is useful in some scenarios, it cannot be adopted easily in interactive video retrieval.

Key-Frame-based browsing

Video browsing mechanisms are often combined with video retrieval methods in order to present the result to a query. In one of the earlier efforts [3], the concept of key frames was proposed for chronological browsing of the content of a video sequence [2], [9]. For visualization of the result, people proposed that good results be displayed in original size, somewhat similar results in a smaller size and bad results in an ever smaller size.

Key frames visualization approaches: page-based grid-like visualization, layered/hierarchical-based visualization, etc.

Video surrogates-based browsing

Video surrogates are alternative representations of the video content. The main purpose of video surrogates is to more quickly communicate the content of a video to the human observer. It's often used as a preview for a video and helps a viewer to decide whether the content is interesting or not.

III. Methods

Shot Boundary Detection

A. Pixel-To-Pixel

Pixel to pixel methods are probably the most straightforward ones. The similarity between two frames is measured by comparing their pixel values. Two pixel-to-pixel methods are implemented: global pixel-to-pixel and cumulative pixel-to-pixel method.

Global Pixel-To-Pixel

The method compare the sum of pixel intensities between two neighboring frames. The formula is as follows:

$$\frac{\left|\sum_{i=1}^{row}\sum_{j=1}^{col} I(t,i,j) - \sum_{i=1}^{row}\sum_{j=1}^{col} I(t-1,i,j)\right|}{256 \cdot row \cdot col} > \tau$$

This sum is computed over the whole image and $I(t,i,j)$ represents the intensity value of image at time frame t in pixel (i,j). A shot boundary is detected if the difference is larger than a predefined threshold value. It's obvious that the local differences between pixels' intensities are ignored and only global changes are taken into account.

Cumulative Pixel-To-Pixel

This method sums the difference between each pixel's intensity value in two neighboring frames. Local intensity values are considered in this method.

$$\frac{\sum_{i=1}^{row}\sum_{j=1}^{col}\left|I(t,i,j) - I(t-1,i,j)\right|}{256 \cdot row \cdot col} > \tau$$

B. Histogram

Histograms give us a better refection of the global property of image. In the area of shot boundary detection, histogram-based methods are more robust to a camera or object motion. However, there are some drawbacks in using histogram based methods, a shot occurring in two frames with the similar histograms will be missed.

Simple Histogram

The simple histogram method calculates histogram of each color channel that form the image, and compute the difference between the bins in each histogram of the two frames using the following formula:

$$\frac{\sum_{c\in\{channels\}}\sum_{b=0}^{bins}\left|H(t,c,b) - H(t-1,c,b)\right|}{2 \cdot \left|pixels\right| \cdot \left|channels\right|} > \tau$$

Where $H(t,c,b)$ is the value of the b bin in color channel c for histogram of the image at time t.

Max Histogram

The max histogram method computes the difference between the histograms of all channels in the two images and choose the channel that has maximum summation.

$$\frac{\max_{c\in\{channels\}}\sum_{b=0}^{bins}\left|H(t,c,b) - H(t-1,c,b)\right|}{2 \cdot \left|pixels\right|} > \tau$$

Weighted Histogram

The weighted-histogram method takes into account the histograms' difference in all channels, and give each one of them a weight determined by the luminance proportion of the corresponding channel and the whole image. Therefore, the prevalent color channel in the image is given more weight.

$$\frac{\sum_{c\in\{channels\}}\sum_{b=0}^{bins}\frac{w_c}{w_{mean}}\left|H(t,c,b) - H(t-1,c,b)\right|}{2 \cdot \left|pixels\right| \cdot \left|channels\right|} > \tau$$

C. Edge Comparison

Hausdorff Method

Hausdorff method performs an edge detection process on the images and compare the location of the edge points produced by the edge detector. The method then checks for each edge point

in one frame, if there exist a correlating edge point in the neighboring frame, within a window of predefined size. If there is no correlating edge point in the other frame, this edge point is labeled as "outlier".

$$\frac{\sum_{(i,j,t)\in\{\text{edge\_points}\}} \Delta_{i,j,t}}{|edges|} > \tau,$$

$$\Delta_{i,j,t} = \begin{cases} 0, & (i+w_1, j+w_2, t-1) \in \{\text{edge\_points}\}, \\ & -|window| \le w_1, w_2 \le |window| \\ 1, & otherwise \end{cases}$$

Key Frame Extraction

In order to alleviate the calculation of the system and avoid frame-by-frame computation, I adopted the simplest solution: extract the first frame in the shot as the key frame of this shot.

Video Browsing

Traditionally, viewers have to go through the whole video to know the content in this video, and the purpose of video browsing is to give the viewers a general idea of the video content. This is achieved by construct a "table of content" using the key frames of the video. In current implementation of the system, I adopted the video-like interaction and key-frame based interface for video browsing: the UI displays a video player window and a key frame window.

IV. Implementation

A. Test Data

I've chosen movie clips and security camera footages as the main data set. All the input videos are digitized at a size of 320×240 pixels at a frame rate of 24 frames per second. The digitized video was stored as AVI, and typically are 3-4 minutes long. The videos contains mostly cuts, some including slow motion and gradual transitions.

The two types of videos I selected were:

1. *Movies*: 4 clips of Godfather, 4 clips of the TV series Friends, 1 clip of The Matrix (including slow motion).
2. *Security camera footages*: 1 clip of an unexpected event.

The characteristic of the test videos is presented in Table 1.

Video Name	Frames No.	Shot Boundaries No.(cut/slow trans)
godfather - 1	3812	14
godfather - 2	2423	5
godfather - 3	3330	9
godfather - 4	6378	38
friends - 1	2114	24
friends - 2	2690	20
friends - 3	3265	24
friends - 4	3072	55
scent - 2	7526	37
matrix - 1	432	18 (17/1)
SC footage - 1	1468	1

TABLE 1. video test data

B. System Implementation

The GUI and core algorithm are implemented using Java with the help of the open source API (JFreeChart, VLCJ) for the dot-line chart and video playing functionality, and can run on Windows platform. With minor modifications, it can also run on Linux or Mac platforms.

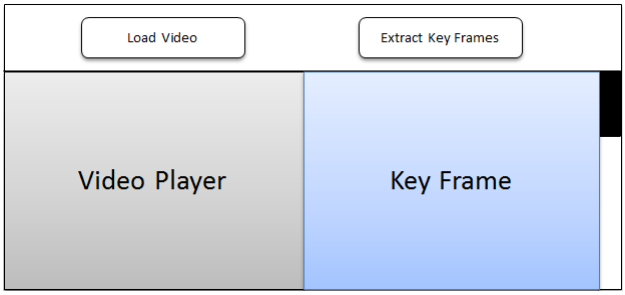

Fig. 1. UI of the System

Figure 1 is the UI of the system, the 'Load Video' button is used to load the video manually and then start the processing of the video. The output is a text file containing the distances between the neighboring frames. The 'Extract Key Frame' button uses the output file to detect the boundaries of shots using user-defined threshold. The UI is updated after shot boundaries detection with a video and key frame combined panel. The left of the panel is used to play processed video and the right one display key frames of the corresponding video.

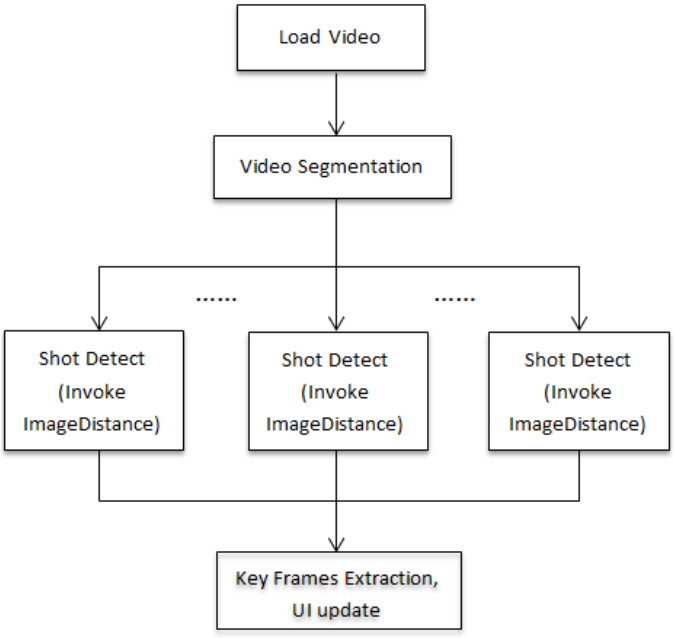

Fig. 2. Flowchart of the software

Fig. 2 is the flowchart of the system showing the relationships of all the working parts.

V. RESULTS

A. *Performance of different Methods*

This section shows the result of shot boundary detection of each algorithm after frame-by-frame computation of the distance between neighboring frames.

Global Pixel-To-Pixel

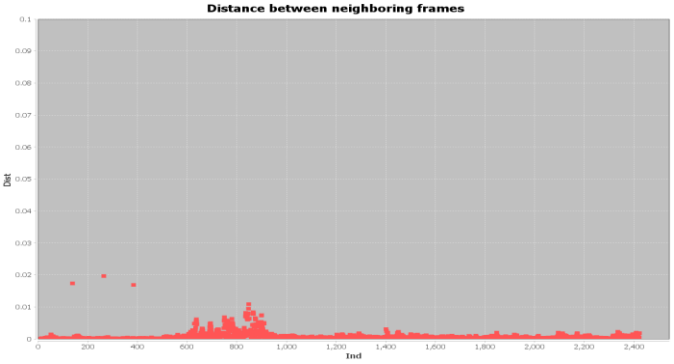

Fig. 3. The distance between neighboring frames using global pixel-to-pixel method

Fig. 4. Extracted Key Frames

Cumulative Pixel-To-Pixel

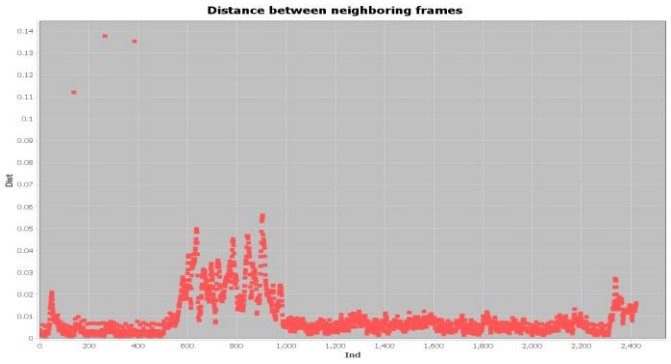

Fig. 5. The distance between neighboring frames using cumulative pixel-to-pixel method

Simple Histogram

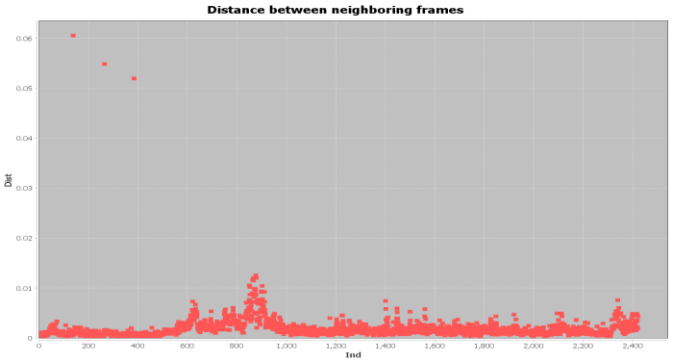

Fig. 6. The distance between neighboring frames using simple histogram method

Max Histogram

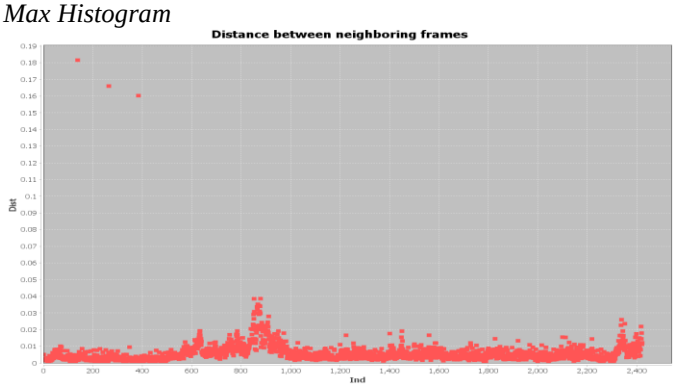

Fig. 7. The distance between neighboring frames using max histogram method

Weighted Histogram

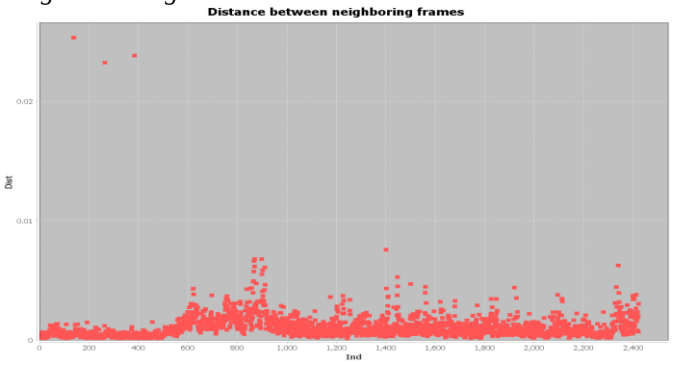

Fig. 8. The distance between neighboring frames using weighted histogram method

As we can see from results of the aforementioned 5 methods, they all respond similarly to the shot boundaries, especially for the first 3 ones. But when it comes to a slow transition, the methods produce a long-stretched peak which can be hard to detect using predefined threshold. This gradual shot transition issue has always been the biggest challenge in the area of shot boundary detection, here we allow more than one key frame extracted for a slow transition shot, namely, as long as the slow transition is detected, we won't penalize the algorithms if more than one key frame are extracted.

B. Experimental Result

It's difficult to compare shot boundary detection methods because it requires a large digitized and manually indexed data set. Few published studies report quantitative results and those that do involve limited test video sequences. In [10] there is a relatively comprehensive comparison.

For each algorithm, each video sequence, we measure the number of shot boundaries that were correctly detected, the number of false positives, and the number of missed boundaries. A gradual transition was correctly detected if any of the frames of the transition was marked as a shot boundary. Algorithm were not penalized for reporting multiple consecutive frames as shot boundaries.

I choose recall and precision as appropriate evaluation criteria. Recall is defined as the percentage of desired items that are retrieved. Precision is defined as the percentage of retrieved items that are desired items.

$$\text{Recall}=\frac{\text{Correct}}{\text{Correct}+\text{Missed}},$$

$$\text{Precision}=\frac{\text{Correct}}{\text{Correct}+\text{FalsePositive}}.$$

I chose two videos to analyze the performance of different algorithms, one is godfather – 3, the other one is matrix. The first one includes only hard cuts while the second one have camera motion and slow transitions.

Method	Precision	Recall
Global pixel2pixel	100%	100%
Cumulative pixel2pixel	100%	100%
Simple histogram	100%	100%
Max histogram	100%	100%
Weighted histogram	100%	100%

TABLE 2. performance of the methods (test data: godfather - 3)

Because in this video clip, there is no camera motion or slow transition, all the shot boundaries are hard cuts, which makes the task of detecting shot boundaries a lot easier. All five algorithm can achieve 100% detection rate with appropriate thresholds.

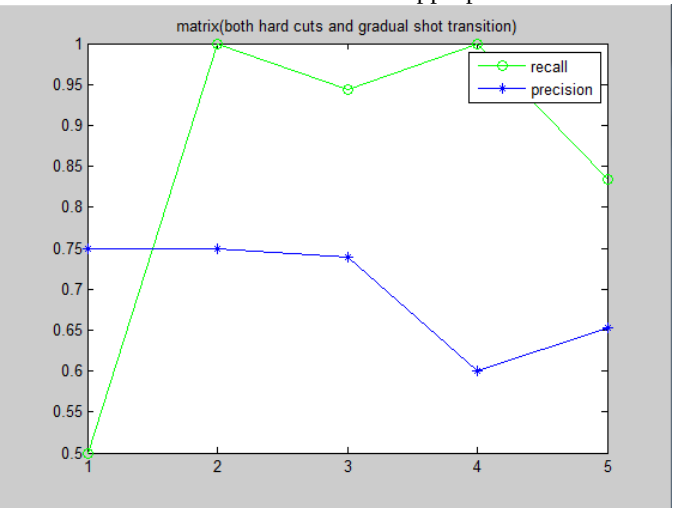

Fig. 9. performance of the algorithms (test data: matrix)

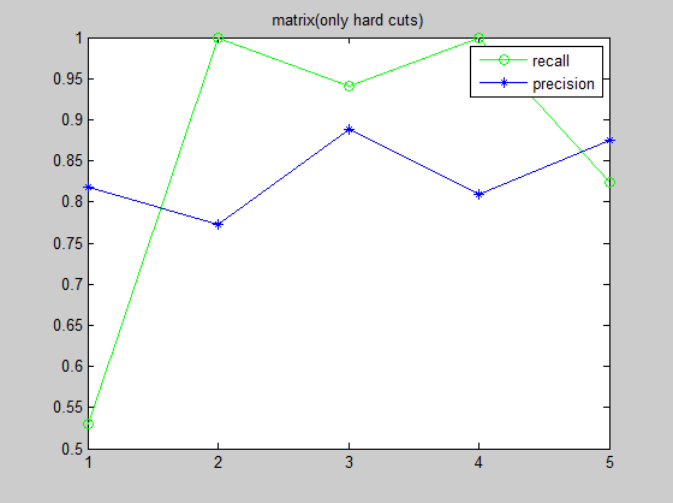

Fig. 10. performance of detecting hard cuts (test data: matrix)

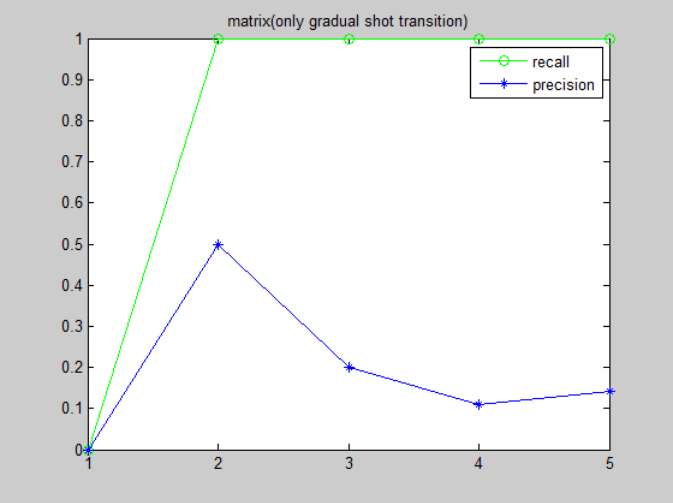

Fig. 11. performance of detecting gradual shot transitions (test data: matrix)

Due to the camera motion and gradual transition of the shots, the precision and recall rate decrease significantly. Most of the hard cuts can be detected really accurately for all five methods, as shown in Figure10, the problem comes from the slow motion in this clip. From the data I collected, the histogram-based algorithm performs similarly in this scenario and cumulative pixel-to-pixel method performs the best. As mentioned in [5], the simpler algorithms often outperform the more complicated ones because these complicated algorithms are sensitive to the threshold settings and "hidden" parameters not specified in the literature. And because cumulative pixel-to-pixel also takes into consideration the local pixel values, thus yields the best result.

However, this shows again that slow transition detection is a much harder situation to detect and none of aforementioned algorithms can yield a satisfactory results.

VI. CONCLUSION

We can see that the algorithm performs perfectly when the videos only contain hard cuts (See Appendix for more tests). But when it comes to detect gradual shot transitions, the

aforementioned five methods all have limitations. Generally, the histogram-based algorithm and cumulative pixel-to-pixel methods perform relatively well when there are slow shot transitions, and the histogram-based methods are relative faster compared to cumulative pixel-to-pixel method.

VII. Future Work

Shot boundary detection is a quite intuitive but really tough research topic. Here are some possible research directions to improve the result.

1. The detection of gradual shot transitions. This issue remains No. 1 challenge to shot boundary detection area. Because the complicated algorithms often yield worse results compared to the simpler ones, while the simpler ones aren't sophisticated enough to achieve good results.
2. The current implementations of the shot boundary detection algorithms all require a user-defined threshold. This is impractical and tedious in practical situations. So there should be ways to automatically determine the thresholds for the various videos.
3. The computation of shot boundary detection is massive because it requires frame-by-frame calculation. So more efficient algorithms or parallel computation should be taken into consideration in order to achieve real-time performance.
4. After detecting the shot boundaries of the videos, an video abstraction is needed to get rid of the redundant information and maintain as much content information as possible. Current implementation of key frame extraction is the simplest method out there, there should be a layered/hierarchical representation to give an more informative overview of the videos.

Appendix

The source code can be found here in Github: https://github.com/imkaywu/ShotDetection. Download it and try it out yourself, feel free to modify the code. Please contact me via email (imkaywu@gmail.com) if you find any bugs, or have some ideas or improvements.

UI-related class	MainFrame.java
	MyMenu.java
	VideoPanel.java
	DistChart.java
Listener class	LoadViedoListener.java
	KeyFrameExtractionListener.java
Multi-thread class	VideoSegmentation.java
	ShotDetect.java
Algorithm-related class	ImageDistance.java
	Histogram.java
	ColorChannelSpliter.java
	ChanelsImageSplitter.java
	KeyFrameExtraction.java

TABLE 3. The structure of the Java Project

The recall and precision for some of the test videos:

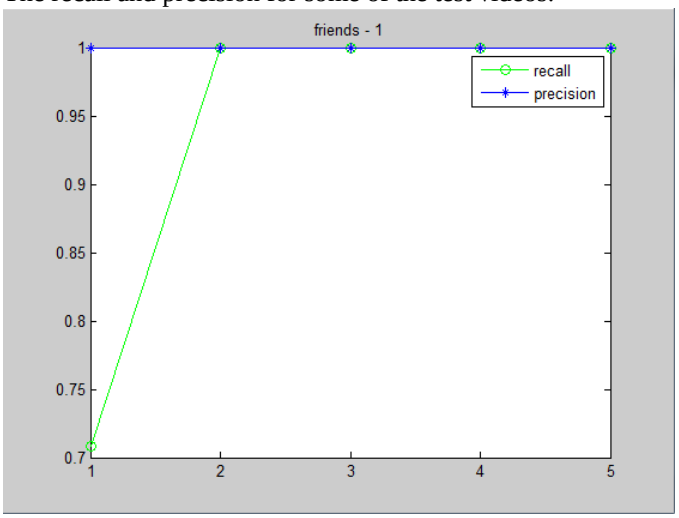

Fig. 12. Recall and precision the test video 'friends - 1'

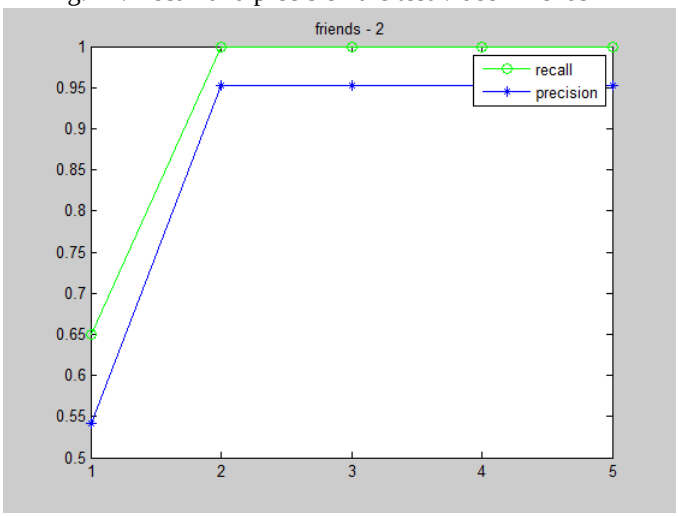

Fig. 13. Recall and precision the test video 'friends - 2'

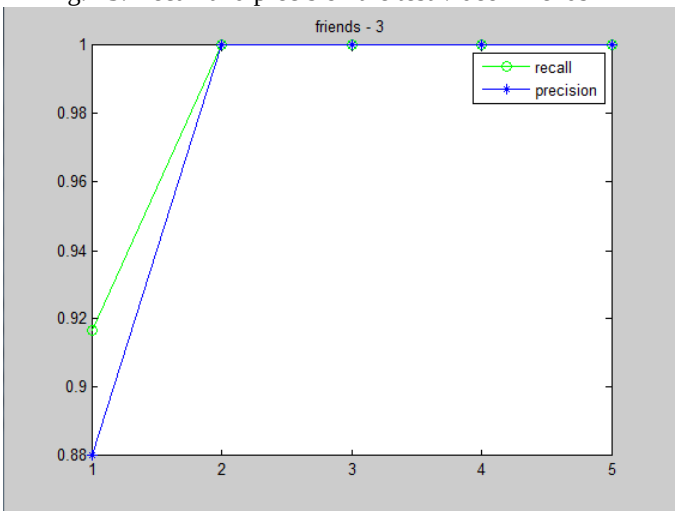

Fig. 14. Recall and precision the test video 'friends - 3'

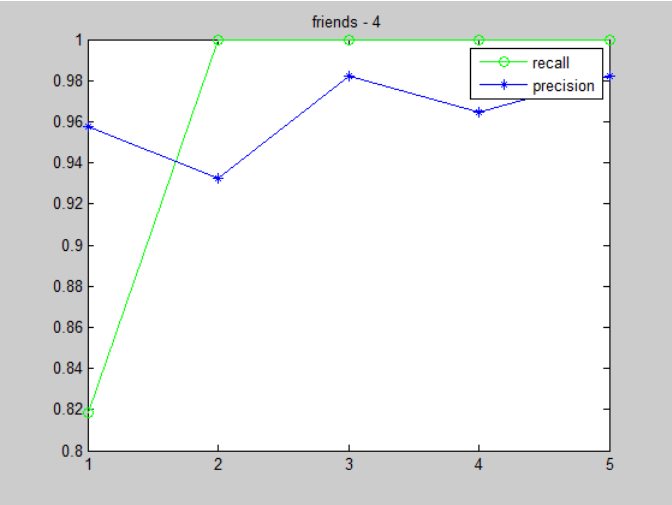

Fig. 15. Recall and precision the test video ‘friends - 4’

REFERENCES

[1] Hu W, Xie N, Li L, et al. A survey on visual content-based video indexing and retrieval[J]. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 2011, 41(6): 797-819.

[2] Schoeffmann K, Hopfgartner F, Marques O, et al. Video browsing interfaces and applications: a review[J]. Journal of Photonics for Energy, 2010: 018004-018004-35.

[3] F. Arman, R. Depommier, A. Hsu, and M. Chiu, “Content-based browsing of video sequences,” Proc. Second ACM Intl. Conf. Multimedia, pp. 97–103 (1994).

[4] Lienhart R W. Comparison of automatic shot boundary detection algorithms[C]//Electronic Imaging'99. International Society for Optics and Photonics, 1998: 290-301.

[5] Boreczky J S, Rowe L A. Comparison of video shot boundary detection techniques[J]. Journal of Electronic Imaging, 1996, 5(2): 122-128.

[6] Truong B T, Venkatesh S. Video abstraction: A systematic review and classification[J]. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), 2007, 3(1): 3.

[7] Mas J, Fernandez G. Video shot boundary detection based on color histogram[J]. Notebook Papers TRECVID2003, Gaithersburg, Maryland, NIST, 2003.

[8] Dimitrova N, Zhang H J, Shahraray B, et al. Applications of video-content analysis and retrieval[J]. IEEE multimedia, 2002, 9(3): 42-55.

[9] Snoek C G M, Worring M. Concept-based video retrieval[J]. Foundations and Trends in Information Retrieval, 2008, 2(4): 215-322.

[10] Smeaton A F, Over P, Doherty A R. Video shot boundary detection: Seven years of TRECVid activity[J]. Computer Vision and Image Understanding, 2010, 114(4): 411-418.

[11] Binshtok M, Greenshpan O. Segmentation of Video Incorporating Supervised Learning, course project of Introduction to Computational and Biological Vision.

[12] Yang Y, Lin M. A Survey on Content based Video Retrieval.