

VIDEO CUT PROCESSING

REPORT OF PROJECT SUBMITTED FOR PARTIAL FULFILLMENT OF
THE REQUIREMENT FOR THE DEGREE OF
BACHELOR OF TECHNOLOGY
In
INFORMATION TECHNOLOGY

By

Abhisikta Halder

Registration No– 101170110090

University Roll No- 11700210038

Shaoni Mitra

Registration No– 101170110148

University Roll No- 11700210039

UNDER THE SUPERVISION OF

Dr. Siddhartha Bhattacharyya

HOD, Department of IT, RCCIIT



AT

RCC INSTITUTE OF INFORMATION TECHNOLOGY
[Affiliated to West Bengal University of Technology]
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA – 700015

RCC INSTITUTE OF INFORMATION TECHNOLOGY

KOLKATA – 700015, INDIA



CERTIFICATE

The report of the Project titled Video Cut Processing submitted by Abhisikta Halder (Roll No.: 11700210038) and Shaoni Mitra (Roll No.: 11700210039) of B. Tech. (IT) 8th Semester of 2014) has been prepared under our supervision for the partial fulfillment of the requirements for B.Tech (IT) degree in West Bengal. University of Technology. The report is hereby forwarded.

Dr. Siddhartha Bhattacharyya
HOD, Dept. of IT
RCCIIT, Kolkata
(Internal Supervisor)

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Dr. Siddhartha Bhattacharyya of the department of Information Technology, whose role was invaluable for the project. We are extremely thankful for the keen interest he took in advising us, for the books and reference materials suggested and for the moral support extended to us.

We convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staff for the gracious hospitality they offered us.

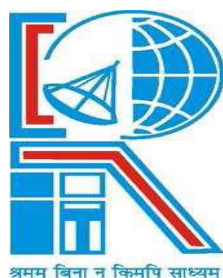
Place: RCCIIT, Kolkata

Date: 16.05.2014

Abhisikta Halder
Registration No– 101170110090
University Roll No- 11700210038
B. Tech (IT) – 8thSemester, 2014, RCCIIT

Shaoni Mitra
Registration No– 101170110148
University Roll No- 11700210039
B. Tech (IT) – 8thSemester, 2014, RCCIIT

RCC INSTITUTE OF INFORMATION TECHNOLOGY
KOLKATA – 700015, INDIA



CERTIFICATE OF ACCEPTANCE

The report of the Project titled Video Cut Processing submitted by Abhisikta Halder (Roll No.: 11700210038) and Shaoni Mitra (Roll No.: 11700210039) of B. Tech. (IT) 8th Semester of 2014) is hereby recommended to be accepted for the partial fulfillment of the requirements for B.Tech (IT) degree in West Bengal. University of Technology.

Name of the Examiner

Signature with Date

1.

.....

2.

.....

3.

.....

4.

.....

SL. NO.	CONTENTS	PAGE NO.
1.	Introduction	6
2.	Problem Statement	7
3.	Problem Definition and Discussion	8-20
4.	Literature Survey	21-32
5.	Analysis and Requirement Specification	33-38
6.	Planning	38
7.	Design	39
8.	Results analysis and Discussion	39
9.	Conclusion and Future Scope	43
10.	References / Bibliography	44

Introduction

What is Video?

- Refers to recording, manipulating, and displaying moving images, especially in a format that can be presented on a television.
- Refers to displaying images and text on a computer monitor. The video adapter, for example, is responsible for sending signals to the display device.
- A recording produced with a video recorder (camcorder) or some other device that captures full motion.

Video data, unlike text data, are hard to index, browse, search and retrieve. Traditional techniques of data access are not appropriate for video as the information it contains is subjective in nature. It is this subjectivity that makes representation and interpretation of video information difficult for automated processing. Manual annotation of a large video archive is possible but it is an arduous task and also not cost effective. The challenge of automatic processing of video data motivates researchers all over the world to engage themselves in finding efficient methodologies that organize video data and extract semantically meaningful information

Hierarchical structure of a video

A video can be hierarchically structured as **video** → **scenes** → **shots** → **frames** → **pixels**.

VIDEO SHOTS

A video shot is a sequence of frames recorded in a single camera take, has been identified as semantically as well as temporally coherent basic unit of a video. Segmenting a video into shots is the first step of video browsing, content based retrieval, genre-classification and many other high level video analysis tasks.

- Segmenting a video into shots is the first step of **video cut processing**,
- **video browsing, content based retrieval, genre-classification** and many other high level video analysis tasks.

The purpose of shot boundary detection is to find location of shot transition, which can be of two types:

- **cut**
- **gradual**

Problem Statement

CUT

- A cut is an abrupt change where one image belongs to the previous shot and the next image is of a new shot that is a scene changes from one to another.
- It is an abrupt transition between two video clips or image sequences.
- It is where one clip ends and another begins.
- It can also mean to shorten a clip or divide it into two.

The *cut* is the most common type of shot boundary detection and find location of shot transition.

In video editing and live switching, cuts are fast and efficient. Once a scene has been established, cuts are the best way to video transition. It simply means replacing one shot instantly with the next.

When you shoot video footage on your camera, there is a cut between each shot, i.e. between when you stop recording and start recording the next shot. Although some cameras do offer built-in transitions, most recorded footage is separated by cuts.

It can keep the action rolling at a good pace. Other types of transition can slow the pace or even be distracting.

Of course there are some situations where fancier transitions are in order. Certain genres of television, for example, rely on a variety of transitions. Even in these productions though, notice how many transitions are still simple cuts.

A common mistake amongst amateurs is to shun the cut in favour of showiness, adding wipes and effects between every shot. Learn to avoid temptation and stick to the basics. The video shots are what the audience wants to see, not how many transitions your editing program can do.

GRADUAL

Gradual shot transition takes place over a short period of time (over a few video frames). The two most common gradual shot transitions are **fade in/out** and **dissolve**.

A **fade out** is a slow decrease of luminance petering out to a monochrome frame and a **fade in** is a gradual increase in luminance starting from a monochrome frame.

A **dissolve** is said to occur if frame images from two consecutive shots temporally superimpose. The luminance of disappearing frame images gradually gets dimmer and that of appearing frame images becomes brighter simultaneously.

Problem Definition and Discussion

VIDEO SCENE CUT DETECTION

For the mainstream hard cut scene detection algorithm based on the color-histogram difference new criteria are added to increase precision and improve computational efficiency. This approach utilizes additional detection criterion based on spatial distribution of luminance blocks difference during scene cut. Also, in order to improve computational efficiency, time sampling at larger time resolutions and detection at smaller time resolutions is used. On the different video examples the results in precision and reliability are given.

Video cut detection has two main purposes:

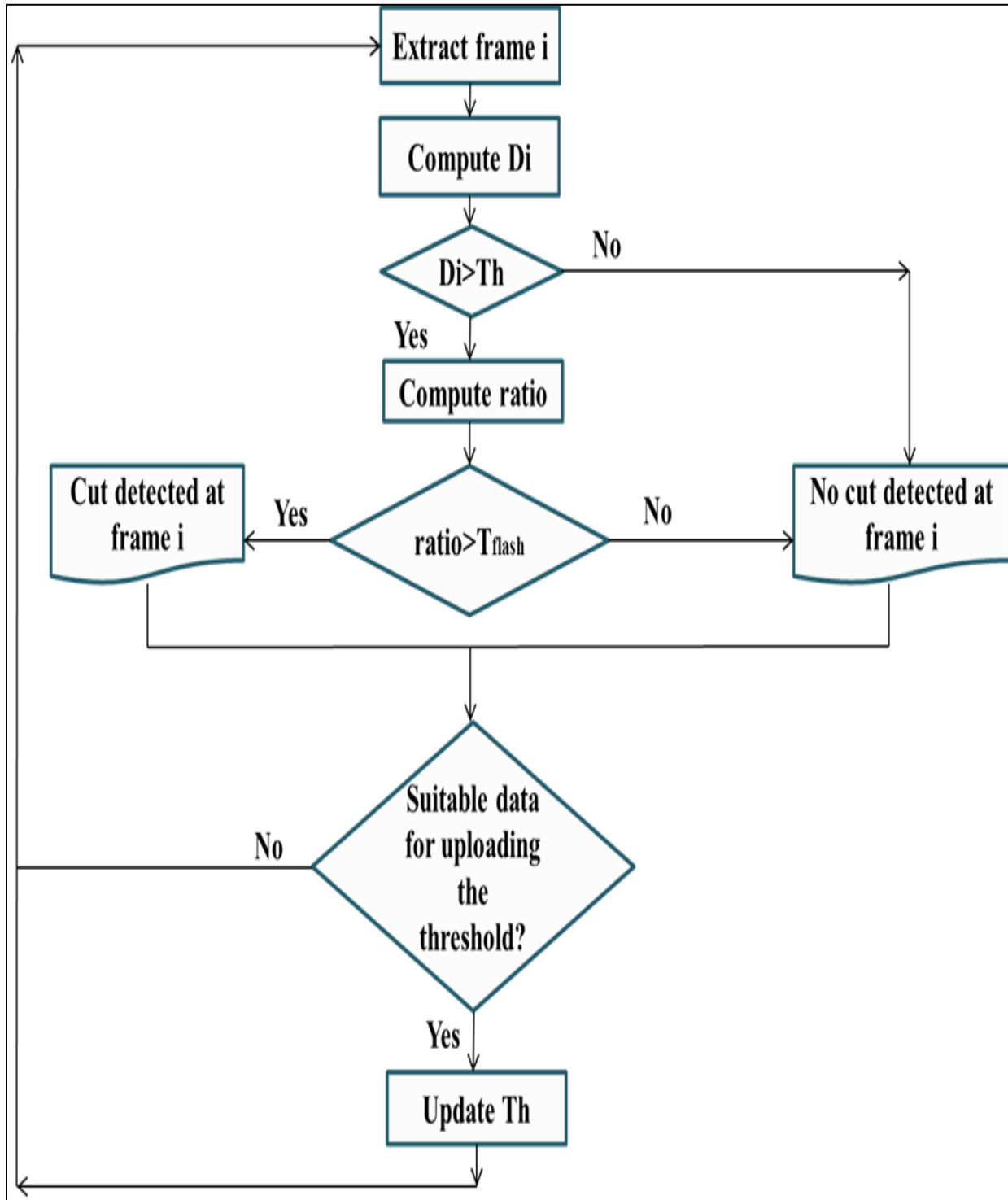
- to delimit the start and the end of the video shots and
- to process the video content in a more efficient way.

The basic idea of video cut detection algorithms is

- to compute the differences between consecutive frames or groups of frames

existing techniques differ in the way these differences are computed.

VIDEO CUT DETECTION ALGORITHM



Reliable Transition Detection in Videos

In recent years the research on automatic shot boundary detection techniques has exploded, and one may wonder why solving the problem of automatic shot boundary detection is that important. There are various reasons why reliable transition detection is needed:

Firstly: shots are generally considered as the elementary units constituting a video. Detecting shot boundaries thus means recovering those elementary video units, which in turn provide the ground for nearly all existing video abstraction and high-level video segmentation algorithms.

Secondly: during video production each transition type is chosen carefully in order to support the content and context of the video sequences. Automatically recovering all their positions and types, therefore, may help the computer to deduce highlevel semantics. For instance, in feature films dissolves are often used to convey a passage of time. Also dissolves occur much more often in features films, documentaries, biographical and scenic video material than in newscasts, sports, comedy and shows. The opposite is true for wipes.

Therefore, automatic detection of transitions and their type canbe used for automatic recognition the video genre. Shot detection is also useful to colorblack-and-white movies. For each shot a different gray-to-color look-up table is chosen.

Often shot boundary detection algorithms are strictly classified by whether the originally proposed detection algorithm was designed to operate on uncompressed or compressed videos streams. In this survey, we do not make this artificial distinction, since practically all proposed detection algorithms can be applied to the compressed as well as to the uncompressed domain. There may be slight differences in how certain features such as color histograms, edge maps, and motion information are derived, but the core concept of the classification algorithm and the kind of feature(s) that is exploited remains basically unaffected by that choice. This is especially true since compressed processing often only stands for working on so-called DC images, i.e., images which are sub-sampled by a factor of 8 after applying an 8x8 block filter.

Shot transitions can be classified into four classes based on the 2D image transformations applied during transition production:

- (i) Identity class: Neither of the two shots involved are modified, and no additional edit frames are added. Only hard cuts qualify for this class.
- (ii) Spatial Class: Some spatial transformations are applied to the two shots involved. Examples are wipe, page turn, slide, and iris effects.
- (iii) Chromatic Class: Some color space transformations are applied to the two shots involved. Examples are fade and dissolve effects.
- (iv) **Spatio-Chromatic Class:** Some spatial as well as some color space transformations areapplied to the two shots involved. All morphing effects fall into this category. Note

that in practice often all effects in the spatial class in principle fall into the spatio-chromatic applied to the two shots involved. All morphing effects fall into this category. Note that in practice often all effects in the spatial class in principle fall into the spatio-chromatic class since some chromatic transformations are always applied at the boundary between the pixels of the first and second shot such as anti-aliasing, smoothing or shading operations.

Transition classification scheme for transition detection

TYPE OF TRANSITION	SPATIALLY SEPARATED	TEMPORALLY SEPARATED
HARD CUT	YES	YES
FADE	YES	YES
WIPE, DOOR, SLIDE	YES	NO
DISSOLVE	NO	NO

VIDEO SCENE CUT DETECTION

Video Scene Cut Detection or Scene Change Detection(SCD) is one of the fundamental problems in the design of a video database management system(VDMS).

Scene change detection(SCD) is one of the fundamental problems in the design of a video database management system(VDMS).It is the first step towards the automatic segmentation, annotation and indexing of video data.SCD is also used in other aspects of VDMS eg. Hierarchical representation and efficient browsing of the video data.

SCD is usually based on some measurements of the image frame, which can be computed from the information contained in the images. Those information can be color, spatial correlation, object shape, motion contained in the video image, or DC coefficients in the case of compressed video data.

TYPES OF SCENE CHANGE DETECTION

Scene change in a video sequence can be of two types:

Scene change in a video sequence can be of two types:

- **abrupt scene change**
- **gradual scene change**

Abrupt scene changes results from editing “cuts” and detecting them. In general, it is easier than gradual scene changes.

Gradual scene changes results from chromatic edits, spatial edits and combined edits. It includes special effects like zoom, camera pan, dissolve and fade in out etc. In general, gradual scene changes are more difficult to detect than the abrupt scene changes, and may cause lots of scene detection algorithms to fail under certain circumstances.

ABRUPT SCENE-CHANGE DETECTION

➤ **Method**

- **Measurement of the Changes Between Frames**
 - Pixel-Based Difference
 - Histogram-Based Difference
- **Static Scene Test**
- **Scene Transition Classification**
- **Detection Algorithm**

PHASES OF DETECTION

- **First phase**
 - locate the highest and the second highest peaks of DC(Direct Cosine) image difference in the midst of the sliding window, and then **calculate the ratio** between the first and second peaks
- **Second phase**
 - **Histogram Measure**
 - **Static Scene Test**

- ✓ most of the false alarms declared by the histogram detector are due to sudden light changes, while the edge information is more or less invariant to these changes

- Scene Transition

PIXEL-BASED DIFFERENCE

If “f” and “g” be two gray value image functions the discontinuity feature value using pixel differences can be calculated using the formula:

$$d(f, g) = \sqrt{\sum_{i=1}^n \sum_{j=1}^m (f(i, j) - g(i, j))^2}$$

If the input image is a color image the value of “d” calculated separately for the “R”, “G” and “B” values.

HISTOGRAM-BASED DIFFERENCE

The color histogram of an image can be computed by dividing a color space e.g. RGB, into discrete image colours called bins and counting the number of pixels that fall into each bin. The difference between two images **I_i** and **I_j** based on their colour histograms **H_i** and **H_j** can be formulated as

$$d(I_i, I_j) = \sum_{k=1}^n |H_{ik} - H_{jk}|$$

The χ^2 histogram computes the distance measure between two image frames as

$$d(I_i, I_j) = \sum_{k=1}^n \frac{(H_{i(k)} - H_{j(k)})^2}{H_{j(k)}}$$

- Efficient representation
 - Easy computation
 - Global color distribution

Insensitive to where **H_{i(k)}** and **H_{j(k)}** are the bin values of the histogram of frame **i** and **j** respectively and **k** is the overall number of bins

Color Histogram

- Efficient representation
 - Easy computation

- Global color distribution

Insensitive to

- Rotation
- Zooming
- Changing resolution
- Partial occlusions

Disadvantage

- Ignore spatial relationship
- Sensitive in illumination changes
- Choose illumination-insensitive color channels

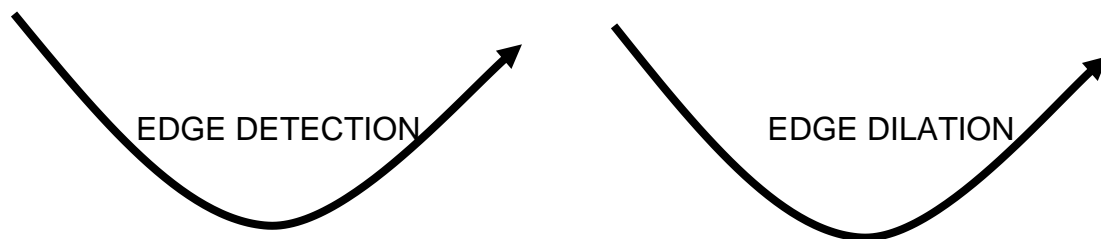
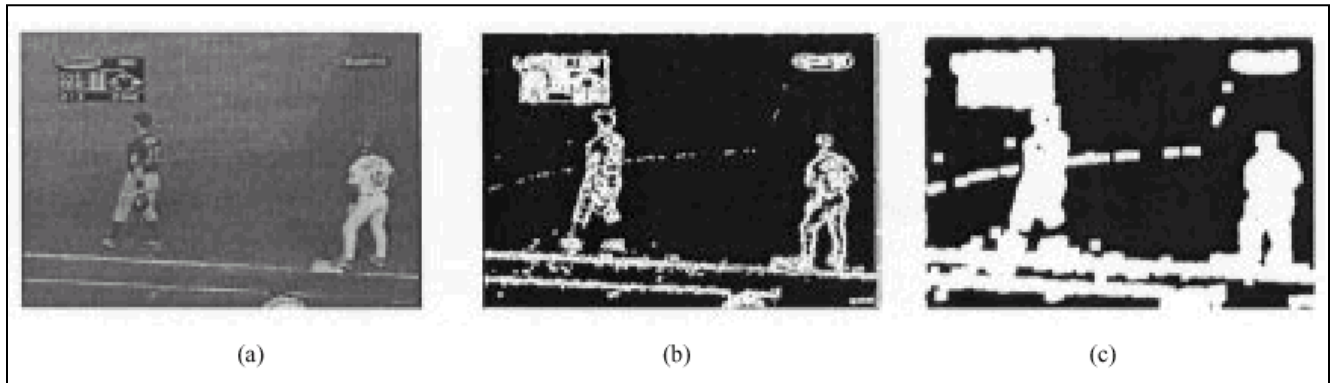
Example

- Color space selection & quantization
 - Use RGB channels
 - Each channel is divided into 2 intervals
 - Total number of bins = $2^3 = 8$
 - H(I): Color histogram for Image I
 - $H_1 = (7, 7, 7, 7, 9, 9, 9, 9)$
 - Image 1 has 7 pixels in each C_1 to C_4 , and 9 pixels in each C_5 to C_8

STATIC SCENE TEST

- **Define**
 - all objects present in the scene exhibit rather small motion compared to the frame size, and global movement caused by the camera is slow and smooth
- **Method**
 - Edge Detection
 - Edge Dilation
- **Result**
 - the transition of two consecutive frames with covering ratio larger than a predefined threshold is considered as a non-static or dynamic scene

Example of edge detection



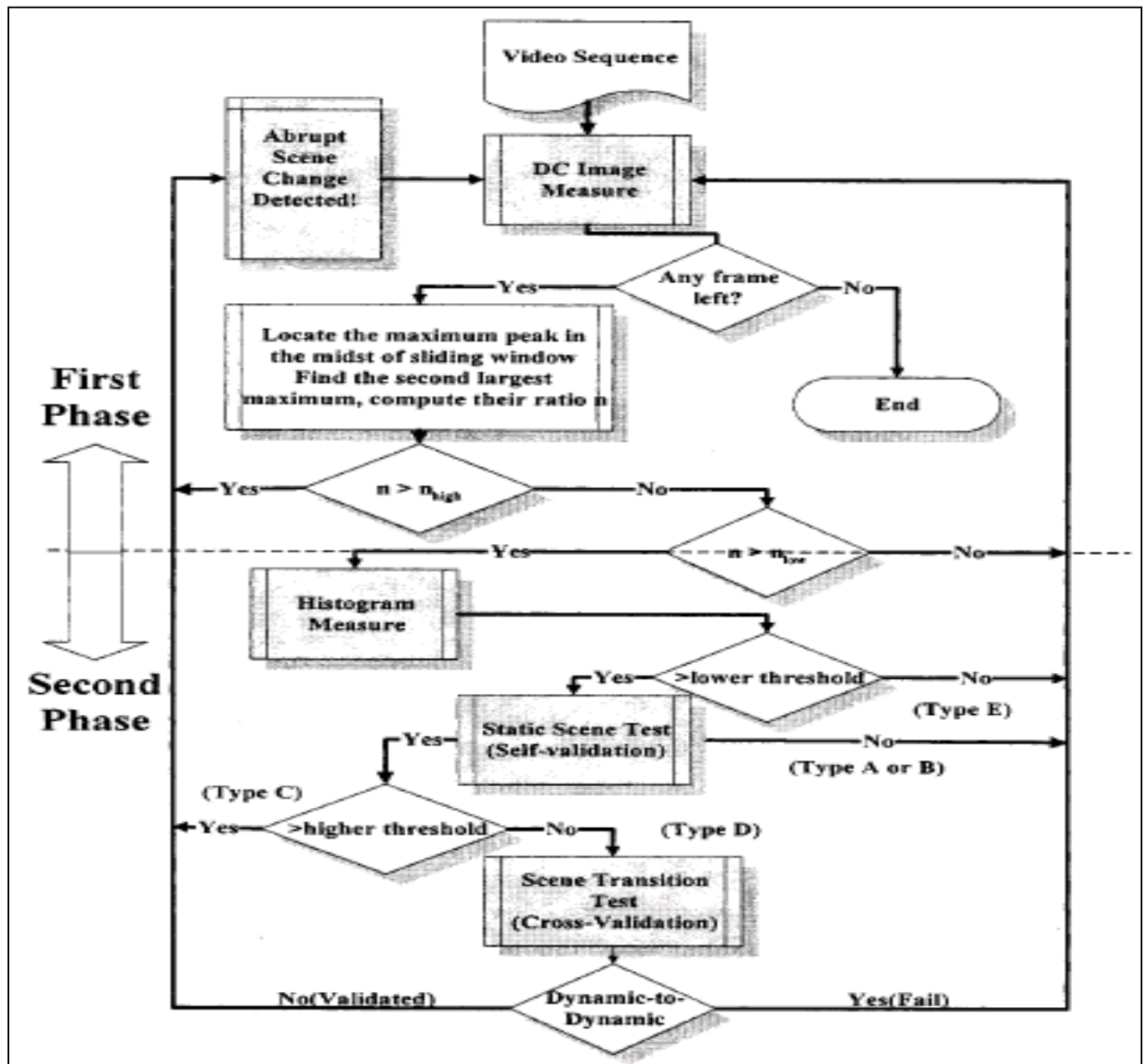
Scene Transition Classification

- Transition Type
 - 1) static scene to static scene
 - 2) dynamic scene to static scene or vice versa
 - 3) dynamic scene to dynamic scene

- **Transition Type**
 - static scene to static scene
 - dynamic scene to static scene or vice versa
 - dynamic scene to dynamic scene

Dynamic-to-dynamic transition usually indicates continuous object or camera motion, rather than a real scene change.

DETECTION ALGORITHM

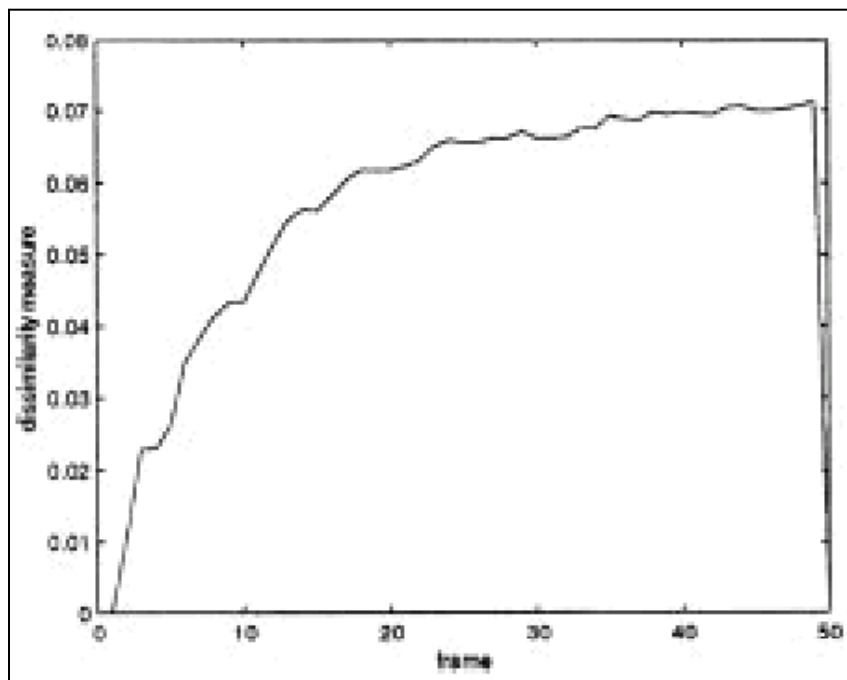


Gradual Scene-Change Detection

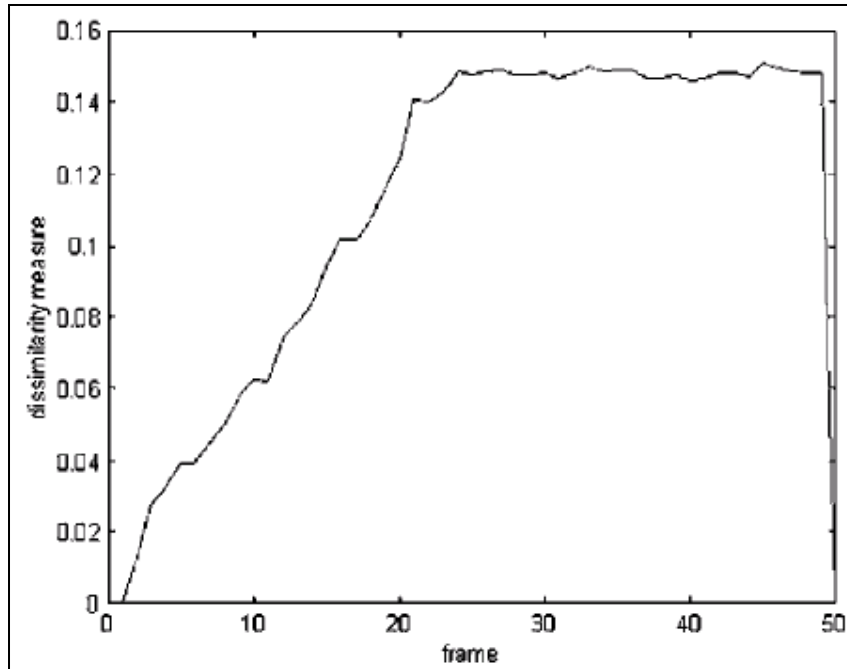
- Gradual Scene-Change
 - Dissolve
 - Fade-In
 - Fade-Out
- Why not easy to detect
 - Camera and object motions always introduce a larger variation than a gradual transition.

Intensity Statistics Model

- Normal Case
 - For any frames near the reference frame, their dissimilarity measure almost increases exponentially with their distance
- Gradual Transition
 - The dissimilarity increases linearly with their distance during the transition
 - After the transition is over, the difference measures are randomly distributed



NORMAL SEQUENCE (ABRUPT)



GRADUAL SEQUENCE

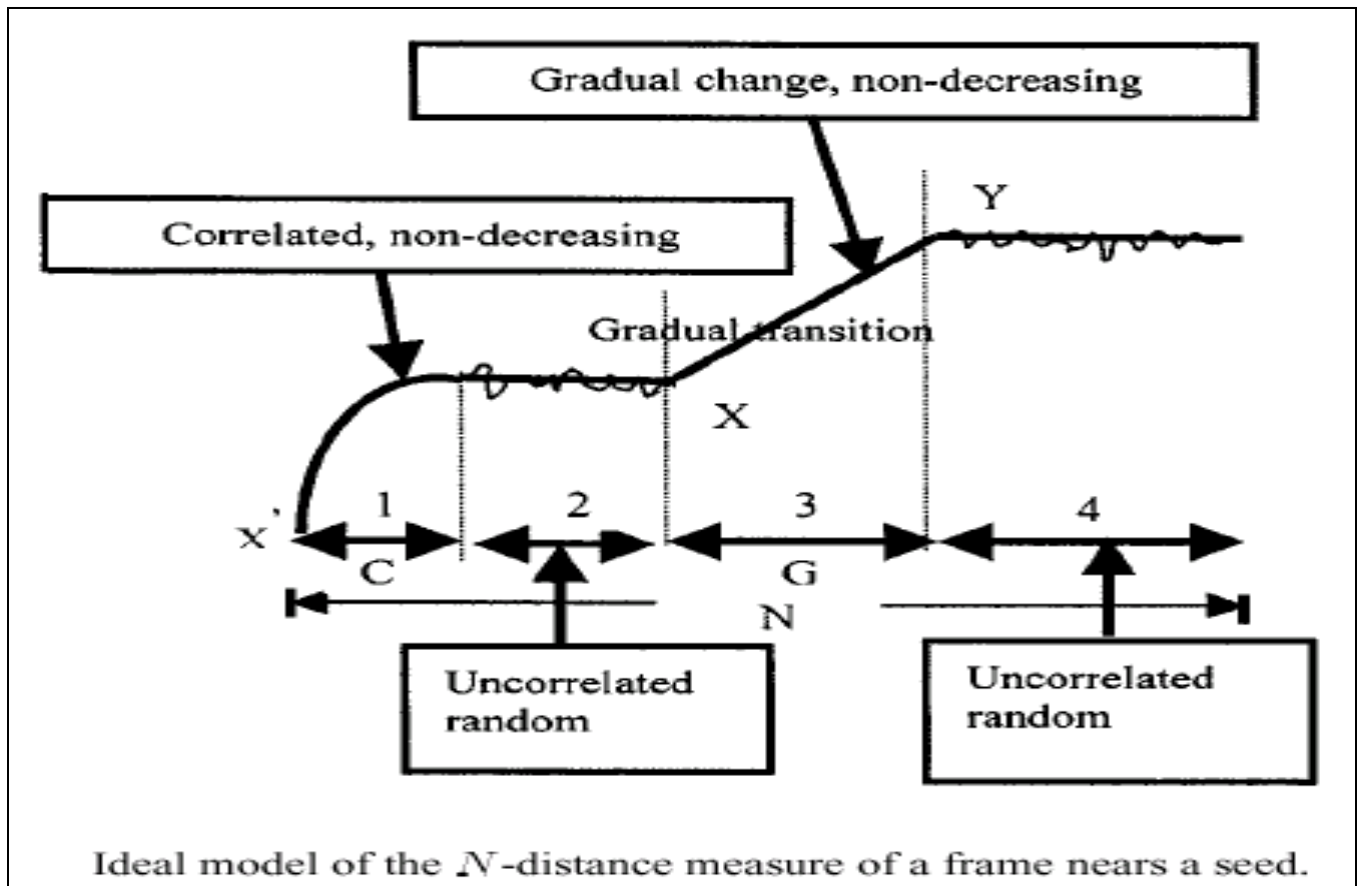
Gradual Scene-Change-Detection Algorithm

N-distance measure

The difference measure generated by comparing a frame with itself and its successive ($N - 1$) frames

$$\underline{M}_{N\text{-distance}} = \{d(X_i, X_j) | j = i, i + 1, \dots, i + N - 1\}$$

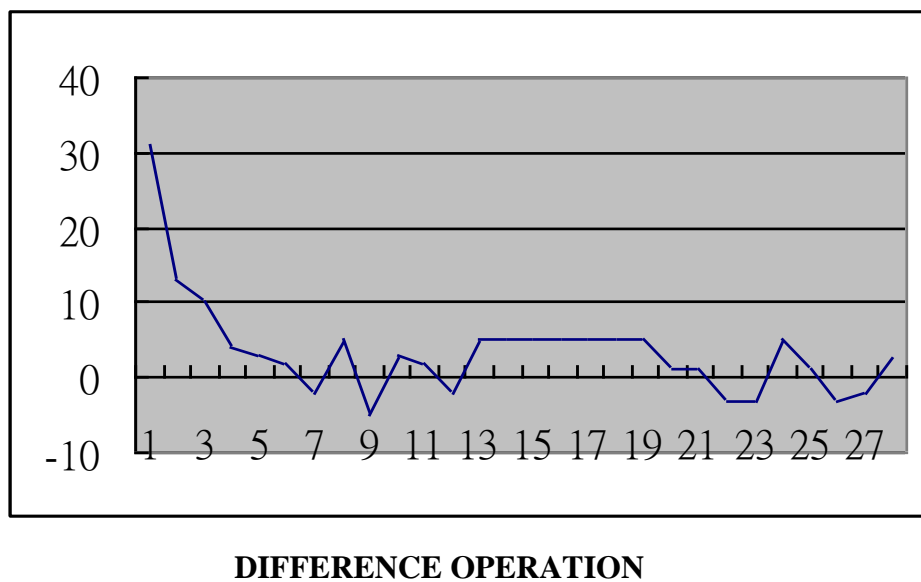
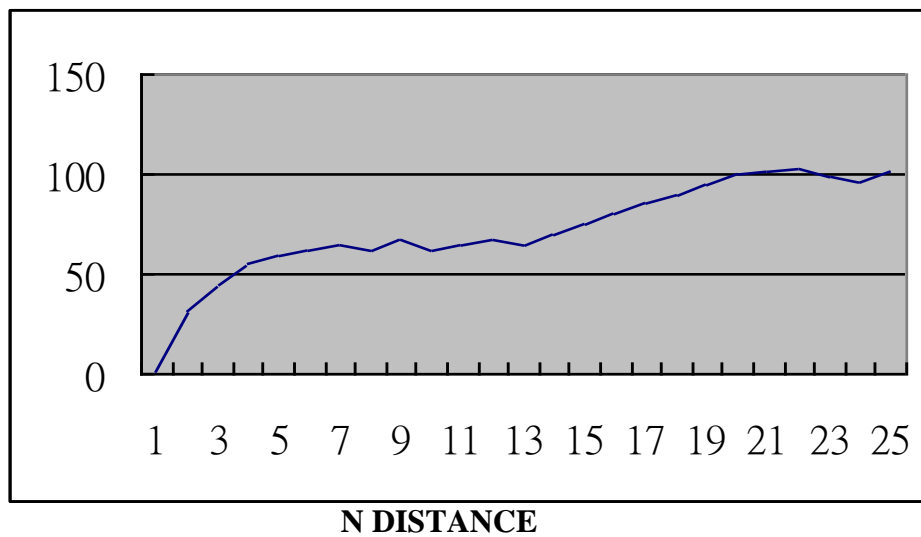
Ideal model of the N-distance measured



Difference operation

$$\begin{aligned} \underline{M}' &= \{d'(X_i, X_j, X_{j+1}) | j = i, i+1, \dots, i+N-2\} \\ &= \{d(X_i, X_{j+1}) \\ &\quad - d(X_i, X_j) | j = i, i+1, \dots, i+N-2\}. \end{aligned}$$

COMPARISION OF N DISTANCE MEASURE N DIFFERENCE OPERATION



Literature survey

Scene change detection techniques for video database systems

Haitao Jiang, Abdelsalam (Sumi) Helal, Ahmed K. Elmagarmid, Anupam Joshi

Department of Computer Science, Purdue University, West Lafayette, IN 47907,USA; e-mail: fjiang,helal,ake,joshig@cs.purdue.edu

Abstract. Scene change detection (SCD) is one of several fundamental problems in the design of a video database management system (VDBMS). It is the first step towards the automatic segmentation, annotation, and indexing of video data. SCD is also used in other aspects of VDBMS, e.g., hierarchical representation and efficient browsing of the video data. In this paper, we provide a taxonomy that classifies existing SCD algorithms into three categories: full videoimage-based, compressed-video-based, and model-based algorithms. The capabilities and limitations of the SCD algorithms are discussed in detail. The paper also proposes a set of criteria for measuring and comparing the performance of various SCD algorithms. We conclude by discussing some important research directions.

Key words: Scene change detection – Video segmentation– Video databases – Survey

1 Introduction

A *video database management system* is a software that manages a collection of video data and provides content-based access to users [10]. There are four basic problems that need to be addressed in a video database management system. These are video data modeling, video data insertion, video data storage organization and management, and video data retrieval. One fundamental aspect that has a great impact on all basic problems is the content-based temporal sampling of video data [24]. The purpose of the content-based temporal sampling is to identify significant video frames to achieve better representation, indexing, storage, and retrieval of the video data. Automatic content-based temporal sampling is very difficult due to the fact that the sampling criteria are not well defined, i.e., whether a video frame is important or not is usually subjective. Moreover, it is usually highly application-dependent and requires high-level, semantic interpretation of the video content. This requires the combination of very sophisticated techniques from computer vision and AI. The state of the art in those fields, however, has not advanced to the point where semantic interpretations would

However, researchers usually can get satisfying results by analyzing the visual content of the video and partitioning it into a set of basic units called *shots*. This process is also referred to as *video data segmentation*. Content-based sampling thus can be approximated by selecting one representing frame from each shot, since a shot is defined as a continuous sequence of video frames which have no significant inter-frame difference in terms of their visual contents.¹ A single shot usually results from a single continuous camera operation. This partitioning is usually achieved by sequentially measuring inter-frame differences and studying their variances, e.g., detecting sharp peaks. This process is often called *scene change detection* (SCD).

Scene change in a video sequence can either be abrupt or gradual. *Abrupt scene changes* result from editing “cuts” (Fig. 1), and detecting them is called *cut detection* [11]. Gradual scene changes result from *chromatic edits*, *spatial edits* and *combined edits* [11]. Gradual scene changes include special effects like zoom, camera pan, dissolve and fade in/out, etc. An example of abrupt scene

change and gradual scene change is shown in Fig. 2. SCD is usually based on some measurements of the image frame, which can be computed from the information contained in the images.

This information can be color, spatial correlation, object shape, motion contained in the video image, or discrete cosine (DC) coefficients in the case of compressed video data. In general, gradual scene changes are more difficult to detect than abrupt scene changes, and may cause lots of scene detection algorithms to fail under certain circumstances. Existing SCD algorithms can be classified in many ways according to, among others, the video features they use and the video objects they can be applied to. In this paper, we discuss SCD algorithms in three main categories:

- (1) approaches that work on uncompressed full-image sequences;
- (2) algorithms that aim at working directly on the compressed video; and
- (3) approaches that are based on explicit



Fig. 1. An example of an abrupt scene change



Fig. 2. An example of a gradual scene change



Fig. 3. An example of a full image and its DC image

models. The latter are also called *top-down approaches* [10], whereas the first two categories are called *bottom-up approaches*. This paper is organized as follows. Section 2 briefly presents some background information about the SCD problem. Then, three categories of existing work are summarized in Sects. 3, 4, and 5, respectively. Their performance, advantages, and drawbacks are also discussed. Section 6 presents some criteria for evaluating the performance of SCD algorithms. Section 7 discusses some possible future research directions.

2 Background

We now introduce some basic notations used in this paper, followed by the notions of DC images, DC sequences and how they can be extracted from compressed video. Several most often used image measurements are also briefly described in terms of their use in measuring the inter-frame difference. It should be noted that they may not work well for scene detection when used separately, thus they usually are combined in the SCD algorithms. For example, Swanberg et al. [28] use a combination of template and histogram matching to measure the video frames.

2.1 Basic notations

Following notations are used throughout this paper. A sequence of video images, whether they are fully uncompressed or spatially reduced, are denoted as I_i ; $0 \leq i < N$, N is the length or the number of frames of the video data. the i th frame. H_i refers to the histogram of the image I_i .

The inter-frame difference between images I_i ; I_j according to some measurement is represented as $d(I_i; I_j)$

2.2 MPEG standard: different frame types

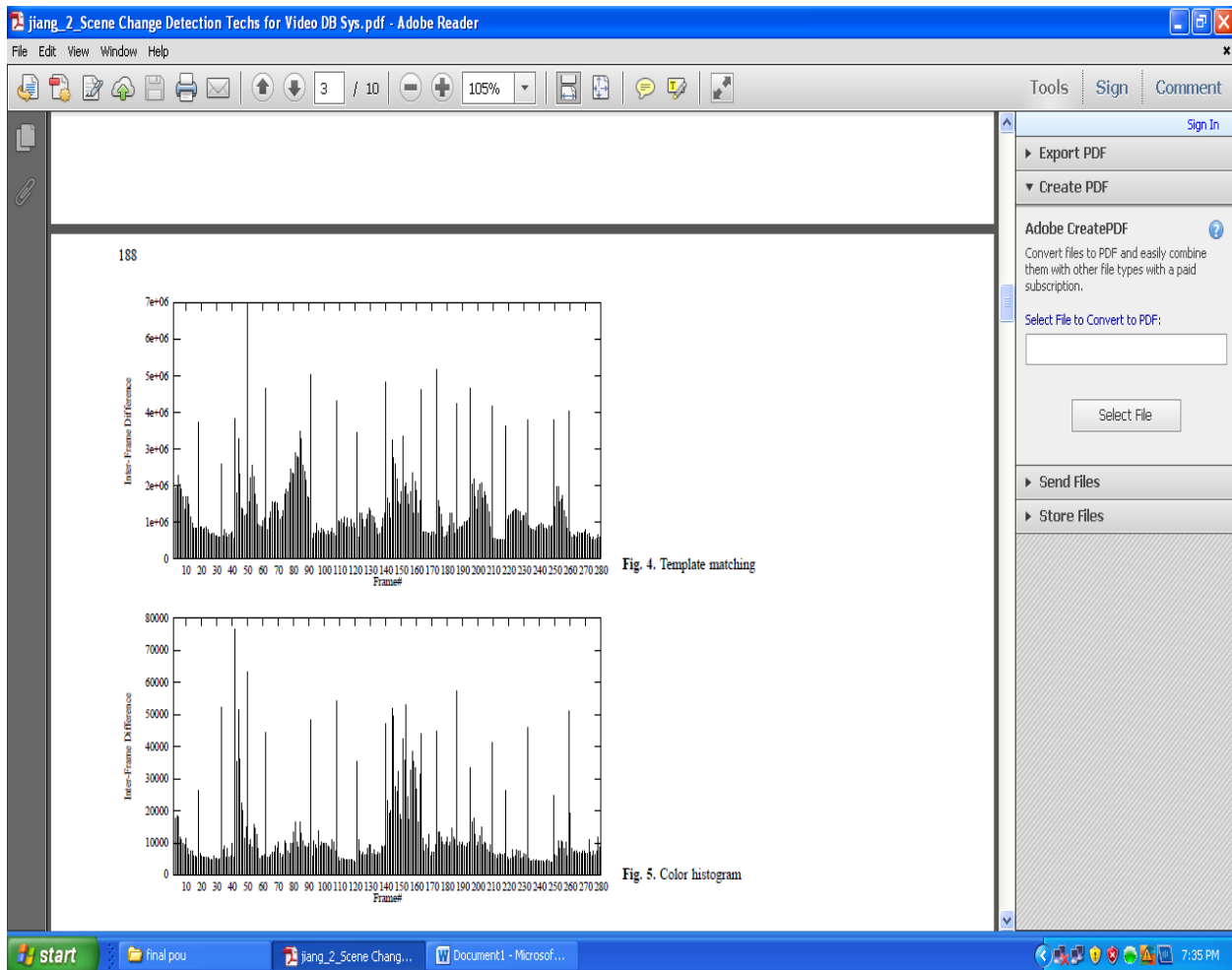
According to the International Standard ISO/IEC 11172 [8], an MPEG-I compressed video stream can have one or more of the following types of frames:

- I (intra-coded) frames are coded without reference to other frames. They are coded using spatial redundancy reduction which is a lossy block-based coding involving DCT, quantization, run length encoding, and entropy coding.
- P (predictive-coded) frames are coded using motion compensation predication from the last I or P frame.
- B (bidirectionally predictive coded) frames are coded using motion compensation with reference to both previous and next I or P frames.
- D (DC-coded) frames are coded using DC coefficients of blocks, thus only contain low-frequency information. D frames are not allowed to co-exist with I/P/B frames and are rarely used in practice.

Obviously, any MPEG compressed video stream must have at least I frames. The data size ratios between frames suggested by the standard are: 3:1 for I:P and 5:2 to 2:1 for P:B. In other words, B frames have the highest degree of compression and I frames have the least one. More details about MPEG video streams can be found in [8].

2.3 DC images, DC sequences and their extraction

A *DC image* [31–34] is a spatially reduced version of a given image. It can be obtained by first dividing the original image into blocks of $n * n$ pixels each, then computing the average value of pixels in each block, which corresponds to one pixel in the DC image. For the compressed video data, e.g., MPEG video, a sequence of DC images can be constructed directly from the compressed video sequence, which is called a *DC sequence*. Figure 3 is an example of a video frame image and its DC image. There are several advantages of using DC images and DC sequences in the SCD for the compressed video.



- DC images retain most of the essential global information for image processing. Thus, lots of analysis done on the full image can be done on its DC image instead.
- DC images are considerably smaller than the full-image frames, which makes the analysis on DC images much more efficient.
- Partial decoding of compressed video saves more computation time than full-frame decompression.

Extracting DC images from an MPEG video stream has been described in Yeo and Liu [31–34]. Extracting a DC image of an I frame is trivial, since it is given by its DCT (discrete cosine transform) coefficients. Extracting DC images from P frames and B frames need to use inter-frame motion information. This may result in many multiplication operations. To speed up the computation, two approximations are proposed: *zero order* and *first order*. The authors claim that the reduced images formed from DC coefficients, whether they are precisely or approximately computed, retain the “global features” which can be used for video data segmentation, SCD, matching and other image analysis.

2.4 Basic measurements of inter-frame difference

2.4.1 Template matching

Template matching is to compare the pixels of two images across the same location and can be formulated as

$$d(I_i; I_j) = \sum_{x=0; y=0}^{x<M; y<N} |I_i(x; y) - I_j(x; y)| \quad (1)$$

where the image size is of $M \times N$. Template matching is very sensitive to noise and object movements, since it is strictly tied to pixel locations. This can cause false SCD and can be overcome to some degree by partitioning the image into several subregions. Figure 4 is an example of inter-frame difference sequence based on template matching. The input video is the one that contains the first image sequence in the Fig. 2.

2.4.2 Color histogram

The color histogram of an image can be computed by dividing a color space, e.g., RGB, into discrete image colors called *bins* and counting the number of pixels falling into each bin [27]. The difference between two images I_i and I_j based on their color histograms H_i and H_j can be formulated as

$$d(I_i; I_j) = \sum_{k=1}^n |H_{ik} - H_{jk}| \quad (2)$$

Which denotes the difference in the number of pixels of two images that fall into same bin. In the RGB color space, the above formula can be written as

$$d_{RGB}(I_i; I_j) = \sum_{k=1}^n [(H_{ri}(k) - H_{rj}(k)) + (H_{gi}(k) - H_{gj}(k)) + (H_{bi}(k) - H_{bj}(k))] \quad (3)$$

Using only simple color histograms may not detect the scene changes very well, since two images can be very different in structure and yet have similar pixel values. Figure 5 is the inter-frame difference sequence of the same video data as in Fig. 2 with color histogram measurement.

2.4.3 X^2 Histogram

The X^2 Histogram computes the distance measure between two image frames as

$$d(I_i; I_j) = \sum_{k=1}^n \left[\frac{(H_i(k) - H_j(k))^2}{H_j(k)} \right] \quad (4)$$

Several researchers [19, 37, 38] have used X^2 histograms in their SCD algorithms, and they report that it generates better results compared to other intensity-based measurements, e.g., color histogram and template matching. Figure 6 is the inter-frame difference sequence of the same video data as in Fig. 2, but computed using a X^2 histogram.

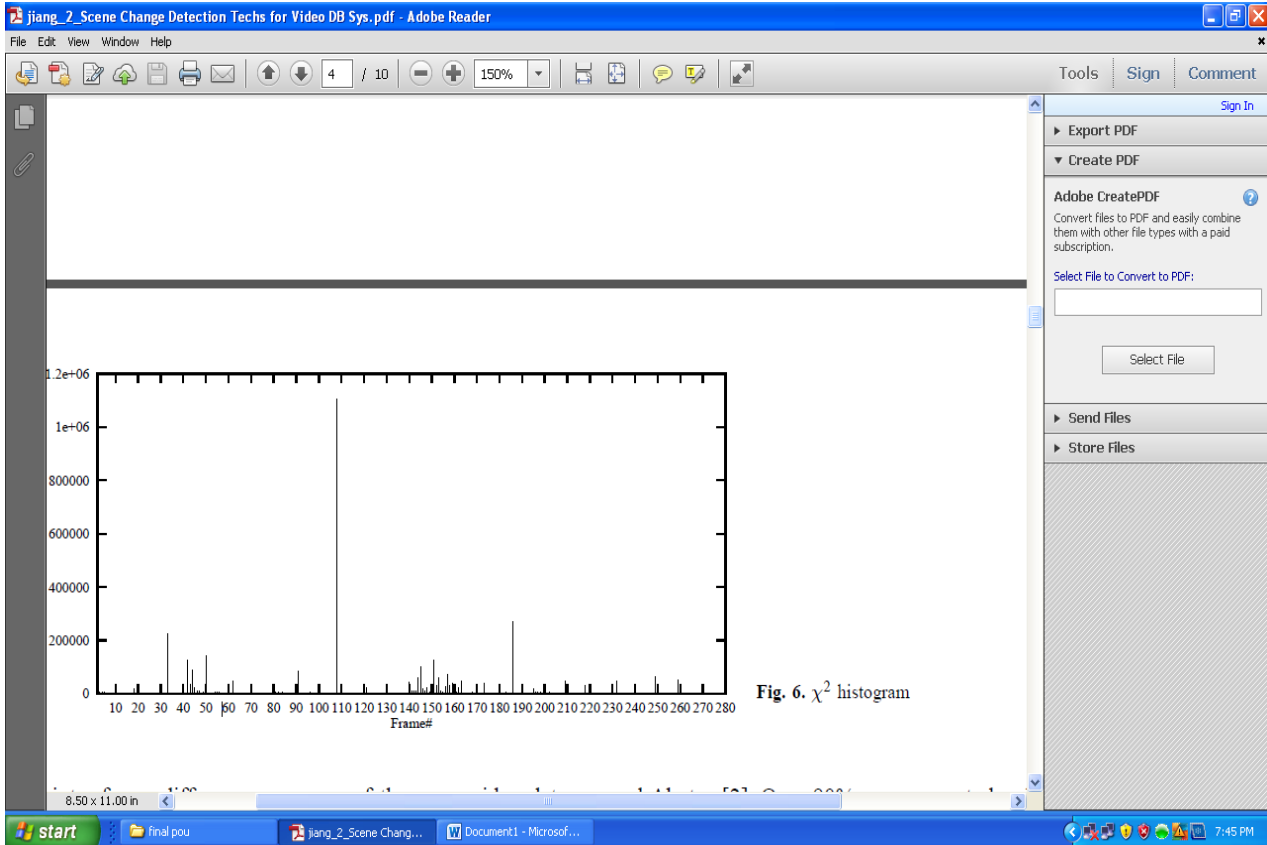


Fig. 6. χ^2 histogram

3 Full-image video SCD

Most of the existing work on SCD is based on full-image video analysis. The differences between the various SCD approaches are the measurement function used, features chosen, and the subdivision of the frame images. Many use the either intensity feature [19–21,30,37,38] or motion information [2, 13, 24] of the video data to compute the inter-frame difference sequence. The problem with intensity-based approaches is that they may fail when there is a peak introduced by an object or camera motion. Motion-based algorithms also have the drawback of being computationally expensive, since they usually need to match the image blocks across the frames. After the inter-frame differences are computed, some approaches use a global threshold to decide a scene change. This is clearly insufficient, since a large global difference does not necessarily imply that there is a scene change as reported, for example, by Yeo [32,34]. In fact, scene changes with globally low peaks constitute one of the situations that often cause the failure of the algorithms. Scene changes, either abrupt or gradual, are localized processes, and should be checked accordingly.

3.1 Detecting abrupt scene changes

Algorithms for detecting abrupt scene changes have been proposed by Nagasaka et al. [19], Hsu [13], Otsuji [20, 21], and Akutsu [2]. Over 90% accuracy rate has been achieved. However, these approaches do not take into account gradual scene changes.

Nagasaka and Tanaka [19] presented an approach that partitions the video frames into 4×4 equal-sized windows and compares the corresponding windows from the two frames. Every pair of

windows are compared and the largest difference is discarded. The difference values left are used to make the final decision. The purpose of the subdivision is to make the algorithm more tolerant to object movement, camera movement, and zooms. Six different type of measurement functions, namely, difference of gray-level sums, template matching, difference of gray-level histograms, color template matching, difference of color histogram, and X^2 comparison of the color histograms have been tested. The experimental results indicate that a combination of image subdivision and X^2 color histogram approach provides the best results of detecting scene changes. The disadvantage of this approach is that it may miss gradual scene transitions such as fading.

Otsuji [20, 21] computed both the histogram and the pixel-bases inter-frame difference based on brightness information to detect scene changes. Projection detection filter is also proposed for more reliable scene detection. The gradual scene changes are not taken into consideration.

Akutsu et al. [2] used both the average inter-frame correlation coefficient and the ratio of velocity to motion in each frame of the video to detect scene change. Their assumptions were that (1) the inter-frame correlation between frames from the same scene should be high, and (2) the ratio of velocity to motion across a cut should be high also. The approach does not address gradual scene changes and is computationally expensive, since computing motion vectors requires the matching of image blocks across frames. Also, how to combine the above two measurements to achieve better result is not clear from the paper.

Hsu et al. [13] treated the scene changes and activities in the video stream as a set of motion discontinuities which change the shape of the spatio-temporal surfaces. The sign of the Gaussian and mean curvature of the spatio-temporal surfaces is used to characterize the activities. Scene changes are detected using an empirically chosen global threshold. Clustering and the split-and-merge approach are then taken to segment the video. The experimental results in the paper are not sufficient to make any judgment on the approach and no comparison results with other existing algorithms are available.

4.1 DC image-sequence-based approach

Yeo and Liu [31, 32, 34] proposed to detect scene changes on the DC image sequence of the compressed video data. They [34] discussed the following measurements: successive pixels difference (template matching) and global color statistic comparison (RGB color histogram). Template matching is sensitive to camera and object motion, and may not produce good result as in the full-frame-image case. However, this measurement is more suitable for DC sequences, because DC sequences are smoothed versions of the corresponding full images, and thus less sensitive to the camera and object movements. Based on comparison experiment results, global color statistic comparison is found to be less sensitive to motion, but more expensive to be computed. Template matching is usually sufficient in most cases and used in their algorithm.

Abrupt scene changes are detected by first computing the inter-frame difference sequence, and then applying a slide window of size m . A scene change is found if

- the difference between two frames is the maximum within a symmetric window of size $2m - 1$;
- the difference is also n times the second largest difference in the window. This criteria is for the purpose of guarding false SCD due to fast panning, zooming, or camera flashing.

The window size m is set to be smaller than the minimum number of frames between any scene change. The selection of parameters n and m relates to the balance of the tradeoff between missed

detection rate and false detection rate, typical values can be $n = 3$ and $m = 10$. The sensitivity of these parameters also experimented with and studied. Gradual scene change may escape the above method.

Gradual scene changes can also be captured by computing and studying the difference of every frame with the previous k th frame, i.e., checking if a “plateau” appears in the difference sequence. They also discuss the detection of flashing-light scenes which may indicate the occurrence of important events or appearance of important person.

Flashing-light scenes can be located by noticing two consecutive sharp peaks in a difference sequence, i.e., in a slide window of the difference sequence:

- the maximum and second largest difference values are very close;
- the two largest difference values are much larger than the average value of the rest.

Detecting scenes with captions is also studied. Their experimental results indicate that over 99% of abrupt changes and 89.5% of gradual changes have been detected and the algorithm is about 70 times faster than that on the full-image sequence. This conforms to the fact that DC images for MPEG video are only 1/ 64 of their original size. Although there may exist the situation that DC images are not sufficient to detect some features [31], this approach is nonetheless very promising and produces best results in the literature.

4.2 DC coefficients-based approach

Arman et al. [7] detected scene changes directly on Motion JPEG compressed video data using the DC coefficients. A frame in the compressed video sequence is represented by a subset of blocks. A subset of the AC coefficients of the $8 * 8$ DCT blocks is chosen to form a vector. It is assumed that the inner product of the vectors from the same scene is small. A global threshold is used for scene changes, and in case there is an uncertainty, a few neighboring frames are selected for further decompression, and color histograms are used on those decompressed frames to find the location of scene change. This approach is computationally efficient. However, it does not address gradual transition like fade and dissolving, and the experimental evaluation of the technique is not very sufficient.

Sethi and Patel [23] used only the DC coefficients of I frames of an MPEG compressed video to detect scene changes based on luminance histogram. The basic idea is that, if two video frames belong to the same scene, their statistical luminance distribution should be derived from a single statistical distribution. If they are not, a scene change can be declared. Their algorithm works as follows first, I frames are extracted from the compressed video streams; second, the luminance histograms of the I frames are generated by using the first DC coefficient. In the third step, the luminance histograms of consecutive frames are compared using one of the three statistical tests (Yakimovsky’s likelihood ratio test, the χ^2 histogram comparison test, or the Kolmogorov-Smirnov test, which compares the cumulative distributions of the two data sets). Different types of video data have been used to test the algorithm, and χ^2 histogram comparison seems to yield better results.

Zhang et al. [39] used DCT blocks and vector information of the MPEG compressed video data to detect scene changes based on a count of nonzero motion vectors. Their observation is that the number of valid motion vectors in P or B frames tend to be low when such frames lie between two different shots. Those frames are then decompressed and full-image analysis is done to detect scene changes. The weakness of this approach is that motion-compensation related.

information tends to be unreliable and unpredictable in the case of gradual transitions, which causes the approach to fail.

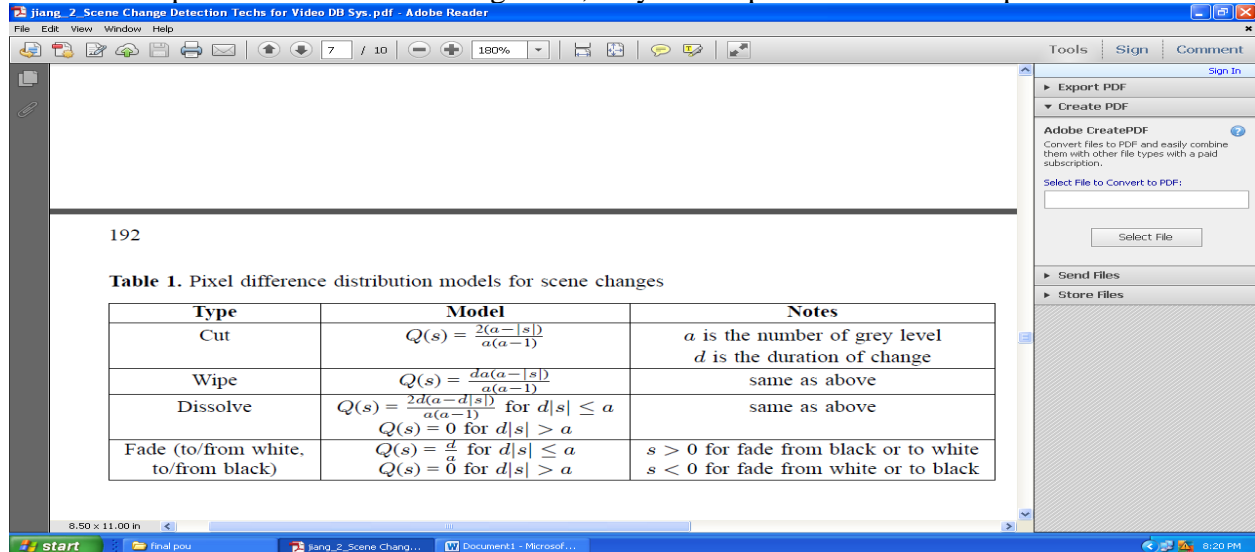
Meng et al. [18] used the variance of DC coefficients in I and P frames and motion vector information to characterize scene changes of MPEG-I and MPEG-II video streams. The basic idea of their approach is that frames tend to have very different motion vector ratios if they belong to different scene, and to have very similar motion vector ratios if they are within the same scene.

The SCD algorithm works as follows. First, an MPEG video is decoded just enough to obtain the motion vectors and DC coefficients, and then inverse motion compensation is applied only to the luminance microblocks of P frames to construct their DC coefficients. Then the suspected frames are marked.

- An I frame is marked if there is a peak inter-frame histogram difference and the immediate B frame before it has a peak value of the ratio between forward and backward motion vectors.
- A P frame is marked if there is a peak in its ratio of intra-coded blocks and forward motion vectors.
- A B is marked if its backward and forward motion vector ratio has a peak value.

Final decisions are made by going through marked frames to check if they satisfy the local window threshold. The threshold is set according to the estimated minimal scene change distance. The dissolve effect is determined by noticing a parabolic variance curve.

As more and more video data are compressed and made available on the Internet and World Wide Web, the above SCD algorithms certainly make good choices in many cases. However, we should notice their limitations. First, current video compression standards like MPEG are optimized for data compression rather than for the representation of the visual content and they are lossy, e.g., they do not necessarily produce accurate motion vectors [36]. Second, motion vectors are not always readily obtainable from the compressed video data, since a large portion of the existing MPEG videos have I frames only [36]. Moreover, some of the important image analyses, e.g., automatic caption extraction and recognition, may not be possible on the compressed data.



5 Model-based video SCD

All the research work introduced so far is solely based on image-processing techniques. It is, however, possible to build an explicit model of the video data to help the SCD process [1, 10, 12].

These algorithms are sometimes referred to as top-down approaches [10, 12], whereas the algorithms in Sects. 3 and 4 are known as bottom-up approaches. The advantage of the model-based SCD is that a systematic procedure based on mathematical models can be developed, and certain domain-specific constraints can be added to improve the effectiveness of the approaches [12]. The performance of such algorithms depends on the models they based on. Hampapur et al. [10, 12] used a *production-model-based classification* for video segmentation. Based on a study of the video production-process and different constraints abstracted from it, a *video edit model* which captures the process of video editing and assembly was proposed. The model includes three components: *edit decision model*, *assembly model*, and *edit effect model*. The edit effect model contains both abrupt scene change (cut) and gradual scene changes (translate, fade, dissolve and morphing, etc.). Templatematching and X^2 histogram measurements are used.

Gradual scene changes (they are called chromatic edits) like fade and dissolve are modeled as chromatic scaling operations. Fade is modeled as a chromatic scaling operation with positive and negative fade rate, and dissolve is modeled as a simultaneous chromatics scaling operations of two images. The first step of their algorithm is to identify the features which correspond to each of the edit classes to be detected, then classify the video frames based on these features. Feature vectors extracted from the video data are used together with the mathematical models to classify the video frames and to detect any edit boundaries. Their approach has been tested using cable TV program video with cut, fade, dissolve and spatial edits, with an overall 88% accurate rate being reported [10]. Aigrain and Joly [1] proposed an algorithm based on a differential model of the distribution of pixel value differences in a motion picture which includes

- a small-amplitude additive zero-centered Gaussian noise which models camera, film and other noises.
- an intra-shot change model for pixel change probability distribution results from object, camera motion, or angle change, focus or light change at a given time and in a given shot.

The first step of their SCD algorithm is to reduce the resolution of frame images by undersampling. Its purpose is to overcome the effects of camera and object motion, as well as to make the computation more efficient in the following steps. Second, compute the histogram of pixel difference values, and count the number of pixels whose change of value is within a certain range determined by studying the above models. Different scene changes are then detected by checking the resulting integer sequence. Experiments show that their algorithm can achieve 94–100% detection rate for abrupt scene changes, and around 80% for gradual scene changes.

6 Evaluation criteria for the performance of SCD algorithms

It is difficult to evaluate and compare existing SCD algorithms due to the lack of objective performance measurements. This is mainly attributed to the diversity of factors involved in the video data. However, various video resources can be used to test and compare algorithms against some user- and application-independent evaluation criteria, and give us indications of their effectiveness. Unfortunately, there is no widely accepted test video data set currently available and many researchers use MPEG movies available on a few WWW archive sites² as inputs for their SCD algorithms.

Such video data may not be a good benchmark to use in testing SCD algorithms. There are several reasons for this. First, the purpose of these movies was not for benchmarking SCD algorithms. So, although some of them may take half an hour to download and may occupy very large disk space (a

1-min MPEG video can easily takes up over 5 MB storage space, depending on the encoding method), they may not be a representative data set for all the possible scene change types. Second, the quality of these movies varies greatly, since they come from different sources and are encoded using various coding algorithms. For example, many MPEG movies have I frames only, which may cause problems for some SCD algorithms on compressed video data. Third, there are no widely accepted “correct” SCD results available for any of those MPEG data sets. Thus, an effort towards building a publicly accessible library of SCD test video data sets will be very useful. Such a test data set should include video data from various applications and different types of scene change, along with the analysis results made and agreed upon by researchers.

We argue that performance measurements of SCD algorithms should include one or more of following:

- CPU time spent for a given video benchmark, e.g., the number of frames processed by an SCD algorithm per time unit.
- average success rate or failure rate for SCD over various video benchmarks. The failure includes both false detection and missed detection, for example, 100% scene change capture rate does not necessarily imply that the algorithm is good, since it may contain very many false change alarms. The results of an SCD algorithm can be compared to human SCD results which can be assumed to be correct.
- SCD granularity. Can the algorithm decide between which frames a scene change occurs? Furthermore, can it also report the type of the scene change, i.e., whether it is fade in or dissolve?
- stability, i.e., its sensitivity to the noise in the video stream. Very often, flashing of the scene and background noises can trigger false detection.
- types of the scene changes and special effects that it can handle.
- generality. Can it be applied to various applications? i.e., which are the different kinds of video data resources it can handle?
- formats of the video it can accept (full-image sequence, MPEG-I, MPEG-II, or AVI video, etc.).

7 Conclusion

In this paper, a taxonomy of existing SCD techniques for the video database system is presented and discussed. Criteria of benchmarking SCD algorithms are also proposed. Existing SCD algorithms have achieved success rates for abrupt scene changes of above 90%, and above 80% for gradual scene changes. These numbers are, in general, fairly acceptable in certain applications. However, there is an obvious need for further improvement.

There are several possible ways to achieve this.

1. Use additional visual, as well as audio information, rather than relying only on the color or intensity information most existing algorithms rely on. Other visual information includes captions, motion of the objects and camera, and object shapes. The problem of how to use audio signals and other information contained in the video data in SCD and video segmentation has not been carefully addressed in the literature so far, although some initial efforts [25,26] have been made for video skimming and browsing support.
2. Develop adaptive SCD algorithms which can combine several SCD techniques and can self-adjust to various parameters. They would choose best criteria optimized for a different given video

data, i.e., a video sequence with frequent object movements (action movies) vs one with very few motions (lecture video).

3. Use a combination of various scene change models. Developing scene change models can be a very difficult task due to the complicated nature of video production and editing. However, different aspects of the video editing and production process can be individually modeled and used in developing detectors for certain scene changes.

Another idea is to develop new video-coding and decoding schemes that include more information about scene content. As pointed out by Bove [3], current motion-compensated video codec standards like MPEG complicate the scene analysis task by partitioning the scene into arbitrary tiles, resulting in a compressed bitstream which is not physically or semantically related to the scene structure. For a complete solution to the problem, however, a better understanding of the human capabilities and techniques for SCD is needed. This would involve using information available from psychophysics [5, 6, 14, 22], and also understanding the neural circuitry of the visual pathway [16]. Techniques developed in computer vision for detecting motion or objects [4,5,7,29] can also be incorporated into SCD algorithms.

Analysis and Requirement Specification

HARDWARE SPECIFICATION :

- PROCESSOR: PENTUIUM IV 2.6 GHz
- RAM:512MB DD RAM
- MONITOR: 15” COLOR
- HARD DISK :250 GB
- CDDRIVE: LG52X
- KEYBOARD: STANDARD 102 KEYS
- MOUSE: OPTICAL MOUSE

SOFTWARE SPECIFICATION:

- FORNT END: MATLAB
- BACKEND: FFMPEG SOFTWARE
- OPERATING SYSTEM: WINDOWS XP/WINDOWS 7

MATLAB (matrix laboratory)

It is a numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

In 2004, MATLAB had around one million users across industry and academia.^[2] MATLAB users come from various backgrounds of engineering, science, and economics. MATLAB is widely used in academic and research institutions as well as industrial enterprises.

History

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s. He designed it to give his students access to LINPACK and EISPACK without them having to learn Fortran. It soon spread to other universities and found a strong audience within the applied mathematics community. Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and founded MathWorks in 1984 to continue its development. These rewritten libraries were known as JACKPAC. In 2000, MATLAB was rewritten to use a newer set of libraries for matrix manipulation, LAPACK.

MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra and numerical analysis, and is popular amongst scientists involved in image processing.

Syntax

The MATLAB application is built around the MATLAB language, and most use of MATLAB involves typing MATLAB code into the Command Window (as an interactive mathematical shell), or executing text files containing MATLAB codes, including scripts and/or functions.

Variables

Variables are defined using the assignment operator, =. MATLAB is a weakly typed programming language because types are implicitly converted. It is a dynamically typed language because variables can be assigned without declaring their type, except if they are to be treated as symbolic objects, and that their type can change. Values can come from constants, from computation involving values of other variables, or from the output of a function. For example:

```

>> x = 17
x =
    17
>> x = 'hat'
x =
    hat
>> y = x + 0
y =
    104     97    116
>> x = [3*4, pi/2]
x =
    12.0000    1.5708
>> y = 3*sin(x)
y =
   -1.6097    3.0000

```

FFMPEG

FFmpeg is a free software project that produces libraries and programs for handling multimedia data. FFmpeg includes **libavcodec**, an audio/video codec library used by several other projects, **libavformat**, an audio/video container mux and demux library, and the **ffmpeg** command line program for transcoding multimedia files. FFmpeg is published under the GNU Lesser General Public License 2.1+ or GNU General Public License 2+ (depending on which options are enabled).

Components

The project comprises several components:

- *ffmpeg* is a command-line tool to convert one video file format to another. It can also grab and encode in real-time from a TV card.
- *ffserver* is an HTTP and RTSP multimedia streaming server for live broadcasts. It can also time shift live broadcast.
- *ffplay* is a simple media player based on SDL and on the FFmpeg libraries.
- *ffprobe* is a command-line tool to show media information.
- *libswresample* is a library containing audio resampling routines.
- *libavresample* is a library containing audio resampling routines from the Libav project, completely different from *libswresample* from *ffmpeg*.
- *libavcodec* is a library containing all the FFmpeg audio/video encoders and decoders. Most codecs were developed from scratch to ensure best performance and high code reusability.
- *libavformat* is a library containing demuxers and muxers for audio/video container formats.

- *libavutil* is a helper library containing routines common to different parts of FFmpeg. This library includes Adler-32, CRC, MD5, RIPEMD, SHA-1, SHA-2, LZO decompressor, Base64 encoder/decoder, DES encrypter/decrypter, RC4 encrypter/decrypter and AES encrypter/decrypter.
- *libpostproc* is a library containing video postprocessing routines.
- *libswscale* is a library containing video image scaling and colorspace/pixelformat conversion routines.
- *libavfilter* is the substitute for vhook which allows the video/audio to be modified or examined between the decoder and the encoder.

Codecs, formats and protocols supported

Codecs

For more details on this topic, see libavcodec.

Codecs which originated from within the FFmpeg project:

- Snow
- FFV1

The FFmpeg developers have implemented among others:

- ITU-T video standards: H.261,^[20] H.262/MPEG-2 Part 2, H.263,^[20] H.264/MPEG-4 AVC^[20] and HEVC (H.265)^[19]
- ITU-T vocoder standards: G.711 μ -law, G.711 A-law, G.721 (aka G.726 32k), G.722, G.722.2 (aka AMR-WB), G.723 (aka G.726 24k and 40k), G.723.1, G.726, G.729 and G.729D
- ISO/IEC MPEG video standards: MPEG-1 Part 2, H.262/MPEG-2 Part 2, MPEG-4 Part 2, H.264/MPEG-4 AVC and HEVC (MPEG-H Part 2)
- ISO/IEC MPEG audio standards: MP1, MP2, MP3, AAC, HE-AAC and MPEG-4 ALS
- ISO/IEC/ITU-T JPEG image standards: JPEG, JPEG-LS and JPEG 2000
- SMPTE video standards: VC-1 (aka WMV3), VC-2 (aka Dirac), VC-3 (aka AVID DNxHD) and DPX image
- SMPTE audio standards: SMPTE 302M
- DVD Forum standards related / Dolby audio codecs: MLP (aka TrueHD) and AC-3
- 3GPP vocoder standards: AMR-NB, AMR-WB (aka G.722.2)
- GSM- and CDMA-related voice codecs: Full Rate, Enhanced Variable Rate Codec
- Windows Media Player related video codecs: Microsoft RLE, Microsoft Video 1, Cinepak, Indeo 2, 3 and 5,^[20] Motion JPEG, Microsoft MPEG-4 v1, v2 and v3, WMV1, WMV2, WMV3 (aka VC-1) and several screen capture codecs.
- Windows Media Player related audio codecs: WMA1, WMA2, WMA Pro and WMA Lossless
- Windows Media Player related voice codecs: WMA Voice and MS-GSM
- RealPlayer-related video codecs: RealVideo 1, 2, 3 and 4
- RealPlayer-related audio codecs: RealAudio 3, 6, 7, 8, 9 and 10

- RealPlayer-related voice codecs: RealAudio 1, 2 (variant of G.728), 4 and 5
- QuickTime-related video codecs: Cinepak, Motion JPEG, ProRes, Sorenson 3 Codec, Animation codec, Graphics codec, Video codec
- QuickTime-related audio codecs: QDesign Music Codec 2 and ALAC
- Adobe Flash Player related video codecs: Screen video, Screen video 2, Sorenson 3 Codec, VP6 and Flash Video (FLV)
- Adobe Flash Player related audio codecs: Adobe SWF ADPCM and Nellymoser Asao
- Xiph.Org: Theora, Speex (via libspeex), Vorbis, Opus (via libopus) and FLAC
- Sony: ATRAC1 and ATRAC3^[20]
- NTT: TwinVQ
- On2: Duck TrueMotion 1, Duck TrueMotion 2, VP3, VP5,^[20] VP6^[20] and VP8
- Google: VP9^[18]
- RAD Game Tools: Smacker video and Bink video
- Truespeech
- RenderWare: TXD

The default MPEG-4 codec used by FFmpeg for encoding has the FourCC of **FMP4**.

Container formats

- ASF
- AVI and also input from AviSynth
- BFI
- CAF
- FLV
- GXF, General eXchange Format, SMPTE 360M
- IFF
- RL2
- ISO base media file format (including QuickTime, 3GP and MP4)
- Matroska (including WebM)
- Maxis XA
- MPEG program stream
- MPEG transport stream (including AVCHD)
- MXF, Material eXchange Format, SMPTE 377M

Protocols

- IETF standards:
 - FTP
 - TCP
 - UDP
 - Gopher
 - HTTP
 - RTP

- RTSP
 - SDP
 - SFTP (via libssh)
- Apple-related protocols: HTTP Live Streaming
- RealMedia-related protocols: RealMedia RTSP/RDT
- Adobe-related protocols: RTMP, RTMPT (via librtmp), RTMPE (via librtmp), RTMPTE (via librtmp) and RTMPS (via librtmp).

Planning

In our project we have defragmented the video that we have used to detect the cut using FFMPEG software. We have used the code `ffmpeg -i CN.mp4 (name of the video) img%d.jpg`.

We got 3680 frames after defragmenting the video. Next we have detected what type of scene change occurs between two consecutive frames.

There are mainly two types of cut detection:

- Abrupt
- Gradual

When the difference between two consecutive frames is very low then we have gradual scene change detection. Otherwise we have abrupt scene change detection.

The difference between two consecutive frames can be calculated using RGB or Grayscale. But in our project we have used Grayscale because it is simple and easy to implement as compared to RGB.

After that we have plotted the correlation matrix in the graph. Then by putting different values at different times we have found out the slope of the graph and detected whether it is abrupt or gradual.

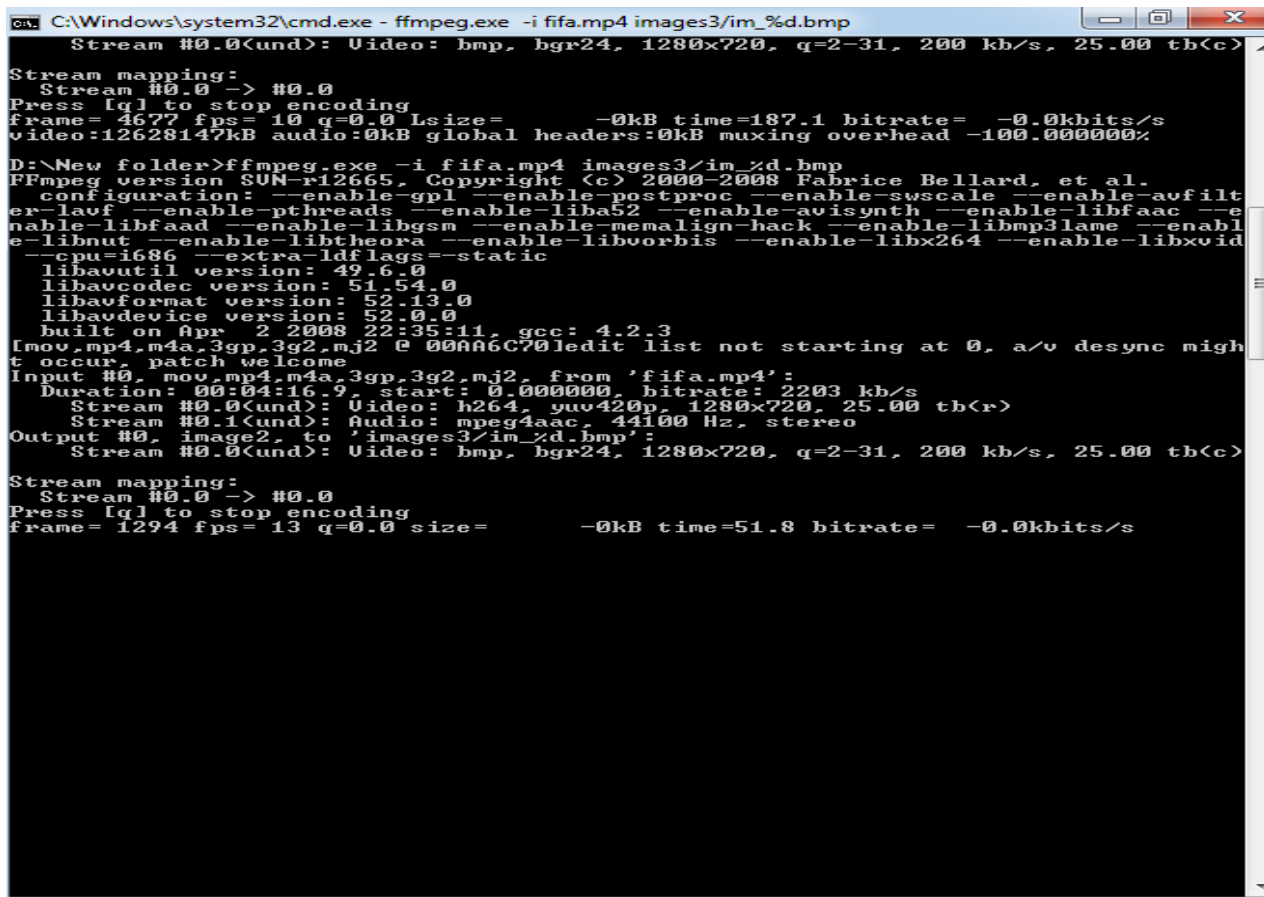
Design

Algorithm:

1. Defragment the video using Ffmpeg software.
2. The video will get defragmented into frames.
3. Convert the image frames into gray scale from rgb scale
4. Compute the inter frame difference
5. Find the correlation Matrix
6. Plot the graph by putting suitable frame numbers

Results analysis and Discussion

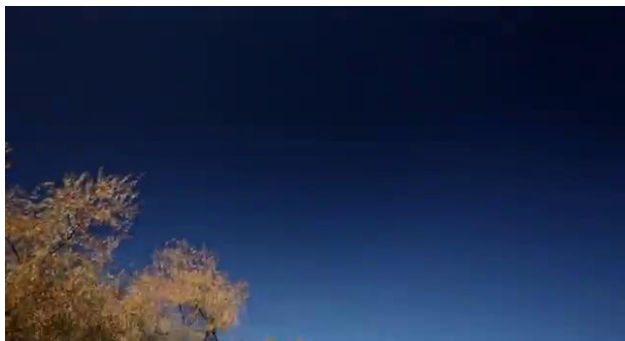
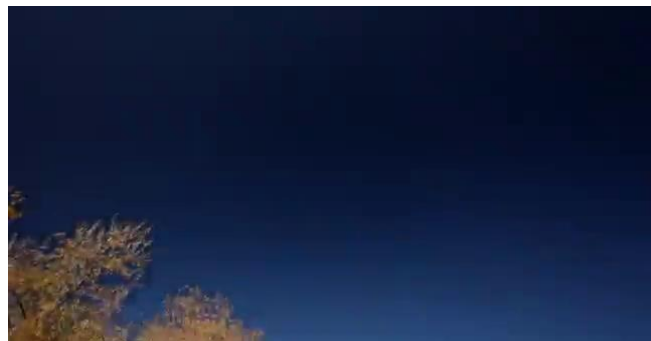
Fragmentation of videos:



```
C:\Windows\system32\cmd.exe - ffmpeg.exe -i fifa.mp4 images3/im_%d.bmp
Stream #0.0<und>: Video: bmp, bgr24, 1280x720, q=2-31, 200 kb/s, 25.00 tb(c)
Stream mapping:
  Stream #0.0 -> #0.0
Press [q] to stop encoding
frame= 4677 fps= 10 q=0.0 Lsize=          -0kB time=187.1 bitrate=  -0.0kbits/s
video:12628147kB audio:0kB global headers:0kB muxing overhead -100.000000%

D:\New folder>ffmpeg.exe -i fifa.mp4 images3/im_%d.bmp
FFmpeg version SUN-r12665, Copyright (c) 2000-2008 Fabrice Bellard, et al.
configuration: --enable-gpl --enable-postproc --enable-swscale --enable-avfilt
er-lavf --enable-pthreads --enable-liba52 --enable-avisynth --enable-libfaac --e
nable-libfaad --enable-libgsm --enable-memalign-hack --enable-libmp3lame --enabl
e-libnut --enable-libtheora --enable-libvorbis --enable-libx264 --enable-libxvid
--cpu=i686 --extra-ldflags=-static
libavutil version: 49.6.0
libavcodec version: 51.54.0
libavformat version: 52.13.0
libavdevice version: 52.0.0
built on Apr  2 2008 22:35:11, gcc: 4.2.3
[moov,mp4,m4a,3gp,3g2,mj2 @ 00AA6C70]edit list not starting at 0, a/v desync migh
t occur, patch welcome
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'fifa.mp4':
  Duration: 00:04:16.9, start: 0.000000, bitrate: 2203 kb/s
  Stream #0.0<und>: Video: h264, yuv420p, 1280x720, 25.00 tb(r)
  Stream #0.1<und>: Audio: mpeg4aac, 44100 Hz, stereo
Output #0, image2, to 'images3/im_%d.bmp':
  Stream #0.0<und>: Video: bmp, bgr24, 1280x720, q=2-31, 200 kb/s, 25.00 tb(c)
Stream mapping:
  Stream #0.0 -> #0.0
Press [q] to stop encoding
frame= 1294 fps= 13 q=0.0 size=          -0kB time=51.8 bitrate=  -0.0kbits/s
```

Fragmented videos:

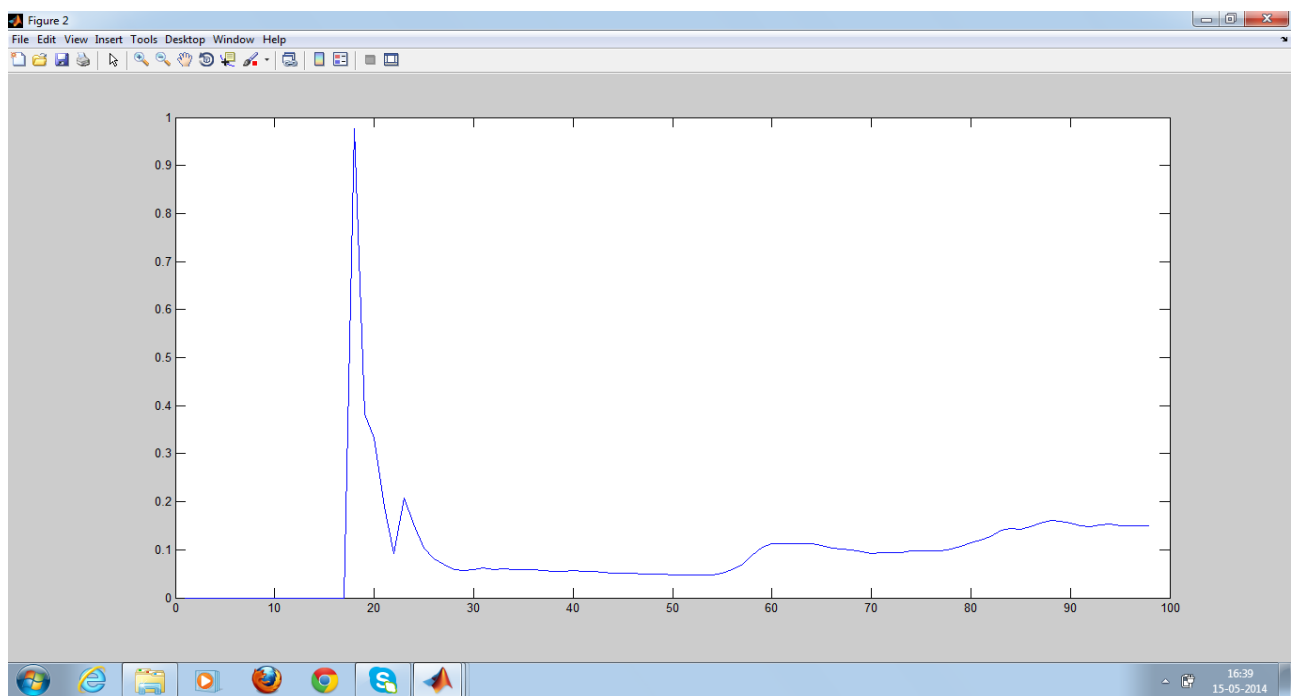


Correlation matrix:



Plotting of graph:

Graph plotted for frames 1,1000



Matlab Coding

```
function cor=cor(str, name, st, en)
corr_arr = [ ];

for num_images = st:en-1
    img_path1 = [str,'\', name, num2str(num_images)+1, '.bmp'];
    img_path2 = [str,'\', name, num2str(num_images), '.bmp'];
    img_arr1 = imread(img_path1);
    img_arr2 = imread(img_path2);
    img_arr1_gray = rgb2gray(img_arr1);
    img_arr2_gray = rgb2gray(img_arr2);
    corr_arr = [corr_arr, cor2(img_arr1_gray, img_arr2_gray, img_arr2_gray)];
end

diff_arr [];
for i=2:size(corr_arr,2)
    diff=corr_arr(1) - corr_arr(i);
    diff_arr = [diff_arr,diff];
end

corr_arr
corr_arr(1)
plot(corr_arr);
diff_arr
figure;
plot(diff_arr);

end
```

Conclusion and Future Scope

In this project hard cut (abrupt) detection of video has been done by computing the inter frame difference and by finding the correlation matrix. In future this detection can also be done using various methods like adaptive dynamic thresholding, based shot boundary detection, destination based cut detection in wireless networks, etc.

Gradual transitions between shots will also be considered and hence can detect various gradual sub cuts like dissolve, fade in, fade out, wipes.

References & Bibliography

http://www.gmr.v.es/~orobles/artics/viip04_robles.pdf

<http://faculty.cs.tamu.edu/stoleru/papers/won11destination.pdf>

<http://www.waset.org/journals/waset/v18/v18-131.pdf>

http://www.irnetexplore.ac.in/IRNetExplore_Volumes/UARJ/UARJ_doc/Volume%201%20Issue%202/paper12.pdf

1. Aigrain P, Joly P (1994) Automatic Real-time Analysis of Film Editing and Transition Effects and its Applications. *Comput Graphics* 18(1):93–103
2. Akutsu A, Tonomura Y, Hashimoto H, Ohbak Y (1992) Video Indexing Using Motion Vectors. In: Maragos P (ed) *Proc of SPIE: Visual Communication and Image Processing 92*. SPIE, Bellingham, WA, USA, pp 1522–1530
3. Bove VM (1996) Multimedia Based on Object Models: Some Whys and Hows. *IBM System Journal* 35(3/4):337–348
4. Cedras C, Shah M (1995) Motion-based Recognition: A Survey. *Image Vision Comput* 13(2):129–155
5. Chellappa R, Wilson CL, Sirohey S (1995) Human and Machine Recognition of Faces: A Survey, *Proc IEEE* 83(5):705–740
6. Dawson MRW (1991) The How and Why of What Went Where in Apparent Motion. *Psychol Rev* 98:569–603
7. Arman F, Hsu A, Chiu M-Y (1993) Image Processing on Compressed Data for Large Video Database In: Rangan P (ed) *Proc ACM Multimedia*, Calif., June 1993. ACM, New York, pp 267–272
8. International Organization for Standardization ISO/IEC 11172 (MPEG)
9. Le Gall D (1991) MPEG: A Video Compression Standard for Multimedia Applications. *Commun ACM* 34(4):46–58