A Statistical Approach to Robust Video Temporal Segmentation

Rajeev Kumar
Computer Science & Engineering Department
Indian Institute of Technology
Kharagpur - 721 302, India
rkumar@cse.iitkgp.ernet.in

Vijay Devatha
Computer Science Department
University of Illinois at Urbana-Champaign
1304 Springfield Ave., Champaign, IL - 61820
devatha@uiuc.edu

Abstract

The problem of video 'cut' detection remains largely an open problem, because of the wide nature of transitions that occur in a digital video. This paper describes a shot boundary detection technique which is an amalgamation of few statistical methods and measures, and robustly detects camera breaks in a full-motion video clip. The proposed algorithm incorporates a weighted histogram, an error-propagation technique for increased robustness, and a curve-fitting technique to extract partitions from the similarity curve for avoiding heuristically chosen threshold value. The algorithm has been validated on many video clips and is shown to give improved results.

1. Introduction

Advances in multimedia technology coupled with explosive growth of internet and the availability of high computing resources at affordable cost have led to the wide-spread use of digital video for varied applications. Digital video has a vast number of advantages over other media that as yet lie unexploited. Video technology, advanced as it may be in the present day, still falls short of placing visual information at the same level of accessibility as certain other media, such as text. Text documents are frequently used to develop ideas, and are often translated into individual compositions. Similar composition using videos, however, remains far in the future. Video composition, as desired should not entail thinking about video pixels any more than text composition entails thinking about ASCII character codes. Thus the effective use of video is beyond our grasp because the effective use of its content is not yet done.

Content based video indexing would be one step towards ensuring that video could be accessed, manipulated, edited and stored as easily or as efficiently as any other medium, [1] and [4]. The obvious problem with retrieval by content is performance. Complex predicates would prove to be extremely expensive to evaluate. The major concern of multimedia research groups, hence, has been to provide access structures and mechanisms that reduce this cost.

The first stage in video indexing is *temporal video segmentation*, also known as *shot detection* or *video partitioning*. This problem can be stated as follows: given a video clip, to define, identify and segregate the elemental units or quanta of the video clip that could be independently accessed, stored and modified. Such elemental units called *shots* would be representative of the entire video clip, and access to them would entail access to the clip in its entirety. The shot-level organization of video documents is considered most appropriate for video browsing and content-based retrieval. A shot or a take, in video parlance, simply refers to a contiguous sequence of one or more video frames depicting a continuous action that remains more or less restricted to a certain region in space and time, [7] and [9].

However, shot boundaries are vague, there exists gradual transitions, illumination changes and/or camera movement within a shot. Video transition can be a *hard-cut*; a gradual transition can be a *fade*, *wipe*, *slide* or a *dissolve* [14]. In recent years, the research on automatic shot detection has exploded, applications are increasing and many algorithms have been published to solve shot detection problem for varying degrees of complexity of real-data. Many performance studies have been carried out and the comparative results published for different classes of video data, e.g., [3], [8] and [16]. All these algorithms have their own merits and demerits - see the recent survey articles by Koprinska & Carrato [11] and Lienhart [14]. An open challenge to almost all such algorithms is to capture *true* transitions and minimize *false* positives/detection in the presence of *unknown* variations. Many algorithms use *heuristically* chosen parameters/procedures which are most effective to a particular class of video using some domain-specific knowledge.

In this paper, we present a new shot detection algorithm which is essentially an amalgamation of some known statistical methods and measures, and robustly detects camera-breaks in a full-motion real video clip. The remainder of the paper is organized as follows. In section 2, we briefly review the existing approaches. We describe, in section 3, our approach which does temporal segmentation with minimal number of user-defined parameters. Then we present

results in section 4 along with a discussion. Finally, we draw conclusions in section 5.

2. Related Work

A large number of shot detection algorithms have been developed, they can be classified based on few core concepts. Essentially, video segmentation is a two-phase phase. Discontinuity or the similarity between two consecutive frames is measured followed by a classifier stage to detect the transition based on some decision strategy. Most important is the underlying detection scheme. Many metrics and the classification algorithms have been proposed in the literature during the past decade, e.g., starting with the initial work of Zhang et al. [22] and Hampapur et al. [10] to some recent work of Gargi et al. [8] and Zhang et al. [21]. In the following paragraphs of this section, we shall briefly review the metrics used and the classification strategy adopted, in general, and build the problem definition for the our work presented in this paper.

Most of the video segmentation algorithms can be divided into two broad categories: pixel-based algorithms and feature-based. Pixel-based techniques are the most rudimentary techniques, and merely use *Inter-frame Difference* (ID) counted in terms of the pixels as the discontinuity measure. The Inter-frame Difference may be simply a count of all the pixels that change, between two successive image frames in the sequence, in value more than some threshold value (Eq. 1). Alternatively, ID is the sum of the absolute difference, in intensity values, of all the pixels between two consecutive frames in a sequence (Eq. 2). The Inter-frame Difference (*ID*) measure would be calculated as:

$$ID_t^{(1)} = \sum_{\forall\, x,y \,\in\, S} 1,\ if\ D_t\left(x,y\right)\ \geq\ I_{threshold} \qquad (1)$$

or

$$ID_t^{(2)} = \sum_{\forall\, x,y \,\in\, S} D_t\left(x,y\right) \qquad (2)$$

where

$$D_t\left(x,y\right)\ =\ \left|I_t\left(x,y\right) - I_{t-1}\left(x,y\right)\right|,\ \forall\ x,y\ \in\ F,$$
$$I_t\left(x,y\right)\ -\ pixel\ value\ at\ (x,y)\ in\ frame\ F_t$$

Pixel-based inter-frame difference methods are the simplest, and have been expanded by breaking images into regions and comparing the statistical measures of the pixels in the respective regions. Since *fades* are produced by linear scaling of the pixel intensities over time, this approach is particularly suited to detect fades in video [14]. The decision regarding presence of a *break* is based on an appropriate selection of the *threshold* value.

Feature-based techniques are based on global or local representation of the image frames. Pixel-value itself can be a feature, we discussed this subclass in the preceeding paragraphs. Most commonly used feature for video segmentation is a histogram, e.g., [6], [16], [18] and [20].

A histogram is created for the current frame by calculating the number of times each of the discrete pixel value appears in the frame. These histogram-based techniques extract and normalize a vector equal in size to the number of levels the image is coded in. The vector is then compared with or matched against other vectors of similar images in the sequence to confirm a certain minimum degree of dissimilarity. If such a criterion is successfully met, the corresponding image is labeled as a *break*. The normalized histogram would be calculated as:

$$H_t\left(i\right) = \frac{1}{N} \cdot \sum_{\forall\ x,y} \zeta(x,y) \qquad (3)$$

where,

$$\zeta(x,y) = 1\ if\ I\left(x,y\right) = i,\ else\ \zeta(x,y) = 0$$
$$i\ -\ index\ of\ the\ histogram\ bin,\ and$$
$$N\ -\ normalization\ factor.$$

However, there are many metrics for matching the histograms. Most of the metrics are derived from the difference, the intersection or the product, or a square - see Duda et al. [5] for details.

Dugad et al. [6] have extended on this work by including a second stage in their detection process. In order to minimize the number of missed detections and the number of incorrect classifications, they proposed use of a *likelihood ratio* for comparing two frame regions. The frames are first divided into smaller image blocks, and these blocks are then compared using statistical measures. Their likelihood ratio is computed as:

$$Lr = Z/\left(\sigma_t\ \times\ \sigma_{t-1}\right) \qquad (4)$$

where

$$Z = \left[\left(\sigma_t - \sigma_{t-1}\right)/2\ +\ \left(\left(m_t - m_{t-1}\right)/2\right)^2\right]^2$$

and, m_t and σ_t denote the statistical mean and standard deviation, respectively, of the given region in frame t. This ratio attains its minimum value if $\sigma_t = \sigma_{t-1}$ and $m_t = m_{t-1}$. There are many other variations proposed for histogram-based algorithms.

A major advantage of using histogram as a feature is that the histogram is relatively insensitive to the object-position in the frame, and thus, this technique is suitable in presence of camera and/or object motion. However, this measure is sensitive to noise, illumination changes and object-scaling and does not scale well with the matching. Nonetheless, histogram remains the most commonly used feature in video segmentation due to its ease of implementation and the effectiveness.

Many other features for video segmentation were used too - e.g., tracking of edges, motion vector differences and eigen space decomposition to maximize the data variation. Apart from thresholding, hidden-Markov models, tree-classifier, supervised learning and clustering were used by different researchers to detect the shots. For merits, limitations and the performance characterization of each of the approaches - see [11].

Another way to categorize the video-partitioning process is the type of video-data used as the input. Some of the algorithms were designed to work on compressed video stream, e.g., Yeo & Liu's algorithm [20] uses MPEG compressed video stream rather than the raw footage. However, this distinction is not very important, since practically all of the algorithms can be applied to the compressed as well as to the uncompressed data. There may be differences in how the certain features are computed, but the core concept of the algorithm remains the same.

Other distinction of video-data is the grey-level input or the color input to compute the discontinuity. Color videos are used most often with different color spaces such as RGB, HSV, YIQ, Munsell and other color spaces [19]. Main advantage of using color is the additional information available for detecting the discontinuity both in spatial and temporal spaces, and the ease of implementation.

Shot detection algorithms are further classified based on their suitability on detecting the specific types of the transitions - hard-cut, fade, wipe, slide or dissolve. Decision parameters vary for transition types and the chosen algorithm. Lienhart [14] discussed the underlying concepts behind each transition types based on the characterization of the video-data in terms of higher-level semantics, and suggested guidelines for use of the tested approaches.

Choice of the best video partitioning method is not straightforward. There are many factors that affect the performance. Most important parameters affecting the performance are the decision parameters which vary from data-to-data, algorithm-to-algorithm and the transition-type-to-another-type. In this work, we attempt to extract the decision parameters from the characterization of the video data in terms of statistical and information-theoretic measures, and apply to detect the appropriate transitions.

3. The Algorithm

The algorithm proposed here is fundamentally a histogram-matching technique. In addition, however, it incorporates three additional features. Firstly, the histograms used are *weighted* histograms; the definition and construction of such histograms is outlined in sub-section 3.1. Secondly, an error-propagation model is introduced to induce robustness, this is described in sub-section 3.2. Matching of the histograms using a χ^2 measure is presented in sub-section 3.3. Lastly, sub-section 3.4 discusses a statistical method based on modified B-Spline curve fitting which obviates the need for any heuristically determined threshold value while deciding upon camera breaks.

3.1. Weighted Histogram

The weighted histogram function is defined below:

$$H_t(i) = \frac{1}{N} \cdot \sum_{\forall\, x,y} \zeta(x,y) \tag{5}$$

where,

$$\zeta(x,y) = \pi(x,y)\ if\ I(x,y) = i,\ else\ \zeta(x,y) = 0,$$
$$\pi(x,y) = 1 - E_p \times k, \quad k = 0, 1, 2, ...\zeta,\ and$$
$$E_p\ -\ a\ parameter\ set\ by\ information\ content.$$

Here, the image matrix is first divided into ζ concentric regions, and each is assigned a weight. The increment to the histogram value $H_t(i)$ is now no longer a binary value; it now depends on the physical location of the pixel under consideration. The ζ concentric regions can be linked to margins drawn around a central region of maximum interest or weight; any pixel existing within this region will contribute a unit increment to the histogram value. This central region is indexed by k = 0, and the outermost region by k = ζ.

Such an algorithm implicitly assigns less weight to the pixels bordering the region of interest, which, in most cases of tracking and panning, is the central region of the frame. The higher the probability of the input video clip containing sequences with rapid tracking or panning, the larger should be the value of E_p chosen, while the nature of the objects being tracked decides the value of the parameter ζ; smaller objects in the sequence would call for a larger value of ζ.

A weighted histogram based approach has been successfully applied by Kumar & Rockett to extracting scale, translation and rotation invariant local features for object recognition from two-dimensional noisy images, and results along with a detailed discussion on weighted histogram are presented in [13].

3.2. Error Propagation

Global measures such as histograms describe the whole matrix by a single n-tuple, n is the number of bins in the histogram. Though the transformation of the complete frame into the representation space is straightforward and needs no heuristic or parameter definition, such a transformation should at the same time ensure that spurious data is recognized as such. Kumar & Rockett, [12] and [13], have examined the need for error-propagation to achieve such robustness in a histogram-matching algorithm. They stressed on the importance of ensuring the right mix of errors and their subsequent propagation, so that the effect of noise on the

ability of the algorithm to correctly match two functions is diminished. This way, they could eliminate the false signals and capture the diminishing signals because of noise and the other unknown variations.

Such noise creeps inevitably in video clips where changes in illumination etc. introduce more than a linear shift of the histogram. Although the innate shape of the histogram has hardly fluctuated, yet a histogram-matching algorithm (which is *largely* based on bin-to-bin comparison, difference or product) would view such a histogram as one which is significantly changed. Any matching criterion would then completely fail, and classify the histogram instance as one representing an entirely different entity.

In some solutions to this problem, the effects of data degradation are encountered by some tolerance at the matching/classification stage. Such approaches sacrifice the discriminatory capability of the matching criteria. Still others rely on a smoothing of the global representation; such global operations, however, may filter out important information, and may again lack discrimination. However, ensuring the right mix of errors and their subsequent propagation for a realistic blur can optimize the discriminatory power of the algorithm in presence of data-noise and degradation - see [12] for implementation details.

In this work, we introduce such an error-propagation by the application of a Gaussian window - normalized to unit area, which runs across the image histogram and blurs it to an extent that should, in the ideal situation, be determined through a process of content identification (in this case, this is done by measuring the information contents in a localized window [7]). This ensures that any insignificant level of noise in the signal is promptly ignored and that it does not manifest as a likely candidate for classification as a transition.

3.3. Matching Strategy

There are many metrics to match two histograms, e.g., simple or weighted difference, intersection or squared difference [5]. Our algorithm uses the Bhattacharrya metric as the matching criterion. The Bhattacharrya distance [2], which is a generalized χ^2 measure, possesses some immunity to noise and clutter. Here one attempts to maximize the energy function, which, here, is merely the dot product between the two tuples under consideration. Mathematically, the Bhattacharrya metric is defined as:

$$M_t = \sum_{i\ over\ all\ bins} H_t\left(i\right) \odot H_{t-1}\left(i\right) \qquad (6)$$

where,

M_t – *matching or similarity measure,*
H_t – *normalized histogram for frame* F_t*, and*
$\odot$ – *dot product of histograms.*

See Kumar & Rockett [13] for the the normalization of the histogram and their dot-product.

3.4. Labeling Partitions

Many algorithms impose a hard cut-off threshold to discriminate between genuine and fake candidates. But such an approach has obvious drawbacks: a video clip can contain gradual shot transitions that would yield neither a sharp nor a deep minimum but would instead give rise to a cluster or rather shallow minima. The issue would then be deciding which single member of the cluster to label as a camera break. It becomes the need, therefore, to devise an algorithm that will be able to recognize a relatively deep minimum, one whose depth and sharpness in that video clip qualifies it as a valid partition. This has been achieved here through a two-stage elimination process, which is discussed in the following sub-sections.

3.4.1 Curve Fitting Using B-Spline Curves

Distinct breaks are always very sharp, often resembling inverted delta functions. So any curve fitting technique is bound to ignore partition points completely. Even a fifth degree interpolating polynomial would not be able to include the cut points in its approximated curve.

One approach would be to apply curve fitting using an elementary polynomial such as third or fourth degree Newton-Gregory polynomial, and then labeling only those points that have been ignored in plotting the final curve. However the results would be unpredictable for two reasons: there is no guarantee that every partition point would necessarily be ignored by the interpolating polynomial; secondly, the algorithm would introduce a number of false minima in attempting to include the partition points.

A simpler and more reliable route to achieving the same goal is to merely take the first difference of the raw curve, and apply any interpolating method to determine the partitions. This is the method adopted here. The degree of the interpolating polynomial should, again, be determined by an approximate beforehand knowledge of the nature of the video clip, although using a higher degree polynomial (of say five) would always give accurate results but it increases the computational complexity of the algorithm.

The interpolating polynomials used here are the cubic B-Spline curves. These curves are like Bezier curves in that they do not ordinarily pass through any of the sets of given data points. They have the following additional features:

- A B-Spline does not begin and end at the extreme points.
- The slopes of the B-Spline do not bear any simple relation with the lines drawn between the points.

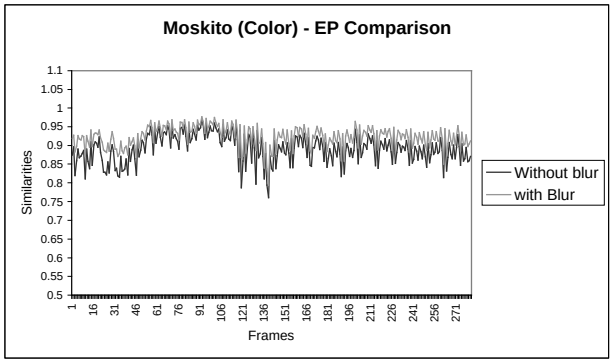

Figure 1: Two plots of the similarity curves for the *Moskito* video clip taken from [15].

The advantage with using B-Splines is that a separate cubic is drawn for each pair of points in the set, which is exactly how the histogram comparison has been done, [5] and [17].

3.4.2 Statistical Thresholding

The minima singled out by the curve fitting are then subjected to a depth-test. If the depth of the minimum under consideration is greater than a threshold value arrived at after a statistical evaluation of the raw data, then that minimum is finally labeled a partition point. To ensure that the minima labeled as partitions are correctly measured by their relative depth in the original similarity curve, the raw data available prior to the curve fitting should be used.

Thus, assuming a normal distribution without loss of generality, the similarity should fall below the following threshold for the point to be labeled a partition:

$$T_h = \mu - \alpha \cdot \sigma \tag{7}$$

where a good value for α would be between five and six.

4. Results and Discussion

We experimented each of the three steps - weighted histogram, error-propagation model and computation of the threshold value - on a large number of real digital videos that have different types of variability, e.g., illumination change and camera movement. The videos vary widely in content and length. We experimented with both types of videos - fast varying signal as well as slow varying signal - in order to test the efficacy of each step of the algorithm.

We include some example results in this part of the section along with a brief analysis. We are unable to include the different clips and the detailed results because of the limitation on the number of pages. Moreover, the purpose of this work is not to compare the detection performance of this work with the existing approaches, so we do not present the comaprison results here. Nevertheless, we include subset of the results which exhibits the improvement because of the distinct characteristics of the algorithm, e.g., the error-propagation model along with the weighted histogram and the computed value of the threshold for detection. The discussion that follows describes the performances of two representative classes of videos.

First we present the results taken from John's heli-pad video clips, there are many video-clip available on the site [15]. In general, each video-clip is a color video, contains a helicopter as the central object, which is moving, background is changing, illumination is getting changed, and camera is also moving in-and-out - visit the site [15] for viewing the video. Thus, this class of video contains the sufficient variations and is a good example to test the algorithm over a class of data to test the stability of the algorithm.

In this section, we include the results computed for a *Moskito* color-video clip as representative results. A helicopter is tracked rapidly with varying background across a sky of varying intensity. There are three sections in the clip that are of interest: the first one occurs between frames 25 and 46, the second between 118 and 139, and the third between frames 256 to 271. We plot similarity match values for the first 280 frames of the clip in Figure 1. In all three sub-shots, many false peaks were correctly eliminated upon error-propagation. For example, the second shot that includes frames 123 to 132 gave many false-breaks if there was no error-propagation, a total of 8 false-peaks were eliminated this way while one true signal at frame 130 remained with the computed threshold value.

Second, we include the representative results computed for a slow-varying video taken from a *kitchen*. In this clip, a kitchen is scanned, two new objects, a window and a lady rapidly enter the scene (frames 115 through 124). The Bhattacharrya match metrics, with and without the error-model, are ploted for part of the clip in Figure 2. The similarity curve with blur-model is clearly more evened out. In the absence of the blur, the algorithm detects three transitions at points 115, 120 and 123. This region is evened out when weighting and the errors are applied. All three false alarms which gave the illusion of a camera break, are eliminated by the computed value of the threshold to detect a *hard* cut. On the other hand, a true transition at point 75 is detected distinctly by both.

We tested our approach with many video clips of different themes and varying nature. We compared the results in terms of false-alarms and miss-detections with and without the proposed improvements: weighted *vs.* windowed histograms, with and without error-propagation, and B-spline curve fitting. We found that the algorithm works well in videos of rapid scanning and for tracking of objects.

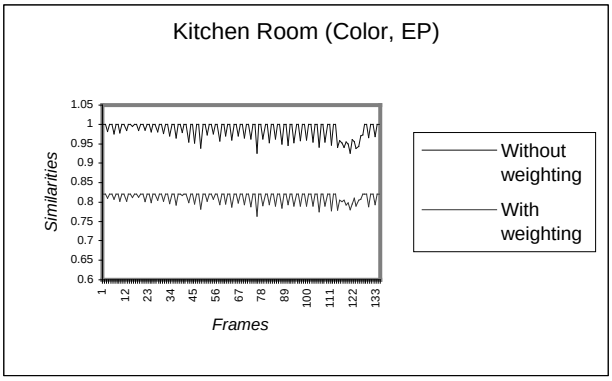

Figure 2: Two plots of the similarity curves for the *kitchen* video clip.

The error-propgation helps in detecting false-alarms due to change in illumination. We are unable to include representative results along with a detailed analysis due to the page-limit; we shall present them during the conference.

5. Conclusions

In this paper, we have described a novel algorithm for shot boundary detection that detects camera break in a full motion video. The technique is a weighted histogram based algorithm which includes an error-propagation model to induce robustness. To avoid the selection of a heuristically selected threshold value for detection, we use an information theoretic measure which is guided by the information content. We experimented with many digital videos of both types - slow varying and fast varying signal - and diverse in content, and thus demonstrated the generic nature of the algorithm. Results are compared with and without introducing the error-propagation, and it is shown that the error-propagation model minimizes the false positives without affecting the presence of the true signals. Extending this work to an effective technique to extracting and setting of various parameters to detecting different types of transition types is within the scope of future work.

References

[1] E. Ardizzonei and M. L. Cascia. Automatic Video Database Indexing and Retrieval. In *Multimedia Tools and Applications*. Kluwer, 1996.

[2] A. Bhattacharyya. On a Measure of Divergence Between Two Statistical Populations Defined by Their Probability Distributions. *Bull. Calcutta Math. Soc.*, 35:99 – 110, 1943.

[3] J. S. Boreczky and L. A. Rowe. Comparison of Video Shot Boundary Detection Techniques. In *SPIE Storage and Retrieval for Still Images and Video Databases IV*, volume 2664, pages 170 – 179, 1996.

[4] P. Bouthemy and F. Ganansia. Video Partitioning and Camera Motion Characterization for Content-Based Video Indexing. In *Proc. Third IEEE International Conference on Image Processing*, 1996.

[5] R. O. Duda, D. G. Stork, and P. E. Hart. *Pattern Classification and Scene Analysis Part 1: Pattern Classification.* John Wiley, 2000.

[6] R. Dugad, K. Ratakonda, and N. Ahuja. Robust Video Shot Detection. In *Proc. Indian Conference on Computer Vision, Graphics and Image Processing*, pages 358 – 364, 1998.

[7] B. Furht, S. W. Smoliar, and H. Zhang. *Video and Image Processing in Multimedia Systems*. Kluwer, 1996.

[8] U. Gargi, R. Kasturi, and S. H. Strayer. Performance Characterization of Video Shot Change Detection Methods. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1), Feb 2000.

[9] W. Grosky, R. Jain, and R. Mehrotra. *The Handbook of Multimedia Information Systems*. Prentice Hall, 1997.

[10] A. Hampapur, R. Jain, and T. Weymouth. Digital Video Segmentation. In *Proc. ACM Multimedia*, pages 357 – 364, 1994.

[11] I. Koprinska and S. Carrato. Temporal Video Segmentation: A Survey. *Signal Processing: Image Communication*, 16:477 – 500, 2001.

[12] R. Kumar. Propagating Errors into Feature Representation for Robustness of Local Invariants. In *Proc. Indian Conference on Computer Vision, Graphics and Image Processing*, pages 159 – 166, 1998.

[13] R. Kumar and P. I. Rockett. Triplet Geometric Representation: A Novel Scale, Translation and Rotation Invariant Feature Representation based on Geometric Constraints for Recognition of 2D Object Features. *Image and Vision Computing*, 15(3):235 – 249, March 1997.

[14] R. Lienhart. Reliable Transition Detection in Videos: A Survey and Practitioner's Guide. *International Journal of Image and Graphics*.

[15] Moskito. John's Heli Pad Video Clip - Radio Controlled Helicopters. `http://www.modelaviation.co.uk/heli/video/model/video.htm`.

[16] N. V. Patel and I. K. Sethi. Video Shot Detection and Characterization for Video Databases. *Pattern Recognition*, 30(4):583 – 592, April 1997.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C++*. Cambridge University Press, 2002.

[18] J. Shi and J. Malik. Motion Segmentation Using Normalized Cuts. In *Proc. Sixth International Conference on Computer Vision*, pages 1154 – 1160, 1998.

[19] M. J. Swain and D. H. Ballard. Color Indexing. *International Journal of Computer Vision*, 7:11 – 32, 1991.

[20] B. Yeo and B. Liu. Rapid Scene Change Analysis on Compressed Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(6):533 – 544, 1995.

[21] D. Zhang, W. Qi, and H. J. Zhang. A New Shot Boundary Detection Algorithm. *Lecture Notes in Computer Science*, 2195:63 –, 2001.

[22] H. J. Zhang, A. Kankanhalli, and S. W. Smoliar. Automatic Partitioning of Full Motion Video. *Multimedia Systems*, 1:10 – 28, 1993.