Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

# Fuzzy color histogram-based video segmentation

Onur Küçüktunç, Uğur Güdükbay *, Özgür Ulusoy

*Bilkent University, Department of Computer Engineering, Bilkent, 06800 Ankara, Turkey*

ARTICLE INFO

*Article history:*
Received 19 March 2009
Accepted 13 September 2009
Available online 18 September 2009

*Keywords:*
Video segmentation
Shot boundary detection
Fuzzy color histogram
Cut/gradual transition
Content-based copy detection
Video analysis

ABSTRACT

We present a fuzzy color histogram-based shot-boundary detection algorithm specialized for content-based copy detection applications. The proposed method aims to detect both cuts and gradual transitions (fade, dissolve) effectively in videos where heavy transformations (such as cam-cording, insertions of patterns, strong re-encoding) occur. Along with the color histogram generated with the fuzzy linking method on $L^*a^*b^*$ color space, the system extracts a mask for still regions and the window of picture-in-picture transformation for each detected shot, which will be useful in a content-based copy detection system. Experimental results show that our method effectively detects shot boundaries and reduces false alarms as compared to the state-of-the-art shot-boundary detection algorithms.

## 1. Introduction

Recent developments in multimedia technology with the significant growth of media resources introduced content-based copy detection (CBCD) as a new research field alternative to the watermarking approach for identification of video sequences. The detection of shots, as in many video indexing and retrieval applications, is the first step of video analysis. A shot is defined as a series of related consecutive frames representing a continuous action in time and space taken by a single camera [1].

A video is composed of several shots combined with abrupt or gradual transitions. An abrupt transition, also known as hard-cut, is the most common and easy to detect transition type. On the other hand, gradual transitions (fades, dissolves and wipes) are spread over a number of frames, thus they are harder to detect.

Various shot-boundary detection algorithms have been proposed [1–6] and compared [7–10]; however, to the best of our knowledge, no shot-boundary detection algorithm specialized for CBCD is found in the literature. Our aim is to propose an automatic shot-boundary detection algorithm for the videos on which various transformations are applied. In contrast to most of the existing methods, we utilize fuzzy logic approach for extracting color histogram to detect shot boundaries. We evaluate the proposed method in the query dataset prepared for CBCD task of TRECVID 2008, and show the accuracy of the system.

The paper is organized as follows: Section 2 discusses related work. Section 3 explains the video transformations used for preparing a dataset for CBCD. Section 4 describes the proposed shot-boundary detection algorithm, as well as the methods for detecting frame-dropping and picture-in-picture transformations, noise detection, and mask generation. We present experimental results and evaluate the performance of the proposed methods in Section 5. Section 6 gives conclusions and future work.

## 2. Related work

Studies on shot-boundary detection are typically based on extracting visual features (color, edge, motion, and interest points) and comparing them among successive frames. Truong et al. [6] propose techniques for cut, fade, and dissolve detections. An adaptive thresholding technique to detect peaks in the color histogram difference curve is presented for detecting hard cuts. Locating monochrome frames and considering luminance mean and variance are the steps for fade and dissolve detection. Danisman and Alpkocak [11] apply a method based on color histogram differences in RGB color space and thresholding for cut detection. They present skip frame interval technique, which reduces the computation time with a slight decrease in the precision. Dailianas et al. [9], Boreczky and Rowe [10] compare early shot-boundary detection algorithms. Lienhart [8] extends this comparison by taking newer algorithms into account, and by measuring their ability to detect the type and temporal extent of the transitions. Cotsaces et al. [7] give an up-to-date review.

In recent years, researchers focus on detecting gradual transitions effectively and avoiding the false alarms caused by flashlight and the motion of large objects in the scene, since the recognition of hard-cuts is very reliable for most of the methods. Huang et al. [2] propose an approach based on local keypoint matching of video

* Corresponding author. Fax: +90 312 266 4047.
*E-mail addresses:* onurk@cs.bilkent.edu.tr (O. Küçüktunç), gudukbay@cs.bilkent.edu.tr (U. Güdükbay), oulusoy@cs.bilkent.edu.tr (Ö. Ulusoy).

1077-3142/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2009.09.008

frames to detect abrupt and gradual transitions. By matching the same objects and scenes using contrast context histogram (CCH) in two adjacent frames, the method decides that there is no shot change. Grana et al. [3] propose a two-step iterative algorithm, unique for both cuts and gradual transitions detection, in the presence of fast object motion and camera operations. Boccignone et al. [1] use a consistency measure of the fixation sequences generated by an ideal observer looking at the video for determining shot changes. A scene-break detection approach based on linear prediction model is proposed in [4]. Shot-boundaries are detected using Bayesian cost functions, by comparing original frame with the predicted frame, estimated using within video shot linear prediction model (WLPM) and dissolve linear prediction model (DLPM). Yuan et al. present a unified shot boundary detection system based on graph partitioning model [5]. The representation of the visual content, the construction of the continuity signal, and the classification of continuity values are handled in this work. The evaluations show that the SVM-based active learning outperforms both thresholding and nonactive learning.

Fuzzy logic introduced by Zadeh [12] is being used in many applications related to image processing. Konstantinidis et al. [13] and Han and Ma [14] utilize fuzzy logic for creating color histograms to be used in content-based image retrieval systems. Chung and Fung [15] introduce fuzzy color quantization to color histogram construction, and evaluate its performance in video scene detection with a very limited video dataset. Fang et al. [16] propose a fuzzy logic approach for temporal segmentation of videos, where color histogram intersection, motion compensation, texture change and edge variances are integrated for cut detection. In [17], histogram differences of consecutive frames are characterized as fuzzy terms, such as small, significant and large, and fuzzy rules for detecting abrupt and gradual transitions are formulated in a fuzzy-logic-based framework for segmentation of video sequences. Das et al. [18] define a unified interval type-2 fuzzy rule based model using fuzzy histogram and fuzzy co-occurrence matrix to detect cuts and various types of gradual transitions.

In the field of CBCD, representing video with a set of keyframes (one or more representative frame for each shot) is a common approach. Some of the recent studies on CBCD task of TRECVID 2008 employ the following techniques. Llorente et al. [19] use an approach based on color histogram and thresholding, extended by [20] for detection of gradual transitions. Douze et al. prefer extracting 2.5 frames per second for query videos, and extracting only a few representative keyframes for the dataset [21]. We also preferred extracting a fixed number of frames per time interval in our CBCD system [22]. Studies in video copy detection domain, therefore, do not necessarily use a shot-boundary detection method.

## 3. Video transformations

In order to develop a shot-boundary detection algorithm specialized for CBCD applications, we need to understand the effects of transformations used for modifying videos. For the first time in 2008, TRECVID [23] evaluated CBCD systems. Each query video is constructed by taking a segment from the test collection, transforming and/or embedding into some other video segment, and finally applying one or more transformations to the entire query segment [24]. Since the query set prepared for CBCD task will be used for evaluation purposes, we will focus on the transformations in Table 1 [25]. These transformations cover most of the video modifications in daily life (cf. Fig. 1). Although some transformations do not have an effect on shot-boundary detection process, different strategies for different transformations should be applied for an effective shot-boundary detection, and also for CBCD process afterward. Here, we discuss the negative effects of video transformations, and possible corrective actions taken by our method:

i *Frame dropping:* Dropped frames should be ignored or estimated; otherwise the shot-boundary detection algorithm decides each blank frame as a cut. Such frames have the mean of intensity values near to zero.
ii *Picture-in-picture:* Regardless of which type is applied to the video segment, detecting the window of picture-in-picture transformation (boundaries of the inner video) is crucial for feature extraction step of the CBCD system [26]. With the extracted window, foreground and background frames can be handled separately.
iii *Insertion of patterns, caption:* Although the insertion of a pattern or text does not affect the shot-boundary detection process strongly, a mask for still regions, which includes the inserted pattern or text, will increase the effectiveness of a CBCD system. Unmasked patterns and captions introduce new edges and regions of interests, and cause changes on color information.
iv *Cam-cording, crop, shift:* These transformations generally produce black framings on one or more sides of the video segment. Since the framings are also still regions, we can ignore these areas during the feature extraction.
v *Strong re-encoding, blur, change of gamma, contrast, compression:* It is important to use a keypoint detector invariant to these changes. These changes have nearly no effect on shot-boundary detection because they are applied on the whole video with the same parameter values.
vi *Noise:* Since the detection of windows for picture-in-picture transformation depends on edge detection, noisy shots should be discovered and handled before further processing.

## 4. Methods

We provide the details of our shot-boundary detection method and other techniques that we use to identify the transformations applied on a query video. The parameters of the system are given in Table 2, and the overview of the proposed algorithm is presented in Fig. 2.

### 4.1. Detection of frame-dropping transformation

Handling frame-dropping transformation is one of the key features of a shot-boundary detection system specialized for CBCD applications; since most of the proposed algorithms consider missing frames as hard-cuts. A dropped frame is either exactly or nearly a blank frame, which has a small overall intensity (less than $th_{bf} = 0.0039$). We define a binary function $fd$ for a given video frame $I_n$ as:

$$fd(I_n) = \begin{cases} 1 & \sum_{i=1}^{h}\sum_{j=1}^{w} G_n(i,j) < th_{bf} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $G_n$ is the grayscale intensity image of $I_n$, and $(h,w)$ is the dimension of the frame.

### 4.2. Noise detection

CBCD applications should handle query videos with heavy noise transformations. For our algorithms to work properly, noisy frames/shots should be identified before any further operation that is based on edge detection or use standard deviation of pixel intensity values.

In image processing, a nonlinear median filter is preferred over a linear filter for cleaning salt and pepper and white noise. Based on this fact, we calculate the average intensity change of an image $I_n$ after a median filter of size $s_{mf} \times s_{mf}$ is applied to the image. If the

**Table 1**
The list of transformations used in the CBCD task.

| # | Transformation details |
|---|---|
| T1 | Cam-cording |
| T2 | Picture-in-picture Type 1 |
| T3 | Insertion of patterns (15 different patterns) |
| T4 | Strong re-encoding (change of resolution, bitrate) |
| T5 | Change of gamma |
| T6 | Combination of 3 transformations amongst: blur, gamma, frame dropping, contrast, compression, ratio, noise (A) |
| T7 | Combination of 5 transformations amongst (A) |
| T8 | Combination of 3 transformations amongst: crop, shift, contrast, caption, flip, insertion of pattern, picture-in-picture Type 2 (original video is behind) (B) |
| T9 | Combination of 5 transformations amongst (B) |
| T10 | Combination of 5 transformations amongst all the transformations from 1 to 9 |

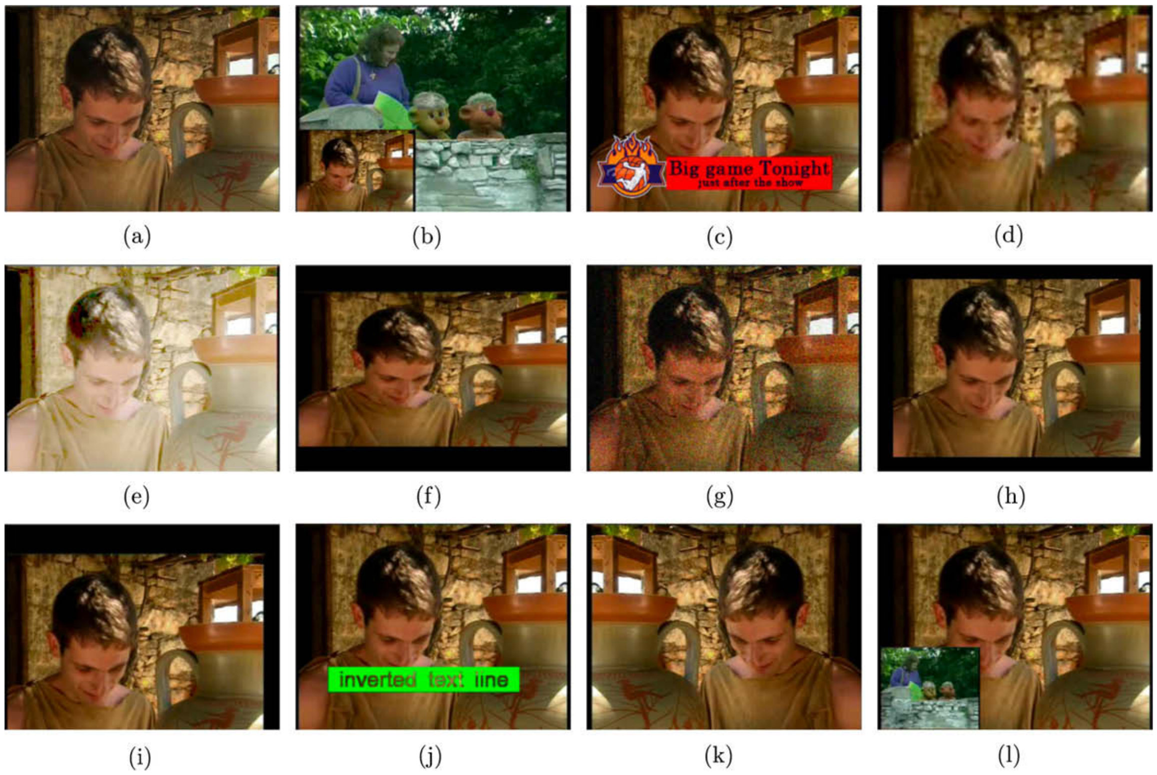

**Fig. 1.** Transformations: (a) original frame, (b) picture-in-picture Type 1, (c) insertion of pattern, (d) strong re-encoding, (e) change of gamma, (f) letterbox, (g) white noise, (h) crop, (i) shift, (j) caption/text insertion, (k) flip, and (l) picture-in-picture Type 2.

**Table 2**
The parameters of the algorithm.

| Parameters | Description |
|---|---|
| $\theta_c$ | Threshold for cut detection |
| $\theta_g$ | Threshold for gradual transition detection |
| $\tau$ | Timescale for central moving average filter |
| $th_{bf}$ | Intensity threshold for blank frame detection |
| $th_n$ | Average intensity-change threshold for noisy image |
| $th_{sr}$ | Threshold for still regions |
| $s_{mf}$ | Size of median filter used in noise detection |

image slightly changes after the median filter, we assume that less noise exists in the image. Otherwise, when the average intensity change exceeds a threshold $th_n$, it is regarded as noisy.

$$nf(I_n) = \begin{cases} 1 & \frac{1}{h\times w}\sum_{i=1}^{h}\sum_{j=1}^{w}|G_n(i,j) - M_n(i,j)| > th_n \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

We evaluate the noise detection method and the impact of the parameters (the size of the median filter and the threshold value) in Section 5.

### 4.3. Mask generation

When a video segment is transformed with various types of transformations summarized in Table 1, it clearly changes the content of the frames regarding color, edge, and shape information. A content-based copy detection system should cut out the artificially inserted texts, patterns, logos, etc., if possible. Besides, it should ignore the bordering black areas produced by shift, crop, and letterbox transformations. As a result, the probability of matching with the original video segment is increased. We calculate the standard deviation of each intensity value of the pixels within the shot, assuming that pixel intensity varies from 0 to 1:

$$M_{shot}(i,j) = \begin{cases} 1 & \sigma_{shot}(i,j) > th_{sr} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

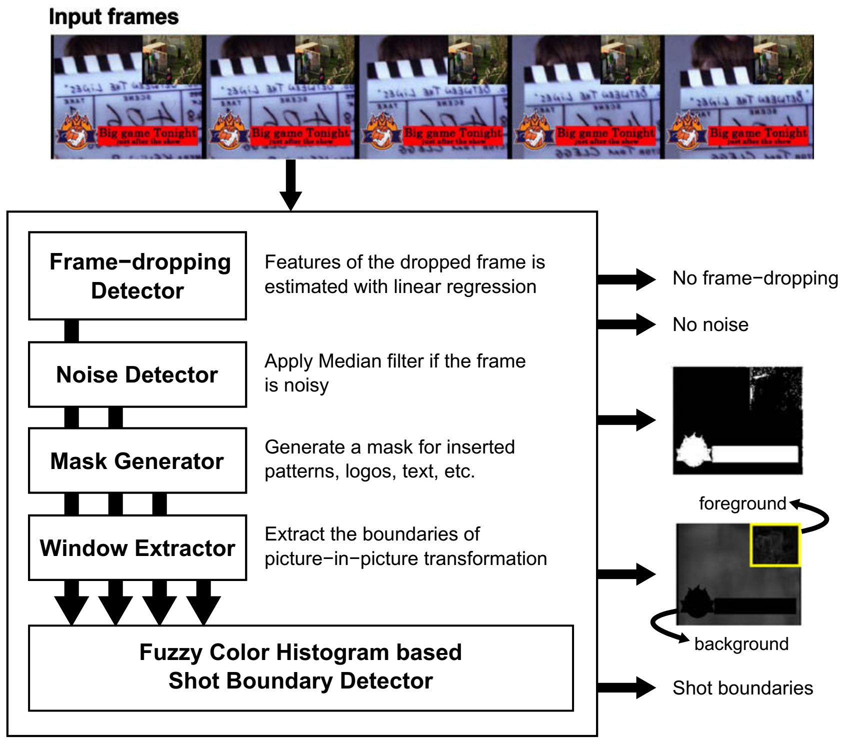
Input frames
Frame−dropping Detector
Features of the dropped frame is estimated with linear regression
No frame−dropping
Noise Detector
Apply Median filter if the frame is noisy
No noise
Mask Generator
Generate a mask for inserted patterns, logos, text, etc.
Window Extractor
Extract the boundaries of picture−in−picture transformation
foreground
background
Fuzzy Color Histogram based Shot Boundary Detector
Shot boundaries

**Fig. 2.** The overview of the proposed algorithm.

We create a mask of standard deviations greater than the threshold $th_{sr} = 0.01$ for each shot representing still regions while detecting shot boundaries. The mean and standard deviation of the pixel intensity values within a video shot of $N$ frames are given by Eqs. (4) and (5), respectively:

$$\mu_{shot}(i,j) = \frac{1}{N}\sum_{k=1}^{N} G_k(i,j) \tag{4}$$

$$\sigma_{shot}(i,j) = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(G_k(i,j) - \mu_{shot}(i,j))^2} \tag{5}$$

The problem here is that today's computers have a limitation that can hold up to a number of frames together in memory. Therefore, we employ the solution for incremental standard deviation calculation discussed by Knuth [27], who cites Welford [28]:

$$\mu_k(i,j) = \mu_{k-1}(i,j) + \frac{G_k(i,j) - \mu_{k-1}(i,j)}{k} \tag{6}$$

$$s_k = s_{k-1} + (G_k - \mu_{k-1}) \times (G_k - \mu_k) \tag{7}$$

$$\sigma_k(i,j) = \sqrt{s_k(i,j)/(k-1)} \tag{8}$$

where $\mu_1(i,j) = G_1(i,j)$ and $s_1(i,j) = 0$ initially. For a shot with $n$ frames, we save the mask $M_{shot} = M_n$ and the standard deviation of the shot $\sigma_{shot} = \sigma_n$ for further use.

### 4.4. Detection of picture-in-picture transformation

In order to detect the window of picture-in-picture transformation, black framings on the sides of the video segment generated by cam-cording, crop, or shift transformations should be extracted first. We mark each row and column starting from the beginning and from the end as border rows if

$$\frac{1}{w}\sum_{c=1}^{w}\sigma_{shot}(i,c) < th_{sr} \tag{9}$$

holds for that row. Similarly, blank columns from the beginning and from the end are identified. If Eq. (9) returns false a row or column, we stop marking borderlines for that edge. Fig. 3 shows an example to border detection.

The next step is to detect the vertical lines. We crop out the borders from $M_{shot}$, and then find the derivatives with a first-order difference from both + and $-x$-axis using the Prewitt edge detector:

$$E_{shot} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * M_{shot} \tag{10}$$

Strong vertical edges are extracted from $E_{shot}$ using Hough lines [29]. Only vertical lines are selected, and compared in order to form a rectangular window. The candidate window(s) and the border information for each shot are stored. Fig. 4 displays examples of frames whose borders and windows are successfully detected.

### 4.5. Shot-boundary detection

We use a color histogram-based method generated with the fuzzy linking method on $L^*a^*b^*$ color space. A brief discussion on why $L^*a^*b^*$ color space is preferred, how the dimensions are subdivided into regions, their ranges, and the results of an experiment with popular colors, are provided in Appendix A.

Fuzzification of the inputs is achieved by using triangular membership functions for each component. $L^*$ is divided into 3 regions (black, gray, white), $a^*$ is divided into 5 regions (green, greenish, middle, reddish, red), and $b^*$ also is divided into 5 regions (blue, bluish, middle, yellowish, yellow). Membership functions of the inputs and the output are shown in Fig. 5.

In conventional color histograms, each pixel belongs to only one histogram bin, depending on whether the pixel is quantized into the bin or not. The conditional probability $P_{i|j}$ of the selected $j$th pixel belonging to the $i$th color bin is defined as a binary equation:

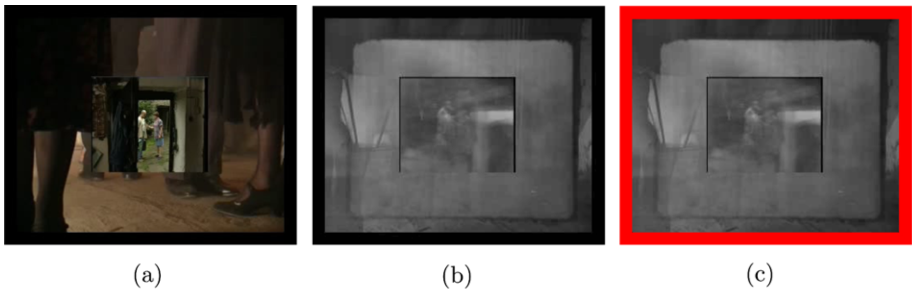
Fig. 3. The detection of borders: (a) first frame of a query video shot on which both the picture-in-picture and crop transformations are applied, (b) the standard deviation of the shot, and (c) the border shown in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

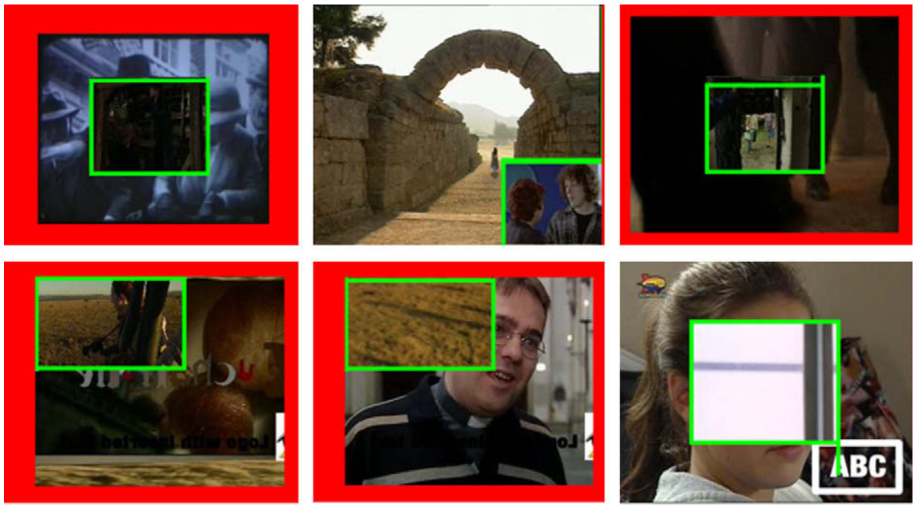
Fig. 4. Detected borders and windows for query frames. The borders are shown in red and the window frames are shown as green rectangles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$P_{ij} = \begin{cases} 1 & \text{if the } j\text{th pixel is quantized into the } i\text{th histogram bin} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

If $L^*a^*b^*$ color space was partitioned into $3 \times 5 \times 5$ (for $L^*$, $a^*$, and $b^*$, respectively) subspaces in a conventional manner, this definition would lead to serious boundary issues and problems related to the partition size. However, in the context of fuzzy color histogram, the degree of association $\mu_{ij}$ of $j$th pixel to $i$th bin is calculated with fuzzy membership functions (see Fig. 5). $L^*$ component of a pixel might have both a degree of *gray* and *white* together, for instance. Therefore, the color of a pixel is better-represented in fuzzy color histograms, even with a small number of membership functions.

Three components are linked in a Mamdani-style fuzzy inference system, according to 26 fuzzy rules (see Appendix B). The final color histogram is constructed using 15 trapezoidal membership functions for each bin of the output color histogram. Because some colors (olive, purple, silver, lime, maroon) reside very close to the others in 3-d $L^*a^*b^*$ space, we selected the remaining 15 colors out of 20 (see Appendix A). Therefore, the final fuzzy color histogram contains 15 bins. The overview of the proposed fuzzy inference system is shown in Fig. 6.

The main advantage of the proposed fuzzy color histogram over a conventional color histogram is its accuracy. Since the system is more robust to illumination changes and quantization errors, it performs better on shot boundary detection. Fig. 7 displays two successive frames in a gradual transition with their fuzzy and gray-scale histograms.

For frame-dropping transformations, we estimate the missing frames using linear regression. The fuzzy color histogram of a dropped frame is predicted by averaging the features of the previous two frames:

$$H_n = \begin{cases} h_n & fd(I_n) = 0 \\ (H_{n-1} + H_{n-2})/2 & \text{otherwise} \end{cases} \tag{12}$$

The essential idea of using color histogram for shot-boundary detection is that color content does not change rapidly within a shot. Therefore, shot changes are detected when fuzzy color histogram difference exceeds a threshold. The dissimilarity between color histograms of successive frames is calculated with Euclidean distance:

$$D(I_n, I_m) = \sqrt{\sum_{i=1}^{b} (H_n(i) - H_m(i))^2} \tag{13}$$

Although the difference between color histograms of successive frames in a video is enough to detect hard-cuts, the detection of gradual transitions (i.e., dissolve and fade) requires special treatment since these transitions are less responsive. In our method, we extend color histogram difference by the algorithm proposed in [20].

$$d_\tau(t) = \frac{1}{\tau} \sum_{i=0}^{\tau-1} D(t+i, t-\tau+i) \tag{14}$$

$d_\tau$ detects the transitions of duration less than or equal to $\tau$. We interpret peaks in $d_2$ greater than $\theta_c = 0.15$ as hard-cuts, and the

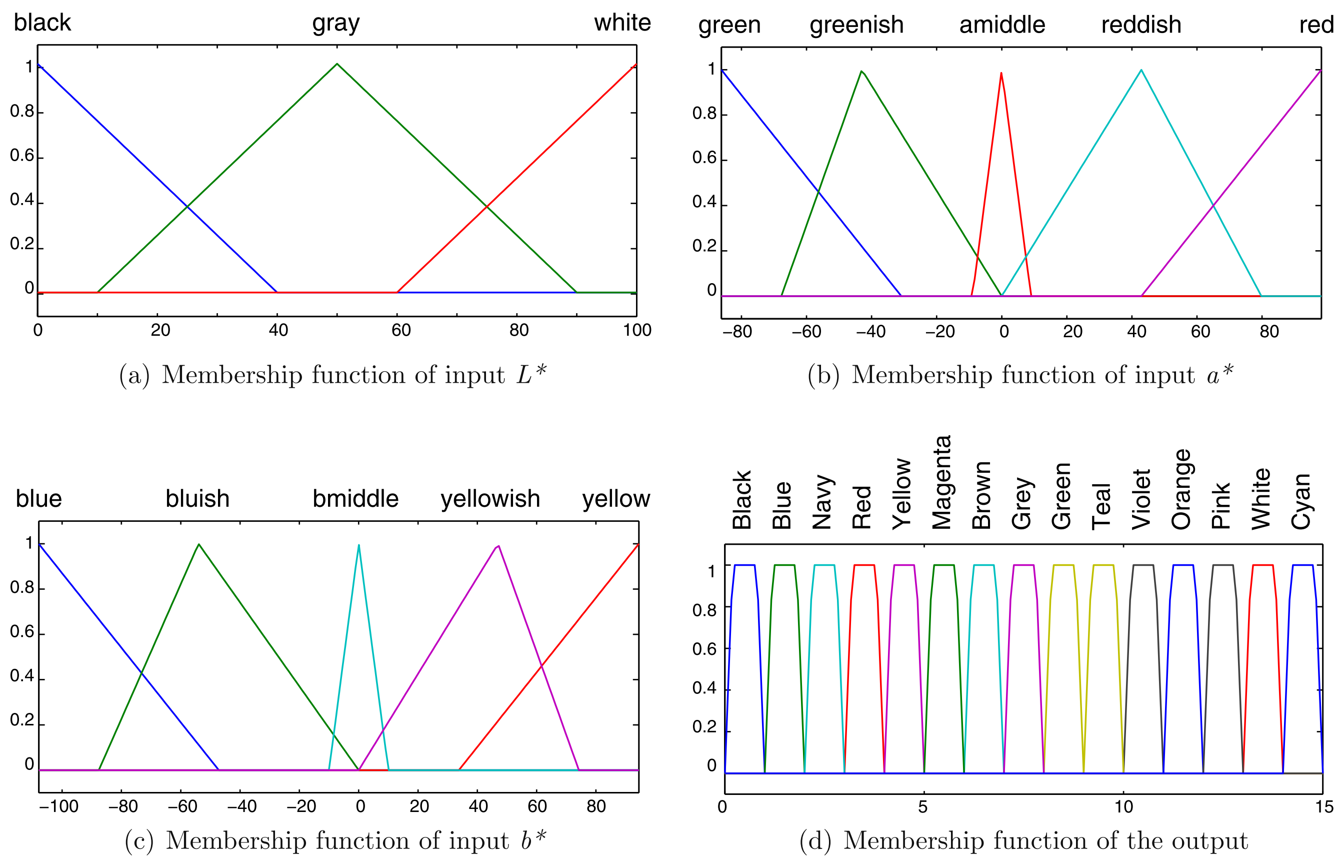

**Fig. 5.** Fuzzy membership functions for the inputs ($L^*$, $a^*$, $b^*$) and output.

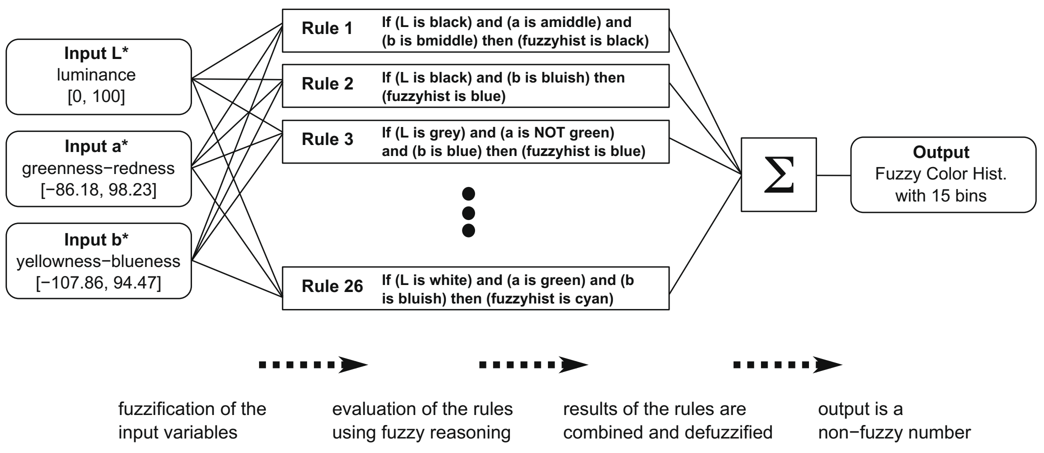
Input L*
luminance
[0, 100]
Input a*
greenness−redness
[−86.18, 98.23]
Input b*
yellowness−blueness
[−107.86, 94.47]
Rule 1 If (L is black) and (a is amiddle) and (b is bmiddle) then (fuzzyhist is black)
Rule 2 If (L is black) and (b is bluish) then (fuzzyhist is blue)
Rule 3 If (L is grey) and (a is NOT green) and (b is blue) then (fuzzyhist is blue)
Rule 26 If (L is white) and (a is green) and (b is bluish) then (fuzzyhist is cyan)
Σ
Output
Fuzzy Color Hist.
with 15 bins
fuzzification of the input variables
evaluation of the rules using fuzzy reasoning
results of the rules are combined and defuzzified
output is a non−fuzzy number

**Fig. 6.** The structure of the fuzzy color histogram.

remaining peaks in $d_4$, $d_8$, and $d_{16}$ greater than $\theta_g = 0.09$ as gradual transitions.

## 5. Evaluations and discussion

Three main parts of the proposed system, noise detection, shot-boundary detection, and window extraction of picture-in-picture transformation, are evaluated. Our test dataset is composed of query videos provided for TRECVID 2008 content-based copy detection task. There are total of 2010 MPEG-1 videos, which is about 80 h of video segments with various transformations applied (see Table 1). The important events that occur in query videos are shown in Fig. 8.

### *5.1. Evaluation of noise detection*

We used the query video set of TRECVID 2008 CBCD task, and extracted 1 frame per 2 s for each of 2010 videos. After decoding the videos, 33,478 images are manually labeled as 1 or 0, indicating that the frame is highly noisy or not.

Median filters of different sizes are evaluated and compared in an ROC curve (see Fig. 9). It is shown through experiments that the setting with $s_{mf} = 3$ and $th_n = 3.51$ gives an accuracy of 90.9% with a false alarm rate of 14.8%.

Most of the false alarms are caused by query videos that have noise originally, but not as a transformation. It should be noted

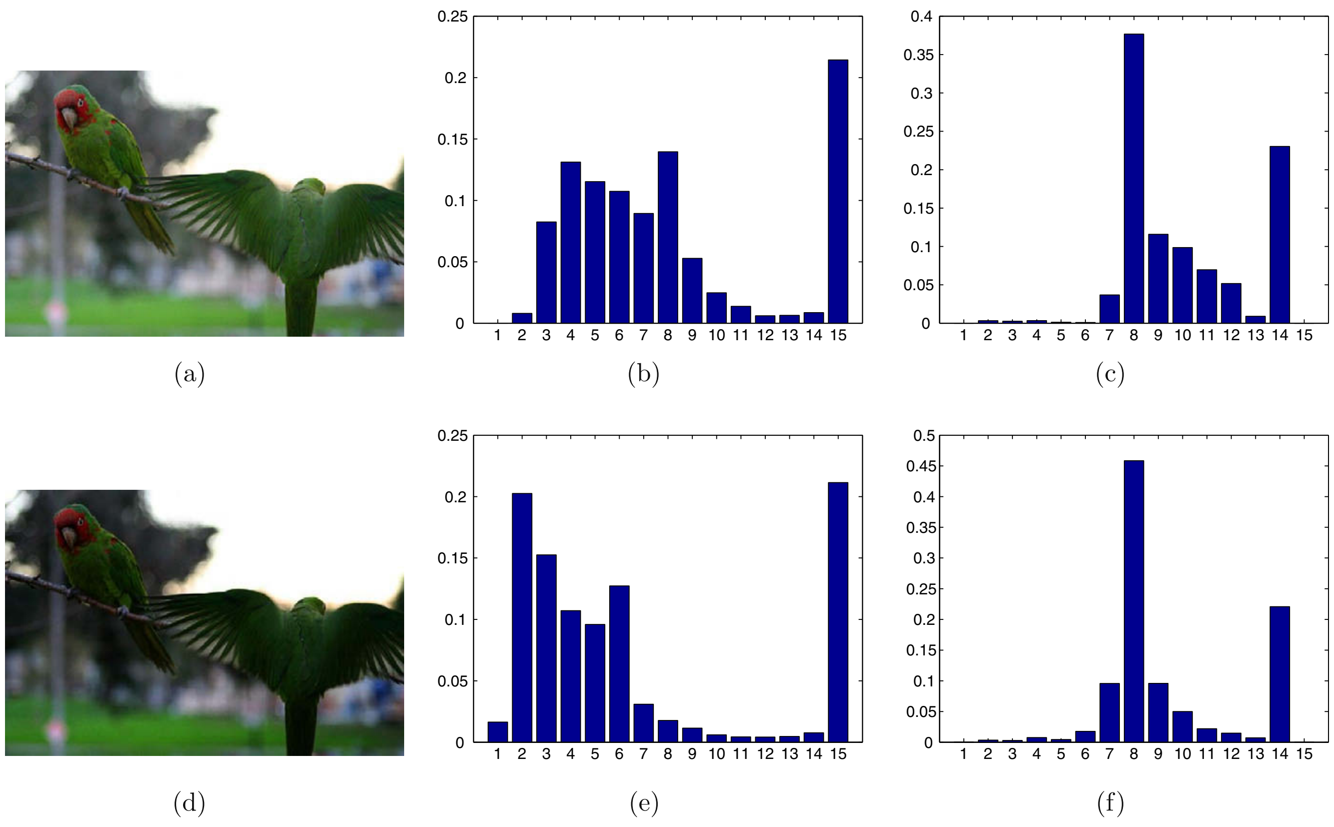

**Fig. 7.** Two successive frames in a gradual transition: (a) *i*th frame; (b) the gray-scale histogram of the *i*th frame; (c) the fuzzy color histogram of the *i*th frame; (d) $(i+1)$th frame; (e) the gray-scale histogram of the $(i+1)$th frame; (f) the fuzzy color histogram of the $(i+1)$th frame.

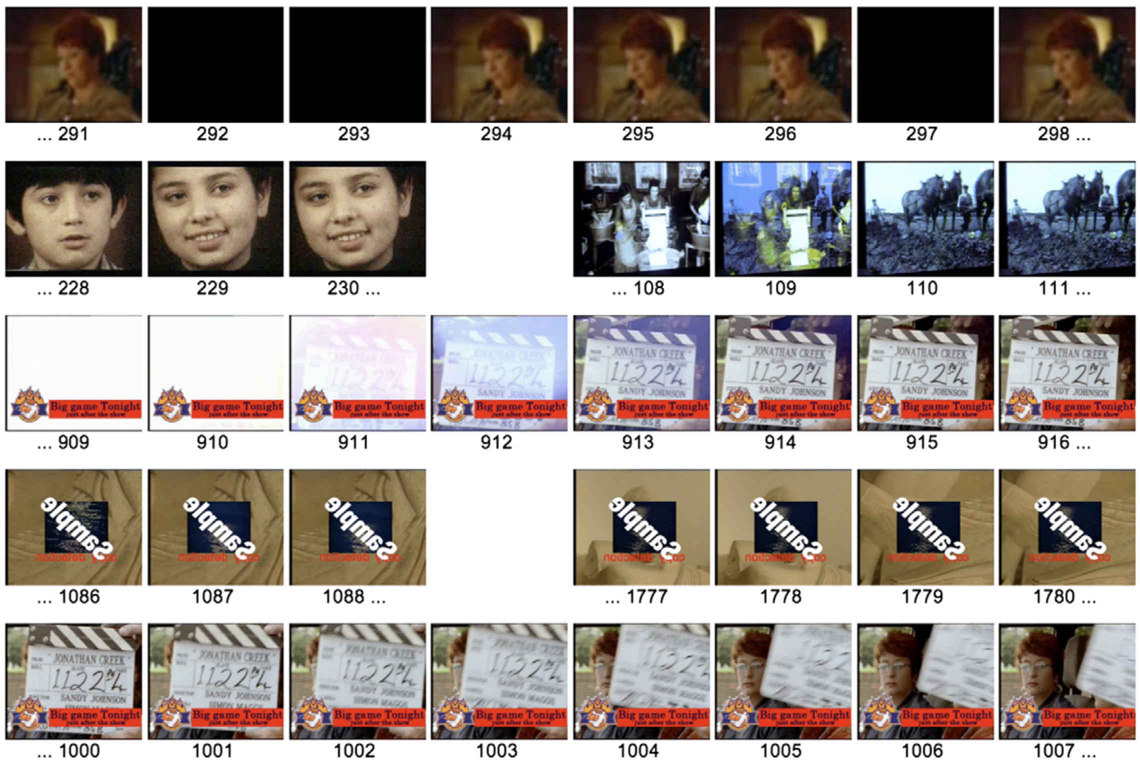

**Fig. 8.** The important events that occur in query videos: successive frames with frame-dropping transformation (*first row*), cut and dissolve transitions (*second row*), fade-in transition (*third row*), shot-boundaries for a picture-in-picture transformation-applied video, foreground changes at 1087, background at 1779 (*fourth row*), fast moving object in the scene (*last row*). Shot-boundaries during cut/gradual transitions (rows 2–3), and for background and foreground videos (row 4) are detected, while dropped frames (row 1) and fast object movements are ignored.

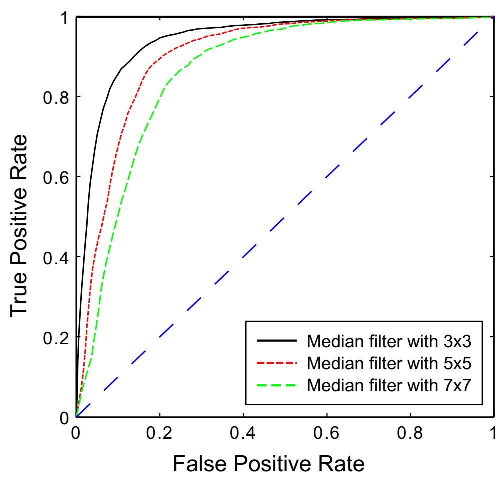

**Fig. 9.** The ROC curve of the noise detection method with different median filter settings.

that frames with sea, wavy water, or a textured background generally give high noise detection outputs.

## 5.2. Evaluation of picture-in-picture transformation detection

Out of 2010 query videos, 545 of them include picture-in-picture transformation. We obtained the scale and offset information of all picture-in-picture transformations by processing the ground-truth data used for generating query videos.

Our method has reached 86.79% of recall rate, where false alarm rate is 16.93%. Missed picture-in-picture transformations are generally caused by complex transformations, i.e., (8)–(10) in Table 1.

## 5.3. Evaluation of shot-boundary detection algorithm

We selected a set of shot-boundary detection algorithms from the literature for a comparative evaluation of our method. The factors we considered for the selection of these algorithms are the ease of their implementation and the presence of distinct features to be used in comparison. The source codes for most of these algorithms are not available. For the algorithms with available source code, the frame-dropping transformation causes many false alarms with default settings. Therefore, we decided to create our own implementations in order have a consistent and fair evaluation of the algorithms. Since many design details are unspecified in the literature, we tried to find the optimum values for the parameters experimentally. The following algorithms are selected for our test:

i *Color histogram (CH):* Short transitions and hard-cuts can be detected by using simple color histogram-based methods. In this method, *RGB* and $L^*a^*b^*$ color spaces are quantized into 27 equal subspaces. The histogram $h_n$ of image $I_n$ is defined as:

$$h_n(b_1 \times b_2 \times b_3) = \frac{|\{p|p \in I_n(r,c)\}|}{h \times w} \tag{15}$$

where $p_r/b = b_1, p_g/b = b_2, p_b/b = b_3$ for *RGB* color space. If the histogram difference of two successive frames exceeds a threshold value, a shot boundary is found. Details of such an algorithm are provided in [7,8,11,9,10].

ii *Probabilistic block intensity (PBI):* Probabilistic block intensity is a statistical method based on the mean and standard deviation of the pixels in image regions. This technique is discussed in [10], and implemented in [30]. Although its tolerance to noise is a great advantage, this method tends to generate many false alarms. In our experiments, each frame is divided into 16 blocks.

iii *Edge tracking (ECR):* Edge change ratio based shot-boundary detection methods are discussed in [8,10]. The ratio of the edges that enter and exit between two successive frames are used to determine shot boundaries. Edge-based methods are less sensitive to illumination changes, and they give better results in gradual transitions.

iv *Local keypoint matching (KM):* Recognizing the objects and scenes throughout the video is the basic idea of the keypoint matching-based shot-boundary detection methods. The algorithm proposed in [2] matches the objects between consecutive frames, and determines if there is a shot boundary. We use scale invariant feature transform [31] and a simple matching algorithm for this purpose.

The algorithms selected for comparison cover most of the major techniques listed in [8,10,7].

Our tests with 50 query videos, which represents each transformation type with at least 4 videos, showed that fuzzy color histogram-based shot-boundary detection method can achieve higher accuracy values, while reducing false alarms. Table 3 gives the recall and precision values for the compared algorithms. F1 scores are also provided as a measure that considers both the precision and the recall rates.

It should be noted that the methods selected for comparison could perform much better for detecting shot-boundaries of videos on which none of the transformations listed in Table 1 are applied. Our test set consists of videos manipulated with these transformations. The challenge here is to detect all shots, including background and foreground videos for picture-in-picture transformations, without being affected by frame-dropping, noise, pattern insertion, strong re-encoding, etc.

Methods have different accuracy values depending on the transformation type. Fig. 10 shows the recall values of shot-boundary detection methods for 10 types of transformations. For most of the transformations, the proposed fuzzy color histogram-based method performs better than other techniques.

Transformations of T2, T8, T9, and T10, which include picture-in-picture transformation, are the most challenging ones. We increase the overall recall rate in these transformations from 48.74% (best among others) to 62.18% (Fuzzy CH). Our method also achieves a lower false alarm rate with a precision of 93.67%, whereas the precision values of the other methods could only reach up to 53.21%.

It would be expected that the proposed fuzzy color histogram-based method may have some drawbacks when the processed videos/frames are in grayscale. In order to evaluate the performance of the proposed method for this case, we converted the same 50 query videos into grayscale, and then run the SBD algorithm for these videos.

Experiments with grayscale videos show that the recall (70.8%) and the precision (75.8%) values were slightly affected by this change. Although our method is a color histogram-based technique, grayscale pixels can be defuzzified into the colors other than white/

**Table 3**
Experimental results of shot-boundary detection algorithms.

| Method | Recall | Precision | F1 |
|---|---|---|---|
| *RGB* CH | 0.6284 | 0.6862 | 0.6560 |
| $L^*a^*b^*$ CH | 0.5939 | 0.6624 | 0.6263 |
| PBI | 0.3218 | 0.0225 | 0.0421 |
| ECR | 0.5862 | 0.3542 | 0.4416 |
| KM | 0.4789 | 0.4496 | 0.4638 |
| Fuzzy CH | 0.7165 | 0.8348 | 0.7711 |

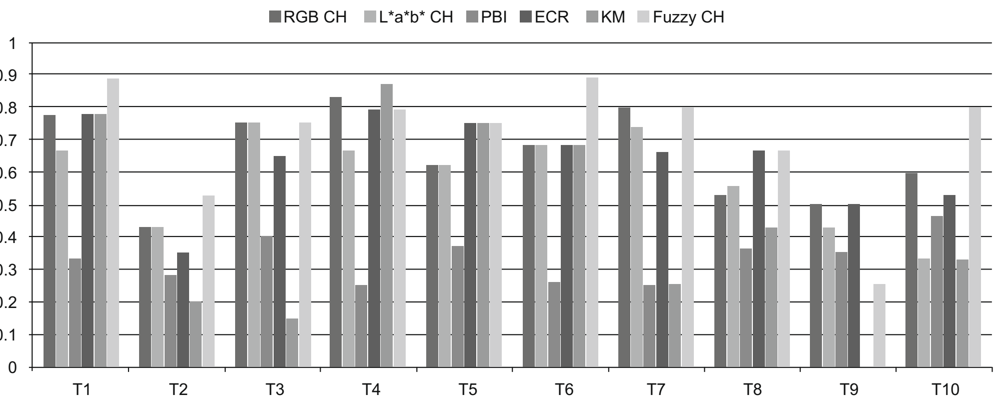

**Fig. 10.** Recall values of shot-boundary detection algorithms for different transformation types.

**Table 4**
The colors and their fuzzy correspondences.

| Color | $L^*$ | $a^*$ | $b^*$ | Fuzzy $L^*$ | Fuzzy $a^*$ | Fuzzy $b^*$ |
|---|---|---|---|---|---|---|
| Black | 0.00 | 0.00 | 0.00 | black | amiddle | bmiddle |
| Blue | 32.30 | 79.19 | −107.86 | black + gray | red | blue |
| Brown | 64.60 | 10.22 | 69.09 | gray | amiddle | yellowish |
| Cyan | 91.11 | −48.09 | −14.13 | white | greenish | bmiddle |
| Magenta | 60.32 | 98.24 | −60.83 | gray | red | bluish |
| Lime | 87.74 | −86.18 | 83.18 | white | green | yellow |
| Gray | 76.19 | 0.00 | 0.00 | gray | amiddle | bmiddle |
| Maroon | 39.03 | 63.65 | 53.41 | gray | reddish | yellowish |
| Navy | 22.38 | 62.93 | −85.72 | black | reddish | blue + bluish |
| Green | 66.44 | −68.49 | 66.10 | gray | green | yellow + yellowish |
| Olive | 73.92 | −17.13 | 75.08 | gray + white | greenish | yellow |
| Orange | 83.91 | 3.43 | 82.63 | white | amiddle | yellow |
| Pink | 92.07 | 11.20 | 1.05 | white | reddish | bmiddle |
| Purple | 44.66 | 78.07 | −48.34 | gray | red | bluish |
| Red | 53.24 | 80.09 | 67.20 | gray | red | yellow + yellowish |
| Silver | 89.53 | 0.00 | 0.00 | white | amiddle | bmiddle |
| Teal | 69.13 | −38.22 | −11.23 | gray + white | greenish | bmiddle |
| Violet | 50.46 | 89.85 | −77.24 | gray | green | blue |
| White | 100.00 | 0.00 | 0.00 | white | amiddle | bmiddle |
| Yellow | 97.14 | −21.55 | 94.48 | white | greenish | yellow |

```
If (L is black) and (a is amiddle) and (b is bmiddle) then (fuzzyhist is black)
If (L is black) and (b is blueish) then (fuzzyhist is blue)
If (L is grey) and (a is NOT green) and (b is blue) then (fuzzyhist is blue)
If (L is white) and (a is amiddle) and (b is blueish) then (fuzzyhist is blue)
If (L is white) and (a is greenish) and (b is blueish) then (fuzzyhist is blue)
If (L is black) and (a is reddish) and (b is blue) then (fuzzyhist is navy)
If (L is grey) and (a is red) and (b is NOT blue) then (fuzzyhist is red)
If (L is grey) and (a is reddish) and (b is bmiddle) then (fuzzyhist is red)
If (L is black) and (a is reddish) and (b is yellowish) then (fuzzyhist is red)
If (L is grey) and (a is reddish) and (b is yellow) then (fuzzyhist is yellow)
If (L is white) and (a is amiddle) and (b is yellow) then (fuzzyhist is yellow)
If (L is white) and (a is greenish) and (b is yellow) then (fuzzyhist is yellow)
If (L is grey) and (a is reddish) and (b is blueish) then (fuzzyhist is magenta)
If (L is white) and (a is reddish) and (b is blueish) then (fuzzyhist is magenta)
If (L is grey) and (a is amiddle) and (b is yellowish) then (fuzzyhist is brown)
If (L is white) and (a is reddish) and (b is yellow) then (fuzzyhist is brown)
If (L is grey) and (a is amiddle) and (b is bmiddle) then (fuzzyhist is grey)
If (L is grey) and (a is greenish) and (b is yellow) then (fuzzyhist is green)
If (L is white) and (a is green) and (b is yellowish) then (fuzzyhist is green)
If (L is grey) and (a is greenish) and (b is bmiddle) then (fuzzyhist is teal)
If (L is grey) and (a is green) and (b is blue) then (fuzzyhist is violet)
If (L is white) and (a is red) and (b is yellow) then (fuzzyhist is orange)
If (L is white) and (a is reddish) and (b is bmiddle) then (fuzzyhist is pink)
If (L is white) and (a is amiddle) and (b is bmiddle) then (fuzzyhist is white)
If (L is white) and (a is greenish) and (b is bmiddle) then (fuzzyhist is cyan)
If (L is white) and (a is green) and (b is bluish) then (fuzzyhist is cyan)
```

**Fig. 11.** Fuzzy rules.

gray/black, depending on the results of 26 fuzzy rules. This property provides enough flexibility for the method to detect shot-boundaries of grayscale videos. We also observed that the increase in false-alarm rate was mostly because of the shot-boundaries detected close to the beginning of gradual transitions. Nevertheless, the proposed method still outperforms other techniques.

## 6. Conclusions and future work

We propose a fuzzy color histogram-based shot-boundary detection method for the videos where heavy transformations (such as cam-cording, insertions of patterns, strong re-encoding) occur. In addition to detecting shot-boundaries using fuzzy color histogram, we extract a mask for still regions and the window of picture-in-picture transformation. Experimental results show that the proposed method effectively detects shot boundaries with a small false alarm rate as compared to the state-of-the-art shot-boundary detection algorithms.

As a future work we will use the detected shot boundaries, masks of still regions, picture-in-picture window boundaries, and the fuzzy color histogram method in our content-based copy detection system.

## Appendix A. Experiments in $L^*a^*b^*$ color space

We have selected popular colors, and experimented with their values in $L^*a^*b^*$ color space. $L^*a^*b^*$ is commonly preferred over RGB or HSV color spaces, because it is one of the perceptually uniform color spaces which approximates the way that human perceive color. In $L^*a^*b^*$ color space, $L^*$ represents luminance, $a^*$ represents greenness–redness, and $b^*$ represents blueness–yellowness.

$a^*$ and $b^*$ components have more weights than $L^*$ component. Therefore the fuzzy linking method in [4] subdivides $L^*$ into 3 (dark, dim, and bright), $a^*$ into 5 (green, greenish, middle, reddish, and red), and $b^*$ into 5 (blue, bluish, middle, yellow, and yellowish) regions.

Range of $L^*a^*b^*$ color space is important for the fuzzy membership functions. $L^*$ coordinate ranges from 0 to 100. The possible range of $a^*$ and $b^*$ coordinates depends on the color space that one is converting from. When converting from RGB, $a^*$ coordinate range is $[-86.1813, 98.2352]$, and $b^*$ coordinate range is $[-107.8617, 94.4758]$.

We have selected 20 colors from List of Colors [32]. Table 4 shows $L^*, a^*, b^*$ values, as well as their fuzzy correspondences for each color.

## Appendix B. Fuzzy rules

Twenty-six fuzzy rules of the fuzzy inference system are listed in Fig. 11. These rules are generated according to the fuzzy correspondences of output colors in Table 4.

## References

[1] G. Boccignone, A. Chianese, V. Moscato, A. Picariello, Foveated shot detection for video segmentation, IEEE Transactions on Circuits and Systems for Video Technology 15 (3) (2005) 365–377.
[2] C. Huang, H. Lee, C. Chen, Shot change detection via local keypoint matching, IEEE Transactions on Multimedia 10 (6) (2008) 1097–1108.
[3] C. Grana, G. Tardini, R. Cucchiara, MPEG-7 compliant shot detection in sport videos, in: Proceedings of the Seventh IEEE International Symposium on Multimedia, 2005, pp. 1–8.
[4] C. Cai, K. Lam, Z. Tan, An efficient scene-break detection method based on linear prediction with Bayesian cost functions, IEEE Transactions on Circuits and Systems for Video Technology 18 (9) (2008) 1318–1323.
[5] J. Yuan, H. Wang, L. Xiao, W. Zheng, J. Li, F. Lin, B. Zhang, A formal study of shot boundary detection, IEEE Transactions on Circuits and Systems for Video Technology 17 (2) (2007) 168–186.
[6] B. Truong, C. Dorai, S. Venkatesh, New enhancements to cut, fade, and dissolve detection processes in video segmentation, in: Proceedings of the Eighth ACM International Conference on Multimedia (MULTIMEDIA '00), 2000, pp. 219–227.
[7] C. Cotsaces, N. Nikolaidis, I. Pitas, Video shot detection and condensed representation: a review, IEEE Signal Processing Magazine 23 (2) (2006) 28–37.
[8] R. Lienhart, Comparison of automatic shot boundary detection algorithms, in: Proceedings of SPIE 3656, 1999, pp. 290–301.
[9] A. Dailianas, R.B. Allen, P. England, Comparison of automatic video segmentation algorithms, in: Proceedings of SPIE, Integration Issues in Large Commercial Media Delivery Systems 2615, 1995, pp. 2–16.
[10] J.S. Boreczky, L.A. Rowe, Comparison of video shot boundary detection techniques, Journal of Electronic Imaging 5 (1996) 122–128.
[11] T. Danisman, A. Alpkocak, Dokuz Eylül University Video Shot Boundary Detection at TRECVID 2006, in: Proceedings of the TREC Video Retrieval Evaluation (TRECVID), <http://www.nlpir.nist.gov/projects/tvpubs/tv6.papers/dokuz.pdf>, 2006.
[12] L. Zadeh, Fuzzy sets, Information Control 8 (1965) 338–353.
[13] K. Konstantinidis, A. Gasteratos, I. Andreadis, Image retrieval based on fuzzy color histogram processing, Optics Communications 248 (4–6) (2005) 375–386.
[14] J. Han, K.-K. Ma, Fuzzy color histogram and its use in color image retrieval, IEEE Transactions on Image Processing 11 (8) (2002) 944–952.
[15] F. Chung, B. Fung, Fuzzy color quantization and its application to scene change detection, in: Proceedings of the Fifth ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR '03), 2003, pp. 157–162.
[16] H. Fang, J. Jiang, Y. Feng, A fuzzy logic approach for detection of video shot boundaries, Pattern Recognition 39 (11) (2006) 2092–2100.
[17] R. Jadon, S. Chaudhury, K. Biswas, A fuzzy theoretic approach for video segmentation using syntactic features, Pattern Recognition Letters 22 (13) (2001) 1359–1369.
[18] S. Das, S. Sural, A. Majumdar, Detection of hard cuts and gradual transitions from video using fuzzy logic, International Journal of Artificial Intelligence and Soft Computing 1 (1) (2008) 77–98.
[19] A. Llorente, S. Zagorac, S. Little, R. Hu, A. Kumar, S. Shaik, X. Ma, S. Rger, Semantic video annotation using background knowledge and similarity-based video retrieval, in: Proceedings of the TREC Video Retrieval Evaluation (TRECVID), <http://www.nlpir.nist.gov/projects/tvpubs/tv8.papers/mmis.pdf>, 2008.
[20] D. Pye, N. Hollinghurst, T. Mills, K. Wood, Audio–visual segmentation for content-based retrieval, in: Proceedings of the Fifth International Conference on Spoken Language Processing, 1998.
[21] M. Douze, A. Gaidon, H. Jegou, M. Marszalek, C. Schmid, INRIA-LEARs video copy detection system, in: Proceedings of the TREC Video Retrieval Evaluation (TRECVID), <http://www.nlpir.nist.gov/projects/tvpubs/tv8.papers/inria-lear.pdf>, 2008.
[22] O. Kucuktunc, M. Bastan, U. Gudukbay, O. Ulusoy, Bilkent University Multimedia Database Group at TRECVID 2008, in: Proceedings of the TREC Video Retrieval Evaluation (TRECVID), <http://www.nlpir.nist.gov/projects/tvpubs/tv8.papers/bilmdg.pdf>, 2008.
[23] A.F. Smeaton, P. Over, W. Kraaij, Evaluation campaigns and TRECVID, in: Proceedings of the Eighth ACM International Workshop on Multimedia Information Retrieval (MIR '06), ACM, New York, NY, USA, 2006, pp. 321–330.
[24] Content-based copy detection, Guidelines for the TRECVID 2008 Evaluation, <http://www.nlpir.nist.gov/projects/tv2008/#4.5>, 2009.
[25] Final List of Transformations, <http://www.nlpir.nist.gov/projects/tv2008/active/copy.detection/final.cbcd.video.transformations.pdf>, 2008.
[26] O.B. Orhan, J. Liu, J. Hochreiter, J. Poock, Q. Chen, A. Chabra, M. Shah, University of Central Florida at TRECVID 2008 Content Based Copy Detection and Surveillance Event Detection, in: Proceedings of the TREC Video Retrieval Evaluation (TRECVID), <http://www.nlpir.nist.gov/projects/tvpubs/tv8.papers/ucf.pdf>, 2008.
[27] D.E. Knuth, Fundamental Algorithms of the Art of Computer Programming, second ed., vol. 2, Addison-Wesley, Reading, MA, 1973.
[28] B. Welford, Note on a method for calculating corrected sums of squares and products, Technometrics 4 (1962) 419–420.
[29] R.O. Duda, P.E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Communications of ACM 15 (1) (1972) 11–15.
[30] R. Kasturi, R. Jain, Dynamic vision, in: R. Kasturi, R. Jain (Eds.), Computer Vision: Principles, IEEE Computer Society Press, Washington, DC, 1991, pp. 469–480.
[31] D.G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2004) 91–110.
[32] List of Colors, Wikipedia, <http://en.wikipedia.org/wiki/list_of_colors>, 2009.