JOURNAL OF PATTERN RECOGNITION RESEARCH 7 (2012) 56-71

Received March 29, 2009. Revised Jujy 24, 2011. Accepted September 14, 2011.

Illustration Extraction from Video Streams*

Ji-Wen Chio and Shu-Yuan Chen *cschen@saturn.yzu.edu.tw*

Department of Computer Science and Engineering,
Yuan Ze University, Chung-Li, Taoyuan, Taiwan.

Abstract

Teachers usually illustrate major pedagogical concepts with graphics and/or images and/or tables and, in doing so, take a considerable amount of time in explanation. Despite the availability of many text extraction methods, they are limited such as when background is noisy, degraded, multicolored or containing graphics. This work presents an illustration extraction method for video streams to increasing learning efficiency among students. The proposed method separates the foreground from video streams. Above limitations of text extraction methods are solved by using contexture between image sequences in video streams and extracting illustrations from video streams. Two stages of the proposed method are training and extraction. Shot boundaries are detected to resample a video stream in a non-redundancy training set during the training phase. A background map together with a set of colors used frequently is then constructed based on the training set, subsequently used for illustration extraction. Next, a foreground map is generated according to the background map and frequent color set for each frame of the video stream during the extraction phase. Finally, illustrations are extracted by region labeling and geometric verification. Experimental results demonstrate the feasibility of the proposed method.

Keywords: Content-based image retrieval, illustration retrieval, illustration extraction, e-Learning, video streaming, background map generation, foreground extraction.

WWW.JPRR.ORG

1. Introduction

Advances in information technology, media acquisition/storage devices, and multimedia compression standards have led to the exponential growth of digitalized video data available to users worldwide. environments for educational purposes are easily constructed to record class scenarios in a video stream format. Although distance learning students can review course contents by watching an video stream, a considerable amount of time is spent in watching video streams to identify the important parts, ultimately degrading the incentive of students to learn. Major concepts are often demonstrated with graphic illustrations, graphics, images and tables. Therefore, illustration extraction from video streaming can help students to focus on the important parts (*i.e.* illustrations), ultimately increasing their learning efficiency. This workaddresses this topic.

Illustration extraction initially involves foreground segmentation. Many methods have been developed to exact text foreground [6]. These methods are typically categorized into connected component (CC)-based [4, 7, 10] or texture-based [1, 5, 8] approaches. However, such methods are limited when the background is noisy, degraded, multicolored or containing many graphics, which often occurs in video streaming. This work attempts to resolve this problem by using temporal contexture in video streams, thus facilitating illustration extraction from video streams.

The proposed method consists of two phases: training and extraction. Figure 1 displays the flowchart of the proposed method.

∗ This work was partially supported by the National Science Council of Taiwan, under Grants NSC-99-2221-E-155-072.

© 2012 JPRR. All rights reserved. Permissions to make digital or hard copies of all or part of this work for personal or classroom use may be granted by JPRR provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or to republish, requires a fee and/or special permission from JPRR.

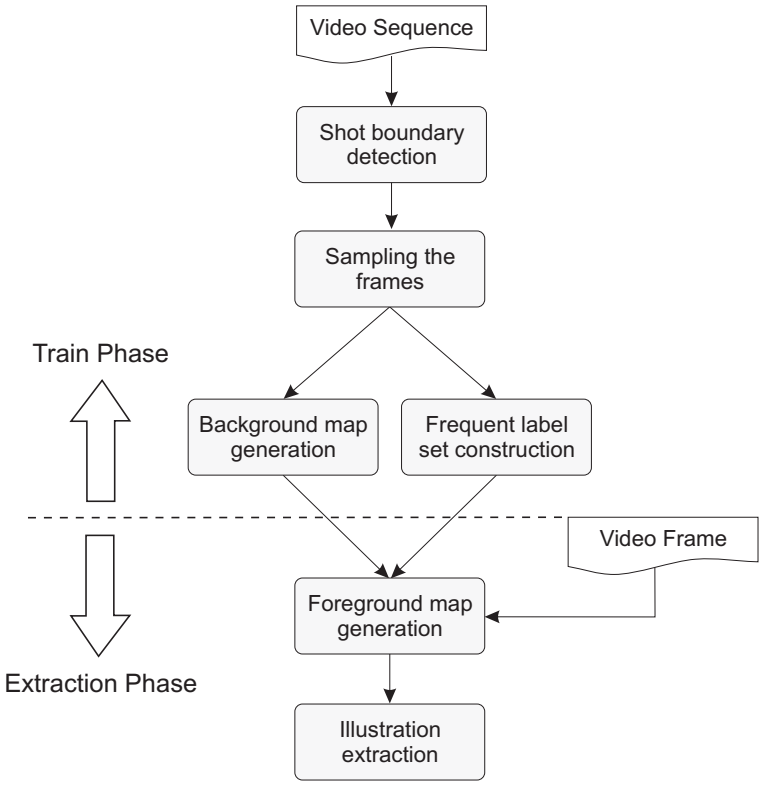


Fig. 1: Flowchart of the proposed method

During the training phase, shot boundary detection is performed to resample a video stream, resulting in a non-redundancy training set. A background map is then generated together with a set of colors used frequently from the training set, for later use in illustration extraction. During the extraction phase, a foreground map is generated for each frame of the video stream according to the background map and frequent color set. Based on the foreground map, illustrations are easily extracted by region labeling and geometric verification.

A shot refers to a sequence of frames captured by a single camera in a single continuous action in time and space. Many approaches have been developed to determine shot boundaries [2, 14]. Liu and Chen [11] performed principal component analysis (PCA) to locate the boundaries in a video. A novel shot boundary refinement algorithm [12], which works independently of the contents and transition length, has a high accuracy under most transition types. Yong et al. [16] developed a shot boundary detection with an automatic threshold. The method of Miene et al. [13] uses three different features to evaluate shot boundaries.

Background image generation is essential for target tracking in the application of traffic monitoring and video surveillance [2, 14]. The common operator for background update in real-time generates background first and then averages the current image and the estimated background with different weights to update the background [17]. More advanced methods focus on learning Gaussian mixture to model background images in video sequence [8, 18]. Some approaches are based on progressive updating [15]. While assuming that movement is smoothness and the image sequence follows non-stationary temporal distributions, such methods fail to handle large movements encountered in e-Learning video streams.

2. Preprocessing

A video stream V_T of length T consists of a set of 2D image frames $f(x, y, t)$, *i.e.*

$$V_T = \{f(x, y, t) \mid t = 1, 2, \cdots, T\} \tag{1}$$

Figure 2 displays examples of frames for the five video streams used in this work. Table 1 lists their details. This section describes the preprocessing of each image frame $f(x, y, t)$ which is essential to the subsequent foreground/background separation.

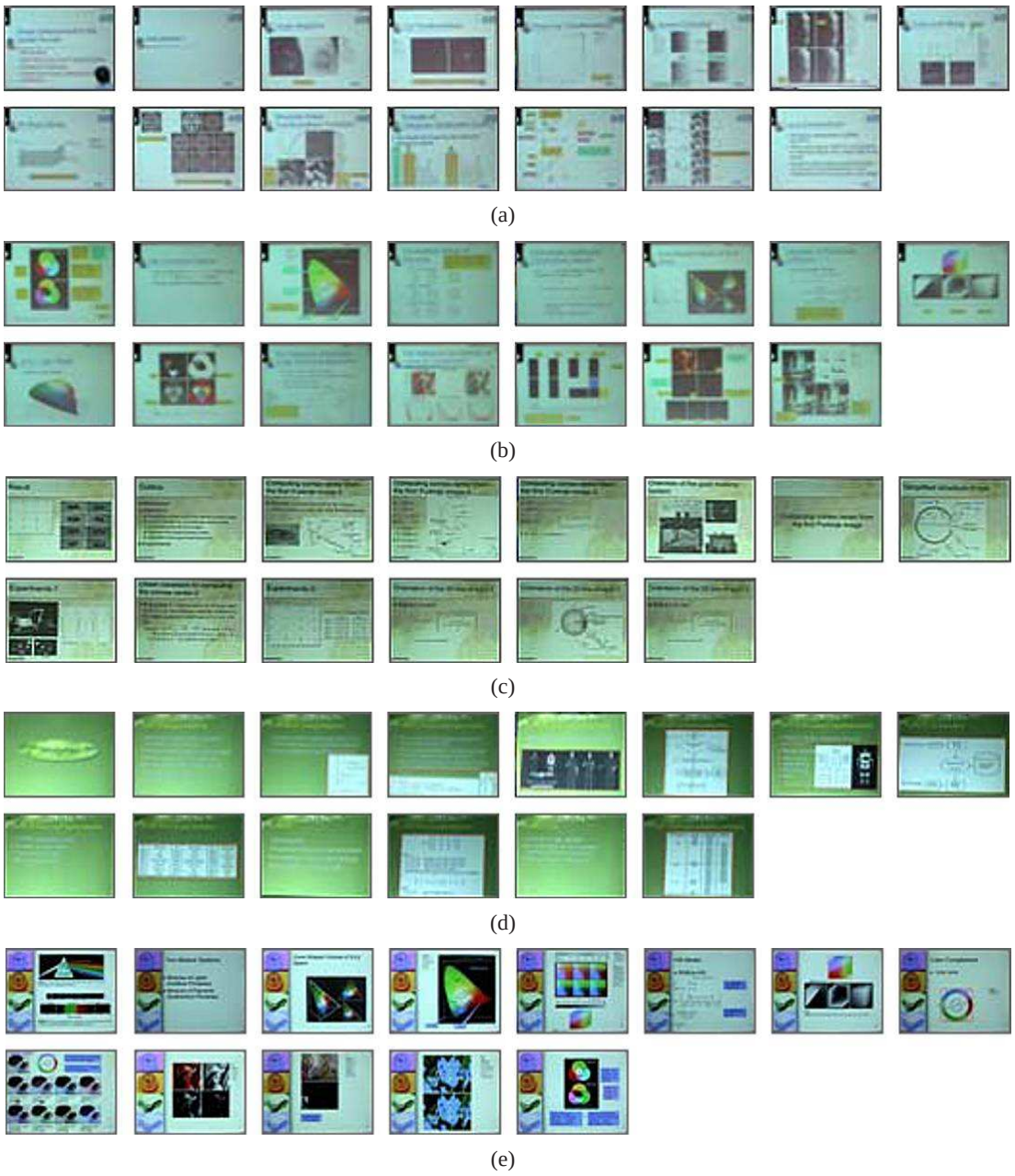


Fig. 2: Examples of frames for each video stream. (a) through (e) for videos 1 through 5.

Table 1: Summaries of video streams.

	Total time	Shots	Time interval	Sampling number
Video 1	01:30	25	5	460
Video 2	02:42	30	5	560
Video 3	00:19	14	5	260
Video 4	00:27	14	5	260
Video 5	00:13	13	5	220

2.1 Color quantization

Figure 3 shows the HSV color model adopted to quantize each image frame.

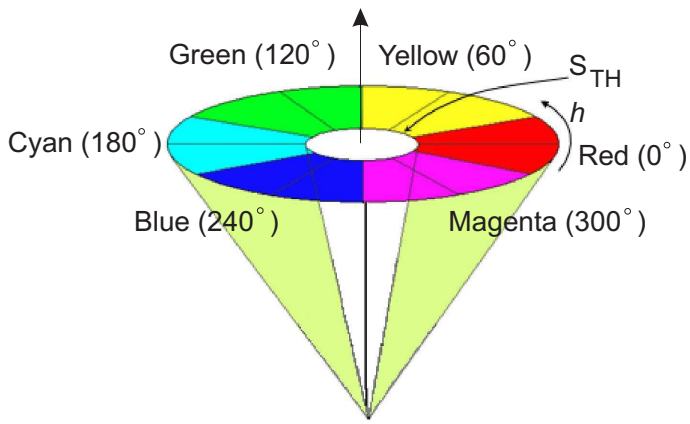


Fig. 3: HSV color space.

First, the red, green, and blue color values of each pixel in the image frame are transformed into HSV color values by using the following equations [3].

$$
\begin{aligned}
m &= \max(r, g, b) \\
n &= \min(r, g, b) \\
\nu &= (r + g + b)/3 \\
s &= \begin{cases} 0.0 & \text{if} \quad m = 0.0, \\ (m - n)/m & \text{if} \quad m \neq 0.0 \end{cases} \\
h &= \begin{cases} -1 & \text{if} \quad s = 0, \\ 60 \times (g - b)/(m - n) & \text{if} \quad m = r \\ 60 \times (b - r)/(m - n) + 120 & \text{if} \quad m = g \\ 60 \times (r - g)/(m - n) + 240 & \text{if} \quad m = b \end{cases}
\end{aligned}
\tag{2}
$$

where r, g, b denote the RGB color values [0, 255] of a pixel; m and n denote the maximum and minimum of r, g, b; and h, s, ν denote the hue [0, 360], saturation [0, 1] and value [0, 255] of the pixel, respectively. Notably, each image frame $f(x, y, t)$ has three corresponding frames $h(x, y, t)$, $s(x, y, t)$, and $\nu(x, y, t)$ related to h, s, ν values, respectively.

The quantized color of each pixel is determined based on its h, s and v values. Pixels with s values higher than s_{TH} are quantized as color labels, $L = 1, 2, \cdots, 60$, according to the h value.

Pixels with s values lower than s_{TH} are quantized as gray labels, $L = 61, \cdots, L_H$, according to the ν value. In this study, L_H is set to 65 and s_{TH} is set to 0.65 empirically.

Figure 4 displays the flowchart of quantized color determination. Following color quantization, each image frame $f(x, y, t)$ has the corresponding labeled frames $H(x, y, t)$, in which each pixel is labeled by $L = 1, 2, \cdots, L_H$.

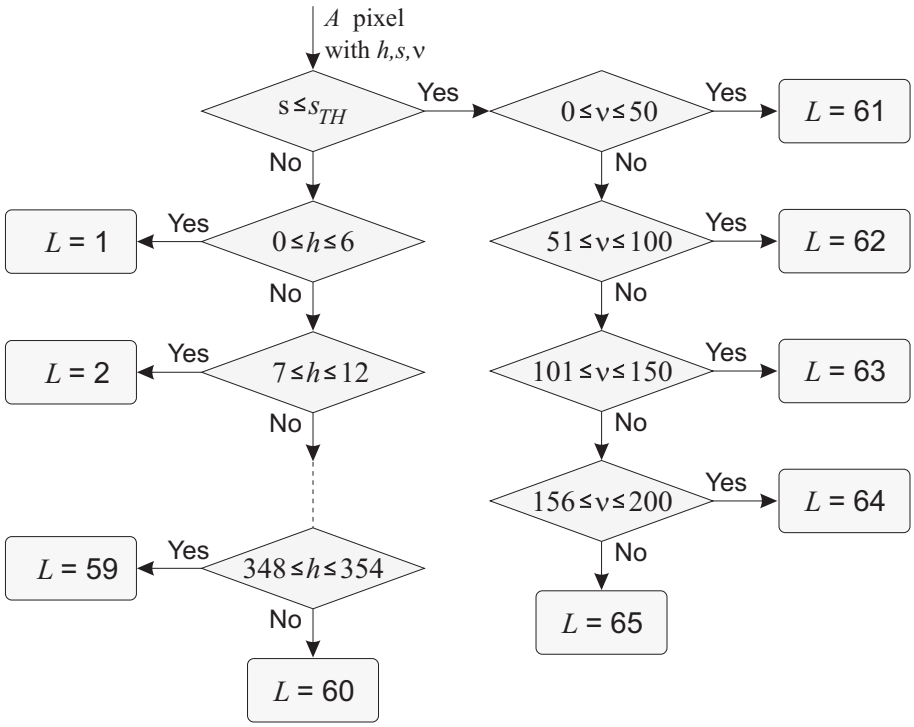


Fig. 4: Flowchart of color quantization.

2.2 Region of interest detection

While slides may not occupy the entire image frame, the region containing the slide should be detected first to reduce processing time and increase extraction accuracy. Restated, the region of interest (ROI) must be located based on the slide boundary. This work attempts to solve this problem by using projection technology.

Horizontal and vertical projections [3] can indicate the location of ROI. Since the methods for both directions resemble each other, this work describes only the method for the horizontal case. The horizontal and vertical projections $p_h(x, t)$ and $p_\nu(y, t)$ can be expressed as

$$\begin{aligned} p_h(y, t) &= \sum_x \nu(x, y, t) \\ p_\nu(x, t) &= \sum_y \nu(x, y, t) \end{aligned} \tag{3}$$

Let W and H denote the width and height of the image frame. The highest and lowest rows with horizontal projections $p_h(y, t)/W$ exceeding a threshold value TH_p are assumed to be the horizontal boundary of ROI. In this work, TH_p is set to two thirds empirically. The same method can be used in the vertical detection. In this manner, the boundary of ROI for each frame is located.

3. Training Phase

During the training phase, shot boundaries are detected to resample a video stream, resulting in a non-redundancy training set. A background map is then generated together with a set of colors used frequently from the training set, which is subsequently used for illustration extraction, as described in Section 4.

3.1 Shot boundary detection

Effectiveness of shot boundary detection depends on the appropriate selection of a similarity metric between two frames. This work adopts the features and the similarity metric for shot boundary detection as defined in [13]. Based on the three similarity metrics, three lists of shot boundaries are obtained, respectively, which are then combined to obtain a final list of shot boundaries.

3.1.1 Similarity metric between frames

The features for shot boundary detection include the mean of intensity values, $S^{(V)}(t)$, histogram of intensity values, $\mathbf{h}^{(V)}(t)$, and FFT of intensity values, $\mathbf{FFT}^{(V)}(t)$. The former two are defined based on the following equations.

$$S^{(V)}(t) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\frac{\nu(x,y,t)}{M\times N} \tag{4}$$

$$\begin{aligned}\mathbf{h}^{(V)}(t) &= \left(h_i^{(V)}(t)\right)\\ h_i^{(V)}(t) &= \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} c_i^{(V)}(x,y,t)}{\sum_{i=1}^{L_\nu}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} c_i^{(V)}(x,y,t)}\\ c_i^{(V)}(x,y,t) &= \begin{cases}1 & \text{if} \quad \nu(x,y,t)=i\\ 0 & \text{otherwise}\end{cases}\end{aligned} \tag{5}$$

Applying FFT to the frame $\nu(x,y,t)$ allows us to obtain the corresponding Fourier transform $F^{(V)}(u,w,t) = R^{(V)}(u,w,t) + iI^{(V)}(u,w,t)$. $\mathbf{FFT}^{(V)}(t)$ is then defined by

$$\mathbf{FFT}^{(V)}(t) = \left(FFT_R^{(V)}(t), FFT_I^{(V)}(t)\right) = \left(\sum_{\omega_x=0}^{M-1}\sum_{\omega_y=0}^{N-1} R^{(V)}(\omega_x,\omega_y,t), \sum_{\omega_x=0}^{M-1}\sum_{\omega_y=0}^{N-1} I^{(V)}(\omega_x,\omega_y,t)\right) \tag{6}$$

Let two consecutive frames denote $f(x,y,t-1)$ and $f(x,y,t)$.

Three similarity metrics, $d_S^{(V)}(t-1,t), d_h^{(V)}(t-1,t)$, and $d_{FFT}^{(V)}(t-1,t)$, which correspond to the three features, $S^{(V)}(t)$, $\mathbf{h}^{(V)}(t)$, and $\mathbf{FFT}^{(V)}(t)$ are defined by the following equations.

$$d_S^{(V)}(t-1,t) = \left|S^{(V)}(t) - S^{(V)}(t-1)\right| \tag{7}$$

$$d_h^{(V)}(t-1,t) = \sum_{i=0}^{255}\frac{\left(h_i^{(V)}(t) - h_i^{(V)}(t-1)\right)^2}{\max\left\{h_i^{(V)}(t), h_i^{(V)}(t-1)\right\}} \tag{8}$$

$$d_{FFT}^{(V)}(t-1,t) = \left|FFT_R^{(V)}(t) - FFT_R^{(V)}(t-1)\right| + \left|FFT_I^{(V)}(t) - FFT_I^{(V)}(t-1)\right| \quad (9)$$

where Eq.(8) is χ^2 distance.

3.1.2 Shot boundary detection and merging

For two consecutive frames $f(x,y,t-1)$ and $f(x,y,t)$ of the video streams, the three corresponding similarity metrics are calculated using Eqs. (7) through (9), respectively, which are subsequently used to obtain three lists of shot boundaries. For each list, a threshold is calculated automatically to determine whether the corresponding frame $f(x,y,t)$ is a shot boundary. In this work, the threshold value is determined adaptively. To calculate the adaptive threshold, the average and maximum of metric values, $d_{X(\text{avg})}^{(V)}$ and $d_{X(\text{max})}^{(V)}$, $X = S, h, FFT$ for each video stream should be obtained first. The threshold can then be defined using the following equation

$$\begin{aligned} TH_X^{(V)} &= d_{X(\text{max})}^{(V)} - \left(d_{X(\text{max})}^{(V)} - d_{X(\text{avr})}^{(V)}\right) \times TH_{\textit{percentage}} \\ d_{X(\text{max})}^{(V)} &= \max_{t=2,\cdots,T}\left(d_X^{(V)}(t-1,t)\right) \\ d_{X(\text{avr})}^{(V)} &= \frac{1}{T}\sum_{t=2}^{T} d_X^{(V)}(t-1,t) \\ X &= S, h, FFT \end{aligned} \quad (10)$$

where percentaage $TH_{\textit{percentage}}$ is set to 0.25 empirically in this work.

Three lists of shot boundaries, $LIST_S^{(V)}$, $LIST_h^{(V)}$, and $LIST_{FFT}^{(V)}$, are obtained as the ordering sets of frame numbers by

$$LIST_X^{(V)} = \left\{t \middle| d_X^{(V)}(t-1,t) \geq TH_X^{(V)}\right\}, X = S, h, FFT \quad (11)$$

However, the two shot boundaries should be merged if the time interval between the two consecutive shot boundaries is less than a threshold value TH_T. In this work, TH_T is set to 30 sec.

Similarly, the three lists of shot boundaries should be integrated to obtain the final list $LIST^{(V)}$ as

$$LIST^{(V)} = \left\{t_k \middle| t_k \in LIST_X^{(V)}, t_k - t_{k-1} \geq TH_T\right\}, X = S, h, FFT, k = 1, \cdots, n_b \quad (12)$$

where b_n denotes the total number of shot boundaries.

3.2 Re-sampling of a video stream

After the list of shot boundaries is obtained, the video stream V_T can be re-sampled into non-redundancy training set R_{nt} of image frames in the following way. Define a shot, $SHOT_k^{(V)}$, as a set of image frames as

$$SHOT_k^{(V)} = \left\{f(x,y,t) \middle| t_k \leq t \leq t_{k+1}, t_k \in LIST^{(V)}\right\}, k = 1, \cdots, n_b \quad (13)$$

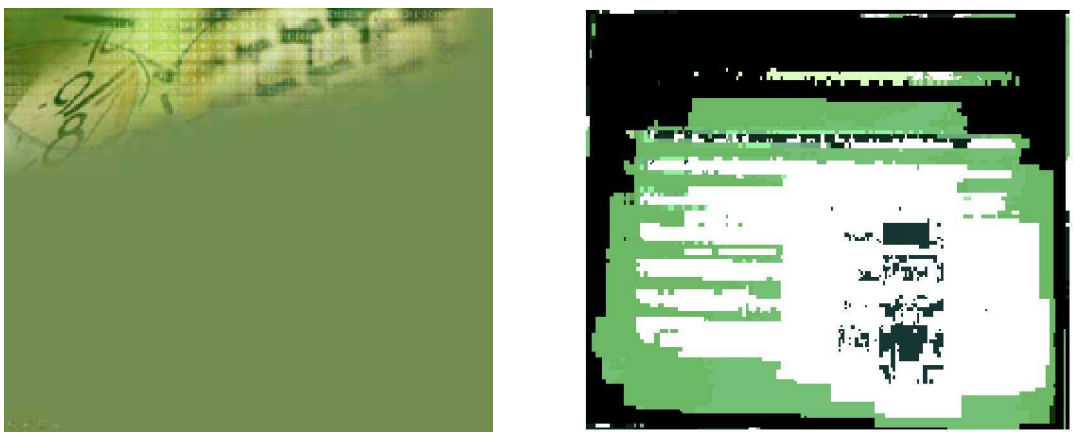
For each shot, a fixed number n_s of frames are re-sampled uniformly to obtain the training set $R_t^{(V)}$ of size $n_t = n_b \times n_s$ denoted by

$$R_t^{(V)} = \left\{ f(x, y, t) \middle| t = t'_1, \cdots, t'_{n_t} \right\} \tag{14}$$

where $t'_1, \cdots, t'_{n_t}$ denote the time indexes of the sampled frames. In this work, n_s is set to 20 empirically.

3.3 Generation of background map

This section describes how a background map is generated based on $L(x, y, t)$ for each image frame in the training set $R_t^{(V)}$. The main concept is to identify the label occurring frequently for each pixel in the image frames of a training set, which is subsequently used to identify the labels occurring frequently for the entire image frame as the background colors.



(a) (b)

Fig. 5: An example of background map generation for Video 4. (a) Slide background; (b) background map with foreground and background pixels marked by white and black, respectively.

The histogram of quantized labels, $h_i^{(L)}(x, y), i = 1, \cdots, L_H$, occurring at fixed coordinates (x, y), is first obtained by

$$\begin{aligned} h_i^{(L)}(x, y) &= \frac{\sum_{t \in R_t^{(V)}} c_i^{(L)}(x, y, t)}{n_t} \\ c_i^{(L)}(x, y, t) &= \begin{cases} 1 & \text{if} \quad H(x, y, t) = i \\ 0 & \text{otherwise} \end{cases}, \quad i = 1, \cdots, L_H \end{aligned} \tag{15}$$

The background map, $B(x, y)$, for the video stream V_T can then be defined by

$$c_i^{(L)}(x, y, t) = \begin{cases} 0 & \text{if} \quad \max_{i=1,\cdots,L_H} h_i^{(L)}(x, y) \geq 0.9 \\ k & \text{if} \quad 0.5 < \max_{i=1,\cdots,L_H} h_i^{(L)}(x, y) > 0.9, \; k = \arg \max_{i=1,\cdots,L_H} h_i^{(L)}(x, y) \\ -1 & \text{if} \quad \max_{i=1,\cdots,L_H} h_i^{(L)}(x, y) \leq 0.5 \end{cases} \tag{16}$$

Notably, $B(x, y)$ with values 0 and -1 indicate that the pixel of coordinates (x, y) belongs to background exactly and foreground possibly. For the other values rather than 0 and -1, this finding implies that the pixel of coordinates (x, y) belongs to uncertainty type; its type determination is deferred until extraction phase.

Figure 5 shows an example of background map generation. The pixels with $B(x, y) = 0, -1, k$ are marked by black, white and corresponding colors of labels k, respectively.

3.4 Construction of a frequent label set

The frequently occurring quantized labels in the training set can provide valuable information for foreground extraction since those labels may belong to the background of the video stream. The frequent label set is generated in the following way.

First, the histogram of quantized labels occurring in image frame of t, $hf_i^{(L)}(t), i = 1, \cdots, L_H$, is obtained by

$$hf_i^{(L)}(t) = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} c_i^{(L)}(x, y, t)}{M \times N}, \quad i = 1, \ldots, L_H, t \in R_t^{(V)} \tag{17}$$

where $c_i^{(L)}(x, y, t)$ is specified in Eq. (15). The candidate label set, $LC^{(L)}$, can then be defined by

$$\begin{aligned} LC^{(L)} &= \left\{ i \middle| hf_i^{(L)}(t) \geq TH_p \times m_f(t),\ i = 1, \ldots, L_H,\ t \in R_t^{(V)} \right\} \\ m_f(t) &= \max_{i=1,\cdots,L_H} hf_i^{(L)}(t) \end{aligned} \tag{18}$$

For each candidate label index i in $LC^{(L)}$, the histogram of quantized label i occurring in the training set, $hl_i^{(L)},\ i = 1, \cdots, L_H$, is obtained by

$$hf_i^{(L)}(t) = \frac{\sum_{t \in R_t^{(V)}} hf^{(L)}(t)}{n_t}, \quad i = 1, \ldots, L_H \tag{19}$$

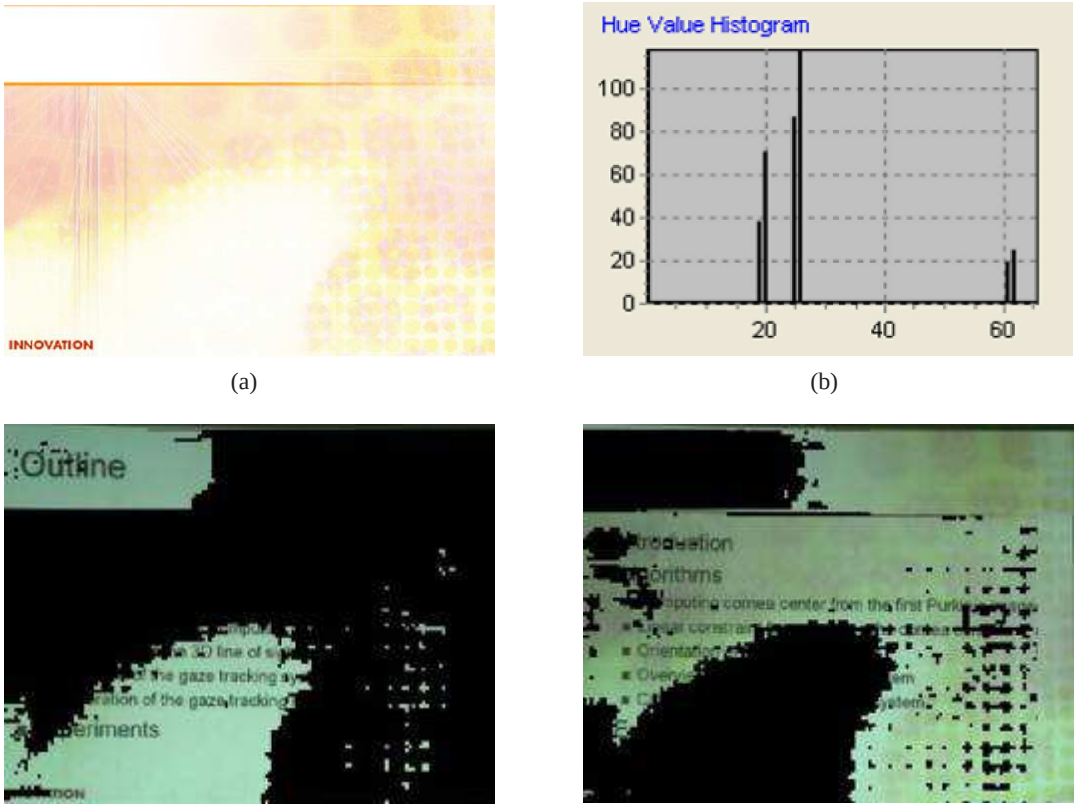
The frequent label set, $LF^{(L)}$ can then be defined by

$$\begin{aligned} LF^{(L)} &= \left\{ i \middle| \left| hf_i^{(L)}(x, y) - m_h \right| \leq TH_p \times m_h \right\} \\ m_h &= \max_{i=1,\cdots,L_H} hl_i^{(L)} \end{aligned} \tag{20}$$

Figure 6 displays OR an example of generating a frequent label set for Video 3. Figure 6(a) displays the slide background. Figure 6(b) displays the histogram of quantized label i, $hl_i^{(L)},\ i = 1, \cdots, L_H$, from which it is easily found that there are two frequent labels, textiti.e. labels 20 and 25. Figures 6(c) and 6(d) display those pixels with labels 20 and 25 in the frame of Video 3, which are marked black, respectively.

4. Extraction Phase

During the extraction phase, a foreground map is first generated for each frame within the video stream based on the background map and frequent label set, as described in Section 3. The illustration is easily extracted based on the foreground map.



(a) (b) (c) (d)

Fig. 6: An example of generating frequent label set for Video 3. (a) Slide background; (b) hue value histogram with frequent labels 20 and 25; (c) and (d) pixels with labels 20 and 25 are marked by black; respectively.

4.1 Generation of foreground map

Foreground map can be generated for each frame within the video stream based on the background map and frequent label set as described in Section 3. In particular, the foreground map $F(x, y, t)$ for the image frame $f(x, y, t)$ is calculated as

$$F(x,y,t) = \begin{cases} 1 & \begin{array}{l} B(x,y) = k, L(x,y,t) \in LF^{(H)}, d(L(x,y,t),k) > TH_k \\ B(x,y) = k, L(x,y,t) \notin LF^{(H)} \\ B(x,y) = -1, L(x,y,t) \notin LF^{(H)} \end{array} \\ 0 & \begin{array}{l} B(x,y) = 0 \\ B(x,y) = k, L(x,y,t) \in LF^{(H)}, d(L(x,y,t),k) \leq TH_k \\ B(x,y) = -1, L(x,y,t) \in LF^{(H)} \end{array} \end{cases} \tag{21}$$

$$d(L(x,y,t),k) = \begin{cases} \left[L(x,y,t) - k + 60\right] \text{mod} 60 & 1 \leq L(x,y,t), k \leq 60 \\ 3 \times \left|L(x,y,t) - k\right| & 61 \leq L(x,y,t), k \leq 65 \\ TH_k + 1 & \text{otherwise} \end{cases}$$

where TH_k is set to 3 empirically in this work.

Figure 7 displays the flowchart of foreground map generation. Finally, the mean filter is implemented to screen out noises in the foreground map. Figure 8(a) shows an example of foreground map for Video 4.

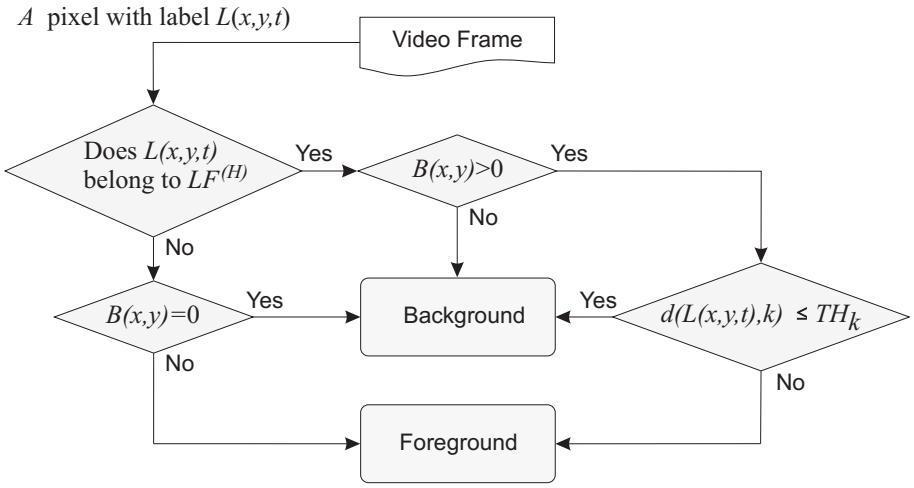


Fig. 7: Flowchart of foreground map generation.

4.2 Illustration extraction

Following generation of the foreground map, region labeling is then performed on the foreground map to identify connected components as illustration candidates. Those candidates satisfying the following constraint are considered illustrations. The bounding box containing the candidate should have a size exceeding a threshold value TH_s. In this work, TH_s is set to $3 \times (M \times N)/100$ empirically. Figure 8 shows an example of illustration extraction for Video 4.

(a) (b)

Fig. 8: Illustration extraction for Video 4. (a) Foreground map; (b) illustration extraction.

5. Experimental Results

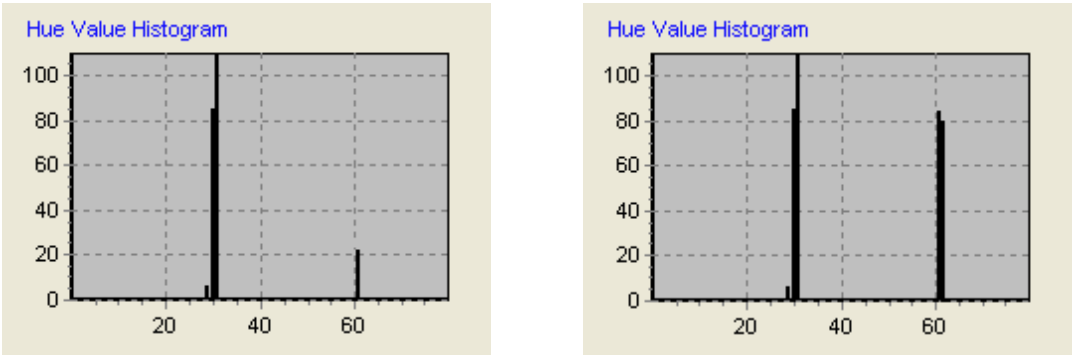
The proposed method was implemented on a personal computer with a single AMD64-3000 CPU and 512 Megabytes DDRAM. The operating system is Microsoft Windows XP Server Chinese version Service Pack 2. The program was developed in the C++ language and compiled under Borland C++ Builder version 6.

The performance of shot boundary detection is evaluated first. Table 2 lists the boundaries detected by the proposed method using the three features, as specified by Eqs. (7) through (9).

Table 2: Results of shot boundary detection.

	Feature of shot detection		
	S	S + h	S + h + FFT
Video 1	15	23	25
Video 2	22	30	30
Video 3	7	10	14
Video 4	10	13	14
Video 5	11	11	13

The detected boundaries are close to the ground truth through subject observation. Figure 9 shows shot boundary detection is necessary. If shot detection is not performed, some scenes of video steams may have a long duration that causes error generation of frequent label set (Fig. 9(b)). Figures 9 (a) and 9(b) display the frequent label sets with and w/o shot detection.



(a) (b)

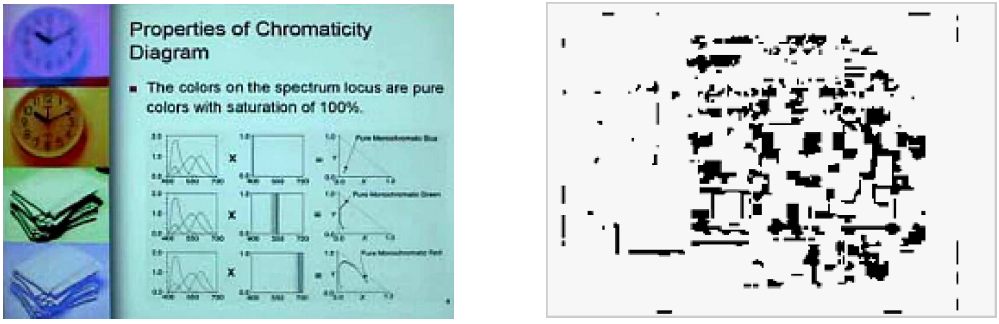
Fig. 9: Comparison of frequent label sets for Video 5. (a) and (b) frequent label sets with and w/o shot detection.

The performance of illustration extraction is then evaluated by precision and recall as defined below. Let TP denote the number of frames, from which, authentic illustrations are extracted and FN denote the number of frames, from which, authentic illustrations are undetected. Also, Let TN denote the number of frames in which no illustrations exist and no illustrations are extracted and FP denote the number of frames in which no illustrations exist but illustrations are extracted. The precision and recall are then defined by Precision=$TP/(TP + FP)$ and Recall=$TP/(TP + FN)$. The proposed method is then applied to the five video streams individually. Table 3 lists the respective precisions and recalls.

Table 3: Precision and recall.

	Precision	Recall	TP	FP	FN
Video 1	1	0.93	72	0	5
Video 2	1	0.89	54	0	6
Video 3	1	0.95	75	0	4
Video 4	1	1	51	0	0
Video 5	1	0.93	59	0	4

Figure 10 displays an example of missing error for Video 5, *i.e.* the original frame contains illustrations (Fig. 10(a); however, none are extracted (Fig. 10(b)).



(a) (b)

Fig. 10: An example of missing error for Video 5. (a) Original frame; (b) results of foreground extraction. The authentic graphics are not extracted.

6. Conclusions

Two phases of the proposed method are training and extraction. During the training phase, shot boundaries are detected first to resample a video stream into a non-redundancy training set. A background map together with a set of labels used frequently is then constructed from the training set, which is subsequently used for illustration extraction. A foreground map is first generated based on the background map and frequent label set for each frame of the video stream during the extraction phase. Finally, illustrations can be extracted based on region labeling and geometric verification. Feasibility of the proposed method is confirmed based on various experiments. Future works can be directed to incorporate illustration recognition into the proposed illustration extraction method to identify the types illustrations such as graphics, images or tables.

Acknowledgments

The authors would like to thank the anonymous reviewers for the valuable comments on the earlier version of this manuscript.

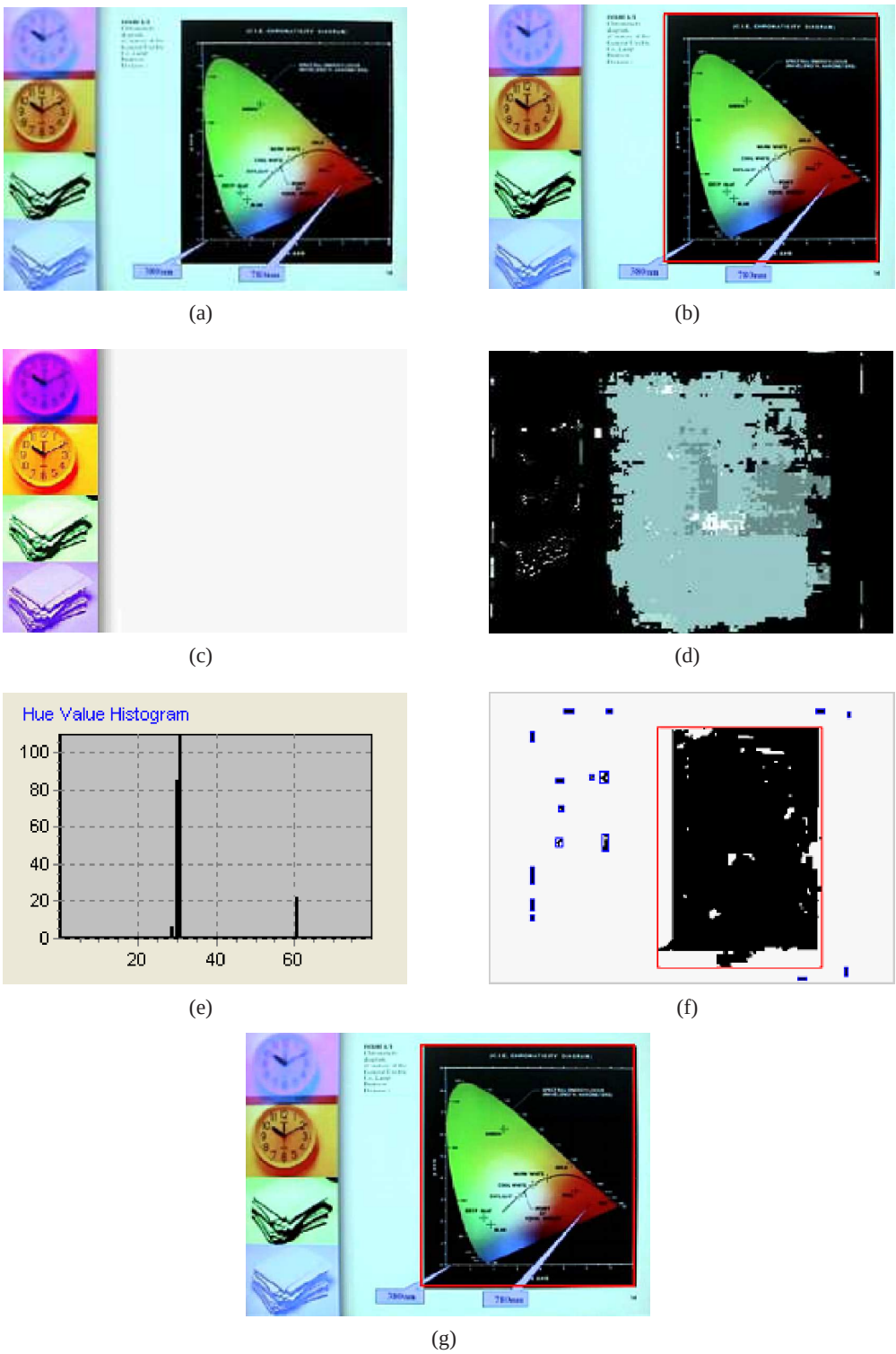
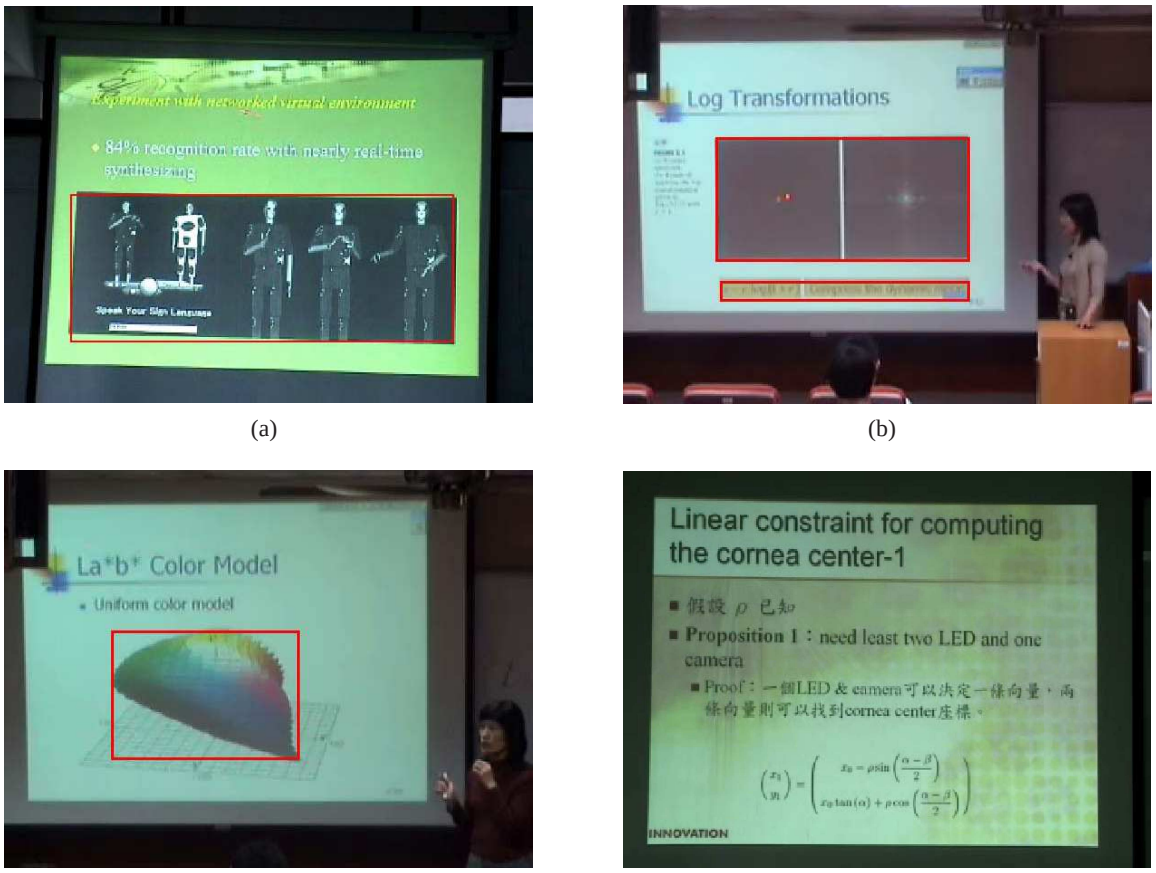


Fig. 11: An example of legend extraction for Video 5. (a) Original frame; (b) detected region of interest; (c) slide background; (d) background map with background pixels marked by black; (e) frequent label set; (f) foreground map with foreground pixels marked by black; (g) illustration extracted.



(a) (b)

(c) (d)

Fig. 12: Some results of illustration extraction.

References

[1] S. Chuai-Aree, C. Lursinsap, P. Sophatsathit, and S. Siripant, "Fuzzy C-mean: A statistical feature classification of text and image segmentation method", International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 9, no. 6, pp. 661-671, 2001.

[2] R.T. Colllin, A.J. Lipton and T. Kanade, "Introduction to the special section on video surveillance", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 745-746, 2000.

[3] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Prentice Hall, 2002.

[4] A.K. Jain and B. Yu, "Automatic text location in images and video frames", *Pattern Recognition*, vol. 31, no.12, pp. 2055-2076, 1998.

[5] A.K. Jain and Y. Zhong, "Page segmentation using texture analysis", *Pattern Recognition*, vol. 29, no. 5, pp. 743-770, 1996.

[6] K. Jung, K.I. Kim, A.K. Jain, "Text information extraction in images and video: a survey", *Pattern Recognition*, vol. 37, pp. 977-997, 2004.

[7] E.Y. Kim, K .Jung, K.Y. Jeong, and H.J. Kim, "Automatic text region extraction using cluster-based templates", in *Proc. Advance in Pattern Recognition and Digital Techniques*, pp. 421-421, 2000.

[8] K.I. Kim, K. Jung, and J.H. Kim, "Texture-based approach for text detection in image using support vector machine and continuously adaptive mean shift algorithm", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1631-1639, 2003.

[9] D.S. Lee, "Effective Gaussian mixture learning for video background subtraction", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 827-832, 2005.

[10] H. Li, D. Doermann, and O. Kia, "Automatic text detection and tracking in digital video", *IEEE Trans. Image Processing*, vol. 9, no. 1, pp. 147-156, 2000.

[11] X. Liu and T. Chen, “Shot boundary detection using temporal statistics modeling”, in *Proc. IEEE Conf. Acoustics, Speech, and Signal Processing*, vol. 4 , pp. IV-3389 - IV-3392, 2002.
[12] H. Lu, Y.P. Tan, X.Y. Xue, and L. Wu, “Shot boundary detection using unsupervisedclustering and hypostudy testing”, in *Proc. IEEE Conf. Communications, Circuits and Systems*, vol. 2 , pp. 932 - 936, 2004.
[13] A. Miene, A. Dammeyer, Th. Hermes, and O. Herzog, “Advanced and adaptive shot boundary detection”, in *Proc. ECDL, WS, Generalized Documents*, pp. 39-43, 2001.
[14] R.J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, “Image change detection algorithms: A system survey” *IEEE Trans. Image Processing*, vol. 14, no. 3, pp. 294-307, 2005.
[15] J.M. Wang and S.W Chen, “Progressive background image generation” in *Proc. IPPR Conf. Computer Vision, Graphics and Image Processing*, pp. 858-865, 2002.
[16] C. Yong, Y. Xu, and X. De, “A method for shot boundary detection with automatic threshold” in *Proc. IEEE Conf. Computers, Communications, Control and Power Engineering*, vol. 1, pp. 582-585, 2002.
[17] H. Zhiwei, L. Jilin and L. Peihong, “New method of background update for video-based vehicle detection”, in *Proc. IEEE Conf. Intelligent transportation systems*, pp. 580-584, 2004.
[18] Y. Zhang, P. Shi, E.G.. Jones and Q. Zhu, “Robust background image generation and vehicle 3D detection and tracking”, in *Proc. IEEE Conf. Intelligent Transportation Systems*, pp. 12-16, 2004.