AUTOMATIC DETECTION OF LOGOS IN VIDEO AND THEIR REMOVAL USING INPAINTING

Dong-Kyu Lee, Jino Lee, Jaeyoon Kim, Ji Won Lee, and Rae-Hong Park

Department of Electronic Engineering, Sogang University, Seoul, Korea

ABSTRACT

In this paper, we propose an algorithm that automatically detects and removes moving logos in video. The proposed algorithm consists of logo detection, tracking and extraction, and removal. In the logo detection step, we automatically detect moving logos using the saliency map and the scale invariant feature transform. In the logo tracking and extraction step, an improved mean shift tracking method and backward tracking technique are used in which only logo regions are extracted by using color information. Finally, in the logo removal step, the detected logo region is filled with the region of neighborhood using an exemplar-based inpainting that can fill a large detected region without artifacts. These steps are effectively interconnected using control flags. Experimental results with various test sequences show that the proposed algorithm is effective to detect, track, and remove moving logos in video.

KEYWORDS

Logo, Logo Detection, Logo Removal, Logo Tracking, Logo Inpainting

1. INTRODUCTION

Recently, with the development of visual media market, an advertisement through visual media has powerful influence on viewers. Due to the properties of video contents that can convey a lot of information to viewers in a very short time, companies advertise for themselves and their products through visual media. Among various types of advertisements, appearance of products in a popular movie or TV program is common. In this case, majority of viewers cannot realize the appearance of logos in movies, dramas, or sports as an advertisement which has commercial purpose. It makes companies advertise themselves and their products without distrust of commercial advertisements. In this type of advertisement, the way how the product is viewed by prospective buyers is considered as important. In particular, logo is often regarded as what has to be clearly brought out.

Sometimes, however, several cases occur in which we need to remove the logo. First, it occurs when there is excessive appearance of logo. Too much frequent appearance of logo can be plain or compulsory for viewers, which prevents viewers from concentrating on a video content. The second case occurs when logos appear regardless of video producer's intent. For example, a documentary or news is produced for the public interest. In this case, the logo is also considered to be removed. Third, television stations put their logo into the background of programs to announce that they have the copyright on the programs. If the original program is rebroadcast by other television stations, the logo of the original station is to be removed [1], [2].

A simple manual logo removal method makes the logo region blurred and requires a lot of time and effort. Also unnatural scene generated from mosaic regions makes viewers feel uncomfortable. Recently, in order to solve this problem more easily and reliably, methods for removing logo using computer vision techniques have been developed [1]–[5].

Generally, an overall logo removal system consists of two steps: logo detection and logo removal. Prior to logo removal, it is important to accurately detect the logo because most of logo removal algorithms have been proposed with a view to automating all processes of logo removal with detected logos. Wang *et al.* [1] used gradient features for template matching, and regularization partial differential equations (PDEs) for logo inpainting. Yan *et al.* [2] proposed Bayesian classifier framework for detecting a logo and used matching based logo removal. Tan and Kankanhalli [3] proposed a removal algorithm using PDEs, which is equivalent to Wang *et al.*'s inpainting method [1]. Yeh and Shih [4] proposed logo processing methods and circuits, which compare a base frame for detection and change color of logo region for removal. These existing algorithms, however, have limited capabilities to deal with moving logos, because these approaches just consider TV logos with an assumption that the position of logo is fixed. And the region to be removed does not show reliable results, giving blurred results. VirtualDub [5] is a video processing utility for capturing and editing video. It provides a function for only logo removal without logo detection and tracking. In this method, a user manually has to find a frame where the logo is best visible and marks the logo by a specific color. The marked logo region is removed in successive frames. In addition, Ozay and Sankur [6] proposed a method of TV logo detection and classification, where TV logos are detected in static regions. Ballan *et al.* [7] proposed a method that detects and recognizes localization of trademarks. Wang *et al.* [8] focused on TV logo detection. However, in this method, the position of a TV logo is assumed to be fixed.

Besides, the logo removal method can be used to remove simple visible watermarks [3]. By using template of watermarks instead of logos, the logo removal method can be applied to watermark removal. It can also be used for inserting another template into an image or video [9]. The region of selected pattern can be replaced by another template.

In this paper, we propose a new algorithm that detects, tracks, and removes moving logos in video. We use the saliency map [10] and the scale invariant feature transform (SIFT) [11]–[13] for logo detection, whereas the mean shift method [14]–[17] for logo tracking. We also use an exemplar-based inpainting method [18]–[20] for logo removal.

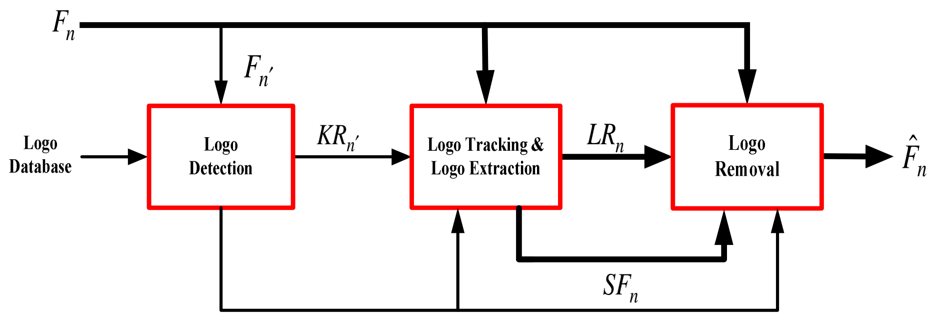
The rest of the paper is organized as follows. In Section 2, the proposed algorithm is outlined. In Section 3, an effective logo detection method is described, and in Section 4, how to track the logo and extract only the logo region is proposed. Section 5 presents how the extracted region is removed by image inpainting. Experimental results of the proposed algorithms are given in Section 6, and finally Section 7 concludes the paper.

2. SYSTEM OVERVIEW

Figure 1 shows the block diagram of the proposed algorithm. The proposed system consists of three steps: logo detection, tracking and extraction, and removal, where tracking and extraction are combined. Let F_n be nth frame of the input video. Logos are detected once every n_Δ frames in a logo detection step, i.e., logo detection performs in an n_Δ-frame interval. $F_{n'}$ denotes the frame where logo detection is performed, in which n' is expressed as,

$$n' = n, \text{ if } n \equiv 1 \bmod n_\Delta. \qquad (1)$$

The saliency map [10] and SIFT [11] are used for logo detection. In experiments, we set $n_\Delta = 5$. If a logo exists in $F_{n'}$, it is detected and a kernel region $KR_{n'}$ is initialized. Then, control flag $CF_{n'}$



is set to 1 and next steps are performed. In a logo tracking and extraction step, using given $KR_{n'}$, forward/backward logo tracking is performed from the frame where a logo is detected. Backward tracking enables tracking logos that appear between two logo detection times $(n' - n_{\Delta})^{th}$ and n'^{th} frames. The mean shift method [14] is used for logo tracking, in which only the logo region LR_n is extracted from KR_n. If the logo is missing or gets out of the frame during tracking, a status flag SF_n is set to 1 and logo removal is not performed. In case that a logo is not detected although it exists in $F_{n'}$ and $F_{n'-1}$, $KR_{n'}$ is not initialized. But if $SF_{n'-1}$ and $SF_{n'}$ are 0 (i.e., a logo

Table 1. Meaning of Control Flags.

Control Flag	State	Meaning
$CF_{n'}$	0	Logo is not detected
	1	Logo is detected (initialized)

Table 2. Function of Each Step with Control Flags.

Step	$CF_{n'}$	SF_{n-1}	SF_n	Function
Tracking & Extraction	0	0	0	ON
	0	1	0	OFF
	0	×	1	OFF
	1	×	×	ON
Removal	×	×	1	OFF
	×	×	0	ON

exists in $F_{n'-1}$ and $F_{n'}$), a logo tracking goes on with the given previous tracking result $KR_{n'-1}$. Detected logo region is removed by using exemplar-based inpainting [18]. $\hat{F}_n$ denotes the final results with the logo removed. Table 1 lists the meaning of control flags and Table 2 shows the function of each step with the values of control flags. An example of a timing diagram for control flags is shown in Figure 2.

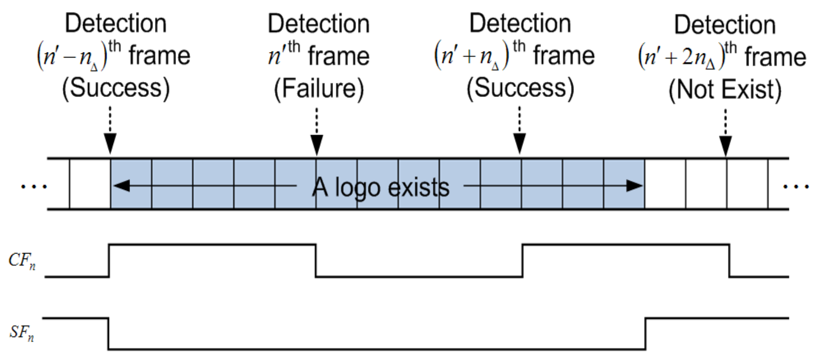


Figure 2. An example of a timing diagram for control flags.

3. Logo Detection

In this section, we introduce how to detect moving logos in video. A large number of papers have been presented to detect objects of interest in video [10]. The interested logo can be found in input frame F_n. Automatic logo detection proceeds in each input frame F_n. High computational complexity of logo detection leads to high computation time of the overall system. In this paper, logo detection is done using the saliency map [10] and SIFT [11]. If a logo is detected in F_n, CF_n is set to 1. When it is detected at first in F_n, the kernel is initialized automatically. Logos have some common features. Logos usually have high-contrast visual features like color and the size of logo is small compared to the whole image size. These common characteristics are used for logo detection.

The first step of logo detection is localization, in which candidate logo regions are extracted from an entire image. Logos have high-contrast regions with visual features as described above. The high-frequency component of F_n can be obtained using a high-pass filter. Then, the region of high-frequency can be a candidate logo region. At first, F_n is downsampled to reduce the computational complexity.

Next, the saliency value is calculated at each sampled pixel of F_n. Then, pixels that have the saliency value greater than threshold become a logo candidate region. The template matching algorithm is needed in order to find the rectangular kernel that includes a logo region among logo candidate regions. For initializing a kernel in logo detection, we use the SIFT [11]. The SIFT has been widely used in video processing and recognition [21, 22], because it considers desirable and effective features of logos in video sequence, such as scale change, rotation, and three-dimensional viewpoint change.

The SIFT consists of four stages. The first stage is detecting extrema in scale space, in which a difference of Gaussians (DoG) is used. The second stage is keypoint localization, where unstable extrema with low contrast are rejected. The pixels with DoG value smaller than threshold are eliminated. Also edge pixels are rejected. The third stage is generating an orientation histogram of neighboring 16 × 16 pixels around keypoints. The orientation histogram is generated from gradient magnitude $m_n(x, y)$ and orientation $\theta_n(x, y)$ at (x, y), which are defined, respectively, as

$$m_n(x,y)=\sqrt{(G_n(x+1,y)-G_n(x-1,y))^2+(G_n(x,y+1)-G_n(x,y-1))^2}, \quad (2)$$

$$\theta_n(x,y)=\tan^{-1}\left(\frac{G_n(x,y+1)-G_n(x,y-1)}{G_n(x+1,y)-G_n(x-1,y)}\right) \quad (3)$$

where G_n represents a Gaussian smoothed image of F_n. The orientation histogram has 36 orientation bins. The fourth stage is creating the descriptor that is obtained by sampling the orientation histogram. In the previous stage, there are orientation histograms of neighboring 16×16 pixels around keypoints. In this stage, 16×16 regions are divided into 4×4 subregions. Each descriptor is created by the orientation histogram with 8 orientation bins of neighboring 4×4 pixels. The descriptor has a $4\times 4\times 8$ =128 element feature vector for each keypoint.

The best matched point has to be found to match the logo that is similar to the given template. There can be many outliers and incorrectly matched points. Therefore, we need a robust way to discard incorrectly matched points. In the SIFT [11], an effective way is used by comparing the distance of the closest neighbor to that of the second-closest neighbor. If the distance ratio of two neighbors is greater than 0.8, the matched point is discarded. In general, a k-d tree search algorithm has good performance [13]. However, its performance is degraded for high-dimensional data. The SIFT descriptor has 128 dimensions. Therefore, a k-d tree algorithm cannot be used because of high computation time. The best-bin first algorithm uses a modified search ordering of their closest distance for the k-d tree algorithm. This heap-base priority queue is used for efficient determination [13]. The result of SIFT matching is shown in Figure 3, in which each matched point pair between logo template and input frame F_n is linked. The template logo of Figure 3 has 48 feature points. This number of feature points is enough to detect a logo. It is difficult to detect a logo that doesn't have enough feature points. If the logo region has a low contrast or a large motion blur, it doesn't have enough feature points. The template logo and logos in the input sequence that have high contrast regions are used in experiment. Also it is assumed that blur does not exist.

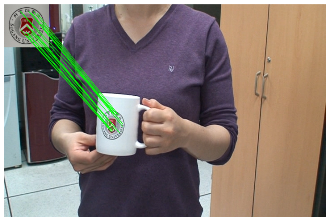


Figure 3. Results of SIFT matching of a logo template and an input frame F_n.

The control flag CF_n at n^{th} frame must be determined depending on the condition as

$$CF_n = \begin{cases} 1, & N_m > N_t \times th_0 \\ 0, & \text{otherwise} \end{cases} \qquad (4)$$

where N_m and N_t denote the total numbers of matched points and feature points of a template, respectively. Experimentally, we set th_0 to 0.15. If CF_n is 1, the size of a kernel that includes a logo has to be calculated

If CF_n is 1, the size and center point of a kernel are determined by matched feature points. There are matched points between a logo template and input frame F_n. The center pixel of the matched points (x_c, y_c) is calculated as

$$(x_c, y_c) = \frac{1}{N_m} \sum_{i=1}^{N_m} (x_i, y_i) \qquad (5)$$

where (x_i, y_i) represents the coordinate of ith matched point. Then, in order to measure the size of a kernel, the Euclidian distance between each pair of matched points is calculated in a logo template and input frame F_n. The Euclidian distance ratio set is defined as

$$E_n = \left\{ ED_{ij}^{F_n} / ED_{ij}^{T} \middle| 1 \le i, j \le N_m (i \ne j) \right\}, \tag{6}$$

$$ED_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{7}$$

where $ED_{ij}^{F_n}$ represents the Euclidean distance between ith and jth matched points in an input frame whereas ED_{ij}^{T} denotes that in a template. The ratio of $ED_{ij}^{F_n}$ and ED_{ij}^{T} is approximately the size ratio of the matched object in template and input frame F_n. Also the estimation of rotation angle of the matched logo in input frame is performed by calculating the difference of slopes between two matched points in both logo template and input frame F_n. The difference of slope set A is defined as

$$A_n = \left\{ a_{ij}^{F_n} - a_{ij}^{T} \mid 1 \le i, j \le N_m (i \ne j) \right\}, \tag{8}$$

$$a_{ij} = \tan^{-1}\left[(y_i - y_j) / (x_i - x_j) \right] \tag{9}$$

where $a_{ij}^{F_n}$ is the slope between ith and jth matched points in an input frame whereas a_{ij}^{T} is that in a template. The difference between slopes is used for estimation of rotation angle. The median values of E_n and A_n are used in order to determine reliable kernel KR_n, which is a quadrangular kernel. The height, width, and center pixel of KR_n are computed. The Euclidean distances from (x_c, y_c) to four vertices in template are calculated. Then, four vertices of KR_n in F_n are calculated using the center pixel of a kernel (x_c, y_c), the size ratio of matched logo in input frame F_n and template, and slope difference between input frame F_n and template. The height, width, and center pixel of KR_n are calculated from four vertices. The region of KR_n is determined in this stage and used in tracking afterwards.

4. LOGO TRACKING AND EXTRACTION

An object tracking technique is used for various applications such as video surveillance, face recognition, and automatic navigation of robot, etc. Particularly, in our case, objects to be tracked are moving logos. Thus, the proposed algorithm has to satisfy two specific conditions, which distinguish the proposed algorithm from conventional methods [1], [23]–[25]. One is that the shape of an object, i.e., the logo, easily changes frame to frame. The other is that the tracking result does not have to contain other region than the logo. By limiting the region to be removed to logo region only, the computation time is decreased, and undesirable artifacts by inpainting are minimized.

Existing logo removal algorithms in video [2]–[5] do not include a tracking step, because they focused on fixed logo in video. In several algorithms [1], [23] the logo to be removed is animated with its motion very limited. To satisfy two above-mentioned conditions, we choose the mean shift tracking method [14] for logo tracking. The mean shift tracking uses color information, thus it gives better performance than other algorithms in circumstance where shape and scale changes exist. And it is simple and its implementation is easy.

Despite the advantages of the mean shift tracking method, it has drawbacks. First, this method is degraded in case of brightness change since it uses only color information. Second is a fixed kernel shape as it is a kernel-based method. In the proposed method, we solve these problems through a model update [17] and logo extraction.

In Section 4.1, an improved mean shift tracking method using background exclusion and model update is described. Backward tracking and logo extraction are presented in Sections 4.2 and 4.3, respectively.

4.1. Improved Mean Shift Tracking Method

The basic procedure of the mean shift tracking method is to iteratively compute the location of a model that maximizes the similarity between a given object model and a new candidate model detected in the current frame [17]. The object model $\mathbf{s}$ is defined as histogram generated from the object region, which is given by KR_n. Similarly, the candidate model $\mathbf{r}$ is defined as histogram generated from the candidate region. For low computational cost, discrete densities, i.e., N_h-bin histograms are used. In the step of logo tracking, we will use boldface letters to denote the location (x, y).

At location $\hat{\mathbf{y}}_0$, which is modified parameter of the current center location by performing Taylor expansion of Bhattacharyya coefficients and the approximation, the new location $\hat{\mathbf{y}}_1$ of the kernel with bandwidth h is recursively computed as [14]

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{N_k} \mathbf{x}_i w_i g\left(\left\| \frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^{N_k} w_i g\left(\left\| \frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h} \right\|^2 \right)} \tag{10}$$

where $\mathbf{x}_i$ is the normalized pixel location in the candidate model region that has a total number of N_k pixels, and $g(x)$ is the derivative of a kernel profile. In this paper, we use the Epanechnikov kernel. At each pixel $\mathbf{x}_i$, the weight is derived from the Bhattacharyya distance, which is given by

$$w_i = \sum_{u=1}^{N_h} \sqrt{\frac{s_u}{r_u(\hat{\mathbf{y}}_0)}} \delta\left(b(\mathbf{x}_i) - u\right) \tag{11}$$

where $\mathbf{s} = \{s_u\}_{1 \le u \le N_h}$, $\mathbf{r}(\hat{\mathbf{y}}_0) = \{r_u(\hat{\mathbf{y}}_0)\}_{1 \le u \le N_h}$, δ is the Kronecker delta function, and $b(\mathbf{x}_i)$ is a histogram bin corresponding to pixel value at location $\mathbf{x}_i$.

We improve the mean shift tracking method by combining background exclusion [15], [16] and model update [17]. Especially, the background exclusion is also used to extract the logo from the kernel. First, we separate between the logo and background to exclude the background [17]. The histogram h^f of the kernel region (foreground), which is called object window, and the histogram h^b of the region (background) around the object window, which is called background window, are separately evaluated. In this paper, the background window is defined as the region that is obtained by subtracting the object window from the expanded region which is 1.2 times larger than the object window. The log-likelihood that pixel $\mathbf{x}_i$ belongs to the logo is calculated as

$$L(\mathbf{x}_i) = \log \frac{\max\left(h^f\left[b(\mathbf{x}_i)\right], \varepsilon\right)}{\max\left(h^b\left[b(\mathbf{x}_i)\right], \varepsilon\right)}, \qquad (12)$$

where ε is a small constant to avoid numerical instability. If $L(\mathbf{x}_i)$ is positive, $\mathbf{x}_i$ is assumed to belong to the logo, otherwise, to the background. Pixels that contain logo are dilated by one pixel in order to improve the reliability of the log-likelihood. Figure 4 illustrates an example of logo extraction. Figure 4(a) shows the object window (green box) and the background window (red box) whereas Figure 4(b) shows the extracted logo (blue pixels). We remove the extracted logo by an inpainting algorithm that is explained in the next step.

(a) (b)

Figure 4. Logo extraction. (a) object window (green box) and background window (red box), (b) extracted logo (blue pixels).

Second, we subtract the background weight from the foreground weight [15], [16]. Zhao and Nevatia [16] replaced w_i in (11) with

$$w_i = \lambda^f w_i^f - \lambda^b w_i^b. \qquad (13)$$

The foreground weight w_i^f is calculated similarly to (11), while the background weight w_i^b is obtained as the same form as in [16]. λ^f and λ^b determine rates of w_i^f and w_i^b, respectively. After the convergence of the mean shift iteration, the logo region LR_n is extracted as

$$LR_n(\mathbf{x}) = \begin{cases} 1, & L_n(\mathbf{x}) > 0 \\ 0, & \text{otherwise} \end{cases} \qquad (14)$$

where $L_n(\mathbf{x})$ represents the log likelihood $L(\mathbf{x})$ after the convergence of the mean shift iteration in nth frame.

Third, after the convergence of the mean shift iteration, the target model is updated under two conditions. The first condition of update is that the number of pixels of the logo in LR_n is larger than a threshold th_1. The second condition is that the Bhattacharyya coefficient ρ between the target model and the candidate model is larger than a threshold th_2. If these two conditions are satisfied, the target model is updated with the current candidate model [17] as

$$\mathbf{s}_n = \begin{cases} \mathbf{r}_n, & \sum_{i=1}^{N_k} T(\mathbf{x}_i) \geq th_1 \ \& \ \rho\left[\mathbf{s}_{n-1}(\mathbf{y}), \mathbf{r}_n\right] \geq th_2 \\ \mathbf{s}_{n-1}, & \text{otherwise} \end{cases} \qquad (15)$$

where $T(\mathbf{x}_i)$ is set to 1, if $L(\mathbf{x}_i)$ is larger than th_3, otherwise, it is set to zero. In this paper, the thresholds th_1 and th_2 are experimentally set to 80% of the total number of pixels in the object window and 0.8, respectively. The threshold th_3 is experimentally optimized for each logo of video sequence.

To check whether the logo exists or not, Bhattacharyya coefficients ρ and the number of pixels of the logo in LR_n are used, similarly to the model update scheme. If Bhattacharyya coefficient ρ and the number of pixels of the logo in LR_n are smaller than thresholds th_4 and th_5, respectively, we decide that the logo is missing or gets out of the current frame and the flag SF_n is determined as

$$SF_n = \begin{cases} 1, & \sum_{i=1}^{N_k} T(\mathbf{x}_i) \le th_4 \; \& \; \rho[\mathbf{s}_{n-1}(\mathbf{y}), \mathbf{r}_n] \le th_5 \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

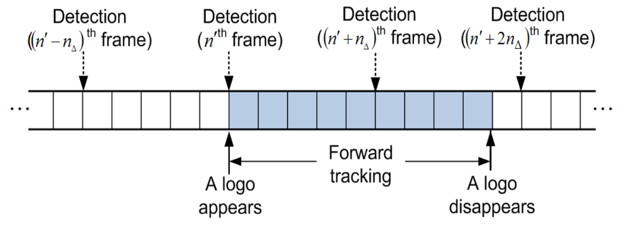
where th_4 and th_5 are experimentally set to 40% of the total number of pixels in the object window and 0.7, respectively.

4.2. Backward Tracking

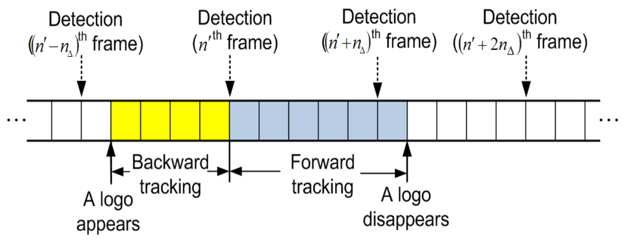
Generally, object tracking is performed in the temporally forward direction. However, in the proposed method, logo tracking is performed in both forward and backward directions from the frame with detected logo, because logo detection is periodically performed in every n_Δ frames. Logo detection is performed once every n_Δ frames in the proposed method. Therefore, we need to detect new logos which appear between logo detection times $(n' - n_\Delta)^{\text{th}}$ and n'^{th} frames. Otherwise, logos which appear between logo detection times remain unremoved until they are detected and removed. To solve this problem, logo tracking is performed backward from the current logo detection frame $(n'^{\text{th}} \text{ frame})$ to previous detection frame $((n' - n_\Delta)^{\text{th}} \text{ frame})$. Figure 5 shows the timing diagram for logo tracking. In Figure 5(a), a new logo appears in n'^{th} frame when logo detection is performed. In this case, backward tracking is not needed. On the other hand, if a new logo appears between logo detection frames $(n' - n_\Delta)$ and n' as shown in Figure 5(b), the moving logo is detected by backward tracking and the next step continues.

In experiments, n_Δ is set to 5. Because logo detection takes longer than logo tracking, small n_Δ excessively increases the computation time. On the other hand, large n_Δ can reduce the accuracy of logo detection.

A decision of the logo existence in the current frame is determined as (16), which initiates forward tracking. The procedure reduces unnecessary logo tracking and removal at frames in which logos do not exist.



(a)



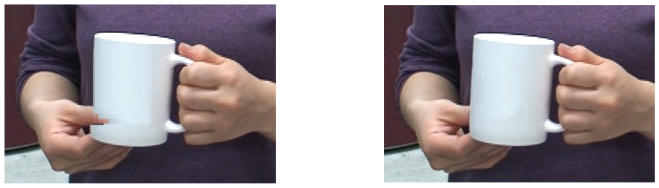
(b)

Figure 5. Timing diagram for logo tracking. (a) forward tracking, (b) forward tracking and backward tracking.

4.3. Logo Extraction

In many tracking algorithms [14], [15], [25], tracking results are shown as rectangles containing objects. However, in our system, we use results of background exclusion that is obtained in the logo tracking step. It enables to minimize the number of pixels in the region to be removed and to reduce the computation time by inpainting.

Reliable logo extraction also prevents undesirable artifacts by inpainting as shown in Figure 6. As shown in Figure 6(a), the undesirable artifact is generated around a thumb. With logo extraction, however, the artifact does not occur as shown in Figure 6(b), in which the logo is removed without error.



(a) (b)

Figure 6. Comparison of logo removal results without and with logo extraction. (a) results without logo extraction, (b) result with logo extraction.

5. Logo Removal Using Inpainting

To remove the logo, we fill the logo region by the image inpainting algorithm [18]–[20], [26], [27]. There are two approaches to image inpainting: PDE-based methods [26], [27] and exemplar-based methods [18]–[20]. A PDE-based method fills the target region, which denotes the logo region to be removed, using neighboring pixel values with a diffusion equation. Its weakness is producing conspicuous blur effects in the large region. Our goal is to fill the target region without noticeable artifacts. Hence, it is not proper to use a PDE-based inpainting method for logo removal. We use an exemplar-based inpainting method that uses block-based texture synthesis without scale dependency. It is proper to fill the large logo region without noticeable artifacts such as blurring.

In the exemplar-based inpainting [18], the target region is reconstructed from texture patches in the source region. The filling order significantly affects the capability of the exemplar-based inpainting. Filling order is determined by the priority of a pixel in the logo region LR_n. We use two terms, confidence term $C(\mathbf{p})$ and data term $D(\mathbf{p})$ at pixel $\mathbf{p}$, to evaluate the priority $P(\mathbf{p})$, which is defined as

$$P(\mathbf{p}) = C(\mathbf{p})D(\mathbf{p}). \tag{17}$$

Confidence term $C(\mathbf{p})$ denotes proportion of pixels belong to the source region SR_n in the target patch $\Psi_{\mathbf{p}}$, which is expressed as

$$C(\mathbf{p}) = \left(\sum_{\mathbf{q} \in \Psi_{\mathbf{p}} \cap SR_n} C(\mathbf{q}) \right) \Big/ \left| \Psi_{\mathbf{p}} \right| \tag{18}$$

where target patch $\Psi_{\mathbf{p}}$ is a fixed size window centered at pixel $\mathbf{p}$ located along the boundary ∂LR_n and $\left|\Psi_{\mathbf{p}}\right|$ is the area (number of pixels) of $\Psi_{\mathbf{p}}$. Initial value of $C(\mathbf{p})$ is given as $C(\mathbf{p}) = 0,\ \forall \mathbf{p} \in LR_n$ and $C(\mathbf{p}) = 1,\ \forall \mathbf{p} \in SR_n$. Therefore, a patch that has a higher proportion is filled first.

Data term $D(\mathbf{p})$ measures edge strength of boundary ∂LR_n for preserving linear structures. We use the cross-isophote model [20] that uses total variation diffusion, in which curvature of intensity is defined as data term. Intensity term denoted as I_n is the gray scale values of nth frame. Priority should be positive, thus $D(\mathbf{p})$ is expressed as the absolute value of a diffusion term,

$$D(\mathbf{p}) = \left| \nabla \cdot \left\{ \frac{\nabla I_n(\mathbf{p})}{\left|\nabla I_n(\mathbf{p})\right|} \right\} \right|. \tag{19}$$

Data term introduces linear structures in the source region to the target region. Thus, occluded structures are restored by propagation of linear structures in the source region. A given patch in the isophote direction, which has large $\left|\nabla I_n\right|$, is synthesized first.

We find pixel $\hat{\mathbf{p}}$ with maximum priority and the most similar source patch $\Psi_{\hat{\mathbf{q}}}$, where $\hat{\mathbf{q}}$ is the center point of the patch. We search whole source region SR_n to find a patch with minimum distance from the patch $\Psi_{\hat{\mathbf{p}}}$, i.e.,

$$\Psi_{\hat{\mathbf{q}}} = \arg \min_{\Psi_{\mathbf{q}} \in SR_n} d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\mathbf{q}}). \tag{20}$$

Distance $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\mathbf{q}})$ is defined as the sum of squared differences expressed as

$$d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\mathbf{q}}) = \sqrt{\frac{1}{N_{\mathbf{p}}} \sum_{i}^{N_{\mathbf{p}}} \left(\left\| \mathbf{C}_{\hat{\mathbf{p}}_i} - \mathbf{C}_{\mathbf{q}_i} \right\|^2 + \left\| \mathbf{G}_{\hat{\mathbf{p}}_i} - \mathbf{G}_{\mathbf{q}_i} \right\|^2 \right)} \tag{21}$$

where $N_{\mathbf{p}}$ is the number of pixels in a patch and $\mathbf{C}$ is the color vector and $\mathbf{G}$ is the image gradient vector.

A target patch $\Psi_{\hat{\mathbf{p}}}$ is replicated by a selected source patch $\Psi_{\hat{\mathbf{q}}}$, and its confidence term is updated as

$$C(\mathbf{p}) = C(\hat{\mathbf{p}}), \quad \forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap LR_n. \qquad (22)$$

Then, the logo region LR_n and boundary ∂LR_n are updated.

Figure 7 describes a notation diagram of an image inpainting algorithm for logo removal. The region adjacent to the target region LR_n (yellow pixels) is used to have a repeated pattern, not a mess or complex structure. Thus, we set the restricted source region SR_n (blue pixels) around the boundary ∂LR_n (red pixels) to reduce the computation time. The source region is set to neighboring pixels from the boundary. In our experiments, we set the source region as 20×20 window at each boundary pixel and the patch size to 7×7.

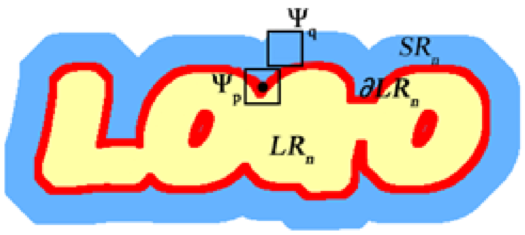


Figure 7. Notation diagram for logo inpainting.

6. EXPERIMENTAL RESULTS AND DISCUSSIONS

We have tested our proposed algorithm using various videos. Here, we present four representative results due to the page limit. We implement the proposed method in C++ on an Intel Core i5 (2.67GHz, 4GB RAM).

We compare the proposed method with five existing logo removal algorithms [1]–[5] in aspects of automatic detection, movement of logo, removal method, and the number of logos in a frame, as listed in Table 3. Most of existing removal methods [1]–[4] as well as the proposed method automatically detect logos. While the proposed method considers moving logos, the others focus on the logo with their motion very limited [1] or do not deal with moving logos [2]–[5]. For logo removal, we use an exemplar-based method [18] that is a state-of-the-art inpainting method with little artifacts. In terms of the number of logos in a frame, the proposed method is superior to the conventional methods [1], [3], [5] that remove a single logo.

Method	Detection	Moving Logo	Removal Method	Number of Logos in a Frame
Wang *et al.* [1]	Automatic	Δ	PDE	Single
Yan *et al.* [2]	Automatic	×	PDE	Multiple
Tan and Kankanhalli [3]	Automatic	×	Extrapolation	Single
Yeh and Shih [4]	Automatic	×	No description	Multiple
VirtualDub [5]	Manual	×	No description	Single
Ours	Automatic	O	Exemplar	Multiple

Figure 8 shows a comparison of two TV logo removal methods: VirtualDub [5] and the proposed method. The location of TV logo is fixed and its shape is not changed. Figure 8(a) shows sampled frames from a drama containing logo "SGBS". This test sequence is provided by Sogang Broadcasting Station at Sogang University (SGBS). The TV logo is surrounded by a red rectangle. In Figures 8(b) and 8(c), results by two methods are enlarged. The proposed method removes the logo without artifacts with boundary lines of a door preserved as shown in Figure 8(c), whereas the region removed by VirtualDub is blurred in Figure 8(b).

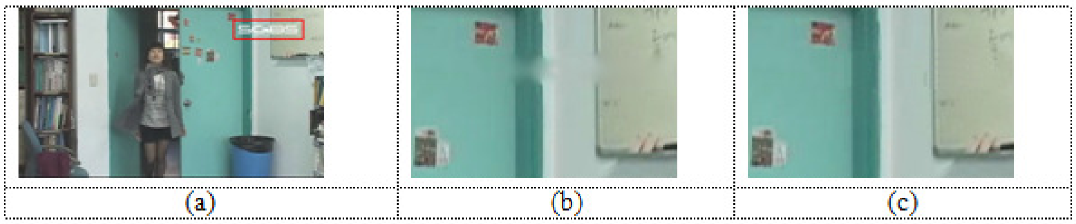


Figure 8. Comparison of two TV logo removal algorithms. (a) original frame (640 × 480), (b) enlarged result with TV logo removed by VirtualDub [5], (c) enlarged result with TV logo removed by the proposed method.

To evaluate the performance of the proposed method, we experiment with video sequences of our own making. Figure 9 shows the original "Cup lid" sequence and its logo removal results. It is tested for brightness change of input sequence. The "Cup lib" sequence has 150 frames of 720 × 480 pixels, which contains a single logo. Figures 9(a) and 9(b) show 36th and 71th frames of the original sequence with the logo, respectively. Because of illumination change, the brightness of the logo in Figure 9(b) differs from that in Figure 9(a). However, the proposed algorithm, using the model update in the tracking step, effectively removes the logo regardless of brightness change.

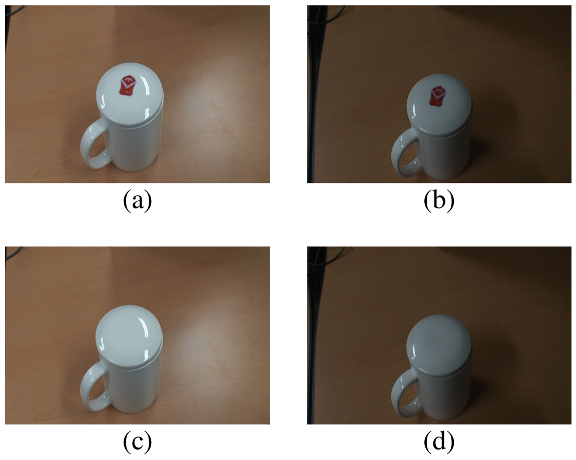


Figure 9. Frames with brightness change of a logo and the logo removal results (720×480, "Cup lid" sequence). (a) 36th frame, (b) 71th frame, (c) 36th frame with the logo removed, (d) 71th frame with the logo removed.

Figure 10 shows frames of the "Cup" sequence and their logo removal results. The "Cup" sequence has 710 frames of 720×480 pixels, which has 540 frames with a logo, and average size of the logo is 4800 pixels. Figure 10 shows the performance of the proposed algorithm in various conditions: the change of location, orientation, and scale of logos in video. Figures 10(a), 10(b), and 10(c) show 241th, 415th, and 539th frames of the original video with the logos, which consist of 47×54, 47×48, and 74×92 pixels, respectively. The number of pixels in the logo is increased from about 2500 pixels in Figure 10(a) to about 6300 pixels in Figure 10(c), with the logo moving. Orientation of the logo also changes by about 120 degrees through 298 frames. Figures 10(d), 10(e), and 10(f) show the frames with the logo removed by the proposed method. The results show that the proposed method automatically and correctly detects and tracks the logo because the SIFT used in the logo detection step gives good detection performance regardless of the scale and rotation change of a logo. The removed region of the logo looks natural.

Figure 11 presents results on the "Cup and Albatross" frames with two logos. It shows that the proposed method is applicable to frames with multiple logos. Figures 11(a), 11(b), and 11(c) show 56th, 262th, and 559th frames of the original sequence with logos, whereas Figures 11(d), 11(e), and 11(f) show those with the logos removed by the proposed algorithm, respectively. In the input sequence, one logo disappears and reappears, and the other continuously exists. For example, the logo "University symbol" that exists in Figure 11(a), disappears in Figure 11(b), and reappears in Figure 11(c). Figures 11(d), 11(e), and 11(f) show that once logos exist in a frame, these logos are successfully removed by the proposed method. Even though logos exist at the same frames as shown in Figures 11(a) and 11(c), the proposed method works normally in Figures 11(d) and 11(f). In terms of error, when location, orientation, and scale of logos change, removed logo regions look natural.

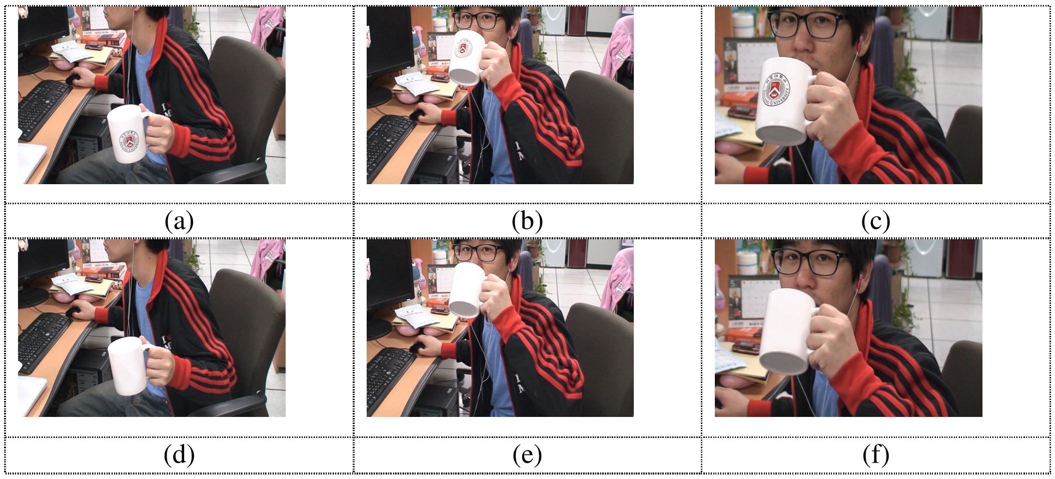


Figure 10. Test frames and logo removal results (720 × 480, “Cup” sequence). (a) 241th frame, (b) 415th frame, (c) 539th frame, (d) 241th frame with the logo removed, (e) 415th frame with the logo removed, (f) 539th frame with the logo removed.

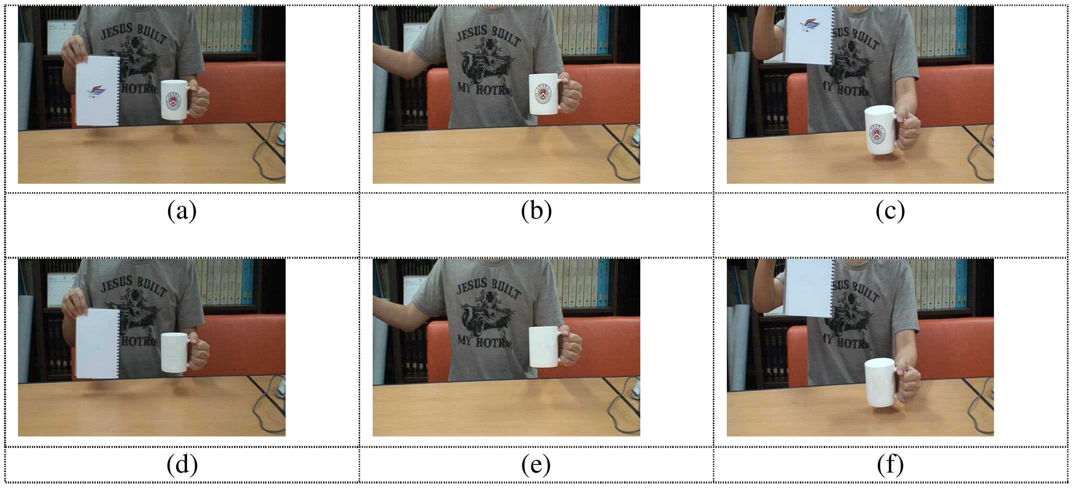


Figure 11. Test frames with two logos and logo removal results (720 × 480, “University symbol and Albatross” sequence). (a) 56th frame, (b) 262th frame, (c) 559th frame, (d) 56th frame with the logo removed, (e) 262th frame with the logo removed (f) 559th frame with the logo removed.

The performance of the proposed method is evaluated on “Cup” and “University symbol and Albatross” sequences, as listed in Table 4. The two sequences have 540 and 634 frames and the proposed method performs 0.705 and 0.497 frames per second, respectively. Its processing time is much smaller than the time taken to manually remove. So the proposed method saves a lot of time and effort compared to manual logo removal method. From Table 4, it is observed that the logo extraction leads to reduction of the average processing time. In “Cup” sequence, average processing time is reduced by 13 %, whereas in “Cup and Albatross” sequence, by about 4 %. On the other hand, as number of logos to be removed increases, the processing time also increases. Average processing time with “Cup and Albatross” sequence with two logos is longer than that with “Cup” sequence with a single logo. Note that average sizes of logos for two sequences are the same. In terms of the processing time, the ratio of the processing time for logo detection and tracking to that for removal is about 1 to 5. It means that processing time cost for inpainting is large.

Table 4. Performance Evaluation of The Proposed Method.

Sequence		"Cup"	"Cup and Albatross"
Size of a frame (pixels × pixels)		720 × 480	720 × 480
Total number of frames		540	634
Maximum number of logos in a frame		1	2
Average size of each logo (pixels)		4800	3000, 1800
Average processing time per frame (sec/frame)	with logo extraction	1.419	2.013
	without logo extraction	1.602	2.084

In various experiments, we find that parameter values may be sensitive to the characteristics of the input sequence, although some parameters are optimized and the others are selected for each logo in a sequence. To select a proper parameter value for each scene, a scene change detection method can be used in our detection method. For frames with a large motion or low image quality, the proposed method gives poor performance because of a small number of feature points, motion blurring, and logos that are not distinguished from their background. However, the time and effort of correction for remaining logos and artifacts after processing by the proposed method are much smaller than those for manual logo removal. Another drawback of the proposed method is a temporal variation of removed region between successive frames. It can be also solved by using a video inpainting method, but this requires a high computational burden, the processing time of which must be reduced.

7. Conclusions

We propose an effective method for detection, tracking, and removal of moving logos in video. In logo detection, using the saliency map and the SIFT, the proposed algorithm automatically detects moving logos in video. In logo tracking, the mean shift algorithm is used, which enables our method applicable to various conditions such as brightness, location, orientation, and scale change. We use control flags in order to interconnect each process of the algorithm for effective moving logo detection and tracking in video. In logo removal, we use an exemplar-based inpainting that is a state-of-the-art inpainting method. In the proposed method, multiple moving logos as well as a single logo are detected automatically, tracked, and removed naturally. The proposed method can be used to remove not only logos but also objects or watermarks. Also it can be used as preprocessing for replacing one template with another template in video.

Acknowledgements

This work was supported in part by the Brain Korea 21 Project.

REFERENCES

[1] J. Wang, Q. Liu, L. Duan, H. Lu, and C. Xu, “Automatic TV logo detection, tracking and removal in broadcast video,” in Proc. 13th Int. Multimedia Modeling Conf., vol. 4352, part II, pp. 63–72, Singapore, Jan. 2007.

[2] W.-Q. Yan, J. Wang, and M. S. Kankanhalli, “Automatic video logo detection and removal,” Multimedia Systems, vol. 10, no. 5, pp. 379–391, July 2005.

[3] W.-Q. Tan and M. S. Kankanhalli, “Erasing video logos based on image inpainting,” in Proc. IEEE Int. Conf. Multimedia and Expo, vol. 2, pp. 521–524, Lausanne, Switzerland, Aug. 2002.

[4] C.-H. Yeh and H.-H. Shih, “Logo processing methods and circuits,” U.S. Patent 7 599 558, Oct. 2009.

[5] K. Suhajda, “DeLogo filter for VirtualDub –version 1.3.2”.

[6] N. Ozay and B. Sankur, “Automatic TV logo detection and classification in broadcast videos,” in Proc. EUSIPCO 2009 17th European Signal Processing Conf., pp. 839–843, Glasgow, Scotland, Aug. 2009.

[7] L. Ballan, M. Bertini, A. D. Bimbo, and A. Jain, “Automatic trademark detection and recognition in sports videos,” in Proc. IEEE Int. Conf. Multimedia and Expo, pp. 901–904, Hannover, Germany, June 2008.

[8] J. Wang, L. Duan, Z. Li, J. Liu, H. Lu, and J. S. Jin, “A robust method for TV logo tracking in video streams,” in Proc. IEEE Int. Conf. Multimedia and Expo, pp. 1041–1044, Toronto, ON, Canada, July 2006.

[9] K. J. Hanna and P.J. Burt, “Video merging employing pattern-key insertion,” U.S. Patent 5 923 719, July 1999.

[10] K. Gao, S. Lin, Y. Zhang, S. Tang, and D. Zhang, “Logo detection based on spatial-spectral saliency and partial spatial context,” in Proc. IEEE Int. Conf. Multimedia and Expo, pp. 322–329, New York, USA, July 2009.

[11] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” Int. Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, Jan. 2004.

[12] X. Liu and B. Zhang, “Automatic collecting representative logo images from the Internet,” Tsinghua Science and Technology, vol. 18, no. 6, pp. 606–617, Dec. 2013.

[13] J. S. Beis and D. G. Lowe, “Shape indexing using approximate nearest-neighbour search in high-dimensional spaces,” in Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition, pp. 1000–1006, San Juan, Puerto Rico, June 1997.

[14] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 25, no. 5, pp. 564–575, May 2003.

[15] D. Caulfield and K. Dawson-Howe, “Evaluation of multi-part models for mean-shift tracking,” in Proc. Int. Machine Vision and Image Processing Conf., pp. 77–82, Coleraine, Northern Ireland, UK, Sep. 2008.

[16] T. Zhao and R. Nevatia, “Tracking multiple humans in crowded environment,” in Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition, vol. 2, pp. 406–413, Washington, DC, USA, June 2004.

[17] J. Jeyakar, R. V. Babu, and K. R. Ramakrichnan, “Robust object tracking using local kernels and background information,” in Proc. 2007 IEEE Int. Conf. Image Processing, vol. 5, pp. 49–52, San Antonio, TX, USA, Sep. 2007.

[18] A. Criminisi, P. Perez, and K. Toyama, “Region filling and object removal by exemplar-based inpainting,” IEEE Trans. Image Processing, vol. 13, no. 9, pp. 1200–1212, Sep. 2004.

[19] O. L. Meur, J. Gautier, and C. Guillemot, “Examplar-based inpainting based on local geometry,” in Proc. 2011 18th IEEE Int. Conf. Image Processing, pp. 3401–3404, Brussels, Belgium, Sep. 2011.

[20] J. Wu and Q. Ruan, “Object removal by cross isophotes exemplar-based inpainting,” in Proc. IEEE Int. Conf. Pattern Recognition, vol. 3, pp. 810–813, Hong Kong, China, Aug. 2006.

[21] S. H. Lee, J. Choi, and J. Park, “Interactive e-learning system using pattern recognition and augmented reality,” IEEE Trans. Consumer Electronics, vol. 55, no. 2, pp. 883–890, May 2009.

[22] D. Pan and P. Shi, “A method of TV logo recognition based on SIFT,” in Proc. 3rd Int. Conf. Multimedia Technology, pp. 1571–1579, 2013.

[23] J. Wang, L. Duan, Z. Li, J. Liu, H. Lu, and J. S. Jin, “A robust method for TV logo tracking in video streams,” in Proc. IEEE Int. Conf. Multimedia and Expo, pp. 1041–1044, Toronto, ON, Canada, July 2006.

[24] D. K. Park, H. S. Yoon, and C. S. Won, "Fast object tracking in digital video," IEEE Trans. Consumer Electronics, vol. 46, no. 3, pp. 785–790, Aug. 2000.

[25] R. Zhang, S. Zhang, and S. Yu, "Moving objects detection method based on brightness distortion and chromaticity distortion," IEEE Trans. Consumer Electronics, vol. 53, no. 3, pp. 1177–1185, Aug. 2007.

[26] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in Proc. ACM SIGGRAPH, pp. 417–424, New Orleans, LA, USA, July 2000.

[27] A. Telea, "An image inpainting technique based on the fast marching method," Journal of Graphics Tool, vol. 9, no. 1, pp. 25–36, Mar. 2004.