Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2010, Article ID 340717, 14 pages
doi:10.1155/2010/340717

Research Article

Automatic Moving Object Segmentation from Video Sequences Using Alternate Flashing System

Jae-Kyun Ahn, Dae-Yeon Lee, Chul Lee (EURASIP Member), and Chang-Su Kim (EURASIP Member)

School of Electrical Engineering, Korea University, Seoul 136-713, Republic of Korea

Correspondence should be addressed to Chang-Su Kim, changsukim@korea.ac.kr

Received 12 December 2009; Revised 15 March 2010; Accepted 31 March 2010

Academic Editor: Hsu-Yung Cheng

A novel algorithm to extract moving objects from video sequences is proposed in this paper. The proposed algorithm employs a flashing system to obtain an alternate series of lit and unlit frames from a single camera. For each unlit frame, the proposed algorithm synthesizes the corresponding lit frame using a motion-compensated interpolation scheme. Then, by comparing the unlit frame with the lit frame, we construct the sensitivity map, which provides depth cues. In addition to the sensitivity term, color, coherence, and smoothness terms are employed to define an energy function, which is minimized to yield segmentation results. Moreover, we develop a faster version of the proposed algorithm, which reduces the computational complexity significantly at the cost of slight performance degradation. Experiments on various test sequences show that the proposed algorithm provides high-quality segmentation results.

1. Introduction

Due to the advances in computation and communication technologies, the interest in video contents has increased significantly, and it has become more and more important to analyze and understand video contents automatically using computer vision techniques. To address the growing demand, various video analysis techniques have been introduced. Among them, moving object segmentation is a fundamental tool, which is widely used in a variety of applications. Especially, it plays an important preprocessing role in vision-based human motion capture and analysis, since the shape of a human subject after the segmentation is one of the main features for understanding human behaviors [1]. For example, in human pose estimation, 2D outlines of a human subject, which are extracted from one or more viewpoints using object segmentation techniques, are employed to reconstruct the 3D shape of a generic humanoid model [2–5]. Also, based on moving object segmentation, the outline of a human body can be tracked and used in human gesture analysis and human-machine interface [6, 7]. Moreover, the body shape and dynamics of a human subject can be used to recognize his or her identity [8, 9]. Therefore, the development of accurate video object segmentation techniques is essential to understand human behaviors.

Many approaches have been proposed for video object segmentation. They can be classified roughly into two categories: semiautomatic and automatic methods. Semiautomatic methods [10–13] first identify regions of interest coarsely using initial user interactions. Then, based on the initial information, they construct color, position, or motion models of objects and the background. The models are then used to separate the objects from the background more accurately. In [10], a background subtraction method was proposed to segment objects in video sequences with static backgrounds. It extracts moving objects by subtracting a given background from each frame in a video sequence. In [11], Criminisi et al. proposed a discriminative model, which is composed of motion, color, and contrast cues with spatial and temporal priors. Their algorithm achieves high quality video segmentation in realtime, but it works only if ground truth data is available for training model parameters. Also, tracking-based algorithms have been proposed in [12, 13]. They extract objects in the first frame based on

users' markings, and then track the objects in subsequent frames using color, position, and temporal cues. These semi-automatic methods [10–13] can achieve relatively accurate segmentation results using initial interactions. However, the interactions prevent them from being used in applications in which full automation is required.

On the other hand, automatic video segmentation methods extract objects without initial interactions [14–17]. They have the object detection stage, which defines objects of interest. Since objects of interest are usually moving, motion information is typically employed to distinguish the objects from the background. The motion field between consecutive frames is estimated, and then regions are classified as object or background based on the motion information. Chien et al. [14] proposed a background registration technique to estimate a reliable background image. Moving objects are extracted by comparing each frame with the estimated background. Tsaig and Averbuch's algorithm [15] divides each frame into small regions, finds the matching regions between consecutive frames, and declares the regions with large motions as objects. Yin et al.'s algorithm [16] learns segmentation likelihoods from the spatial contexts of motion information, and extracts objects automatically with tree-based classifiers. In [17], Zhang et al. proposed estimating the depth information of sparse points to detect foreground objects. These automatic methods [14–17] are effective, provided that objects and the background exhibit different motion characteristics. However, they may not provide accurate results for sequences with no or small object motions.

Recently, a new approach to automatic object segmentation, which uses extra information, such as depth, flash/no-flash difference, and depth-of-field (DoF), has been introduced [18–21]. Kolmogorov et al.'s algorithm [18] uses a stereo camera to estimate depth information, which is in turn used to extract foreground objects. It does not depend on the motion information between successive frames, but on the disparity information between stereo views. However, the disparity estimation is another challenging task, requiring heavy computational loads. In [19, 20], a flash is used to extract foreground objects using a single camera. After acquiring an ordinary image without flashing, it also captures an additional image lit by a flash. Then, by comparing the flash image with the no-flash one, color and intensity differences are obtained to extract objects. An alternative method is to use a matting model [21]. In an image with a shallow DoF, objects are focused while the background is not. Thus, the focused objects can be extracted automatically.

In this paper, we propose a novel algorithm to extract objects as well as humans from video sequences automatically. We extend the image segmentation techniques in [19, 20], which use a pair of flash and no-flash images, to the video segmentation case. The proposed algorithm is a tracking-based scheme using an alternate flashing system. When acquiring a video sequence, we capture even and odd frames with and without flash lights, respectively. Then, we find matching points between lit and unlit frames to construct a sensitivity map, from which the depth information can be inferred. In addition to the sensitivity map, color and temporal features are used to define an energy function, which is minimized by a graph cut algorithm to yield segmentation results. Simulation results demonstrate that the proposed algorithm provides reliable segmentation results.

The main contributions of this paper can be summarized as follows. First, we design a dedicated flashing system to capture an alternate series of lit and unlit frames. Second, we develop an efficient motion-compensated interpolation scheme, which matches lit and unlit frames to construct a sensitivity map. Third, we use the sensitivity map to accurately extract complex and deformable objects, especially humans, which are hard to segment out using conventional segmentation algorithms. Last, we implement a faster version of the proposed algorithm, which can be employed in real-time segmentation applications.

The rest of this paper is organized as follows. Section 2 describes our flashing system. Section 3 explains the features for segmentation, and Section 4 details the energy minimization scheme. Section 5 discusses implementation issues for real-time segmentation. Section 6 provides simulation results. Finally, Section 7 concludes the paper.

2. Flashing System

2.1. Video Acquisition. The proposed algorithm extracts objects based on the depth information estimated from pairs of lit and unlit frames. To capture an alternate series of lit and unlit frames, we construct a dedicated flashing system in Figure 1, which is composed of an 8×8 array of light emitting diodes (LEDs). An LED has a short response time and thus can be turned on or off quickly. When a camera starts to acquire a scene, the LEDs are turned on to light it. Then, they are turned off, before the next frame is captured. The flashing system is connected to a camera (Grasshopper [22]), which provides a trigger signal to the LEDs to synchronize with the image capturing system. We capture lit and unlit frames alternately by switching the LEDs on only when even frames are acquired. We capture lit frames with a short exposure time and unlit frames with a long exposure time as illustrated in Figure 2. Unlit frames, from which we extract objects, exhibit natural colors with an adequate exposure time. On the other hand, lit frames contain brighter objects and the darker background than unlit frames due to flash lights and a shorter exposure time. By comparing unlit frames with lit frames, we can perform high-quality segmentation of unlit frames.

When we capture a video of human subjects, alternate flashing may annoy them. To alleviate the annoyance, we set the frame rate to 120 frames/s, which corresponds to the flashing frequency of 60 Hz. At this relatively high frequency, humans can hardly notice flickering and the lights appear to be turned on steadily.

Since the proposed algorithm can achieve accurate segmentation results, it can be employed in various applications, in which the flashing system can be installed. For example, it can be used to understand human behaviors in

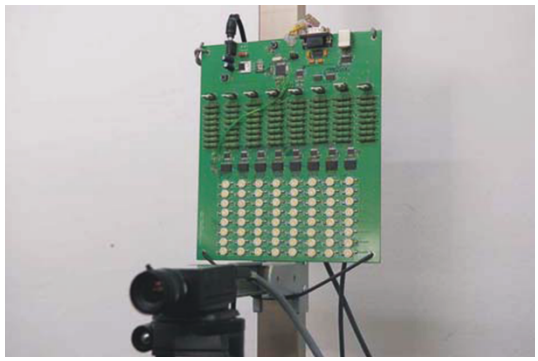


Figure 1: A dedicated flashing system, which captures lit and unlit frames alternately.

indoor environments [1], to substitute backgrounds in video conferencing applications [23, 24], and for mobile robots to detect obstacles [25]. It is noted that the flashing system is less effective in bright outdoor environments. Also, the current prototype of the flashing system is relatively bulky, but we expect that its size would be reduced by sophisticated packaging and it would be combined into a handheld camera system.

2.2. Matching between Lit and Unlit Frames. Using the alternate flashing system, we capture an input sequence with the frame rate of 120 frames/s. The proposed algorithm extracts the object layer from the unlit sequence with the frame rate of 60 frames/s. As shown in Figure 2, for an unlit frame U_i at time instance i, the proposed algorithm synthesizes the corresponding lit frame $\hat{L}_i$, and then compares U_i with $\hat{L}_i$ to derive the depth information.

A synthesized lit frame $\hat{L}_i$ is interpolated from the neighboring frames U_{i-2}, L_{i-1}, L_{i+1}, and U_{i+2}, as shown in Figure 2. To employ a motion-compensated interpolation scheme to synthesize $\hat{L}_i$, we develop a two-step motion estimation procedure. First, we estimate the global motion from U_i to L_{i-1}, which represents the motion of the background. Second, we refine the local motions of objects in a bilateral manner using the information in the subsequent frames L_{i+1}, U_{i+2} as well as the past frames U_{i-2}, L_{i-1}.

We first estimate the global background motion from U_i to L_{i-1}. However, the motion estimation between lit and unlit frames using image intensities is unreliable, since the scene irradiance varies dramatically according to lighting conditions. Instead, assuming that the background maintains a constant velocity within a short-time interval, we estimate the global motion field from U_i to U_{i-2} and use half the motion field approximately as the motion field from U_i to L_{i-1}. The global motion $\mathbf{v}^g$ from U_i to U_{i-2} is represented by an affine model, given by

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} a_3 \\ b_3 \end{pmatrix}, \tag{1}$$

where (x_0, y_0) and (x_1, y_1) denote the coordinates of the matching pixels in U_i and U_{i-2}, respectively. Then, the motion of the pixel (x_0, y_0) in U_i is given by $(x_1 - x_0, y_1 - y_0)$. The unknown six parameters, $a_1, a_2, a_3, b_1, b_2, b_3$ in (1) are estimated based on the optical flow equation [26],

$$I_x \cdot (x_1 - x_0) + I_y \cdot \ (y_1 - y_0) + I_t = 0, \tag{2}$$

where I_x and I_y denote the spatial derivatives, and I_t denotes the temporal derivative of the image intensity. By plugging (1) into the optical flow equation in (2), a linear system of equations for the unknown six parameters are derived and then solved using the least square method [27]. Note that an equation is set up for each (x_1, y_1) in the already segmented background layer of U_{i-2} only to avoid the effects of individual object motions in the global background motion estimation. Then, the global motion between L_{i-1} and U_i is approximated as the half of that between U_{i-2} and U_i.

After the global motion estimation, some pixels, especially object pixels, may experience different motions, which are not faithfully represented by the global motion model. Therefore, the local motion of a pixel, whose matching error is larger than a threshold after the global motion compensation, is refined using a bilateral block matching procedure. We assume that objects and the background have constant velocities within a short time interval. Thus, if the motion from U_i to U_{i-2} is $\mathbf{v}$, then the motion from U_i to L_{i-1} is regarded as $\mathbf{v}/2$. Similarly, the motion from U_i to U_{i+2} is $-\mathbf{v}$, and the motion from U_i to L_{i+1} is $-\mathbf{v}/2$, as shown in Figure 2. Therefore, the local motion $\mathbf{v}^l(\mathbf{p})$ of pixel $\mathbf{p}$ from U_i to U_{i-2} is computed, using the block matching algorithm in a bilateral manner, by

$$\mathbf{v}^l(\mathbf{p}) = \arg\min_{\mathbf{v}} \sum_{\mathbf{q}\in\mathbf{B}(\mathbf{p})} \|S(\mathbf{q},\mathbf{v}) - T(\mathbf{q},\mathbf{v})\|, \tag{3}$$

where

$$S(\mathbf{q},\mathbf{v}) = \begin{pmatrix} U_i(\mathbf{q}) \\ U_i(\mathbf{q}) \\ L_{i+1}(\mathbf{q} - \mathbf{v}/2) \end{pmatrix}, \quad T(\mathbf{q},\mathbf{v}) = \begin{pmatrix} U_{i+2}(\mathbf{q} - \mathbf{v}) \\ U_{i-2}(\mathbf{q} + \mathbf{v}) \\ L_{i-1}(\mathbf{q} + \mathbf{v}/2) \end{pmatrix}, \tag{4}$$

$L_i(\mathbf{q})$, $U_i(\mathbf{q})$ are intensities of lit and unlit frames at pixel $\mathbf{q}$, and $\mathbf{B}(\mathbf{p})$ is the 9×9 block around pixel $\mathbf{p}$. This bilateral motion estimation attempts to obtain coherent forward and backward motion vectors.

Finally, $\hat{L}_i$ is synthesized as

$$\hat{L}_i(\mathbf{p}) = \begin{cases} L_{i-1}\left(\mathbf{p} + \dfrac{\mathbf{v}^g}{2}\right), & \text{if } |R(\mathbf{p})| < 10, \\ \dfrac{1}{2}\left[L_{i-1}\left(\mathbf{p} + \dfrac{\mathbf{v}^l(\mathbf{p})}{2}\right) + L_{i+1}\left(\mathbf{p} - \dfrac{\mathbf{v}^l(\mathbf{p})}{2}\right)\right], & \text{otherwise,} \end{cases} \tag{5}$$

where $R(\mathbf{p})$ denotes the pixel matching error after the global motion compensation.

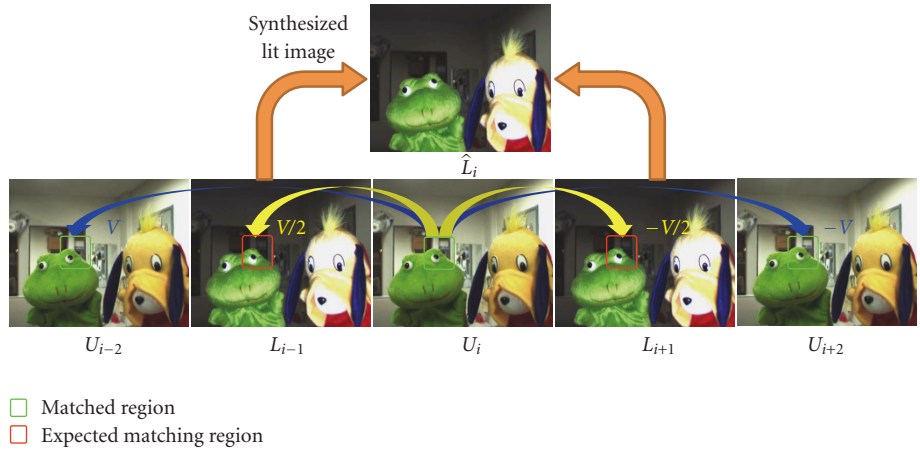


FIGURE 2: The proposed algorithm acquires an alternate series of lit and unlit frames. For an unlit frame U_i, the proposed algorithm estimates the motion vector field, and then synthesizes the corresponding lit frame $\hat{L}_i$ using a motion-compensated interpolation scheme.

3. Features for Segmentation

3.1. Sensitivity. In general, as an object is closer to the flashing system, it reflects stronger light. The irradiance is inversely proportional to the squared distance between the flash and the object. Therefore, the depth information of objects and the background can be inferred from a sensitivity map [28, 29], which represents the ratio of the amounts of light reaching each pixel $\mathbf{p}$ in the unlit frame and the lit frame. More specifically, the sensitivity map is given by

$$S(\mathbf{p}) = \frac{E_{U_i}(\mathbf{p})\Delta t_U}{E_{\hat{L}_i}(\mathbf{p})\Delta t_L}, \tag{6}$$

where E_{U_i} and $E_{\hat{L}_i}$ denote the irradiances of unlit and lit frames, Δt_U and Δt_L are the corresponding exposure times. The amount of light, $E(\mathbf{p})\Delta t$, is obtained using the camera response function $f(\cdot)$ [30]. Since the camera response function is monotonically increasing, the amount of light on pixel $\mathbf{p}$ is derived as

$$\begin{aligned} E_{U_i}(\mathbf{p})\Delta t_U &= f^{-1}(U_i(\mathbf{p})), \\ E_{\hat{L}_i}(\mathbf{p})\Delta t_L &= f^{-1}\left(\hat{L}_i(\mathbf{p})\right). \end{aligned} \tag{7}$$

Figure 3 shows the normalized sensitivity map, obtained from the lit and unlit frames in Figure 2. Note that sensitivities in the background layer are bigger than 1 due to the short exposure time for the lit frame. On the other hand, sensitivities in the objects are less than 1, since the objects react to the flash light strongly.

After computing the sensitivity map, we smooth the sensitivity distribution using a uniform kernel and cluster the sensitivities into two Gaussian distributions using the expectation-maximization (EM) algorithm [31]. The smoothing is performed to prevent local convergence of the EM algorithm. The sensitivity distribution is then modeled by

$$\begin{aligned} &p_{\text{sens}}(S(\mathbf{p})) \\ &\quad = P_0 \cdot p_{\text{sens}}(S(\mathbf{p}) \mid \mu_0, \sigma_0^2) + P_1 \cdot p_{\text{sens}}(S(\mathbf{p}) \mid \mu_1, \sigma_1^2), \end{aligned} \tag{8}$$

where $\mu_0 > \mu_1$, and P_0 and P_1 are the mixing weights of the Gaussian mixture model. Also, $p_{\text{sens}}(S \mid \mu_0, \sigma_0^2)$ denotes the Gaussian distribution with mean μ_0 and variance σ_0^2, and represents the sensitivity distribution of background pixels. Similarly, $p_{\text{sens}}(S \mid \mu_1, \sigma_1^2)$ represents the sensitivity distribution of object pixels. Therefore, $p_{\text{sens}}(S(\mathbf{p}) \mid \mu_0, \sigma_0^2)$ and $p_{\text{sens}}(S(\mathbf{p}) \mid \mu_1, \sigma_1^2)$ can be interpreted as the likelihoods that pixel $\mathbf{p}$ belongs to the background layer and the object layer, respectively.

Since the sensitivity is a main feature for the classification, the exposure time Δt_L for lit frames, which affects the quality of the sensitivity map, should be selected carefully. Note that the exposure time Δt_U for unlit frames is set to record the natural moods and colors of scenes properly. If Δt_L is identical to Δt_U, the intensities of object pixels may be saturated due to the limited dynamic range of the camera, making the sensitivity map unreliable. On the other hand, if the exposure time Δt_L is too short, the pixels in the lit frame may be underexposed. Therefore, in this work, we set the exposure time for lit frames by considering the tradeoff between the saturation and the underexposure problems.

Although the sensitivity is a robust feature for segmentation, there are limitations in separating objects from the background using only the sensitivity map. Since the amount of reflected light is determined not only by the distance from the camera to the object but also by the surface albedo and normals, the sensitivity map does not match the depth information perfectly. Therefore, to achieve more reliable segmentation results, we use color and temporal coherence as additional features.

(a) (b) (c)

Figure 3: (a) Unlit frame, (b) lit frame, and (c) the sensitivity map.

3.2. Color. Colors make it easy to distinguish the layers, since they do not change dramatically between adjacent frames. After segmenting the last unlit frame U_{i-2}, we estimate the probability density functions (pdf's) of colors in the object layer and the background layer, respectively, and use those pdf's to segment the current unlit frame U_i. We regard the sensitivity as an additional color component, and represent colors in the (R, G, B, S) space, where S represents the sensitivity.

Let M_0 and M_1, respectively, denote the set of pixels in the background layer and the object layer in the last unlit frame U_{i-2}. Also, let $\mathbf{x}_\mathbf{p}$ denote the four-dimensional color vector of pixel $\mathbf{p}$. Then, using the sample sets M_0 and M_1, nonparametric color pdf's for the background layer and the object layer, respectively, are estimated by

$$p_{\text{color}}(\mathbf{x} \mid M_\alpha) = \frac{1}{h^d |M_\alpha|} \sum_{\mathbf{p} \in M_\alpha} K\left(\frac{\mathbf{x} - \mathbf{x}_\mathbf{p}}{h}\right), \quad \alpha = 0, 1, \tag{9}$$

where K is a kernel function, h is the bandwidth of the kernel, d is the dimension of $\mathbf{x}$, $|M_\alpha|$ means the number of pixels in M_α, and the label α equals 0 for the background or 1 for the object. Note that $d = 4$ in this work. We use the multivariate Epanechinikov kernel [32], given by

$$K(\mathbf{x}) = \begin{cases} \frac{1}{2} c_d^{-1} (d+2)(1 - \mathbf{x}^t \mathbf{x}) & \text{if } \mathbf{x}^t \mathbf{x} < 1 \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where c_d is the volume of a unit d-dimensional sphere: $c_1 = 2, c_2 = \pi, c_3 = 4\pi/3, c_4 = \pi^2/2$, and so forth. The Epanechinikov kernel is radially symmetric and uni-modal. It has an advantage that it can be calculated more quickly than the Gaussian kernel.

3.3. Temporal Coherence. Unlike image segmentation, video segmentation can exploit temporal coherence between consecutive frames. Most pixel positions in a current frame, which are already classified as the object layer in past frames, tend to belong to the object layer again. Thus, we define the temporal coherence map, which represents the likelihood of each pixel to belong to the background or object layer by counting the number of times that it is labeled as object in past frames [12]. Let $\alpha_\mathbf{p}$ be the binary label of pixel $\mathbf{p}$ in the current frame U_i: $\alpha_\mathbf{p}$ equals 1 if $\mathbf{p}$ is an object pixel, and 0 otherwise. Then, the temporal coherence $T(\alpha_\mathbf{p})$ for each label is defined as

$$T\left(\alpha_\mathbf{p}\right) = \begin{cases} \dfrac{C_{i-2}(\mathbf{p} + \mathbf{v}(\mathbf{p}))}{N}, & \text{if } \alpha_\mathbf{p} = 1, \\ 1 - \dfrac{C_{i-2}(\mathbf{p} + \mathbf{v}(\mathbf{p}))}{N}, & \text{if } \alpha_\mathbf{p} = 0, \end{cases} \tag{11}$$

where $C_{i-2}(\mathbf{q})$ denotes the number of times that pixel $\mathbf{q}$ is classified as the object layer in the N last unlit frames $U_{i-2N}, U_{i-2N+2}, \ldots U_{i-2}$, and $\mathbf{v}(\mathbf{p})$ is the motion vector of pixel $\mathbf{p}$ in the current frame U_i, which is estimated using the method in Section 2. $T(\alpha_\mathbf{p})$ is higher, if $\mathbf{p}$ is assigned the same label as its predecessors.

After the segmentation of the current frame, the accumulated count $C_i(\mathbf{p})$ is updated by

$$C_i(\mathbf{p}) = \begin{cases} \min\{C_{i-2}(\mathbf{p} + \mathbf{v}(\mathbf{p})) + 1, N\}, & \text{if } \alpha_\mathbf{p} = 1, \\ \max\{C_{i-2}(\mathbf{p} + \mathbf{v}(\mathbf{p})) - 1, 0\}, & \text{if } \alpha_\mathbf{p} = 0. \end{cases} \tag{12}$$

4. Segmentation by Energy Minimization

We assign a label, $\boldsymbol{\alpha}_\mathbf{p}$, to each pixel $\mathbf{p}$ in the current frame U_i by minimizing an energy function. Let $\boldsymbol{\alpha}$ denote the label image composed of the pixel labels. The energy function is composed of the sensitivity, color, temporal coherence, and smoothness terms, which impose constraints on the pixel labels.

First, the sensitivity term is defined as

$$E_{\text{sens}}(\boldsymbol{\alpha}) = -\sum_{\mathbf{p} \in U_i} p_{\text{sens}}(S(\mathbf{p})) \log p_{\text{sens}}\left(S(\mathbf{p}) \mid \mu_{\alpha_\mathbf{p}}, \sigma_{\alpha_\mathbf{p}}\right). \tag{13}$$

This sensitivity term indicates that, if pixel $\mathbf{p}$ is labeled or classified as $\alpha_\mathbf{p}$, its sensitivity probability in that class should be higher than that in the other class. Note that $p_{\text{sens}}(S(\mathbf{p}))$ is the Gaussian mixture model of the overall sensitivity distribution in (8), which can be regarded as the reliability of the sensitivity $S(\mathbf{p})$. By incorporating $p_{\text{sens}}(S(\mathbf{p}))$ as a weight in the summation in (13), pixels with more reliable sensitivity values play more important roles in the energy minimization.

Second, the color term is similarly defined as

$$E_{\text{color}}(\boldsymbol{\alpha}) = -\sum_{\mathbf{p}\in U_i} p_{\text{sens}}(S(\mathbf{p}))\log p_{\text{color}}\left(\mathbf{x}_{\mathbf{p}} \mid M_{\alpha_{\mathbf{p}}}\right), \quad (14)$$

which constrains that each pixel should be assigned the label with the higher color probability.

Third, the temporal coherence term is defined as

$$E_{\text{temporal}}(\boldsymbol{\alpha}) = \sum_{\mathbf{p}\in U_i}\left(1 - T\left(\alpha_{\mathbf{p}}\right)\right). \quad (15)$$

The temporal coherence term attempts to reduce outliers by giving a penalty to a pixel, which is assigned a different label from its temporal predecessors.

Finally, the smoothness term enforces the constraint that neighboring pixels of similar intensities should be assigned the same label [33], which is defined as

$$E_{\text{smooth}}(\boldsymbol{\alpha}) = \sum_{\substack{(\mathbf{p},\mathbf{q})\in\mathcal{N}\\ \alpha_{\mathbf{p}}\neq\alpha_{\mathbf{q}}}} \|\mathbf{p}-\mathbf{q}\| \exp\left(-\frac{\left\|\mathbf{z}_{\mathbf{p}}-\mathbf{z}_{\mathbf{q}}\right\|^2}{2\sigma^2}\right), \quad (16)$$

where $\mathcal{N}$ is the set of pairs of 8-adjacent pixels in U_i, $\mathbf{z}_{\mathbf{p}}$ is the (R, G, B) vector of pixel $\mathbf{p}$, and σ is the standard deviation of $\|\mathbf{z}_{\mathbf{p}} - \mathbf{z}_{\mathbf{q}}\|$ over all pairs in $\mathcal{N}$.

The overall energy function is then defined as a weighted sum of the four terms, given by

$$\begin{aligned} E(\boldsymbol{\alpha}) = {} & \omega_{\text{sens}}E_{\text{sens}}(\boldsymbol{\alpha}) + \omega_{\text{color}}E_{\text{color}}(\boldsymbol{\alpha}) \\ & + \omega_{\text{temporal}}E_{\text{temporal}}(\boldsymbol{\alpha}) + E_{\text{smooth}}(\boldsymbol{\alpha}). \end{aligned} \quad (17)$$

The energy minimization is carried out through the graph cut algorithm in [34], which is an effective energy minimization method. The min-cut of a weighted graph provides the segmentation that best separates objects from the background.

5. Real-Time Segmentation

The proposed algorithm, described in Sections 2–4, achieves high quality segmentation results, but its computational complexity is relatively high. In this section, we develop a faster version of the proposed algorithm, which reduces the computational complexity significantly at the cost of slight performance degradation. The faster version can be used in real-time applications.

The proposed real-time segmentation algorithm also employs sensitivity, color, and coherence features. However, the feature computations are simplified as follows.

5.1. Simplified Motion Estimation. To compute the sensitivity map for an unlit frame U_i, we synthesize the corresponding lit frame $\hat{L}_i$ based on the motion-compensated interpolation. Since the motion estimation demands high complexity, it is simplified in the following way. While the global motion estimation and the local motion refinement are performed in Section 2.2, the real-time algorithm carries out the global motion estimation only. Furthermore, it uses the translation model instead of the affine model in (1). The two parameters for the translational motion are also estimated using the optical flow equation in (2) and the least square method. Pixels near object boundaries tend to have high matching errors after the global motion compensation. Thus, we mark those pixels as void and use only the non-void pixels, whose matching errors are less than a threshold, to synthesize $\hat{L}_i$ and compute the sensitivities. The sensitivity distribution in (8) is also obtained and modeled using the EM algorithm, excluding the void pixels.

5.2. Block-Based Color Model. Instead of the four-dimensional color vector (R, G, B, S) in Section 3.2, the real-time algorithm uses the three-dimensional color vector (R, G, B).

Moreover, the color pdf's are estimated at the block level, rather than at the frame level as done in (9). Specifically, we divide each frame into blocks of size 16×16. Let B be a block. Then, we model the color pdf's for the background and the object layers in B by

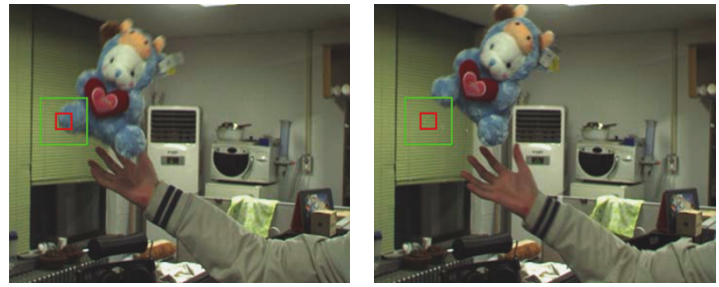
$$\begin{aligned} & p_{\text{color}}(\mathbf{z} \mid B, M_\alpha) \\ & \quad = \frac{1}{h^d |M_\alpha \cap B|}\sum_{\mathbf{p}\in M_\alpha\cap B} K\left(\frac{\mathbf{z}-\mathbf{z}_{\mathbf{p}}}{h}\right), \quad \alpha = 0, 1, \end{aligned} \quad (18)$$

where $\mathbf{z}_{\mathbf{p}}$ is the (R, G, B) vector of pixel $\mathbf{p}$. This equation is the same as (9), except for the reduction of the sample space from the frame to the block and the reduction of the color dimension. To reduce the complexity, a uniform kernel with a narrow bandwidth is employed in (18) and the color distributions for each block are saved as lookup tables.

The block-based color model is spatially adaptive, since it estimates the color pdf's for each block separately. However, when an object moves fast from one block to another, the block-based color model may fail to represent the sudden changes in color distributions correctly, leading to classification errors. To address this problem, we employ the notion of macroblock. Specifically, suppose that we need to use the color model for a pixel. Let B denote the block containing the pixel. Instead of using the color model for B, we use the average of the color models of nine blocks, consisting of B and its eight neighbors. More specifically, the color pdf's of the macroblock, composed of the nine blocks, are estimated by

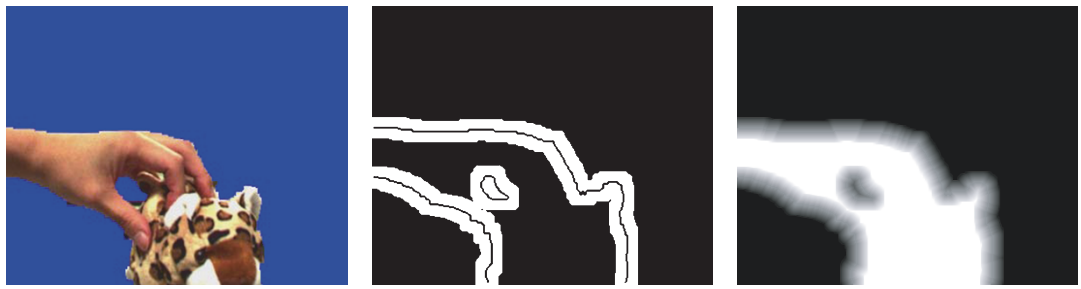
$$p_{\text{color}}(\mathbf{z} \mid B_M, M_\alpha) = \frac{1}{9}\sum_{k\in\mathcal{K}} p_{\text{color}}\left(\mathbf{z} \mid B^k, M_\alpha\right), \quad \alpha = 0, 1, \quad (19)$$

where B_M is the expanded macroblock, and B^k, $k \in \mathcal{K}$, denotes one of the nine blocks. Figure 4 shows two frames, where a green square depicts an expanded macroblock and a red square depicts a block for maintaining the probability look-up tables. By expanding the block size, the color pdf's can be estimated more reliably, even when objects experience fast motions.



(a) (b)

Figure 4: A red square depicts a block, whereas a green square depicts a macroblock. The pdf's for the background and the object layers are estimated for each block separately, but the pdf's for the nine blocks, which compose a macroblock, are averaged for the color modeling.



(a) (b) (c)

Figure 5: (a) Segmentation result, (b) coherence strip, and (c) spatio-temporal coherence map.

5.3. Coherence Strip. The real-time algorithm exploits the property that an object generally does not change its positions abruptly between consecutive frames. Specifically, given the object contour in the previous frame, we construct a coherence strip [13], in which the object contour in the current frame is likely to be located. The notion of coherence strip helps to extract a spatio-temporally coherent video object as well as to reduce the computational complexity.

Figure 5(a) is a segmented object, and Figure 5(b) shows a coherence strip along the object contour. If the object does not move abruptly, the object contour in the next frame tends to be located within the coherence strip. We construct the spatio-temporal coherence map that represents the likelihood that each pixel belongs to the object class. Specifically, the spatio-temporal coherence of a pixel within the coherence strip can be computed by

$$T_{\text{cohere}}\left(\alpha_{\mathbf{p}}=1\right) = \begin{cases} 0.5+\dfrac{d_{\mathbf{p}}}{w}, & \text{if } \mathbf{p} \text{ is inside the object contour,} \\ 0.5-\dfrac{d_{\mathbf{p}}}{w}, & \text{otherwise,} \end{cases} \tag{20}$$

where w is the width of the strip, $d_{\mathbf{p}}$ is the minimum distance of $\mathbf{p}$ from the object contour in frame $(i-1)$, and $0 \le d_{\mathbf{p}} \le w/2$. Also,

$$T_{\text{cohere}}\left(\alpha_{\mathbf{p}}=0\right) = 1 - T_{\text{cohere}}\left(\alpha_{\mathbf{p}}=1\right). \tag{21}$$

Notice that $T_{\text{cohere}}(\alpha_{\mathbf{p}}=1)$ becomes lower, as pixel p moves away from the contour to the outgoing direction. Figure 5(c) shows the spatio-temporal coherence map, where brighter pixels belong to the object layer with higher probabilities.

The pixels inside the coherence strip are classified as object, while the pixels outside the strip as background. Then, the segmentation is performed only on the pixels within the coherence strip, reducing the computational complexity. The width w of the strip should be determined carefully. If the width is too wide, many pixels should be classified, taking a longer processing time. On the other hand, if the width is too narrow, the object contour may move outside the coherence strip, when the object has fast or abrupt motion. Thus, we determine the width w based on the motion vector of the object. Similar to the global motion estimation, the motion vector of the object, $\mathbf{v}$, is estimated using the translational motion model. Finally, we set w to be determined by the magnitude of $\mathbf{v}$ via

$$w = w_{\min} + \|\mathbf{v}\|, \tag{22}$$

where $w_{\min}$ is set to 10 for all experiments in this work. In addition, we translate the coherence strip in (20) by the object motion vector $\mathbf{v}$ [13]. The shifted spatio-temporal coherence is more accurate than (20) especially for an object with fast motion.

Figure 6: Segmentation of the "Dolls" sequence by the proposed algorithm I. The leftmost column shows a pair of unlit and lit frames, while the other columns show a series of unlit frames and their segmentation results.

5.4. Energy Minimization. An energy function is defined over the pixels within the coherence strip, and then minimized using the graph cut algorithm as done in Section 4. By applying the energy minimization to the strip only, instead of the whole frame, the computational complexity is reduced significantly. The sensitivity term is the same as (13), except that the likelihood $p_{\text{sens}}(S(\mathbf{p}) \mid \mu_\alpha, \sigma_\alpha)$ is set to 0 for both classes $\alpha = 0$ and 1, if pixel $\mathbf{p}$ is void and its sensitivity is not estimated. The color term is defined as

$$E_{\text{color}}(\boldsymbol{\alpha}) = -\sum_{\mathbf{p}\in U_i} \log p_{\text{color}}\left(\mathbf{z} \mid B_{M(\mathbf{p})}, M_\alpha\right), \tag{23}$$

where $B_{M(\mathbf{p})}$ denotes the expanded macroblock for pixel $\mathbf{p}$. The spatio-temporal coherence term is defined as

$$E_{\text{cohere}}(\boldsymbol{\alpha}) = \sum_{\mathbf{p}\in U_i} \left(1 - T_{\text{cohere}}\left(\alpha_{\mathbf{p}}\right)\right), \tag{24}$$

using the prior probabilities in (20) and (21). The smoothness term is the same as (16), except that the 8-neighborhood is replaced by the 4-neighborhood. Then, the energy is defined as the weighted sum of the four terms

$$\begin{aligned} E(\boldsymbol{\alpha}) = {} & \omega_{\text{sens}} E_{\text{sens}}(\boldsymbol{\alpha}) + \omega_{\text{color}} E_{\text{color}}(\boldsymbol{\alpha}) \\ & + \omega_{\text{cohere}} E_{\text{cohere}}(\boldsymbol{\alpha}) + E_{\text{smooth}}(\boldsymbol{\alpha}). \end{aligned} \tag{25}$$

6. Experimental Results

The proposed video object segmentation algorithm is implemented in the C++ language on a personal computer with Pentium-IV 3.0 GHz CPU and 2 Gbyte memory. Two versions of the proposed algorithm are implemented: the proposed algorithm I denotes the algorithm described in Sections 2–4, whereas the proposed algorithm II denotes the faster algorithm in Section 5 for real-time applications.

We use several test sequences of CIF size (352×288), captured using the alternate flashing system. As mentioned in Section 2, the sequences are captured with the frame rate of 120 frames/s, and the segmentation is performed only on the unlit frames with the frame rate of 60 frames/s. For the proposed algorithm I, the bandwidth h of the kernel in (9) is set to 2, N in (11) is set to 8, and the weights $\omega_{\text{sens}}, \omega_{\text{color}}, \omega_{\text{temporal}}$ in (17) are fixed to 0.3, 0.32, 0.012, respectively. For the proposed algorithm II, h in (18) is 1, and $\omega_{\text{sens}}, \omega_{\text{color}}, \omega_{\text{cohere}}$ in (25) are 0.2, 0.12, 0.04.

Figures 6, 7, 8, and 9 show the segmentation results on some test sequences, obtained by the proposed algorithm I. In the "Dolls" sequence in Figure 6 and "Throwing Doll" sequence in Figure 7, some parts of the objects have the same colors as the backgrounds. Moreover, there are quick motions in the "Throwing Doll" sequence. However, since the proposed algorithm I estimates the sensitivity maps reliably using the pairs of lit and unlit frames, it segments the objects correctly. The "Moving Object and Background" sequence in Figure 8 contains motions in both background and object layers, but the proposed algorithm I still provides faithful segmentation results. The "Woman in Red" sequence in Figure 9 is the most challenging sequence, since the same color is found across the object contours. The red shirt and the red wall are difficult to separate even with user interactions. However, we see that the proposed algorithm automatically extracts the person from the background and the resulting contours are very accurate.

Next, we examine how the three features, that is, the sensitivity, color, and temporal coherence terms in the energy function in (17), affect the segmentation results of the proposed algorithm I. Figure 10 compares the error rates of the segmentation results on every 5th frame in the sequences in Figures 6, 7, 8, and 9. To compute the error rates, every 5th frame is segmented using the GrabCut algorithm with human interactions [35], and the result is regarded as the ground truth data. Three cases are considered: First, only the sensitivity term among the three features is minimized. Second, the sensitivity term and the color term are minimized together. Third, all three terms are considered. In all these cases, the smoothness term is included in the energy minimization. When only the sensitivity is considered, the average error rate over all frames and all sequences is about 0.75%. The average error rate is reduced to 0.51% if the color term is combined with the sensitivity term. It is further reduced to 0.50% if all three terms, including the temporal

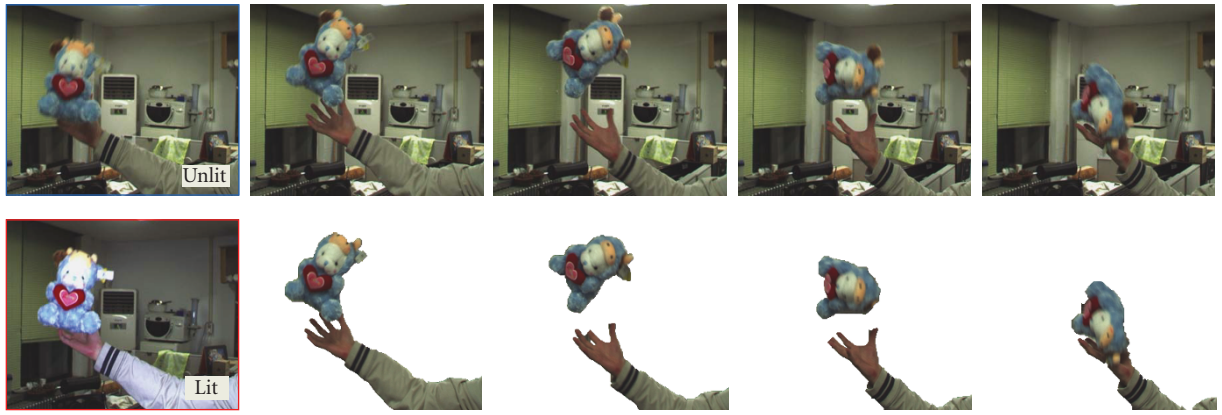


Figure 7: Segmentation of the "Throwing Doll" sequence by the proposed algorithm I. The leftmost column shows a pair of unlit and lit frames, while the other columns show a series of unlit frames and their segmentation results.

Figure 8: Segmentation of the "Moving Object and Background" sequence by the proposed algorithm I. The leftmost column shows a pair of unlit and lit frames, while the other columns show a series of unlit frames and their segmentation results.

Figure 9: Segmentation of the "Woman in Red" sequence by the proposed algorithm I. The leftmost column shows a pair of unlit and lit frames, while the other columns show a series of unlit frames and their segmentation results.

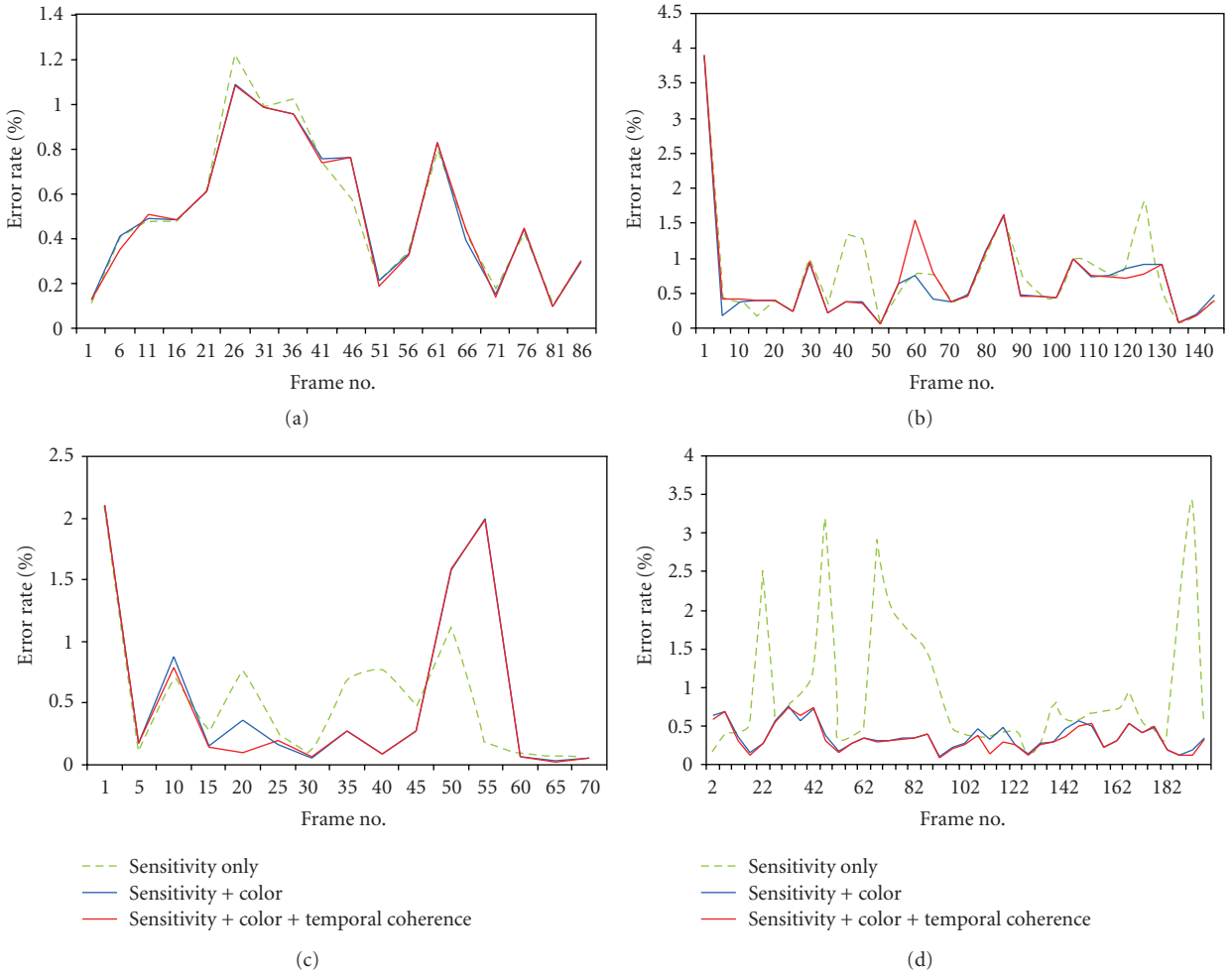


Figure 10: The error rates of the proposed algorithm I on the (a) "Dolls," (b) "Throwing Doll," (c) "Moving Object and Background," and (d) "Woman in Red" sequences. Each curve corresponds to a different combination of features. In every curve, the smoothness term is included in the energy minimization.

Figure 11: Segmentation results of the proposed algorithm I on the "Moving Object and Background" sequence: (a) sensitivity term only, (b) sensitivity and color terms, and (c) sensitivity, color, and temporal coherence terms.

coherence term, are considered. The reduction of the average error rate due to the temporal coherence term appears minor. However, the temporal coherence term contributes to more reliable segmentation, as illustrated in Figures 11 and 12. When only the sensitivity term is employed, some pixels are misclassified as indicated by red ellipses in Figures 11 and 12. Those misclassified pixels are corrected, when the color term and the sensitivity term are included in the energy minimization. In these examples, the background pixels, which are misclassified as object pixels, are corrected by adding the color term, and the object pixels on the dark hairs are correctly classified only after the temporal coherence

(a) (b) (c)

Figure 12: Segmentation results of the proposed algorithm I on the "Woman in Red" sequence: (a) sensitivity term only, (b) sensitivity and color terms, and (c) sensitivity, color, and temporal coherence terms.

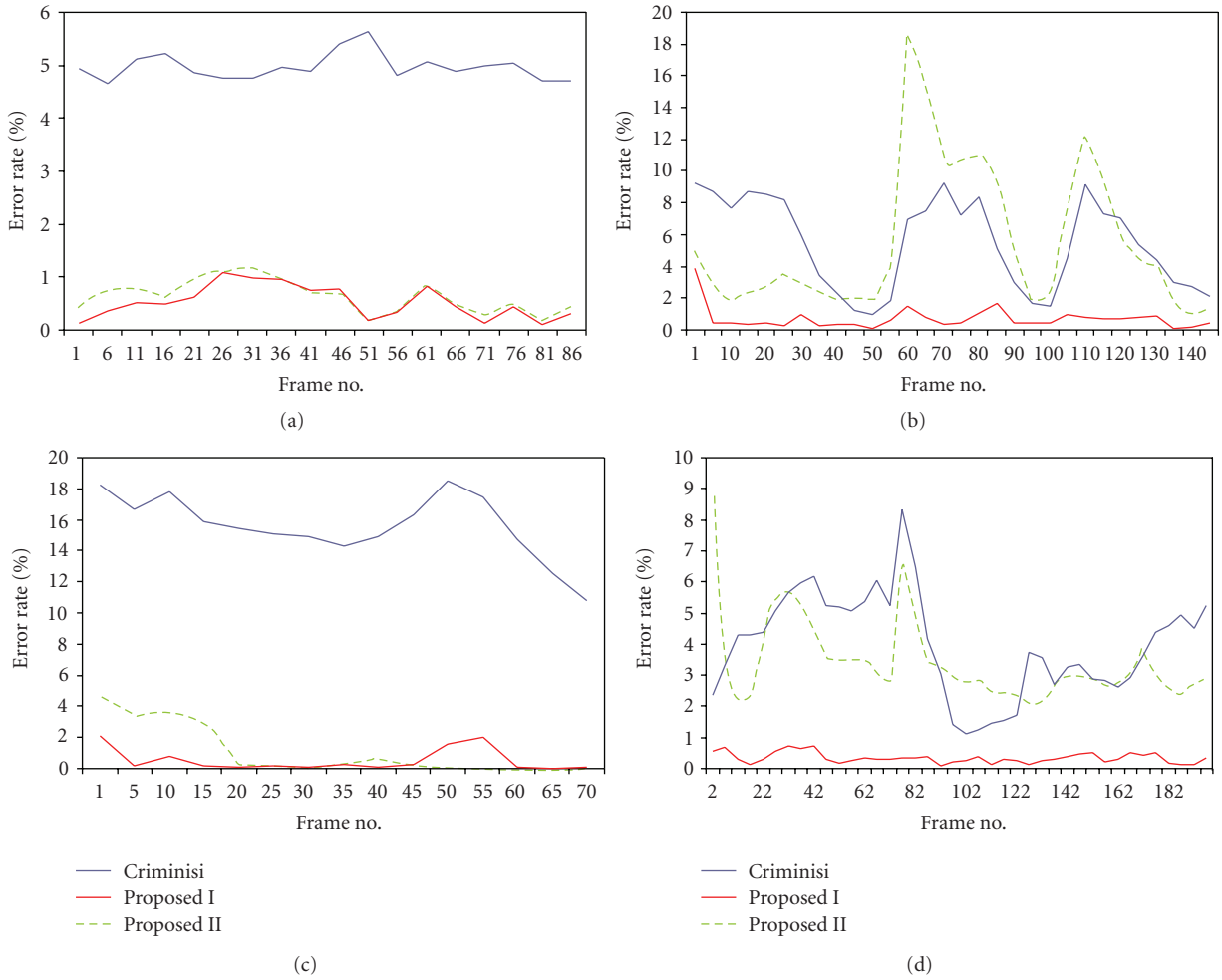


Figure 13: Comparison of the error rates on the (a) "Dolls," (b) "Throwing Doll," (c) "Moving Object and Background," and (d) "Woman in Red" sequences.

term is included. These results indicate that the proposed algorithm I achieves reliable segmentation by combining the three features efficiently.

Figure 13 compares the error rates of the proposed algorithm I with those of the Criminisi et al.'s algorithm [11] and the proposed algorithm II. The Criminisi et al.'s algorithm is one of the state-of-the-art segmentation methods. It constructs a model, composed of spatial prior, temporal prior, motion, contrast, and color cues, which are trained from the ground truth data for every 10th frame. Notice that the Criminisi et al.'s algorithm does not use the information in lit frames, but it implicitly requires human interactions since it uses ground truth data periodically. It does not provide reliable segmentation results, especially when an

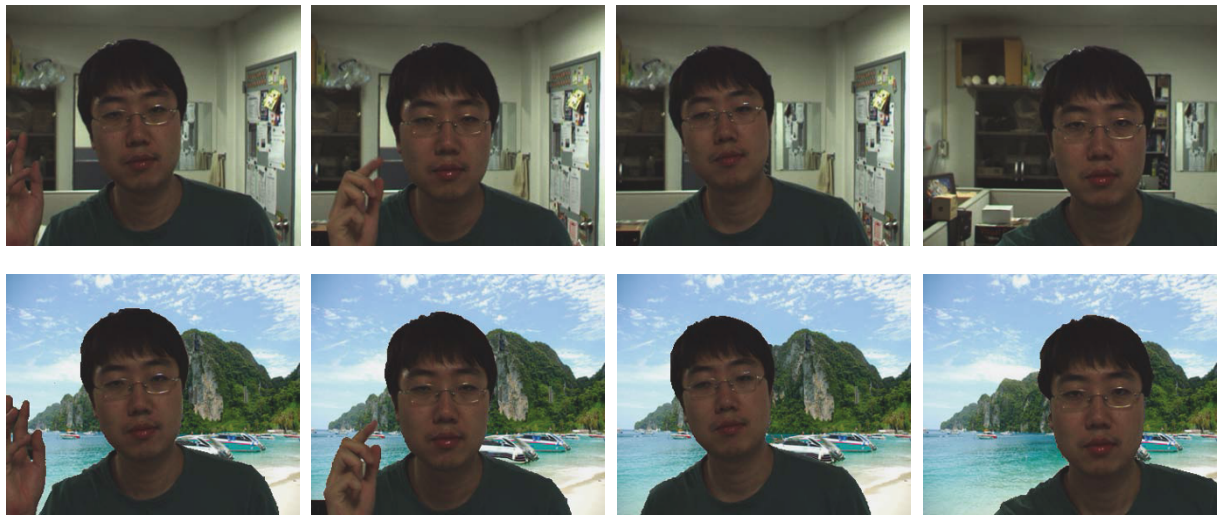


Figure 14: Background substitution. The 1st and 3rd column shows original frames, and the 2nd and 4th columns show the synthesized frames with the replaced background. The proposed algorithm I is used for the segmentation.

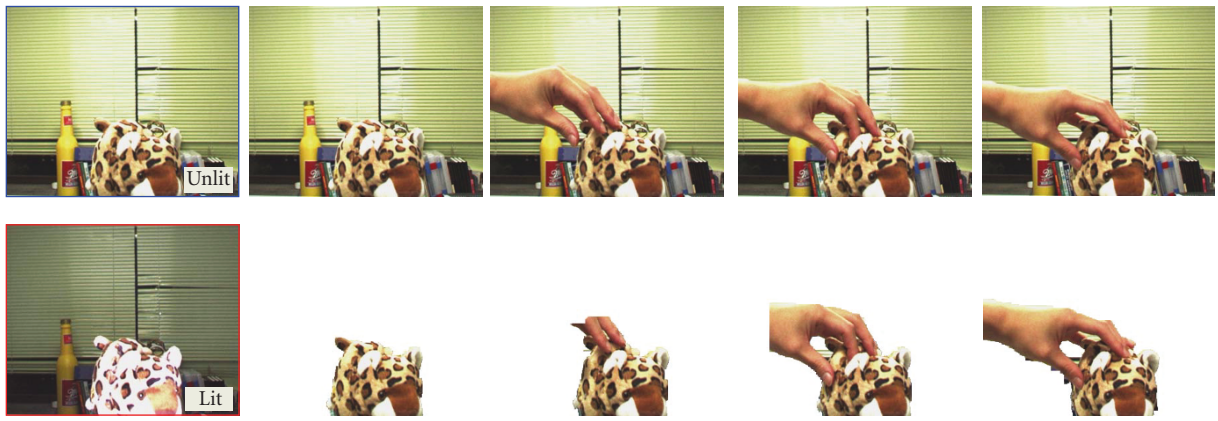


Figure 15: Segmentation of the "Tiger" sequence by the proposed algorithm II. The leftmost column shows a pair of unlit and lit frames, while the other columns show a series of unlit frames and their segmentation results.

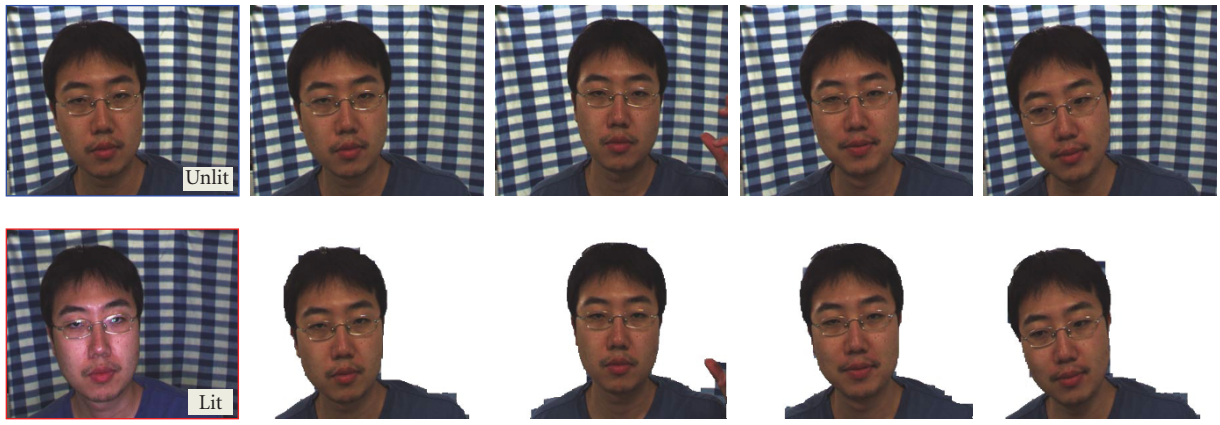


Figure 16: Segmentation of the "Man in Blue" sequence by the proposed algorithm II. The leftmost column shows a pair of unlit and lit frames, while the other columns show a series of unlit frames and their segmentation results.

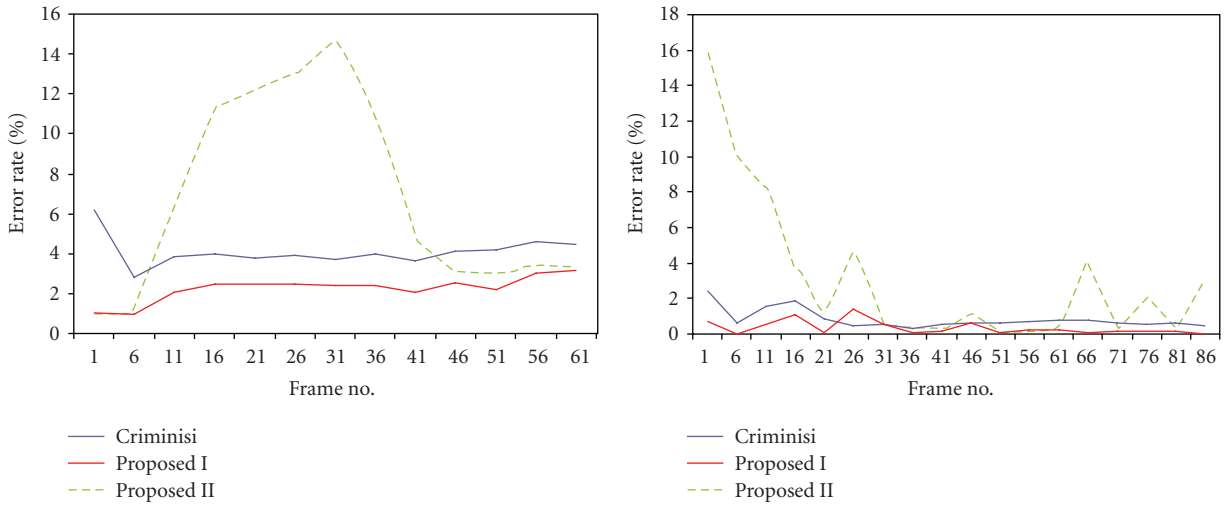


Figure 17: Comparison of the error rates on the (a) "Tiger" and (b) "Man in blue" sequences.

object and the background have the same color across their boundary. From Figure 13, we see that the proposed algorithm I consistently outperforms the Criminisi et al.'s algorithm. We also observe that the proposed algorithm II provides comparable or lower error rates than the Criminisi et al.'s algorithm. However, it yields worse performance than the proposed algorithm I, since it approximates several procedures to reduce the computational complexity.

Since the proposed algorithm provides accurate segmentation results, it can be employed in various applications. As a simple example, we substitute the background of a test sequence in Figure 14. The background substitution can be used to protect privacy in mobile video communications, when a user does not want to reveal the background.

Figures 15 and 16 show the segmentation results of the proposed algorithm II. These sequences are captured with the frame rate of 30 frames/s, and only the unlit frames with the frame rate of 15 frames/s are segmented. In the "Tiger" sequence in Figure 15, some fingers and the background near the hand are not correctly classified, since they move very fast beyond the coherence strip. However, those pixels are finally classified correctly in the subsequent frames. The "Man in Blue" sequence in Figure 16 is more accurately segmented. In our PC implementation, the processing speed of the proposed algorithm II is about 15.5 frames/s. Thus, unlit frames can be segmented in real time, while they are captured. Figure 17 compares the error rates on the sequences in Figures 15 and 16. It shows similar tendency to Figure 13.

7. Conclusions

In this paper, we proposed an automatic video segmentation algorithm, which can provide high quality results using the alternate flashing system. By comparing unlit frames with lit ones, the proposed algorithm obtains the sensitivity map indicating depth information. The proposed algorithm also obtains the color pdf's for the object and the background layers, and constructs coherence likelihoods. By minimizing the energy function, composed of the sensitivity, color, coherence, and smoothness terms, the proposed algorithm obtains accurate segmentation results. Moreover, we developed a faster version of the proposed algorithm, which reduces the computational complexity significantly to achieve real-time segmentation. Experimental results on various test sequences demonstrated that the proposed algorithm provides reliable and accurate segmentation results.

Acknowledgments

This paper was supported partly by Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (2009-0083495), and partly by Seoul R & BD Program (no. ST090818).

References

[1] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 90–126, 2006.

[2] R. Plänkers and P. Fua, "Articulated soft objects for multiview shape and motion capture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 9, pp. 1182–1187, 2003.

[3] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 569–577, 2003.

[4] A. Agarwal and B. Triggs, "3D human pose from silhouettes by relevance vector regression," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '04)*, pp. 882–888, 2004.

[5] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas, "Discriminative density propagation for 3D human motion estimation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, pp. 20–25, 2005.

[6] Y. Cui and J. Weng, "Appearance-based hand sign recognition from intensity image sequences," *Computer Vision and Image Understanding*, vol. 78, no. 2, pp. 157–176, 2000.

[7] P. Song, H. Yu, and S. Winkler, "Vision-based 3D finger interactions for mixed reality games with physics simulation," *International Journal of Virtual Reality*, vol. 8, no. 2, pp. 1–6, 2009.

[8] L. Wang, T. Tan, H. Ning, and W. Hu, "Silhouette analysis-based gait recognition for human identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1505–1518, 2003.

[9] A. Kale, A. Sundaresan, A. N. Rajagopalan, et al., "Identification of humans using gait," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1163–1173, 2004.

[10] J. Sun, W. Zhang, X. Tang, and H. Shum, "Background cut," in *Proceedings of the European Conference on Computer Vision*, pp. 628–641, 2006.

[11] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov, "Bilayer segmentation of live video," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp. 53–60, 2006.

[12] Z. Liu, L. Shen, Z. Han, and Z. Zhang, "A novel video object tracking approach based on kernel density estimation and Markov random field," in *Proceedings of the 14th IEEE International Conference on Image Processing (ICIP '07)*, pp. 373–376, 2007.

[13] J.-K. Ahn and C.-S. Kim, "Real-time segmentation of objects from video sequences with non-stationary backgrounds using spatio-temporal coherence," in *Proceedings of the International Conference on Image Processing (ICIP '08)*, pp. 1544–1547, San Diego, Calif, USA, October 2008.

[14] S.-Y. Chien, S.-Y. Ma, and L.-G. Chen, "Efficient moving object segmentation algorithm using background registration technique," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 7, pp. 577–586, 2002.

[15] Y. Tsaig and A. Averbuch, "Automatic segmentation of moving objects in video sequences: a region labeling approach," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 7, pp. 597–612, 2002.

[16] P. Yin, A. Criminisi, J. Winn, and I. Essa, "Tree-based classifiers for bilayer video segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'07)*, 2007.

[17] G. Zhang, J. Jia, W. Xiong, T.-T. Wong, P.-A. Heng, and H. Bao, "Moving object extraction with a hand-held camera," in *Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV '07)*, 2007.

[18] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, "Bi-layer segmentation of binocular stereo video," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, pp. 407–414, 2005.

[19] J. Sun, S. B. Kang, Z.-B. Xu, X. Tang, and H.-Y. Shum, "Flash cut: foreground extraction with flash and no-flash image pairs," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'07)*, 2007.

[20] J. Sun, Y. Li, S. B. Kang, and H.-Y. Shum, "Flash matting," *ACM Transactions on Graphics*, vol. 25, no. 1, pp. 772–778, 2006.

[21] H. Li and K. N. Ngan, "Unsupervised video segmentation with low depth of field," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 12, pp. 1742–1751, 2007.

[22] Point Greay Research, "Triclops on-line manual," http://www.ptgrey.com/.

[23] H. Baker, N. Bhatti, D. Tanguay, et al., "Understanding performance in coliseum an immersive videoconferencing system," *ACM Transactions on Multimedia Computing, Communications, and Applications* , vol. 1, no. 2, pp. 190–210, 2005.

[24] L. Gharai, C. Perkins, R. Riley, and A. Mankin, "Large scale video conferencing: a digital amphitheater," in *Proceedings of the 8th International Conference on Distributed Multimedia Systems*, 2002.

[25] S. Soumare, A. Ohya, and S. Yuta, "Real-time obstacle avoidance by an autonomous mobile robot using an active vision sensor and a vertically emitted laser slit," in *Intelligent Autonomous Systems 7*, IOS Press, 2002.

[26] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1–3, pp. 185–203, 1981.

[27] A. Smolić, T. Sikora, and J.-R. Ohm, "Long-term global motion estimation and its application for sprite coding, content description, and segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 8, pp. 1227–1242, 1999.

[28] R. Raskar, K.-H. Tan, R. Feris, J. Yu, and M. Turk, "Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 679–688, 2004.

[29] A. Agrawal, R. Raskar, S. K. Nayar, and Y. Li, "Removing photography artifacts using gradient projection and flash-exposure sampling," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 828–835, 2005.

[30] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *Proceedings of the ACM Conference on Computer Graphics (SIGGRAPH '97)*, pp. 369–378, 1997.

[31] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[32] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Champman and Hall, London, UK, 1986.

[33] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images," in *Proceedings of the 8th International Conference on Computer Vision*, pp. 105–112, 2001.

[34] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[35] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut"—interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, 2004.