Hindawi Publishing Corporation
EURASIP Journal on Image and Video Processing
Volume 2010, Article ID 763920, 10 pages
doi:10.1155/2010/763920

Research Article

Joint Rendering and Segmentation of Free-Viewpoint Video

Masato Ishii, Keita Takahashi, and Takeshi Naemura

Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

Correspondence should be addressed to Masato Ishii, masato@nae-lab.org

Received 7 December 2009; Revised 31 May 2010; Accepted 18 August 2010

Academic Editor: Christophe De Vleeschouwer

This paper presents a method that jointly performs synthesis and object segmentation of free-viewpoint video using multiview video as the input. This method is designed to achieve robust segmentation from online video input without per-frame user interaction and precomputations. This method shares a calculation process between the synthesis and segmentation steps; the matching costs calculated through the synthesis step are adaptively fused with other cues depending on the reliability in the segmentation step. Since the segmentation is performed for arbitrary viewpoints directly, the extracted object can be superimposed onto another 3D scene with geometric consistency. We can observe that the object and new background move naturally along with the viewpoint change as if they existed together in the same space. In the experiments, our method can process online video input captured by a 25-camera array and show the result image at 4.55 fps.

1. Introduction

As a powerful tool for representing 3D visual information of the real world, a technology called image-based rendering has attracted many researchers [1, 2]. This technology can synthesize new images which are observed from arbitrary viewpoints, using multiview images as the input. It can provide realistic 3D experiences thanks to the arbitrary control of the viewpoint (Figure 1). In addition to the viewpoint control, image editing methods such as object segmentation are desired to extend the capacity of image-based rendering.

In this paper, we focus on object segmentation task for image-based rendering. Our method jointly performs synthesis of free-viewpoint images and object segmentation in a combined scheme and aims to achieve real-time processing of online video inputs. The method enables to extract a 3D object from one real scene and to superimpose it onto another 3D (real or CG) scene. Since this composition is achieved in a view-dependent manner, we can observe that both the object and new background move naturally along with the viewpoint change as if they existed together in the same space. Experimental results using our camera array system are presented to show the effectiveness of our method.

The problem to solve in this paper consists of two parts: (a) synthesis of free-viewpoint images, and (b) object segmentation. We perform (a) and (b) in this order for the computational efficiency. If (a) and (b) are executed in the reversed order as in [3], (b) should be applied to each of the input images. Meanwhile, if (a) and (b) are executed in this order, (b) can be restricted to the image resulting from (a), which saves much calculation cost. Moreover, we found that the matching costs calculated through (a) are useful also for (b). Therefore, these values are shared between (a) and (b) in our algorithm. Consequently, (a) and (b) are not conducted individually but are jointly performed in a combined scheme.

The features of the proposed method are as follows. First, matching costs (mentioned later in Section 3) calculated through the free-viewpoint image synthesis process are reused for the following segmentation process. Thereby, we can increase useful cues for the segmentation without additional calculations. Second, in the segmentation process, we adaptively combine the matching cost with the color likelihood depending on the reliability of the matching cost for each pixel. Finally, we adopt the graph cut algorithm [4]

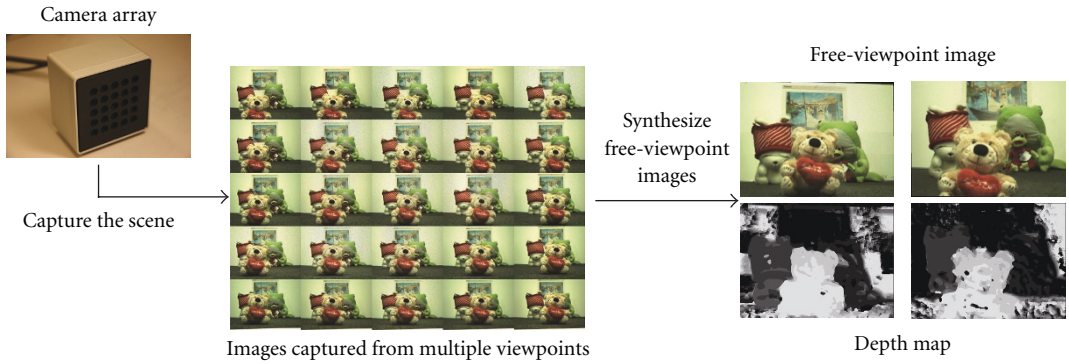

Figure 1: Overview of free-viewpoint image synthesis. The inputs are multiview images captured through a camera array. The outputs are free-viewpoint images, that is, images which are observed from new viewpoints. Depth maps are also calculated behind this process, however, they are unreliable in the less textured regions.

for the segmentation process to obtain globally optimized results.

The remainder of this paper is organized as follows. In Section 2, we describe backgrounds of our work. In Section 3, we present our method which jointly performs synthesis and object segmentation of free-viewpoint images. An overview and results of the experiments are shown in Section 4. Finally, we conclude this paper in Section 5. A preliminary version of this work was presented in [5].

2. Backgrounds

As mentioned in Section 1, the goal of this work is real-time and online segmentation of free-viewpoint video. One may think that depth-keying like methods are straightforward for this goal because multiview images are given as the input. In fact, depth maps are calculated behind the free-viewpoint image synthesis, as shown in Figure 1. However, these depths are unreliable in less textured regions. Although this unreliability is not a significant problem for the purpose of free-viewpoint image synthesis (because the resulting colors of such regions are still correct even if the estimated depths are incorrect to some extent), it is unacceptable for object segmentation. Our method also adopts a depth-related value (matching cost, mentioned later) but fuses it with other information in a probabilistic framework to obtain sufficient results.

In the context of object segmentation, probabilistic approaches are successfully applied to combine multiple cues. Boykov and Jolly [4] introduced an optimization technique using graph cut, which is capable of combining color and contrast cues effectively. For more stable and accurate segmentation, other types of information have been appended to the graph cut algorithm. Criminisi et al. [7] adopted motion cue and temporal prior. Kolmogorov et al. [8] used stereo matching information obtained from stereo cameras. Thermal vision information is used in [9] for extraction of human regions. Our method also uses graph cut algorithm to fuse the matching cost with other information.

The most similar works to ours are found in [8, 10–13]. These works use multiview images as input and combine depth-related cues with object segmentation or matting. The features of our work are as follows.

(1) Object segmentation is obtained directly from an arbitrary viewpoint. In other words, the viewpoint for the segmentation is not restricted to the input viewpoints.

(2) Instead of the explicit depth value, a depth-related evaluation value (referred to as matching cost in this paper) is used for a segmentation cue. Explicit depth estimation is unnecessary in this scheme.

The first feature is common to the methods in [13], the second feature is common to the methods in [8, 10]. However, to our knowledge, there are no works except ours that has both features simultaneously.

The first feature is suitable for online video processing because we need to perform segmentation only once to achieve object segmentation from an arbitrary viewpoint. This is the direct approach to our goal as mentioned in Section 1. Otherwise, we have to iterate segmentation for each of the multiviewpoint images, which will greatly increase the computational cost. The second feature is preferable because accurate depth estimation in real time is far beyond the capability of the current state-of-the-art stereo technologies. However, depth-related values can be easily obtained and used successfully to increase the robustness of the segmentation. Thereby, combining these two features is a reasonable choice to increase the speed and robustness in object segmentation of free-viewpoint video.

3. Algorithm

The input to our algorithm is a set of images captured from a 2D array of cameras. The purpose is to extract a target object from the scene and simultaneously to see the object from arbitrary viewpoints. Our algorithm jointly performs

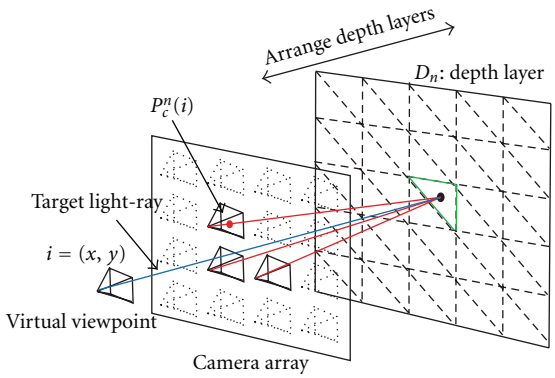

FIGURE 2: Configuration with a certain depth layer for free-viewpoint image synthesis.

synthesis and segmentation of free-viewpoint images. The algorithm consists of the following two steps.

(i) We synthesize an image from an arbitrarily given viewpoint. The matching costs (mentioned later) calculated here are stored for the next step.

(ii) We extract the target object with graph cut. The likelihoods are calculated from the matching cost, pixel color, and temporal coherency. The reliability of the matching cost is evaluated from the local texture and considered in the calculation.

The remainder of this section describes each of the steps.

3.1. Free-Viewpoint Image Synthesis. For synthesis of free-viewpoint images, we adopt the view-dependent rendering algorithm presented in [6].

The configuration is illustrated in Figure 2. The camera array captures input images. The target viewpoint represents the position where we want to synthesize an image. The method locates a set of depth layers over the target scene. Let D_n denote the nth depth layer ($n = 1 \cdots N$), and let i be the index of the pixel position (x, y) on the target image.

For each depth layer D_n, we compute the color $I_n(i)$ and the matching cost $M_n(i)$ as follows:

$$I_n(i) = \sum_{c \in C_i} a_c v_c(P_c^n(i)),$$
$$M_n(i) = \text{smooth}\left(\text{variance}\left(v_c(P_c^n(i))\,|_{c \in C_i}\right)\right). \quad (1)$$

P_c^n represents a matrix transforming a pixel position on the virtual viewpoint into corresponding pixel position on the cth input camera. To perform this transform, we first trace back the light ray associated with each of the pixels on the target image (target light ray in Figure 2) and find the intersection with the nth depth layer. Then we project the intersection point onto the cth input camera to obtain the corresponding pixel position on it. v_c represents the cth input camera, and a_c is its nonnegative weight. C_i denotes a set of cameras referred to for the ith pixel. Those cameras are determined according to the vicinity to the target light ray. In order to reduce the calculation cost, the vicinity is evaluated per triangle mesh as shown in Figure 2. The number of the referred cameras is set to 3 in this paper. smooth$(*)$ is a smoothing filter applied on each depth layer, and 15×15 square kernel is used in this paper.

Let $I(i)$ be the resulting free-viewpoint image. $I_n(i)$ can be regarded as the pixel color of $I(i)$ if there is an object point at the depth D_n, and its confidence can be measured by $M_n(i)$. Therefore, we can synthesize the target image by searching for the minimum of $M_n(i)$ over n for each i and coloring $I(i)$ accordingly as follows:

$$I(i) = I_{n^*(i)}(i), \quad (2)$$

where

$$n^*(i) = \arg\min_{n \in [1,N]} M_n(i). \quad (3)$$

In the above, $n^*(i)$ can be regarded as a depth map as shown in Figure 1. In the original algorithm [6], the minimum search is performed all over the depth range of the target scene, resulting in an all-in-focus image (Figure 3(a)). By contrast, in this work, the range can be limited to the volume of the target object. We can reduce the calculation cost by reducing the number of depth layers. The resulting image is visually correct only around the target object (Figure 3(b)), but that is not a problem because we aim to extract that object from the scene and discard the remains. Here, we assume that we know the depth of the target object. We also assume that there is nothing but the target object in that depth, because if there is another object around the depth of the target object, the matching cost may be useless to discriminate them. To specify the target object, we implement a user interface using synthetic focusing method similarly to [10], which also provides an approximate depth of the focused object.

Through the minimum search in (3), the minimum of $M_n(i)$ is calculated necessarily as follows:

$$M(i) = M_{n^*(i)}(i). \quad (4)$$

As shown in Figure 3(c), $M(i)$ takes relatively small values for the target object. Meanwhile, $M(i)$ is more likely to take larger values for the background except textureless regions. Although $M(i)$ alone is insufficient for the object segmentation, it surely bears useful information. Shown in Figure 4 are the histograms of $M(i)$ for two different frames. The distributions of the object and background are obviously different although they have large overlaps. Consequently, we keep $M(i)$ in order to use it in the following segmentation step. $M(i)$ is referred to as the matching cost in the remainder of this paper.

3.2. Segmentation. The inputs to the segmentation step are the target image viewed from the virtual viewpoint and the corresponding matching costs, both of which are obtained from the first step. In this step, each pixel of the target image is categorized into either "object" or "background". For this purpose, the matching costs are successfully fused with other information in the graph cut algorithm [4]. Figure 5 illustrates the overview of this step.

(a) (b) (c)

Figure 3: Example images of the free-viewpoint synthesis step. (a) is an all-in-focus free-viewpoint image obtained by [6]. (b) and (c) are object-focused free-viewpoint image ($I(i)$ in (2)) and visualized matching cost map ($M(i)$ in (4)) obtained by our method.

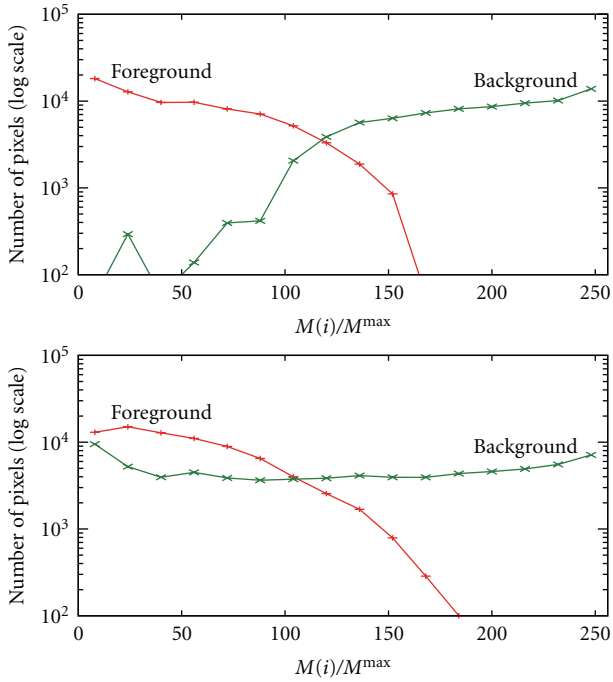

Figure 4: The histograms of the matching cost values. $M^{\max}$ is a positive constant mentioned in (14).

3.2.1. Segmentation Using Graph Cut. First, we make a trimap $\mathbf{T} = \{T(i)\}$ from the segmentation result of the previous frame $\mathbf{S}^* = \{S^*(i)\}$. Each pixel of a trimap takes either "object", "background", or "unknown" as follows:

$$T(i) = \begin{cases} \text{"object"} & \text{if smooth}\left(\hat{S}^*(i)\right) = 255, \\ \text{"background"} & \text{if smooth}\left(\hat{S}^*(i)\right) = 0, \\ \text{"unknown"} & \text{otherwise,} \end{cases} \tag{5}$$

where

$$\hat{S}^*(i) = \begin{cases} 255 & \text{if } S^*(i) = \text{"object"}, \\ 0 & \text{if } S^*(i) = \text{"background"}. \end{cases} \tag{6}$$

Gaussian kernel is used for the smoothing filter in (5). $T(i)$ takes "unknown" near the boundary of the object and background in the previous frame, but the labels ("object"/"background") for the other regions are retained from the previous frame (Figure 6). It implies that the object does not move or deform significantly between the temporally-consecutive two frames. We apply the graph cut process only to the "unknown" regions and their boundaries to reduce the calculation cost as well as to obtain temporally stable segmentations.

Let $\mathbf{S} = \{S(i)\}$ be a binary segmentation of the image, where $S(i)$ is the label for the ith pixel taking either "object" or "background". The graph cut algorithm minimizes the Energy as follows:

$$E(\mathbf{S}) = \sum_i \text{Data}(i, S(i)) + \lambda \cdot \sum_{(i,j)\in N_p, S(i)\neq S(j)} \text{Smth}(i, j), \tag{7}$$

where $\text{Data}(i, S(i))$ represents the data term which evaluates pixel-wise costs, N_p denotes a set of 8-neighboring pixel pairs, and $\text{Smth}(i, S(i))$, referred to as the smoothing term, evaluates interpixel costs. λ is a nonnegative weighting coefficient and was set to 20 in this paper.

The data term is calculated from the object/background likelihoods. The likelihoods can be estimated from various types of cues such as color, motion, and temporal coherency. In this paper, temporal coherency, color, and the matching cost information are fused, which is detailed in Section 3.2.2.

As a typical form of the smoothing term, we use

$$\text{Smth}(i, j) = \frac{e^{-\|I(i)-I(j)\|^2/(2\sigma^2)}}{\|\text{dist}(i, j)\|}, \tag{8}$$

where $\text{dist}(i, j)$ is the distance between the ith and jth pixels. σ is a nonzero constant. This function charges a cost to each of the label changes between the neighboring pixels. The cost decreases as the contrast between neighboring pixel colors increases. As a result, the segmentation boundaries tend to go along with the high-contrast contours.

As mentioned before, only the "unknown" regions and their boundaries are included in the graph. For the boundary pixels, we define fixed likelihood values as follows:

$$\text{Data}(i, S(i)) = \begin{cases} 0 & \text{if } S(i) = T(i), \\ D_{\max} & \text{otherwise,} \end{cases} \tag{9}$$

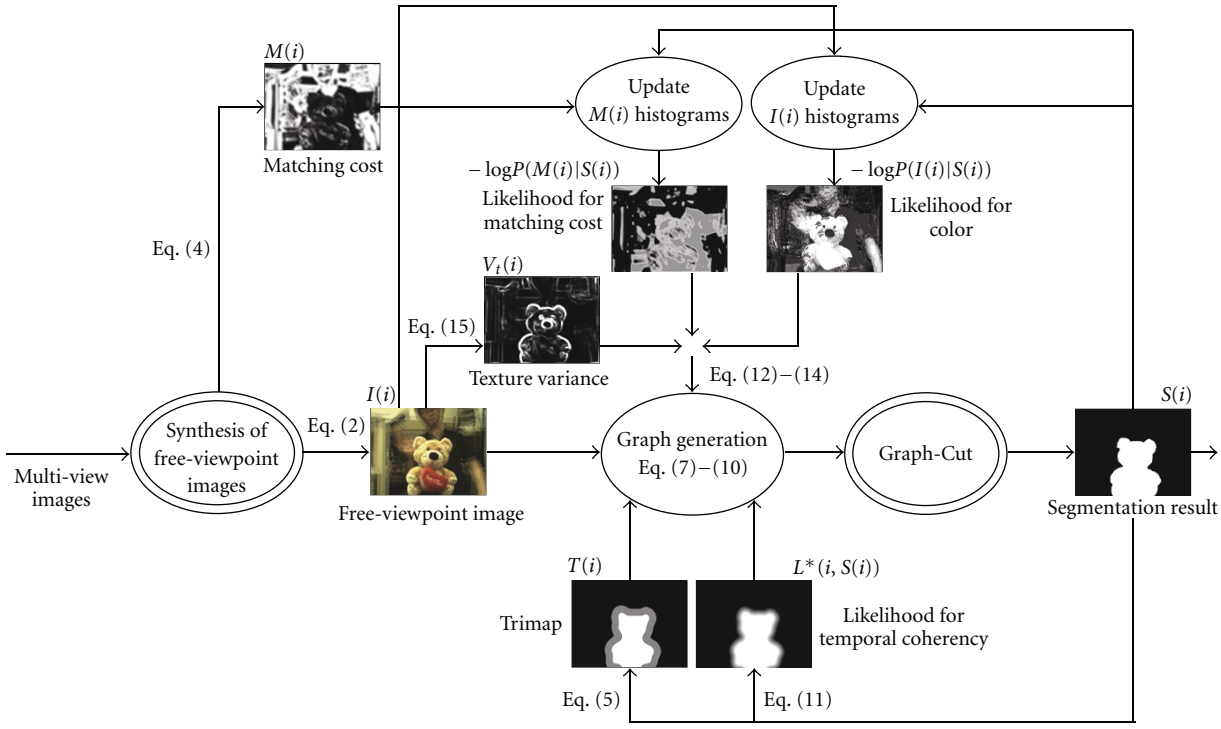

Figure 5: Overview of the proposed method.

where $D_{\max}$ is a sufficiently large constant. Thereby, those pixels are classified correctly into the predefined labels. Meanwhile, the data term for "unknown" pixels is obtained from the procedure in Section 3.2.2.

3.2.2. Detailed Design of the Data Term. For "unknown" pixels, we define the data term as the weighted sum of the likelihood for temporary coherency $L^*(i, S(i))$ and that from the current frame $L(i, S(i))$. $\mu \in [0, 1]$ is a weighting factor

$$\text{Data}(i, S(i)) = \mu L^*(i, S(i)) + (1 - \mu) L(i, S(i)). \tag{10}$$

The likelihood for temporal coherency $L^*(i, S(i))$ is obtained from the segmentation result of the previous frame

$$L^*(i, S(i)) = \begin{cases} -\log\left(\dfrac{\text{smooth}\left(\hat{S}^*(i)\right)}{255}\right) \\ \qquad (S(i) = \text{“object”}), \\ -\log\left(1 - \dfrac{\text{smooth}\left(\hat{S}^*(i)\right)}{255}\right) \\ \qquad (S(i) = \text{“background”}). \end{cases} \tag{11}$$

This calculation process is common to the trimap generation in (6).

The likelihood from the current frame, $L(i, S(i))$ in (10), is calculated from the pixel color $I(i)$ and the matching cost $M(i)$, obtained by (2) and (4), respectively,

$$\begin{aligned} L(i, S(i)) = {} & w(i) \log(P(M(i) \mid S(i))) \\ & + (1 - w(i)) \log(P(I(i) \mid S(i))), \end{aligned} \tag{12}$$

where $w(i)$ is a weighting coefficient and takes pixel-dependent values, which is described later. The conditional probabilities, $P(M(i)|S(i))$ and $P(I(i)|S(i))$, are obtained from the object/background histograms of the matching cost and color, respectively. The histograms are 3D for $I(i)$ (for RGB channels, resp.) and 1-D for $M(i)$. Because we cannot obtain any histograms for the first frame of the video, we need to assume that a ground truth of the first frame is given or to exceptionally define

$$L(i, S(i)) = \begin{cases} -L^{\max} \cdot (S(i) \wedge \text{“background”}) \\ \qquad (M(i) \le \theta), \\ -L^{\max} \cdot (S(i) \wedge \text{“object”}) \\ \qquad (M(i) > \theta), \end{cases} \tag{13}$$

where $L^{\max}$ is a sufficiently large positive constant and θ is a threshold value. After the second frame, we update the histograms based on the segmentation result of the previous frame. We take weighted averages between the current histograms and those constructed from the segmentation result in the ratio of $1 - \alpha : \alpha$. With a large α, the histograms

(a) (b) (c)

Figure 6: Trimap generation, the segmentation result of the previous frame (a) is smoothed (b) and digitalized into 3 values (c): "object" (white), "background" (black), and "unknown" (gray). The graph cut process is applied only to "unknown" regions and their boundaries.

(a) (b)

Figure 7: Free-viewpoint image (a) and its texture intensity (b). The darker pixels are less textured, indicating that the corresponding matching costs are less reliable.

would quickly adjust to dynamic scenes, but they would be unstable. In this paper, α is set to 1/6. The likelihoods for the first frame may be erroneous to some extent, but they become increasingly accurate as the frame proceeds.

One of our main contributions is the way to determine $w(i)$ in (12). As mentioned before, the matching cost $M(i)$ surely contains useful information for the object segmentation but has some unreliability. Therefore, we consider the reliability for each pixel using the local texture information to determine $w(i)$. The matching cost represents the depth correctness for the sufficiently-textured regions, but that is not the case for the weakly-textured regions. For such regions, matching cost is always small regardless of the depth correctness.

Based on the above observation, we determine $w(i)$ as follows:

$$w(i) = \max\left(\frac{M(i)}{M^{\max}}, \frac{V_t(i)}{V_t^{\max}}\right), \tag{14}$$

where $V_t(i)$ is the texture intensity around the ith pixel calculated by

$$V_t(i) = \text{variance}\,(I(i))|_{i\in W_i}, \tag{15}$$

where W_i denotes the pixels in the local region whose center is the ith pixel in the synthesized image, and a 15×15 square window was used in this paper. $M^{\max}$ and $V_t^{\max}$ are positive constants and were set to 255/30 and 255/10, respectively. Equation (14) indicates that the matching cost is reliable if either $M(i)$ or $V_t(i)$ is large; the matching cost is unreliable if and only if both $M(i)$ and $V_t(i)$ are small.

An example of $V_t(i)$ and the corresponding $I(i)$ is shown in Figure 7. The texture intensities are largely different pixel by pixel, and therefore so are the reliabilities. Determining $w(i)$ as in (14) brings pixel-wise adaptivity to (12). $w(i)$ reflects the reliability of the matching cost $M(i)$. Consequently, larger reliability results in a larger impact on the likelihood $P(M(i)|S(i))$.

4. Experiments

We used a camera array that consists of 25 (5 by 5) cameras shown in Figure 1 to capture the input images. These cameras are arranged in a 2D grid array with intervals of 15 mm, and each camera has 640×480 pixels. The camera array is connected to a PC through PCI Express interfaces. We can obtain multiview video from the camera array up to 30 fps. The PC has an Intel 2.66 GHz CPU, 3.0 GByte main memory, and a graphic card with NVIDIA GeForce 8800 GT. The video sequences used for our experiments contain motions; both the object and background are moving.

We implemented the algorithm with the above mentioned setup. First, multiview images are captured from the camera array and rectified using the method presented in [14]. Next, we perform free-viewpoint image synthesis. The

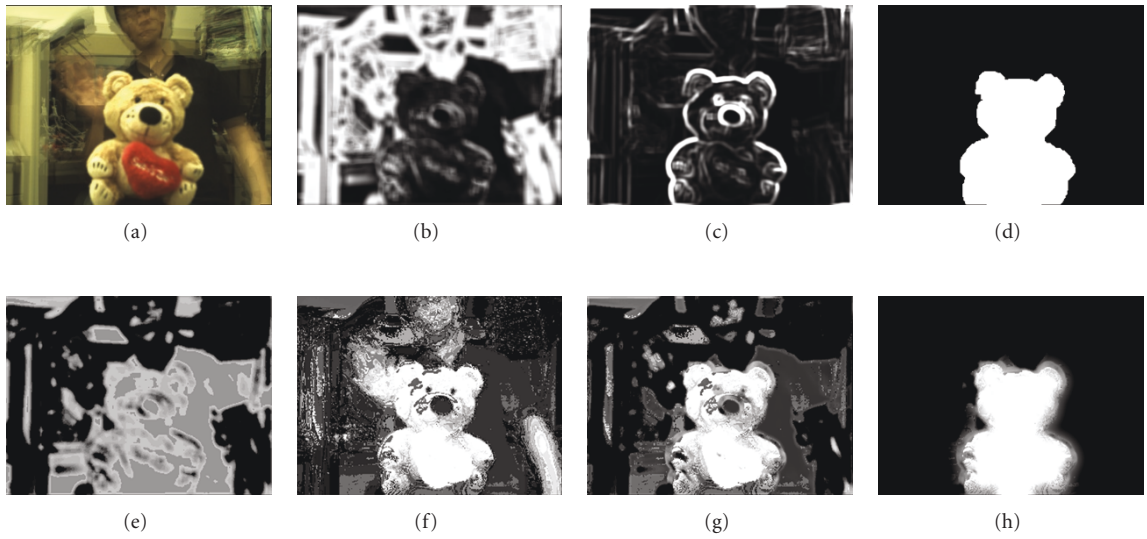

Figure 8: Example images of an experimental result ((a) free-viewpoint image before the segmentation step, (b) matching cost $M(i)$, (c) texture variance $V_t(i)$, (d) the segmentation result, (e) the likelihood for the matching cost, (f) the likelihood for the pixel color, (g) the likelihood combining the matching cost and the pixel color, and (h) the final likelihood to which the temporal likelihood is appended).

viewpoint can be controlled by intuitive mouse operations. The target object can be easily specified via a simple UI we developed (We implemented the synthetic focusing algorithm [15] with a function to control the focused depth by the mouse operation. This interface is intuitive because the target object is specified as the one that is sharply focused in the displayed image. A similar interface was also adopted by [10].) Then, we perform object segmentation from the free-viewpoint images. We use min-cut/max-flow software in [16] to implement the segmentation. The free-viewpoint image synthesis is performed in 640×480 pixels, but the segmentation process is in 320×240 pixels to reduce calculation costs. Finally, the extracted object viewed from the free-viewpoints can be superimposed onto another 3D scene with geometric consistency.

Several video sequences of the experimental results are available from http://www.nae-lab.org/project/FVVSegmentation/.

4.1. Online Segmentation and Background Substitution in 3D. We performed online segmentation and background substitution in 3D using our method. The number of depth layers, the kernel size in (5), and μ in (10) were set to 5, 41, and 0.5, respectively, in this experiment. Equation (13) is used for the first frame.

The total computational time of our algorithm is 128.4 ms for each frame, where 45.9 ms is spent for free-viewpoint image synthesis and the remaining 82.5 ms for object segmentation. (The processing time would be greatly increased if those two procedures were performed in the opposite order; segmentation is applied for each of the 25 input images, and then free-viewpoint image synthesis is performed. The estimated processing time for this scenario is 2108.4 ms ($82.5 \times 25 + 45.9$) for each frame.) An additional 93.6 ms is required for the multiview image capturing and texture uploading. Consequently, the total processing time of our system is 222.0 ms. As a result, we can process online multiview video input in 4.55 fps.

Figure 8 visualizes the procedure of our method for a certain frame. Shown in Figures 8(a) and 8(b) are $I(i)$ of (2) and $M(i)$ of (4). It can be observed that the toy, the target object we specified in this experiment, appears clearly in Figure 8(a), and the corresponding region appears in black in Figure 8(b). The background regions with weak textures are also black in Figure 8(b), and therefore, simple thresholding to $M(i)$ is insufficient for accurate segmentation. Our algorithm estimates the reliability of the matching cost from itself and the texture variance shown in Figure 8(c) according to (14). Depending on the reliability, the likelihoods for color and for matching costs are combined for each pixel adaptively as in (12). Thanks to this scheme, the segmentation is successfully achieved as shown in Figure 8(d).

Figures 8(e)–8(h) show the effectiveness of our likelihood design. Those figures illustrate the difference between Data(i, "object") and Data(i, "background"); the brighter pixel is more likely to be labeled as "object". Figure 8(e) is the likelihood for the matching cost. This likelihood alone is obviously insufficient for the object segmentation, and the method like [10] yields awful results for this scene. Figure 8(f) is the likelihood for the pixel color, which is also insufficient by itself. Figure 8(g) is the likelihood in which the matching cost and pixel colors are combined according to (12). Finally, the temporal likelihood is added by (10) to produce Figure 8(h), which clearly captures the silhouette of the target object.

Figure 9 shows the experimental results for a video sequence. Our method is capable of dealing with dynamic scenes; motions are allowed for the target object and background in the original scene. The extracted object from the target scene is superimposed onto another real scene

Figure 9: Results of background substitution over several frames in video sequences. We can extract the target object from free-viewpoints with a dynamic background. (a) free-viewpoint images before the segmentation step, (b) likelihoods used in the graph cut process, (c) segmentation results, (d) free-viewpoint images for the new background, and (e) composed images. The images in one row belong to the same frame, and time passes from top to bottom.

with geometric consistency. The background scene is also rendered by the method in Section 3.1. We can observe the composed scenes from an arbitrary viewpoint as if both the object and new background exist together in the same space.

4.2. Quantitative Evaluation. Our algorithm assumes that there is no other object around the depth of the target object. Therefore, if accurate depth information is available, object segmentation reduces to a trivial task. However, natural scenes usually contain complex structures and weakly-textured parts, whose accurate depth estimation is very hard even with the most sophisticated computer-vision technologies. To simulate a case in which accurate per-pixel depth information was used for the segmentation, we made a hand-labeled object mask from a free-viewpoint video. This ground truth mask is used as the reference in the following evaluation.

To quantitatively evaluate the performance of our method, we compared four scenarios with different settings of $w(i)$ in (12) as follows:

(i) $w(i)$ is defined by (14). This is the proposed method,

(ii) $w(i) = 0$ for all i. The matching-cost likelihood is not considered,

(iii) $w(i) = 0.5$ for all i. Adaptation of $w(i)$ is turned off,

(iv) $w(i) = 1$ for all i. The color likelihood is not considered.

We stored a multiview video sequence to give the same input to the four scenarios. Figure 10 shows several free-viewpoint video frames. Due to the limited speed of hardware access, the algorithm cannot run in real-time for this setup. In this experiment, we assume that a ground truth of the first frame can be used for the first frame. The number of depth layers, the kernel size in (5), and μ in (10) were set to 8, 21, and 0.3, respectively.

We used F-measure to evaluate the quality of the segmentation results as follows:

$$\begin{aligned}
\text{F-measure} &= \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \\
\text{precision} &= \frac{\sum_i \Big((S(i) = \text{“object”}) \wedge \big(S_g(i) = \text{“object”}\big) \Big)}{\sum_i \big(S_g(i) = \text{“object”}\big)}, \\
\text{recall} &= \frac{\sum_i \Big((S(i) = \text{“object”}) \wedge \big(S_g(i) = \text{“object”}\big) \Big)}{\sum_i (S(i) = \text{“object”})},
\end{aligned} \tag{16}$$

where $S(i)$ is the object/background label for ith pixel produced by the algorithm, and $S_g(i)$ is the ground truth which we manually labeled. F-measure takes a real value between 0 (worst) and 1 (best). We calculated F-measures for every six frames over the free-viewpoint video sequence.

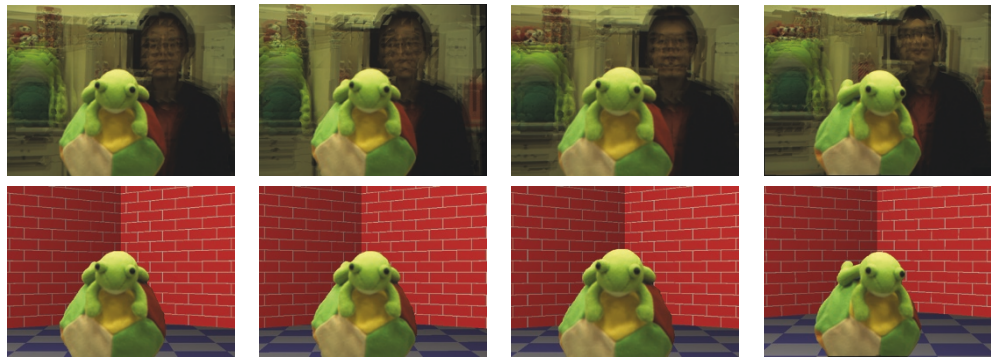

Figure 10: Top: a free-viewpoint video sequence, bottom: composed sequence using ground truth mask.

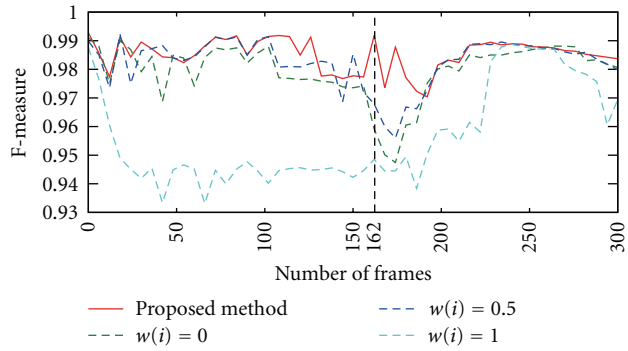

Figure 11: Comparisons of F-measures through the free-viewpoint video sequence.

Figure 12: Examples of final likelihoods and segmentation results. Left: a free-viewpoint image, right: final likelihoods (top) and segmentation results (bottom).

Figure 11 shows F-measures over the video sequence. We can see that the proposed method (scenario (i)) keeps high F-measures (above 0.97) over the whole sequence. Meanwhile, scenario (iv) fails considerably because the matching cost often becomes unreliable due to large textureless regions in the scene. For the same reason, scenario (iii) cannot improve the results over scenario (ii) so much, though scenario (iii) uses both color and matching-cost cues. Scenario (ii) produces relatively good results in this experiment but fails around the frames between 160 and 190. Figure 12 shows the free-viewpoint image, the final likelihoods, and the segmentation results for the 162th frame. The proposed method produces effective likelihoods for the segmentation.

This result supports the effectiveness of using both color and matching-cost cues and adaptive fusion of them according to the reliability.

5. Conclusion

In this paper, we proposed a method that jointly performs synthesis and segmentation of free-viewpoint video. The method consists of two steps: the synthesis step and the segmentation step. In the synthesis step, we synthesize free-viewpoint images using multiview images as the input and store the matching costs which are naturally calculated through the synthesizing process. In the segmentation step,

we adaptively fuse the matching cost with other cues depending on the reliability and perform the graph cut-based algorithm for the segmentation. The reliability is calculated from the matching cost values themselves and from the texture intensity of local regions.

Experimental results show the effectiveness of our method. Our method can process online multiview video input at an interactive rate. The results indicate that the adaptive fusing of the cues is effective to estimate more accurate likelihoods although each of the cues alone is insufficient for this estimation. By superimposing the extracted object onto other free-viewpoint images, we can observe that the object and new background move naturally along with the viewpoint change as if they existed together in the same space.

Our future work will be mainly focused on speedup of the graph cut process as well as overall optimization of our algorithm. In our current implementation, free-viewpoint image synthesis is fast enough thanks to hardware acceleration of GPU. Meanwhile, the graph cut process is not optimized enough in speed. For real-time processing of our algorithm, we will adopt a fast graph cut such as multilevel banded algorithm [17] or graph cut on the GPU [18].

Acknowledgments

This research was supported by National Institute of Information and Communication Technology (NICT), and Special Coordination Funds for Promoting Science and Technology of Japan Science and Technology (JST) Agency.

References

[1] H.-Y. Shum, S. B. Kang, and S.-C. Chan, "Survey of image-based representations and compression techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 11, pp. 1020–1037, 2003.

[2] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multiview imaging and 3DTV," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 10–21, 2007.

[3] S.-C. Chan, Z.-F. Gan, K.-T. Ng, K.-L. Ho, and H.-Y. Shum, "An object-based approach to image/video-based synthesis and processing for 3-D and multiview televisions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 6, pp. 821–831, 2009.

[4] 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 IEEE International Conference on Computer Vision*, vol. 1, pp. 105–112, 2001.

[5] M. Ishii, K. Takahashi, and T. Naemura, "Online segmentation of free-viewpoint video," in *Proceedings of the IEEE International Workshop on 3-D Digital Imaging and Modeling (3DIM '09)*, pp. 1638–1645, 2009.

[6] Y. Taguchi, K. Takahashi, and T. Naemura, "Real-time all-in-focus video-based rendering using a network camera array," in *Proceedings of the 3DTV-Conference: The True Vision—Capture, Transmission and Display of 3D Video*, pp. 241–244, May 2008.

[7] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov, "Bilayer segmentation of live video," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR '06)*, vol. 1, pp. 53–60, 2006.

[8] 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)*, vol. 2, p. 1186, June 2005.

[9] P. V. Quoc, K. Takahashi, and T. Naemura, "Live video segmentation in dynamic background using thermal vision," in *Proceedings of the Pacific-Rim Symposium on Image and Video Technology (PSIVT '09)*, pp. 143–154, 2009.

[10] N. Joshi, W. Matusik, and S. Avidan, "Natural video matting using camera arrays," in *Proceedings of the ACM Transactions on Graphics (ACM SIGGRAPH '06)*, pp. 779–786, 2006.

[11] B. Goldlucke and M. A. Magnor, "Joint 3d-reconstruction and background separation in multiple views using graph cuts," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR '03)*, vol. 1, pp. 683–688, 2003.

[12] J. Y. Guillemaut, A. Hilton, J. Starck, J. Kilner, and O. Grau, "A bayesian framework for simultaneous matting and 3d reconstruction," in *Proceedings of the IEEE International Conference on 3-D Digital Imaging and Modeling (3DIM '07)*, pp. 167–176, 2007.

[13] J. Y. Guillemaut, J. Kilner, and A. Hilton, "Robust graph-cut scene segmentation and reconstruction for free-viewpoint video of complex dynamic scenes," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 809–816, 2009.

[14] N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto, "A novel rectification method for two-dimensional camera array by parallelizing locus of feature points," in *Proceedings of the International Workshop on Advanced Image Technology*, 2008.

[15] A. Isaksen, L. McMillan, and S. J. Gortler, "Dynamically reparameterized light fields," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pp. 297–306, 2000.

[16] 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, pp. 1124–1137, 2004.

[17] H. Lombaert, Y. Sun, L. Grady, and C. Xu, "A multilevel banded graph cuts method for fast image segmentation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV '05)*, vol. 1, pp. 259–265, 2005.

[18] V. Vineet and P. J. Narayanan, "Cuda cuts: fast graph cuts on the gpu," in *Proceedings of the Computer Vision and Pattern Recognition Workshops (CVPRW '08)*, pp. 1–8, 2008.