Pacific Graphics 2009
S. Lee, D. Lischinski, and Y. Yu
(Guest Editors)

Volume 28 (2009), Number 7

Live Video Montage with a Rotating Camera

Zilong Dong, Lei Jiang, Guofeng Zhang, Qing Wang†, Hujun Bao†

State Key Lab of CAD&CG, Zhejiang University
{zldong, jianglei, zhangguofeng, qwang, bao}@cad.zju.edu.cn

Abstract

High-quality video editing usually requires accurate layer separation in order to resolve occlusions. However, most of the existing bilayer segmentation algorithms require either considerable user intervention or a simple stationary camera configuration with known background, which is difficult to meet for many real world online applications. This paper demonstrates that various visually appealing montage effects can be online created from a live video captured by a rotating camera, by accurately retrieving the camera state and segmenting out the dynamic foreground. The key contribution is that a novel fast bilayer segmentation method is proposed which can effectively extract the dynamic foreground under rotational camera configuration, and is robust to imperfect background estimation and complex background colors. Our system can create a variety of live visual effects, including but not limited to, realistic virtual object insertion, background substitution and blurring, non-photorealistic rendering and camouflage effect. A variety of challenging examples demonstrate the effectiveness of our method.

Categories and Subject Descriptors (according to ACM CCS): I.4.6 [Image Processing and Computer Vision]: Segmentation—Pixel classification; I.4.3 [Image Processing and Computer Vision]: Enhancement—Registration

1. Introduction

Over the past decade, video editing has been steadily gaining in importance, due to its wide applications [ST04, ZKU*04, KMTC06, BZS*07, ZDJ*09]. With the increasing prevalence of portable video capturing devices (e.g. a hand-held or web camera), more and more on-line and off-line videos are shared and broadcasted over internet, which can be accessed by the home-users in daily life. It necessitates the development of efficient tools for online video editing and enhancement. Several interactive video segmentation/ matting techniques [CAC*02, LSS05, WBC*05, BWSS09] have been developed. However, all of them require considerable user interaction and the computational cost is expensive. In video matching, Sand and Teller [ST04] proposed to produce spatiotemporal alignment between two videos with similar camera trajectories for applications such as background subtraction, compositing, and increasing dynamic range. Kang et al. [KMTC06] proposed a space-time video montage method for video summarization. Recently, Zhang et al. [ZDJ*09] presented a content-based video editing system for creating various kinds of refilming effects based on dense depth recovery and layer separation. In summary, all these video editing systems can not support real-time processing, and hence can not be applied to online applications.

Some augmented reality (AR) systems [ABB*01, CMPC06] have been successfully developed, which deal with the combination of real-world and computer-generated data (virtual reality) in real-time. However, most of these solutions focus on the geometry consistency of virtual and real scenes and thus require precise motion estimation [CPG01, KM07]. Although there are some studies on resolving virtual-real occlusions [FRB03, KS05, KVKO08], these methods typically require either stationary configuration [KVKO08] or stereo [KS05] cameras, and the scene has to be simple enough for a background subtraction method to work [FRB03].

With a known background, the most efficient bilayer segmentation approach is background subtraction [EHD00, YZPL04], which detects foreground pixels according to color differences. Moving object detection methods [ZS03, YZPL04, RDX07] can be used to effectively detect and seg-

† Correspondence authors: Qing Wang and Hujun Bao

Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

ment out the moving objects from video sequences. However, the segmentation results of these methods are easily error-prone and not accurate enough for high quality foreground extraction in our application.

In order to extract the dynamic foreground objects with high quality, some more sophisticated bilayer separation methods [KCB*05, CCBK06, SZTS06] are proposed by assuming that both the camera and background are mostly stationary. In [KCB*05], color, contrast and stereo matching information are fused to infer the foreground layer from a binocular stereo video sequence in real time. Later on, Criminisi et al. [CCBK06] proposed a method to extract the dynamic foreground with high quality from a single stationary web camera using spatial and temporal priors. Sun et al. [SZTS06] introduced a background contrast attenuation algorithm to reduce the layer extraction errors caused by background clutter. For all these methods, if the camera undergoes unknown motions and the background has complex colors, the foreground object is difficult to be accurately extracted. Zhang et al. [ZJX*07] proposed to combine dense motion and depth estimation to accurately detect and extract moving objects from a video sequence taken by a hand-held camera. However, the computation of this method is time consuming and far away from real-time performance. Recently, Bai et al. [BWSS09] proposed a robust video object cutout system by the collaboration of a set of local classifiers. However, it needs keyframe-based user interaction and the computation is not efficient enough (about one second per frame) for online applications.

In this paper, we propose a novel online video editing system which can create visually appealing montage effects from a live video captured by a rotating camera. Our contributions are summarized as follows. First, we propose a robust feature-based online camera tracking method, which can rapidly retrieve the camera rotation as well as the background image for each online frame. Second, we introduce a novel fast bilayer segmentation method which can extract the dynamic foreground with high quality under rotational camera configuration. Especially, in order to effectively address the problem due to imperfect background estimation and alignment, we develop a novel color model with improved background contrast attenuation algorithm. In order to address the problem that the per-pixel color model may result in the holes in foreground region if the foreground and background colors are very similar, we introduce an effective hole filling method to further improve the robustness. For speed-up, we also introduce a parallel computing framework with multi-scale implementation. Finally, with the retrieved camera parameters, background and foreground images, various visual effects can be created online.

2. Framework Overview

Figure 1 illustrates an overview of our system. Our system employs a parallel computing scheme and contains several modules. The input is a live video captured by a rotating camera. With the precomputed background environment, our system can automatically recover the camera rotation and segment out the dynamic foreground in near real-time. Then, we are able to perform live video montage, such as background substitution/blurring, realistic virtual object insertion, non-photorealistic rendering and camouflage effect.

2.1. Background Modeling and Registration

Our system requires an offline background environment modeling. We need to capture a set of reference images from the static background scene for creating a wide view panoramic image mosaics. Panoramic stitching is a well studied problem. Here, we simply use existing techniques for registration. Specifically, we use a feature-based alignment method [BL03]. We extract and match SIFT features [Low04] among all of the reference images. All matched SIFT features have the similar descriptors [Low04]. For the sake of efficiency, we cluster them and unify their representation by averaging the SIFT descriptors (we use the 64D descriptors). Each of the resulting feature clusters is denoted as feature track $\mathcal{X}$, which contains a series of matched features in multiple frames.

With the matched features, we need to estimate the rotational camera parameters to align the reference images (the intrinsic matrix is assumed to be constant and known). The rotational camera parameters can be reliably estimated by the method in [BL03], and further refined using a global bundle adjustment [TMHF99]. With the camera parameters, we can align the images to a single coordinate system by projecting them onto a cylindrical or sphere surface. Figure 2(a) shows a reconstructed background panoramic image.

While constructing the panorama, we can get a set of feature tracks, which we call *reference features*. Each reference feature has a 64D descriptor and a 3D position corresponding to a 3D ray through the camera center in the reference coordinate system. For efficient searching, a k-d tree is constructed for all reference features with their descriptors.

In the online module, we estimate the camera parameters for each input frame given the captured live video in the same space. For each live frame, we detect the SIFT features, and match them with reference features to obtain a set of 2D-3D correspondences. With these correspondences, the camera rotation can be estimated, and we can warp the panoramic background onto current view to estimate the background. Figure 2 (b) and (c) show one online frame and its recovered background image.

3. Fast Bilayer Segmentation

Let I denote the current online image being processed and I^B its corresponding estimated background image. For each pixel i in image I, its color is denoted as I_i, where the range

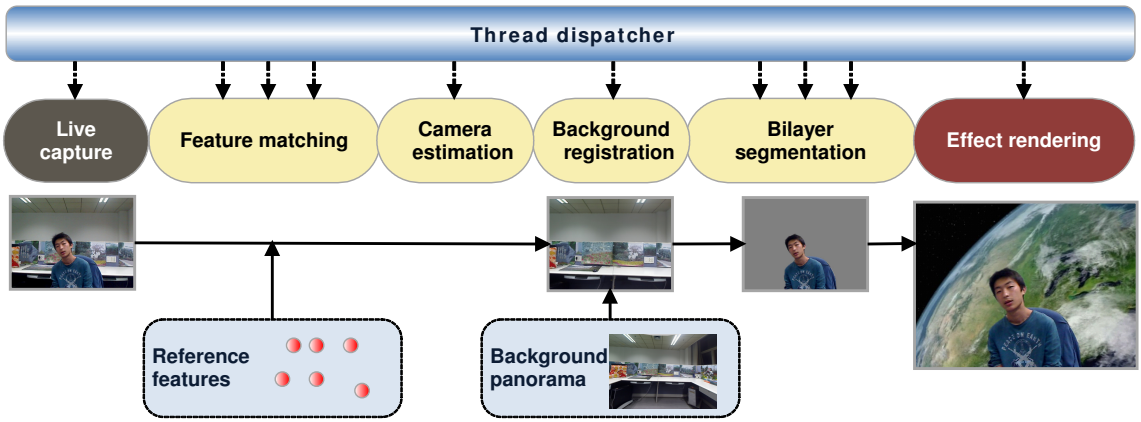


Figure 1: *Framework overview. Our system uses a parallel computing scheme and contains several modules connected by thread-safe buffers. These modules run on separate working threads, and are synchronized by the frame time stamp.*

Figure 2: *(a) The background panorama. (b) and (c) An online frame with the estimated background image.*

of each color channel is [0, 255]. The goal of bilayer segmentation is to estimate a binary variable α_i for each pixel i. $\alpha_i = 1$ if the pixel belongs to the foreground, and $\alpha_i = 0$ if it belongs to the background. Thus the bilayer segmentation can be formulated as a binary labeling problem. The labeling variables $\alpha = \{\alpha_i\}$ can be obtained by minimizing a Gibbs energy $E(\alpha)$:

$$E(\alpha) = \sum_{i \in V} E_d(\alpha_i) + \lambda \sum_{(i,j) \in \mathcal{E}} E_s(\alpha_i, \alpha_j), \qquad (1)$$

where V is the set of pixel nodes in I, and $\mathcal{E}$ is the set of neighboring edges in I. E_d encodes the data cost of pixel i with label α_i, and E_s is the smooth term of edge (i, j) while they are labeled differently as α_i and α_j.

3.1. Data Cost Definition

For each online image, its rotational parameters can be reliably estimated by the method introduced in Section 2.1. Then we can warp the background panorama to the current view to estimate the background image I^B. Considering the global illumination variation, we can estimate the average of intensity scales between the matched features to globally adjust the aligned background image. Due to inevitable estimation error, the warped point may slightly deviate from its correct position. We thus apply a local search algorithm to find the best match to reliably compute background subtraction:

$$\mathcal{S}_i = \min_{j \in W(i)} ||I_i - I^B_j||, \qquad (2)$$

where $W(i)$ is a searching window centered at pixel i, and set to 5×5 in our experiments. For efficiency, we only use gray scale information here.

The data likelihood based on the above improved background subtraction can be defined as follows:

$$\begin{aligned} L_S(I_i|\alpha_i = 0) &= \frac{\mathcal{S}_i^2}{\mathcal{S}_i^2 + \delta^2}, \\ L_S(I_i|\alpha_i = 1) &= \frac{\delta^2}{\mathcal{S}_i^2 + \delta^2}, \end{aligned} \qquad (3)$$

where δ is a threshold to determine whether the pixel belongs to foreground or background. If $S_i > \delta$, then $L_S(I_i|\alpha_i = 0) > 0.5 > L_S(I_i|\alpha_i = 1)$. That is, the pixel i is more likely to be a foreground pixel.

However, such background subtraction model has ambiguity for accurate bilayer segmentation if the foreground pixels have similar colors with the background. Therefore, more cues should be considered.

Gaussian mixture models (GMM) are usually employed in bilayer segmentation algorithms [WBC*05,SZTS06]. The typical implementation is to use known foreground and background pixels to build two GMM models, i.e. one for the colors found in the background and the other for those found in the foreground. Then the global background color model $p(I_i|\alpha_i = 0)$ can be defined as follows:

Figure 3: *Bilayer segmentation result using global GMM model. Because the color distribution of the background is very complex and contains similar colors with foreground, the global GMM method could not perform well.*

$$p_g(I_i|\alpha_i=0)=\sum_{k=1}^{K_b} w_k^B N(I_i|\mu_k^B,\Sigma_k^B), \tag{4}$$

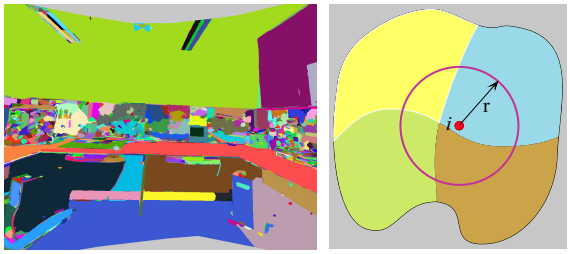
where w_k^B is the weight corresponding to the k^{th} component of the background GMM, and μ_k^B and Σ_k^B are the mean color and covariance of the k^{th} component of the background GMM. Similarly, the global foreground model can be defined as follows:

$$p_g(I_i|\alpha_i=1)=\sum_{k=1}^{K_f} w_k^F N(I_i|\mu_k^F,\Sigma_k^F), \tag{5}$$

where w_k^F is the weight corresponding to the k^{th} component of the foreground GMM, and μ_k^F and Σ_k^F are the mean color and covariance of the k^{th} component.

If foreground and background colors are both simple and distinctive to each other, the above global GMM method usually works well. However, we found that if background and foreground contain complex and similar colors, it is very challenging for global GMM method to perform well. Especially, the component numbers of the foreground and background GMMs should be small and close to each other so that the color distribution could be balanced. Unfortunately, in reality, the background colors may be very complex, especially in outdoor scenes. An example is shown in Figure 3. For this example, using global GMM color model is difficult to obtain a good result.

To tolerate the quick variations in dynamic environment, Zhong et al. [ZYS*08] proposed a GMM based per-region background model, which uses k-means to cluster the pixels in the local region to obtain a local GMM estimation for each pixel. Bai et al. [BWSS09] also used localized color models to improve the classification result. Here, we introduce a novel background color statistics model based on local color clustering, which is more efficient and can be precomputed. We first use Mean Shift algorithm [CM02] to segment the panoramic background image, as shown in Figure 4(a). For each segment S_k, we can estimate a Gaussian distribution. Thus we can obtain a set of background Gaussian distributions $\{N(\mu_1^b,\Sigma_1^b),N(\mu_2^b,\Sigma_2^b),\dots\}$. For each pixel i in the panoramic background, we use m_i to denote its segment label, i.e. $i\in S_{m_i}$. In order to model the background color distribution more robustly, for each pixel, we seek a local estimate of the background color with the clustered Gaussian



(a) (b)

Figure 4: *Background color likelihood estimation using local color model. (a) The segmented panoramic background image by Mean Shift algorithm, where each color represents one segment index. (b) Local Gaussian distribution searching in the local neighborhood area for pixel i. Four Gaussians are found within radius r.*

distributions. An illustration is shown in Figure 4(b). Thus, for each pixel, we sample a group of estimated background pixels from its neighborhood. The local neighborhood area is defined to have a radius of $r=3$ around pixel i. We assume there are l background samples in the local neighborhood area. It can be done in preprocessing when the panoramic background image is reconstructed. Then our background color model can be defined as follows:

$$L_G(I_i|\alpha_i=0)=1-\max_{j=1}^{l} N(I_i|\mu_{m_j}^b,\Sigma_{m_j}^b). \tag{6}$$

With the known foreground samples that are manually labeled in the offline stage, we use standard GMM method to obtain a set of foreground Gaussian distributions $\{N(\mu_1^f,\Sigma_2^f),N(\mu_2^f,\Sigma_2^f),\dots N(\mu_{K_f}^f,\Sigma_{K_f}^f)\}$. The foreground color model can be simply defined as follows:

$$L_G(I_i|\alpha_i=1)=1-\sum_{k=1}^{K_f} w_k^f N(I_i|\mu_k^f,\Sigma_k^f), \tag{7}$$

where w_k^f is the weight corresponding to the k^{th} component of the GMM, and K_f is the component number. Since the foreground colors are usually simple, K_f is generally set to 5 in our experiments.

Combining the background subtraction and local color statistics models, our data cost is finally defined as follows:

$$E_d(I_i|\alpha_i)=\begin{cases} L_S(I_i|\alpha_i) & if\ \ c_i>0 \\ 0.5 & otherwise, \end{cases} \tag{8}$$

where c_i is the labeling consistency check of background subtraction and local color statistics, defined as follows:

$$c_i=(L_S(\alpha_i=0)-L_S(\alpha_i=1))\cdot(L_G(\alpha_i=0)-L_G(\alpha_i=1)).$$

If the labelings by background subtraction and the local color model are consistent, we simply use background subtraction. Otherwise, we just set the data costs of foreground and background the same. The labeling uncertainty will be resolved by spatial smoothness term.

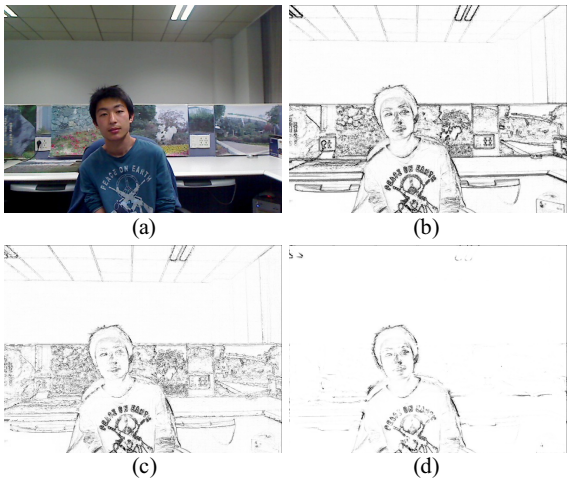


Figure 5: *Background contrast attenuation comparison. (a) One original image. (b) The original contrast image. (c) The background contrast attenuation result using the method proposed in [SZTS06]. (d) The background contrast attenuation result by our method.*

3.2. Spatial Smoothness Term

As indicated in [SZTS06], high contrasts (strong edges) from the background may bias the bilayer segmentation result. However, we found that using the background attenuation method proposed in [SZTS06] can not effectively eliminate the background edges in our examples, as demonstrated in Figure 5(c). The reason is that under the rotating camera configuration, the estimated background image is seldom perfect due to inevitable misalignment problem. Therefore, directly subtracting the contrast of the estimated background image could not effectively attenuate the background contrast.

Here, we introduce a novel background contrast attenuation method which has moderate tolerance to imperfect background estimation and slight misalignment problem. The image gradients can be computed by convolving the grayscale image with Gaussian first derivative filter:

$$\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}) = (I * \frac{\partial G}{\partial x}, I * \frac{\partial G}{\partial y}),$$

where G is a Gaussian function defined as:

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{x^2+y^2}{2\sigma^2}),$$

where σ is a standard deviation, and is set to around 3.0 in our experiments. Figure 7(b) shows the estimated gradient map ∇I.

Thus we can obtain two gradient images ∇I and ∇I^B. Then we use ∇I^B to attenuate the background gradients in ∇I. We apply a gradient distortion scale for the background gradient to make it as close to the foreground gradient as possible. The scale factor should be a positive value and bounded by threshold. Our attenuated gradient magnitude can be computed as follows:

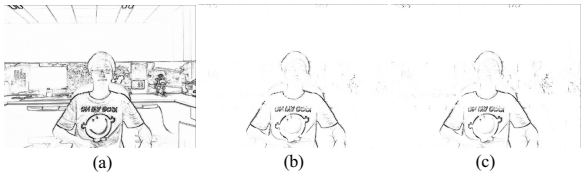


Figure 6: *Background contrast attenuation with different misalignment configurations. (a) The original contrast image. (b) The background contrast attenuation result with 3 pixels misalignment. (c) The background contrast attenuation result with 5 pixels misalignment.*

$$\bar{g} = (1 - \exp(-\frac{(1-\min\{\gamma, 1/\gamma\})^2}{\sigma_\gamma^2})) \cdot \min_\gamma |\nabla I - \gamma \cdot \nabla I^B|, \tag{9}$$

where γ is the scale factor, and computed as follows:

$$\gamma = \min\{\max\{\frac{\nabla I \cdot \nabla I^B}{||\nabla I^B||^2}, 0\}, \tau\},$$

where τ is the threshold. Ideally, γ should be 1 if the pixel is in background region. However, considering the illumination change and blending effect, we change γ in range $[0, \tau]$ to seek a best value to maximumly attenuate the contrast. In our experiments, τ is usually set to 10.

Due to noise interference, the gradient directions might change, so that the background gradients could not be completely eliminated only by $\min_\gamma |\nabla I - \gamma \cdot \nabla I^B|$. Therefore, we further add an attenuation factor in (9) so that gradient could be further attenuated if the best γ is close to 1 since it is very likely that the current pixel is on background. σ_γ is the parameter to adaptively control the attenuation strength according to γ, and set to 0.5 in our experiments.

The above background gradient attenuation method is robust to illumination change as well as small misalignment. Since illumination change would not affect the direction of the gradient, we always can find a best γ to make $|\nabla I - \gamma \nabla I^B|$ near 0 if it is indeed a background pixel. In addition, the gradient is computed by the Gaussian first derivative kernel, which can reduce the problem caused by image misalignment. Especially, we can use a larger Gaussian kernel to smooth the edges so that the strong gradients in I^B and I could be overlapped to attenuate background gradients. Figure 7(c) shows the attenuated gradient map ∇I.

Since the attenuated gradient map $\bar{g}$ has already effectively eliminated the background edges, our spatial smoothness term can be simply defined as follows:

$$E_s(\alpha_i, \alpha_j) = |\alpha_i - \alpha_j| \cdot \exp(-\beta d_{ij}), \tag{10}$$

where i, j are neighboring pixels, and β is a robust parameter that weights the color contrast, and can be set to

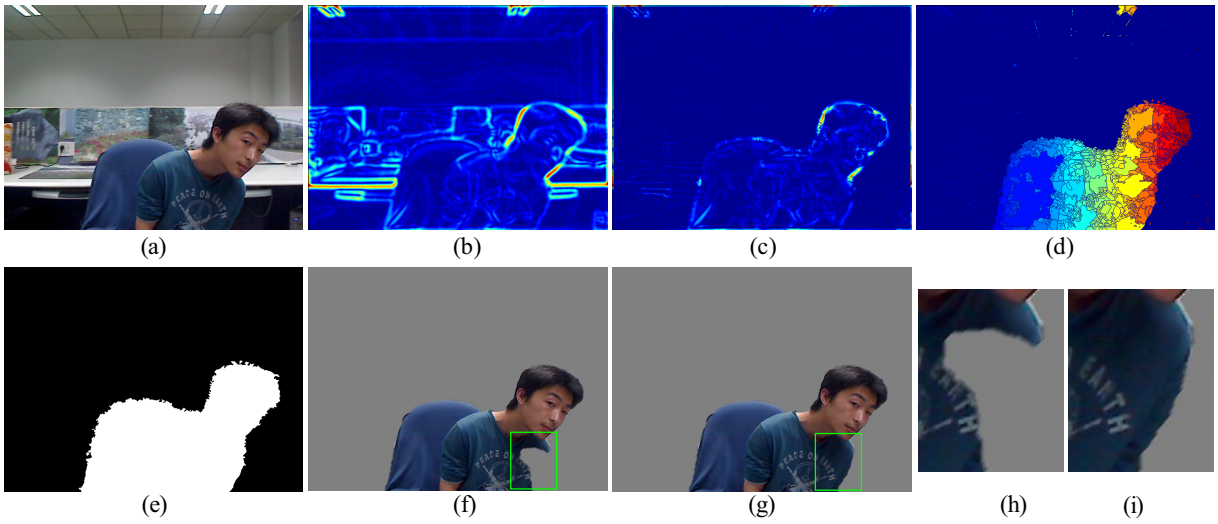


Figure 7: *Background contrast attenuation with foreground hole filling. (a) One original image. (b) The gradient image ∇I. (c) The attenuated gradient image $\bar{g}$. (d) Segmentation result by watershed transformation. (e) The foreground segment candidates. (f) The extracted foreground without foreground hole filling. (g) The extracted foreground combined with foreground hole filling. (h) The magnified view of (f). (i) The magnified view of (g).*

$\beta = (2 < ||I_i - I_j||^2 >)^{-1}$ [SZTS06]. d_{ij} is the attenuation factor, defined as:

$$d_{ij} = \|I_i - I_j\|^2 \cdot \frac{1}{1 + \frac{K^2}{\max\{\|\bar{g}_i\|^2, \|\bar{g}_j\|^2\}}}, \qquad (11)$$

where K is a constant to control the strength of attenuation. $\bar{g}_i$ denotes the attenuated color gradient of pixel i. Our attenuated contrast map is shown in Figure 5(d).

To further demonstrate the effectiveness of our background contrast attenuation algorithm under misalignment configuration, we make an additional experiment by offsetting the background image with different distances. As shown in Figure 6, even with 5 pixels misalignment, our background contrast attenuation algorithm is still effective, which is sufficient in the context of a rotating camera.

3.3. Foreground Hole Filling

In order to minimize the energy $E(\alpha)$ in Equation (1), we use graph cuts [BVZ01] to effectively solve α for bilayer segmentation. Although the above per-pixel model usually produces good results in most frames, if the foreground and background colors are too close, it may result in holes in the foreground object. An example is shown in Figure 7(f). Increasing spatial smoothness only compromises the labeling of one pixel to its neighborhood, but does not help too much to correct large holes. In addition, too strong smoothness term may introduce other distracting artifacts and destroy the fine structures. In order to effectively correct these segmentation errors, we introduce a segmentation-based foreground hole filling method to further improve the bilayer separation result.

With the attenuated gradient map $\bar{g}$, we apply the efficient rain falling watershed algorithm [SP00] to segment the image. The edge threshold value is set to 0.6 to avoid over segmentation of background regions. Figure 7(d) shows the segmentation result.

Because the background gradients are attenuated, most background pixels will be clustered into a very large region. In contrast, the foreground pixels are often over segmented into relatively small regions. Therefore, we only need to consider those small segments which have large chance to be foreground. For these segments, we compute the ratio:

$$\varepsilon = \frac{1}{|S_k|} \sum_{i \in S_k} h(I_i), \qquad (12)$$

where

$$h(I_i) = \begin{cases} 1 & E_d(I_i|\alpha_i = 1) < E_d(I_i|\alpha_i = 0), \\ 0 & E_d(I_i|\alpha_i = 1) \geq E_d(I_i|\alpha_i = 0). \end{cases}$$

For each segment S_k, if $|S_k|/M < 0.03$ (M is the image resolution) and $\varepsilon > 0.2$, it is very likely to be a foreground segment. Figure 7(e) shows the foreground segment candidates. Generally, we can directly label these segments as foreground. Alternatively, based on an observation that the labeling is generally correct if background subtraction and color statistic model judge consistently, a more conservative way is to only directly label the pixels i in these segments that $E_d(I_i|\alpha_i = 1) = E_d(I_i|\alpha_i = 0) = 0.5$ as foreground, and then solve the energy $E(\alpha)$ in Equation (1) to get the final bilayer segmentation. Our foreground hole filling method can effectively reduce the ambiguity due to color similarity, and correct most foreground holes, as shown in Figure 7(g).

Module	Time per frame
Feature matching	≈ 100 ms
Camera estimation	≈ 10 ms
Background registration	≈ 20 ms
Bilayer segmentation	≈ 100 ms
Effect rendering	$5 \sim 30$ ms

Table 1: *Process time per frame for different modules with a single thread.*

4. Implementation and Experimental Results

In this section, we describe the details of our multi-scale implementation and parallel computing, and show several experimental results. In our experiments, indoor videos are captured by a Logitech Quick-Cam Pro 9000 web camera, and outdoor videos are captured by a video camera. The processing frame rate is about 12 fps for a 640×480 video on a desktop PC with Intel(R) Core(TM)2 Quad CPU Q9550 @ 2.83GHz.

4.1. Multi-Scale Implementation

For acceleration, we use a 3-level multi-scale implementation. The first, second and third levels have the resolution of 160×120, 320×240 and 640×480, respectively. SIFT feature detection and matching are performed on the second level. Especially, the image gradients at the coarse level should be computed on the original image to avoid edge blending. While computing the background subtraction in (2), we do not need to search all pixels in the local window $W(i)$. If we find a pixel j satisfying that $||I_i - I_j^B|| < 0.8\delta$, the local search is stopped, and let $\mathcal{S}_i = ||I_i - I_j^B||$. It can greatly save the computation without affecting much the computed data cost. The result at the third/second level is computed in a narrow band (20 pixels width) around the result at the second/first level. After bilayer segmentation, we employ a feathering operation to further improve the object boundary.

The configuration of the parameters in our system is easy. Most parameters are just fixed in our experiments. Specifically, $\lambda = 0.5$, $K = 5$, $\tau = 10$, $\sigma_\gamma = 0.5$. For outdoor videos, we usually set $\delta = 4 \sim 10$. For indoor videos, due to obvious illumination variation in background region caused by foreground occlusion, δ generally needs to be set to a larger value, usually around 22 in our experiments.

4.2. Parallel Computing

Table 1 shows the computation time spent in different modules for one input frame of the indoor cubicle example on a single-core CPU. The time spent on effect rendering depends on the type of visual effect. For most effect rendering, it spends no more than 30 ms. Therefore, the frame rate is around 4 fps on a single-core CPU (bilayer segmentation can operate at 10 fps). For further speed-up, we employ a parallel computing technique using a multi-core CPU to improve the system's performance.

Figure 1 illustrates our parallel computing framework. There are five modules including feature matching, camera estimation, background registration, bilayer segmentation and effect rendering. Our framework contains two parallel hierarchies. First, we build a frame pool which contains several frame buffers. Each frame is assigned at least one thread, so that the frame buffers can be simultaneously computed. Because the system latency (the timestamp difference between the capturing and rendering frames) is related to the computation time of each frame, we further split the frame computation for intra-parallel computing. Since some modules (e.g., feature detection and matching, bilayer segmentation) can be easily parallelized, we assign multiple threads on these modules. With this intra-frame parallelism, the latency can be effectively reduced. On a computer with a 4-cores CPU (Intel(R) Core(TM)2 Q9550 @ 2.83GHz), our system operates at around 12 fps.

4.3. Experimental Results

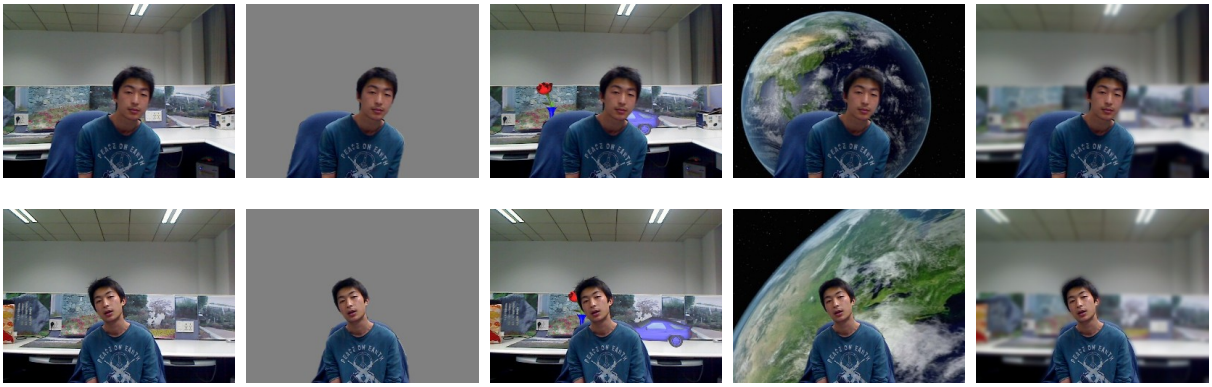
We first show an indoor cubicle example in Figure 8. With the estimated camera parameters, we can insert virtual objects into the scene. With the bilayer separation, the occlusions between the virtual objects and foreground can be accurately resolved (Figure 8(c)). We also can substitute the background, as shown in Figure 8(d). All these effect rendering can be performed in real time.

With the retrieved background and foreground images, we also can filter the background to obtain various visual effects. For example, we can blur the background as shown in Figure 8(e). For acceleration, we can blur the panoramic background image beforehand. Then for online processing, we only need to warp the blurred panoramic background image onto the current frame to create background blurring effect. We also can camouflage the actor by blending the foreground with the estimated background. To further simulate the "predator" effect, we can add refractive and wavy distortion to the blending region (Figure 9).

Figure 9 shows an outdoor example. Another outdoor example is included in our supplementary video. The background colors in these examples are very complex, which are challenging for previous state-of-the-art algorithms to accurately extract the foreground. Based on our robust online camera tracking and foreground extraction, various montage effects can be live created. Please refer to our supplementary video that gives a better presentation of the results.

5. Discussion and Conclusion

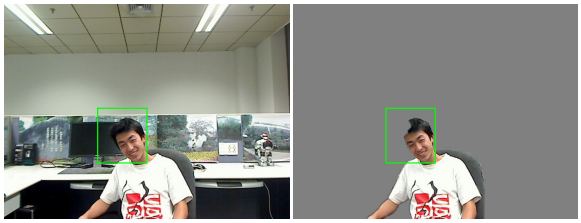
We have presented an online video editing system that allows for creating visually plausible montage effects from a live video captured by a rotating camera. Our system requires an offline background environment modeling which



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

Figure 8: *The results of the indoor cubicle example. (a) The input online frames. (b) The extracted foreground images. (c) Object insertion with occlusion handling. (d) Background substitution. (e) Background blurring.*

Figure 9: *The result of an outdoor example.*



(a) (b)

Figure 10: *A failure example. (a) An online image. (b) The extracted foreground. As highlighted in the green rectangle, the black hair overlaps the black monitor screen, and the overlapping edge is very weak, which has inherent difficulty to accurately separate them.*

reconstructs a panoramic background image from a set of reference background images. Then for each online image, we detect the SIFT features and match them with the reference features, so that the camera as well as background image can be accurately registered. For high-quality foreground extraction, we introduce an efficient and robust bilayer segmentation method which can work well under the rotational camera configuration and outperforms the previous state-of-the-art algorithm. With the retrieved camera rotation, background and foreground images, a variety of effects can be live created.

5.1. Limitations and Future Work

The current system still has some limitations. First, there should be sufficient textured background region for feature detection and matching, otherwise the camera parameters and background image may not be accurately estimated, and the following stages may fail. Second, when the foreground and background colors are too similar, and the foreground edges are also very weak, incorrect separation may happen around these regions, and result in noticeable artifacts for effect rendering. Figure 10 shows a failure example. The black hair overlaps the black monitor screen, which has inherent difficulty for accurate segmentation only using color information. The overlapping edge is also too weak to be accurately detected, which makes our foreground hole filling fail. Using more cues, such as motion and shape priors [BWSS09] may help address this problem. Using temporal coherence also helps improve segmentation quality, but may violate the real-time demand, since it typically requires dense motion estimation and solving multiple frames simultaneously, which will cause obvious latency.

Our system can work well for small global illuminance change by globally adjusting the background image based on color histogram. However, if large illumination change occurs, bilayer segmentation may fail. In this case, we need to re-estimate the background panorama. So one possible direction of our future work is to make the background environment can be updated online so that our system can work well even there is sudden large illumination change. Last, the current system assumes a static background. If the background contains dynamic elements, our method may fail. How to resolve this problem remains to be our future work.

Acknowledgements

The authors would like to thank Prof. Xiaofei He for his enormous help in revising this paper, and the anonymous reviewers for their valuable comments. This work is supported by the 973 program of China (No. 2009CB320804), NSF of China (No. 60633070), and the 863 program of China (No. 2007AA01Z326).

References

[ABB*01] AZUMA R., BAILLOT Y., BEHRINGER R., FEINER S., JULIER S., MACINTYRE B.: Recent advances in augmented reality. *IEEE Computer Graphics and Applications 21*, 6 (2001), 34–47.

[BL03] BROWN M., LOWE D. G.: Recognising panoramas. In *ICCV* (2003), pp. 1218–1227.

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11 (2001), 1222–1239.

[BWSS09] BAI X., WANG J., SIMONS D., SAPRIO G.: Video snapcut: Robust video object cutout using localized classifiers. *ACM Trans. Graph. (Proc. SIGGRAPH 2009)* (2009).

[BZS*07] BHAT P., ZITNICK C. L., SNAVELY N., AGARWALA A., AGRAWALA M., CURLESS B., COHEN M., KANG S. B.: Using photographs to enhance videos of a static scene. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)* (June 2007), Kautz J., Pattanaik S., (Eds.), Eurographics, pp. 327–338.

[CAC*02] CHUANG Y.-Y., AGARWALA A., CURLESS B., SALESIN D., SZELISKI R.: Video matting of complex scenes. *ACM Trans. Graph. 21*, 3 (2002), 243–248.

[CCBK06] CRIMINISI A., CROSS G., BLAKE A., KOLMOGOROV V.: Bilayer segmentation of live video. In *CVPR* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 53–60.

[CM02] COMANICIU D., MEER P.: Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 5 (2002), 603–619.

[CMPC06] COMPORT A. I., MARCHAND E., PRESSIGOUT M., CHAUMETTE F.: Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (July-August 2006), 615–628.

[CPG01] CORNELIS K., POLLEFEYS M., GOOL L. J. V.: Tracking based structure and motion recovery for augmented video productions. In *VRST* (2001), pp. 17–24.

[EHD00] ELGAMMAL A. M., HARWOOD D., DAVIS L. S.: Nonparametric model for background subtraction. In *ECCV* (London, UK, 2000), Springer-Verlag, pp. 751–767.

[FRB03] FISCHER J., REGENBRECHT H., BARATOFF G.: Detecting dynamic occlusion in front of static backgrounds for ar scenes. In *EGVE '03: Proceedings of the workshop on Virtual environments 2003* (New York, NY, USA, 2003), ACM, pp. 153–161.

[KCB*05] KOLMOGOROV V., CRIMINISI A., BLAKE A., CROSS G., ROTHER C.: Bi-layer segmentation of binocular stereo video. In *CVPR* (Washington, DC, USA, 2005), vol. 2, IEEE Computer Society, pp. 407–414.

[KM07] KLEIN G., MURRAY D.: Parallel tracking and mapping for small AR workspaces. In *ISMAR 2007* (Nov. 2007), pp. 225–234.

[KMTC06] KANG H.-W., MATSUSHITA Y., TANG X., CHEN X.-Q.: Space-time video montage. In *CVPR (2)* (2006), pp. 1331–1338.

[KS05] KIM H., SOHN K.: 3D reconstruction from stereo images for interactions between real and virtual objects. *Signal Processing: Image Communication 20* (2005), 61–75.

[KVKO08] KAKUTA T., VINH L. B., KAWAKAMI R., OISHI T.: Detection of moving objects and cast shadows using a spherical vision camera for outdoor mixed reality. In *VRST '08: Proceedings of the 2008 ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2008), ACM, pp. 219–222.

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[LSS05] LI Y., SUN J., SHUM H.-Y.: Video object cut and paste. *ACM Trans. Graph. 24*, 3 (2005), 595–600.

[RDX07] RAO N. I., DI H., XU G.: Panoramic background model under free moving camera. In *FSKD '07: Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 639–643.

[SP00] SMET P. D., PIRES R. L. V. P. M.: Implementation and analysis of an optimized rainfalling watershed algorithm. In *IS&TSPIE's 12th Annual Symposium Electronic Imaging* (2000), pp. 759–766.

[ST04] SAND P., TELLER S. J.: Video matching. *ACM Trans. Graph. 23*, 3 (2004), 592–599.

[SZTS06] SUN J., ZHANG W., TANG X., SHUM H.-Y.: Background cut. In *ECCV* (2006), pp. 628–641.

[TMHF99] TRIGGS B., MCLAUCHLAN P. F., HARTLEY R. I., FITZGIBBON A. W.: Bundle adjustment - a modern synthesis. In *Workshop on Vision Algorithms* (1999), pp. 298–372.

[WBC*05] WANG J., BHAT P., COLBURN A., AGRAWALA M., COHEN M. F.: Interactive video cutout. *ACM Trans. Graph. 24*, 3 (2005), 585–594.

[YZPL04] YANG T., Z.LI S., PAN Q., LI J.: Real-time and accurate segmentation of moving objects in dynamic scene. In *VSSN '04: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks* (New York, NY, USA, 2004), ACM, pp. 136–143.

[ZDJ*09] ZHANG G., DONG Z., JIA J., WAN L., WONG T.-T., BAO H.: Refilming with depth-inferred videos. *IEEE Transactions on Visualization and Computer Graphics 15*, 5 (2009), 828–840.

[ZJX*07] ZHANG G., JIA J., XIONG W., WONG T.-T., HENG P.-A., BAO H.: Moving object extraction with a hand-held camera. In *ICCV* (2007), pp. 1–8.

[ZKU*04] ZITNICK C. L., KANG S. B., UYTTENDAELE M., WINDER S. A. J., SZELISKI R.: High-quality video view interpolation using a layered representation. *ACM Trans. Graph. 23*, 3 (2004), 600–608.

[ZS03] ZHONG J., SCLAROFF S.: Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In *ICCV* (Washington, DC, USA, 2003), IEEE Computer Society, p. 44.

[ZYS*08] ZHONG B., YAO H., SHAN S., CHEN X., GAO W.: Hierarchical background subtraction using local pixel clustering. In *ICPR* (2008), pp. 1–4.