Re-filming with Depth-inferred Videos

Guofeng Zhang[1*], Zilong Dong[1], Jiaya Jia[2], Liang Wan[2], Tien-Tsin Wong[2], and Hujun Bao[1*]

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

[2]The Chinese University of Hong Kong
{leojia, lwan, ttwong}@cse.cuhk.edu.hk

Abstract—Compared to still image editing, content-based video editing faces the additional challenges of maintaining the spatio-temporal consistency with respect to geometry. This brings up difficulties of seamlessly modifying video content, for instance, inserting or removing an object. In this paper, we present a new video editing system for creating spatio-temporally consistent and visually appealing re-filming effects. Unlike the typical filming practice, our system requires no labor-intensive construction of 3D models/surfaces mimicking the real scene. Instead, it is based on an unsupervised inference of view-dependent depth maps for all video frames. We provide interactive tools requiring only a small amount of user input to perform elementary video content editing, such as separating video layers, completing background scene, and extracting moving objects. These tools can be utilized to produce a variety of visual effects in our system, including but not limited to video composition, "predator" effect, bullet-time, depth-of-field, and fog synthesis. Some of the effects can be achieved in real-time.

Index Terms—video editing, re-filming, depth estimation, composition, background completion, and layer separation.

1 Introduction

The wide availability of portable video capturing devices allows home-users to access the image/video contents in daily life. This can be evidenced by the fact that more and more home videos are shared and broadcasted over internet. Nevertheless, compared to the advancement of the image editing algorithms (e.g., inpainting, segmentation, and matting), the development of the content-based video editing is still left far behind in terms of the diversity, practicability, and user-friendliness. A major difficulty comes from the multi-frame nature of a video that requires high temporal consistency over frames. Unfortunately, such consistency is widely known as challenging to maintain due to the difficulty in acquiring the accurate geometry.

In the film industry, the typical solution to creating a visually plausible video editing result is to use specially designed equipments and set up a user-controlled environment. The typical configurations include blue-screen background and motion capture. To enable the modification of video content, 3D models are usually constructed, rendered with carefully tuned lighting, and overlaid onto the video. All these procedures involve the manual intervention by the skilled professionals. Attempts have been made recently in [1] to design a more user-friendly video editing system for interactively constructing 3D models. However, creating a 3D model with sufficient geometry details, e.g. a tree with many leaves, is still intractable, as a large number of fine objects need to be modeled from sparse feature points.

In this paper, we propose a new system for editing casual videos without explicitly reconstructing 3D geometry models. The input single or multiple video clips are allowed to be taken by a hand-held moving camera. Our system contributes in the following respective areas. We describe an efficient level-expansion algorithm to increase the precision of the depth estimates obtained from the algorithm of Zhang et al. [2]. The output from this step is a set of temporally-consistent dense depth maps that are accurate enough to maintain sharp object boundaries. Based on these depth estimates, we introduce a few elementary video editing tools, allowing the user to locally modify the object color and structures.

The first tool is for inferring the missing depth and color of the background pixels through an information propagation process over the video frames. Then we introduce a robust moving object (sprite) extraction method taking account of both the depth and color information. The depth of the extracted sprite is also inferred and represented by a 3D plane. Finally, we describe a method to naturally separate the static background into fine layers. With these tools, a spectrum of special effects such as depth-of-field, fog synthesis, view interpolation (e.g., bullet-time effect), "predator" effect (camouflaging effect), and video composition, can be created. Figure 1 shows a set of these re-filming effect examples. Our semi-automatic depth estimation in the system makes visual effect generation not in a frame-by-frame manner, and thus reduces the possible user interactions. It benefits not only professionals but also novice home-users to produce visually appealing effects without requiring significant cost and manpower.

2 Related Work

In video editing, several interactive image/video segmentation/matting techniques [3], [4], [5], [6], [7], [8], [9], [10], [11] have been developed. Most of them only use the color information or require special camera configurations. In video matching, Sand and Teller [12] proposed to produce spatio-temporal alignment between two videos following spatially similar camera trajectories. The 2D motion information is used in this method to align video frames. This method is not

Corresponding authors: Guofeng Zhang and Hujun Bao

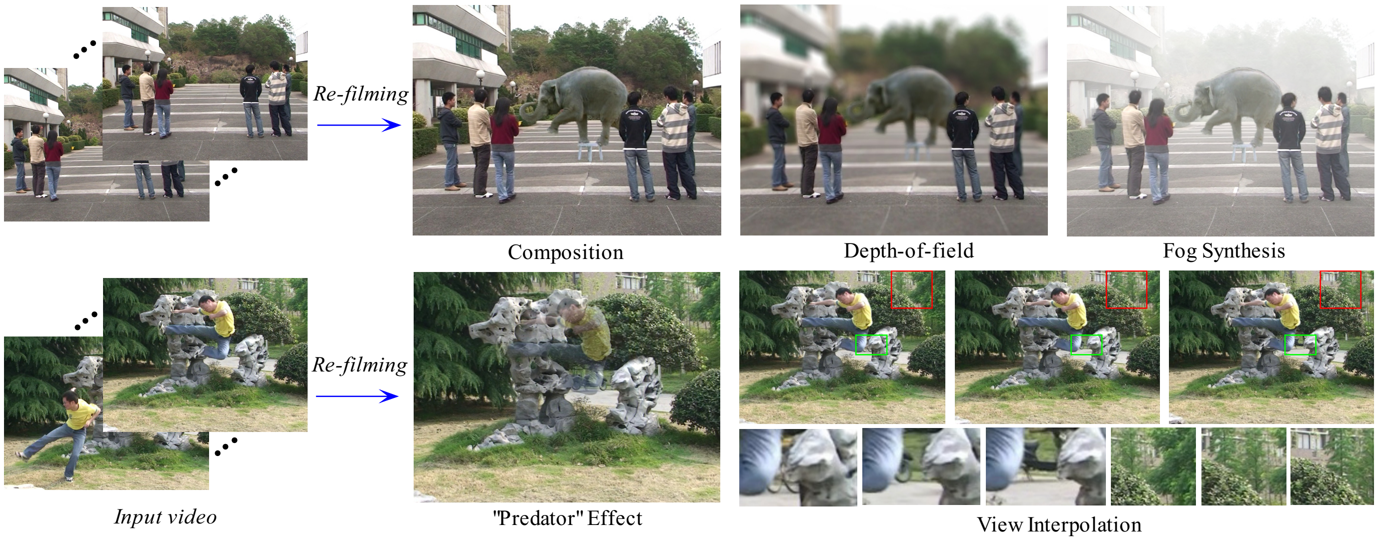


Fig. 1. The left column shows the snapshots of the input videos. The image frames on the right show a set of exemplar re-filming effects from our system.

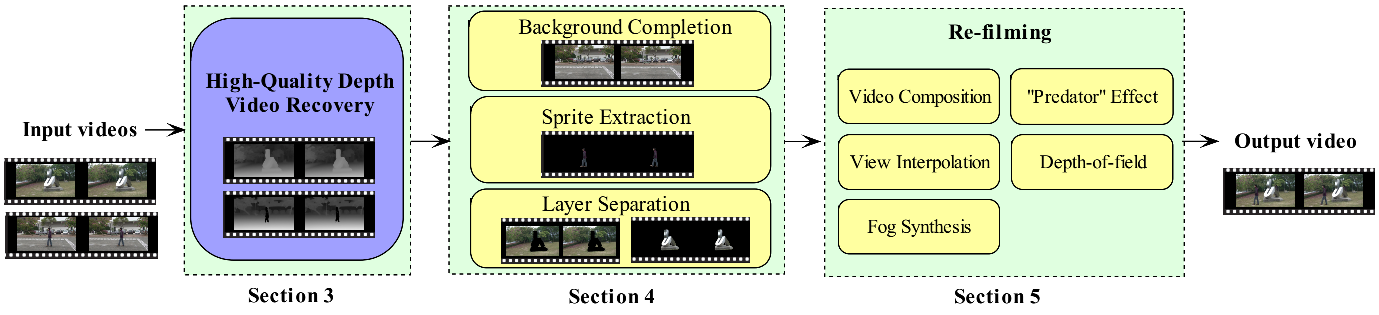


Fig. 2. The overview of our system.

applicable to a single video input. Given two video sequences of different scenes acquired with moving cameras, Xiao et al. [13] proposed seamlessly transferring a static 3D object from one sequence to the other. However, they did not discuss the problem of registering moving objects.

Recovering the camera motion is essential for video editing. This can be achieved using the structure-from-motion (SFM) techniques. The state-of-the-art SFM algorithms can automatically recover sparse 3D points and camera position for a large class of camera motions [14], [15], [16]. Our system employs the SFM method of Zhang et al. [16]. For content-based video editing, using SFM to compute a sparse set of 3D points is not sufficient as it does not resolve the geometrical relation of the scene. Hengel et al. [1] introduced an interactive approach to build 3D models from a video. However, for complex natural scenes (e.g. a tree with many leaves), such an interactive reconstruction becomes labor-intensive and may not even be tractable.

Given an input of multiple images, dense depth maps can be estimated by multi-view stereo algorithms [17], [18], [19], [20], [21]. However, these methods compute the depth map for each frame independently and may not preserve temporal consistency. Kang and Szeliski [22] addressed this problem by simultaneously optimizing a set of depth maps at multiple key-frames, by adding a temporal smoothness term. Most recently, Zhang et al. [2] proposed a bundle optimization method to reconstruct temporally consistent video depth maps. In our system, we improve this method by dramatically increasing the depth precision without introducing much computational overhead. Bhat et al. [23] introduced an image-video framework for automatically enhancing videos using several high-resolution photographs. This method is also limited to only handling the videos of static scenes.

3 Creating Depth Video

Figure 2 illustrates an overview of our system. The input can be a single or multiple video clips. Our system automatically recovers the camera parameters and the complementing depth video. With these view-dependent per-frame depth maps, we are able to interactively perform sprite extraction, layer separation, and background completion. Finally, in the re-filming module, various visual effects can be created using these extracted sprites, separated layers, and backgrounds, without explicit 3D model/surface reconstruction.

We employ a multi-view stereo algorithm to compute the depth map for each frame, in order to generate a high-quality depth video. We improve the method of Zhang et al. [2] which includes a bundle optimization and incorporates the

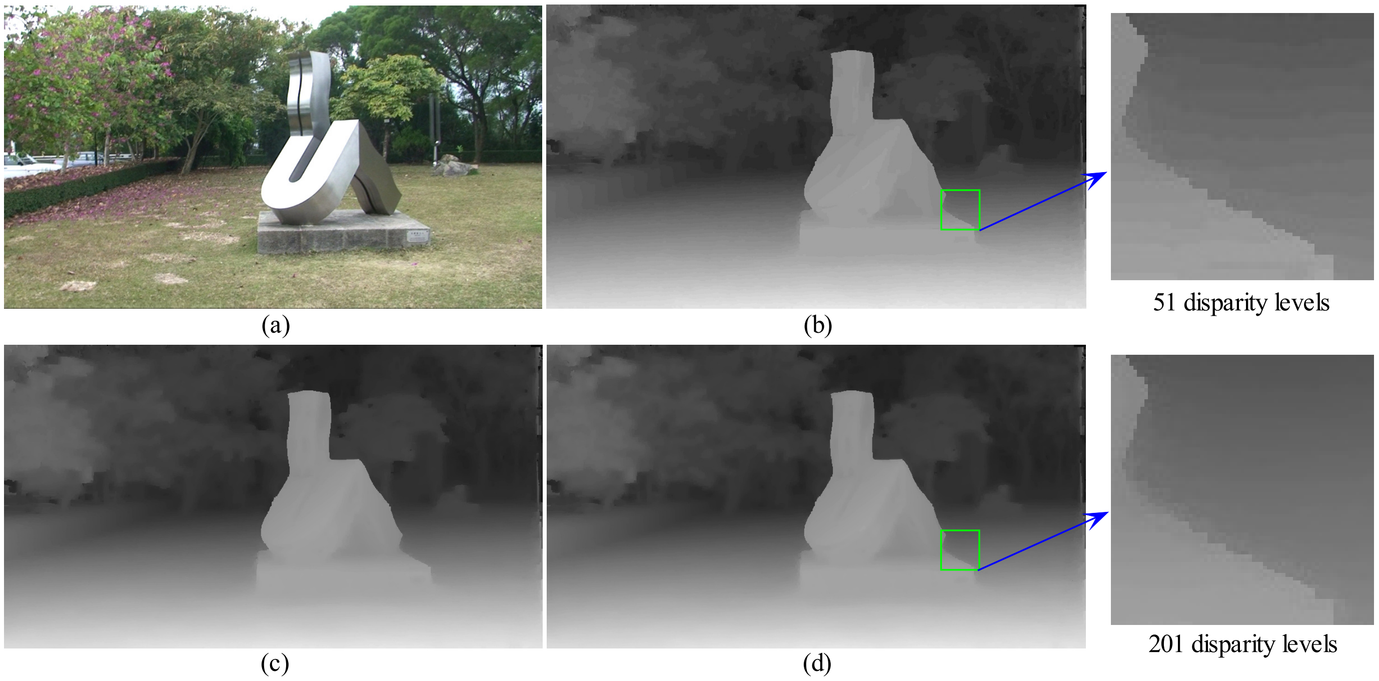


Fig. 3. Depth-level expansion. (a) One frame from the input video. (b) The estimated disparity map with 51 disparity levels in a single BP pass. The banding artifact due to aliasing is noticeable. (c) The estimated disparity map with 201 disparity levels in a single BP pass. (d) The estimated disparity map with our two-pass BP. The results of (c) and (d) are quite comparable in quality.

geometric coherence constraint to overcome the vulnerability of the depth estimation to image noises, occlusion, and other problems. Our improvement lies in a depth-level expansion algorithm to increase the depth precision without introducing much computational overhead. The recovered depth maps are not only temporally consistent but also accurate to retain the sharp discontinuous object boundaries. Such quality is especially important for generating visually plausible re-filmed videos. In what follows, we first briefly describe the bundle optimization algorithm.

3.1 Bundle Optimization

Consider a video sequence with n frames, $\hat{I} = \{I_t | t = 1, ..., n\}$, the objective of depth recovery is to obtain a set of disparity maps $\hat{D} = \{D_t | t = 1, ..., n\}$. $I_t(\mathbf{x})$ represents the color (or intensity) of pixel $\mathbf{x}$ in I_t. It is a 3-vector in a color image or a scalar in a grayscale image. The disparity value $D_t(\mathbf{x})$ is defined as the inverse of depth $z_{\mathbf{x}}$, i.e., $D_t(\mathbf{x}) = 1/z_{\mathbf{x}}$. For simplicity's sake, the terms "depth" and "disparity" are used interchangeably.

The depth video estimation is based on a bundle optimization model with the energy defined as

$$E(\hat{D};\hat{I}) = \sum_{t=1}^{n} (E_d(D_t;\hat{I},\hat{D}\backslash D_t) + E_s(D_t)), \tag{1}$$

where the data term E_d measures how well the disparity $\hat{D}$ fits the given sequence $\hat{I}$, and the smoothness term E_s encodes the spatial smoothness. We minimize $E(\hat{D};\hat{I})$ to estimate the video depth maps.

The data term E_d is defined as

$$E_d(D_t;\hat{I},\hat{D}\backslash D_t) = \sum_{\mathbf{x}} 1 - u(\mathbf{x}) \cdot L(\mathbf{x}, D_t(\mathbf{x})), \tag{2}$$

where the normalization factor is written as

$$u(\mathbf{x}) = 1/\max_{D_t(\mathbf{x})} L(\mathbf{x}, D_t(\mathbf{x})).$$

$L(\mathbf{x}, D_t(\mathbf{x}))$ is the disparity likelihood term proposed in [2], counting in both the color and geometry constraints.

The smoothness term is defined as

$$E_s(D_t) = \sum_{\mathbf{x}} \sum_{\mathbf{y} \in N(\mathbf{x}_t)} \lambda \cdot \min\{|D_t(\mathbf{x}) - D_t(\mathbf{y})|, \eta\}, \tag{3}$$

where $N(\mathbf{x})$ denotes the set of neighbors of pixel $\mathbf{x}_t$, λ is a smoothness weight, and η determines the upper limit of the cost. In all our experiments, $\lambda = 5/(d_{\max} - d_{\min})$, $\eta = 0.05 \cdot (d_{\max} - d_{\min})$, as $[d_{\min}, d_{\max}]$ is the range of disparity.

To minimize $E(\hat{D};\hat{I})$ and accordingly estimate the optimal disparity values, we uniformly quantize the disparity into discrete values. We use the method proposed in [2] to first initialize the disparity maps, and then refine them by minimizing the energy in (1) using an efficient loopy belief propagation [24]. Each pass starts from frame 1. To reduce the computational complexity, when disparity map t is being refined, the depth estimates of all other frames are fixed. In this case, (1) is simplified to

$$E_t(D_t) = E_d(D_t) + E_s(D_t), \tag{4}$$

as all $E_d(D_{t'})$ and $E_s(D_{t'})$, where $t' \neq t$, have the fixed energy. The data term $E_d(D_t)$ associates frame t with about

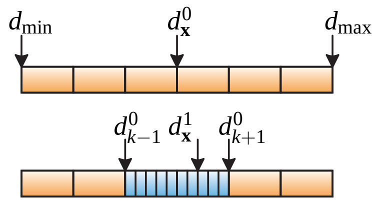


Fig. 4. Depth level expansion. The top row shows that we first estimate a coarse-level disparity value $d_{\mathbf{x}}^0$ between $d_{\min}$ and $d_{\max}$. The bottom row shows that the refined disparity $d_{\mathbf{x}}^1$ is obtained by splitting a coarse level to many fine levels around $d_{\mathbf{x}}^0$.

40 neighboring frames in energy computation according to its definition. Depth maps are sequentially refined until frame n has been processed. In our experiments, two passes of optimization on the whole video are usually sufficient.

3.2 Depth-Level Expansion

In the above depth estimation, we use belief propagation (BP) to minimize the energy in (4). The computational complexity is linear to the number of labels, i.e., the number of depth levels. Hence, accurate depth estimation using a large number of levels implies large memory consumption. Here, we propose a level expansion method to densify the levels of depth. It is a coarse-to-fine approach to significantly refine the inferred depth without introducing expensive computation. Kang et al. [25] proposed a hierarchical graph cut algorithm to accelerate the global optimization for multi-view stereo. The complexity of each level is quadratic to the number of labels. In contrast, our depth-level expansion is based on a two-pass BP algorithm, where in each pass the computational complexity is linear to the number of labels. Another reason that we choose BP is that the distribution of the data costs in our bundle optimization model is usually distinctive, which makes BP converges very quickly (10 iterations are sufficient in our experiments).

To minimize $E_t(D_t)$ for all t, we first quantize all disparities into 51 levels, where the kth level

$$d_k^0 = d_{\min} + \frac{k}{50}\cdot(d_{\max} - d_{\min}), \qquad k = 0,\ldots,50.$$

Then we apply BP to minimize the energy (4) and refine the depth maps. We denote the estimated disparity value for pixel $\mathbf{x}$ as $d_{\mathbf{x}}^0$, where $d_{\mathbf{x}}^0 = d_k^0$. Figure 4 illustrates this idea. Afterwards, we construct finer disparity levels for pixel $\mathbf{x}$ only in range $[d_{k-1}^0, d_{k+1}^0]$ except the extremal values at $k=0$ and $k=50$. It is done by quantizing the depths in $[d_{k-1}^0, d_{k+1}^0]$ into another 21 levels, where the new ith level is

$$d_i^1 = d_k^0 + \frac{i}{20}\cdot(d_{k+1}^0 - d_{k-1}^0), \qquad i = 0,\ldots,20.$$

The optimization method described in Section 3.1 is then used again to refine the depth values. Only two passes of BP can efficiently compute the depth with hundreds of levels.

We demonstrate one of our inferred video depth maps in Figure 3(d). Readers are also referred to our supplementary video[1] for inspecting the quality of the inferred depth maps. Compared to the single-pass BP with depth levels of 501, our two-pass approach consumes only 10% memory space and runs 7 times faster. The quality of the inferred depths is comparable. Figure 3 shows a comparison. Due to the limitation of memory, we are only able to run the one-pass BP with 201 disparity levels (instead of 501 disparity levels) for comparison. Figure 3(b) is the result of a single-pass BP with 51 disparity levels. Figure 3(c) shows the result of a single-pass BP with 201 disparity levels. Figure 3(d) is the result of our two-pass BP with 201 disparity levels (the first pass with 51 disparity levels and the second pass with 9 disparity levels). The total number of iterations is 10. There is nearly no visual difference between (c) and (d), and the average quantitative difference is $0.00214(d_{\max} - d_{\min})$. Only 0.16% of the pixels are with disparity difference larger than $\frac{1}{50}(d_{\max} - d_{\min})$. If our first-pass BP runs with more labels, the difference could be further reduced.

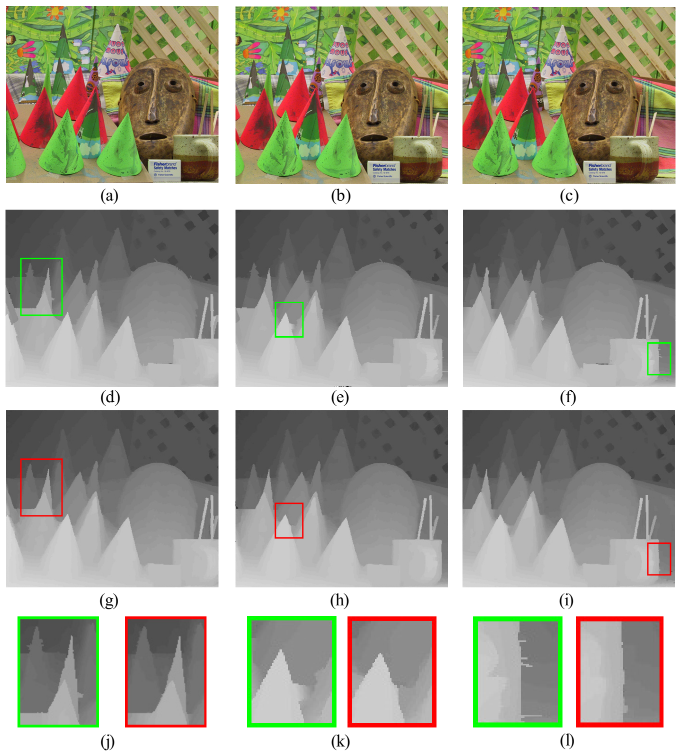


Fig. 5. "Cone" example. (a)-(c) The $2th$, $4th$, and $6th$ image of "Cone" sequence; (d)-(f) The initial disparity maps for (a)-(c) respectively. (g-i) The disparity maps after bundle optimization. (j-l) The close-ups.

3.3 Evaluation with Middlebury Stereo Images

For quantitative evaluation, we test our method on the "Cones" example (Middlebury stereo vision webpage: http://vision.middlebury.edu/stereo/) where ground truth data are given. The results are shown in Figure 5 with statistics given in Table 1.

1. The supplementary video can be found from the following site: **http://www.cad.zju.edu.cn/home/gfzhang/projects/refilming/**

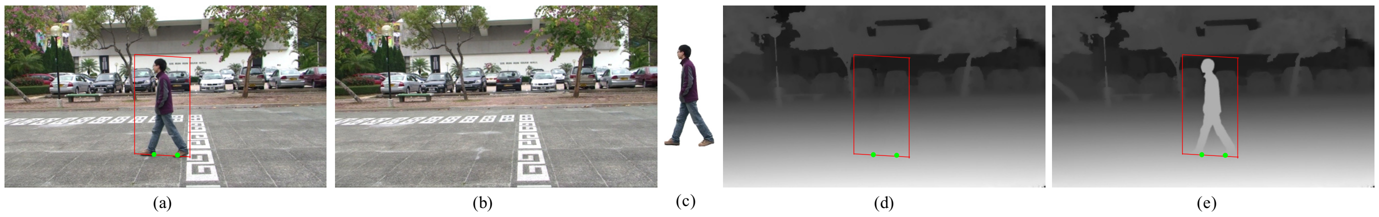


Fig. 6. Video sprite extraction with background completion. (a) One frame from an input video in which we want to extract the man. (b) The completed background by re-projecting pixels from other frames. (c) The extracted sprite. (d) The automatically computed background depth map. (e) To determine the position of the sprite plane, we only need two clicks at the contact points between the person and the ground.

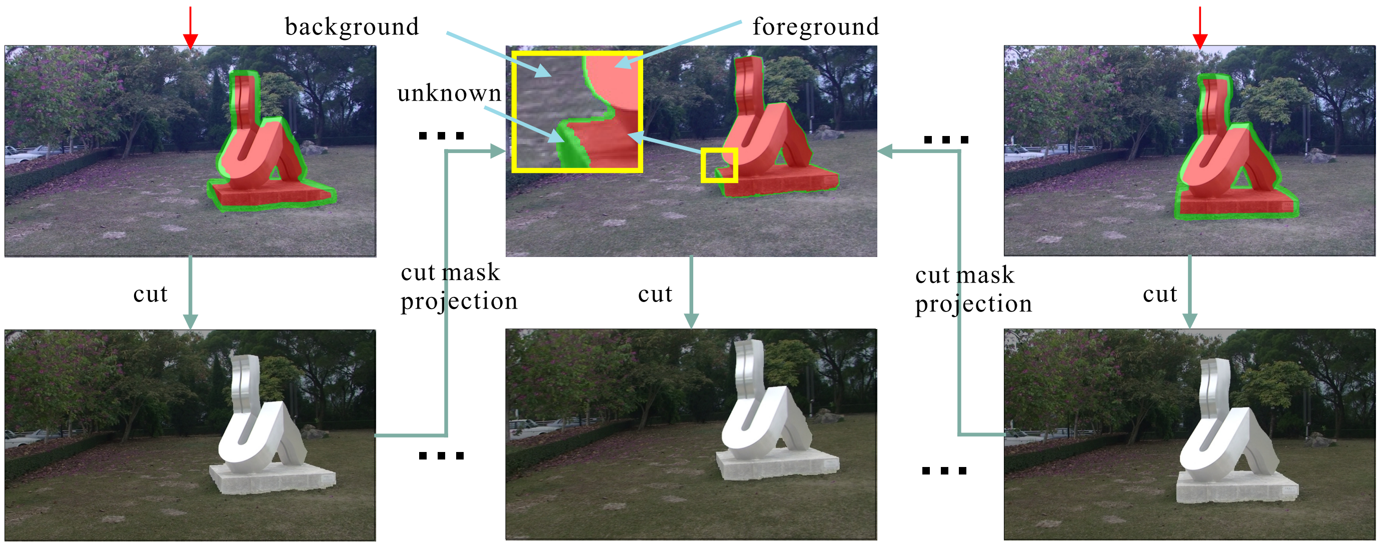


Fig. 7. Bilayer separation. Starting from the embracing key frames (frames on the left and right columns), the depth maps are separated into two layers as indicated in the object masks. Then the masks are projected onto the in-between frames to automatically generate the corresponding trimaps by identifying the difference between two projected masks. Pixels receiving consistent mask values are labeled as either foreground or background, while the inconclusive pixels are labeled as unknowns (colored in green).

TABLE 1
Error statistics on the "Cones" example.

Algorithm	Cones		
	nonocc	all	disc
Initialization	3.86	6.17	10.7
Bundle Optimization	2.89	5.76	8.10

The "Cones" example contains 9 images. In disparity initialization, for each image, we employ the method in [2] to initialize its disparity map, making use of all other 8 images (as shown in Figure 5(d-f)). The segmentation errors cause a few visual artifacts around the discontinuous object boundaries. Then we perform bundle optimization for two passes. In each pass, we refine disparity map t while fixing the others. This process takes for each frame about 15 seconds, where 2 seconds are spent on running BP (with a Quad-core Xeon 2.66GHz CPU). We have tested with a similar procedure using the α-β-swap graph cuts and found that several minutes are required to produce a comparable result.

After bundle optimization, the disparities are improved and their maps become more consistent with each other. Figure 5(j-l) show the close-up comparisons where the final results have much less artifacts around the discontinuous boundaries. We also performed the quantitative evaluation (the ground truth map of the 2nd image is publicly available) and show the statistics in Table 1. It demonstrates that the disparity errors are reduced after the final bundle optimization.

4 Layer Separation and Completion

Once the dense depth maps are available, we perform layer separation and background completion, to prepare for the visual effect generation in the following re-filming module. We provide users with interactive tools to locally modify the video content in a temporally consistent manner. For these operations, users only need to process one or a few frames, while the effect is propagated to the rest of the video.

4.1 Sprite Extraction and Background Completion

We first describe how to use the inferred depth to estimate the background and to extract the sprites (moving objects) from the input video. Object segmentation/extraction is one of the most popular tools in image editing [26], [27]. It is also essential in many video applications, such as object removal and view interpolation. Its major challenge is how to obtain a sharp and clear boundary that is temporally consistent over the frames. In our depth-inferred video, these moving objects usually do not receive valid depth information because they violate the multi-view geometry constraint [14]. However, similar to the video matting method of Chuang et al. [4], our sprite extraction can benefit from background completion, which makes the matting solver robust.

In particular, we estimate the missing background pixels by projecting the neighboring views (frames) to the current frame given the dense depth maps. In order to avoid ghosting artifacts and the loss of high-frequency details, we employ the method proposed in [23] to reconstruct consistent view-dependent backgrounds for all frames. This background completion method is not only applicable to object extraction, but also usable for creating "predator" effect and view interpolation to be described in the next section.

After the background is completed, we create an object trimap (a mask containing three regions, indicating the foreground, background, and unknown pixels) for each frame and extract the moving objects by applying the video matting method [4] with the estimated background. For illustration, in Figure 6(a), we show an input frame containing a walking man. The background is completed in (b). The matting result in (c) is generated using the color information from both (a) and (b).

4.2 Sprite Representation

To make an inserted video sprite appear natural in another video, we typically require more than just the sprite matte and color. One example is that if we naïvely (that is, through directly copying and pasting) composite the sprite of a walking man to a target video, the sprite may look like floating in the new scene, due to the difference of the two camera motions. Therefore, in order to achieve seamless insertion, the depth, scale, and position of the sprite relative to the camera must be recorded during extraction and be accounted during insertion.

Nonetheless, moving objects usually do not satisfy the rigid color constancy constraint, and hence their depth values cannot be estimated by multi-view stereo algorithms. In experiments, we found that a coarse depth estimation for the extracted sprite is usually sufficient in regard to the purpose of video composition. For instance, the depth of the walking man in Figure 6 can simply be represented by a 3D plane.

In our system, we use a plane perpendicular to the ground to approximate the video sprite in each frame. The position and orientation of the 3D plane are identified by two anchor points (as illustrated as the green dots in Figure 6(a)). Moreover, if the 3D plane is known to be orthogonal to the camera viewing direction, one anchor point is sufficient. The anchor points are obtained by asking the user to click at the contact points between the sprite and the ground (feet of the man in Figure 6(e)). Since the depth of the static background is available, the depth values at the contact points are then looked up from the background and are used to compute the position and orientation of the 3D plane. The user only operates on key frames, and the 3D sprite plane in other frames can be linearly interpolated from two embracing key frames.

4.3 Layer Separation for Static Scene

As the inferred depth only represents a partial geometry, in order to achieve practical video content editing, it is sometimes necessary to explicitly separate the static scene (with inferred depth) into layers, e.g. simulating the scene occlusion. There is no need to perform layer separation respectively for each frame. Instead, our system allows the user to operate in a sparse number of key frames, and propagates the layer information to the rest of the video.

By treating the depth as an additional color channel, we perform a bilayer segmentation using the method of Rother et al. [5] to iteratively separate the static objects in different layers. The process is briefly illustrated in the left- and right-most columns of Figure 7. Then we develop a novel method to automatically propagate the cutout information from the key frames to all others. This information can be used to reliably form a trimap with a narrow unknown region for each frame. Note that automatic propagation of one trimap to other frames by optical flow is usually difficult and unreliable [4].

Our cutout propagation is based on a geometry projection process and is illustrated in Figure 7. First, the user selects key frames I_{k0}, I_{k1}, $\cdots$, I_{kn} with every interval of about 50 frames and cuts the object out. Using depth information, we then project the object cut-out masks M_{k_i} $(0 \leq i \leq n)$, to other in-between frames and make each of them receive two object-mask projections. The masks only contain binary values where the object pixel is labeled 1 and non-object pixel is labeled 0. Suppose frame j $(k_i < j < k_{i+1})$ receives two mask projections M'_{k_i} and $M'_{k_{i+1}}$. We compare the projected values $M'_{k_i}(\mathbf{x})$ and $M'_{k_{i+1}}(\mathbf{x})$ for each pixel $\mathbf{x}$ and label it as unknown if $M'_{k_i}(\mathbf{x}) \neq M'_{k_{i+1}}(\mathbf{x})$. Otherwise, the pixel with $M'_{k_i}(\mathbf{x}) = M'_{k_{i+1}}(\mathbf{x}) = 0$ is defined as background while the pixel with $M'_{k_i}(\mathbf{x}) = M'_{k_{i+1}}(\mathbf{x}) = 1$ is regarded as foreground. Thus, a trimap for frame j is formed with the estimated foregound, background, and unknown regions. The unknown regions are usually very narrow along the layer boundary due to the high accuracy of our depth estimation. In experiments, the method can generate 50 trimaps per second. A working example is illustrated in the middle column of Figure 7. Finally, we apply the border matting [5] to further refine the binary segmentation.

5 Applications

With the above tools and the depth-inferred video(s), various visual effects can be created in our system, that are temporally and spatially consistent. As the video results are dynamic in nature, readers are referred to our supplementary video that gives a better presentation of the results.

5.1 User Interface

We first briefly describe the design of our user interface. It is to facilitate navigating the sprite library and flexibly inserting the

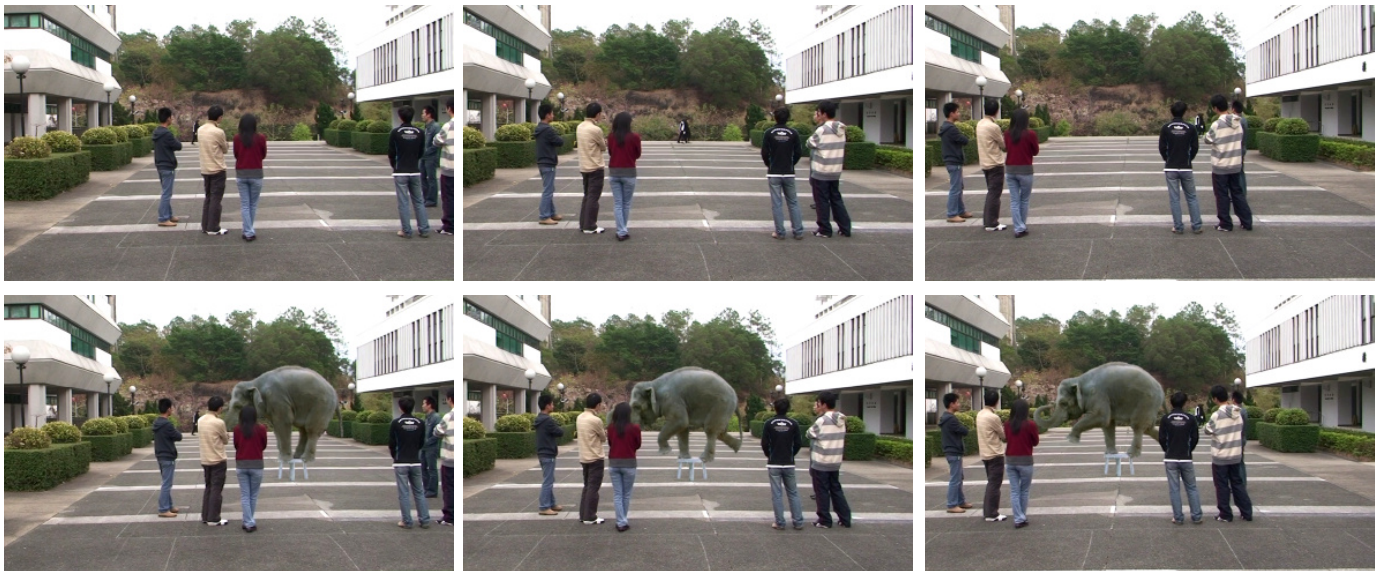


Fig. 9. Object insertion. The top row shows the original frames from a video captured by a hand-held camera. The bottom row shows the corresponding frames after inserting a performing elephant. Note how its position and scale are accurately aligned by our system. The occlusion of the elephant by the audience is also correctly resolved by object extraction and layer separation.

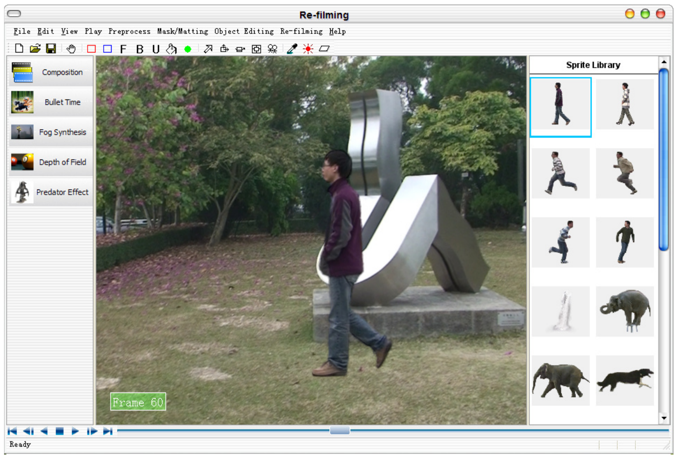


Fig. 8. User Interface. The top editing toolbar provides tools to seamlessly insert sprites into the target editing video. The side panel on the right shows the sprite instances. The left panel lists the command buttons for generating different visual effects.

selected sprite into a target video. A set of interactive object editing (such as transforming and cloning sprites) and shadow synthesis tools are also provided.

A screen-shot of our UI is shown in Figure 8. The top-row icons are for the masking/matting, object clone, transformation, and shadow synthesis tasks. The right panel displays the object/sprite instances. It is allowed to select the desired sprite, and insert it into the current video. Our system then automatically aligns the contact points utilizing the depth information. The object transformation tools enable adjusting the position, orientation, and scale of the sprite. The left panel lists the command buttons for generating visual effects.

5.2 Video Composition

During the insertion of a video sprite into a target background video, we need to ensure the consistency of the camera motion, color tone, and lighting. Figures 9 and 10 show two composition examples.

Geometry registration with occlusion resolving As the camera motions of the source and target videos can be substantially different, we need to compensate them in order to achieve a harmonic alignment. We first register the coordinate systems between the source and target videos using the camera parameters and denote the perspective transformation between the two coordinate systems as a 4×4 matrix $\mathbf{P}_A$. So any time when the user modifies the object position, orientation, and size in the target video, the original $\mathbf{P}_A$ is multiplied by a corresponding 3D rotation, translation, or scaling matrix. These matrix multiplications guarantee that the sprite is correctly aligned to the new scene without causing the drift artifacts. We also resolve the potential occlusion between the sprite and the separated layers in the target video, by simply sorting the depth in layers and rendering them in a far-to-near order. The registration can be done in real-time, and hence facilitates the interactive modifications.

Color tone adjustment Color tone compatibility between the inserted sprites and the target background video is also important to make the composition result realistic. In our system, we employ the method in [28] to adjust the chromaticity and intensity of the video sprites.

Shadow synthesis Shadow is an important visual cue for creating the vivid impression of realism. Without it, the inserted sprite usually looks unnatural. The shadow extracted from a source video may not be useful for our video composi-

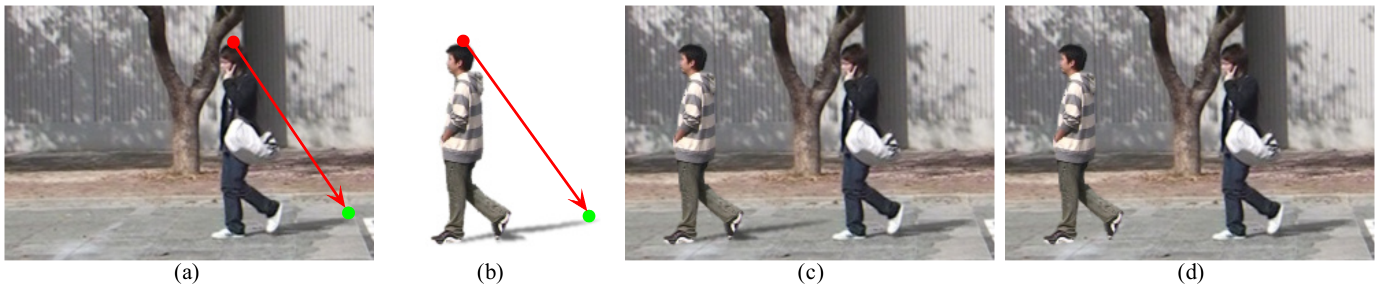


Fig. 10. Shadow synthesis. (a) The vector of the directional sunlight can be determined by the red and green points indicated by the user. Their 3D coordinates can be retrieved from the depth map. (b) Synthetic shadow cast. Comparison of the composition results (c) with and (d) without shadowing.

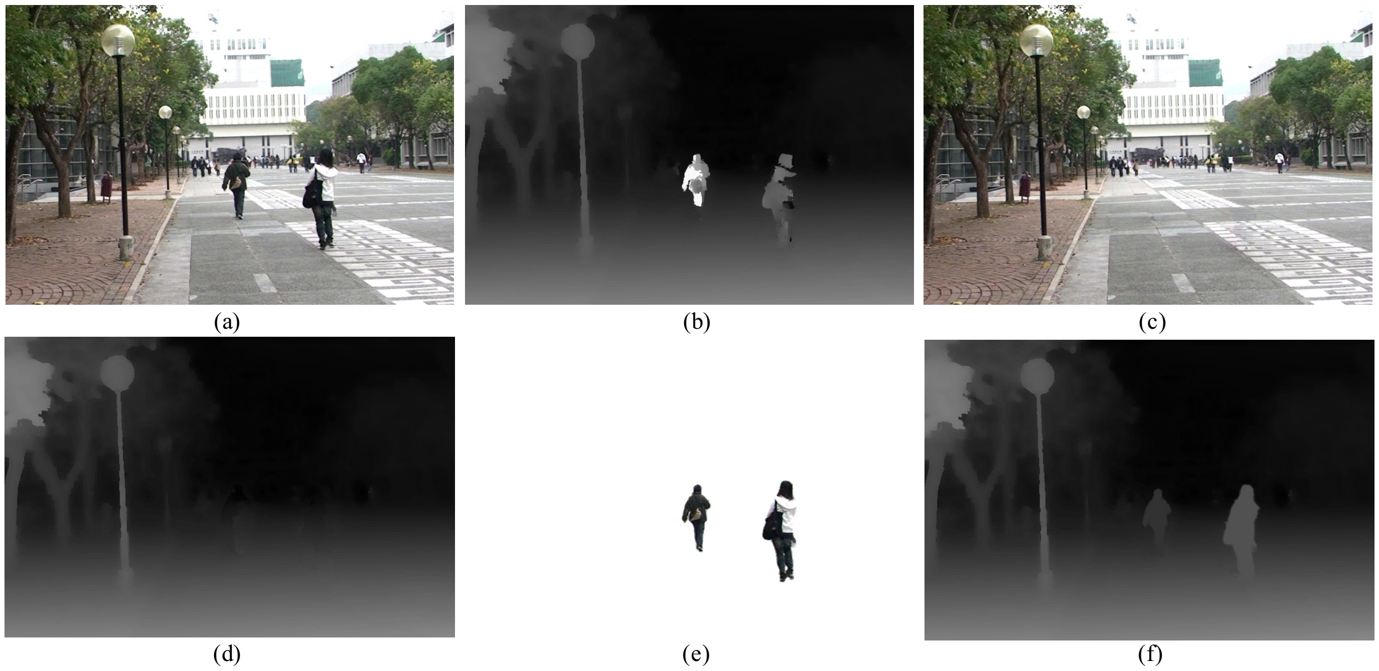


Fig. 11. Depth recovery with moving objects. (a) One frame from the input video. (b) The estimated depth map without matting out the moving objects. Although the depths of the moving object are not accurate, they do not affect nearby background depth estimation. (c) The completed background image by depth projection. (d) The completed background depth map. (e) The extracted moving objects. (f) The final depth map.

tion due to the possible variation of the background geometry and light direction. Moreover, previous work in shadow extraction either requires controlled lighting conditions [29], or makes restrictive assumptions about the camera, lighting, and shadow properties [30].

Our system synthesizes the shadow taking account of the depth estimates. The target video shown in Figure 10(a) is captured by a moving camera where the scene depths are recovered. Our system only requires the minimal user interaction to specify the lighting direction in one frame through just two clicks, as shown as the red and green points in Figure 10(a). Then the direction of the sunlight is constructed by connecting the 3D point (red dot) to its projection on the ground (green dot). The walking person in Figure 10(b) is also extracted from a video captured by a moving camera and is represented as a view-dependent 3D plane, which is similar to that illustrated in Figure 6. Since the sunlight direction is estimated, with the anchored 3D plane, the shadow can be synthesized by projecting the sprite onto the ground as shown in Figure 10(b). To improve the realism, we attenuate the shadow intensity to simulate the indirect illumination, and Gaussian-blur the shadow boundary to create the penumbra. Figures 10 (c) and (d) compare the inserted walking man with and without shadowing.

5.3 Depth-of-field

Depth-of-field is a common trick for emphasizing the object of interest in filming and photography practices. Producing a depth-of-field effect in video usually requires special and expensive equipments [31]. In our system, such depth-of-field can be easily obtained and modified even after video

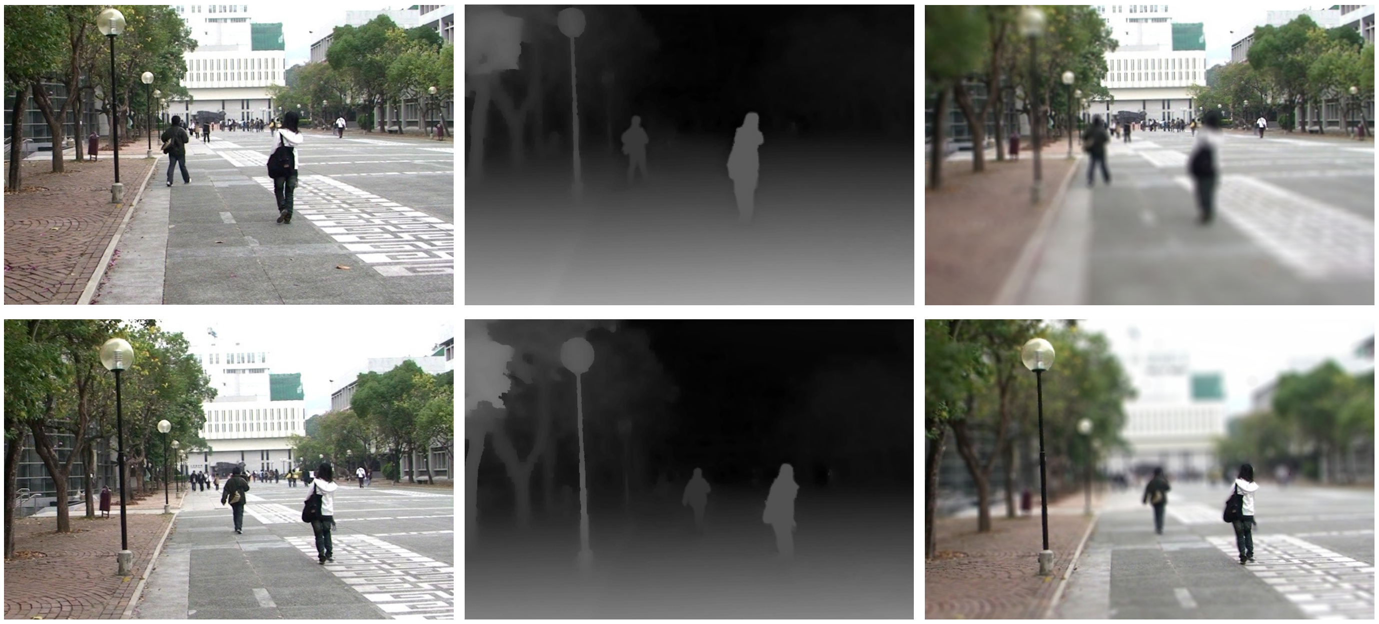


Fig. 12. Depth-of-field. The left-most column shows two frames from an input video. The middle column shows the recovered depth maps. The right-most column shows the result of changing the depth of field.

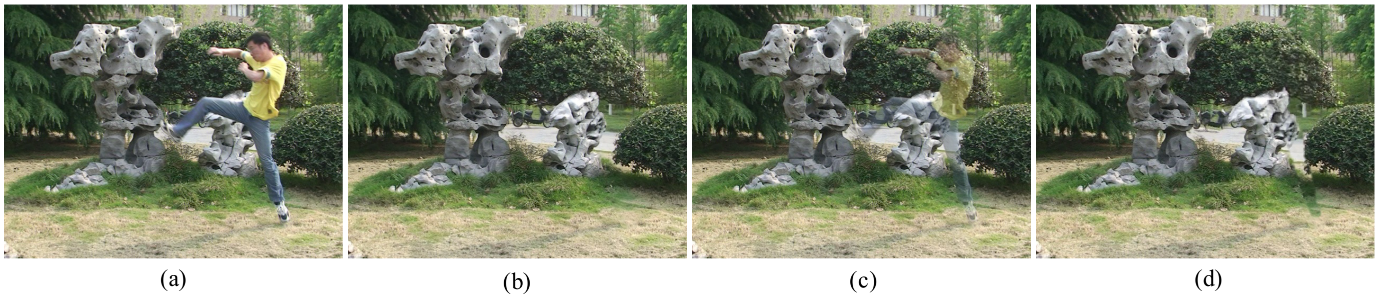


Fig. 13. “Predator” effect. (a) One original frame. (b) The recovered background image. (c) The actor becomes transparent. (c) The actor is camouflaged with the “predator” effect.

acquisition. The estimated depth provides us with sufficient information to re-focus the input video.

In our implementation, we employ the ray-tracing based method proposed in [31] to focus on different objects. We can freely control the focus plane as well as the width of the depth-of-field by adjusting the size of out-of-focus blur circle. Figure 12 illustrates how we change the focus towards the walking girl and the street lamp. Note that this example contains moving objects, whose depths cannot be directly recovered by multi-view stereo method. Fortunately, moving objects do not harm our depth estimation for the visible background pixels. The moving pixels are regarded as temporal “noise” as they are inconsistent among frames and do not satisfy the multi-view geometry constraint. We show an illustration in Figure 11. Though the estimated depths of the moving objects are not accurate, it does not affect the depth estimate of other background pixels in our system, as shown in Figure 11(b).

To assign reasonable depth values to the moving object, we first roughly mask the foreground out and estimate the occluded background colors and depths by projecting neighboring views to the current frame according to the recovered depth information. The completed background image and depth map are shown in Figure 11(c-d), respectively. The estimated background helps matte foreground out. The moving objects are represented by view-dependent 3D planes with assigned depths, as shown in Figure 11(e). Finally, we put the moving objects back to get the refined depth map (Figure 11(f)).

5.4 “Predator” Effect

Figure 13 demonstrates the “predator” effect (camouflaging effect) on a video taken by a hand-held camera. It is produced using our provided tools through a few steps. It should be noted that only separating the foreground layer may not be sufficient for creating the “predator” effect. It is because the occluded background has to be shown behind the transparent characters. So we first recover the video depth maps. Then we extract the object (the actor in this example) and complete the video background using the technique described in Section 4.1. One frame of the completed background is shown in Figure 13(b). Finally, we camouflage the actor by blending the input frames with the completed background

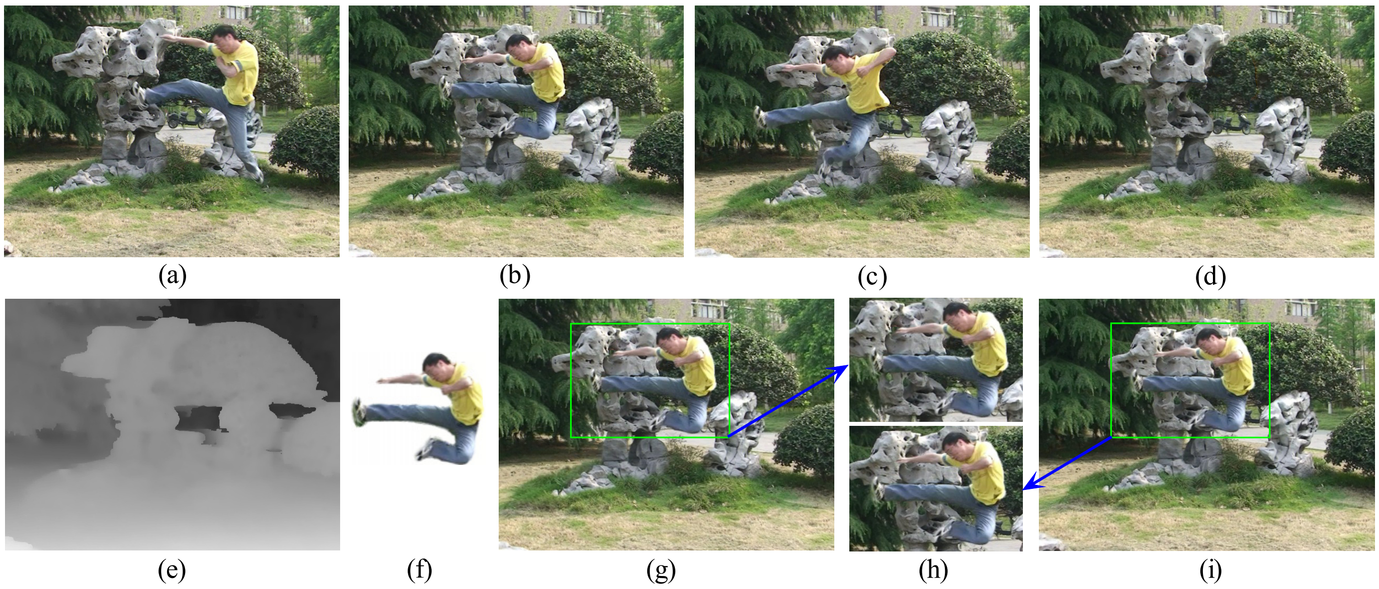


Fig. 14. Bullet-time effect. (a-c) Three frames from an input video. (d) and (e) The object removed color frame and depth map after background completion for the frame in (b). (f) The separated video sprite. (g-i) The frozen kicking man viewed from two different angles.

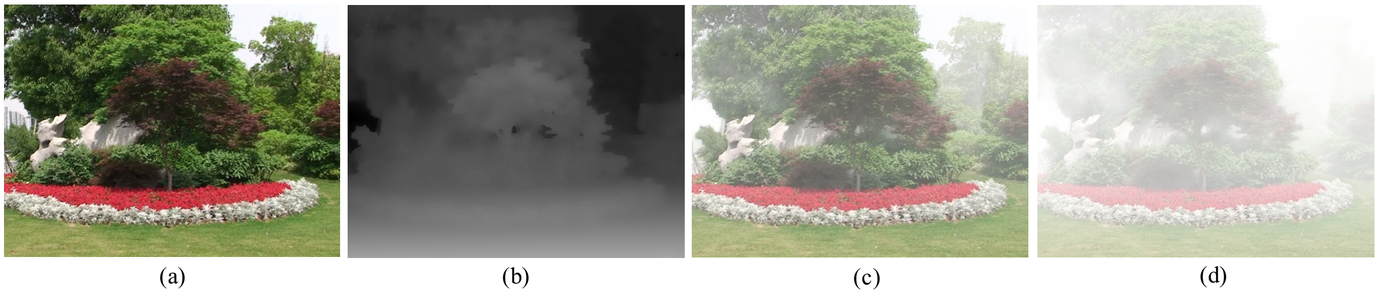


Fig. 15. Fog synthesis. (a) and (b) One frame with recovered depth map. (c) and (d) With the estimated depths, we add fog to the scene with different densities.

frames (Figure 13(c)). To simulate the “predator” effect, we add refractive and wavy distortion to the blending region (Figure 13(d)).

5.5 Bullet-time Effect

Bullet-time effect refers to the effect of freezing an object (e.g. pouring water) and meanwhile changing the camera viewpoint. A representative example is in movie *The Matrix*. To create this effect, a dense video camera array or a sparse camera array with the view interpolation technique [18] is usually required.

In our system, a limited bullet-time effect can be created with a single video input. Figure 14 shows a jumping and kicking person frozen in space. The input video is taken by a hand-held moving camera. To generate this visual effect, after recovering the depth for the background, we first remove the man by using a background completion tool (Section 4.1). One such frame is shown in Figure 14(d). The corresponding depth map is shown in Figure 14(e). Then we set a point on the sprite as a center for camera rotation, and create the virtual camera viewpoints around it. Using depth assignments for the background and sprite (represented as a view-dependent 3D plane), the synthesized views from the virtual cameras can naturally be interpolated from the nearest input frames.

5.6 Fog Synthesis

Figure 15 makes use of the depth-inferred video to create the fogging effect. We use a simple fog model introduced in [32] to compute the attenuated intensity for each pixel:

$$I_c = I_o \cdot e^{-\beta z} + I_{fog} \cdot (1 - e^{-\beta z})$$

where I_{fog} is the color of fog, I_o is the original pixel color. β is the scattering coefficient of the atmosphere and z is the depth value. By adjusting the value of β, we can modify the fog density.

6 Discussion

To demonstrate the versatility of our system, we have chosen several natural videos of different sceneries for visual effect

TABLE 2
Timing statistics for creating the visual effects.

Visual Effect	Preprocessing			Effect Rendering
	Background Completion	Sprite Extraction	Layer Separation	
Video Composition (Fig. 9)	30 sec./frame	1 min./frame	5 sec./frame	real time
“Predator” (Fig. 13)	10 sec./frame	30 sec./frame	–	real time
Fog Synthesis (Fig. 15)	–	–	–	real time
Depth-of-field (Fig. 12)	20 sec./frame	1 min./frame	–	10 sec./frame
Bullet Time (Fig. 14)	20 sec./frame	30 sec./frame	–	5 sec./frame

generation. Most videos are taken by a hand-held camera. With our system, the user can flexibly generate the described the visual effects, including video composition, depth-of-field, “predator” effect, bullet-time, and fog synthesis.

Note that accurate depth maps are required to generate realistic re-filming effects for several applications. Excessively coarse depth maps may lead to unpleasing visual artifacts, such as drifting, blurriness, and distortions, in video composition and view interpolation. Background completion also demands very accurate depths in order to correctly complete the background in several keyframes. Contrary to these applications, a certain class of re-filming effects, such as depth-of-field and fogging, has relatively lower precision requirement for disparities.

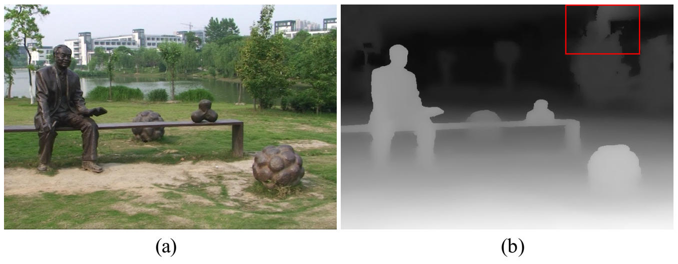


Fig. 16. A video frame (a) and the estimated depth map (b). In the red rectangle, because the disparities around the tree branches have inherent depth ambiguity regarding the nearly-constant-color background sky, the depth estimates are not all correct.

6.1 Running Time

Our system can be divided into a few unsupervised operations and phases requiring simple user interactions. The former includes recovering camera parameters with a depth video. With multi-thread programming, it takes about 2 minutes to process one frame (640×480 pixels) using a Quad-core Xeon 2.66GHz CPU.

The static layer separation process is fast, which typically takes less than 5 minutes to process hundreds of frames. The time spent on background completion depends on the frame resolution and the size of missing region. It typically takes about 20 seconds to process a frame with 640×480 pixels. The time for sprite extraction and layer separation depends on the number of objects to process and the complexity of the object boundary. With the estimated background, moving object extraction typically takes about 30 seconds (including user interaction) to extract one moving object per frame. Besides the computation, what the user inputs is no more than a few clicks and sketching. Thus its usage is not tedious.

Table 2 lists the statistics of the average time spent in creating different visual effects by several novice users with just a few practices and short learning curve. As most computation is spent on the pre-processing of layer separation and completion, the visual effects can be created rapidly. Especially, if there is no moving object in the scene, the fogging and depth-of-field effects can be created immediately after the automatic depth recovery. With necessary pre-processing, video composition, fog synthesis and the “predator” effect can be produced in real-time.

6.2 Limitations and Future Work

In our experimental results, we have demonstrated that the depth recovery method can handle well a large class of camera motions. However, similar to other multi-view stereo algorithms, if there is not sufficient camera movement, the recovered depths could be defective. This problem has been observed and widely studied in multi-view geometry [14]. Fortunately, this problem does not affect significantly the object insertion and removal, because small camera motion only results in less accurate depth estimation, but does not cause the drift artifacts.

Another limitation of our depth recovery is that, if the scene contains extremely textureless regions, there may exist inherent ambiguity for depth inference. For these regions (such as the clear blue sky), our system does not guarantee to produce correct depth initialization even with color segmentation, which could further affect the succeeding bundle optimization. As shown in Figure 16, the near-constant color in the background sky can be assigned with different depth values, all of which happen to satisfy photo-consistency constraint. So, without prior knowledge, inferring correct depth values in these regions is extremely difficult. Part of our future work is along the direction of solving this problem.

The quality of SFM or depth maps affect the finally produced re-filming effects. It would be desirable if, given an input video, the system automatically tells whether the estimates are sufficiently good or not. However, automatically evaluating SFM is challenging and the possible solutions include visually inspecting the recovered 3D structure as well as the camera trajectory, and inserting a virtual object to see if it drifts. For the inferred depth maps, the quality could possibly be measured by the degree of temporal consistency.

For video composition, if the illumination in the source and the target videos are substantially different, even using color adjustment, unrealistic results may still be produced. We plan to investigate building accurate model for sprites such that the illumination information can be estimated.

Finally, for video composition, since we only have partially approximated geometry information, i.e., the simplified view-dependent 3D planes, for the moving object, our system requires that different views of the moving object between the source and target videos are not too large to avoid unnatural object insertion. For example, if the source camera is panning whereas the target camera rotates, distortion of the inserted sprite may be produced. Our bullet-time effect also has the similar limitation. Note that the layer represented sprite does not cause any problem in producing the “predator”, fog synthesis, and depth-of-field effects even if the camera or object moves along a complex trajectory.

7 Conclusions

We have presented a comprehensive and semi-automatic video editing system, that allows for creating visually plausible re-filming effects. The cornerstone of our system is a robust video depth estimation method to automatically produce temporally consistent and highly accurate depth. Using this information, background completion, sprite extraction, and layer separation are achieved with only a small amount of user interaction mostly on sparse key frames. Our system also contributes a set of convenient tools allowing user to flexibly create convincing visual effects, including composition, “predator” effect, view interpolation, depth-of-field, and fog synthesis, avoiding challenging 3D modeling and refitment.

Acknowledgments

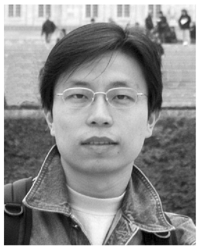
We would like to thank the associate editor and all the reviewers for their constructive comments to improve the manuscript. We also thank Hanqing Jiang, Li Xu, Jianing Chen, Hin Shun Chung, Liansheng Wang and Xiaopei Liu for their help in user testing, video capture and production. Thanks to Carl Jantzen for the video narration. This work is supported by the 973 program of China (No. 2009CB320802), NSF of China (No. 60633070), and the Research Grants Council of the Hong Kong Special Administrative Region, under General Research Fund (Project No. 412708 and 417107).

References

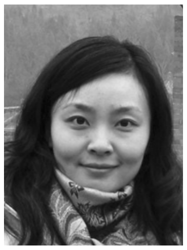
[1] A. van den Hengel, A. R. Dick, T. Thormählen, B. Ward, and P. H. S. Torr, “Videotrace: rapid interactive scene modelling from video,” *ACM Trans. Graph.*, vol. 26, no. 3, 2007.
[2] G. Zhang, J. Jia, T.-T. Wong, and H. Bao, “Recovering consistent video depth maps via bundle optimization,” in *CVPR*, 2008.
[3] Y.-Y. Chuang, B. Curless, D. Salesin, and R. Szeliski, “A bayesian approach to digital matting,” in *CVPR (2)*, 2001, pp. 264–271.
[4] Y.-Y. Chuang, A. Agarwala, B. Curless, D. Salesin, and R. Szeliski, “Video matting of complex scenes,” *ACM Trans. Graph.*, vol. 21, no. 3, pp. 243–248, 2002.
[5] C. Rother, V. Kolmogorov, and A. Blake, ““Grabcut”: interactive foreground extraction using iterated graph cuts,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004.
[6] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, “Lazy snapping,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 303–308, 2004.
[7] A. Levin, D. Lischinski, and Y. Weiss, “A closed form solution to natural image matting,” in *CVPR (1)*, 2006, pp. 61–68.
[8] J. Wang and M. F. Cohen, “Optimized color sampling for robust matting,” in *CVPR*, 2007.
[9] J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. F. Cohen, “Interactive video cutout,” *ACM Trans. Graph.*, vol. 24, no. 3, pp. 585–594, 2005.
[10] Y. Li, J. Sun, and H.-Y. Shum, “Video object cut and paste,” *ACM Trans. Graph.*, vol. 24, no. 3, pp. 595–600, 2005.
[11] X. Bai and G. Sapiro, “A geodesic framework for fast interactive image and video segmentation and matting,” in *ICCV*, 2007.
[12] P. Sand and S. J. Teller, “Video matching,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 592–599, 2004.
[13] J. Xiao, X. Cao, and H. Foroosh, “3D object transfer between non-overlapping videos,” in *IEEE Virtual Reality Conference*, 2006.
[14] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.
[15] M. Pollefeys, L. J. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, “Visual modeling with a hand-held camera.” *International Journal of Computer Vision*, vol. 59, no. 3, pp. 207–232, 2004.
[16] G. Zhang, X. Qin, W. Hua, T.-T. Wong, P.-A. Heng, and H. Bao, “Robust metric reconstruction from challenging video sequences,” in *CVPR*, 2007.
[17] J. Sun, N.-N. Zheng, and H.-Y. Shum, “Stereo matching using belief propagation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 7, pp. 787–800, 2003.
[18] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. A. J. Winder, and R. Szeliski, “High-quality video view interpolation using a layered representation,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 600–608, 2004.
[19] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, “A comparison and evaluation of multi-view stereo reconstruction algorithms,” in *CVPR (1)*, 2006, pp. 519–528.
[20] C. L. Zitnick and S. B. Kang, “Stereo for image-based rendering using image over-segmentation,” *International Journal of Computer Vision*, vol. 75, no. 1, pp. 49–65, 2007.
[21] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, “Multi-view stereo for community photo collections,” in *ICCV*, 2007.
[22] S. B. Kang and R. Szeliski, “Extracting view-dependent depth maps from a collection of images,” *International Journal of Computer Vision*, vol. 58, no. 2, pp. 139–163, 2004.
[23] P. Bhat, C. L. Zitnick, N. Snavely, A. Agarwala, M. Agrawala, B. Curless, M. Cohen, and S. B. Kang, “Using photographs to enhance videos of a static scene,” in *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*, J. Kautz and S. Pattanaik, Eds. Eurographics, June 2007, pp. 327–338.
[24] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient belief propagation for early vision,” *International Journal of Computer Vision*, vol. 70, no. 1, pp. 41–54, 2006.
[25] S. B. Kang, R. Szeliski, and J. Chai, “Handling occlusions in dense multi-view stereo,” Microsoft Corporation, Tech. Rep. MSR-TR-2001-80, Sept. 2001.
[26] A. Agarwala, M. Dontcheva, M. Agrawala, S. M. Drucker, A. Colburn, B. Curless, D. Salesin, and M. F. Cohen, “Interactive digital photomontage,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 294–302, 2004.
[27] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. M. Winn, and A. Criminisi, “Photo clip art,” *ACM Trans. Graph.*, vol. 26, no. 3, p. 3, 2007.
[28] J. A. Selan, “Merging live video with synthetic imagery.” Master’s thesis, Cornell University, 2003.
[29] Y.-Y. Chuang, D. B. Goldman, B. Curless, D. Salesin, and R. Szeliski, “Shadow matting and compositing,” *ACM Trans. Graph.*, vol. 22, no. 3, pp. 494–500, 2003.
[30] G. D. Finlayson, S. D. Hordley, C. Lu, and M. S. Drew, “On the removal of shadows from images,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 1, pp. 59–68, 2006.
[31] F. Moreno-Noguer, P. N. Belhumeur, and S. K. Nayar, “Active refocusing of images and videos,” *ACM Trans. Graph.*, vol. 26, no. 3, p. 67, 2007.
[32] S. G. Narasimhan and S. K. Nayar, “Interactive (de)weathering of an image using physical models,” in *IEEEWorkshop on Color and Photometric Methods in Computer Vision*, 2003.

Guofeng Zhang received the BS degree in Computer Science from Zhejiang University, P.R.China, in 2003. He is currently a PhD candidate in computer science at State Key Laboratory of CAD&CG, Zhejiang University. His research interests include camera tracking, 3D reconstruction, augmented reality and video enhancement.

Zilong Dong received his BS degree in Computer Science from Zhejiang University, P.R.China, in 2004. He is currently a PhD candidate in computer science at State Key Laboratory of CAD&CG, Zhejiang University. His main research interests include realtime camera tracking, augmented reality, and video segmentation.



Jiaya Jia received the Ph.D. degree in Computer Science from the Hong Kong University of Science and Technology in 2004. He joined the Department of Computer Science and Engineering at the Chinese University of Hong Kong in September 2004. He is currently an Assistant Professor. His research interests include vision geometry, image/video editing and enhancement, image deblurring, and motion analysis. He has served on the program committees of ICCV, CVPR, ECCV, and ACCV. He is a member of IEEE.



Liang Wan received the B.Eng. and M.Eng. degrees in computer science and engineering from Northwestern Polytechnical University in China, in 2000 and 2003 respectively, and the PhD degree in computer science from the Chinese University of Hong Kong in 2007. She is currently a Research Fellow in the Department of Electronic Engineering, City University of Hong Kong. Her main research interest is computer graphics, including precomputed lighting, non-photorealistic rendering and GPU programming.

Tien-Tsin Wong received the B.Sci., M.Phil., and PhD. degrees in computer science from the Chinese University of Hong Kong in 1992, 1994, and 1998, respectively. Currently, he is a Professor in the Department of Computer Science & Engineering, Chinese University of Hong Kong. His main research interest is computer graphics, including computational manga, image-based rendering, natural phenomena modeling, and multimedia data compression. He received IEEE Transactions on Multimedia Prize Paper Award 2005 and Young Researcher Award 2004.

Hujun Bao received the BS and PhD. degrees in applied mathematics from Zhejiang University in 1987 and 1993, respectively. Currently, he is a Professor and the director of State Key Laboratory of CAD&CG at Zhejiang University. His main research interest is computer graphics and computer vision, including realtime rendering technique, geometry computing, virtual reality and structure-from-motion.