High-Quality Depth Recovery via Interactive Multi-View Stereo

Weifeng Chen[1] Guofeng Zhang[1*] Xiaojun Xiang[1] Jiaya Jia[2] Hujun Bao[1]

[1]State Key Lab of CAD&CG, Zhejiang University [2]The Chinese University of Hong Kong

Abstract

Although multi-view stereo has been extensively studied during the past decades, automatically computing high-quality dense depth information from captured images/videos is still quite difficult. Many factors, such as serious occlusion, large textureless regions and strong reflection, easily cause erroneous depth recovery. In this paper, we present a novel semi-automatic multi-view stereo system, which can quickly create and repair depth from a monocular sequence taken by a freely moving camera. One of our main contributions is that we propose a novel multi-view stereo model incorporating prior constraints indicated by user interaction, which makes it possible to even handle Non-Lambertian surface that surely violates the photo-consistency constraint. Users only need to provide a coarse segmentation and a few user interactions, our system can automatically correct depth and refine boundary. With other priors and occlusion handling, the erroneous depth can be effectively corrected even for very challenging examples that are difficult for state-of-the-art methods.

1. Introduction

With the rapid development and popularity of digital capture and display devices, 3D vision techniques are steadily gaining in importance. Along with the great success of recent 3D films, the problem of how to conveniently produce 3D models and 3D videos receives unprecedented attention.

Given multiple images or video sequences, structure-from-motion and multi-view stereo techniques [12, 23, 34] can be used to recover camera motion as well as dense depth information. However, the robustness is still an issue. Many factors, such as noise, occlusion, textureless regions and reflection, make depth recovery erroneous. These limitations seriously hinder the practicality of many systems in solving real-world problems.

In this paper, we present a semi-automatic multi-view stereo system, which can quickly create or repair the depth from a monocular sequence taken by a moving camera. Our system provides a user-friendly interface where users simply need to draw a few scribbles to coarsely mask the areas containing erroneous depth. Then our system can automatically correct these pixels by incorporating the prior constraints in multi-view stereo. Our system also incorporates shape priors to help correcting the depth of a Non-Lambertian surface which violates multiview stereo constraint. Our method does not require very fine object segmentation with precise boundaries. By using complementary information from multiple frames, not only accurate depth values but also high-quality boundaries can be obtained even for very challenging video sequences. The recovered high-quality depth video can benefit many applications, such as stereoscopic video synthesis, video editing and 3D modeling.

2. Related Work

As our system involves procedures of multi-view stereo, 2D-to-3D conversion and video segmentation, we briefly review related work in these areas.

2.1. Multi-View Stereo and 2D-to-3D Conversion

Although multi-view stereo was studied in many methods [12, 34], automatically obtaining accurate and dense depth information from natural images/videos is still very challenging because there are several issues not addressed. Some interactive image-based modeling methods [28, 27, 25, 32, 10] were proposed. However, these methods can only recover specific types of static objects (such as hair, tree or urban building), a single object [10], or relatively simple and coarse models [28, 25]. They are obviously not enough for many applications needing high-accuracy and complex geometric models.

Recently, Zhang [33] proposed an interactive stereo video matching system, which corrects disparity maps in keyframes and then propagates them to intermediate ones. Similar to other keyframe-based propagation methods, it has the following disadvantages. First, creating the perfect depth map for each key frame may require much user interaction. Second, propagation may introduce and accumulate errors especially around discontinuous boundaries.

*email: zhangguofeng@cad.zju.edu.cn

Third, for high-quality depth map generation in challenging scenes, this method may require dense key frames.

Study in [15, 11, 9] shows that relatively coarse depth with accurate depth order is enough for creating an immersive stereoscopic effect. Therefore, 2D-to-3D conversion methods [11, 9, 17] were developed, which resort to simple user interaction (e.g., drawing a few scribbles) to help infer visually plausible scene depth. Some researchers [16, 29] proposed using object-based tracking with depth propagation to convert 2D videos to 3D content. In short, these methods may produce reasonable depth for stereoscopic video synthesis, but are not enough for high-quality 3D modeling.

2.2. Interactive Video Segmentation/Matting

Several interactive video segmentation/matting techniques [2, 30, 3, 21, 35] have also been developed. Early methods (e.g. [30]) generally assume that the background is known to simplify segmentation. Later on, Bai *et al*. [3] proposed a robust interactive video segmentation system using localized classifiers, which can handle more challenging video sequences. Price *et al*. [21] also proposed to combine kinds of cues to construct a similar video object segmentation framework. Zhong *et al*. [35] presented a bidirectional propagation strategy and combined different classifiers based on a learning-based method. However, these methods are generally designed for handling moving objects, and are not specifically optimized for handling static objects and depth repairing.

3. Framework Overview

The input is a video sequence $\hat{I} = \{I_t | t = 1, ..., n\}$ taken by a moving camera. $I_t(\mathbf{x})$ represents the RGB color of pixel $\mathbf{x}$ in frame t. The output of our system is a set of disparity maps $\hat{D} = \{D_t | t = 1, ..., n\}$. $D_t(\mathbf{x})$ is defined as $D_t(\mathbf{x}) = 1/z_{\mathbf{x}}$ where $z_{\mathbf{x}}$ denotes the depth value of pixel $\mathbf{x}$. We first employ an automatic MVS method [34] which is used in software ACTS 2.0 [1] to automatically recover camera parameters and depth maps for each frame. Then, in the following steps, we repair regions with inaccurate depth. Generally, the user only needs to draw one or a few scribbles, and at most two additional scribbles to indicate depth range or prior if necessary, to highlight the area that contains inaccurate depth. Shape priors if available also can be incorporated into the optimization framework. Then our system automatically corrects the erroneous depth including those around discontinuous boundaries.

4. Fast Area Masking

We first segment the areas with imperfect depth data. Since the video may contain multiple regions with incorrect depth, we segment one region and repair its depth each time. Please note that the segmentation does not need to be very accurate in this step (our supplementary document[1] shows an example with imperfect segmentation results), since our following depth correction and spatio-temporal optimization can automatically refine it.

4.1. Paint Selection by Global Color Model

Similar to other bilayer segmentation methods, we define the area that we would like to complete as "foreground". The others are "background" accordingly. Like paint selection [18], we draw one or a few foreground scribbles to establish foreground color Gaussian Mixture Model (GMM) Φ_f with 6 components in our experiments. Background color model Φ_b is obtained by randomly sampling a number of background pixels, typically forming 8-12 components. Background brush can be used to update Φ_b for better color modeling and improving the segmentation result. We use the graph cut algorithm [5] to minimize the following energy function

$$E_B(\alpha) = \sum_{\mathbf{x}} (E_d(\alpha_{\mathbf{x}}) + w_s \sum_{\mathbf{y} \in N(\mathbf{x})} E_s(\alpha_{\mathbf{x}}, \alpha_{\mathbf{y}})), \quad (1)$$

where $E_d(\alpha_{\mathbf{x}})$ is the data term, encoding the cost when the label of pixel $\mathbf{x}$ is $\alpha_{\mathbf{x}}$. $E_s(\alpha_{\mathbf{x}}, \alpha_{\mathbf{y}})$ is the smoothness term, which penalizes label difference between neighboring pixels. w_s is the smoothness weight and set to 50 in our experiments. $E_d(\alpha_{\mathbf{x}})$ is defined as

$$E_d(\alpha_{\mathbf{x}}) = \begin{cases} -\log p_c(I_{\mathbf{x}}|\Phi_f), & \alpha_{\mathbf{x}} = 1 \\ -\log p_c(I_{\mathbf{x}}|\Phi_b), & \alpha_{\mathbf{x}} = 0 \end{cases} \quad (2)$$

where $I_{\mathbf{x}}$ denotes the RGB color of pixel $\mathbf{x}$. $\alpha_{\mathbf{x}}$ has a binary value. $\alpha_{\mathbf{x}} = 1$ when the pixel belongs to the marked area, i.e., foreground, otherwise, $\alpha_{\mathbf{x}} = 0$. $p_c(I_{\mathbf{x}}|\Phi_f)$ and $p_c(I_{\mathbf{x}}|\Phi_b)$ are the probabilities computed from GMMs.

For adjacent pixels $\mathbf{x}$ and $\mathbf{y}$, the smoothness term $E_s(\alpha_{\mathbf{x}}, \alpha_{\mathbf{y}})$ is defined as

$$E_s(\alpha_{\mathbf{x}}, \alpha_{\mathbf{y}}) = |\alpha_{\mathbf{x}} - \alpha_{\mathbf{y}}| \cdot \exp(-\beta \|I_{\mathbf{x}} - I_{\mathbf{y}}\|^2), \quad (3)$$

where $\beta = (\langle \|I_{\mathbf{x}} - I_{\mathbf{y}}\|^2 \rangle)^{-1}$ [22]. After segmentation, Φ_f and Φ_b are recorded by our algorithm and will be used for color model propagation across frames.

4.2. Robust Propagation with Occlusion Handling

Sequentially, we incorporate shape and color information gathered from each frame and propagate segmentation to the following ones. The process is similar to that of [3]. The major difference is that we estimate a global homography to propagate segmentation instead of passing a set of overlapped local windows.

[1]The supplementary document and video can be found at the website http://www.cad.zju.edu.cn/home/gfzhang/

With the foreground mask of a frame t, we randomly sample pixels, denoted as V_t, with numbers generally smaller than 1000 on the foreground region. Then we employ KLT tracking [19, 24] to find a set of correspondences in frame $t+1$ and denote them as V_{t+1}. A homography is estimated from the tracked correspondences V_t and V_{t+1} using RANSAC [7]. It is used to warp the segmented foreground mask $M^t(\mathbf{x})$ at frame t to frame $t+1$. The warped foreground mask is convolved with a Gaussian Filter to obtain the shape prior $p_s(\mathbf{x})$ of frame $t+1$. We then define the regularization term as $E_r(\alpha_\mathbf{x}) = w_r \sum_\mathbf{x} |\alpha_\mathbf{x} - p_s(\mathbf{x})|$. This term is effective for regularizing segmentation, but is sensitive to occlusion. Adjusting the weight w_r can alleviate this problem but still fail when pixel colors around salient boundaries are quite similar.

In order to robustly handle strong occlusion, we allow the user to explicitly indicate the occlusion boundary, which actually belongs to adjacent areas occluding the foreground. We track this kind of occlusion boundary along with the adjacent background region S_o. The area tracking of S_o can be solved by minimizing the following energy function:

$$E_{\text{tracking}} = \sum_{\mathbf{x}\in S_o} ||I_\mathbf{x} - I'_{\mathbf{x}'}||^2 + \lambda_\Delta \sum_{\mathbf{x}\in\Omega_o} (|\Delta I_\mathbf{x}| - |\Delta I'_{\mathbf{x}'}|)^2, \tag{4}$$

where λ_Δ is a weight and set to 20 in our experiments. Δ is the Laplacian operator and Ω_o denotes occlusion boundary. $\mathbf{x}'$ is the correspondence of $\mathbf{x}$. $\hat{\mathbf{x}}' = A\hat{\mathbf{x}}$, where A is a 2×3 affine matrix. The second term in (4) enforces that the occlusion boundary Ω_o should coincide with the strong intensity/color change. We adopt the Levenberg-Marquardt algorithm to effectively solve (4) for A.

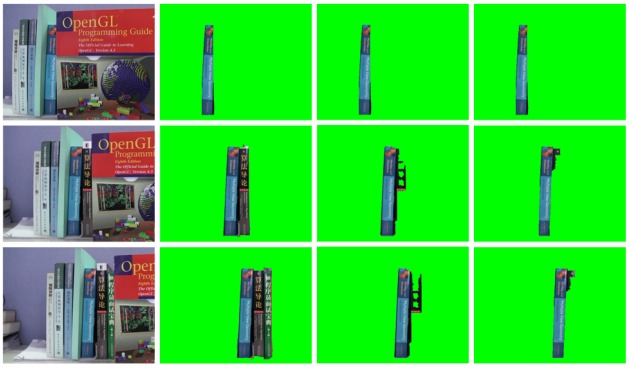
With tracked sub-contours, we sample pixels around Ω_o and assign them as the background V_B if they are in S_o or foreground V_F otherwise. By incorporating this constraint, we define the energy function

$$\begin{aligned} E(\alpha) = \sum_\mathbf{x} (E_d(\alpha_\mathbf{x}) + w_s \sum_{\mathbf{y}\in N(\mathbf{x})} E_s(\alpha_\mathbf{x}.\alpha_\mathbf{y})) + E_r(\alpha_\mathbf{x}) \\ + w_c(\sum_{\mathbf{x}\in V_B} |\alpha_\mathbf{x} - 0|^2 + \sum_{\mathbf{x}\in V_F} |\alpha_\mathbf{x} - 1|^2), \end{aligned} \tag{5}$$

where w_c is a weight and set to 200 in our experiments. We use graph-cut to solve (5). Figure 1 shows a segmentation result and corresponding comparison. Without occlusion handling, the propagated segmentation result in Figure 1(b) contains obvious visual artifacts after a few frames. By considering and coping with occlusion, the segmentation result is successfully propagated over dozens of frames. We have compared our method with the most recent state-of-the-art work [3, 35]. Our propagation method is more robust especially when there is strong occlusion. Figure 2 shows the comparison. Our method can accurately recognize the occluded region and successfully segment it during propagation.

(a) (b) (c)

Figure 1. Propagation comparison. (a) Two selected source frames. (b) Propagation result without occlusion handling. (c) Propagation with occlusion handling.



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

Figure 2. Comparison with PopMat [35] and SnapCut [3]. (a) A few selected source frames. (b) Propagation results obtained by our method. (c) Propagation results obtained by PopMat. (d) Propagation results obtained by SnapCut.

5. Depth Repairing

In this section, we discuss our steps to correct depth in the marked regions. According to multi-view geometry [12], depth of static objects can be accurately computed if matching is perfect. However, stereo matching is vulnerable to occlusion and illumination change, even using state-of-the-art algorithms. Here, we show by incorporating prior constraints into the multi-view stereo model, these general difficulties can be effectively addressed.

5.1. Depth Correction with Occlusion Handling

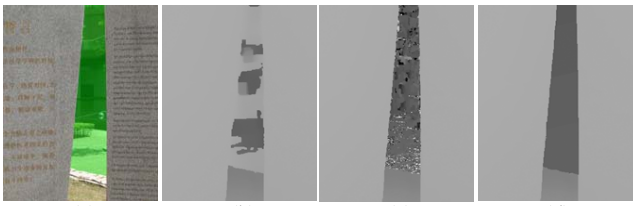
Stereo matching with a global algorithm generally can be formulated in an MRF energy minimization framework as

$$E_D(D_t; \hat{I}) = \sum_{\mathbf{x}\in F_t} (L_d(\mathbf{x}, D_t(\mathbf{x})) + L_s(\mathbf{x})), \tag{6}$$

where L_d is the data cost, defined as

$$L_d(\mathbf{x}, d) = 1 - \frac{1}{|\phi(\mathbf{x}_t, d)|} \sum_{t'\in\phi(\mathbf{x}_t,d)} \frac{\sigma_c}{\sigma_c + ||I_t(\mathbf{x}) - I_{t'}(\mathbf{x}')||}.$$

$\phi(\mathbf{x}_t, d)$ denotes the set of selected frames for pixel $\mathbf{x}_t$ given disparity d that is inversely proportional to actual depth. We



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

Figure 3. Result comparison with and without occlusion handling. (a) One selected frame with marked segment. (b) Original depth map recovered by ACTS. (c) Estimated depth map with occlusion handling. (d) Estimated depth map with occlusion handling and plane fitting.

give the concrete definition of $\phi(\mathbf{x}_t, d)$ later. σ_c is a constant parameter and set to 10 in our experiments. $\mathbf{x}'$ is the corresponding pixel in frame t' for $\mathbf{x}$, given disparity d. F_t denotes the marked region that needs to be updated. $L_s(\mathbf{x})$ encodes disparity smoothness, simply defined as

$$L_s(\mathbf{x}) = \lambda_s \sum_{\mathbf{y} \in N(\mathbf{x})} \min(|D_t(\mathbf{x}) - D_t(\mathbf{y})|, \eta),$$

where λ_s is the smoothness weight and η is the truncated parameter to avoid over-smoothing at depth boundaries. In our experiments, we generally set $\lambda_s = 5/(d_{\max} - d_{\min})$ and $\eta = 0.1(d_{\max} - d_{\min})$, where $[d_{\min}, d_{\max}]$ denotes the disparity range. This energy function can be efficiently solved by loopy BP [6].

Data-cost definition is very important for stereo matching. If the function for the majority of pixels is ill-posed, the disparity estimates would be erroneous. We found most inaccurate regions are caused by significant occlusion. For more reliable matching, it is better to only pick frames where corresponding pixels exist. Thus we need to classify visible pixels and occluded ones for each selected frame. Although there are frame selection algorithms [14], robustness is still a problem. Here, we introduce a more effective method by using segmentation priors.

We adopt the following strategy to first infer disparity range $[d_{\min}, d_{\max}]$. We locally search the maximum and minimal disparity values around boundary of F_t. Generally, a simple assignment to put the largest disparity as $d_{\max}$ and set $d_{\min}$ to zero suffices. Our system also allows the user to pick two disparity values from the recovered disparity map for more accurate disparity range assignment.

Given disparity d, for each pixel $\mathbf{x} \in F_t$, we project it to another frame t'. The projected pixel is denoted as $\mathbf{x}'$. We expand F_t with the bandwidth of 100 pixels to obtain a larger area $\hat{F}_t$. The surrounding area of F_t is defined as $\overline{F}_t = \{\mathbf{x} | \mathbf{x} \in \hat{F}_t \ \& \ \mathbf{x} \notin F_t\}$, where the depth is generally accurate. So we project $\overline{F}_t$ to frame t' according to the estimated depth information. The projected area is denoted as $\overline{F}_{t \to t'}$. If $\mathbf{x}' \in \overline{F}_{t \to t'}$, occlusion generally happens. With the estimated visibility information, we let $\phi(\mathbf{x}_t, d)$ to satisfy one condition – that is, if frame $t' \in \phi(\mathbf{x}_t, d)$, it should hold that $\mathbf{x}' \notin \overline{F}_{t \to t'}$.

Using the above data cost definition, depth of most of regions can be accurately estimated. Figure 3 shows an example. For better handling textureless regions, we represent each repaired region as a 3D plane with parameters $[a, b, c]$ such that $D_t(\mathbf{x}) = ax + by + c$ for each pixel $\mathbf{x} \in F_t$. Estimating disparity variable $D_t(\mathbf{x})$ is equivalent to estimating plane parameters. We use plane fitting [34] to estimate their parameters. If the region is extremely textureless, there is actually no way to estimate its correct depth by matching even using plane fitting.

It is easy for the user to draw a scribble in another region, telling the system that the area to update should have similar disparities or surface normals as this surface. We estimate the 3D plane $[a^*, b^*, c^*]$ of the reference region using least squares. Then we add one of the following terms into the energy function as a soft constraint.

$$L_p(a, b, c) = ||a^* - a||^2 + ||b^* - b||^2 + ||c^* - c||^2 \quad (7)$$

$$L_p(a, b, c) = ||a^* - a||^2 + ||b^* - b||^2. \quad (8)$$

They require that the region to repair has similar plane parameters or normal as the reference one. Our system provides an interface to allow the user to manually select (7) or (8) for plane fitting. By incorporating this prior, (6) is modified to

$$E_D(a, b, c) = \sum_{\mathbf{x} \in F_t} (L_d(\mathbf{x}, D_t(\mathbf{x})) + L_s(\mathbf{x})) + |F_t| \lambda_p L_p(a, b, c), \quad (9)$$

where λ_p is the weight. Similar to [34], we use cubic-Hermite interpolation method to construct the continuous function for $E_D(a, b, c)$ and then effectively solve it by Levenberg Marquardt algorithm.

5.2. Depth Correction with Shape Priors

For a non-Lambertian surface, since it violates photo-constancy, stereo matching is not applicable. Although there are some automatic MVS methods proposed to reconstruct non-Lambertian surfaces, they either can obtain smooth but somehow coarse object models [13], or require the image data to be captured in a special way [31]. Shape priors also can help address this problem [4]. Here, we use a simpler method for depth estimation.

The idea is based on the fact that prior 3D models may be different from the video objects. We thus parameterize prior models, so that they can better fit the object in the image sequence. For example, a cylinder can be represented as many circles in top view. The radius of each circle can be adjusted to represent cones and more complex objects.

The repairing object generally contains many inaccurate 3D points, we need to exclude as many unreliable ones as

possible. For each pixel, we use the following criterion similar to that in [20] to measure the confidence of disparity estimate d_0:

$$C(\mathbf{x}) = \left(\frac{1}{|h(d_0)|} \sum_{d \in h(d_0)} e^{-\frac{{L_d}^2(\mathbf{x},d) - {L_d}^2(\mathbf{x},d_0)}{\sigma^2}} \right)^{-1}, \quad (10)$$

where σ is a constant. $h(d_0) = \{d_0 - 2\Delta d, d_0 - \Delta d, d_0 + \Delta d, d_0 + 2\Delta d\}$ and $\Delta d = 0.02(d_{\max} - d_{\min})$. $C(\mathbf{x})$ is large when the local matching cost has a sharp minimum and $L_d(\mathbf{x}, d_0)$ is a local optimum. We select pixels only when $C(\mathbf{x})$ is larger than a threshold. If necessary, our system also allows the user to draw a few strokes to manually sample reliable 3D points.

We first manually align the prior model with the selected sparse 3D points. Then for each 3D point X_i, we find its closest 3D vertex V_i in the prior model. If $||X_i - V_i||$ is smaller than a threshold, X_i and V_i are regarded as corresponding. With a set of correspondences, we further refine the alignment by solving for an Euclidean transform $[R|T]$ through minimizing function

$$E_{align}(R, T) = \sum_i ||RX_i + T - V_i||^2. \quad (11)$$

We then transform the prior model according to $[R|T]$ to update the alignment, and re-compute the correspondence.

Due to shape difference, there could be residual deviation between the aligned model and video object. We use Laplacian mesh deformation [26] to further reduce it. An example is shown in Figure 4. Because the reconstructed 3D points in the repairing area are noisy, directly taking them as control points to deform the prior model may cause unnatural result, as shown in Figure 4(e). Considering the fact that many object parts are symmetric or rigid, we can constrain the control points instead of directing apply them for deformation. For the example in Figure 4, we estimate the optimal radiuses for the circles which can find corresponding 3D points. Then we uniformly sample the vertices in the adjusted circles as control points to deform the prior model, as shown in (f). We repeat the alignment and deformation in two passes.

After deformation, we render the depth from the deformed model to the mask area, and inpaint depth values. The original and modified disparities for pixel $\mathbf{x}$ are denoted as $d_{\mathbf{x}}$ and $d'_{\mathbf{x}}$. Then we compute confidence of estimated disparity by MVS for each pixel $\mathbf{x}$ in the mask as $u(\mathbf{x}) = e^{-\frac{||d_{\mathbf{x}} - d'_{\mathbf{x}}||^2}{2\sigma_p^2}}$, where σ_p is a constant. $u(\mathbf{x})$ is large when its original and updated disparities are close, which will be used in the following bundle optimization. Figures 4(g) and (h) show the final depth map after bundle optimization, where erroneous depth is corrected and accurate depth details are faithfully preserved.

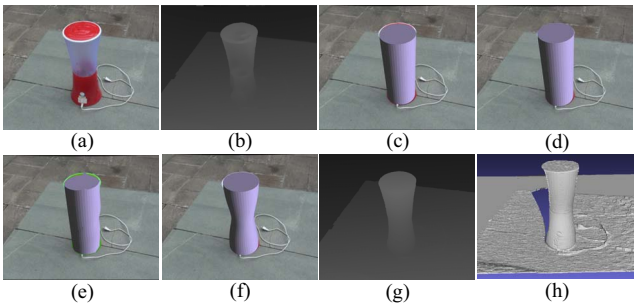


Figure 4. "Bottle" example. (a) One Selected frame. (b) Estimated depth map by ACTS. (c) Manually aligned result. (d) Refined alignment. (e) Direct deformation result using the spare 3D points as control ones. (f) Deformation result with the circle constraint. (g) The corrected depth map. (h) The triangular mesh of (g).

5.3. Spatio-Temporal Depth Optimization

The above steps only coarsely segment the region and estimate its depth independently for each frame. The resulting depth may not be temporally consistent, especially around region boundaries. To improve quality, a spatio-temporal optimization is needed.

Zhang *et al*. [34] proposed bundle optimization and incorporated a geometric coherence constraint to associate multiple frames in a statistical way. We adopt this method but with various modifications. In order to remove boundary artifacts, we dilate marked regions in all frames. The dilated segment in frame t is denoted as F'_t. Assuming the marked region appears in subsequence $[i, j]$, we denote optimized regions as $\{F'_i, F'_{i+1}, ..., F'_j\}$. The depth of F'_t is optimized by solving energy function

$$E'_D(D_t; \hat{I}) = \sum_{\mathbf{x} \in F'_t} (L'_d(\mathbf{x}, D_t(\mathbf{x})) + L_s(\mathbf{x})), \quad (12)$$

where $L'_d(\mathbf{x}, D_t(\mathbf{x}))$ is the data cost combing both color and geometric coherence constraints. It is defined as

$$L'_d(\mathbf{x}, d) = 1 - \frac{1}{|\phi(\mathbf{x}, d)|} \sum_{t' \in \phi(\mathbf{x},d)} \frac{\sigma_c(\mathbf{x}) \cdot p_v(\mathbf{x}, d)}{\sigma_c(\mathbf{x}) + ||I_t(\mathbf{x}) - I_{t'}(\mathbf{x}')||}, \quad (13)$$

where $\sigma_c(\mathbf{x})$ is an adaptive parameter, which will be introduced in Section 5.4. $p_v(\cdot)$ is the geometric coherence constraint as

$$p_v(\mathbf{x}, D_t(\mathbf{x})) = \frac{\sigma_d^2}{\sigma_d^2 + ||P_{t' \to t}(D_{t'}(\mathbf{x}')) - D_t(\mathbf{x})||^2}. \quad (14)$$

$\mathbf{x}'$ is the projected point in frame t'. $P_{t' \to t}(D_{t'}(\mathbf{x}'))$ denotes the transformed disparity value according to the camera parameters between frames t and t'. The transformed disparity is denoted as $P_{t' \to t}(D_{t'}(\mathbf{x}'))$. Please refer to the supplementary document for more details. The geometric coherence constraint requires that $P_{t' \to t}(D_{t'}(\mathbf{x}'))$ equals to

(a) (b) (c)

Figure 5. "Stela" example. (a) One Selected frame. (b) The estimated depth map by ACTS. (c) The refined depth map by our system.

$D_t(\mathbf{x})$. Different from the method of [34], our geometric coherence term is defined in the disparity space instead of image space. For each F'_t, we fix the disparities of all other frames and optimize the disparity in F'_t by minimizing the energy function defined in (12).

5.4. Adaptive Parameter Tuning

Here, we introduce an adaptive parameter tuning scheme to further optimize our system. For marked region F'_t, we measure the temporal photo-consistency error ε_c and disparity consistency error ε_d as

$$\varepsilon_c = \frac{1}{|F'_t|}\sum_{\mathbf{x}\in F'_t}\sqrt{\sum_{t'\in\phi(\mathbf{x},t)}||I_t(\mathbf{x}) - I_{t'}(\mathbf{x}')||^2/|\phi(\mathbf{x},t)|},$$

$$\varepsilon_d = \frac{1}{|F'_t|}\sum_{\mathbf{x}\in F'_t}\sqrt{\sum_{t'\in\phi(\mathbf{x},t)}|P_{t'\to t}(D_{t'}(\mathbf{x}')) - D_t(\mathbf{x})|^2/|\phi(\mathbf{x},t)|}.$$

In the first-pass of spatio-temporal optimization, we generally set $\sigma_d = \max\{1,(\varepsilon_c/K)^2\}\cdot\max\{0.02(d_{\max}-d_{\min}),\varepsilon_d\}$, where K is a constant parameter. The intuition is that if ε_c is small, the estimated disparities are generally good except minor temporal noise exists. In this case, σ_d should be close to the standard deviation of disparity to suppress temporal noise. Contrarily, if ε_c is large, the initialized depth must significantly deviate from the ground truth, so σ_d is set to a larger value to let the photo-consistency constraint dominate the data term. After the first-pass optimization, we adjust $\sigma_d = 0.8\varepsilon_d$. Along the spatio-temporal optimization, both ε_c and ε_d become smaller and smaller in iterations for quick convergence.

The above strategy generally works well. However, if F'_t is a Non-Lambertian surface that violates the photo-consistency constraint, this scheme could be inappropriate. If this happens, we can set σ_c to a large value to weaken the photo-consistency constraint. Especially, if the depth is corrected with shape priors, we can obtain the confidence of original disparities. So we can adjust σ_c for each pixel by $\sigma_c(\mathbf{x}) = 5/u(\mathbf{x})$. If $u(\mathbf{x})$ is small, we weaken the photo-consistency constraint for pixel $\mathbf{x}$. σ_d is generally set to $\min\{0.02(d_{\max}-d_{\min}),\max\{0.001(d_{\max}-d_{\min}),0.8\varepsilon_d\}\}$ in our experiments.

6. Experimental Results

We experiment with different challenging video sequences. Table 1 lists the statistics of the testing sequences.

Seq.	Frames	Repaired Areas	Running Time (min.) Masking	IDC	STO
Rock	115	9	∼10	∼7	∼12
Stela	143	8	∼10	∼9	∼13
Bottle	100	2	∼7	∼5	∼8
Car	121	4	∼15	∼7	∼12
Indoor	141	14	∼15	∼16	∼19
KITTI Seq000	21	5	∼4	∼4	∼2
KITTI Seq070	21	8	∼7	∼6	∼2
KITTI Seq101	21	8	∼5	∼5	∼2
KITTI Seq107	21	6	∼4	∼5	∼2
KITTI Seq186	21	13	∼8	∼7	∼3

Table 1. Running time. IDC denotes initial depth correction, and STO denotes spatio-temporal optimization.

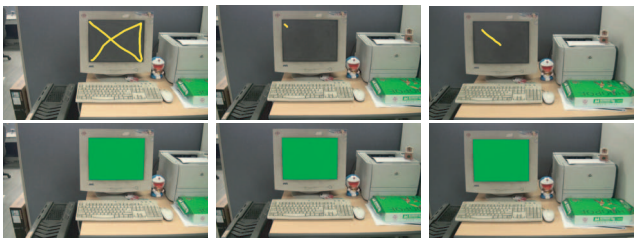


Figure 6. Area masking for "Indoor" example. Top: three edited consecutive key frames (i.e. frames 50, 74 and 98). Yellow strokes denote foreground pixels and blue strokes denote background pixels. Bottom: the segmented region. For masking the black screen, the user only needs to edit one key frame about every 25 frames on average.

The running times are measured on a desktop PC with Intel Xeon E3 3.30GHz (4 cores) CPU, 8GB memory, and Nvidia GeForce GTX 560SE GPU. For a sequence with 100 frames, our system generally takes only $1 \sim 3$ minutes to produce the mask of one area to repair. This speed depends on the repaired area size and appearance/structure complexity. User interaction is only drawing a few strokes. Drawing each stroke generally requires $1 \sim 2$ seconds. For Indoor example with 141 frames, we draw about 150 strokes in total for 14 repaired areas (i.e. 11 strokes for each repaired area on average). Figure 6 shows some edited frames with drawn strokes for this example. Pure interaction time is less than 5 minutes. Depth correction is almost automatic and very efficient. With GPU acceleration, it takes about 2 seconds to optimize 320×240 pixels per frame excluding IO time, so that the user can receive instant response from the system. Our system is rather efficient compared to other semi-automatic methods. For instance, the method proposed in [33] takes about 5-10 minutes to refine each key-frame disparity map, and 10-20 seconds per-frame to compute disparity. To generate similar quality results to ours, this method requires more key frames. So the interaction time is much longer.

Figure 5 shows a challenging outdoor example, which

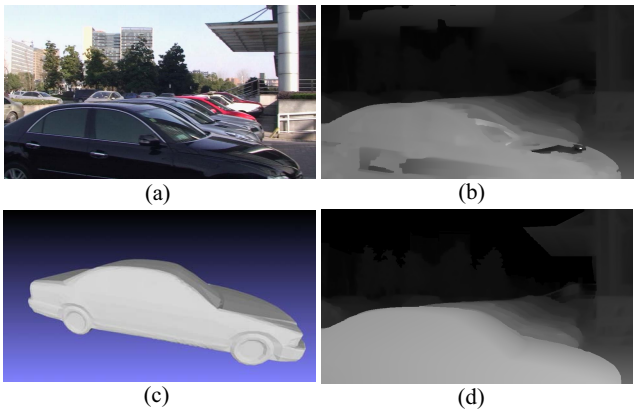


Figure 7. "Car" example. (a) One Selected frame. (b) The estimated depth map by ACTS. (c) The prior model. (d) The refined depth map by our system.

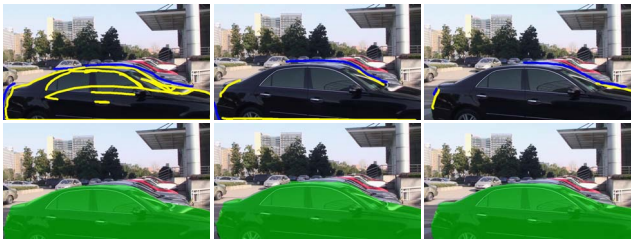


Figure 8. Area masking for "Car" example. Top: three edited consecutive key frames (i.e. frames 61, 67 and 73). Yellow strokes denote foreground pixels and blue strokes denote background pixels. Bottom: the segmented region. For masking the car, the user needs to edit one key frame every 5 frames on average.

contains very strong occlusions in narrow areas (the texture is also weak and geometry is quite complex). We segment and repair 8 areas in total[2]. Figure 7 shows another challenging example, which contains many Non-Lambertian surfaces that cannot be recovered by traditional multi-view stereo methods. It can be evidenced that the recovered depth maps by ACTS contain many distracting artifacts. Especially, the black car contains many reflection areas whose depth cannot be recovered well. We first quickly segment the car, and then use a prior model (Figure 7(c)) to help correct depth. Due to similar colors and specular reflection problems, this example requires more user interactions to segment the car. As shown in Figure 8, the user needs to draw a few additional strokes in a key frame every 5 frames on average. After bundle optimization, a set of spatio-temporally consistent depth maps are achieved. Please refer to the supplementary video for the complete refined depth maps.

We also evaluate our method on the KITTI Vision Benchmark datasets [8], whose data are captured by a stereo

[2]Our supplementary document shows the edited key frames with drawn strokes for some key repaired regions.

Figure 9. Results of KITTI sequence 000107. Top: two selected frames. Middle: the initialized depth maps by ACTS. Bottom: the refined depth maps by our system.

camera on a moving car. Each sequence (*we only use left sequence*) contains 21 frames, and neighboring frames contain much larger baselines than normal video sequences, which makes area masking more difficult. We experiment with 5 sequences in different scenes. For example, the sequence in Figure 9 is with thin structures and several cars with specular surfaces. With these challenges, the recovered depth maps by ACTS contain many errors. With our system, accurate and spatio-temporally consistent depth maps are obtained. Please refer to our supplementary video for the complete results.

7. Discussion and Conclusions

We have presented a rapid semi-automatic multiview stereo system for recovering high-quality depth maps from a monocular sequence taken by a moving camera. Different from traditional interactive 3D modeling systems, the user only needs to give a few simple operations in our system. The erroneous depth values then can be effectively and intelligently corrected, without requiring tedious manual modeling. A quick segmentation tool is provided in our system, which allows the user to conveniently indicate areas to update by drawing a few strokes. By smartly incorporating prior constraints indicated by the user or prior models, the new multi-view stereo energy function can effectively correct depth and generate more accurate object boundaries.

Our method is based on multi-view geometry and is restricted to static scene handling. For moving objects, since their depth cannot be estimated by multiview geometry, our method cannot be applied. Extending our work to properly cope with dynamic scenes will be our future work.

Acknowledgements

The authors would like to thank Qianling Li for his kind help in this work. This work is partially supported by NSF of China (Nos. 61272048 and 61103104), the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20110101130011), the Fundamental Research Funds for the Central Universities (No.

2014FZA5017), a Foundation for the Author of National Excellent Doctoral Dissertation of PR China (No. 201245), and a grant from the Research Grants Council of the Hong Kong Special Administrative Region (Project No. 413113).

References

[1] ACTS 2.0: Automatic Camera Tracking System. http://www.zjucvg.net/acts/acts.html, 2014.

[2] A. Agarwala, A. Hertzmann, D. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph.*, 23(3):584–591, 2004.

[3] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. *ACM Trans. Graph.*, 28(3), 2009.

[4] S. Y.-Z. Bao, M. Chandraker, Y. Lin, and S. Savarese. Dense object reconstruction with semantic priors. In *CVPR*, pages 1264–1271, 2013.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[6] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

[7] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[9] M. Guttmann, L. Wolf, and D. Cohen-or. Semi-automatic stereo extraction from video footage. In *ICCV*, 2009.

[10] M. Habbecke and L. Kobbelt. An intuitive interface for interactive high quality image-based modeling. *Comput. Graph. Forum*, 28(7):1765–1772, 2009.

[11] P. Harman, J. Flack, S. Fox, and M. Dowley. Rapid 2D to 3D conversion. In *Proc. SPIE 4660, Stereoscopic Displays and Virtual Reality Systems IX*, pages 78–86, 2002.

[12] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[13] H. Jin, S. Soatto, and A. J. Yezzi. Multi-view stereo reconstruction of dense shape and complex appearance. *International Journal of Computer Vision*, 63(3):175–189, 2005.

[14] S. B. Kang and R. Szeliski. Extracting view-dependent depth maps from a collection of images. *International Journal of Computer Vision*, 58(2):139–163, 2004.

[15] J. J. Koenderink, A. J. van Doorn, A. M. L. Kappers, and J. T. Todd. Ambiguity and the 'mental eye' in pictorial relief. *Perception*, 30(4):431 C 448, 2000.

[16] Z. Li, X. Xie, and X. Liu. An efficient 2D to 3D video conversion method based on skeleton line tracking. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pages 1 – 4, 2009.

[17] M. Liao, J. Gao, R. Yang, and M. Gong. Video stereolization: Combining motion analysis with user interaction. *IEEE Trans. Vis. Comput. Graph.*, 18(7):1079–1088, 2012.

[18] J. Liu, J. Sun, and H.-Y. Shum. Paint selection. *ACM Trans. Graph.*, 28(3), 2009.

[19] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981.

[20] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nistér, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *ICCV*, pages 1–8, 2007.

[21] B. L. Price, B. S. Morse, and S. Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *ICCV*, pages 779–786, 2009.

[22] C. Rother, V. Kolmogorov, and A. Blake. "Grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[23] 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)*, pages 519–528, 2006.

[24] J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600, 1994.

[25] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3D architectural modeling from unordered photo collections. *ACM Trans. Graph.*, 27(5):159, 2008.

[26] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Symposium on Geometry Processing*, pages 175–184, 2004.

[27] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan. Image-based tree modeling. *ACM Trans. Graph.*, 26(3):87, 2007.

[28] 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.*, 26(3):86, 2007.

[29] H.-M. Wang, C.-H. Huang, and J.-F. Yang. Depth maps interpolation from existing pairs of keyframes and depth maps for 3d video generation. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3248 – 3251, 2010.

[30] J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 24(3):585–594, 2005.

[31] S. Wanner and B. Goldluecke. Reconstructing reflective and transparent surfaces from epipolar plane images. In *Pattern Recognition - 35th German Conference, GCPR 2013, Saarbrücken, Germany, September 3-6, 2013. Proceedings*, pages 1–10, 2013.

[32] J. Xiao, T. Fang, P. Tan, P. Zhao, E. Ofek, and L. Quan. Image-based façade modeling. *ACM Trans. Graph.*, 27(5):161, 2008.

[33] C. Zhang, B. L. Price, S. Cohen, and R. Yang. High-quality stereo video matching via user interaction and space-time propagation. In *3DV*, pages 71–78, 2013.

[34] G. Zhang, J. Jia, T.-T. Wong, and H. Bao. Consistent depth maps recovery from a video sequence. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(6):974–988, 2009.

[35] F. Zhong, X. Qin, Q. Peng, and X. Meng. Discontinuity-aware video object cutout. *ACM Trans. Graph.*, 31(6):175, 2012.