Recovering Depth Map from Video with Moving Objects

Hsiao-Wei Chen and Shang-Hong Lai

Computer Science, National Tsing Hua University,
No. 101, Section 2, Kuang-Fu Road, Hsinchu, Taiwan 30013, R.O.C.
wei@hotmail.com, lai@cs.nthu.edu.tw

Abstract. In this paper, we propose a novel approach to reconstructing depth map from a video sequence, which not only considers geometry coherence but also temporal coherence. Most of the previous methods of reconstructing depth map from video are based on the assumption of rigid motion, thus they cannot provide satisfactory depth estimation for regions with moving objects. In this work, we develop a depth estimation algorithm that detects regions of moving objects and recover the depth map in a Markov Random Field framework. We first apply SIFT matching across frames in the video sequence and compute the camera parameters for all frames and the 3D positions of the SIFT feature points via structure from motion. Then, the 3D depths at these SIFT points are propagated to the whole image based on image over-segmentation to construct an initial depth map. Then the depth values for the segments with large re-projection errors are refined by minimizing the corresponding re-projection errors. In addition, we detect the area of moving objects from the remaining pixels with large re-projection errors. In the final step, we optimize the depth map estimation in a Markov random filed framework. Some experimental results are shown to demonstrate improved depth estimation results of the proposed algorithm.

Keywords: Depth map recovery, structure from motion, Markov random field.

1 Introduction

In recent years, 3D imaging industry emerges rapidly. More and more movies take advantage of advanced 3D technology to reconstruct 3D motion or 3D scene for movie production or post-processing, which would produce amazing visual effect as if it were really happening. With the rapid development of the 3D movies, the 3D television also follows the trend. In each mall, we can see 3D TV exhibition in the spotlight. The digital camera also starts to include some 3D imaging function that supports 3D stereo images, 3D panorama shooting mode, and so on. In addition to these products, 3D digital photo frame and 3D digital printing are also the new related 3D products, and there will be more 3D products in the future.

For those 3D display monitors, it needs the image files that contain 3D information. If we can convert the captured 2D images or videos into the 3D format, we can view the images and videos with 3D display. Thus, the 2D-to-3D media conversion is very important for the development of the 3D imaging industry.

Y.-S. Ho (Ed.): PSIVT 2011, Part II, LNCS 7088, pp. 335–346, 2011.

Most of the 2D-to-3D conversion technologies either cannot provide satisfactory results or require additional equipment, such as the stereo cameras or 3D range sensors. Although using the additional devices can provide more accurate results, they are more expansive and they cannot work for previous media. Therefore, we develop a new approach to estimating the depth map from a video sequence. The algorithm would automatically detect moving objects and estimate the depth map from a video sequence by considering both the spatial and temporal coherence in the video.

2 Previous Works

Depth estimation is an important and challenge problem in computer vision. The methods of depth map reconstruction can be roughly divided into two types: monocular vision based and multi-view based approaches. We briefly discuss them in this section.

2.1 Monocular Vision

In monocular vision, most methods used learning techniques or additional assumptions to obtain more information in a single image. Saxena et al. [1] proposed a learning algorithm to reconstructed 3D structure environment from a single image. They used supervised learning to learn how different visual cues are associated with different depths and used Markov Random Field (MRF) model to combine all the information. They also added other properties, including image features and a set of plane parameters of superpixels, into MRF to estimate the 3D positions and orientations for all superpixels. Unlike the previous approach which attempted to map from appearance features to depth, the algorithm in [2] first used semantic segmentation based on learning algorithm in an image to identify the positions of sky, tree, road, etc., in an image. They constructed a descriptor for each pixel, which includes local appearance features and global geometry features, and used a learning model to predict the initial depth map. Next they used some geometric constrains added to MRF framework to construct a smooth depth map. In addition, Hoiem [9] recovered the occlusion boundaries and depth ordering in the scene. They proposed a hierarchical segmentation process based on agglomerative merging to re-estimate boundary strength as the segmentation progresses.

2.2 Multi-view Vision

The most popular topic in multi-view vision is stereo reconstruction. Sun et al. [10] formulates the stereo matching problem in an MRF framework and solved it by belief propagation (BP) which works via iteratively passing messages to neighbor nodes. The drawback of this approach is that it spends too much time in processing the messages. Therefore, Felzenszwalb et al. [11] proposed a hierarchical BP algorithm to significantly reduce the computational complexity.

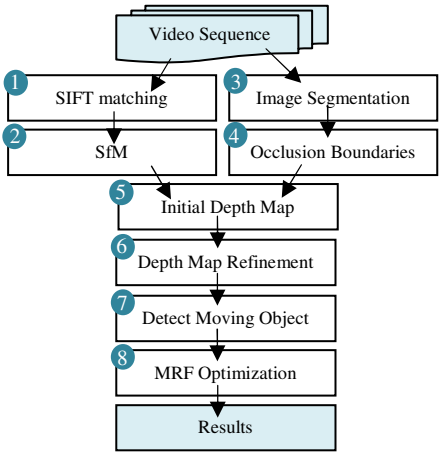

Fig. 1. The flowchart of the proposed method

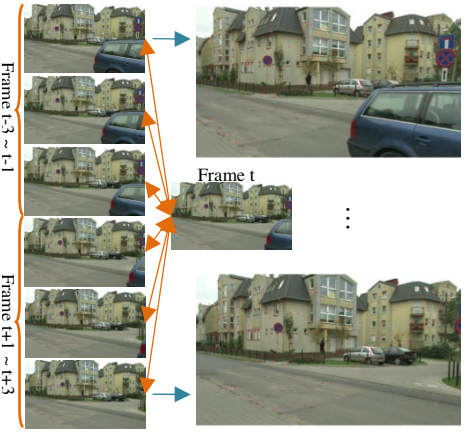

Fig. 2. The flowchart of SIFT matching

Another type of multi-view vision is for a video or multi-images captured from different views by using structure from motion (SfM) to compute the camera parameters for the multi-view images. Most of the SfM methods [5] defined energy functions on the 3D volume and used MRF to obtain the 3D surface. Recently, Zhang et al. [3, 4] used not only color constancy constraint but also the geometric coherence constraint which can maintain the temporal coherence in the video. For estimating the accurate disparity in textureless region, they added segmentation information based on color information to construct the initial disparity map. Next they used iteratively optimization which they called bundle optimization to refine the disparities in a pixelwise manner. More recently, Newcombe and Davison [6] used point-based structure from motion (SfM) to compute the camera poses first, computed the optical flow across frames, and used the triangulation method iteratively to optimize the 3D surface.

3 Proposed Method

Fig. 1 is the flow chart of the proposed depth map estimation algorithm. The goal for the first four steps is to collect information for recovering depth map; and the goal from 5th to 8th steps is to reconstruct and refine the depth map. First, we apply SIFT feature matching [12] to extract the corresponding points in the neighbor frames and use these points to compute the camera parameters by the SfM algorithm [13][14]. Then, we apply the mean-shift segmentation algorithm [7] and determine the occlusion boundaries based on [9] to classify the sky, ground, vertical regions. By using the 3D coordinate points from SfM, we construct a rough depth map based on the above over-segmentation. Next, we refine the depth map in the segments with large re-projection errors. However, the re-projection errors are still large after the depth refinement process in the regions containing moving objects. Thus, we use these regions as seeds to detect the regions of moving objects. In the final step, we optimize the depth map in the MRF framework to obtain accurate depth estimation.

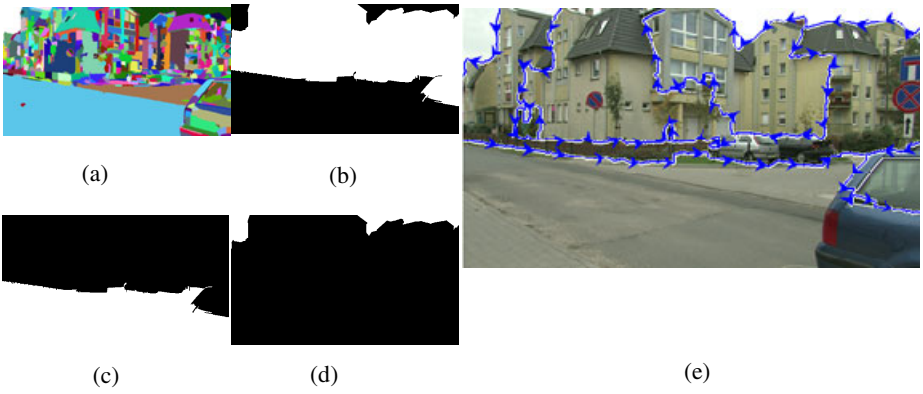

Fig. 3. (a) Mean-shift segmentation. (b-d) The mask of "vertical", "ground" and "sky". (e) Occlusion boundaries. The left side of the arrow is front of the right side of the arrow.

3.1 SIFT Matching

To construct the depth map from video, we would like to establish the relationship between the neighboring frames. The first step is to find reliable corresponding points in the video. Only having the corresponding points, we can calculate the camera parameters for all frames.

SIFT is an algorithm to detect and describe interest points in the images. The SIFT feature description is robust against the changes due to image scale, noise, illumination and rotation. For the above advantages, we apply the SIFT matching to compute the corresponding points across the six neighboring frames. Fig. 2 depicts the idea of the SIFT matching step.

3.2 Structure from Motion

In this step, we apply the SfM algorithm [13][14] to the SIFT correspondence points to estimate the camera parameters and 3D structures of these SIFT feature points. To estimate the depth map at a specific frame, I used seven neighboring consecutive frames into the SfM algorithm to compute the camera parameters and the 3D structures. These 3D points are the key to reconstruct the initial depth map in our algorithm. The details will be described in section 3.5.

3.3 Image Segmentation

Image segmentation is to divide an image into multiple homogeneous segments. The pixels in the same segment have similar visual characteristics, such as color, location, texture, etc. For these reasons, I use segmentation to help us improve the depth estimation, which will be described in section 3.6. In this paper, the mean-shift algorithm is chosen for the image over-segmentation, and an example result is depicted in Fig. 3(a). The segments are used for recovering the occlusion boundaries from an image, which will be discussed subsequently.

1. Set known 3D points
1.1 A sparse of 3D points from SfM

2. Set depth to each segments
2.1 For each segment, construct a smooth depth map by (2) if it has enough information
2.2 Propagate the depth value to neighbor segments
2.3 Repeat step 2.1 and 2.2

Fig. 4. The flowchart of making initial depth map

Fig. 5. The initial depth map

3.4 Occlusion Boundaries

One of the important key in reconstructing depth map is occlusion relation. Owing to the change of view, some objects would be occluded by something in front of them, which would cause some problems in depth estimation. For example, sky is always covered by houses, trees, and so on. When we project a pixel to other frames, the labels may be different due to occlusion. This would lead to large errors in the image re-projection.

Hoiem et al. proposed an algorithm [9] to recover the occlusion boundaries and detecting the regions of "sky", "vertical" and "ground" from an image. We incorporate their results of the occlusion boundaries and segment type classification into the depth map estimation framework to optimize the depth map estimation. An example of applying the algorithm in [9] is depicted in Fig. 3.

3.5 Initial Depth Map

We have collected a lot of useful information for reconstructing depth map in the previous steps. Therefore, we use all the information to construct an initial depth map. The main idea is that the neighboring pixels which have similar colors may have similar depth. In the SfM step, in addition to the camera parameters for all frames, we also obtain a sparse set of 3D feature points. Thus, these 3D points are used to propagate to the entire image. The flowchart of this step is shown in Fig. 4.

First, we scan all the 3D points to decide the range of the depth in the current frame, and record which pixels are already assigned with depth values. Then, for each segment, we counted how many pixels are already assigned with depth values. If the number is less than a threshold (5 in our implementation), we do nothing because there is not enough information to assign the depth for this segment. Otherwise, we consider each segment as a plane, and construct a smooth depth map for the segment. Hence, we set a linear plane, as in eq. (1), for each segment. Note that x and y are the coordinates of the pixel, a, b, c are the plane parameters, and d is the depth value. By using the least-square fitting, we calculate parameter a, b, c from some known points, $(x_1, y_1) \ldots (x_n, y_n)$ with the relation given in eq. (2).

$$d = ax + by + c \tag{1}$$

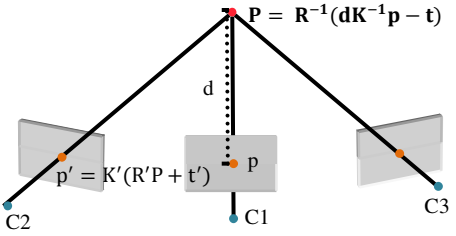

Fig. 6. The chart of projecting to the neighbor frame

Fig. 7. The depth map after refining

$$\begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \tag{2}$$

Not all segments have enough information to construct the depth. Hence, we propagate the depth from the known segments to unknown segments. In the edge of known segment and unknown segment, we check the similar degree of color. If the color is similar, then set the same depth value to the neighboring pixels. The steps of constructing depth and propagating depth are iterative. Fig. 5 depicts some results in this step. The red regions represent no depth information.

3.6 Depth Map Refinement

In the previous step, the initial depth map is rough and there is no depth information in some segments. Hence, in this step I would correct those segments which cause big error or have no depth information in the depth map.

First, I compute the projection error for each pixel in the whole image. The processing of projection is shown in Fig. 6. The blue points are the camera position and the orange points are the 2D coordinate in each frame. The red point is the corresponding 3D point of the orange points in the real world. Assuming the current frame is in the center of Fig. 6. p is one of pixel in the current frame. Giving the depth value of p and camera parameters of current frame and target frame, we can compute the corresponding coordinates, p', in target frame by (3).

$$P = R^{-1}(dK^{-1}p - t) \ , \ p' = K'(R'P + t') \tag{3}$$

where p is the 2D image coordinate of a pixel, represented as [x y 1]' in the homogeneous coordinate, P is the 3D coordinate of p in the world coordinate, represented as [X Y Z]', K, R, t are the camera parameters at the current frame, K', R', t' are the camera parameters at the target frame, and p' is the corresponding coordinate of p in the target frame. If the depth value is accurate, then p and p' must have similar colors. Therefore, we define the re-projection error as a measure of difference between the colors at p and p'. Here, we use l_2-norm to measure the difference between two colors in our implementation.

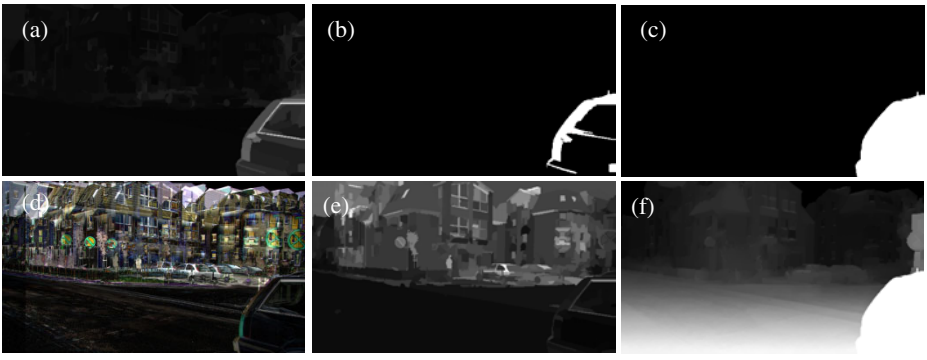

Fig. 8. (a) is the average error for each segments after depth refined. (b) is the seeded segments. (c) is the full region of moving object. (d) is the final error map based on pixels. (e) is the final error map based on segments. (f) The depth mp after optimization of MRF.

Due to the occlusion problem around the object boundary, we compute the average re-projection error for each segment instead of a pixel. If the average re-projection error in one segment is larger than a threshold, it denotes the depth value in this segment is inaccurate. For these segments, we search a new depth value within the depth range by finding the minimum re-projection error for the segment. Fig. 7 depicts an example of the depth map after the above refinement process.

3.7 Moving Object Detection

Most of the proposed approaches for the depth map reconstruction from a video sequences, such as [3], [4], and [6], do not account for moving objects in the video sequence. They all assume the scene is static; but in practice it is very common for the videos to contain moving objects.

It is not suitable to compute depth of moving objects by multi-view camera projections due to the unknown object motion. From the previous step, we have the average re-projection error for each segment after the depth map refinement, as shown in Fig. 8(a). However, there are still some segments with large errors, which means these segments are likely to be the moving object. Thus, we use these segments, as shown in Fig. 8(b), as the seeds to determine the regions of moving objects. Not all segments belonging to moving objects have large re-projection errors, just like the body of car in Fig. 8(a). Next, all the SIFT matching points in the seeded segments are used to calculate the displacement of moving object in the neighboring frames. The displacement is the difference of two corresponding points in x and y axes. To assure robust displacement estimation, we choose the median of all the displacements instead of mean. The relation equations are given in eq. (4) and (5), where p and p' are the corresponding points in the current frame and a neighboring frame, D_x and D_y are the displacements in x and y axes, and $\widehat{D_x}$ and $\widehat{D_y}$ are the final displacement.

$$D_{xi} = p_{xi} - p'_{xi} \ , \quad D_{yi} = p_{yi} - p'_{yi} \tag{4}$$

$$\widehat{D_x} = median_i\ (D_{xi})\ ,\ \widehat{D_y} = median_i\ (D_{yi}) \tag{5}$$

Because we assume the moving object in the neighboring frame and current frame would be the same shape and not be deformed. Then, we shift the neighboring frame by the displacement and subtraction by the current frame. We call the result as an error map. In our experiment, we use two neighboring frames, so we have two error maps. These two error maps are summed to form a combined error map. The final error map in Fig. 8(d) shows the probability of moving object. The pixels with darker color are more likely to be the moving object.

The process of finding the moving object is also based on segments. Therefore, the combined error map based on segments is depicted in Fig. 8(e). We use the seeded segments as the center, and visit the neighboring segments to merge the neighboring segment if the associated error is small. We repeat this segment merging process iteratively until convergence. Fig. 8(c) shows the region of detected moving object in this step.

3.8 MRF Optimization

The previous steps of reconstructing and refining depth map are all segment-based. Because these steps do not account for the relationship between neighboring segments, the computed depth map usually contains obvious errors. To improve the depth map estimation, we integrate the information in a Markov Random Field (MRF) framework to optimize the depth map.

Let the depth map of the current frame be represented as D. The image of the current frame is denoted by I, and $\widehat{\mathrm{I}'} = \{I'_t\ |\mathrm{t} = 1, \dots, \mathrm{n}\}$ is the set of reference frames. Then, we define the following energy function for MRF:

$$\mathrm{E}\left(\widehat{D};\mathrm{I},\widehat{I'}\right) = E_D\left(\widehat{D};\ \mathrm{I},\widehat{I'}\right) + \alpha\, E_s\left(\widehat{D};\ \mathrm{I}\right) \tag{6}$$

where the data term $\mathrm{E_D}$ measures the projection error, the smoothness term $\mathrm{E_S}$ represents the smoothness on the depth map between neighboring pixels, And α is the weight used to balance these two terms. $\widehat{D}$ is the refined depth map by MRF.

The definition of the data term $E_D(\widehat{D};\widehat{I'})$ is given in eq. (7). It computes the color difference between the corresponding points in the neighboring frames based on the current depth estimate.

$$E_D\left(\widehat{D};\ \mathrm{I},\widehat{I'}\right) = \sum_{p\in \mathrm{I}}\sum_{t=1}^{n} \min\left(abs\left(I(p) - I'_t(p_t)\right), \delta\right) \tag{7}$$

$$p_t = K_t\left(R_t\left(R^{-1}\left(\widehat{D}(p)K^{-1}p - t\right)\right) + t_t\right), \tag{8}$$

$\mathrm{p_t}$ is the corresponding point of a pixel p in the $\mathrm{t^{th}}$ reference frame, and δ is a threshold for the color difference. The symbols $\mathrm{K_t}$, $\mathrm{R_t}$, $\mathrm{t_t}$ denote the camera parameters at the $\mathrm{t^{th}}$ reference frame, and K, R, t are the camera parameters at the current frame. $\widehat{D}(p)$ is the depth value of p.

The smoothness term is defined by

$$E_s(\widehat{D};\, \mathrm{I}) = \sum_{p_1 \in I} \sum_{p_2 \in N(p_1)} \frac{abs(\widehat{D}(p_1) - \widehat{D}(p_2))}{abs(\mathrm{I}(p_1) - \mathrm{I}(p_2)) + 1} \tag{9}$$

If p_1 and p_2 are neighbors, then their absolute difference in depth are divided by the corresponding absolute color difference, as in the denominator, and then added into smoothness term.

There are many algorithms based on MRF framework [16], and we choose graph-cut with swap algorithm [17][18] to refine our final depth map by minimizing the energy function given in eq. (6). The pixels belonging to moving objects, as determined in section 3.7, are not involved in the depth map refinement. After the optimization, we set an appropriate depth value to each of these regions of moving objects. We assume the moving object is usually vertical on the ground; therefore we assign the depth values of the pixels on the bottom of the moving object region as the depth for the region. An example of the MRF optimization result is depicted in Fig. 8(f).

4 Experiment Results

We apply in the proposed depth estimation algorithm to six real data sets. The data sets can be classified into two types: static scene and the scene containing moving objects. There are three videos belonging to static scene, i.e. "Road", "Stair" and "Temple" [3][4], and we will compare the results with those in [3][4]. There are three videos containing moving objects. The videos "lovebird1" and "lovebird2" come from [19] and "Poznań" comes from [20].

The experimental results on the dataset "Poznań" are shown in Fig. 9-11. We focus on the passing vehicle in this video. In addition, there is a pedestrian in Fig. 9-11, but we only detect the moving person in Fig. 9. The key is the speed of the person is too slow to be detected as a moving object. Nevertheless, the result of depth map estimation in this example is acceptable. For the three video sequences, we detect the moving object successfully, and this helps to recover accurate depth maps.

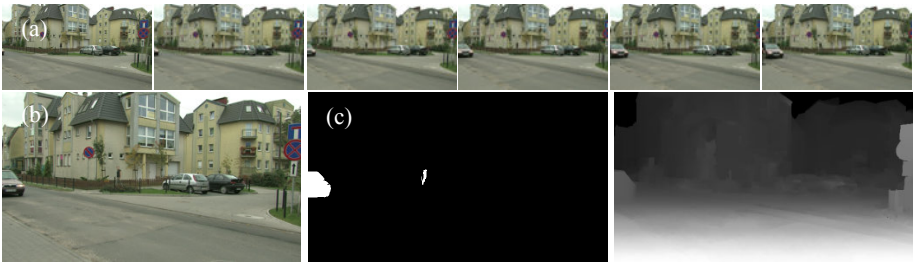

Fig. 9. The 77th frame in the data set "Poznań". (a) The reference frames. (b) The current frame. (c) The white region is the detected moving object. (d) Final depth map.

For the other datasets, we focus on the moving people in "lovebird1" and "lovebird2" videos. However, because of the high frame rate in these two videos, the movement of people is quite small in the neighboring frames. Therefore, we cannot detect moving object in these two videos, but it does not affect our results. The final estimated depth maps are depicted in Fig. 12. Although there is slight movement in these two videos, we still can obtain accurate depth map estimation.

Although we focus on handling moving object in this paper, we also test our algorithm on the three datasets of static scene. The results are shown in Fig. 13 and we also compare the depth estimation results with those by the state-of-the-art method [4]. From the figure, we can see our results are comparable to the results from [4] in these experiments for the static scene.

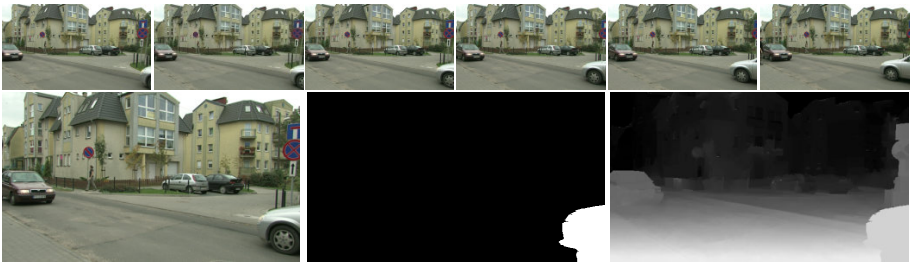

Fig. 10. The 130th frame in the data set "Poznań". (a) The reference frames. (b) The current frame. (c) The white region is the detected moving object. (d) Final depth map.

Fig. 11. The 427th frame in the data set "Poznań". (a) The reference frames. (b) The current frame. (c) The white region is the detected moving object. (d) Final depth map.

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

Fig. 12. The (a)&(c) images and (b)&(d) estimated depth maps for the 4th frame in the "lovebird1" and "lovebird2" sequences, respectively

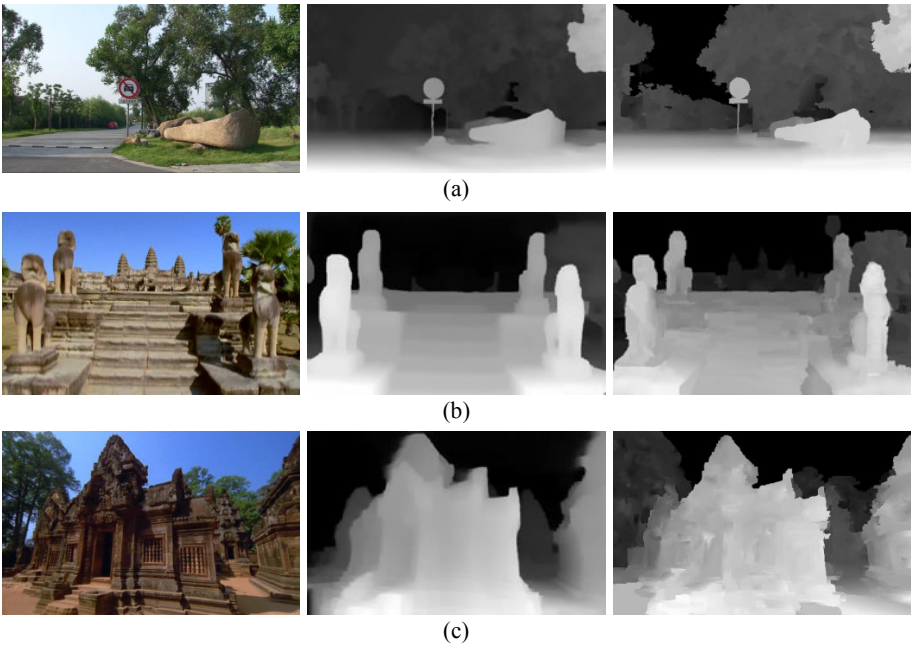

Fig. 13. From left to right: an image sample, depth map estimated by [4], depth map estimated by the proposed algorithm for the (a) Road, (b) Stair, and (c) Temple video sequences, respectively

5 Conclusion

In this paper, we proposed a robust system that reconstructs depth map from a video sequence automatically. In our system, we employ several computer vision technologies to help us construct accurate depth map from video. We integrate SIFT feature matching, SfM, mean-shift over-segmentation, occlusion boundary analysis to obtain some 3D information. To deal with moving object, we detect the segments of moving objects by selecting pixels with large re-projection errors as seeds in an iterative merging process. Finally, we integrate the 3D information in an MRF formulation to optimize the depth map. We demonstrate the proposed algorithm can provide satisfactory depth estimation results for videos with moving objects.

References

1. Saxena, A., Sun, M., Ng, A.Y.: Make3D: Learning 3D Scene Structure from a Single Still Image. IEEE Trans. on Pattern Analysis and Machine Intelligence (2008)
2. Liu, B., Gould, S., Koller, D.: Single Image Depth Estimation From Predicted Semantic Labels. In: CVPR 2010 (2010)
3. Zhang, G., Jia, J., Wong, T., Bao, H.: Recovering Consistent Video Depth Maps via Bundle Optimization. In: CVPR (2008)

4. Zhang, G., Jia, J., Wong, T., Bao, H.: Consistent Depth Maps Recovery from a Video Sequence. IEEE Trans. on Pattern Analysis and Machine Intelligence 31(6), 974–988 (2009)
5. Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In: CVPR (2006)
6. Newcombe, R.A., Davison, A.J.: Live Dense Reconstruction with a Single Moving Camera. In: CVPR (2010)
7. Comanicu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. on Pattern Analysis and Machine Intelligence (May 2002)
8. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient Graph-Based Image Segmentation. International Journal of Computer Vision 59(2) (September 2004)
9. Hoiem, D., Efros, A.A., Hebert, M.: Recovering Occlusion Boundaries from an Image. In: IJCV (2010)
10. Sun, J., Shum, H.Y., Zheng, N.N.: Stereo Matching Using Belief Propagation. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2351, pp. 510–524. Springer, Heidelberg (2002)
11. Felzenszwalb, P., Huttenlocher, D.: Efficient belief propagation for early vision. In: IJCV, pp. 1–8 (2007)
12. Pele, O., Werman, M.: A Linear Time Histogram Metric for Improved SIFT Matching. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 495–508. Springer, Heidelberg (2008)
13. Martinec, D., Pajdla, T.: 3D Reconstruction by Fitting Low-Rank Matrices with Missing Data. In: CVPR 2005, pp. 198-205, IEEE (June 2005)
14. Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. Intern. Journal of Computer Vision 59(3), 207–232 (2004)
15. Alsabti, K., Ranka, S., Singh, V.: An Efficient k-means Clustering Algorithm. Pattern Recognit. Lett. 14(10), 763–769 (1993)
16. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A Comparative Study of Energy Minimization Methods for Markov Random Fields. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part II. LNCS, vol. 3952, pp. 16–29. Springer, Heidelberg (2006)
17. Boykov, Y., Veksler, O., Zabih, R.: Fast Approximate Energy Minimization via Graph Cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(11), 1222–1239 (2001)
18. Kolmogorov, V., Zabih, R.: What Energy Functions can be Minimized via Graph Cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence 26(2), 147–159 (2004)
19. Um, G., Bang, G., Hur, N., Kim, J., Ho, Y.-S.: Test Sequence "Lovebird1&2"
20. Domański, M., Grajek, T., Klimaszewski, K., Kurc, M., Stankiewicz, O., Stankowski, J., Wegner, K.: Poznań Multiview Video Test Sequences and Camera Parameters. ISO/IEC JTC1/SC29/WG11 MPEG 2009/M17050, Xian, China (October 2009)