Interactive Video Layer Decomposition and Matting

Yanli Li[1,2], Zhong Zhou[1,2], and Wei Wu[1,2]

[1] State Key Lab. of Virtual Reality Technology and Systems, Beihang University
[2] School of Computer Science and Engineering, Beihang University
liyl@vrlab.buaa.edu.cn

Abstract. The problem of accurate video layer decomposition is of vital importance in computer vision. Previous methods mainly focus on the foreground extraction. In this paper, we present a user-assisted framework to decompose videos and extract all layers, which is built on the depth information and over-segmented patches. The task is split into two stages: i) the clustering of over-segmented patches; ii) the propagation of layers along the video. Correspondingly, this paper has two contributions: i) a video decomposition method based on greedy over-segmented patches merging; ii) a layer propagation method via iteratively updating color Gaussian Mixture Models(GMM). We test this algorithm on real videos and verify that it outperforms state-of-the-art methods.

1 Introduction

Video decomposition is one of the most fundamental vision tasks. It extracts multiple layers from videos, which can be further used for kinds of Augmented Reality applications. Generally, it solves two problems: layer clustering and layer segmentation. The first problem, which has been extensively studied in the space clustering field, is to estimate the number of layers in every frame. The second problem is to assign each pixel to the corresponding layer.

For the last decades, researchers have presented various approaches for this task, which lie in the fields of motion segmentation and figure-ground separation. However, most motion segmentation methods fail to accurately separate layers, mainly due to an improper energy formulation and the unreliable optical flow fields, while most figure-ground separation approaches only focus on the foreground object extraction, they seldom consider the multi-layer separation.

In this paper, we provide an interactive framework to decompose videos. It combines the merits of motion segmentation and figure-ground separation. With only several clicks, the user can accurately decompose the video into multiple layers. Comparing with motion segmentation methods, our algorithm can segment more "meaningful" layers, and the fine information is preserved well. Compared to figure-ground separation methods, our algorithm is less labor-intensive and can soft-segment every layer. Fig. 1 provides a high level overview of the algorithm pipeline. Our algorithm is based on the depth information and over-segmented patches. In Stage I, we utilize a greedy bottom-to-up scheme to merge

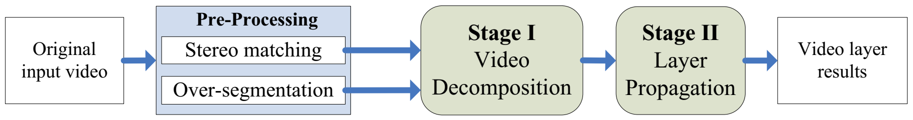

Fig. 1. The framework of decomposing static scene videos

patches into layers, and provide a User Interface(UI) for users to refine layers. In Stage II, a layer propagation method is employed to extract the layer sequence along the entire video. We will explain the details in following sections.

The remainder of the paper is organized as follows: Section 2 reviews related work; Section 3 explains the video decomposition scheme; Section 4 describes the User Interface; Section 5 gives a description of the layer propagation method; Section 6 demonstrates the experimental results; Finally, Section 7 discusses our algorithm's limitations and further works.

2 Related Work

The idea of video decompostion was introduced by Darrell et al.[1]. Wang et al.[2] present the first precise mathematical formulation for this problem. Since then, researchers have developed various models for effective motion segmentation, such as Linear Subspace[3]. Although the clustering number can be successively decided, the pixel assignments are unsatisfactory in most cases.

In parallel, the accurate figure-ground separation is being extensively studied. Y. Boykov et al.[4] formulate the problem as a global energy function in Markov Random Fields(MRF) and solve it with Graph Cuts[5]. C. Rother et al.[6] extend the graph-cut approach[4] by developing a more powerful, iterative version-“GrabCut”. These segmentation approaches are both based on uniform color information. There are some other methods using texture cue[7], or symmetry cue[8]. More recently, S. Bagon et al.[9] present an approach which unifies those cues into a framework. It is based on the concept of “Segmentation by Composition”. By developing a description for a segment and maximizing the difference in description lengths, they extract good figures.

Although the above figure-ground separation methods can be extended to videos and obtain more accurate results, it is hard to optimize boundaries of some objects, such as hairs, as they mix the background and foreground colors. Therefore, image matting[10], as a soft segmentation technique, evolves to accurately extract foreground objects. It is extensively used to recover the foreground per-pixel opacity from the background.

However, neither figure-ground separation methods nor image matting focuses on multi-layer extraction, thus their extension to video object extraction mainly lies in foreground separation, including the moving object extraction and the static foreground layer separation. The former such as [11][12] extracts objects by tracking and optimizing the boundaries, when applied to the occluded background layer, it fails as some boundaries disappear in subsequence frames; The latter such as [13] utilizes scene depth as a cue and can automatically extracted the foreground layer, but they are constrained to bi-view separation.

The most similar works to us are J. Xiao et al.[14] and G. Zhang et al.[15]. J. Xiao et al.[14] present an algorithm of motion layer extraction and matting for short video clips. They first establish a novel MRF framework to solve the motion segmentation problem, and then the Poisson matting[16] is employed to refine the foreground segmentation. However, it is impossible to separate several objects in the same background layer. G. Zhang et al.[15] present a general re-filming system, in which foreground layer matting is extracted by an interactive tool and the cut out information is propagated from key frames to the other frames automatically. They improve the optical-flow-based Bayesian video matting[17] by geometry projection of depth information. Just as the authors stated, the limitation of their system lies on depth ambiguity in extremely textureless regions (such as the clear blue sky). We overcome these problems by combining the color and depth information.

What's more, video matting approaches such as[12] are cumbersome, even in the process of initial key frame matting. While the easy-to-use GrabCut[6] often fails to construct a satisfactory result when the foreground colors are similar to the background colors. Here we developed a more robust and easy-manipulated framework for layer extraction.

3 Video Decomposition

The dense correspondence we established is initialized by a quasi-dense correspondence method[18], also called point propagation. Although lots of pixels are matched after point propagation, there are still some unmatched pixels. To obtain a total dense correspondence, we take the problem as a global cost function in Markov Random Fields, formulate a MAR-MRF model which is same to [19] and solve it by the max-flow algorithm[5].

The Pedro's method[20] is adopted for over-segmentation. Based on the depth map and over-segmentation patches, we employ a graph-based scheme to merge patches into layers. Taking each patch ν_i as a vertex, we construct an undirected weighed graph $G = \langle V, E \rangle$. The edge $(\nu_i, \nu_j) \in E$ connects two adjacent patches, its weight is defined as:

$$\omega(i,j) = \gamma_1 \omega_c(i,j) + \gamma_2 \omega_d(i,j) + \gamma_3 \omega_s(i,j) \tag{1}$$

$\omega_c(i,j)$ measures the similarity of color information, which is defined as:

$$\omega_c(i,j) = \exp(-\frac{min(\|\mu_c(i) - \mu_c(j)\|_2, T_c)}{\sigma_c}) \tag{2}$$

$\omega_d(i,j)$ measures the similarity of depth information, which is defined as:

$$\omega_d(i,j) = \exp(-\frac{min(|\mu_d(i) - \mu_d(j)|, T_d)}{\sigma_d}) \tag{3}$$

$\omega_s(i,j)$ measures the minimum size of the two regions, it is defined as:

$$\omega_s(i,j) = 1 - min(\sqrt{\frac{\mu_s(i)}{S}}, \sqrt{\frac{\mu_s(j)}{S}}) \tag{4}$$

where γ_1, γ_2, γ_3 are weighting values, they are all in the range of $[0,1]$ and $\gamma_1+\gamma_2+\gamma_3 = 1.0$; $\mu_c(\cdot)$, $\mu_d(\cdot)$ are the mean color and depth values of the region; T_c, T_d are truncation values(empirically set to 15 and 1.7); $\sigma_c = 255$, σ_d is taken as the maximum disparity value; S is the image size, $S = width * height$, and $\mu_s(\cdot)$ is the region size.

It is obvious that the edge weight formulation defined above encourages to priorly join two adjacent regions with similar color information, or/and with similar depth information, or/and with smaller size.

We use a greedy scheme to merge patches one by one. Each time, we select the edge with the maximum weight value and unite its two patches. And then the weight of the edges which connect either of the two newly united patches are recomputed. This step repeats until all patches are merged into one. We record the clustering process, so that the user can rebroadcast the process and select a satisfactory clustering result by a granular value. The maximum granular value is just the number of over-segmented patches.

4 User-Assisted Layer Refinement

Although plenty of over-segmented patches are clustered into compact components under a granularity, components of the same object may still be isolated, e.g. the two sides of an occluded wall, while further adjusting the granularity may lead to under-segmentation. Therefore, an interactive User Interface (UI) is necessary to increase the diversity of “meaningful” segmentation.

The user interactions in our system involve two stages. The first stage is to merge components into a complete layer. And the second one is to refine the layer. In the first stage, the user needs to choose a granularity with a slider first, and then click several components to acquire a complete layer. In the second stage, the user should refine the layer by clicking a button first, and then draw scribbles to refine boundaries if needed.

Here, we employ the approach of Lazy Snapping[21] to refine the layer. To apply to our task, we make some adjustments. Lazy Snapping solves the problem by formulating a global “Gibbs” energy function in patch level, while we built a similar model in pixel level. The input of Lazy Snapping is a rectangle, while our input is a layer mask with arbitrary shape. The layer mask is created by enlarging the contour of the original layer mask outwards with 3 pixels size. What’s more, we use the depth information in our model, which is unavailable in Lazy Snapping.

The “Gibbs” energy function is formulated as follows:

$$E = \sum_{i\in V} E_1(l(x_i)) + \lambda \sum_{(i,j)\in Neigh} E_2(l(x_i), l(x_j)) \tag{5}$$

It consists of a data term E_1 and a smooth term E_2, λ is the weighting value. The data term E_1 is defined as:

$$E_1(l(x)) = \begin{cases} \dfrac{d_f(x)}{d_f(x)+d_b(x)}, & l(x)=0 \\ \dfrac{d_b(x)}{d_f(x)+d_b(x)}, & l(x)=1 \end{cases} \tag{6}$$

where $l(x) = 1$ indicates x locates in the foreground layer, while $l(x) = 2$ indicates x locates in the background layer; $d_f(x) = \max_k \|I(x) - C_k^F\|_2$, $d_b(x) = \max_k \|I(x) - C_k^B\|_2$, $k = 1 \ldots 5$; $\{C_k^B\}$ and $\{C_k^F\}$ are the centroids of GMM(Gaussian Mixed Models) of the background and foreground colors, which are obtained by the K-Means method. If the user draws some scribbles, the data term of the marked pixels is taken as following:

$$\begin{cases} E_1(0) = 0 \quad E_1(1) = \infty, & if(x \in \text{“}foreground\ scribbles\text{”}) \\ E_1(0) = \infty \quad E_1(1) = 0, & if(x \in \text{“}background\ scribbles\text{”}) \end{cases} \tag{7}$$

The smooth term E_2 is defined as:

$$E_2(l(x), l(y)) = \begin{cases} \dfrac{\|I(x) - I(y)\|_2}{\varepsilon + 1} |l(x) - l(y)| & if\ ((D(x) = D(y)) \\ \dfrac{\|I(x) - I(y)\|_2}{\varepsilon + 1} (1 - |l(x) - l(y)|) & if\ ((D(x)! = D(y)) \end{cases} \tag{8}$$

where $D(\cdot)$ stands for the depth value. The above function E is a two-labels “Gibbs” function. The max-flow algorithm[5] is invoked to minimize it. Now we obtain a refined layer mask. A trimap is generated by automatically dilating the layer boundaries with 5 pixels. Then Bayesian matting[22] is applied to soft segment the layer using the trimap.

The above stages are all repeatable. The user manually clicks the slider to control the clustering granularity, clicks components to merge or separate them, adds some strokes on the layer to refine boundaries, and clicks a button to examine the matte until satisfied.

5 Spatial-Temporal Layer Propagation

To extract layers along the entire video, we require a trimap in every frame. Constructing the trimaps manually is a tedious and time-consuming work. Moreover, layer matting applied frame-by-frame produces temporally incoherency as the small errors are stochastic in each individual frame. In Section 3, we have built dense correspondences for pairwise frames. Based on the spatial-temporal depth maps, we propagate the trimaps from the key frames to the rest of the frames.

For a video clip $\hat{I} = \{I_i, i = 1 \ldots n\}$ with a depth map sequence $\hat{D} = \{D_i, i = 1 \ldots n\}$, we assume the key frames are sampled and soft segmented. We denote the trimap sequence by $\hat{T} = \{T_i, i = 1 \ldots n\}$ and successively propagate the trimaps based on the depth maps. Suppose the trimap T_i of the frame I_i is available, we first create a tri-labeling mask L_{i+1} for the frame I_{i+1}:

$$L_{i+1}(x) = \begin{cases} \texttt{`F'}, & if\ x' + D_i(x') = x\ and\ T_i(x') = \texttt{`F'}\ for\ at\ most\ one\ x' \in I_i \\ \texttt{`B'}, & if\ x' + D_i(x') = x\ and\ T_i(x') = \texttt{`B'}\ for\ at\ most\ one\ x' \in I_i \\ \texttt{`U'}, & otherwise \end{cases}$$

Then we build a foreground GMM(Gaussian Mixture Model) and a background GMM. The foreground GMM is built with pixels whose L_{i+1} = 'F' and the background GMM is built with pixels whose L_{i+1} = 'B'. The GMM components $\{C_k^B\}$ and $\{C_k^F\}$ are individually computed by the K-Means clustering, where $k = 1 \ldots 5$. By optimizing a global "Gibbs" energy function which is the same to formula (8) for pixels whose L_{i+1} = 'U', we obtain a foreground/background mask M_{i+1} for the layer. Note that the layer refinement in formula (5) is only applied for the layer mask, while it is applied for the whole image here. The data terms of definite labeled pixels, i.e, $L_{i+1}(x)$ = 'F' or 'B', are taken as formula (7). Finally, the boundaries of the mask M_{i+1} are dilated with 5 pixels size to generate the trimap. Bayesian matting is further applied to soft-segment the layer.

Generally speaking, this scheme takes advantage of spatial-temporal depth information and involves one stage of optimization. The depth map is used to preserve the intra-frame trimap coherence and the optimization process is used for inner-frame refinement.

6 Experimental Results

We apply our interactive layer decomposition algorithm to a number of video clips, involving of indoor and outdoor scenes. For the outdoor scenes, we demonstrate our result on the standard flower garden sequence. For the indoor scenes, we test several video clips from the multi-view stereo dataset[23]. Fig. 2 shows four clustering results for a still frame of the flower garden sequence. The results in video form are available in supplementary material. The granularity is defined as the number of components here. Just as Fig. 2 shows, the components always keep semantically consistent when they are merged, mainly due to our method taking account of both the color and depth information.

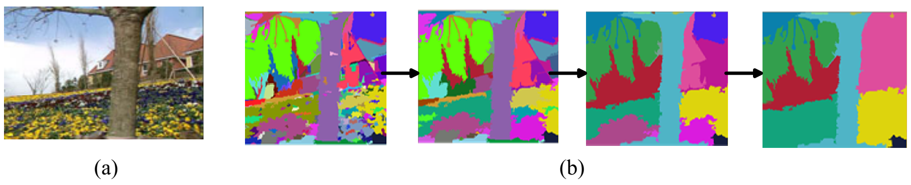

Fig. 2. The clustering results under different granularity. (a) Original frame. (b)Four clustering results, consisting of 203, 89, 16 and 10 components individually.

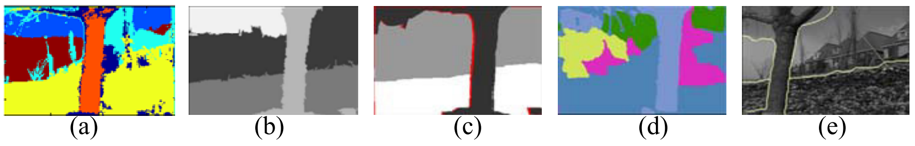

Fig. 3. Previous results for the flower garden sequence. (a) Result of S. Khan et al.[24]. (b) Result of Q. Ke et al.[3]. (c)Result of J. Xiao et al.[25]. (d) Result of R. Dupont et al.[26]. (e) Result of T. Schoenemann et al.[27].

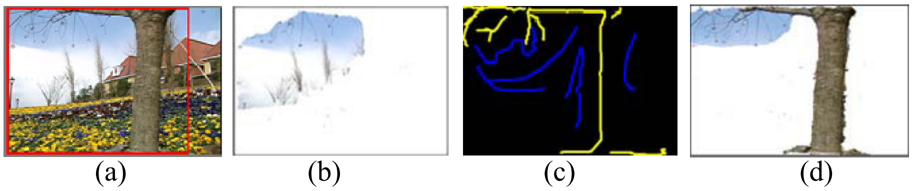

Fig. 4. Results using GrabCut. (a) The initial input is a red rectangle. (b) Result after drawing the rectangle. (c) The additional inputs are some scribbles, in which the yellow scribbles indicate the foreground and the blue scribbles indicate the background. (d) Result after applying those scribbles.

We compare the result with five motion segmentation methods[24][3][25][26] [27]. As showed in Fig. 3, methods of [3][25][27](shown in Fig. 3(b)(c)(e)) all fail to extract the red house, and they do not separate some tree trunks from the sky. [24]'s method presents too many noises in the whole image, while [26]'s method under-segments several components, e.g., the tree is extracted without the bottom root, and portions of the red house are merged into the flower bed.

Compared to their results, ours preserves the layer integrity well. As demonstrated in Fig. 2 (or the videos in supplementary material), the red house is always isolated from the sky until the number of components is lower than 10, and the thin tree trunks are always preserved until the granularity is lower than 4. Taking an edge value defined in formula(1) as a threshold, our method will automatically generate a clustering result too. The drawback of our method is that it fails to merge two sides of the same occluded layer, such as the flower bed of the flower garden sequence. This is because we only merge two adjacent components each time.

We verify the UI efficiency of our algorithm by comparing with GrabCut[6]. To extract the tree layer in the flower garden scene through GrabCut, we first draw a bounding rectangle covering the tree, and then draw scribbles to refine the foreground layer. Fig. 4(b) is the layer extraction result after we draw a rectangle (Fig. 4(a)). Fig. 4(d) is the refined layer result after we draw several scribbles (Fig. 4(c)). It is obviously cumbersome to fulfill this task through GrabCut. In contrast, our method extracts a more satisfactory layer using only several clicks(Fig. 5).

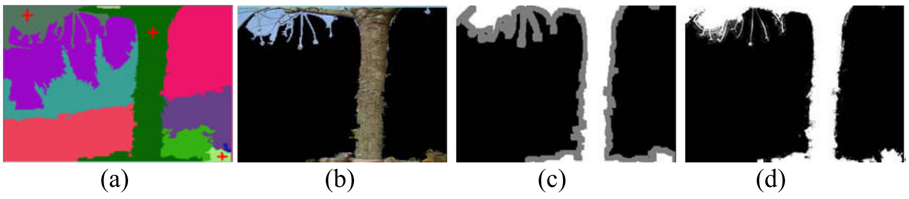

Fig. 5. Layer decomposition and matting for a flower garden frame. (a)The inputs are several clicks. (b)Refined results of the tree layer, in which some boundaries artifacts are removed after applying the layer refinement. (c)The generated trimap. (d)Mattes of the layer.

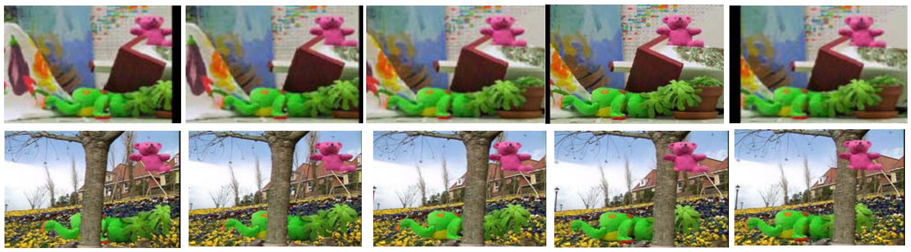

Fig. 6. Results of layer propagation on the teddy sequence. The teddy sequence consists of 9 frames, we only show the 1st, 2nd, 5th, 8th and 9th frame. The first row shows the original frames. The second row displays the composition results of two extracted toys on the flower garden clips.

Table 1. Timings for each stage of the algorithm

Sequence (352*240)	Over-Segmentation	Stereo Matching	Layer Clustering	Layer Refining	Layer Matting	Total Time
Flower Garden (20 frames)	2.67sec	73.60sec	3.32sec	12.80sec	45.40sec	137.79sec
Cone (9 frames)	1.19sec	44.58sec	0.48sec	3.52sec	14.84sec	64.61sec
Teddy (9 frames)	1.22sec	42.74sec	0.50sec	6.80sec	10.32sec	61.58sec

Fig. 6 demonstrates the layer propagation results on the teddy sequence. The two toys are extracted manually in the first frame, and the layer results propagate to the rest frames automatically. Even if there are some newly appeared regions, including the image borders and the previous occluded regions, our method can still extract the whole layer in the rest frames.

The running time is shown in Table 1, which is tested on an Intel 3.0GHz CPU with 2.0G RAM. All frames are reduced to 352*240. The key frames are sampled at every 8 frames. Other interactions all give real-time feedbacks. Obviously, the bottlenecks are the stereo matching and layer matting. It is well known that the Bayesian matting[22] in layer matting and the max-flow solution[5] in the stereo matching are both time-consuming. In the pre-processing of our framework(as shown in Fig. 1), we compare our stereo matching method with the method[19], reporting that the method[19] costs 88.39sec, 53.56sec and 54.53sec for three sequences respectively, and our method can find stronger local minima.

7 Conclusion

In this paper, we proposed an interactive algorithm for decomposing and soft-segmenting various complicated videos. The major contribution of our algorithm is the easy-manipulated framework to fulfill the task, which is built on the depth information and over-segmented patches. We also speed up the global bi-view stereo solution via point propagation. By dynamically updating the foreground and background color models in a global energy formulation, our algorithm can handle the occluded layer matting problem well. The limitation of our algorithm is that it is constrained to stabilized videos. In the future, we will incorporate the multi-view stereo into our framework for applying to various hand-hold videos. We will also exploit multi-layer matting solutions in order to simultaneously soft-segment multiple layers of videos.

Acknowledgement

This work is supported by the Industry-Academy-Research Program of Guangdong Province and the Ministry of Education(2008A090400020), National Science and Technology Supporting Program(2008BAH37B08), and the Fundamental Research Funds for the Central Universities of China.

References

1. Darrell, T., Pentland, A.: Cooperative robust estimation using layers of support. IEEE Trans. on Pattern Analysis and Machine Intelligence 17, 474–487 (1991)
2. Wang, J., Adelson, E.: Representing moving images with layers. IEEE Trans. on Image Processing Special Issue: Image Sequence Compression 3, 625–638 (1994)
3. Ke, Q., Kanade, T.: Robust subspace clustering by combined use of knnd metric and svd algorithm. In: CVPR (2004)
4. Boykov, Y., Jolly, M.: Interactive graph cuts for optimal boundary region segmentation of objects in n-d images. In: ICCV (2001)
5. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. on Pattern Analysis and Machine Intelligence 26, 1222–1239 (2001)
6. Rother, C., Kolmogorov, V., Blake, A.: "grabcut": Interactive foreground extraction using iterated graph cuts. ACM Trans. on Graphics 23, 309–314 (2004)
7. Galun, M., Sharon, E., Basri, R., Br, A.: Texture segmentation by multiscale aggregation of filter responses and shape elements. In: ICCV (2003)

8. Riklin-raviv, T., Kiryati, N., Sochen, N.: Segmentation by level sets and symmetry. In: CVPR (2006)
9. Bagon, S., Boiman, O., Irani, M.: What is a good image segment? A unified approach to segment extraction. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 30–44. Springer, Heidelberg (2008)
10. Rhemann, C., Rother, C., Wang, J., Gelautz, M., Kohli, P., Rott, P.: A perceptually motivated online benchmark for image matting. In: CVPR (2009)
11. Bai, X., Wang, J., Simons, D., Sapiro, G.: Video snapcut: Robust video object cutout using localized classifiers. In: SIGGRAPH (2009)
12. Li, Y., Sun, J., Shum, H.: Video object cut and paste. ACM Trans. on Graphics 24, 595–600 (2005)
13. Zhu, J., Liao, M., Yang, R., Pan, Z.: Joint depth and alpha matte optimization via fusion of stereo and time-of-flight sensor. In: CVPR (2009)
14. Xiao, J., Shah, M.: Accurate motion layer segmentation and matting. In: CVPR (2005)
15. Zhang, G., Dong, Z., Jia, J., Wan, L., Wong, T., Bao, H.: Refilming with depth-inferred videos. IEEE Trans. on Visualization and Computer Graphics 15, 828–840 (2009)
16. Sun, J., Jia, J., Tang, C., Shum, H.: Poisson matting. ACM Trans. on Graphics 23, 315–321 (2004)
17. Chuang, Y., Agarwala, A., Curless, B., Salesin, D., Szeliski, R.: Video matting of complex scenes. ACM Trans. on Graphics 21, 243–248 (2002)
18. Lhuillier, M., Quan, L.: Match propagation for image-based modeling and rendering. IEEE Trans. on Pattern Analysis and Machine Intelligence 24, 1140–1146 (2002)
19. Kolmogorov, V., Zabih, R.: Computing visual correspondence with occlusions via graph cuts. In: ICCV, pp. 508–515 (2001)
20. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. Int. J. of Computer Vision 70, 109–131 (2004)
21. Li, Y., Sun, J., Tang, C., Shum, H.: Lazy snapping. ACM Trans. on Graphics 23, 303–308 (2004)
22. Chuang, Y.Y., Curless, B., Salesin, D.H., Szeliski, R.: A bayesian approach to digital matting. In: CVPR (2001)
23. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. of Computer Vision 47, 7–42 (2002)
24. Khan, S., Shah, M.: Object based segmentation of video using color, motion and spatial information. In: CVPR (2001)
25. Xiao, J., Shah, M.: Motion layer extraction in the presence of occlusion using graph cut. In: CVPR (2004)
26. Dupont, R., Paragios, N., Keriven, R., Fuchs, P.: Extraction of layers of similar motion through combinatorial techniques. In: Rangarajan, A., Vemuri, B.C., Yuille, A.L. (eds.) EMMCVPR 2005. LNCS, vol. 3757, pp. 220–234. Springer, Heidelberg (2005)
27. Schoenemann, T., Cremers, D.: High resolution motion layer decomposition using dual-space graph cuts. In: CVPR (2008)