Motion Layer Extraction in the Presence of Occlusion Using Graph Cuts

Jiangjian Xiao, *Student Member*, *IEEE*, and Mubarak Shah, *Fellow*, *IEEE*

Abstract—Extracting layers from video is very important for video representation, analysis, compression, and synthesis. Assuming that a scene can be approximately described by multiple planar regions, this paper describes a robust and novel approach to automatically extract a set of affine or projective transformations induced by these regions, detect the occlusion pixels over multiple consecutive frames, and segment the scene into several motion layers. First, after determining a number of seed regions using correspondences in two frames, we expand the seed regions and reject the outliers employing the graph cuts method integrated with level set representation. Next, these initial regions are merged into several initial layers according to the motion similarity. Third, an occlusion order constraint on multiple frames is explored, which enforces that the occlusion area increases with the temporal order in a short period and effectively maintains segmentation consistency over multiple consecutive frames. Then, the correct layer segmentation is obtained by using a graph cuts algorithm and the occlusions between the overlapping layers are explicitly determined. Several experimental results are demonstrated to show that our approach is effective and robust.

Index Terms—Layer-based motion segmentation, video analysis, graph cuts, level set representation, occlusion order constraint.

1 Introduction

Layer-based motion segmentation has been investigated by computer vision researchers for a long time [2], [27], [1], [28], [18], [12], [31]. Once motion segmentation is achieved, a video sequence can be efficiently represented by different layers. The major steps of motion segmentation consist of: 1) determining the layer descriptions, which include the number of layers and the motion parameters for each layer and 2) assigning each pixel in the image sequence to the corresponding layer and identifying the occluded pixels. In general, this is a hard problem since it may not be possible to segment a scene just based on 2D parametric motion as shown in Fig. 1c. Hence, this kind of scene cannot be simply represented by planar layers and it may be necessary to compute the full disparity map using a stereo algorithm. However, in this paper, we assume that the pixels in a given video can be partitioned into several layers, and each layer can share a single 2D motion, such as affine or projective transformation, as shown in Fig. 1b.

Using this assumption, several different approaches have been proposed to solve this problem. The typical motion segmentation methods employ: optical flow with K-mean clustering [27], Expectation-Maximization (EM) framework [1], normalized graph cut [20], or linear subspace [12]. While most of the existing approaches have only focused on the layer description and representation, comparatively little work has been done to handle the occlusion problem between the overlapping layers. In contrast, the occlusion problem has been widely studied in the context of stereo algorithms [14], [15], [10]. Detection of the occluded areas is very important to improve the dense disparity map and the quality of 3D reconstruction. Similarly, in the motion segmentation area, this occlusion problem is essential to detect the discontinuities between the overlapping layers and improve the quality of the layer boundaries.

Another important problem in motion segmentation is how to determine layer description or layer clustering. Merging a set of small patches with the similar motion is a popular approach used in different methods such as K-mean clustering or linear subspace. However, the motion parameters estimated from a small patch are usually not reliable due to the overfitting problem. Therefore, before performing the layer clustering process, we need to design a key step to expand the small regions and obtain more robust motion parameters for the initial patches.

In this paper, we propose a novel approach to extract layer representations from a video sequence and explicitly determine occlusions between the overlapping layers. Our algorithm is implemented in two stages as shown in Fig. 2. In the first stage, we determine seed correspondences over a short video clip (three to five frames). Then, we gradually expand each seed region from an initial square patch into an enlarged support region of an arbitrary shape to eliminate the overfitting problem and detect the outliers. This is achieved using a graph cuts approach integrated with the level set representation. After that, we employ a two-step merging process to obtain a layer description of the video clip. In the second stage, we introduce an occlusion order constraint over multiframe segmentation, which enforces that the occlusion areas increase with a temporal order and effectively maintains segmentation consistency in the consecutive frames. After applying this constraint on the graph cuts framework, we obtain stable video segmentation in terms of layers and their 2D motion parameters. At the same time, the occluded pixels between

• *The authors are with the School of Computer Science, University of Central Florida, Orlando, FL 32816. E-mail: {jxiao, shah}@cs.ucf.edu.*

Manuscript received 10 June 2004; revised 9 Feb. 2005; accepted 10 Feb. 2005; published online 11 Aug. 2005.
Recommended for acceptance by M. Pietikainen.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0294-0604.

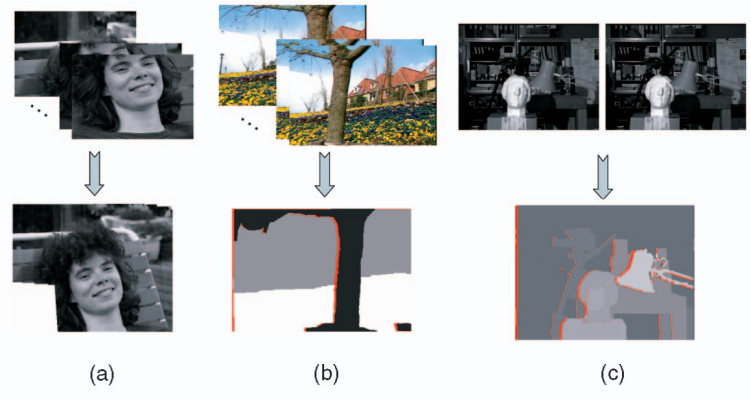

Fig. 1. Depending on the complexity of a scene, it can be represented by one layer ((a) image registration or mosaic), multiple layers ((b) motion segmentation), or disparities ((c) stereo).

overlapping layers are identified, which greatly improves the quality of the layer boundaries.

The paper is organized as follows: Section 2 reviews the previous work related to layer extraction. Section 3 addresses how to extract layer descriptions from a short video clip. Section 4 deals with the use of the occlusion order constraint, three-state pixel graph, and a multiframe graph cuts algorithm for obtaining layer segmentation in the presence of occlusion. Finally, in Section 5, we demonstrate several results obtained by our method.

2 Related Work

Automatic extraction of layers from a video sequence has broad applications, such as video compression and coding, recognition, and synthesis. In an earlier work, Wang and Adelson propose the use of optical flow to estimate the motion layers, where each layer corresponds to a smooth motion field [27]. Ayer and Sawhney combine Minimum Description Length (MDL) and Maximum-Likelihood Estimation (MLE) in Expectation-Maximization (EM) framework to estimate the number of layers and the motion model parameters for each layer [1]. Several other approaches use Maximum A-Posteriori (MAP) or MLE for estimation of model parameters assuming different constraints and motion models [22], [28], [34], [13], [24], [18], [22]. For instance, Khan and Shah [13] employ MAP framework combining multiple cues, like spatial location, color, and motion, for segmentation.

Another class of motion segmentation approaches group the pixels in a region by using linear subspace constraints. In [12], [11], Ke and Kanade expand the seed regions into the initial layers by using k-connected components. After enforcing a low-dimensional linear affine subspace constraint on multiple frames, they cluster these initial layers into several groups and assign the image pixels to these layers. Zelnik-Manor and Irani use the homography subspace for planar scenes to extract a specific layer and to register the images based on this layer [33].

In the motion segmentation area, only a few researchers have tried to formulate the occlusion problem between overlapping layers. Giaccone and Jones propose to use four-label system, "background, uncovered, covered, and foreground," to label image pixels [8]. The uncovered and covered pixels, respectively, correspond to occluded or reappeared pixels between two frames. However, they restrict their framework to two-layer (foreground and background) sequences, and their segmentation results are

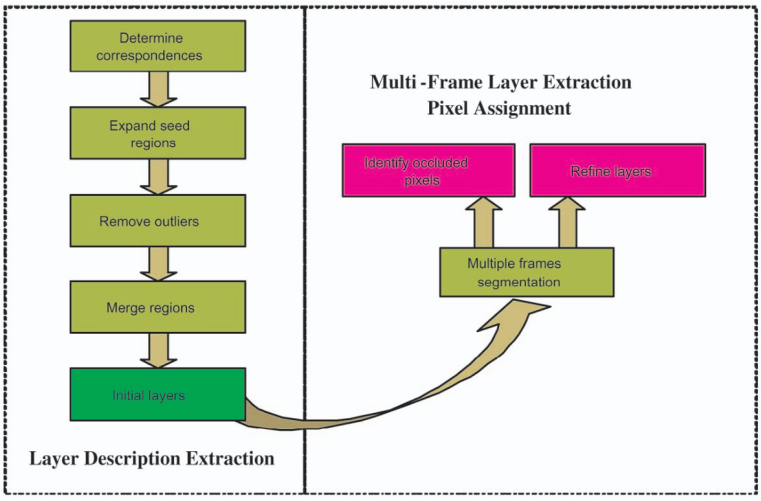

Fig. 2. The flow chart of our algorithm.

not that smooth. Based on the observation that the segmentation boundary is usually not accurate due to the occlusion, Bergen and Meyer propose to use motion estimation errors to refine the segmentation boundary but no occlusion pixel is identified in their results [3]. Compared to the previous work, our approach not only formulates the occlusion problem into a graph cuts framework, but also explicitly identifies the occluded pixels for the scene containing multiple layers.

Nevertheless, the explicit occlusion model has been widely studied in stereo area [9], [14], [15], [10]. Ishikawa and Geiger propose a set of cost models to formulate occlusions, discontinuities, and epipolar-line interaction in stereo problem, and apply a maximum-flow algorithm to solve the optimization problem using these models. Kolmogorov and Zabih introduce an occlusion cost into the graph cuts framework to solve stereo problem [14]. In [15], they extend this framework into multiple frames and introduce another visibility term to enforce the visibility constraint among these frames. Similar to Kolmogorov's occlusion model in the stereo problem, our paper integrates occlusion penalties into the graph cuts framework by using a set of three-state pixel graphs. Beyond that, we introduce an occlusion order constraint between multiple frames and this constraint is seamlessly integrated into our graph cuts framework.

Recently, graph cuts approaches [5], [14], [15] were proposed to successfully minimize energy functions for various computer vision problems, such as stereo, image segmentation, image restoration, and texture synthesis [16]. For these energy minimization problems, an approximate optimal solution can be obtained in a polynomial time by using multiway graph cuts algorithm. As an extension of the smoothness of the geodesic active contour approach, Boykov and Kolmogorov modify the n-neighbor system of the graph cuts to simulate Riemannian metrics and reduce metrication error in 2D and 3D image restoration [6]. Xu et al. present an approach to refine the active contour by iteratively using the graph cuts method [32]. Birchfield and Tomasi propose an approach to use a graph cuts framework to combine layer segmentation and stereo for the scene with slant surfaces, where the stereo disparities and discontinuities are improved by the explicit layer segmentation [4].

In the motion segmentation area, Shi and Malik use the normalized graph cut to extract layers from a video sequence [20]. However, since they group pixels based on the affinity of motion profile, a local measurement, their method ignores the global constraints and appears unstable for noisy image sequences. Wills et al. propose the use of graph cuts to extract layers between two wide baseline images [29]. After employing the RANSAC technique, they first cluster the correspondences into several initial layers, then perform the dense pixel assignment via graph cuts.

Beyond the 2D motion segmentation, 3D motion (or multibody) segmentation of multiple moving objects is another interesting topic in computer vision. In this area, the layer clustering is based on 3D geometry information, such as fundamental matrices [25] and trifocal tensors [26]. Currently, this multibody segmentation is mainly focused on sparse point segmentation and clustering, the dense motion segmentation of 3D scene is still on the research.

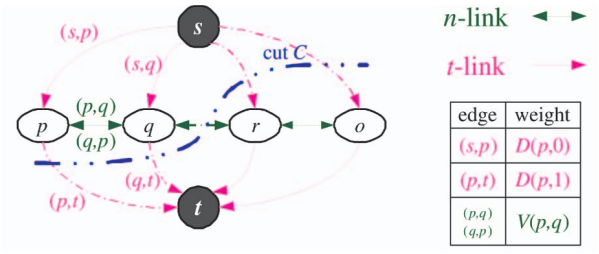

Fig. 3. An example of a graph $\mathcal{G}$ for a 1D image. Nodes p, q, r, and o correspond to the pixels in the image. After computing a minimum cut $\mathcal{C}$, the nodes are partitioned into supporting pixels p, q (source) and unsupporting pixels r, o (sink). The weights of the links are listed in the table on the right.

2.1 Graph Cuts Notation

In this paper, we use terminology and notations similar to [5]. For example, Fig. 3 shows a typical weighted graph with four nodes. This graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is defined by a set of nodes $\mathcal{V}$ (image pixels) and a set of directed edges $\mathcal{E}$ which connect these nodes as shown in Fig. 3. In this graph, there are two distinct nodes (or terminals) s and t, called the source and sink, respectively. The edges connected to the source or sink are called t-links, such as (s, p) and (p, t). The edges connected to two neighboring pixel nodes are called n-links, which are bidirectional, such as (p, q) and (q, p). The weights of these n-links in both directions may not be equal. A cut $\mathcal{C}$ is a subset of edges which separates the nodes into two parts; one part belongs to the source and the other belongs to the sink. The cost of a cut is the summation of the weights of its edges. Using a standard max-flow algorithm, the cost of the cut is globally minimized and such a cut is called a minimum cut. Given a labeling system f, each pixel in the image will be assigned one label. In this paper, we use f_p to represent the label of pixel p. $D(p, f_p)$ is the data penalty of pixel p when it is assigned a label f_p. $V(p, q)$ is a smoothness penalty function between two neighboring pixels p and q.

3 Extracting the Layer Descriptions

In our approach, the first stage is to extract the layer descriptions from the video sequence, which includes the number of layers and the motion parameters for each layer. In this stage, we first detect the robust seed correspondences over a short video clip. Then, using the shape prior of the previous seed region, the region's front is gradually propagated along the normal direction using a bipartitioning graph-cuts algorithm integrated with the level set representation. Third, we design a two-step merging process to merge the seed regions into several groups, such that each group has an independent motion field.

3.1 Determining Robust Seed Correspondences

In order to correctly extract the layer descriptions, we consider a short video clip $\mathcal{K}$ instead of only two consecutive frames. The reason is that if the motion between two consecutive frames is too small, the motion parameters between different layers are not distinct. Therefore, we use an

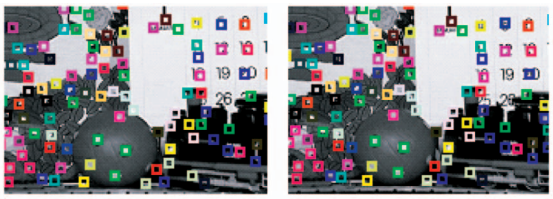

Fig. 4. The corner tracking results for frames 1 and 5 of the mobile-calendar sequence.

average pixel flow $\bar{\nu}$ of the seed correspondences as a measurement to decide the number of frames in the video clip $\mathcal{K}$:

$$\bar{\nu} = \frac{1}{N}\sum_{i=0}^{N} |\nu_i|,$$

where ν_i is the pixel flow from the current frame I_n to the first frame I_1 for correspondence i, and N is the total number of the seed correspondences. If $\bar{\nu}$ between I_n and I_1 is greater than some threshold (i.e., three pixels), the number of frames $\mathcal{K}$ is set to n.

In our approach, we detect the Harris corners in the first frame, then we use the KLT tracking algorithm [21] or our algorithm [30] to track the corners over this short period using a 17×17 pixel window. Compared to the KLT algorithm, our wide baseline matching approach can efficiently compensate for the rotation component between two corners by breaking the affine transformation into different parts. Fig. 4 shows the tracking results for frames 1 to 5 of the mobile-calendar sequence using [30], where the corresponding corners on the ball are accurate even with a large rotation.

Since the Harris corners are located at the textured areas, the affine transformations of our seed regions are more reliable. The nontextured areas are skipped, where the motion parameter estimation is unreliable.

3.2 Expanding Seed Regions

Once the seed correspondences are determined between frames I_1 and I_n,[1] we consider a patch (17×17 window) around each seed corner as an initial layer, which corresponds to a planar patch in the scene. This way, we get a number of initial layers and each layer is supported by a small patch with a corresponding affine transformation. Nevertheless, the affine motion parameters estimated using the small patches may overfit the pixels inside the region, and may not correctly represent the global motion of a larger region. Particularly, when the corner is located at the boundary between two true layers, the estimated affine transformation may overfit the pixels of one layer and may cause a serious distortion on the patch after warping.

One straightforward solution is to simply extend the region by including neighboring pixels which are consistent with the affine transformation. Such pixels can be determined by applying a threshold to the difference map (Fig. 5d) computed between the patch in the first image (Fig. 5a) and the corresponding patch warped from the second image (Figs. 5b and 5c). However, this scheme has two problems: First, the resulting expanded region may not be compact and smooth. Second, the new patch may include pixels from multiple layers and may not be consistent with a single planar patch in the scene. Fig. 5e shows one sample result obtained by using this simple scheme. The seed region is originated from the seed on the rotating ball (Fig. 5a). After expanding the boundary and bipartitioning by applying a threshold, the region is not smooth and it also includes pixels from the other layers.

In order to deal with these problems, we propose a novel approach to gradually expanding the seed region by identifying the correct supporting pixels by using the bipartitioning graph cuts method integrated with the level set representation. We introduce a smoothness energy term, which can maintain the partitions piecewisely smooth and naturally solve the first problem. Then, using the level set representation, the contour of the seed region is gradually evolved by propagating the region's front along its normal direction, which effectively blocks the pixels from the other layer and eliminates the second problem.

The process of expanding the seed region can be easily formulated into the graph cuts framework as a bipartitioning problem of a node set. In this framework, we seek the labeling function f by minimizing the energy

$$\begin{aligned} E &= E_{smooth}(f) + E_{data}(f) \\ &= \sum_{(p,q)\in\mathcal{N}} V(p,q)\cdot T(f_p \neq f_q) + \sum_{p\in\mathcal{P}} D(p,f_p), \end{aligned} \tag{1}$$

where E_{smooth} is a piecewise smoothness term, E_{data} is a data error term, $\mathcal{P}$ is the set of pixels in the image, $V(p,q)$ is a smoothness penalty function (2), $D(p,f_p)$ is a data penalty function (3), $\mathcal{N}$ is a four-neighbor system, f_p is the label of a pixel p, and $T(\cdot)$ is 1 if its argument is true and 0 otherwise. In this bipartitioning problem, the label f_p is either 0 or 1. If $f_p = 1$, the pixel p is supporting the seed region, otherwise, this pixel is not supporting the region. We define $V(p,q)$ as follows:

$$V(p,q) = \begin{cases} 3\lambda & \text{if } \max\left(|I_1(p) - I_1(q)|, \quad |I_n(p) - I_n(q)|\right) < 8, \\ \lambda & \text{otherwise.} \end{cases} \tag{2}$$

Instead of using the simple truncated quadratic function $D(p,f_p) = \min(|\delta(p) - c(f_p)|^2, const)$,[2] we use an approximate Heaviside function to compute the data penalty as follows:

$$D(p,f_p) = \begin{cases} \tan^{-1}(\delta(p)^2 - \tau) + \pi/2 & \text{if } f_p = 0, \\ \pi/2 - \tan^{-1}(\delta(p)^2 - \tau) & \text{if } f_p = 1, \end{cases} \tag{3}$$

where $\delta(p)$ is the absolute intensity difference of pixel p between the first frame and the warped version of the second frame as shown in Fig. 5d and τ is an empirical threshold, which is set to 64 for all our experiments. If there is more noise in images, we can slightly increase τ to compensate for the noise. When $\delta(p) = \tau$, $D(p,0) = D(p,1)$ and the corresponding value $D = \pi/2$, which is called a critical value $\hat{D}$. The value of λ in (2) is related to the critical value $\hat{D}$. Here, we usually set $\lambda = \hat{D}$. Since λ is used to

1. In the rest of this section, we refer to I_1 as the first frame and I_n as the second frame for the convenience of description since our layer clustering algorithm will only use these two frames.

2. $c(f_p)$ is the ideal value for layer f_p.

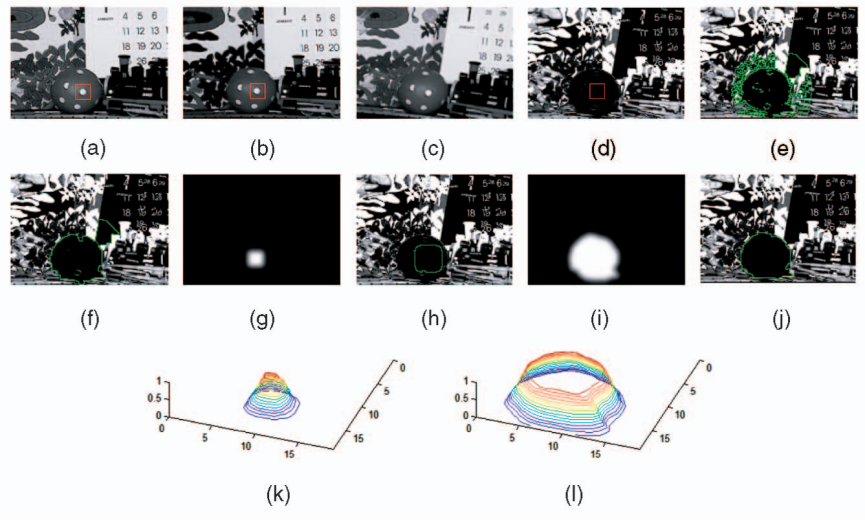

Fig. 5. A procedure for expansion of an initial 17×17 seed region to a large support region. (a) An initial seed region in the first frame. (b) The corresponding seed region in the second frame. (c) The warped version of the second frame using the estimated affine parameters. (d) The difference map between (a) and (c). (e) The result after simple expansion and partitioning. (f) The result after bipartitioning without the level set representation. (g)-(j) are the intermediate steps of bipartitioning with the level set representation. (g) and (i), respectively, are the expansions of the seed region during the first and fourth iterations using the level set representation. (h) and (j) are the results obtained after the graph cuts partitioning, where the new region can have an arbitrary compact contour. (k) and (l) are 3D visualization of the level sets of (g) and (i). *Note:* The red box is the initial seed region. The green contours are obtained after using bipartitioning algorithm.

control the smoothness penalty, a large λ will cause more piecewisely smooth segmentation.

Fig. 6 compares the Heaviside function with the truncated quadratic function. Since the Heaviside function provides a sharp change around $\sqrt{\tau}$, it effectively distinguishes the values in an ambiguity region around $\sqrt{\tau}$. Our experiments also show that using the Heaviside function the partition is more suitable for obtaining good discontinuities. In Fig. 3, we show the detailed graph for this bipartition problem. After assigning weights $D(p,0)$ to the source side t-links, $D(p,1)$ to the sink side t-links, and $V(p,q)$ to n-links in graph $\mathcal{G}$, we can compute the minimum cut $\mathcal{C}$ using the standard graph cuts algorithm and obtain a piecewise smooth partition of the supporting region.

However, the partitioning using graph cuts cannot guarantee the gradual expansion or shrinking of a region along the normal direction as shown in Fig. 5c, where some pixels not belonging to this region are also included. As a result, the computed transformation may not be representative of the real layer. Since the contour information of the initial seed region is not integrated in the function given in (1), the graph cuts algorithm cannot correctly evolve the region contour along the normal direction (Fig. 7a). In order to solve this problem, we use the contour of the seed region as a prior to compute the level set, v, of this region. Then, we apply v on the t-links at the sink side and adjust the weights of the t-links for pixels outside of the region in graph $\mathcal{G}$. Therefore, we effectively restrict the graph cuts algorithm to gradually expand the seed region (Fig. 7b). The detailed process is described as follows:

- *Step 1*: Construct a mask β of the original seed region, which has a value in [0, 1], where the inside pixels of the region are marked by 1 and the others are marked by 0. Then, compute a level set v (Fig. 5d) by simply convolving the region mask, β, with a Gaussian kernel such as: $v = G * \beta$, where G is the Gaussian kernel.

  Note: For a pixel, p, inside of the seed region, v_p, has a high constant value and the v_p outside of the region falls down along the normal direction of the contour until $v_p = 0$ (Fig. 5d). Therefore, we obtain an implicit surface for this contour evolution, which can be represented by level set [17], [19]. Here, we propose a novel approach to evolving the region contour by integrating the level set representation into graph cuts method as the next two steps.
- *Step 2*: Warp the second image using the corresponding affine transformation and compute SSD between the warped image and the first frame. Construct a graph $\mathcal{G}$ for the pixels with $v_p > 0$. Compute data

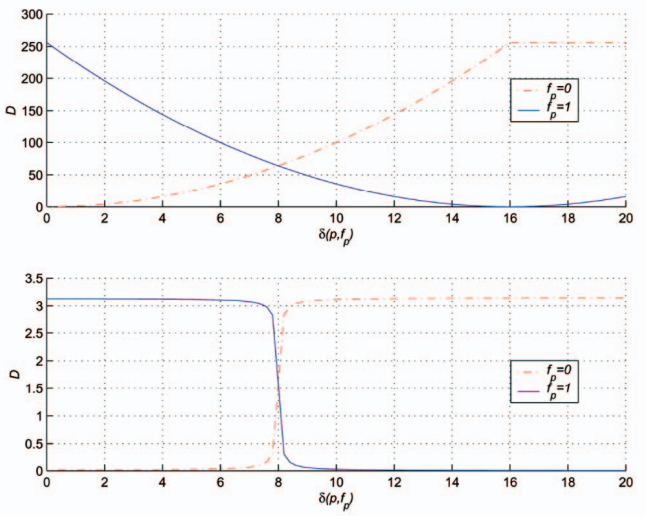

Fig. 6. Comparison between the approximate Heaviside function and the truncated quadratic function. Top: The truncated quadratic function. Bottom: Heaviside function.

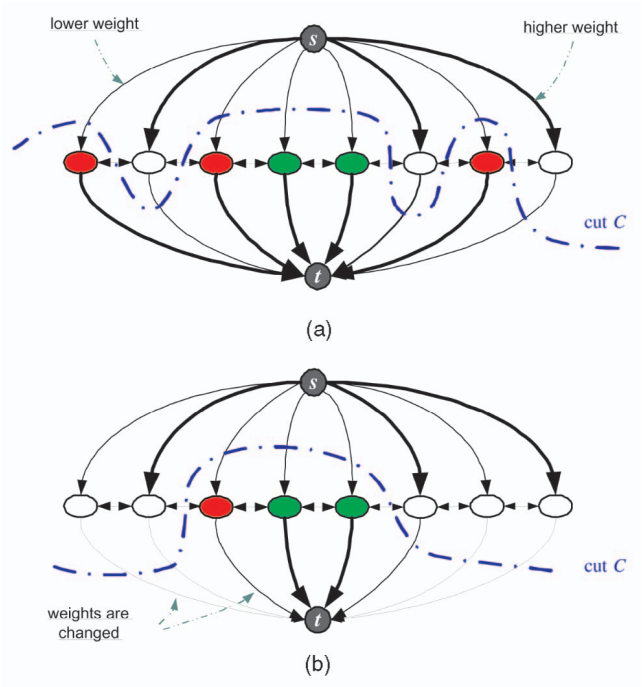

Fig. 7. Comparison between the graph cuts method (a) without the level set representation and (b) with the level set representation. The green nodes correspond to the center of the initial seed region, which belong to the sink. After bipartitioning in (a), the nodes far away from the center may be assigned the sink label, which cannot guarantee the gradual expansion or shrinking of a region along the normal direction. After integrating the level set representation, the weights of the links on the sink side are changed. Therefore, the cut is always made at the adjacent area around the original seed region (b). *Note:* We use the thickness to represent the weights of the links.

penalty D according to the computed SSD and smoothness penalty $V(p, q)$ for each link in this graph.

- *Step 3*: Multiply the weights of the sink side t-links by v_p and then compute the minimum cut $\mathcal{C}$.

Note: The weights of the pixels inside the region will almost not change, while the weight (p, t) will decrease if the pixel p is further away from the boundary as shown in Fig. 7b. As a result, the minimum cut $\mathcal{C}$ is most likely to cut the outside pixels and label them as the unsupporting pixels for this region. This way, the seed region will gradually propagate from the center (high confident area) to the exterior (low confident area) as shown in Fig. 5e.

- *Step 4*: Use the new computed region as the seed region to compute the new affine transformation by minimizing the image residue inside the region, then go to *Step 1* to do the next iteration. If the new region shrinks to a fraction of the original seed region area (e.g., 75 percent coverage threshold), we discard it as a poor initial layer.

After a few iterations of the above steps, the front of the seed region will either expand or shrink along the normal direction of the contour.

Fig. 5 and Figs. 8a and 8b show the detailed process for seed region expansion started from different seeds. Fig. 5g shows the level set representation obtained from the initial seed region (Fig. 5a). Fig. 5h and 5j are the partitioning results after the first and fourth iterations. In Fig. 8c, we show some good results for seed region expansion of the mobile-calendar and flower-garden sequences. Fig. 8d shows that we can identify the poor seed regions using the coverage threshold. Most of these poor seed regions are located at the boundary of multiple layers.

3.3 Region Merging Process

After the region expansion, each good seed region becomes an initial layer. Most of these layers may share the same affine transformation. Therefore, we use a two-step merging algorithm to merge these layers to obtain the layer descriptions.

In the first step, we only merge the overlapping layers. Given two regions R_1 and R_2, we test whether the number of

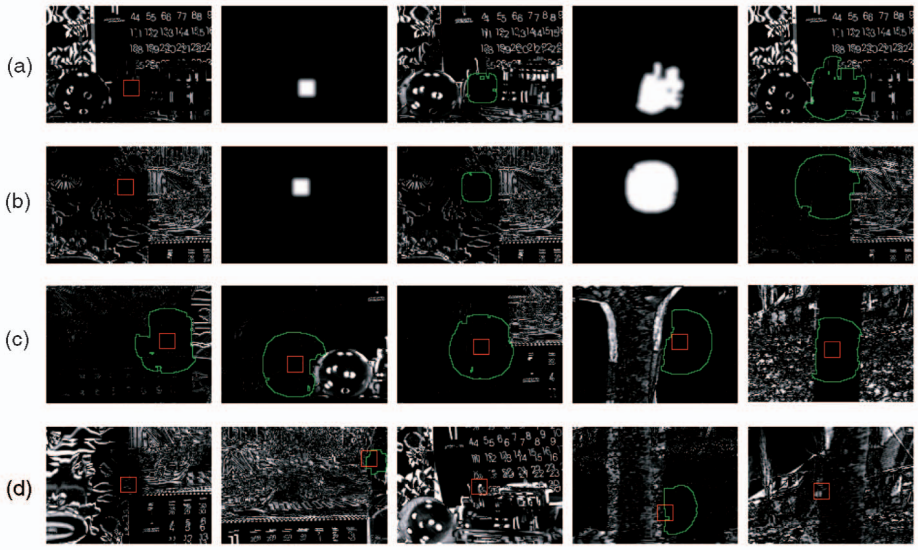

Fig. 8. Region expansion process. (a) Seed region expansion started from a seed on the train in the mobile-calendar sequence, which is similar to the process in Figs. 5d, 5e, 5f, and 5g. (b) Seed region expansion started from a seed on the background in the mobile-calendar sequence. (c) Some results of the good regions (inliers) after expansion. (d) Some results of the poor regions (outliers) after expansion, where the new region cannot cover the most of the area of the original seed. *Note:* The red box is the initial seed region. The green contours are obtained after using the bipartitioning algorithm.

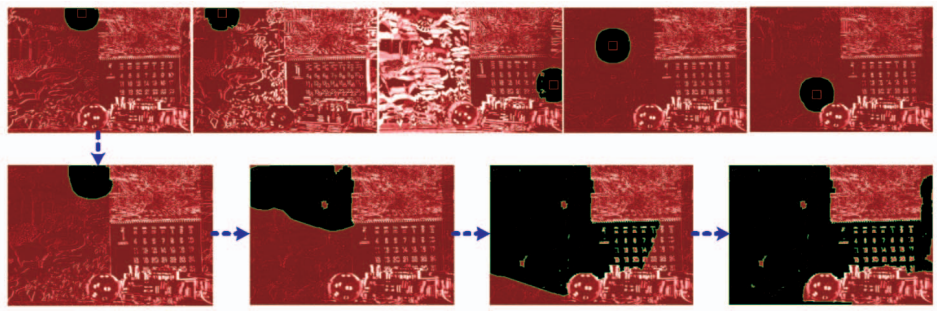

Fig. 9. Merging process. Top: Several initial seed regions sharing the same affine transformation. Bottom: The intermediate steps of the merging process when it starts from the top left seed region.

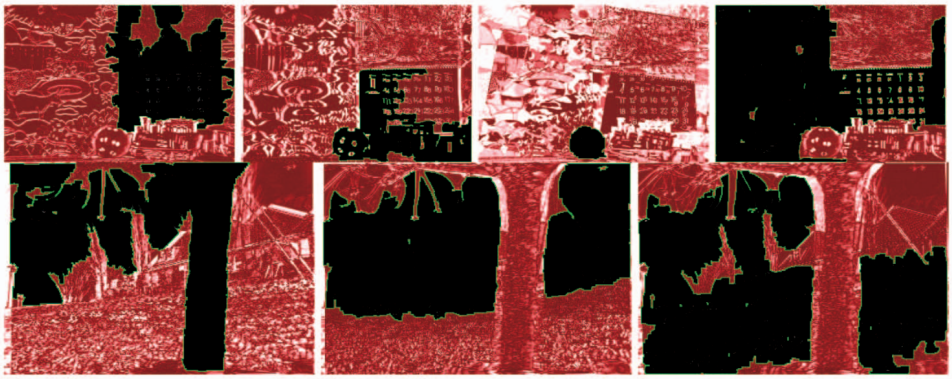

Fig. 10. Extracting layer descriptions. Top: Four layers of the mobile-calendar sequence, which correspond to the calendar, train, ball, and wall, respectively. Bottom: Three layers of the flower-garden sequence, which correspond to the tree, house, and flower-garden, respectively. The green contour is the region boundary, and nonsupporting pixels are marked by red. *Note:* The nontextured areas may belong to several layers due to their ambiguities, such as the white paper at the lower part of the calendar in the mobile-calendar sequence, and the blue sky in the flower-garden sequence.

overlapping pixels is more than half of the pixels in the smaller region. If this is true, we compute the SSD by warping the first region, R_1, using the transformation H_2 of the second region R_2. Using this SSD as the measure and employing the graph cuts algorithm, we can detect how many pixels support H_2. If the majority (say 80 percent) pixels of R_1 support R_2, we merge these two regions and recompute the motion parameters using the merged pixels. After that, we use the bipartitioning graph cuts algorithm again to prune the unsupporting pixels from the new region.

If only a few pixels of R_1 support H_2, we repeat the process by warping R_2 using the transformation H_1 of R_1. In order to achieve large merged regions, we iterate the whole process a few times (typically 3 to 4) to make sure that the merging process converges.

After the first step of merging, only a few large regions may survive. Some of the nonoverlapping regions may still share a single motion transformation. During the second step, we also merge these nonoverlapping regions. Fig. 9 shows the two-step merging process for the background layer of the mobile-calendar sequence. Fig. 10 shows the results for the mobile-calendar and flower-garden sequences.

4 Multiframe Layer Segmentation in Presence of Occlusion via Graph Cuts

After extracting layer descriptions in a short video clip using our proposed method in the previous section, the number of layers in the scene and the corresponding motion transformation of each layer are known. However, these layer descriptions can only provide rough layer representations and layer boundaries may be incorrect. Moreover, some nontextured areas may have multiple labels due to their ambiguities as shown in Fig. 10. In this section, we will solve this problem: Given the extracted layer descriptions, compute an accurate layer segmentation in presence of occlusion using *multiple* frames in a video sequence.

In this process, we explicitly identify the occluded pixels which will be assigned a new occlusion label, ζ. Then, using the occlusion order constraint over multiple frames, the consistency of the layer segmentation between frame pair $(1, 2)$ and frame pair $(1, j)$ $(j > 2)$ is maintained, and the quality of segmentation boundary is visibly improved. First, we will state the occlusion order constraint. Next, we introduce a three-state pixel graph to handle the occlusion problem between two frames in motion segmentation. Third, based on this pixel graph, we integrate this occlusion order constraint in a novel multiframe graph model which can be minimized using the graph cuts algorithm.

4.1 Occlusion Order Constraint

With the intention of computing an accurate motion layer segmentation of a video clip, let us first study the occlusion process. Fig. 11 shows the occlusion has a temporal order for a linearly moving object. It is obvious that occlusion area is increasing with the temporal order. During a short period (three to five frames), this observation is not violated if the object is not thin and not moving fast. Therefore, based on this assumption, we state the *occlusion order constraint* as follows:

- If a pixel, p_{j-1}, is assigned a label $f_{p_1} \neq \zeta$ between frames 1 and j, then pixel p_{k-1} should be assigned either f_{p_1} or ζ between frames 1 and k, where $k > j$,

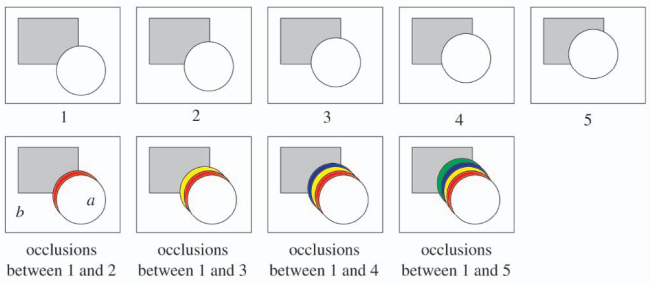

Fig. 11. The occlusion order in a short video clip containing five consecutive frames (the first image is the reference image). The top row shows the five-frame sequence, where a solid circle is moving along the left-top direction. The bottom images show the occlusions (color areas) between the first frame and other frames. It is clear that the occlusion area is increasing with time.

and pixel p_{k-1} should be assigned f_{p_1} between frames 1 and k, where $k < j$.[3]

From this occlusion order constraint, we can further clarify that during a short period, if a pixel is occluded between frames 1 and j, this pixel will also be occluded between frames 1 and $(j+1)$. According to this occlusion order constraint, only pixels at the same image coordinates in two consecutive frame pairs can influence each other, such as p_1 and p_2 in frame pair $(1,2)$ and $(1,3)$, which can effectively maintain the segmentation consistency between the consecutive frame pairs.

Now, the multiframe motion segmentation problem can be formulated as an energy minimization problem of the following energy function:

$$
\begin{aligned}
E &= \sum_{j=1}^{n-1}\left(E_{smooth_j}(f)+E_{data_j}(f)+E_{oc_j}(f)\right)+\sum_{j=1}^{n-2}E_{order_j}(f) \quad (4)\\
&= \sum_{j=1}^{n-1}\left(\sum_{i=0}^{1}\left(\sum_{(p_j,q_j)\in\mathcal{N}} V(p_{j,i},q_{j,i})\cdot T(f_{p_j}\neq f_{q_j})\right)\right.\\
&\quad + \sum_{p_j\in\mathcal{P}_j}\left(D(p_j,f_{p_j})\cdot T(f_{p_j}\neq\zeta)\right)\\
&\quad \left.+\sum_{p_j\in\mathcal{P}_j}\left(D(p_j,\zeta)\cdot T(f_{p_j}=\zeta)\right)\right)\\
&\quad +\sum_{j=1}^{n-2}\left(\sum_{p_j\in\mathcal{P}_j}\left(\infty\cdot T(f_{p_{j+1}}\neq\zeta\wedge f_{p_j}\neq f_{p_{j+1}})\right)\right)\\
&= \sum_{j=1}^{n-1}\left(\sum_{i=0}^{1}\left(\sum_{(p_j,q_j)\in\mathcal{N}} V(p_{j,i},q_{j,i})\cdot T(f_{p_j}\neq f_{q_j})\right)\right.\\
&\quad \left.+\sum_{p_j\in\mathcal{P}_j} D(p_j,f_{p_j})\right)\\
&\quad +\sum_{j=1}^{n-2}\left(\sum_{p_j\in\mathcal{P}_j}\left(\infty\cdot T(f_{p_{j+1}}\neq\zeta\wedge f_{p_j}\neq f_{p_{j+1}})\right)\right), \quad (5)
\end{aligned}
$$

where j is the frame number and n is the total number of frames. Compared to (1), there are two additional terms in this equation. The first one is $E_{oc}(f)$, which is used to impose the occlusion penalties for the occluded pixels between frames 1 and $(j+1)$. The second one is $E_{order}(f)$, which is used to impose penalties for maintaining the occlusion order constraint on each consecutive image pairs, such as frame pair $(1,2)$ and $(1,3)$. In this multiple labeling system, given a pixel p_j, the label f_{p_j} of pixel p_j is assigned one label from a label set $\mathcal{L}=\{l_1,l_2,\cdots,l_k\}\cup\{\zeta\}$, where k is the number of layers extracted using the method proposed in the previous section. $V(p_{j,i},q_{j,i})$ is the smoothness penalty term, $D(p_j,f_{p_j})\cdot T(f_{p_j}\neq\zeta)$ is the data penalty term, $D(p_j,\zeta)\cdot T(f_{p_j}=\zeta)$ is the occlusion penalty term, and $\infty\cdot T(f_{p_{j+1}}\neq\zeta\wedge f_{p_j}\neq f_{p_{j+1}})$ is occlusion order penalty term. $T(\cdot)$ is 1 if its argument is true and 0 otherwise. The detailed penalty functions are given in the following sections.

4.2 Three-State Pixel Graph

In order to minimize the above energy function, we can use α-expansion or $\alpha-\beta$ swapping techniques [5], [14] to solve the multiway cuts problem [4]. In a multiway cuts problem, given a label set $\mathcal{L}=\{l_1,l_2,\cdots,l_k\}$ as shown in Fig. 12a, each node in the graph is to be assigned one of these labels according to its data energy (t-links) and smoothness energy (n-links). However, Dahlhaus et al. have already shown that, to find a minimum cost, multiway cuts is NP-complete [7]. One feasible solution for this problem is to use multiple two-terminal subgraphs to achieve an approximate result as shown in Figs. 12b and 12c. In each step, we randomly or sequentially pick one label as the source terminal which is named as α (such as l_1 in Fig. 12b) and merge all other labels into one sink terminal. Then, the maximum flow algorithm is applied to obtain an optimal solution for this bipartition problem, which has a linear computational complexity. In each step, the α-expansion of f allows any set of pixels to change their original labels to α in one step as shown in Figs. 12b and 12c. Finally, a global approximation of the multiway cuts problem is obtained when the energy does not reduce further for any label in $\mathcal{L}$.

Traditionally, each node in the graph is only associated with one pixel, such as Fig. 3 and Fig. 12. Thus, each pixel has one individual two-state pixel graph as shown in Fig. 13a. In this pixel graph, each pixel either gets a new label α or keeps its original label f_p, which can be naturally represented by two states, [0] or [1].[4]

However, in motion segmentation application, a pixel can be assigned three labels α, f_p, or ζ at one single step of α-expansion. One node per pixel cannot represent the three states. In order to provide three states for a single pixel, we use two nodes to construct a pixel graph, which provides four possible combination of states: [0,0], [0,1], [1,0], and [1,1].

Given a pixel, p_1, in image pair (1,2), the corresponding pixel graph is constructed as shown in Figs. 13b, 13c, 13d, and 13e. There are two nodes, $p_{1,0}$ and $p_{1,1}$, and one pair of *occlusion n-links*, $(p_{1,0},p_{1,1})$ and $(p_{1,1},p_{1,0})$, associated with

3. Due to the multiple image pairs, we use p_1 to denote the pixel in the reference image for image pair $(1,2)$, pixel p_2 to denote the pixel in the reference image for image pair $(1,3)$, and pixel p_{k-1} to denote the pixel in the reference image for image pair $(1,k)$, and so on. *Note*: p_1, p_2, $\cdots$, p_{k-1} have the same location p in the reference image for every frame pair. The reference image is always frame 1.

4. After the graph cuts partition, each node is assigned either to the source [0] or to the sink [1]. There are two possible states for each node.

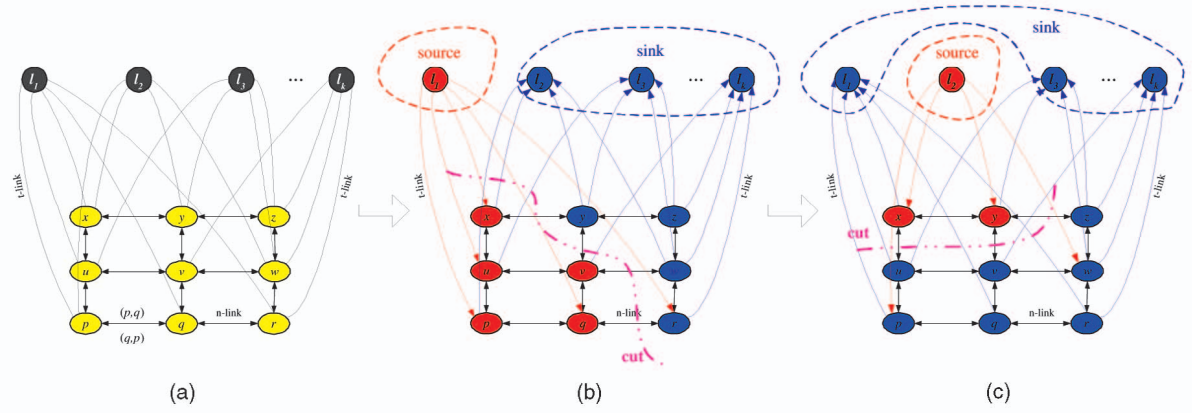

Fig. 12. A typical multiway cuts problem. (a) $\{l_1, l_2, \cdots, l_k\}$ is the label set. $p, q, r, \cdots, x, y, z$ are the nodes associated with the pixels in an image. (b) After selecting l_1 as the α label (or source) and the other label terminals are grouped as one sink, a bipartition can be achieved, where red nodes are assigned label α and the blue nodes will keep original label f_p. (c) An example for another step of α-expansion. After these two steps ((b) and (c)), p, q, u, and v will be assigned by label l_1, and x and y will be assigned by label l_2.

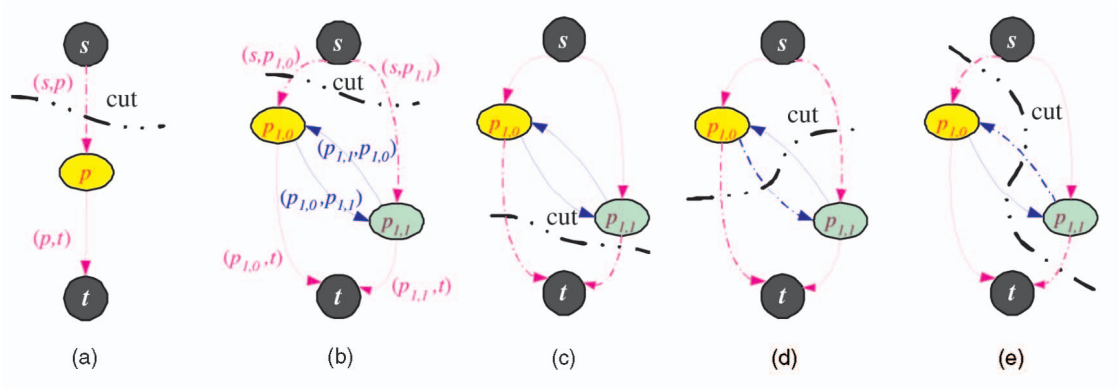

Fig. 13. Pixel graphs. (a) A two-state pixel graph has two states which are corresponding to the sink or source respectively. (b)-(e) Three-state pixel graphs which can handle the occlusion label. After one α-expansion of an independent pixel graph p, there are three possible cuts. (b) The two nodes belong to the sink and p will be assigned the new label α. (c) The two nodes belong to the source and p will keep the original label f_p. (d) One node belongs to source and the other belongs to the sink, therefore, p is occluded and assigned the label ζ. (e) Impossible case due to the link $p_{1,1}, p_{1,0} = \infty$.

this pixel. If the minimum cut, $\mathcal{C}$, cuts the link $(p_{1,0}, p_{1,1})$, the pixel p_1 is occluded. Using the link weights given in Table 1, Fig. 13 shows three cases for $\mathcal{C}$ after bipartitioning of this pixel graph. Let f_{p_1} be the original label of pixel p_1 in the reference image. The pixel will be assigned a new label $f_{p_1}^{\mathcal{C}}$ as follows:

$$f_{p_1}^{\mathcal{C}} = \begin{cases} \alpha & \text{if } (s, p_{1,0}) \in \mathcal{C}, (s, p_{1,1}) \in \mathcal{C}\ (Fig.\ 13b), \\ f_{p_1} & \text{if } (p_{1,0}, t) \in \mathcal{C}, (p_{1,1}, t) \in \mathcal{C}\ (Fig.\ 13c), \\ \zeta & \text{if } (p_{1,0}, t) \in \mathcal{C}, (s, p_{1,1}) \in \mathcal{C}, (p_{1,0}, p_{1,1}) \in \mathcal{C}\ (\text{Fig. 13c}) \end{cases} \tag{6}$$

where ζ is the occlusion label. In the first case, pixel p_1 is assigned a new label α, which is represented by the node state [0,0]. In the second case, pixel p_1 will keep its original label, which is represented by the node state [1,1]. In the occlusion case, both data penalties $D(p_1, \alpha)$ and $D(p_1, f_{p_1})$ of pixel p_1 are greater than the occlusion penalty $D(p_1, \zeta)$, which is a fixed empirical value.[5] Therefore, it is not suitable to assign either the original label, f_{p_1}, or the new

5. We usually set the value of $D(p_1, \zeta)$ slightly larger than the critical value $\hat{D}$ (see (3)). If $D(p_1, \zeta)$ increases, the number of the occlusion pixels will reduce.

TABLE 1
Weights of the Links

Edge	Weight	for
$(s, p_{j,1}), (p_{j,0}, t)$	0	$p_j \in \mathcal{P}_j$
$(p_{j,1}, t)$	$D(p_j, f_{p_j})$	$p_j \in \mathcal{P}_j \wedge f_{p_j} \neq \zeta \wedge f_{p_j} \neq \alpha$
$(p_{j,1}, t)$	∞	$p_j \in \mathcal{P}_j \wedge (f_{p_j} = \zeta \vee f_{p_j} = \alpha)$
$(s, p_{j,0})$	$D(p_j, \alpha)$	$p_j \in \mathcal{P}_j$
$(p_{j,0}, p_{j,1})$	$D(p_j, \zeta)$	$p_j \in \mathcal{P}_j$
$(p_{j,1}, p_{j,0})$	∞	$p_j \in \mathcal{P}_j$
$(p_{j,i}, q_{j,i})$ $(q_{j,i}, p_{j,i})$	$V(p_{j,i}, q_{j,i})$	$\{p_j, q_j\} \in \mathcal{N} \wedge \{p_j, q_j\} \in \mathcal{P}_j$
$(b_{j,0}^{\alpha}, b_{j,1}^{f_b})$	$D(p, \zeta)$	$b_j \in \mathcal{P}_j \wedge b_j^{\alpha} \in \mathcal{P}_j \wedge b_j^{f_b} \in \mathcal{P}_j$
$(b_{j,1}^{\alpha}, b_{j,0}^{f_b})$	∞	$b_j \in \mathcal{P}_j \wedge b_j^{\alpha} \in \mathcal{P}_j \wedge b_j^{f_b} \in \mathcal{P}_j$
$(p_{(j+1),0}, p_{j,0})$	0	$p_j \in \mathcal{P}_j \wedge p_{(j+1)} \in \mathcal{P}_{(j+1)}$
$(p_{j,0}, p_{(j+1),0})$	∞	$p_j \in \mathcal{P}_j \wedge p_{(j+1)} \in \mathcal{P}_{(j+1)}$
$(p_{(j+1),1}, p_{j,1})$	∞	$p_j \in \mathcal{P}_j \wedge p_{(j+1)} \in \mathcal{P}_{(j+1)}$
$(p_{j,1}, p_{(j+1),1})$	0	$p_j \in \mathcal{P}_j \wedge p_{(j+1)} \in \mathcal{P}_{(j+1)}$

Note: Occlusion penalty $D(p, \zeta)$ is an empirical constant. $b_{j,1}^{\alpha}$ and $b_{j,0}^{f_b}$ are the symmetric node pair to enforce the symmetric occlusion property, such as the nodes $p_{1,0}$ and $r_{1,1}$ in Fig. 16.

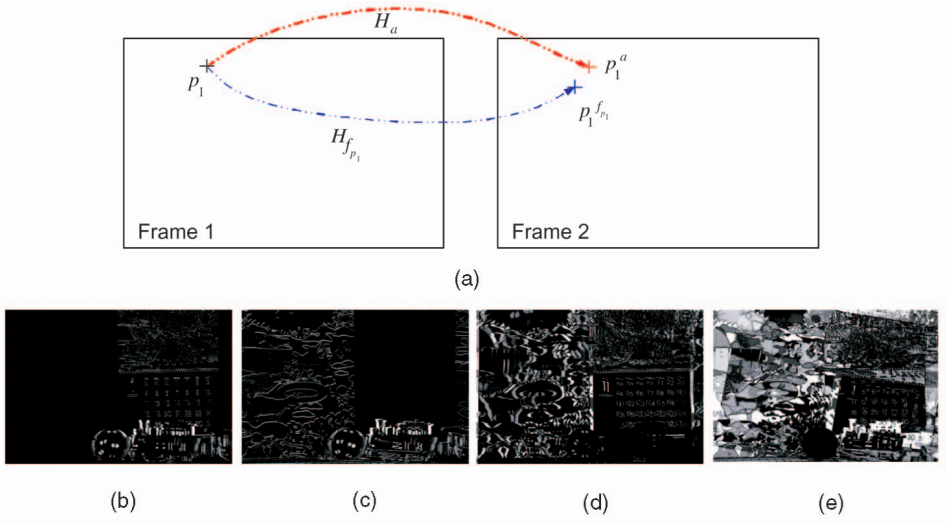

Fig. 14. Determine data penalty for pixel p_1. (a) Pixel p_1 in the first frame may be projected on different locations, $p_1^{f_{p_1}}$ and p_1^{α}, in the second frame by transformations $H_{f_{p_1}}$ and H_{α}, respectively. (b)-(e) are difference maps corresponding to background, calendar, mobile, and ball, respectively, where the image intensities are corresponding to the differences.

label, α, to this pixel. This pixel is an occluded pixel and is assigned ζ (Fig. 13c), which is represented by the node state [0,1]. Since the weight of $(p_{1,1}, p_{1,0})$ is infinite, the fourth node state [1,0] is disabled.

To compute the data penalties $D(p_1, f_{p_1})$ and $D(p_1, \alpha)$, we first need to determine the image difference $\delta(p_1)$ related to each label for pixel p_1. As shown in Fig. 14a, pixel p_1 in frame 1 may be projected to different locations, $p_1^{f_{p_1}}$ and p_1^{α}, in frame 2 by transformations $H_{f_{p_1}}$ and H_{α}, respectively. Therefore, $\delta(p_1)$ can be obtained as follows:

$$\delta(p_1, f_{p_1}) = |I_1(p_1) - I_2(p_1^{f_{p_1}})|, \tag{7}$$

$$\delta(p_1, \alpha) = |I_1(p_1) - I_2(p_1^{\alpha})|, \tag{8}$$

where $I_1(p_1)$ is the intensity value of pixel p_1 in the first frame and $I_2(p^{f_{p_1}})$ is the intensity value of pixel $p_1^{f_{p_1}}$ in the second frame. p_1 and $p_1^{f_{p_1}}$ are corresponding points under transformation $H_{f_{p_1}}$. The notations related to α have the similar meanings. Then, the data penalties are computed as follows:

$$D(p_1, f_{p_1}) = \tan^{-1}(\delta(p_1, f_{p_1})^2 - \tau) + \pi/2, \tag{9}$$

$$D(p_1, \alpha) = \tan^{-1}(\delta(p_1, \alpha)^2 - \tau) + \pi/2, \tag{10}$$

where τ is set a fixed value (3). To improve computational performance, we precompute the difference map $\delta(p_j)$ for each possible label as shown in Figs. 14b, 14c, 14d, and 14e, where four difference maps are computed by using frame 1 and 5 of the mobile-calendar sequence.

4.3 Multiframe Motion Segmentation via Graph Cuts

Using the pixel graphs, the multiframe motion segmentation graph can be constructed as shown in Fig. 15. To illustrate the occlusion order constraint in multiframe segmentation, we stack four pairs of image nodes together in this graph. Note that each image pair involves the first frame (the reference frame) and one of the other frames, which is consistent with Fig. 11.

In Fig. 15, each image pair is separated by the red dotted lines. In each image pair $(1, j+1)$, $j > 1$, only the pixels in the reference image (frame 1) will be used to construct pixel graphs. For each pixel p_j,[6] a pixel graph is created with two nodes $p_{j,0}$ and $p_{j,1}$. In each image pair $(1, j+1)$, p_j belongs to a pixel set, $\mathcal{P}_j$, which is the set of pixels in the reference image for image pair $(1, j+1)$. In Fig. 15, there are four pixels subsets $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, and $\mathcal{P}_{\triangle}$ corresponding to each image pair.

According to the occlusion order constraint, a set of *order n-links* (blue edges), such as $(p_{3,0}, p_{2,0})$ and $(p_{2,0}, p_{3,0})$, are added in the graph $\mathcal{G}$ to interact with the pixel graph at the same image coordinates. In order to simplify graph $\mathcal{G}$, we only show two nodes from one particular pixel p_j for each image pair to illustrate these *order n-links*. The detailed subgraph $\mathcal{G}_{1,2}$ for the first image pair is redrawn in Fig. 16.

Before we describe how to minimize the energy E for the whole graph $\mathcal{G}$, we first discuss the interaction of the nodes in subgraph $\mathcal{G}_{1,2}$ and then discuss how to assign the weights to these links. To reduce the complexity, we show only three pixels p_1, q_1, and r_1 of frame 1 in graph $\mathcal{G}_{1,2}$, where each pixel has one pixel graph.

In graph $\mathcal{G}_{1,2}$ (Fig. 16), the smoothness energy function, $E_{smooth}(f)$, is implemented by the *smoothness n-links*, which connect each pair of neighboring pixel graphs such as $(q_{1,1}, p_{1,1})$ and $(p_{1,1}, q_{1,1})$. In order to compute the smoothness penalty term $V(p_{1,i}, q_{1,i})$ of a link $(p_{1,i}, q_{1,i})$, we warp the second image I_2 to obtain the warped image $I_2^{H_{f_i}^{-1}}$ by applying the inverse motion transformation $H_{f_i}^{-1}$, corresponding to label f_i, for each label in the layer descriptions. Here,

$$f_i = \begin{cases} \alpha & \text{if } i = 0 \\ f_{p_j} & \text{if } i = 1. \end{cases} \tag{11}$$

6. p_j has the same location p in frame 1 for every frame pair $(1, j+1)$.

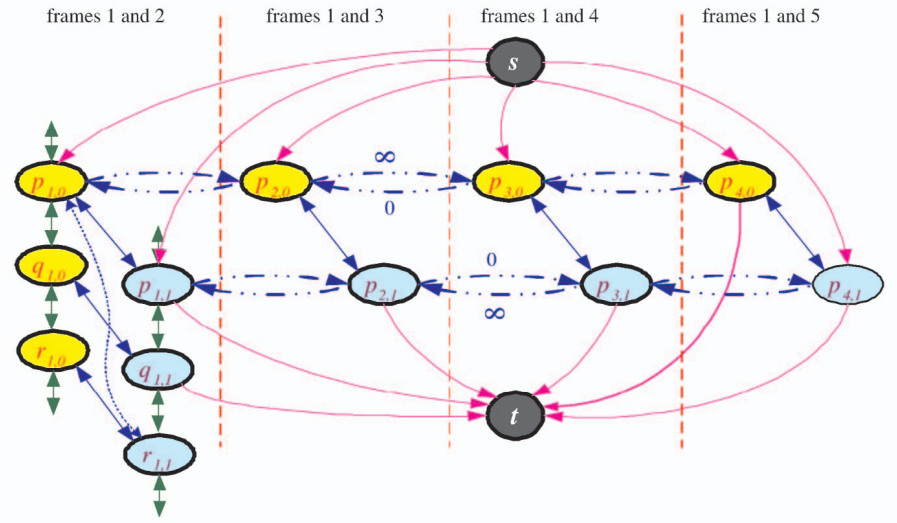

Fig. 15. This graph is constructed using five consecutive frames, which have four image pairs related to the reference image. The red lines separate each pair of images into one block. The blue n-links are introduced to maintain the occlusion order constraint. *Note: Only some of the nodes and links are shown here.*

Therefore, the smoothness penalty term $V(p_{j,i}, q_{j,i})$ can be computed as

$$V(p_{j,i}, q_{j,i}) = \begin{cases} 4\lambda & \text{if } \max(|I_1(p) - I_1(q)|, \quad |I^{H_{f_i}^{-1}}_{(j+1)}(p) - I^{H_{f_i}^{-1}}_{(j+1)}(q)|) < 4, \\ 2\lambda & \text{if } 4 \le \max(|I_1(p) - I_1(q)|, |I^{H_{f_i}^{-1}}_{(j+1)}(p) - I^{H_{f_i}^{-1}}_{(j+1)}(q)|) < 8, \\ \lambda & \text{otherwise,} \end{cases} \tag{12}$$

where I_1 is the first frame, $I^{H_{f_i}^{-1}}_{(j+1)}$ is the warped version of $I_{(j+1)}$ obtained by applying the inverse transformation $H_{f_i}^{-1}$, and λ is an empirical constant (2).

In order to model the symmetric properties of the occlusion, a set of new symmetric occlusion n-links are added to connect the related nodes. Given a pixel b in the second frame, two pixels b^{α} and b^{f_b} in the first frame can be mapped to the same pixel b by transformations H_{α} and H_{f_b}, where f_b is the current label of b in the second frame. After applying the inverse transformations H_{α}^{-1} and $H_{f_b}^{-1}$ to b, we can determine both b^{α} and b^{f_b}. Then, a pair of n-links are added to connect these two nodes, such as the blue dotted links $(r_{1,1}, p_{1,0})$ and $(p_{1,0}, r_{1,1})$ shown in Fig. 16. With the help of these symmetric occlusion n-links, the occlusion penalties from frame 2 to frame 1 are also specified.

4.4 Energy Minimization with Occlusion Order Constraints

After assigning weights 0, ∞, ∞, and 0 to *order n-links* $(p_{(i+1),0}, p_{i,0})$, $(p_{i,0}, p_{(i+1),0})$, $(p_{(i+1),1}, p_{i,1})$, and $(p_{i,1}, p_{(i+1),1})$, respectively, the occlusion order constraint is fully satisfied.

Using a three state pixel graph system, if the label of pixel $p_j \neq \zeta$, the energy of the pixel graph is depended on the t-link weights (or data penalty), such as $(s, p_{j,0})$ or $(p_{j,0}, t)$. If the label of pixel $p_j = \zeta$, the energy of the pixel graph is depended on the occlusion n-link weights (or occlusion penalty), $(p_{j,0}, p_{j,0})$. Therefore, we can merge the data penalty term and occlusion penalty term together to obtain a new data and occlusion penalty term $D(p_j, f_{p_j})$. From the occlusion order penalty term, we can see that if the label of pixel p_{j+1} is not ζ, pixels p_{j+1} and p_j should have the same label, otherwise, an infinity penalty will be imposed on the corresponding occlusion order n-links. This is consistent with our occlusion order constraint discussed in Section 4.1.

We can easily verify the occlusion constraint by assuming the minimum cut position. For example, after assuming that the pixel p_2 keeps the original label $f_{p_2} \neq \zeta$ between frames 1 and 3 during α-expansion (Fig. 17a), the nodes $p_{2,0}$ and $p_{2,1}$ belong to the source and the cut will cross the t-links $(p_{2,0}, t)$ and $(p_{2,1}, t)$. Since the weight of n-link $(p_{2,1}, p_{1,1})$ is ∞, node $p_{1,1}$ cannot belong to the sink. Thus, $p_{1,0}$ and $p_{1,1}$ both belong to the source and the pixel p_1 between frame pair $(1, 2)$ will also keep the original label f_{p_1}. Similarly, node $p_{3,0}$ will belong to the source due to $(p_{2,0}, p_{3,0}) = \infty$. As a result, there are three possible cases as shown in Figs. 17b, 17c, and 17d.

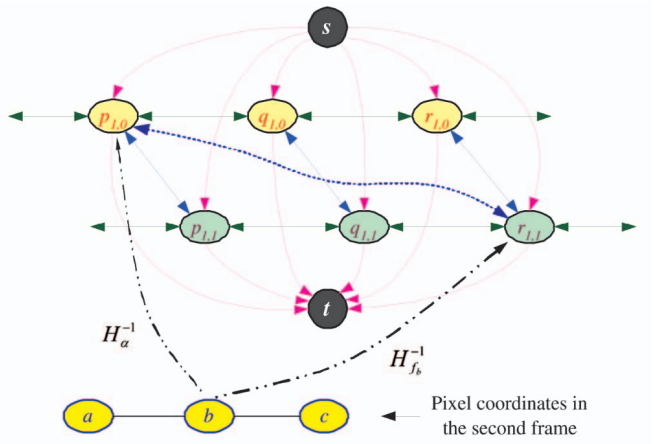

Fig. 16. A graph $\mathcal{G}_{1,2}$, where three basic pixel graphs are shown corresponding to pixels p, q, and r, respectively. The n-links between neighboring pixels are to enforce the smoothness penalties, such as $(p_{1,1}, q_{1,1})$ and $(q_{1,1}, p_{1,1})$. After applying the the inverse transformations H_{α}^{-1} and $H_{f_b}^{-1}$ to pixel b in the second frame such that $b^{\alpha} = p_{1,0}$ and $b^{f_b} = r_{1,1}$, a pair of n-links is introduced to enforce the symmetric property of the occlusion, such as $(p_{1,0}, r_{1,1})$ and $(r_{1,1}, p_{1,0})$.

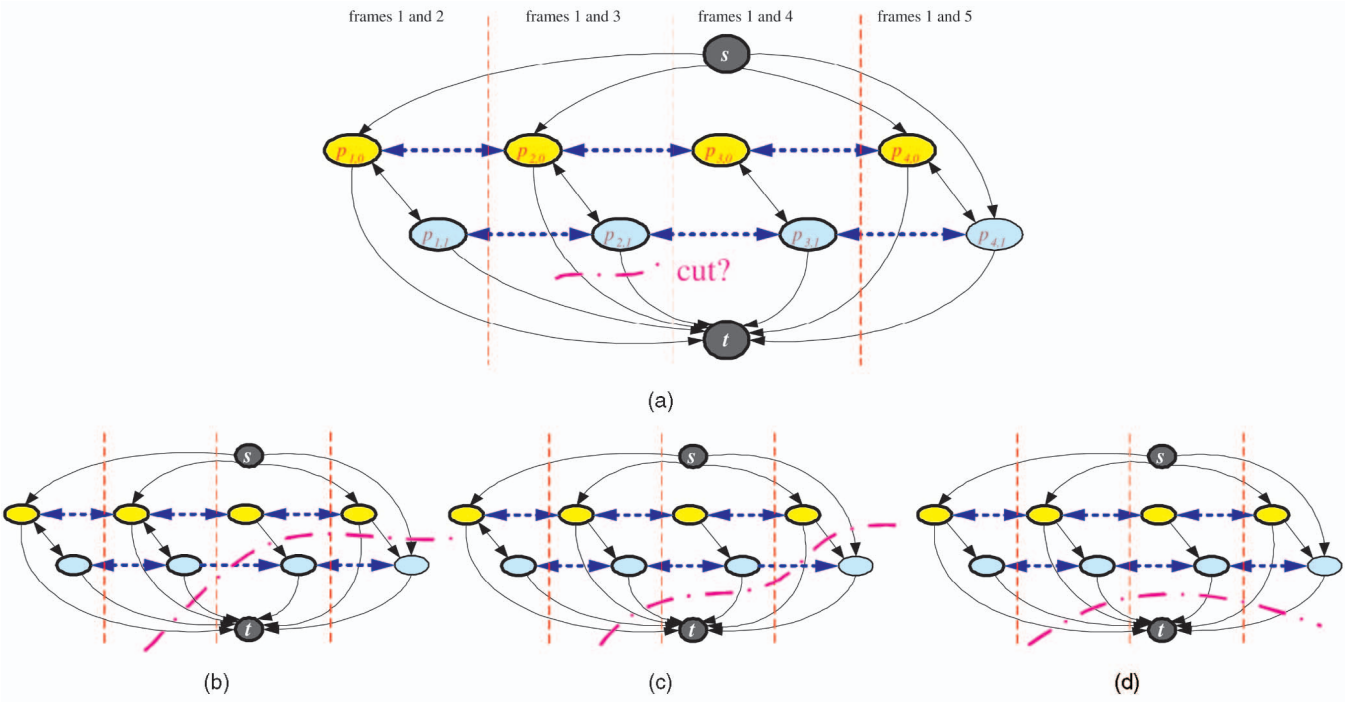

Fig. 17. Verification for the occlusion order constraint. Three possible cases (b)-(d) after assuming a cut across $(p_{2,0}, t)$ and $(p_{2,1}, t)$ in (a). All of the sink side t-links before the $(p_{2,0}, t)$ and $(p_{2,1}, t)$ are definitely crossed by the minimal cut $\mathcal{C}$, while the sink side t-links after the $(p_{2,0}, t)$ and $(p_{2,1}, t)$ may or may not be cut. *Note*: in this case, none of the source side t-links can be crossed.

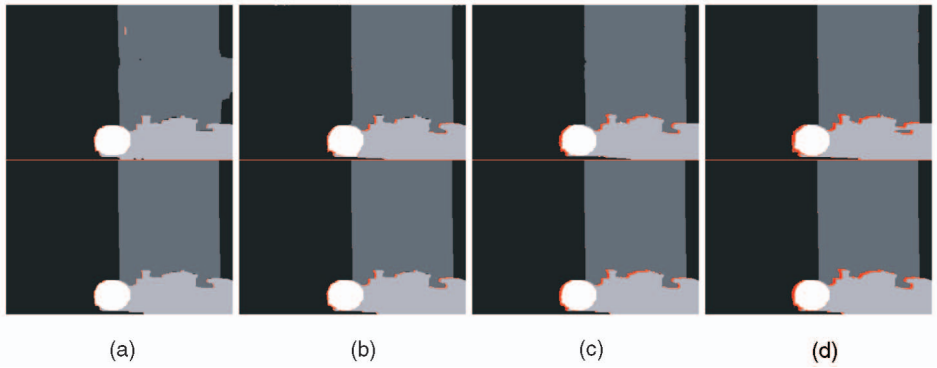

Fig. 18. Segmentation results for frame 1 of the mobile-calendar sequence. Top: The segmentation results obtained by using only two frames. Bottom: The segmentation results obtained by using five frames (1-5) with the occlusion order constraint. (a) Segmentation results between frames 1 and 2. (b) Segmentation results between frames 1 and 3. (c) Segmentation results between frames 1 and 4. (d) Segmentation results between frames 1 and 5. The red pixels in the segmented images are the occluded pixels, which are consistently increasing with time in the bottom row. After using the occlusion order constraint on these frames, the segmentation results on the reference image (frame 1) are much more consistent than those without the occlusion order constraint (top).

In order to obtain segmentation results for each pair of neighboring fames in a sequence, we always use three to five consecutive frames to preform segmentation since our assumption is more valid and feasible in a short period (we will discuss this further in the next section). For example, after obtaining the initial layer descriptions between frames 1 and 5 from Section 3, we can easily estimate the motion parameters between frame pairs $(1, 2)$, $(1, 3)$, and $(1, 4)$ for each real layer label. Based on these motion parameters, five consecutive frames 1, 2, 3, 4, and 5 are used in multiframe graph cuts algorithm to compute the segmentation, where we can simultaneously achieve the segmentation for each frame pair $(1, j), 2 \leq j \leq 5$. In order to perform segmentation for frames 2 and 3, we first estimate the initial layer descriptions (support regions and corresponding motion parameters of each layer) from the previous segmentation results between $(1, 2)$. Then, based on the initial layer descriptions, another set of five consecutive frames 2, 3, 4, 5, and 6 will be used to refine the segmentation between $(2, 3)$ by employing the multiframe graph cuts algorithm. Finally, the segmentation of the whole sequence can be achieved by repeating the above process.

Fig. 18 compares the segmentation results obtained using five frames with those obtained using only two frames. Due to the use of multiple frames with the occlusion order constraint, the artifacts are removed and the segmentation results are more consistent, as shown in Figs. 18b, 18c, and 18d. Moreover, it is obvious that the occluded areas between the overlapping layers increase over time.

Fig. 19. The segmentation results for the mobile-calendar sequence. The red pixels are occluded pixels. Please visit our Web site [35] for the video sequences.

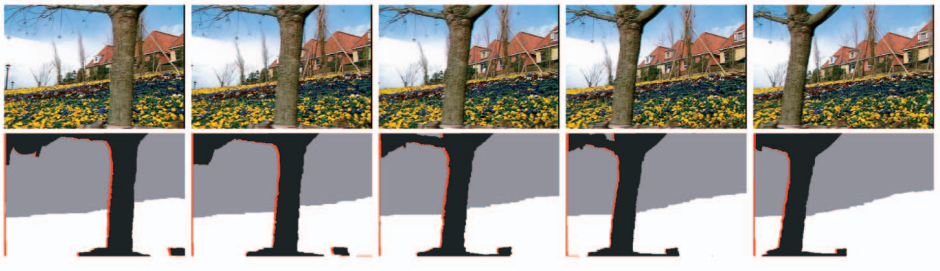

Fig. 20. The segmentation results for the flower-garden sequence. The red pixels are occluded pixels. Please visit our Web site [35] for the video sequences.

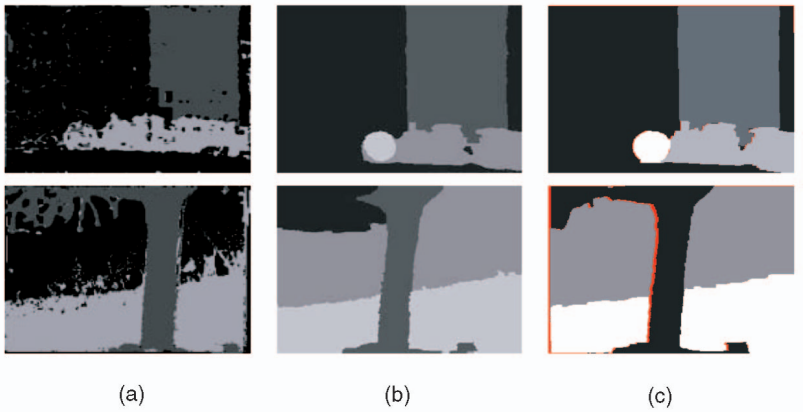

Fig. 21. Comparison on mobile-calendar and flower-garden sequences with two previous methods. (a) Results of Ayer and Sawhney [1]. (b) Results of Ke and Kanade [12]. (c) Our results. In our results, we cannot only obtain the accurate layer segmentation, but also explicitly detect the occluded pixels (red pixels). *Note*: (a) and (b) are reproduced from papers [1], [12], respectively.

5 Experiments

In this section, we demonstrate our results on two standard motion sequences, mobile-calendar (Fig. 19) and flower-garden (Fig. 20), and two more sequences.

Figs. 19 and 20 show the segmentation results for the mobile-calendar and flower-garden sequences. We used five frames to extract the layers for the mobile-calendar sequence and used three frames to extract the layers for the flower-garden sequence. We also compared our results with other methods [12], [11], [27], [1] for these two standard sequences. Since the ground truth for these sequences is not available, we have to limit our analysis to qualitative comparisons. Fig. 21 shows the comparison on these two standard motion sequences. Compared to the previous approaches, our method not only explicitly determines the occluded pixels, but also provides more precise and finer boundaries between the overlapping layers. Fig. 22 shows the plot of energy versus number of iteration for these two standard sequences. After a few iterations (which usually correspond to the number of layers), the energy converges to an approximately optimized solution.

We also applied our method to our own sequence, car-map, with a large occlusion as shown in Fig. 23, where the car is moving behind the map and the scale of the car is apparently changed. The sequence was taken by a handheld moving video camera. During some frames, most parts of the car are occluded by the map. Once the car moves behind the map, it is very difficult to compute the correct motion parameters for the car layer based on a small region of the car due to the overfitting problem. Therefore, we use a common tracking technique to predict the motion parameters based on the previous frames. If the region shrinks by some amount (say 20 percent) and the predicted motion parameters are much different than the new estimated parameters, we maintain the predicted parameters to perform the segmentation. The results are shown in Fig. 23.

Fig. 24 shows a scene, box-card, with multiple layers, which was also taken by a handheld moving video camera.

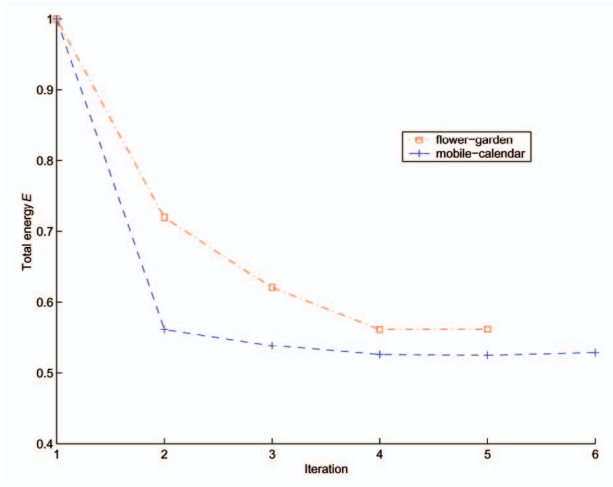

Fig. 22. Plot of energy versus number of iteration for two standard sequences. Initially, we assign the occlusion label to all pixels in our graph. In each iteration or α-expansion step, one layer label is selected as the α label. After several iterations, an approximately global solution is found in both cases.

In order to align the imagery, we have to segment this video clip into four planar layers such that each layer shares one projective transformation. Due to the apparently projective distortion and large ego-rotation of the camera, the affine transformation cannot fully capture the transitions between the consecutive frames. Instead of the affine transformation used for other sequences, we used the Levenberg-Marquardt method [23] to compute the projective transformation for each layer. It is clear that we have obtained good results for this sequence as shown in Fig. 24.

Since our occlusion order constraint is based on the assumption that the moving object is not thin and not moving fast, the occlusion may not have a temporal order if the object is thin or moving back and forth randomly. In this case, the segmentation around this object may not be accurate, e.g., the tree branch in the flower-garden sequence (Fig. 20). Nevertheless, in a real video sequence, the object motion usually is coherent and it is very rare to observe a randomly moving object. In a short period (three to five frames), a linear approximation of the object movement is a good choice for most of vision applications, such as object tracking. Consequently, as we expected, the experiment results using this constraint are better than those without this constraint (Fig. 18).

In all of our experiments, once the layer descriptions are extracted, the average computational time for one frame (352×240) segmentation is less than 30 seconds on Pentium IV 2.0G. However, in the first stage of the layer description extraction, our approach requires a relative long time, which is about 8-10 minutes including all steps. The most time-consuming steps are the seed regions expansion and

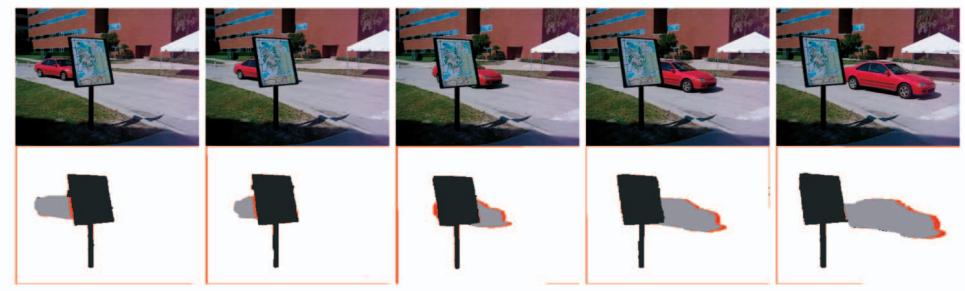

Fig. 23. Segmentation results for the car-map sequence. Top: Several frames from the sequence. Bottom: The segmentation results, where the layers are accurately extracted even though the most parts of the moving car are occluded in some frames. There are three layers corresponding to map, car, and background building, respectively.

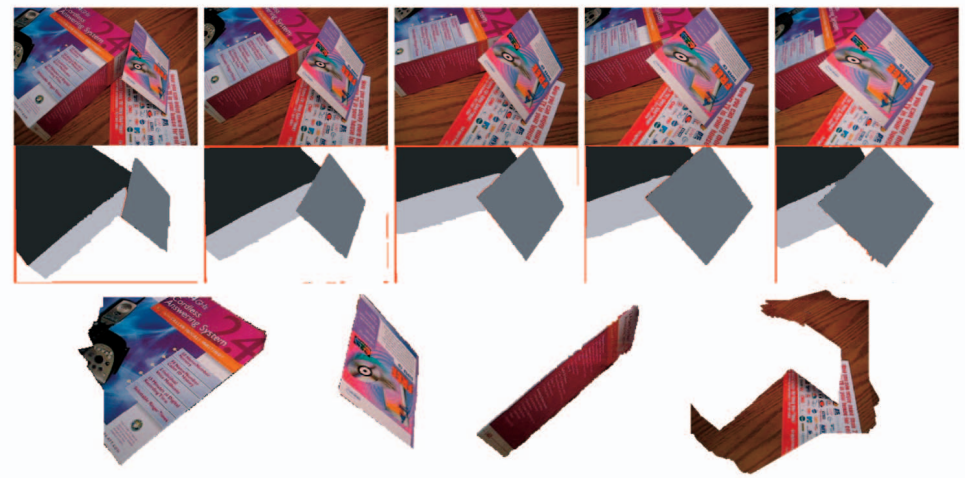

Fig. 24. Segmentation results for the box-card sequence using projective transformation. Top: Several frames from the sequence. Middle: The segmentation results using homography transformation. There are four layers corresponding to two sides of the box, card, and desk, respectively. Bottom: The mosaics of four layers after registering all frames on the first frame.

region merging. The complexity order of these steps can be approximately described as

$$O_e = n_s m_e A, \tag{13}$$

$$O_m = n_g(n_g - 1)m_m A_m, \tag{14}$$

where O_e is the complexity order of the seed regions expansion step, n_s is the number of the seed correspondences, m_e is the iteration number ($m_e = 4$ in our experiments), A is the area of the seed region, O_m is the complexity order of the region merging step, n_r is the number of the good seed correspondences, m_m is the iteration number ($m_m = 4$ in our experiments), and A_m is the area of the merged seed region. Here, O_m is the complexity order for the worst case such that the all regions are not merged. In practice, the computation complexity of the region merging step is dramatically reduced when the regions are quickly merged together into three to four clusters. In order to speed up the algorithm, reducing the node number for the graph construction is the most efficient approach. For example, we can restrict the graph construction only for the boundary pixels of the seed regions. *Note: All of our results are also available on our Web site [35].*

6 Conclusions

In this paper, we presented an effective method to extract robust layer descriptions and to perform an accurate layer segmentation for the image sequences containing 2D motion (affine or homography). Our contributions consist of: 1) Initial layer descriptions by integrating the level set representation into the graph cuts method to obtain gradually expanding seed regions. 2) Using the occlusion order constraints, we successfully combine multiple frames to compute accurate and consistent layer segmentation and explicitly detect the occluded pixels, which has not been done before.

In the future, we will investigate the relationship between the level set and graph cuts methods and unify these two approaches into one framework for different applications.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive critiques. They would also like to thank Khurram Shafique, Yunjun Zhang, Lisa Spencer, and Yaser Ajmal for helpful discussions. This material is based upon work funded in part by the US Government. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US Government.

References

[1] S. Ayer and H. Sawhney, "Layered Representation of Motion Video Using Robust Maximum-Likelihood Estimation of Mixture Models and MDL Encoding," *Proc. Int'l Conf. Computer Vision,* 1995.

[2] G. Adiv, "Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 7, no. 4, pp. 384-401, Apr. 1985.

[3] L. Bergen and F. Meyer, "A Novel Approach to Depth Ordering in Monocular Image Sequences," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2000.

[4] S. Birchfield and C. Tomasi, "Multiway Cut for Stereo and Motion with Slanted Surfaces," *Proc. Int'l Conf. Computer Vision,* 1999.

[5] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 11, pp. 1222-1239, Nov. 2001.

[6] Y. Boykov and V. Kolmogorov, "Computing Geodesics and Minimal Surfaces via Graph Cuts," *Proc. Int'l Conf. Computer Vision,* 2003.

[7] E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis, "The Complexity of Multiway Cuts," *Proc. ACM Symp. Theory of Computing,* pp. 241-251, 1992.

[8] P. Giaccone and G. Jones, "Segmentation of Global Motion Using Temporal Probabilistic Classification," *Proc. British Machine Vision Conf.,* 1998.

[9] H. Ishikawa and D. Geiger, "Occlusions, Discontinuities, and Epipolar Lines in Stereo," *Proc. European Conf. Computer Vision,* 1998.

[10] S. Kang, R. Szeliski, and J. Chai, "Handing Occlusions in Dense Multi-View Stereo," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2001.

[11] Q. Ke and T. Kanade, "A Subspace Approach to Layer Extraction," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2001.

[12] Q. Ke and T. Kanade, "A Robust Subspace Approach to Layer Extraction," *Proc. IEEE Workshop Motion and Video Computing,* 2002.

[13] S. Khan and M. Shah, "Object Based Segmentation of Video Using Color, Motion and Spatial," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2001.

[14] V. Kolmogorov and R. Zabih, "Visual Correspondence with Occlusions Using Graph Cuts," *Proc. Int'l Conf. Computer Vision,* 2001.

[15] V. Kolmogorov and R. Zabih, "Multi-Camera Scene Reconstruction via Graph Cut," *Proc. Eourpean Conf. Computer Vision,* 2002.

[16] V. Kwatra, I. Essa, A. Schodl, G. Turk, and A. Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," *Proc. ACM SIGGRAPH,* 2003.

[17] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces.* Springer-Verlag, 2003.

[18] I. Patras, E. Hendirks, and R. Lagendijk, "Video Segmentation by MAP Labeling of Watershed Segments," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 3, pp. 326-332, Mar. 2001.

[19] J. Sethian, *Level Set Methods and Fast Marching Methods.* Cambridge Univ. Press, 1999.

[20] J. Shi and J. Malik, "Motion Segmentation and Tracking Using Normalized Cuts," *Proc. Int'l Conf. Computer Vision,* 1998.

[21] J. Shi and C. Tomasi, "Good Features to Track," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 1994.

[22] P. Smith, T. Drummond, and R. Cipolla, "Layered Motion Segmentation and Depth Ordering by Tracking Edges," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 4, pp. 479-494, Apr. 2004.

[23] R. Szeliski, "Video Mosaics for Virtual Environments," *IEEE Computer Graphics and Applications,* vol. 16, no. 2, pp. 22-30, 1996.

[24] H. Tao, H. Sawhney, and R. Kumar, "Object Tracking with Bayesian Estimation of Dynamic Layer Representations," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 1, pp. 75-89, Jan. 2002.

[25] P. Torr and D. Murray, "Outlier Detection and Motion Segmentation," *Proc. SPIE Sensor Fusion Conf. VI,* pp. 432-443, 1993.

[26] R. Vidal and Y. Ma, "A Unified Algebraic Approach to 2-D and 3-D Motion Segmentation," *Proc. European Conf. Computer Vision,* 2004.

[27] J. Wang and E. Adelson, "Representing Moving Images with Layers," *IEEE Trans. Image Processing,* vol. 3, no. 5, pp. 625-638, 1994.

[28] Y. Weiss, "Smoothness in Layers: Motion Segmentation Using Nonparametric on Homographics," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 1997.

[29] J. Wills, S. Agarwal, and S. Belongie, "What Went Where," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2003.

[30] J. Xiao and M. Shah, "Two-Frame Wide Baseline Matching," *Proc. Int'l Conf. Computer Vision,* 2003.

[31] J. Xiao and M. Shah, "Motion Layer Extraction in the Presence of Occlusion Using Graph Cut," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2004.
[32] N. Xu, R. Bansal, and N. Ahuja, "Object Segmentation Using Graph Cuts Based Active Contours," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2003.
[33] L. Zelnik-Manor and M. Irani, "Multi View Subspace Constraints on Homographies," *Proc. Int'l Conf. Computer Vision,* 1999.
[34] Y. Zhou and H. Tao, "A Background Layer Model for Object Tracking Through Occlusion," *Proc. Int'l Conf. Computer Vision,* 2003.
[35] http://www.cs.ucf.edu/~vision/projects/motion_layer_extraction/, 2005.

Jiangjian Xiao received the BS degree in material engineering and the MS degree in automatic control, both from the Beijing University of Aeronautics and Astronautics, Beijing, P.R. China, in 1994 and 1997, respectively. He also received the MS and PhD degrees in computer science, both from the University of Central Florida (UCF), Orlando, Florida, in 2001 and 2004, respectively. His research interests include wide baseline matching, multiview synthesis, stereo, motion segmentation and tracking, video registration, video synthesis, and image-based rendering. He is a student member of the IEEE.

Mubarak Shah is a professor of computer science and the founding director of the Computer Vision Laboratory at the University of Central Florida (UCF), where he is a researcher in computer vision. He is the coauthor of two books, *Video Registration* (2003) and *Motion-Based Recognition* (1997), both by Kluwer Academic Publishers. He has supervised several PhD, MS, and BS students to completion and is currently directing 20 PhD and several BS students. He has published close to 150 papers in leading journals and conferences on topics including activity and gesture recognition, violence detection, event ontology, object tracking (fixed camera, moving camera, multiple overlapping, and nonoverlapping cameras), video segmentation, story and scene segmentation, view morphing, ATR, wide-baseline matching, and video registration. Dr. Shah is a fellow of IEEE, was an IEEE Distinguished Visitor speaker for 1997-2000, and is often invited to present seminars, tutorials, and invited talks all over the world. He received the Harris Corporation Engineering Achievement Award in 1999, the TOKTEN awards from UNDP in 1995, 1997, and 2000, the Teaching Incentive Program award in 1995 and 2003, the Research Incentive Award in 2003, and the IEEE Outstanding Engineering Educator Award in 1997. He is an editor of the international book series on *Video Computing,* editor in chief of the *Machine Vision and Applications* journal, and an associate editor of the *Pattern Recognition* journal. He was an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and a guest editor of the special issue of the *International Journal of Computer Vision* on video computing.