British Machine Vision Conference (BMVC)
Edinburgh, Scotland, September 2006

Motion Segmentation at Any Speed

Shrinivas J. Pundlik and Stanley T. Birchfield
Electrical and Computer Engineering Department
Clemson University, Clemson, SC 29634 USA
{spundli,stb}@clemson.edu

Abstract

We present an incremental approach to motion segmentation. Feature points are detected and tracked throughout an image sequence, and the features are grouped using a region-growing algorithm with an affine motion model. The primary parameter used by the algorithm is the amount of evidence that must accumulate before features are grouped. Contrasted with previous work, the algorithm allows for a variable number of image frames to affect the decision process, thus enabling objects to be detected independently of their velocity in the image. Procedures are presented for grouping features, measuring the consistency of the resulting groups, assimilating new features into existing groups, and splitting groups over time. Experimental results on a number of challenging image sequences demonstrate the effectiveness of the technique.

1 Introduction

Common fate, also known as common motion, is a powerful cue for image understanding. According to Gestalt psychology, the human visual system groups pixels that move in the same direction in order to focus attention on perceptually salient regions of the scene. As a result, the ability to segment images based upon pixel motion is important for automated image analysis.

Previous methods for motion segmentation can be divided into several categories. Layered approaches assign pixels to layers, compute a parametric motion for each layer, and determine the number of layers; these techniques often use expectation-maximization (EM) [13, 14, 1, 5] or graph cuts [16, 15] to minimize a functional. Other methods formulate the problem as one of multi-body factorization, which is solved using subspace constraints on a measurement matrix computed over some number of frames [12, 7, 6]. Additional approaches include clustering pixels based on their motion profiles using eigenvectors [9], minimizing a continuous energy functional over a spatiotemporal volume using spatio-temporal gradients [3], or applying the rank constraint to feature correspondences in order to divide the sequence into locally coherent regions [8]. Object-level grouping of affine patches in a video shot for the purpose of video retrieval has also been explored [11].

A common theme among these algorithms is their batch processing of image sequences. Previous techniques operate on either two images at a time or on a spatio-temporal volume containing a fixed number of images. In the case of multiple frames, the

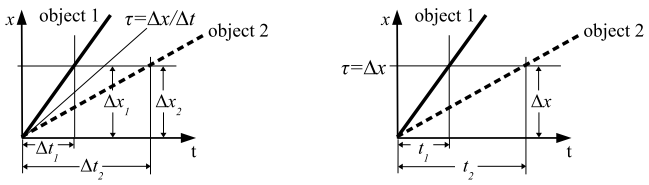


Figure 1: A fast object (object 1) and a slow object (object 2) move against a stationary background. LEFT: If the threshold τ is dependent upon velocity, then the slowly moving object is never detected. RIGHT: Making τ relative to a fixed reference frame enables both objects to be detected independently of their speed, as soon as enough image evidence accumulates (time t_1 for object 1 and t_2 for object 2).

motion of the object is often considered to be constant or slowly changing throughout the sequence of frames under consideration.

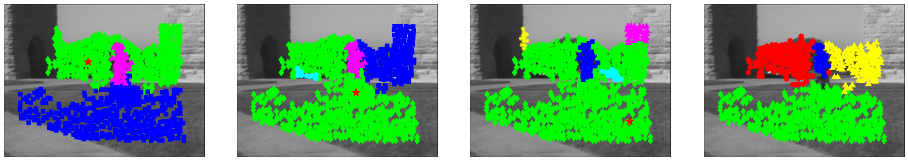
The drawback of using a fixed number of image frames is illustrated with a simple example in Figure 1 in which two objects move at different speeds, $\Delta x_1/\Delta t_1$ and $\Delta x_2/\Delta t_2$, respectively, relative to a static background. With a fixed number of frames, the reference frame constantly changes as the current frame changes. The amount of evidence in the block of frames is dependent upon the velocity of the object, so that the slowly moving object is never detected because $\Delta x_2/\Delta t_2 < \tau$. On the other hand, using a fixed reference frame leads to a variable number of image frames in the block, thus enabling objects to be detected independently of their speed once enough evidence has accumulated so that $\Delta x_i > \tau$. Note that the problem becomes even more acute in the case of multiple objects, all moving at different velocities, because realistic image noise may prevent a single parameter τ from working even if it is selected manually for the specific sequence.

In this paper we present an incremental approach to motion segmentation. Sparse feature points are detected and tracked throughout the image sequence, and segmentation is performed each time a new image frame becomes available. An efficient region-growing technique is used to group the features according to a low-order parametric model. Objects are detected incrementally as enough evidence accumulates to indicate that they are distinct from their surroundings. The only parameter of the algorithm is the amount of evidence (in terms of motion variation) needed to decide that features belong to different groups. The number of groups is determined automatically and dynamically as objects move around and enter and leave the scene. We demonstrate the technique on several challenging and long sequences exhibiting unpredictable and non-rigid motion, occlusion and disocclusion, and containing up to six objects at a time.

2 Tracking Features

Feature points are automatically selected and tracked using the Kanade-Lucas-Tomasi (KLT) feature tracker [2], which computes the displacement $\mathbf{d} = [d_x \quad d_y]^T$ that minimizes the sum of squared differences between consecutive image frames I and J:

$$\int\int_W \left[I(\mathbf{x} - \frac{\mathbf{d}}{2}) - J(\mathbf{x} + \frac{\mathbf{d}}{2}) \right]^2 d\mathbf{x},$$



seed point 1 seed point 2 seed point 3 consistent groups

Figure 2: The first three images show results of the algorithm `GroupFeatures` on the statue sequence using different random seed points. The start and end frames are 1 and 7, respectively. The red star indicates the initial seed point. The last image shows the consistent groups found by `GroupConsistentFeatures`.

where W is a window of pixels at $\mathbf{x} = [x \quad y]^T$ around the feature point. This nonlinear error is minimized by repeatedly solving its linearized version:

$$Z\mathbf{d} = \mathbf{e},$$

where

$$\begin{aligned} Z &= \sum_{\mathbf{x}\in W} \mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x}) \\ \mathbf{e} &= \sum_{\mathbf{x}\in W} \mathbf{g}(\mathbf{x})[I(\mathbf{x}) - J(\mathbf{x})], \end{aligned}$$

and $\mathbf{g}(\mathbf{x}) = \partial \frac{I(\mathbf{x})+J(\mathbf{x})}{2} / \partial \mathbf{x}$ is the spatial gradient of the average image. These equations are similar to the standard Lucas-Kanade equations but are symmetric with respect to the two images. As in [10], features are automatically selected as those points in the image for which both eigenvalues of Z are greater than a minimum threshold. The affine test comparing the appearance of a feature in the current image with its appearance in the first image is used to declare features lost [10].

3 Grouping Features Using Two Frames

Once features are tracked from one image frame to another, the features are grouped using an affine motion model on the displacements of the coordinates of the features. In other words, whether a feature f is incorporated into a group with an affine motion model A is determined by measuring the difference

$$\texttt{Diff}(f, A) = ||A\mathbf{x}^{(ref)} - \mathbf{x}^{(curr)}||,$$

where $\mathbf{x}^{(curr)}$ and $\mathbf{x}^{(ref)}$ are the coordinates of the feature in the current and reference frames, respectively, and $||\cdot||$ is the L_2 norm. For simplicity, homogeneous coordinates are used in this equation, so that A is a 3×3 matrix with $[0 \quad 0 \quad 1]$ as the bottom row.

A region growing approach is adopted, as shown in the algorithm `GroupFeatures`. First, all the features are labeled as 'ungrouped', and a feature point is selected at random as the seed point to begin a new group $\mathscr{F}$. The motion of the group is computed by fitting affine parameters to the motion of the feature and all of its immediate ungrouped neighbors $\mathscr{N}_u(f)$ using a Delaunay triangulation of the features in the first frame. The

Algorithm: `GroupFeatures`

Input: A set of features with motion vectors
Output: A set of groups of features

1. Set all features to 'ungrouped'
2. While at least one feature is 'ungrouped',
   (a) Select a random 'ungrouped' feature f
   (b) Set $\mathscr{F} \leftarrow \{f\} \bigcup \mathscr{N}_u(f)$
   (c) Compute affine motion model A of $\mathscr{F}$
   (d) Repeat until $\mathscr{F}$ does not change
      i. Set $\mathscr{F} \leftarrow \{f'\}$, where f' is the feature closest to the centroid of $\mathscr{F}$
      ii. Repeat until $\mathscr{S}$ is empty
         (a) Find similar nearby features $\mathscr{S} \leftarrow \texttt{Similar}(\mathscr{N}_u(\mathscr{F}), A, \tau)$
         (b) Set $\mathscr{F} \leftarrow \mathscr{F} \bigcup \mathscr{S}$
         (c) Compute affine motion model A of $\mathscr{F}$
   (e) Set all features in $\mathscr{F}$ to 'grouped'

process continues to add any neighbor of a feature in the group whose motion is similar to the motion of the group. The function $\mathscr{S} \leftarrow \texttt{Similar}(\mathscr{N}_u(\mathscr{F}), A, \tau)$ returns all the ungrouped neighbors f of the features in $\mathscr{F}$ for which $\texttt{Diff}(f, A) < \tau$, where τ is a threshold indicating the maximum motion difference allowed. When no more features can be added to the group, the group is reset to the feature closest to the centroid of the group, and the process begins again. Convergence is usually obtained within two or three iterations.

Now that a single group has been found, all the features in the group are labeled with a unique group id. The procedure then starts again using another random feature as the seed point among all those that have not yet been grouped, and the process is repeated until all features have been grouped. Note that the algorithm automatically determines the number of groups.

If there is not enough information in the motion vectors to reliably group the features, then the output of `GroupFeatures` will be sensitive to the randomly chosen seed points. To solve this problem, we introduce a *seed-point consistency check* which is reminiscent of the left-right consistency check of stereo matching [4]. In the algorithm `GroupConsistentFeatures`, the grouping algorithm is run N_s times starting from different random seed points. A consistency matrix c is maintained in which $c(f, f')$ is the number of results in which f and f' belong to the same group. A set of features is said to form a consistent group if $c(f, f') = N_s$ for all features in the set. The collection $\mathscr{C}$ of consistent groups larger than a minimum size $n_{\min}$ are retained, and all the remaining features are set again to 'ungrouped'. Figure 2 displays the varying groups for different seed points on an example sequence, along with the consistent groups.

Algorithm: `GroupConsistentFeatures`

Input: A set of features with motion vectors
Output: A set of groups of features, and a set of ungrouped features

1. $c(f, f') \leftarrow 0$ for every pair of features f and f'
2. for $i \leftarrow 1$ to N_s,
   (a) Run `GroupFeatures`
   (b) For each pair of features f and f', increment $c(f, f')$ if f and f' belong to the same group
3. Set $\mathscr{C} \leftarrow \{\}$ (empty set)
4. Repeat until all features have been considered,
   (a) Gather a maximal set $\mathscr{F}$ of consistent features such that $c(f, f') = N_s$ for all pairs of features in the set
   (b) if $|\mathscr{F}| > n_{\min}$, then $\mathscr{C} \leftarrow \mathscr{C} \bigcup \mathscr{F}$
5. Set all features that are not in a large consistent feature set (i.e., there is no $\mathscr{F} \in \mathscr{C}$ such that $f \in \mathscr{F}$) to 'ungrouped'

4 Maintaining Feature Groups Over Time

Like previous algorithms, the grouping procedure of the last section operates on exactly two image frames. If the frames are spaced closely together, then slowly moving objects will not be detected. On the other hand, if the frames are spaced far apart, then the affine motion assumption and the feature tracking are likely to fail. As a result, algorithms that fix the inter-frame spacing (whether operating on a pair of frames or on a spatiotemporal block of frames) make potentially dangerous assumptions about the amount of motion of the objects in the scene. The dynamic nature of feature points, particularly their loss over time as the scene changes, should be taken into account when comparing successive images.

As shown in the algorithm `MaintainGroups`, our approach performs three computations when a new image frame becomes available. First, the consistent grouping procedure just described is applied to all the ungrouped features. This step generates additional groups if sufficient evidence for their existence has become available.

Secondly, the consistent grouping procedure is applied to the features of each existing group. If a group exhibits multiple motions, then it will be split into multiple groups and/or some of its features will be discarded from the group and labeled instead as 'ungrouped'. Because groups tend to be stable over time, we have found that this procedure does not need to be performed every frame. The function `Lost` returns the number of features in the reference frame of a group that have been lost. If this number exceeds a threshold (λ), then the reference frame is updated by setting it to the current frame, and

Algorithm: `MaintainGroups`

Input: A set of groups, and a set of ungrouped features
Output: A set of groups, and a set of ungrouped features

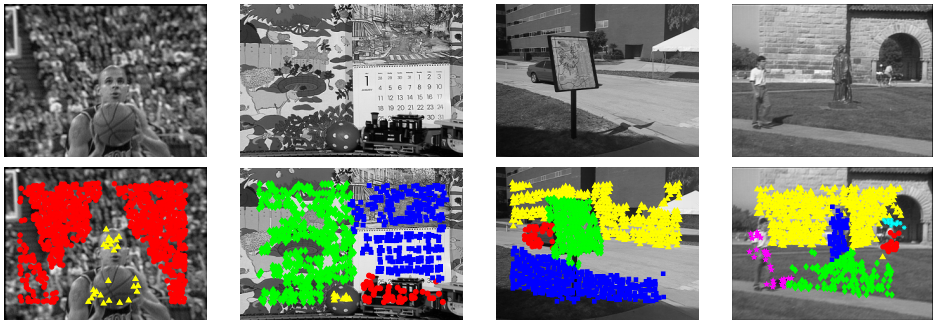
1. *Grouping.* Run `GroupConsistentFeatures` on all the ungrouped features
2. *Splitting.* For each group $\mathscr{F}$ such that $\texttt{Lost}(\mathscr{F}) \geq \lambda$,
   (a) Update reference frame for $\mathscr{F}$.
   (b) Run `GroupConsistentFeatures` on the features in $\mathscr{F}$
3. *Assimilation.* For each 'ungrouped' feature f,
   (a) $\mathscr{S} \leftarrow \mathscr{N}_g(f)$
   (b) If $\mathscr{S}$ is nonempty,
      i. Set $\mathscr{G}$ to the set of groups to which the features in $\mathscr{S}$ belong
      ii. For each $\mathscr{F} \in \mathscr{G}$,
         (a) Compute affine motion model A of $\mathscr{F}$
         (b) Set $\mathscr{F} \leftarrow \mathscr{F} \bigcup \{f\}$ and f to 'grouped' if
            - $\texttt{Diff}(f,A) \leq \tau$ and
            - $\texttt{Diff}(f,A) \leq \texttt{Diff}(f,A') + \tau$ for the affine motion A' of any other group $\mathscr{F}' \in \mathscr{G}$

the grouping procedure is run. We set λ to 25% of the features in the group.

The third computation is to assimilate ungrouped features into existing groups. For each ungrouped feature f, we consider its immediate neighbors $\mathscr{N}_g$ (in the Delaunay triangulation) that are already grouped. If there are no such neighbors, then no further tests are performed. If there is exactly one such neighbor, then the feature is assimilated into the neighbor's group if the motion of the feature is similar to that of the group, using the same threshold τ used in the grouping procedure. If there is more than one such neighbor belonging to different groups, then the feature is assimilated into one of the groups only if its motion is similar to that of the group and is dissimilar to that of the other groups, using the threshold τ.

5 Experimental Results

The algorithm was tested on four grayscale image sequences, shown in Figure 3. Because of space limitations, only one image frame from each of the sequences is shown. In the first sequence (20 frames total), a basketball player moves down in the image as he prepares to shoot a freethrow, while the camera moves slightly down. The player (yellow triangles) is successfully segmented from the background (red circles). In the second sequence (101 frames), a toy train pushes a ball to the left and a calendar slides down in front of a textured background, while the camera zooms out and moves slightly left.



freethrow mobile-calendar car-map statue

Figure 3: TOP: A frame from each of the four sequences. BOTTOM: Results of the proposed algorithm on the images, with different groups indicated by different colors and shapes.

All of the objects are detected, even though the ball (yellow triangles) and train (red circles) move faster than the calendar (blue squares) and background (green diamonds). In the third sequence (35 frames), a car drives in a straight line behind a map while the camera remains stationary. The car (red circles), map (green diamonds), foreground (blue squares), and background (yellow triangles) are detected.

The fourth sequence (520 frames) is by far the most difficult. In this sequence a hand-held camera is moved in an uncontrolled fashion around a statue, while a bicyclist drives behind the statue and a pedestrian walks in front of the statue. The motion of the objects is not linear, and several objects appear and disappear over the course of the sequence. The last image of Figure 3 shows that the algorithm successfully detects the statue (blue squares), the wall behind the statue (yellow triangles), the grass (green diamonds), the pedestrian (magenta stars), the bicyclist (red circles), and the portion of the background visible through the arch (cyan asterisks).

As mentioned earlier, the algorithm operates in an incremental fashion, creating and maintaining groups of features as more evidence becomes available. The number of groups is determined automatically. Figure 4 displays the dynamic progress of the results on each of the sequences. In the first sequence the basketball player becomes separable from the background almost immediately. In the second sequence the faster train and ball become separable after only two frames, while six frames are needed to separate the calendar and background. In the third sequence the objects are detected one by one.

To better illustrate the progress of the algorithm over time, several images from the statue sequence are shown in Figure 5. With just two frames the algorithm is able to separate the background (containing the wall and the trees) from the foreground (containing the grass and the statue). By frame 8 the statue has been separated from the grass. The bicyclist enters in frame 151 (detected in frame 185), becomes occluded by the statue in frame 312, and emerges on the other side of the statue in frame 356 (detected again in frame 444). Because the algorithm does not attempt correspondence between occluded and disoccluded objects, the bicylist group receives a different group id after the disocclusion. The pedestrian enters the scene in frame 444 and is segmented successfully, although the non-rigid motion prevents the feature tracker from maintaining a large num-

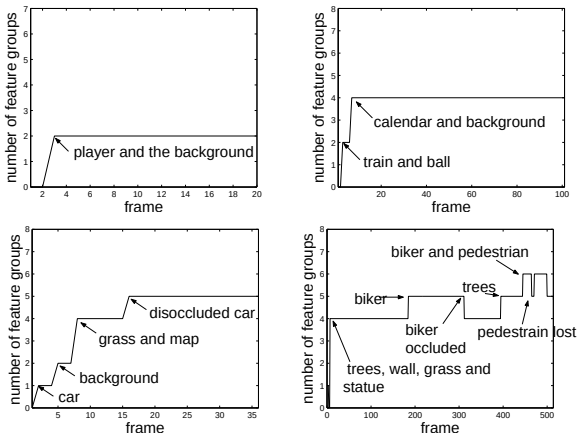


Figure 4: The dynamic progress of the algorithm on the four sequences, plotted as the number of feature groups versus time. The number of groups is determined dynamically and automatically by the algorithm.

ber of features throughout. The pedestrian occludes the statue from frames 346 to 501, after which the statue is regrouped. Features on the grass are not regrouped due to an apparent error in the KLT feature tracker that prevents features from being replenished on the grass after the pedestrian passes.

One of the advantages of the proposed algorithm is its lack of parameters. The primary parameter, τ, which was set to 1.5 for all the results in this section, governs the amount of image evidence needed before features are declared to be moving consistently with one another. The same parameter is also used to determine when to split an existing group. Significantly, the results are insensitive even to this parameter: If τ is increased, then the algorithm simply waits longer before declaring a group (assuming that the motion difference between objects accumulates over time), while if τ is decreased then the groups are declared sooner. As an example, Figure 6 shows essentially the same grouping results on the statue sequence using the value $\tau = 0.7$, but the groups appear nearly twice as fast as before. Similarly, nearly identical results when the algorithm is run on every other image of the sequence (thus doubling the motions) without changing any parameters (i.e., $\tau = 1.5$). The algorithm is fast, requiring only 20 ms per frame (excluding feature tracking) for a sequence of 320×240 images with 1000 features on a 2.8 GHz P4 computer using an unoptimized C++ implementation.

6 Conclusion

We have approached the problem of motion segmentation from a novel point of view by removing the usual restriction of batch processing. Addressing the fact that motion is inherently a differential concept, our technique processes images in an incremental fashion. Objects are segmented as soon as enough evidence is available to distinguish them from their surroundings, and their identities are maintained over time until they are occluded or leave the scene. The algorithm uses a single parameter, namely the amount of motion

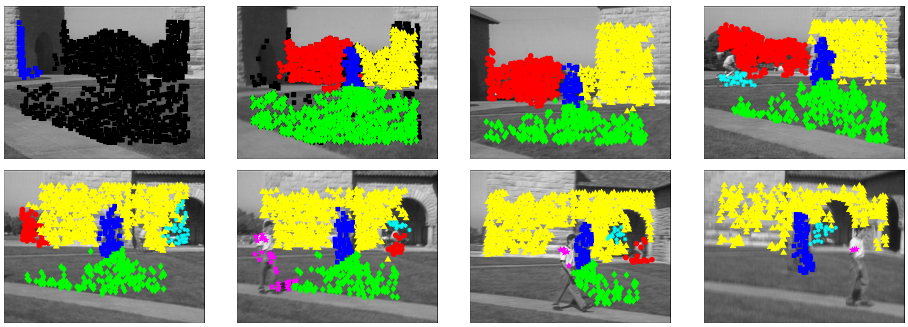


Figure 5: Results of the algorithm on several frames of the statue sequence (shown in lexicographic order: frames 3, 8, 64, 185, 395, 445, 487, and 520). The feature points in black in the first two frames are 'ungrouped'. After the first frame in which all of the features are ungrouped, subsequent images show the formation of new groups or the splitting of existing groups due to the arrival or departure of entities in the scene.

variation allowable within a region, and it automatically and dynamically computes the number of objects in the scene. The technique presented here groups sparse features using an efficient region-growing algorithm that utilizes a novel consistency check to ascertain whether the results should be trusted or more evidence is needed.

The work presented here can be extended in several ways. First, the algorithm does not maintain the identity of objects after they have been occluded. A natural solution would be to use a layered representation, which would help particularly in the case of partial occlusion. Secondly, the boundaries around the groups should be incorporated into the algorithm in order to enable the technique to operate on this complementary information. Finally, for applications in which dense segmentation is required, the technique could be combined with pixel-level assignments in order to fill in the areas left empty by the current sparse representation.

References

[1] S. Ayer and H. S. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *Proceedings of the 5th Intl. Conference on Computer Vision*, pages 777–784, June 1995.

[2] S. Birchfield. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker, http://www.ces.clemson.edu/~stb/klt/.

[3] D. Cremers and S. Soatto. Motion competition: a variational approach to piecewise parametric motion. *Intl. Journal of Computer Vision*, 62(3):249–265, May 2005.

[4] P. Fua. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 1292–1298, 1991.

[5] N. Jojic and B. J. Frey. Learning flexible sprites in video layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

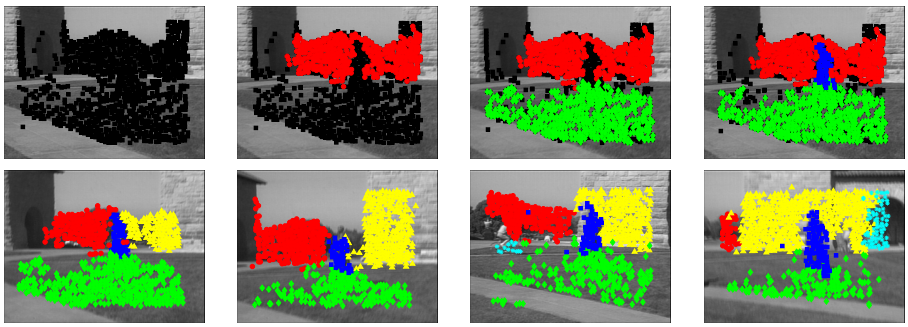


Figure 6: TOP: Results on the statue sequence with $\tau = 0.7$. By reducing the threshold by a factor of two, the segmentation occurs in nearly half the time. BOTTOM: Results on a modified statue sequence in which every other frame has been discarded, thus doubling the motion. Shown are frames 8, 64, 188, and 395, corresponding to the frame numbers of the original sequence shown in Figure 5.

[6] Q. Ke and T. Kanade. A subspace approach to layer extraction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages I: 255–262, 2001.

[7] M. Machline, L. Zelnik-Manor, and M. Irani. Multi-body segmentation: Revisiting motion consistency. In *Workshop on Vision and Modeling of Dynamic Scenes*, 2002.

[8] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Segmenting, modeling, and matching video clips containing multiple moving objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[9] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *Proc. of the 6th Intl. Conference on Computer Vision*, pages 1154–1160, 1998.

[10] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

[11] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. In *Proceedings of the European Conference on Computer Vision*, 2004.

[12] R. Vidal and S. Sastry. Optimal segmentation of dynamic scenes from two perspective views. In *CVPR*, 2003.

[13] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, Sept. 1994.

[14] Y. Weiss and E. H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 321–326, 1996.

[15] J. Wills, S. Agarwal, and S. Belongie. What went where. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages I: 37–44, 2003.

[16] J. Xiao and M. Shah. Accurate motion layer segmentation and matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.