Real-Time Motion Segmentation of Sparse Feature Points at Any Speed

Shrinivas J. Pundlik and Stanley T. Birchfield, *Senior Member, IEEE*

***Abstract*—We present a real-time incremental approach to motion segmentation operating on sparse feature points. In contrast to previous work, the algorithm allows for a variable number of image frames to affect the segmentation process, thus enabling an arbitrary number of objects traveling at different relative speeds to be detected. Feature points are detected and tracked throughout an image sequence, and the features are grouped using a spatially constrained expectation–maximization (EM) algorithm that models the interactions between neighboring features using the Markov assumption. The primary parameter used by the algorithm is the amount of evidence that must accumulate before features are grouped. A statistical goodness-of-fit test monitors the change in the motion parameters of a group over time in order to automatically update the reference frame. Experimental results on a number of challenging image sequences demonstrate the effectiveness and computational efficiency of the technique.**

***Index Terms*—Expectation–maximization (EM), feature tracking, motion segmentation.**

I. Introduction

COMMON fate, also known as common motion, is a powerful cue for image understanding [13], [32]. 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 impacting a number of important applications, including object detection [40], tracking [30], surveillance [16], robotics [20], image and video compression [2], scene reconstruction [12], and video matting [45].

Traditional motion segmentation algorithms limit themselves to using the information between times t and $t+K$, where K is a constant parameter, in order to determine the number and composition of the groups [7], [8], [18], [32], [34], [42], [46]. Ignoring the fact that motion is inherently a differential concept, such an approach is similar to estimating the derivative of a function using finite differences with a fixed window size: Too small of a window increases susceptibility to noise, whereas too large of a window ignores important details.

The drawback of using a fixed number of image frames is shown in Fig. 1(a) with two objects moving at different speeds, $\Delta x_1/\Delta t_1$ and $\Delta x_2/\Delta t_2$, respectively, relative to a static background, where $\Delta x_1 = \Delta x_2$. Because the amount of evidence in the block of frames is dependent upon the velocity of the object relative to the background, the slowly moving object is never detected (i.e., separated from the background) because $\Delta x_2/\Delta t_2 < \tau$, where $\tau = \Delta x/\Delta t$ is a threshold indicating the minimum amount of relative motion between two objects required to separate them. The threshold must be set above the noise level (of the motion estimator) in order to avoid oversegmentation, but if it is set too high, then objects moving slowly relative to each other will not be distinguished. The solution to this problem is to use a fixed reference frame with the threshold $\tau = \Delta x$ indicating the amount of relative displacement needed between two objects, as shown in Fig. 1(b). As additional images become available over time, evidence for the motion of an object is allowed to accumulate so that objects are detected regardless of their speed once their overall displacement exceeds the threshold, i.e., $\Delta x_i > \tau$.

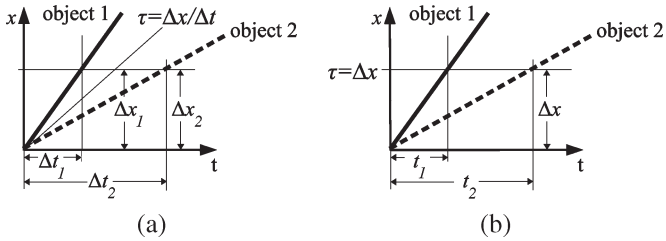

Fig. 1. Fast object (object 1) and a slow object (object 2) move against a static background. (a) If the threshold τ is dependent upon velocity, then the slowly moving object is never detected because $\Delta x_2/\Delta t_2 < \tau$. (b) In contrast, a fixed reference frame enables both objects to be detected independently of their speed, as soon as enough image evidence accumulates (time t_1 for object 1 and t_2 for object 2).

Of course, in practice, the reference frame must be updated eventually due to the divergence over time of the actual pixel motion from the low-order model of the group motion. Thus, a crucial issue in designing a motion segmentation system that operates on variable speeds is to adaptively update the reference frame. To do so, the system must be able to distinguish between two common cases. First, the pixels in a region may not be moving coherently due to the presence of multiple objects occupying the region, in which case the group should be split. Second, the motion divergence may be due to unmodeled effects in the underlying motion model, in which case the reference frame should be updated.

In this paper, we present an algorithm for segmenting sparse feature points in long image sequences by processing images sequentially [28]. We introduce a spatially constrained expectation–maximization (EM) framework which replaces the traditional assumption of conditional independence between feature labels with a Markov assumption between them. A

Manuscript received December 16, 2006; revised August 22, 2007. This paper was recommended by Associate Editor I. Bloch.

The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634-5124 USA.

Digital Object Identifier 10.1109/TSMCB.2008.919229

region-growing approach incorporating a novel consistency check is used to efficiently initialize the EM procedure in a greedy manner [41], which maintains the feature groups over time by automatically creating new groups from ungrouped features, assimilating ungrouped features into existing groups, and splitting groups as needed. The chi-square (χ^2) test is used to determine whether the motions of the features in each group can be explained by the motion models and whether the reference frame needs to be updated. By using a variable number of image frames in performing the segmentation, results are therefore achieved with any object speed.

The algorithm performs in real time on a standard computer, handles an arbitrary number of groups, and is demonstrated on several challenging sequences involving independently moving objects, occlusion, and parallax effects. The number of groups is determined automatically and dynamically as objects move relative to one another and as they enter and leave the scene. The primary parameter of the algorithm is a threshold that captures the amount of evidence (in terms of motion variation) needed to decide which features belong to different groups. We present results of the algorithm on a variety of sequences to demonstrate the effectiveness of the approach.

II. Previous Work

Motion segmentation is a classic problem in computer vision that has been explored by various researchers over the years. One traditional approach has been to assign the pixels to layers and to compute a parametric motion for each layer, following Wang and Adelson [1], [42], [43]. This approach determines a dense segmentation of the video sequence by minimizing an energy functional, typically using either EM or graph cuts. In a series of papers, Jojic *et al.* [6], [18], [19] demonstrate algorithms that are capable of segmenting sequences and representing those sequences using example patches. In other recent work, Smith *et al.* [34] present a technique for dense motion segmentation that applies EM to the edges in a sequence. Xiao and Shah [46] combine a general occlusion constraint, graph cuts, and alpha matting to perform accurate dense segmentation. Kumar *et al.* [23] combine loopy belief propagation with graph cuts to densely segment short video sequences. Cremers and Soatto [8] minimize a continuous energy functional over a spatiotemporal volume to perform two-frame segmentation, a technique which is extended by Brox *et al.* [5]. Spatiotemporal coupling has been enforced using graph cuts and hidden layers representing occlusion [11] and by dynamic Bayesian networks [35].

An alternate approach is to formulate the problem as one of multibody factorization, which is solved using subspace constraints on a measurement matrix computed over a fixed number of frames, based upon the early work of Costeira and Kanade [7]. Ke and Kanade [21] extended this work by presenting a low-dimensional robust linear subspace approach to exploit the global spatial–temporal constraints. Zelnik-Manor *et al.* [48] expand upon traditional measures of motion consistency by taking into account the temporal consistency of behaviors across multiple frames in the video sequence, which can then be applied to 2-D, 3-D, and some nonrigid motions. Vidal *et al.* [38], [39] show that multiple motions can, in theory, be recovered and segmented simultaneously using the multibody epipolar constraint, although segmentation of more than three bodies has proved to be problematic in practice. In recent work, Yan and Pollefeys [47] have examined the effects of articulated and degenerate motion upon the motion matrix, to which recursive spectral clustering is applied to segment relatively short video sequences. In other recent work, Gruber and Weiss [15] extend the standard multibody factorization approach by incorporating spatial coherence.

The problem has been approached from other points of view as well. Various researchers have utilized the assumption that the dominant motion is that of the background in order to detect independently moving objects [17], [27], [30]. Other researchers have explored the connection between bottom-up and top-down processing, noting that some top-down evidence will be needed for segmentation algorithms to produce the results expected by human evaluators [22], [25], [36]. Wills *et al.* [44] combine sparse feature correspondence with layer assignments to compute dense segmentation when objects undergo large interframe motion, followed by more recent work in which the time linearity of the homographies obtained under the assumption of constant translation is exploited in order to segment periodic motions from nonperiodic backgrounds [24]. Shi and Malik [31], [32] cluster pixels based on their motion profiles using eigenvectors, a technique that has proved successful for monocular cues but which does not take occlusion information into account. Rothganger *et al.* [29] apply the rank constraint to feature correspondences in order to divide the sequence into locally coherent 3-D regions. In two pieces of recent interesting work, Sivic *et al.* [33] use object-level grouping of affine patches in a video shot to develop a system for video retrieval, and Criminisi *et al.* [9] present a real-time foreground/background segmentation technique with sufficient accuracy for compositing the foreground onto novel backgrounds.

Surveying this literature, several common themes emerge. First, they generally process the images in batch, operating either on two images at a time or on a spatiotemporal volume containing a fixed number of images. In the case of multiple frames, the motion of the object is often considered to be constant or slowly changing throughout the sequence of frames under consideration to simplify the integration of information over time. Second, the techniques are usually limited to a small time window in which the motion of all of the objects is expected to be well behaved. Additionally, it is generally the case that the focus of the research is not upon computationally efficient algorithms, leading, in some cases, to techniques that require orders of magnitude more than what is available in real-time applications. Finally, some of the techniques are limited to a small number of objects (two or three) due to either the computational burden or more fundamental aspects of the algorithm. To date, no system has been produced that processes frames in real time, can handle an arbitrary number of objects undergoing complex motions, and operates on arbitrarily long sequences.

Note that an alternate spatially constrained approach to EM segmentation is presented in [10], in which the mixing weights

of neighboring pixels are averaged in each iteration, for the purpose of non-real-time segmentation of single images. In contrast, our approach enforces spatial continuity by growing from a region centroid for the purpose of real-time segmentation of video sequences.

III. Formulation

Let $f^{(i)}, i = 1, \ldots, n$ be the sparse features tracked in a video sequence, and let $f_t^{(i)}$ represent the (x, y) coordinates of the ith feature in image frame t. Let $x^{(i)} = \langle f_1^{(i)}, \ldots, f_T^{(i)} \rangle$ be the trajectory of the ith feature, where T is the maximum frame number, and let $\mathcal{X} = \langle x^{(1)}, \ldots, x^{(n)} \rangle$ be all the trajectories collectively.

The trajectories of neighboring features typically exhibit a strong correlation because they follow the motion of the same surface in the world. Let $\Theta = \langle \theta_1, \ldots, \theta_k \rangle$ be the motion models of the k components from which the feature trajectories arise. Our goal is to find the maximum-likelihood explanation of the data

$$\Theta^* = \arg\max_{\Theta} P(\mathcal{X}|\Theta). \tag{1}$$

Assuming that the different trajectories are independent given Θ, we have

$$P(\mathcal{X}|\Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} P\left(x^{(i)}|c_j^{(i)}, \Theta\right) P\left(c_j^{(i)}|\Theta\right) \tag{2}$$

where $c_j^{(i)}$ is a binary variable that indicates whether feature $f^{(i)}$ belongs to component j.

Let $\phi(x^{(i)}; \theta_j) = P(x^{(i)}|c_j^{(i)}, \Theta)$ measure how well the trajectory $x^{(i)}$ fits the jth model, and let $\pi_j^{(i)} = P(c_j^{(i)}|\Theta)$ be the weight indicating the probability that feature $f^{(i)}$ belongs to component j given Θ; thus, $\sum_{j=1}^{k} \pi_j^{(i)} = 1$. Then, by converting to a log likelihood, we can rewrite the expression as

$$\Theta^* = \arg\max_{\Theta} \sum_{i=1}^{n} \log g_k\left(x^{(i)}\right) \tag{3}$$

where

$$g_k\left(x^{(i)}\right) = \sum_{j=1}^{k} \pi_j^{(i)} \phi\left(x^{(i)}; \theta_j\right). \tag{4}$$

Learning the mixture involves estimating the weights $\pi_j^{(i)}$'s and the parameters θ_j's. To do this, we use a variation of the greedy EM algorithm [41], which incrementally adds components to determine k automatically. Because we process the sequence causally, in the following discussion, T should be interpreted as the maximum frame number encountered so far.

IV. Grouping Features Using Two Frames

As with existing motion segmentation algorithms, the core of our approach involves grouping features between a pair of (not necessarily consecutive) image frames. In this paper, we use an affine motion model, so that

$$\phi(x^{(i)}; \theta_j) = \frac{1}{\sqrt{2\pi\sigma_f^2}} \exp\left\{\frac{-\|A_j f_t^{(i)} - f_{r_j}^{(i)}\|^2}{2\sigma_f^2}\right\} \tag{5}$$

where A_j is the 3×3 matrix of affine parameters (homogeneous coordinates are used, with a slight abuse of notation), r_j specifies the reference image frame of the jth group, and σ_f^2 is the variance of the Gaussian distribution. The parameters of a group are $\theta_j = \langle A_j, r_j, \mu_j \rangle$, where μ_j is the centroid.

Notice that (2) assumes that the binary labels of the features are independent given Θ, i.e., $P(c_j^{(1)}, \ldots, c_j^{(n)}|\Theta) = \prod_{i=1}^{n} P(c_j^{(i)}|\Theta)$. A more realistic formulation would take the spatial continuity of regions into account. For simplicity, assume that the features are ordered in a linear chain starting from the feature that is closest to the centroid of the group. Then, the requirement of spatial continuity yields a Markov chain

$$\begin{aligned} P\left(c_j^{(1)}, \ldots, c_j^{(n)}|\Theta\right) &= \prod_{i=1}^{n} P\left(c_j^{(i)}|c_j^{(i-1)}, \Theta\right) \\ &= \prod_{i=1}^{n} P\left(c_j^{(i)}|\Theta\right) c_j^{(i-1)} \end{aligned} \tag{6}$$

where the last equality arises from $c_j^{(i-1)}$ being a binary variable. Extending this result to 2-D, let $\epsilon_j^{(i)}$ be a binary indicator variable whose value is one if and only if there exists a path (according to a predefined neighborhood) from $f^{(i)}$ to the feature that is closest to the centroid such that $c_j^{(\ell)} = 1$ for all features $f^{(\ell)}$'s along the path. Because we do not have access to the actual labels, we instead use an estimate $\hat{\epsilon}_j^{(i)}$, which is set to one if and only if $P(c_j^{(\ell)}|\Theta) > p_\tau$ for all the features on the path.

This analysis leads to a spatially constrained EM algorithm. For each component j, log-likelihood maximization is performed using the following iterative update equations:

$$\pi_j^{(i)} \leftarrow \frac{\pi_j^{(i)} \phi\left(x^{(i)}; \theta_j\right) \hat{\epsilon}_j^{(i)}}{\sum_{j=1}^{k} \pi_j^{(i)} \phi\left(x^{(i)}; \theta_j\right) \hat{\epsilon}_j^{(i)}} \tag{7}$$

$$\hat{\epsilon}_j^{(i)} \leftarrow \left\{\min_{\ell} \pi_j^{(\ell)}\right\} > p_\tau \tag{8}$$

$$\mu_j \leftarrow \frac{\sum_{i=1}^{n} \pi_j^{(i)} f^{(i)}}{\sum \pi_j^{(i)}} \tag{9}$$

$$A_j \leftarrow \arg\min_{\mathbf{a}} \|W(F_t \mathbf{a} - F_{r_j})\|^2 \tag{10}$$

where W is a diagonal weighting matrix with elements $W_{ii} = \pi_j^{(i)}$, F_t is a matrix containing the features at frame t, and $\mathbf{a}$ is a vectorization of the affine matrix.

Although EM is guaranteed to converge to a local minimum, it requires a good initial guess. We adopt a "winner-take-all" strategy [26], shown in the algorithm `GroupFeatures`,

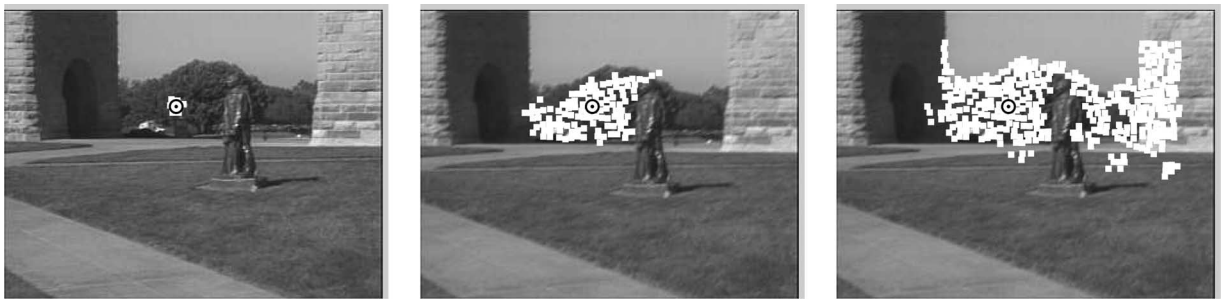

Fig. 2. Formation of a feature group by region growing, using the motion between two frames of a sequence. (Left) The initial group with the seed point f (bull's eye of concentric black, white, and black circles) and its immediately neighboring ungrouped features $\mathcal{N}_u(f)$ in the Delaunay triangulation. (Center) The group after the first iteration when $\mathcal{S}$ is empty. (Right) The final feature group after `GroupFeatures` has converged on a solution. Repeated iterations do not produce any changes in the feature group.

to provide this starting point. Groups are added one at a time by region growing from a random ungrouped feature, and the region growing is performed iteratively for each group after adjusting the centroid using all the features gathered in the previous iteration. The function $\mathcal{N}(i;t)$ returns the indices of all the features that are immediate neighbors of feature $f^{(i)}$ in the Delaunay triangulation at frame t, and the binary vector b keeps track of which features have already been considered for grouping. The output of this procedure is the number of groups, along with the binary weights indicating the membership of the features in the groups.

`GroupFeatures`
Input: Features $f^{(i)}$, $i = 1, \ldots, n$ and frames t and r
Output: k (number of groups), and $\pi_j^{(i)}$, $j = 1, \ldots, k$
1) Set $k \leftarrow 0$
2) Set $\pi_{k+1}^{(i)} \leftarrow 0$, $i = 1, \ldots, n$
3) Set $b^{(i)} \leftarrow 0$, $i = 1, \ldots, n$
4) Repeat until a random feature cannot be found
a) Select a random $f^{(\ell)}$ such that $b^{(\ell)} = 0$
b) Set $\pi_{k+1}^{(i)} \leftarrow 1, \forall i \in \{\ell, \mathcal{N}(\ell;t)\}$
c) Set $r_{k+1} \leftarrow r$, and compute A_{k+1} using (10)
d) Repeat until μ_{k+1} does not change
i) Set μ_{k+1} using (9)
ii) Set $\pi_{k+1}^{(i)} \leftarrow 0, \forall i \neq \ell$,
where $f^{(\ell)}$ is the feature closest to μ_{k+1}
iii) Repeat as long as $\pi_{k+1}^{(i)}$ changes for some i
(a) For each ℓ such that $\pi_{k+1}^{(\ell)} = 1$,
if $i \in \mathcal{N}(\ell;t)$ and $\pi_{k+1}^{(i)} = 0$
and $\phi(x^{(i)};\theta_j) > p_\tau$, then set $\pi_{k+1}^{(i)} \leftarrow 1$
(b) Compute A_{k+1} using (10)
e) Set $b^{(i)} \leftarrow \max\{b^{(i)}, \pi_{k+1}^{(i)}\}$, $i = 1, \ldots, n$
f) If $\sum_{i=1}^{n} \pi_{k+1}^{(i)} \geq n_{\min}$, then $k \leftarrow k + 1$

Fig. 2 shows the growing procedure for a single group. When no more features can be added to the group, the group is reset to the feature that is closest to the centroid of the group, and the process begins again. Convergence is usually obtained within two or three iterations. Once the first group has been found, the procedure is then repeated using another random ungrouped feature as the new seed point. Note that the algorithm automatically determines the number of groups using the single parameter p_τ, along with the minimum size $n_{\min}$ of a group.

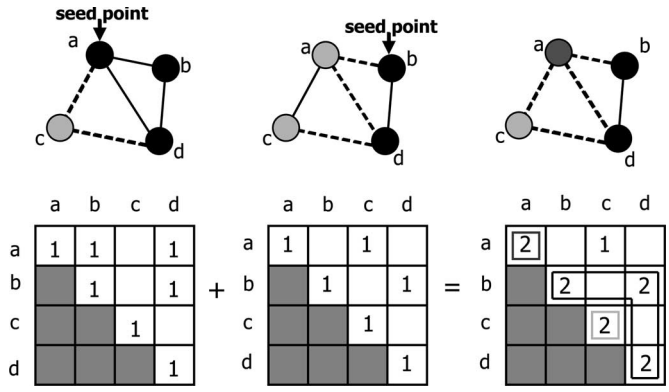

Fig. 3. Formation of consistent feature groups using the consistency matrix. The first run of `GroupFeatures` groups **a**, **b**, and **d** together while placing **c** in a separate group. The second run, using a different random seed point, groups **a** and **c** together and **b** and **d** together. Shown on the right are the three consistent groups: **b** and **d** together, **a** by itself, and **c** by itself.

If the motion models of neighboring groups are similar, then the assignment of the features will depend heavily upon 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 [14]. The grouping algorithm `GroupFeatures` is run multiple times, starting from different random seed points. A consistency matrix is maintained in which $c_{i\ell}$ is the number of results in which $f^{(i)}$ and $f^{(\ell)}$ belong to the same group. A set of features is said to form a consistent group if the features always belong to the same group as each other, i.e., $c_{i\ell} = N_s$ for all features in the set, where N_s is the number of times that `GroupFeatures` is run. The collection of consistent groups larger than the minimum size $n_{\min}$ is retained, whereas the remaining features receive zero weight for all groups. This `GroupConsistentFeatures` algorithm is shown in Fig. 3 for a simple example. The dependency of `GroupFeatures` on the random seed point, along with the results of `GroupConsistentFeatures` on an example pair of images, is shown in Fig. 4.

`GroupConsistentFeatures`
Input: Features $f^{(i)}$, $i = 1, \ldots, n$ and frames t and r
Output: k (number of groups), and $\pi_j^{(i)}$, $j = 1, \ldots, k$
1) Set $c_{i\ell} \leftarrow 0$ for every pair of features $f^{(i)}$ and $f^{(\ell)}$
2) For $i \leftarrow 1$ to N_s
a) Run `GroupFeatures`

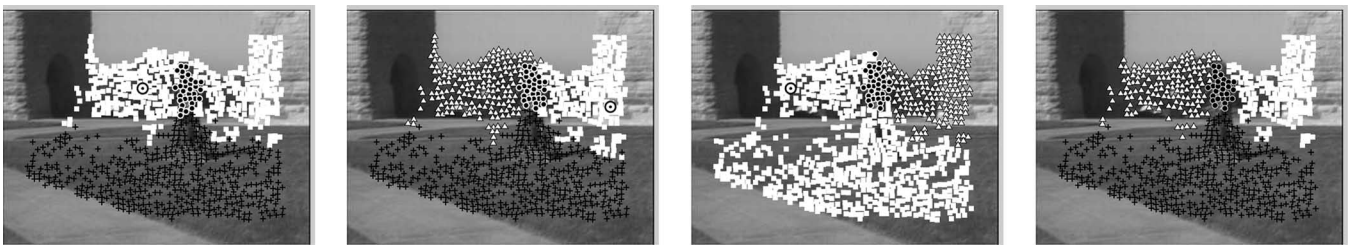

Fig. 4. (Right) Consistent groups obtained by applying the `GroupConsistentFeatures` algorithm to the results of running the algorithm `GroupFeatures` with (left three images) three different seed points. The bull's eye indicates the first seed point of each run. Notice that although the original groups are highly sensitive to the seed point, the consistent groups effectively segment the four regions of the image: (Black circles) statue, (white squares) wall, (black +'s) grass, and (white triangles) trees.

b) For each pair of features $f^{(i)}$ and $f^{(\ell)}$, increment $c_{i\ell}$ if $f^{(i)}$ and $f^{(\ell)}$ belong to the same group

3) Set $k \leftarrow 0$

4) Repeat until all features have been considered

a) Set $\pi_{k+1}^{(i)} \leftarrow 0, i = 1, \ldots, n$

b) Gather a maximal set $\mathcal{F}$ of consistent features such that $c_{i\ell} = N_s$ for all pairs of features in the set

c) If $|\mathcal{F}| > n_{\min}$, then

i) Set $\pi_{k+1}^{(i)} \leftarrow 1, \forall i$ such that $f^{(i)} \in \mathcal{F}$

ii) Set $k \leftarrow k + 1$

The algorithm `GroupConsistentFeatures` is used to initialize the number k of groups, the centroids μ_j's and affine parameters A_j's of the groups, and the weights $\pi_j^{(i)}$'s of the features. After initialization, the spatially constrained EM algorithm described in (7)–(10) is then applied. The interdependency between $\hat{\epsilon}_j^{(i)}$ and $\pi_j^{(i)}$ requires care because any weight set to zero by (8) will remain zero due to its reuse in (7). Recognizing that the prior $\pi_j^{(i)}$ in (7) does not affect the shape of the distribution represented by the weights at the stationary point, we implement the algorithm by resetting to a uniform prior in each iteration. In other words, for each group j, we perform the following steps for all $i = 1, \ldots, n$.

1) Set $\pi_j^{(i)} \leftarrow 1$.
2) Set $\pi_j^{(i)} \leftarrow \pi_j^{(i)} \phi(x^{(i)}; \theta_j)$.
3) Set $\hat{\epsilon}_j^{(i)}$ using (8) by region growing from μ_j.
4) Set $\pi_j^{(i)} \leftarrow \pi_j^{(i)} \hat{\epsilon}_j^{(i)}$.

After all the groups have been considered, the weights are normalized according to $\pi_j^{(i)} \leftarrow \pi_j^{(i)} / \sum_{j=1}^{k} \pi_j^{(i)}$. Together, this procedure constitutes the E-step. The M-step involves the simple application of (9) and (10). Concerning convergence, in our experience, the procedure settles onto a solution in few iterations, although proof of convergence is left for future work.

V. Maintaining Feature Groups Over Time

The grouping procedure of the previous section operates on exactly two (not necessarily consecutive) image frames, assuming a fixed reference frame r_j for each group. As such, it exhibits the same limitations of existing algorithms. If the time difference between the two frames being compared is short, then slowly moving objects will not be detected. On the other hand, if the time difference is large, then the affine

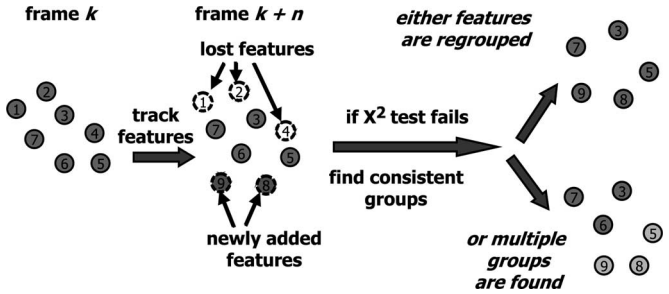

Fig. 5. Splitting an existing feature group. If the χ^2 test fails to uphold the assumption of coherent motion within the group, then the algorithm `GroupConsistentFeatures` is applied to the features in the group to facilitate regrouping. This results either in multiple groups or the discarding of outlier features (feature number 6).

motion assumption is likely to fail, and fewer features will be successfully tracked between the two frames. In this section, we embed the two-frame algorithm within a procedure for updating the groups over time in an incremental fashion so that the objects can be detected regardless of their speed. Our goal is to have a method that adapts the time difference and captures the dynamic behavior of features and objects as observed in long real-world image sequences.

The incremental procedure involves three steps. First, the initialization algorithm `GroupConsistentFeatures` is applied to all the features that have not yet been grouped in order to add new groups to the existing ones. Second, ungrouped features are assimilated into existing groups using the spatially constrained EM procedure of the previous section to update their weights. Different groups may have different reference frames, so any new feature whose start frame (the frame in which the feature was first detected) is more recent than a reference frame is not considered for grouping.

The last of the three steps is by far the most difficult. The inescapable question at this point is as follows: How can one determine whether a group exhibits coherent motion in such a way that the result is achieved for any object speed? In other words, the coherence of motion is determined by comparing the feature coordinates in the current frame with those in the reference frame. If the reference frame is never updated, then the number of features successfully tracked between the two frames will decrease (eventually to zero), and the underlying motion model will become a poor fit to the real noisy data (eventually causing incoherent motion even in a single object).

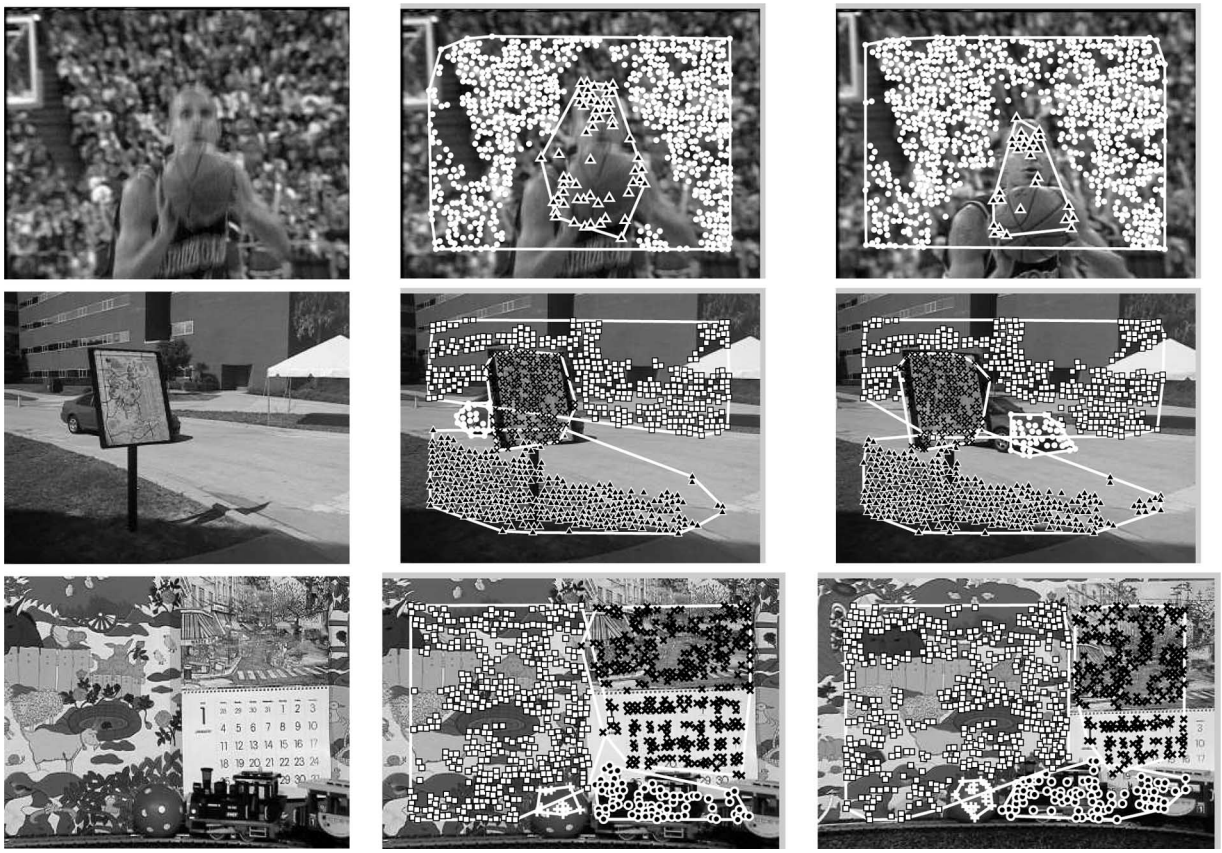

Fig. 6. Results of the algorithm on three image sequences: (top) free-throw, (middle) car map, and (bottom) mobile calendar. (Left) The original image, (middle) the feature groups overlaid on the image, and (right) the feature groups detected on another image later in the sequence. Features belonging to different groups are indicated by markers of different shapes, and solid lines outline the convex hull of each group. The top row shows frames 9 and 14, the middle shows frames 11 and 20, and the bottom shows frames 14 and 69.

On the other hand, if the reference frame is updated at a constant rate, as is commonly done, then the differential nature of motion is being ignored, and the result will depend upon object speed.

EM cannot solve this dilemma. Maximizing (3) with respect to r_j, $j = 1, \ldots, k$ would yield the trivial solution of setting the reference frame to the current frame, just as maximizing the equation with respect to k would yield the trivial solution of producing exactly one group per feature. Just as EM requires k to be fixed, so it also requires r_j to be fixed for all j's. As a result, we are forced to turn to an *ad hoc* technique, in much the same way that others have resorted to suboptimal methods for determining the number of groups [37], [41].

To solve the dilemma, we then turn to the χ^2 test. This nonparametric statistical test compares observed data with an expected probability distribution in order to decide whether to reject the null hypothesis H_0 that the data were drawn from the distribution. The test is asymmetric: Although a large χ^2 value indicates that H_0 should be rejected, a small value says nothing about whether H_0 should be accepted, but only that insufficient evidence exists to reject it. The test is therefore a natural fit to the problem of motion segmentation, in which one can never conclude, based on low-level image motion alone, that features belong to the same object. Instead, either the features belong to different objects with high probability, or there is insufficient evidence in the data to conclude that they belong to different objects.

To apply the χ^2 test, we compute a distribution of the residues of all the features in a group, using the motion model of the group. The distribution is quantized into five bins, with each of them having a width of $0.3\sigma_d$, where σ_d is the standard deviation of the distribution. We reject the assumption that the motion of the group is coherent if $\chi^2 = \sum_{i=1}^{n}(O_i - E_i)^2/E_i > \chi^2_{\alpha;k}$, where O_i is the observed frequency for bin i, E_i is the expected frequency for bin i, and $\chi^2_{\alpha;k}$ is the critical threshold for a χ^2 distribution with k degrees of freedom and significance level α. We use $\alpha = 0.01\%$ and $k = 4$.

Initially, we planned to compute the observed distribution using the current and reference frames and to use a zero-mean unit-variance Gaussian for the expected distribution, i.e., a group would not be split if its residues follow a Gaussian distribution. However, we found this approach to fail due to the sparse distribution sampling (only five bins) and the variable interframe spacing, which together cause single-object distributions to be non-Gaussian. Instead, we have adopted an approach in which the expected distribution is generated from the motion residues using the reference frame r_j, and the observed distribution is generated using the frame $\texttt{round}(t - \beta_e(t - r_j))$, where $0 < \beta_e < 1$. This method allows the distribution to adapt to the changing characteristics of individual objects over time.

The features in a group are dynamically adjusted over time as features are lost due to the feature tracking and as new features are added by assimilation. At each frame, the χ^2 test is applied

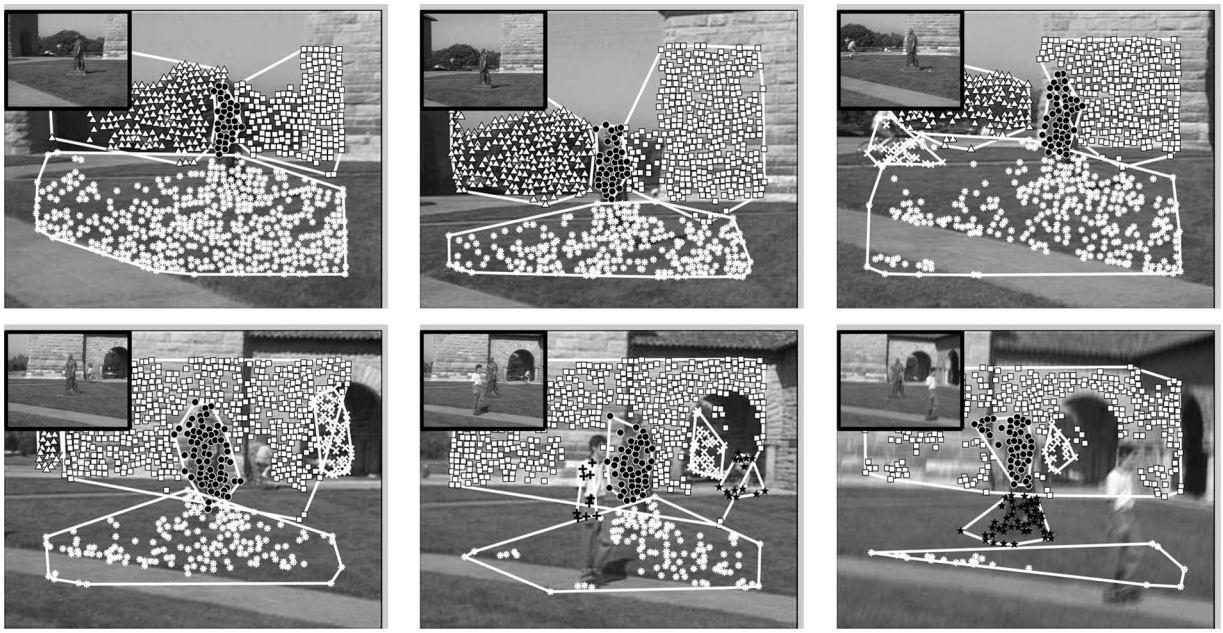

Fig. 7. Results on the statue sequence, with the original image being shown in the upper left inset. In lexicographic order, the image frames are 6, 64, 185, 395, 480, and 520. The algorithm forms new groups or splits existing groups due to the arrival or departure of entities in the scene.

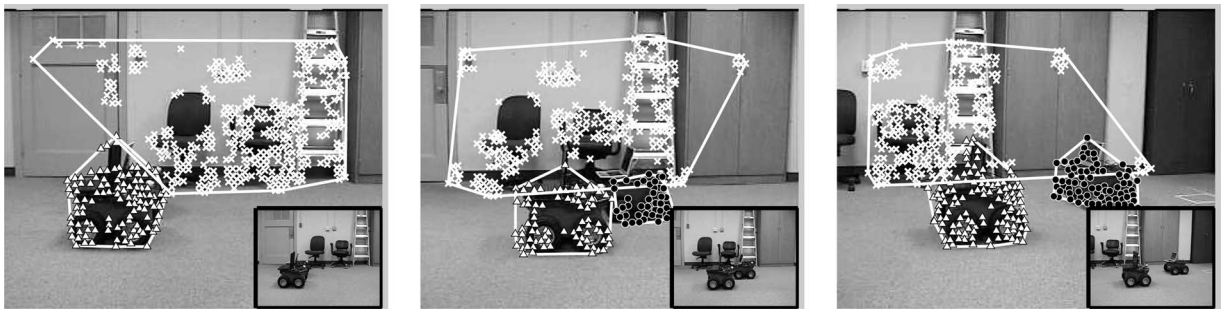

Fig. 8. Results on the robot sequence (frames 35, 120, and 100), with the original image being shown in the bottom-right inset. The algorithm splits the group belonging to the robots into two separate groups as the farther robot accelerates.

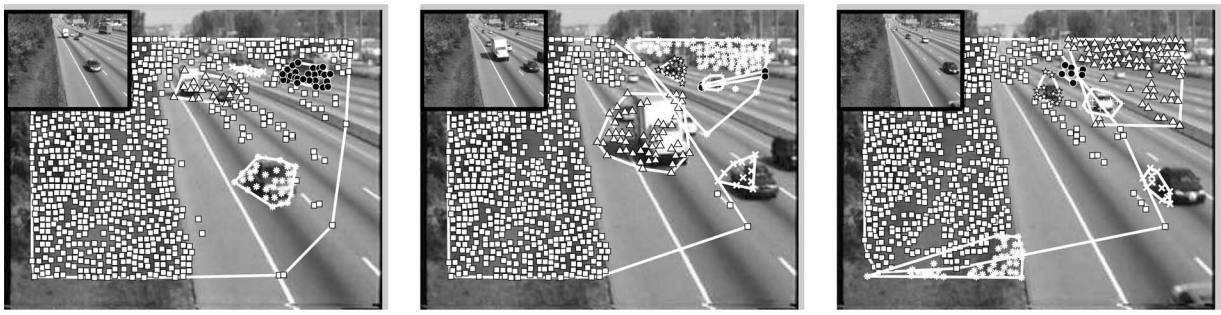

Fig. 9. Results on the highway sequence (frames 15, 39, and 61), with the original image being shown in the top-left inset. The algorithm forms new groups or splits existing groups due to the arrival or departure of vehicles in the scene.

to the features in the group. If the test fails, then the features are regrouped by using the initialization procedure mentioned in the previous section. This computation results in either the group splitting into multiple groups due to the presence of multiple objects or in causing the outlier features to be discarded from the group. Once a split has been attempted for a group, the reference frame is updated to the frame $\texttt{round}(t - \beta_r(t - r_j))$, where $0 < \beta_r < 1$. In our implementation, we set $\beta_e = 0.1$ and $\beta_r = 0.25$. The procedure is shown in Fig. 5.

VI. Experimental Results

The algorithm was tested on a total of six grayscale image sequences. Motion segmentation results for three of these sequences are shown in Fig. 6, with features assigned to the group with the highest weight.[1] In the free-throw sequence, a

[1]Videos of the results can be found at http://www.ces.clemson.edu/~stb/research/ motion_segmentation.

basketball player moves down in the image as he prepares to shoot a free-throw, whereas the camera moves slightly down. Two groups are found by the algorithm, one for the player (indicated by black triangles) and one for the crowd in the background (indicated by white circles). In the car map sequence, a car drives in a straight line behind a map, whereas the camera remains stationary. The car (white circles), map (black x's), ground (black triangles), and background (white squares) are detected. The car is occluded for a period of time behind the map and is then detected again as it reappears on the other side. In the mobile calendar sequence, a toy train pushes a ball to the left, and a calendar slides down in front of a textured background, whereas the camera zooms out and moves slightly left. All of the objects are detected, even though the ball (white +'s) and train (black circles) move faster than the calendar (black x's) and background (white squares). It should be noted that the white borders around the feature groups are shown only for the sake of clarity and are not to be considered the object boundaries.

The statue sequence, shown in Fig. 7, is the most challenging one. These images were captured by a hand-held camera moving in an uncontrolled fashion around a statue, whereas a bicyclist drove behind the statue, and a pedestrian walked 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. 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 6, four groups are found: the statue (black circles), the grass (white asterisks), the trees (white triangles), and the stone wall (white squares). Although some of the trees are inadvertently grouped with the stone wall initially, over time, they are correctly joined with the rest of the trees as more evidence becomes available. The bicyclist enters in frame 151, is detected in frame 185 (white x's), becomes occluded by the statue in frame 312, emerges on the other side of the statue in frame 356, and is detected again in frame 444 (black stars). Although the algorithm currently does not attempt correspondence between occluded and disoccluded objects, a straightforward extension would maintain the identity of the bicyclist through the occlusion. The pedestrian enters the scene in frame 444 and is segmented successfully (black +'s), although the nonrigid motion prevents the feature tracker from maintaining a large number of features throughout, and it prevents the affine motion model from well approximating the actual motion. The pedestrian occludes the statue from frames 486 to 501, after which the statue is regrouped into separate groups for top and bottom. Near the end of the sequence, the lack of texture on the ground, combined with motion blur of the shaking camera, prevents the feature tracker from replenishing the features on the grass after the pedestrian passes.

Results for the robot sequence are shown in Fig. 8. In this sequence, two robots move in the same direction roughly parallel to the plane of the camera, although there is a significant pan of the camera toward the end of the sequence. The robots start from the same initial location and travel together at the same speed for several seconds, after which the robot that is farther from the camera accelerates and overtakes the other robot. As seen in the figure, the group belonging to the robots splits into two groups (one per robot) when their relative speeds change, whereas the background is maintained as a single group throughout.

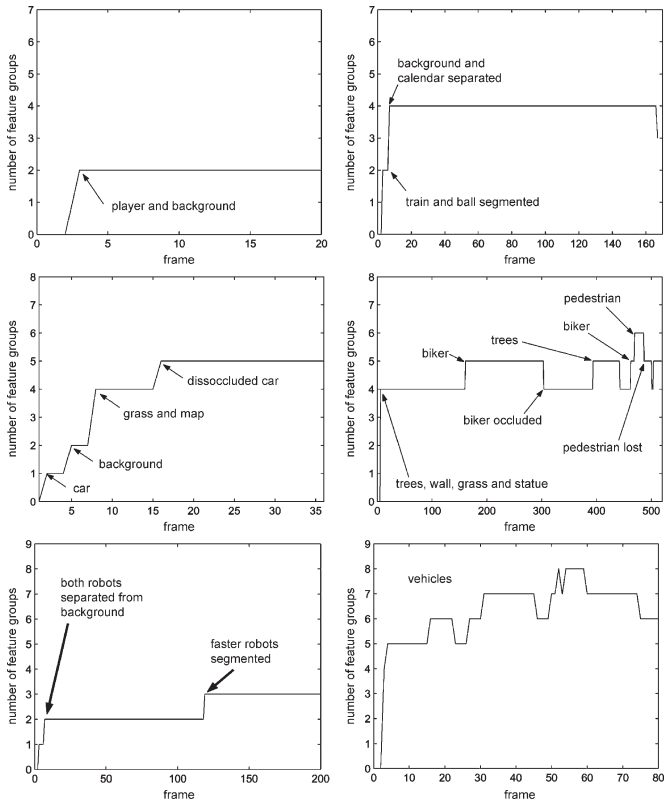

Fig. 10. Algorithm automatically and dynamically determines the number of feature groups. Plotted are the numbers of groups versus image frames for each of the six sequences: free-throw, mobile calendar, car map, statue, robot, and vehicle, in lexicographic order.

Fig. 9 shows a highway scene captured from a low-angle camera. Fourteen vehicles enter and exit the scene during the 90 frames of the sequence. Of the ten vehicles in the three nearby lanes (approaching traffic), 80% of the vehicles were segmented from the background correctly. The two vehicles in the nearby lanes that were not detected were close to adjacent vehicles traveling at the same speed (see the car behind the truck in the middle image). In addition, the algorithm segmented four vehicles in the far lanes (receding traffic), even though their image size is small (on an average of approximately 50 pixels). The background is split into two large regions in the middle image of the figure because the vehicle traffic removes the adjacency of the background features in that portion of the image. Also, the grass on the left side of the image is further split from the trees due to movement of the latter.

Because 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 and dynamically. Fig. 10 shows the dynamic progress of the results on all of the six sequences (free-throw, mobile calendar, car map, statue, robot, and vehicle). In the first sequence, the basketball player becomes separable from the background almost immediately. In the second sequence, the faster train

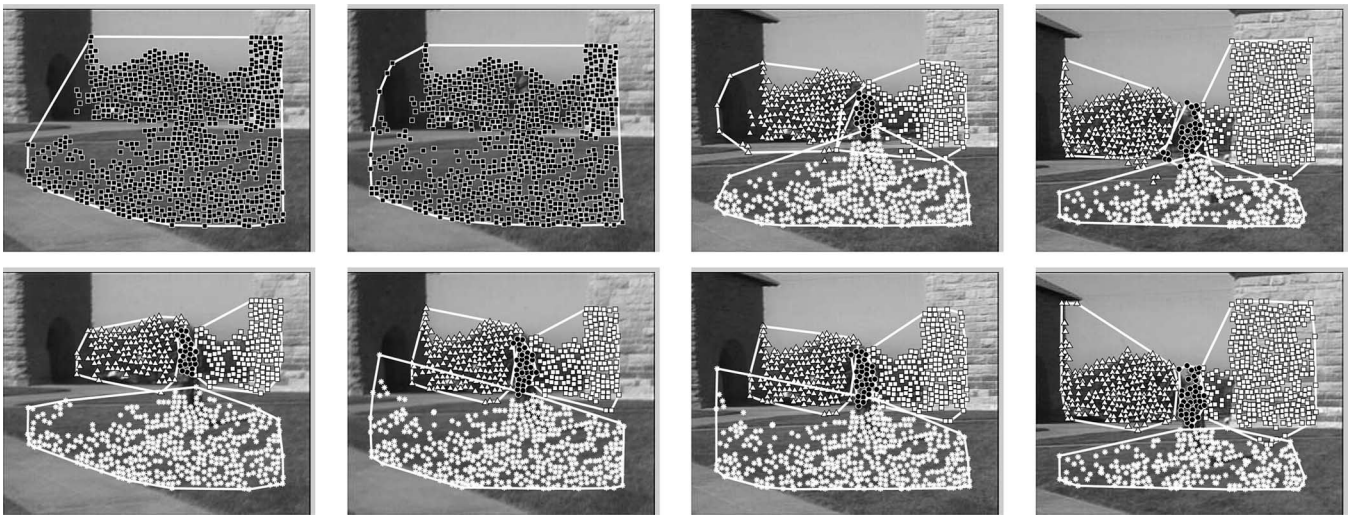

Fig. 11. Insensitivity to parameters. Segmentation results shown for two different values of τ for (from left to right) frames 4, 8, 12, and 64 of the statue sequence. (Top) $\tau = 3.0$. (Bottom) $\tau = 0.7$.

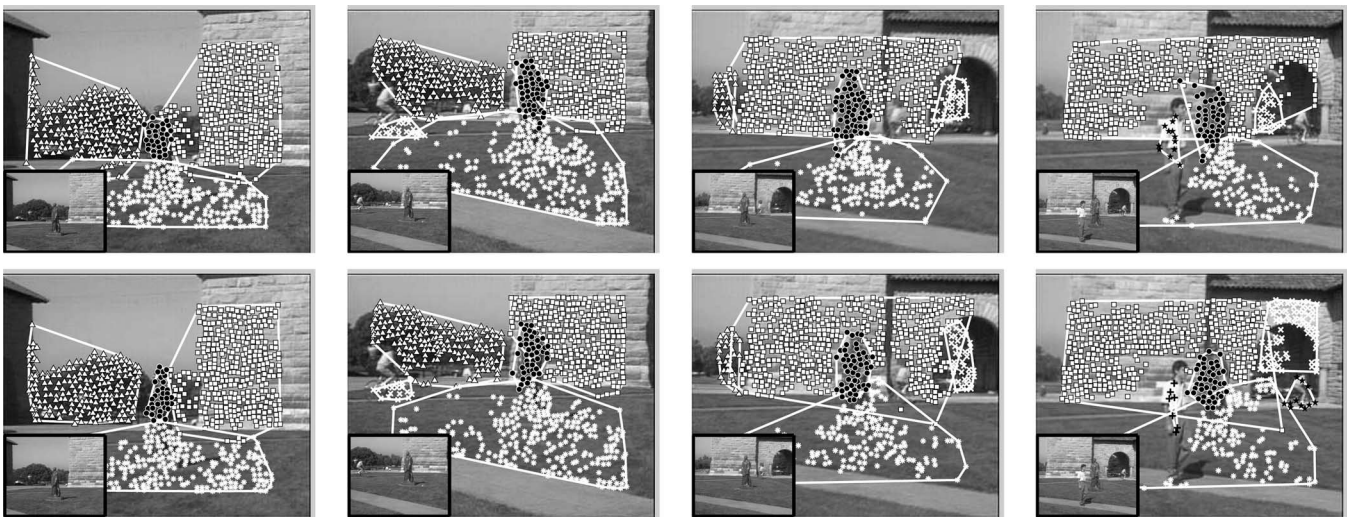

Fig. 12. Algorithm is insensitive to speed. (Top) Results on a modified statue sequence in which each frame occurs twice, thus reducing the motion by half. (Bottom) Results on a modified statue sequence in which every other frame has been discarded, thus doubling the motion. Shown are frames 64, 185, 395, and 480 of the original sequence.

and ball become separable after only two frames, whereas six frames are needed to separate the calendar and background. In the third sequence, the objects are detected one at a time, with all four objects being segmented by frame 16. In the statue sequence, the four primary areas of the scene are segmented after just a few frames. Then the bicyclist and pedestrian are detected as they enter the scene and are removed as they leave. In the robot sequence, the moving robots are separated from the background, and after a moment, the faster robot is separated from the slower one. Finally, in the vehicle sequence, a large number of vehicles appear and disappear throughout the length of the sequence.

One of the advantages of this algorithm is its lack of parameters. The 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. It is used to compute $p_\tau = (1/\sqrt{2\sigma_f^2})\exp\{-(\tau^2/2\sigma_f^2)\}$ for (8), where $\sigma_f = 0.7$. Significantly, the results are insensitive to this parameter: If τ is increased, then the algorithm simply waits longer before declaring a group by accumulating the motion difference between the objects over time, whereas if τ is decreased, then the groups are declared sooner. Fig. 11 shows this insensitivity. Similar experiments reveal the insensitivity of the results to the other parameters, such as β_e, β_r, and $n_{\min}$.

Insensitivity to speed is shown in Fig. 12. Qualitatively similar results are obtained by running the algorithm on the original statue sequence and on a sequence generated by replicating each frame in the sequence (thus effectively decreasing the relative speed of the objects by half). Although not shown due to lack of space, the same result occurs by further replication (i.e., reducing the speed by any positive factor). Similarly, nearly identical results are obtained by running the algorithm on every other image of the sequence (thus doubling the motions). All these results were obtained without changing any parameters of the algorithm.

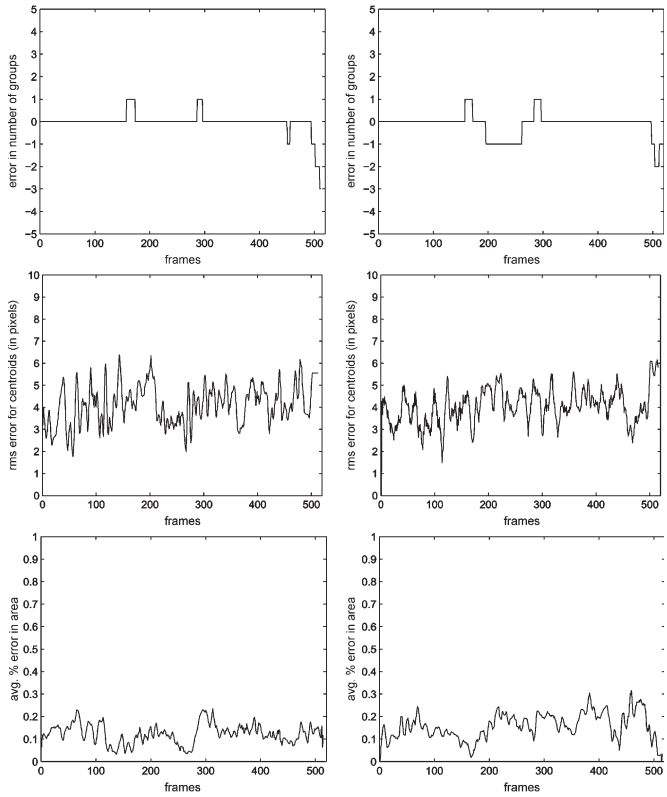

Fig. 13. Quantitative analysis of the insensitivity of the algorithm to speed for (left) the upsampled (slower) sequence and (right) the downsampled (faster) sequence. The plots compare the original and modified sequences using (top) the number of groups detected, the (middle) root-mean-square error of the centroids, and (bottom) the average percentage difference between the areas of the corresponding groups.

Quantitative results are shown in Fig. 13 for these down- and upsampled statue sequences. Except for the end of the sequence, where the errors in the feature tracking cause mismatch in the groups detected, the maximum error in the number of groups found is one. These spikes, near frames 160 and 300, occur due to the late detection and early loss of the bicyclist, thus indicating a mere temporal misalignment error from which the algorithm recovers. The difference in the centroids of the groups is small, averaging 4 pixels over the entire sequence and never exceeding 6.5 pixels. Similarly, the average errors in the areas of the groups, computed by the convex hull of the features in each group, are 12% and 15% for the up- and downsampled sequences, respectively. These errors are relatively small, keeping in mind that the sparse algorithm is not designed to recover accurate shape of the objects and, thus, is subject to artifacts of feature tracking and density. Moreover, the errors do not increase with further upsampling.

Fig. 14 shows the updating of the reference frame over time for two feature groups in the statue sequence: the statue itself and the trees behind the statue. Because the tree group is large and contains nonplanar surfaces in the real world, it contains a fair number of outliers. These outliers cause the χ^2 test for that group to fail often, thus necessitating the reference frame to be updated frequently. Other groups in the sequence, such as the grass and the wall, exhibit similar behavior. In contrast, the small and stable statue group requires only infrequent updating

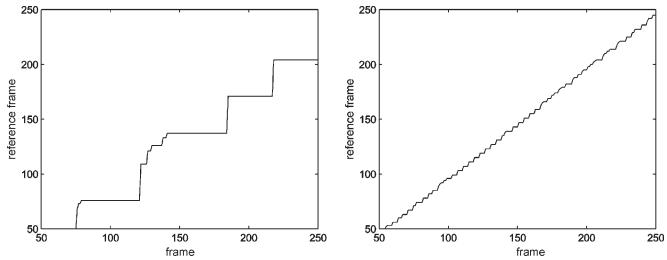

Fig. 14. Reference frame versus time for two groups in the statue sequence. (Left) The statue. (Right) The trees behind the statue.

TABLE I
COMPARISON OF THE COMPUTATIONAL TIME OF VARIOUS MOTION SEGMENTATION ALGORITHMS. THE RIGHTMOST COLUMN INDICATES THE MAXIMUM NUMBER OF GROUPS FOUND BY EACH ALGORITHM IN THE REPORTED RESULTS

Algorithm	Run time (sec / frame)	Number of groups
Xiao and Shah [46]	520	4
Kumar et al. [23]	500	6
Smith et al. [34]	180	3
Rothganger et al. [29]	30	3
Jojic and Frey [18]	1	3
Cremers and Soatto [8]	40	4
our algorithm	0.16	6

of the reference frame. Even though the statue is not planar, its extent allows the affine model to approximate its motion well.

VII. COMPARISON WITH OTHER APPROACHES

In terms of computation, our algorithm is orders of magnitude faster than other recent techniques, as shown in Table I. The algorithm requires only 160 ms per frame for a sequence of 320×240 images with 1000 features on a 2.8-GHz Pentium IV computer using an unoptimized Visual C++ implementation with the use of the Kanade–Lucas–Tomasi feature tracker [3] within the Blepo library [4]. Most of this computation (140 ms) is used by the feature tracking, with only 20 ms needed by the segmentation algorithm. In [46], 95% of the computation is spent on the preprocessing stage to determine the number of groups along with their motion models, which is what our algorithm produces. Thus, our approach can be seen as a computationally efficient front end for initializing one of these more expensive dense segmentation methods in order to drastically reduce their computational load.

It is difficult to compare the quality of our segmentation with those of other algorithms because the goals are different. As an example, Fig. 15 shows the groups found by the algorithm of Kumar *et al.* [23] by batch processing small clips from three of the sequences. Because the algorithm assumes that objects move parallel to the image plane, it performs well when that assumption holds, enabling a crisp delineation of the regions on these clips. However, even on the short clip of the statue sequence, their algorithm fails to separate the trees on the left from the wall on the right, and it erroneously merges much of the grass with the tree/wall region. More importantly, the algorithm cannot process the entire video sequence, both because of its computational cost and of the assumptions that it makes regarding the presence and motion of objects. In a similar manner, the algorithm does not perform as

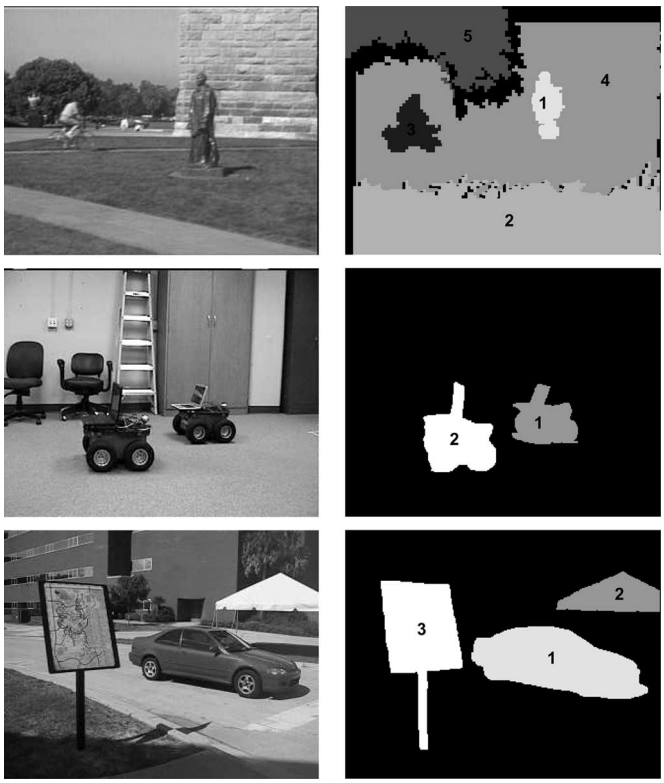

Fig. 15. Segmentation results of [23] on portions of the statue, robot, and car map sequences. The algorithm processed frames 161–196, 150–175, and 25–35, respectively. Shown are (left) a sample image from each sequence and (right) the results for that image.

favorably on the other sequences (e.g., mobile calendar, freethrow, and vehicles) because of the large rotations and the appearance/disappearance of objects.

Although other algorithms exhibit strengths according to the goals for which they were designed, they perform less favorably on our sequences. For example, the technique of Jojic and Frey [18] requires a static background, so it is unable to properly segment these sequences in which the camera moves considerably. The hard limit of the algorithm of Smith *et al.* [34] to a maximum of three regions would also prevent its obtaining a proper segmentation. Similarly, Cremers and Soatto [8] detect up to four synthetic regions using the intersection of two contours, an approach that is unlikely to generalize to the complexity of sequences containing multiple independently moving objects. Moreover, their approach handles just two image frames and requires the contours to be initialized, which is not possible in the context of online automatic segmentation of live video. Similarly, the approach of Xiao and Shah [46] computes accurate dense motion layers, but it detects the number of layers initially and keeps this number constant throughout the sequence. Finally, Rothganger *et al.* [29] group sparse feature points by processing a small block of image frames in batch.

VIII. Conclusion

We have addressed the problem of motion segmentation in a manner that takes into consideration the differential nature of motion. As a result, our approach processes image sequences incrementally and segments objects that move at different speeds. The algorithm groups sparse features using a spatially constrained EM approach that models the interactions of neighboring features with a Markov assumption. A region-growing algorithm with a novel consistency check is used to efficiently initialize the EM algorithm in a greedy manner. 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 detects a relatively large number of objects and automatically and dynamically determines the number of objects in the scene, as well as their motions. A χ^2 goodness-of-fit test is used to adaptively update the reference frame by distinguishing between multiple motions within a group and an obsolete reference frame. The algorithm operates in real time and produces accurate estimations on challenging sequences.

There are many ways to improve upon this work. An obvious application of the algorithm is to serve as a front end for detecting dense object boundaries and motion discontinuities in live video, with the boundaries being refined using dense pixel motion, texture, intensity gradients, and/or color. Various aspects of the feature tracking could be improved to better handle nonrigid objects, periodic motion, occlusion, and low-texture areas. Another enhancement would be to generate a hierarchical representation of motion segmentation that allows regions of the image that move differently but share a common relationship, such as articulated objects, to be accurately modeled. Moreover, combining the nontextured regions, the sparse segmentation, and the motion discontinuities and contours would yield a novel representation of video.

Acknowledgment

The authors would like to thank P. Kumar for the assistance in evaluating the algorithm and Z. Chen for the help in capturing the robot sequence.

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 *Proc. 5th Int. Conf. Comput. Vis.*, Jun. 1995, pp. 777–784.

[2] A. Barbu and S. C. Zhu, "On the relationship between image and motion segmentation," in *Proc. SCMVA Workshop (in Conjunction With ECCV)*, 2004, pp. 51–63.

[3] S. Birchfield, *KLT: An implementation of the Kanade–Lucas–Tomasi feature tracker*. [Online]. Available: http://www.ces.clemson.edu/~stb/klt/

[4] S. Birchfield, *Blepo Computer Vision Library*. [Online]. Available: http://www.ces.clemson.edu/~stb/blepo/

[5] T. Brox, A. Bruhn, and J. Weickert, "Variational motion segmentation with level sets," in *Proc. Eur. Conf. Comput. Vis.*, May 2006, pp. 471–483.

[6] V. Cheung, B. Frey, and N. Jojic, "Video epitomes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2005, pp. 42–49.

[7] J. Costeira and T. Kanade, "A multi-body factorization method for motion analysis," in *Proc. Int. Conf. Comput. Vis.*, 1995, pp. 1071–1076.

[8] D. Cremers and S. Soatto, "Motion competition: A variational approach to piecewise parametric motion," *Int. J. Comput. Vis.*, vol. 62, no. 3, pp. 249–265, May 2005.

[9] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov, "Bilayer segmentation of live video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2006, vol. 1, pp. 53–60.

[10] A. Diplaros, N. Vlassis, and T. Gevers, "A spatially constrained generative model and an EM algorithm for image segmentation," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 798–808, May 2007.

[11] R. Dupont, O. Juan, and R. Keriven, "Robust segmentation of hidden layers in video sequences," in *Proc. IAPR Int. Conf. Pattern Recog.*, 2006, vol. 3, pp. 75–78.
[12] P. Favaro and S. Soatto, "A variational approach to scene reconstruction and image segmentation from motion blur cues," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2004, vol. 1, pp. 631–637.
[13] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Englewood Cliffs, NJ: Prentice-Hall, 2003.
[14] P. Fua, "Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities," in *Proc. 12th Int. Joint Conf. Artif. Intell.*, 1991, pp. 1292–1298.
[15] A. Gruber and Y. Weiss, "Incorporating non-motion cues into 3D motion segmentation," in *Proc. Eur. Conf. Comput. Vis.*, May 2006, pp. 84–97.
[16] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 809–830, Aug. 2000.
[17] M. Irani and P. Anandan, "A unified approach to moving object detection in 2D and 3D scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 6, pp. 577–589, Jun. 1998.
[18] N. Jojic and B. J. Frey, "Learning flexible sprites in video layers," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2001, pp. 199–206.
[19] N. Jojic, J. Winn, and L. Zitnick, "Escaping local minima through hierarchical model selection: Automatic object discovery, segmentation, and tracking in video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2006, vol. 1, pp. 117–124.
[20] B. Jung and G. S. Sukhatme, "Detecting moving objects using a single camera on a mobile robot in an outdoor environment," in *Proc. 8th Conf. Intell. Auton. Syst.*, 2004, pp. 980–987.
[21] Q. Ke and T. Kanade, "A subspace approach to layer extraction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2001, vol. 1, pp. 255–262.
[22] I. Kokkinos and P. Maragos, "An expectation maximization approach to the synergy between image segmentation and object categorization," in *Proc. Int. Conf. Comput. Vis.*, Oct. 2005, vol. 1, pp. 617–624.
[23] M. P. Kumar, P. H. S. Torr, and A. Zisserman, "Learning layered motion segmentations of video," in *Proc. Int. Conf. Comput. Vis.*, Oct. 2005, vol. 1, pp. 33–40.
[24] I. Laptev, S. J. Belongie, P. Pérez, and J. Wills, "Periodic motion detection and segmentation via approximate sequence alignment," in *Proc. Int. Conf. Comput. Vis.*, Oct. 2005, vol. 1, pp. 816–823.
[25] A. Levin and Y. Weiss, "Learning to combine bottom-up and top-down segmentation," in *Proc. Eur. Conf. Comput. Vis.*, May 2006, pp. 581–594.
[26] R. M. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*, M. I. Jordan, Ed. Norwell, MA: Kluwer, 1998.
[27] A. S. Ogale, C. Fermüller, and Y. Aloimonos, "Motion segmentation using occlusions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 988–992, Jun. 2005.
[28] S. Pundlik and S. T. Birchfield, "Motion segmentation at any speed," in *Proc. BMVC*, Sep. 2006, pp. 427–436.
[29] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "Segmenting, modeling, and matching video clips containing multiple moving objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2004, pp. 477–491.
[30] H. S. Sawhney, Y. Guo, and R. Kumar, "Independent motion detection in 3D scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, pp. 1191–1199, Oct. 2000.
[31] J. Shi and J. Malik, "Motion segmentation and tracking using normalized cuts," in *Proc. 6th Int. Conf. Comput. Vis.*, 1998, pp. 1154–1160.
[32] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[33] J. Sivic, F. Schaffalitzky, and A. Zisserman, "Object level grouping for video shots," in *Proc. Eur. Conf. Comput. Vis.*, 2004, vol. 2, pp. 85–98.
[34] P. Smith, T. Drummond, and R. Cipolla, "Layered motion segmentation and depth ordering by tracking edges," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 4, pp. 479–494, Apr. 2004.
[35] M. Toussaint, V. Willert, J. Eggert, and E. Körner, "Motion segmentation using inference in dynamic Bayesian networks," in *Proc. Brit. Mach. Vis. Conf.*, 2007, pp. 12–21.
[36] Z. Tu, X. Chen, A. L. Yuille, and S.-C. Zhu, "Image parsing: Unifying segmentation, detection, and recognition," *Int. J. Comput. Vis.*, vol. 63, no. 2, pp. 113–140, Jul. 2005.
[37] J. J. Verbeek, N. Vlassis, and B. Kröse, "Efficient greedy learning of Gaussian mixture models," *Neural Comput.*, vol. 15, no. 2, pp. 469–485, Feb. 2003.
[38] R. Vidal and S. Sastry, "Optimal segmentation of dynamic scenes from two perspective views," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2003, vol. 2, pp. 281–286.
[39] R. Vidal and D. Singaraju, "A closed form solution to direct motion segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2005, pp. 510–515.
[40] P. A. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *Int. J. Comput. Vis.*, vol. 63, no. 2, pp. 153–161, Jul. 2005.
[41] N. Vlassis and A. Likas, "A greedy EM algorithm for Gaussian mixture learning," *Neural Process. Lett.*, vol. 15, no. 1, pp. 77–87, Feb. 2002.
[42] J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 625–638, Sep. 1994.
[43] Y. Weiss and E. H. Adelson, "A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 1996, pp. 321–326.
[44] J. Wills, S. Agarwal, and S. Belongie, "What went where," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2003, vol. 1, pp. 37–44.
[45] J. Xiao and M. Shah, "Accurate motion layer segmentation and matting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2005, pp. 698–703.
[46] J. Xiao and M. Shah, "Motion layer extraction in the presence of occlusion using graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1644–1659, Oct. 2005.
[47] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," in *Proc. Eur. Conf. Comput. Vis.*, May 2006, pp. 94–106.
[48] L. Zelnik-Manor, M. Machline, and M. Irani, "Multi-body factorization with uncertainty: Revisiting motion consistency," *Int. J. Comput. Vis.*, vol. 68, no. 1, pp. 27–41, 2006.

Shrinivas J. Pundlik received the B.E. degree in electronics engineering from the University of Pune, Pune, India, in 2002, and the M.S. degree in electrical engineering from Clemson University, Clemson, SC, in 2005, where he is currently working toward the Ph.D. degree in electrical engineering in the Department of Electrical and Computer Engineering.

His research interests include image and motion segmentation, human motion analysis, and biometrics.

Stanley T. Birchfield (S'91–M'99–SM'06) received the B.S. degree in electrical engineering from Clemson University, Clemson, SC, in 1993, and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, in 1996 and 1999, respectively.

While at Stanford, his research was supported by a National Science Foundation Graduate Research Fellowship, and he was part of the winning team of the 1994 AAAI Mobile Robotics Competition. From 1999 to 2003, he was a Research Engineer with Quindi Corporation, a start-up company in Palo Alto, CA. Since 2003, he has been an Assistant Professor with the Department of Electrical and Computer Engineering, Clemson University, where his research interests include visual correspondence, tracking, and segmentation, particularly applied to real-time systems.