Motion Segmentation and Tracking Using Normalized Cuts

Jianbo Shi and Jitendra Malik
Computer Science Division
University of California at Berkeley, Berkeley, CA 94720
{jshi,malik}@cs.berkeley.edu

Abstract

We propose a motion segmentation algorithm that aims to break a scene into its most prominent moving groups. A weighted graph is constructed on the image sequence by connecting pixels that are in the spatiotemporal neighborhood of each other. At each pixel, we define motion profile vectors which capture the probability distribution of the image velocity. The distance between motion profiles is used to assign a weight on the graph edges. Using **normalized cuts** *we find the most salient partitions of the spatiotemporal graph formed by the image sequence. For segmenting long image sequences, we have developed a recursive update procedure that incorporates knowledge of segmentation in previous frames for efficiently finding the group correspondence in the new frame.*

1 Introduction

Grouping based on common motion, or what the Gestaltists[23] called the factor of "Common Fate", is one of the strongest cues for segmenting an image sequence into separate objects. However, implementing this perceptual capability has proved to be very challenging for computer vision systems. Early approaches were based on trying to estimate optical flow first, and then looking for discontinuities. This proved difficult because of a number of reasons

1. Optical flow measurement is difficult in areas of little texture or primarily one-dimensional texture. Any real image is bound to have large regions with these properties.
2. To deal with difficulty (1) enforcing smoothness constraints to interpolate in the flow field were proposed. However this raises the requirement that one must first know the segmentation so as to avoid smoothing across motion discontinuities!

To cope with these problems, over the last few years a new framework has appeared based on the idea of simultaneous estimation of multiple global motion models and their spatial supports (so-called "layers"). This idea has evolved through a number of papers [3, 22, 6, 20, 11, 12]. Perhaps the cleanest current formulations are based on using the Expectation-Maximization (EM) algorithm[7]. Typically the motion models are 2D parametric models, translational, affine or projective, the E-step is used to solve for the layers given the motions, and the M-step for solving for the motions given the layers.

EM approaches offer a number of advantages over the previous approaches based on initial local measurement of optical flow. By combining information over large regions of the images, the motion estimates found are considerably more robust. Video data can be quite noisy because of camera jitter and repeated occlusion and discocclusion events, and the global analysis provides a way to overcome these difficulties. The layers that are extracted provide the desired scene segmentation. On the other hand, the assumption that image sequence have to follow a global rigid planar motion is clearly too restrictive, and recently Weiss[21] has developed a variation of EM approach that is based on a non-parametric mixture model using a probability distribution over flow fields that favors smooth flow fields. When only sparse point correspondences are sought, Torr and Murray[19] have developed an alternative approach based on characterizing rigid motions using Fundamental matrices.

In our opinion, the principal weakness of the EM approach to layered motion segmentation is in the initialization phase. How many models should one initialize and where and what should they be, and how can one ensure a global optimal solution have been reached? A representative approach due to Ayer and Sawhney[3] uses the Minimum Description length principle for selecting the number of models. Initialization is done by dividing up the image into a fixed number of tiles; estimating the initial motion parameters in these tiles and then using these as the initial conditions for the EM algorithm. Our experience however has shown that finding a good initialization remains a nagging problem. Undoubtedly, further research in this area will provide improvements in this area; however in this paper we have chosen to develop an alternative approach that incorporates motion information across spatial and temporal neighborhoods and searches for a globally optimal segmentation solution without the difficulty of initialization.

We consider motion segmentation as a special instance of a more general grouping problem. Each pixel in the image sequence is treated as a point living in a large feature space. The features correspond to its

spatiotemporal position, color and motion, etc. The question then is – Given these points in this large feature space, what is the best way of partitioning them? We believe that image partitioning should be done at the 'big picture' level, rather like a painter first marking out the major areas and then filling in the details.

This idea can be formalized using a graph partitioning criterion called *normalized cut*[16]. Given a motion sequence, a weighted graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is constructed by taking each pixel as a node, and connecting nodes that are in a spatiotemporal neighborhood of each other. The weight on each graph edge $w(\mathbf{i}, \mathbf{j})$ is a function of the similarity between pair of nodes $\mathbf{i}$ and $\mathbf{j}$. Similarity can be estimated on the basis of any of a number of features – color, brightness, texture, motion, disparity etc. By connecting each node to other nodes in its spatiotemporal neighborhood, we provide an effective way for the similarity information to be integrated over space and time, thus increasing the robustness of the segmentation, without imposing any explicit global motion constraint.

The results in this paper are based on using only motion information so as to permit a fair comparison with alternative motion segmentation schemes. For measuring motion similarity, we will define a motion feature vector at each pixel called *motion profile*. Each *motion profile* is the probability of different displacement of at each point in the image, which captures not only the direction of the motion, but also the uncertainty associated with it.

Once the weighted graph is constructed, the *normalized cut* criterion is used to recursively partition the graph. As shown in [16], *normalized cut* is a global measure which reflects both the similarity within the partitions, as well as dissimilarity across the partitions. Furthermore, this criterion can be computed efficiently by solving a generalized eigenvalue system.

Successful partition of G gives us spatiotemporal volumes corresponding to different moving objects. By taking time slices of such a volume we can indicate image groups in each frame as well as identify what their corresponding groups are across time. This notion of *group correspondence* is very useful because it leads to a measure of motion which applies to the group as a whole, not individual pixels as in the case of optical flow. Such a measure is considerably more robust and could be used for estimating gross measures such as divergence, deformation, rotation which have been shown to be useful variables for visual guidance of locomotion and manipulation [13, 5]. Snakes have been used in the computer vision literature[5] previously for this purpose. They are computationally efficient but difficult to initialize.

The paper is organized as follows. Section 2 briefly explains our grouping algorithm based on the *normalized cut* graph partitioning criterion. This section follows our previous work for static image segmentation described in[16]. Section 3 describes the motion profile feature vector, and the development of motion segmentation using *normalized cuts* follows in section 4. Section 5 shows how we can efficiently segment long image sequences. We conclude in section 6.

2 Normalized Cuts

In our previous paper[16], we have developed a grouping algorithm based on minimizing a graph partitioning criteria called *normalized cut*.

Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a weighted graph. Graph $\mathbf{G}$ can be partitioned into two disjoint sets, A, B, $A \cup B = \mathbf{V}$, $A \cap B = \emptyset$, by simply removing edges connecting the two parts. The total weights on those edges removed reflect the degree of disassociation between the sets A and B. In graph theoretic language, it is called the *cut*:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v). \quad (1)$$

Although there are efficient computational algorithm for finding partitions that minimizes the *cut* value, this criterion favors partitions which have small sizes[16]. To fix this undesired bias, we will define *normalized cut* (*Ncut*) as unbiased measure of dissassociation:

$$Ncut(A, B) = \frac{cut(A, B)}{asso(A, V)} + \frac{cut(A, B)}{asso(B, V)}, \quad (2)$$

where $asso(A, V) = \sum_{u \in A, t \in V} w(u, t)$ is the total connection from nodes in A to all nodes in the graph, and $asso(B, V)$ is similarly defined. We can also define a unbiased measure for total association within groups for a given partition: $Nasso(A, B) = \frac{asso(A,A)}{asso(A,V)} + \frac{asso(B,B)}{asso(B,V)}$, where $asso(A, A)$ and $asso(B, B)$ are total weights of edges connecting nodes within A and B respectively. A simple calculation shows $Ncut(A, B) = 2 - Nasso(A, B)$. Hence minimizing the disassociation between the groups and maximizing the association within the group, can be satisfied simultaneously.

Let $\mathbf{W}$ be the graph weight matrix, and $\mathbf{D}$ be the diagonal matrix with $\mathbf{D}(i, i) = \sum_j \mathbf{W}(i, j)$. In [16], we showed that minimizing $Ncut$ can be reduced to minimizing a Rayleigh quotient:

$$min.\, Ncut = min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{y}}{\boldsymbol{y}^T \boldsymbol{D} \boldsymbol{y}}, \quad (3)$$

with the condition $\boldsymbol{y}_i \in \{1, -b\}$ and $\boldsymbol{y}^T \boldsymbol{D} \mathbf{1} = 0$. By relaxing $\boldsymbol{y}$ to take on real values, we can minimize equation (3) with its constraint by solving for the second smallest eigenvector of the generalized eigenvalue system,

$$(\mathbf{D} - \mathbf{W})\boldsymbol{y} = \lambda \mathbf{D} \boldsymbol{y}. \quad (4)$$

The vector $\boldsymbol{y}$ can be thought of an indicator vector for the partition. Furthermore, the subsequent eigenvectors are the real valued solutions that form the optimal sub-partition.

The *normalized cut* criterion have been used successfully for segmenting static images based on brightness, color, and texture information[16]. To extend it to motion segmentation, we need to first define an appropriate $\mathbf{W}$. This will be based on the notion of motion profile.

3 Motion Profile

Traditionally, local motion information in an image sequence is represented by optical flow, which can be estimated by using the outputs of spatiotemporal filters [1, 9, 8], or by using differential techniques based the brightness constancy assumption [14]. Although these two techniques differ in the details of their formulation, fundamentally they are equivalent[18]. The basic limitations of those techniques are also quite similar – one can not determine the image velocity reliably at locations where the intensity profile is flat, such as the image of a featureless wall, or image regions with a one dimensional intensity profile, such as an extended edge. Figure (1) illustrates these difficulties in one typical image sequence.

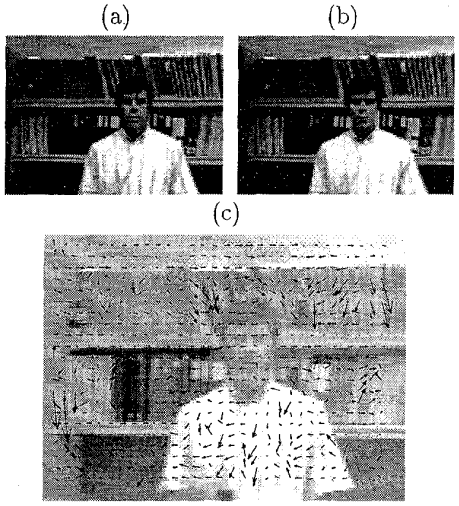

Figure 1: Subimage (a) and (b) shows two frames of an image sequence, and optical flow computed by the Lucas-Kanade algorithm is shown in (c). Notice that the optical flow estimates are reasonable in the textured regions, while in constant brightness regions such as the shirt and the area below the bookshelf, or in regions of repetitive structure, the algorithm performs poorly.

There have been various attempts to fix these problems in optical flow computation. Some restrict their algorithms to be run only at the places where velocity can be computed reliably[17], while others impose a smoothness constraint and apply regularization to obtain a smooth looking output[10, 2, 15, 4]. Alternatively, one could combine the process of motion measurement with image segmentation as has been done successfully in recent layer based approaches to motion analysis in the EM framework.

In our framework, the segmentation is going to emerge as a *result* of finding partitions that minimize the normalized cut, so evidently the feature similarity should be based on local measures of motion that can be computed *before* the segmentation is known. Instead of deciding locally and prematurely on the optical flow vector, we use the *motion profile*, a measure of the probability distribution of the image velocity at each pixel as our motion feature vector. Let $\boldsymbol{I}^t(\mathbf{X})$ denote a window centered at the pixel at location $\mathbf{X} \in R^2$ at time t. We denote by $P_i(\mathbf{dx})$ the probability of an image patch at node i, $\boldsymbol{I}^t(\mathbf{X}_i)$, at time t corresponding to another image patch $\boldsymbol{I}^{t+1}(\mathbf{X}_i + \mathbf{dx})$ at time $t+1$. $P_i(\mathbf{dx})$ can be estimated by first computing the similarity $S_i(\mathbf{dx})$ between $\boldsymbol{I}^t(\mathbf{X}_i)$ and $\boldsymbol{I}^{t+1}(\mathbf{X}_i + \mathbf{dx})$, and normalizing it to get a probability distribution:

$$P_i(\mathbf{dx}) = \frac{S_i(\mathbf{dx})}{\sum_{\mathbf{dx}} S_i(\mathbf{dx})}. \tag{5}$$

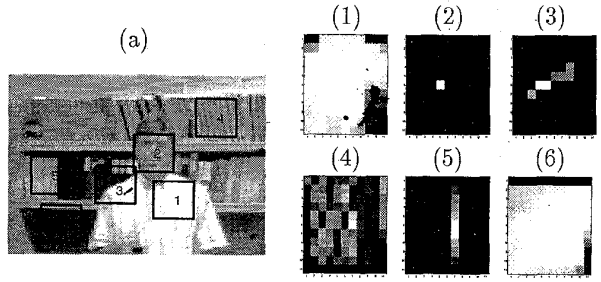

Figure 2: (a) outlines 6 image regions with various intensity profiles. Subimages (1)-(6) shows the corresponding motion profiles at pixels centered in the regions shown in (a). Note these motion vectors have captured the image velocities at those points as well as their associated uncertainties.

There are many ways one can compute similarity between two image patches; we will use a measure that is based on the SSD difference:

$$S_i(\mathbf{dx}) = exp(-\sum_{\mathbf{w}} (\boldsymbol{I}^t(\mathbf{X}_i+\mathbf{w}) - \boldsymbol{I}^{t+1}(\mathbf{X}_i+\mathbf{dx}+\mathbf{w}))^2 / \sigma_{ssd}^2), \tag{6}$$

where $\mathbf{w} \in R^2$ is within a local neighborhood of image patch $\boldsymbol{I}^t(\mathbf{X}_i)$. Figure (2) shows the motion profile computed according to the above definition on various image patches on an image shown in figure (1).

4 Motion Segmentation

To segment a motion sequence, a weighted graph is constructed by taking each pixel as a node, and connecting nodes that are in a spatialtemporal neighborhood of each other. The weight on a graph edge connecting two image pixels reflects the similarity between their motion profiles. We found cross-correlation of the two motion profiles to be a simple, yet effective, measure of motion similarity. Define the distance between two image patches i and j as

$$\mathbf{d}(i,j) = 1 - \sum_{\mathbf{dx}} P_i(\mathbf{dx}) P_j(\mathbf{dx}), \tag{7}$$

where $\mathbf{dx}$ range over possible displacements. The weight on graph edge (i,j) is then given by $w_{ij} = exp(-\mathbf{d}(i,j)/\sigma_m^2)$.

It should be noted that this measure of motion similarity will distinguish between two pixels which have exactly the same true motion, but where the brightness profiles are such that the associated motion uncertainties are very different. If one of the pixels is in a region of constant brightness and another in a region of rich texture this will happen. We believe that this is entirely appropriate given local information; the consequence can be over segmentation of a single rigidly moving object. This however is trivially handled in a postprocessing step.

For computational complexity reasons, we limit the number of nodes in the graph by subsampling the image (factor of 3 from the full spatial resolution in our examples), and limit the number of edges in the graph by having nonzero $w(i,j)$ only for nodes in a spatiotemporal window of ± 3 frames, and ± 5 in x and y. The image patch size used for computing patch-patch comparisons is 5×5 pixels.

To partition the graph, we construct the association matrix $\boldsymbol{W}$ with entry $\boldsymbol{W}(i,j) = w_{ij}$, and solve for the first few eigenvectors of the generalized eigensystem $(\boldsymbol{D} - \boldsymbol{W})\mathbf{y} = \lambda \boldsymbol{D}\mathbf{y}$. This generalized eigensystem can be transformed into a standard eigensystem: $\boldsymbol{A}\mathbf{z} = \lambda \mathbf{z}$, where $\boldsymbol{A} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{D}^{-\frac{1}{2}}$, and $\mathbf{z} = \boldsymbol{D}^{\frac{1}{2}}\mathbf{y}$. The Lanczos method can be used to obtain the first few eigenvectors efficiently if $\boldsymbol{A}$ is sparse. The running time of the Lanczos algorithm is dominated by the $O(mM)$ term where m is the maximum allowed iterations for solving the eigenvectors, and $O(M)$ the cost of computing matrix vector multiplication $\boldsymbol{A}\mathbf{y} = \mathbf{y}^*$. To see why this matrix vector multiplication costs only $O(n)$, where n is the number of nodes in the graph, we will look at the cost of inner product of one row of $\boldsymbol{A}$ with a vector $\mathbf{y}$. Let $\mathbf{y}^*_i = \boldsymbol{A}_i \cdot \mathbf{y} = \sum_j \boldsymbol{A}_{ij}\mathbf{y}_j$. For a fixed i, $\boldsymbol{A}_{ij}$ is only nonzero if node j is in a fixed space-time neighborhood of i. Hence there are only a fixed number of operations required for each $\boldsymbol{A}_i \cdot \mathbf{y}$, and the total cost of computing $\boldsymbol{A}\mathbf{y}$ is $O(n)$. Furthermore, each of those $\boldsymbol{A}_i \cdot \mathbf{y}$ inner product corresponds to a local space-time convolution type of operation, and therefore can be implemented efficiently on parallel processors.

Figures (3) shows the computed generalized eigenvectors for the two frame image sequences shown in figure (1). Recall the eigenvectors are real valued solutions to our recursive *normalized cut* problem. Ideally they should take on discrete values, and successive eigenvectors should subpartition the previous segments. This is indeed the case for the two smallest eigenvectors in figure (3). The first eigenvector segments out the moving person in the foreground, while the second eigenvector separates out the left side of the wall under the bookshelf. However, the values in the third eigenvector exhibit a more smoothly varying profile, particularly in the bookshelf area which it is attempting to subpartition. There are many ways of interpreting this phenomena. From the perspective of segmentation, one can interpret such an eigenvector as unstable in the sense that there are many different partitioning points which have similar $Ncut$ values. In our current segmentation scheme, we simply choose to ignore all those eigenvectors which have smoothly varying eigenvector values. One simple diagnostic measure for detecting such instability is based on first computing the histogram of the eigenvector values, and then take the ratio between the minimum and maximum values in the bins. When the eigenvector values vary continuously, the values in the histogram bin will stay relatively the same, and the ratio will be relatively high. In our experiments, we find a simple threshold on that ratio, set to be 0.06 in all our experiments, can be used effectively.

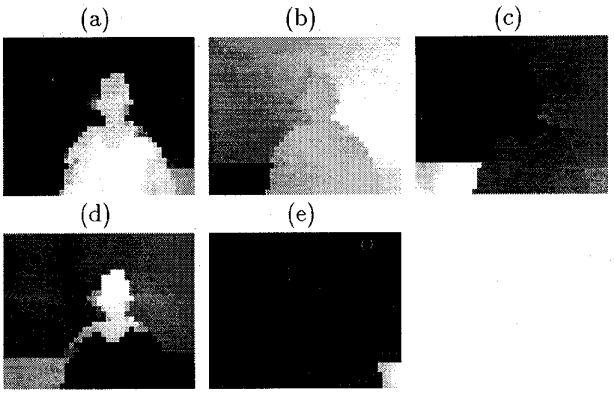

Figure 3: Subplots (a) to (e) show the 5 smallest generalized eigenvectors with eigenvalue less than 0.06 computed for segmenting the image sequence in figure (1). Using the first eigenvector in (a), the foreground person can be segmented from the background, while later on using the eigenvector in (d), the head of the person can be segmented from the body. See figure (4).

In summary, our grouping algorithm can be described as:

1. Given an image sequence, set up a weighted graph $G = (V, E)$ by taking a subsample of the image pixels as the node of the graph(A factor of 3 from the full resolution in our examples). Connect nodes that are less than r_s superpixels apart in space, and r_t frames apart in time. For each pair of nodes compute their motion profiles, and define the weight of the graph edge connecting them as in equation (12). Summarize the information into $\mathbf{W}$, and $\mathbf{D}$. In our experiments, $r_s = 5$ and $r_t = 3$.

2. Solve $(\mathbf{D} - \mathbf{W})\boldsymbol{y} = \lambda \mathbf{D}\boldsymbol{y}$ for eigenvectors with the smallest eigenvalues.

3. Use the eigenvector with second smallest eigenvalue to bipartition the graph by finding the splitting point such that $Ncut$ is minimized,

4. Decide if the current partition should be used by checking the stability of the cut, and make sure $Ncut$ is below pre-specified value,

5. Recursively repartition the segments using the next smallest eigenvectors.

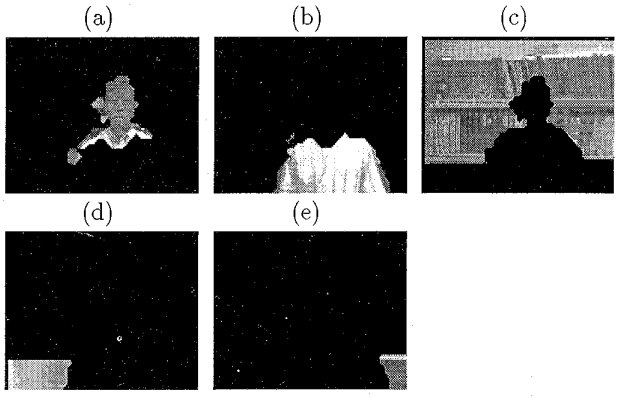

Figure 4: Segmentation of image sequence in figure (1) based on the eigenvectors in figure (2). Segments in (a) and (b) correspond to the person in the foreground, and segments in (c) to (e) correspond to the background. The reason that the head of the person is segmented away from the body is that although they have similar motion, their motion profiles are different. As we saw in figure (2) the head region contains 2D textures and the motion profile are more peaked, while in the body region the motion profiles are more spread out. Segment (c) is broken away from (d) and (e) for the same reason.

The number of groups segmented by this method is controlled directly by the maximum allowed $Ncut$, which is set to 0.05 in all our experiments.

Figure (4) shows the segmentation result on the two frame image sequence in figure (1) based on eigenvectors computed in figure (3).

Figure (5) shows the result of motion segmentation on a seven frame Carl Lewis running image sequence. The image sequence is used because 1) it has very poor image quality: the image noise is very high, and image contrast is relatively low, and 2) it contains an articulated body with different motions on each of the limbs. As the camera is panning to keep Carl Lewis in the center of the frame, there is a moving background which would make background subtraction techniques fail. We wanted to see if the algorithm is able to find the most dominant motion blocks in the image sequence. Results in figure (5) shows that we indeed achieve this goal even under poor image conditions.

5 Segmenting and Tracking Long Image Sequences

The method described above takes in a fixed number of image frames, and produce a segmentation in a *batch* type of operation. The advantage of computing segmentation based on multiple image frames is that one can incorporate information across several frames to produce the best partition. However, this becomes computationally impractical and inefficient if we have to segment a very long image sequence this way.

To solve this problem, we will use only a fixed number of image frames centered around each incoming image frame in the time domain to compute the segmentation. Figure (6) illustrates this idea.

Because there is a significant overlap of the image frames used to compute the segmentation from one time step to another, we can use it to our advantage to speed up our computation. The place where we can gain most of the speed up is in the step of solving the generalized eigensystem $(\boldsymbol{D} - \boldsymbol{W})\mathbf{y} = \lambda \boldsymbol{D}\mathbf{y}$, or its equivalent form $\boldsymbol{A}\mathbf{z} = \lambda\mathbf{z}$, where $\boldsymbol{A} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{D}^{-\frac{1}{2}}$, and $\mathbf{z} = \boldsymbol{D}^{\frac{1}{2}}\mathbf{y}$. We exploit the fact that the Lanczos method of computing eigenvector for a sparse matrix is closely related to the problem of computing orthonormal bases for the *Krylov subspace* associated with matrix $\boldsymbol{A}$. If we have a good guess of the vectors that spans that subspace, we can arrive at the solution very quickly. We shall see that this is indeed achieved by our algorithm.

Let $\boldsymbol{A}^t$ denote the matrix constructed at time t from image frame $t - k$ to $t + k$. We can break $\boldsymbol{A}^t$ into submatrices corresponding to the connections between

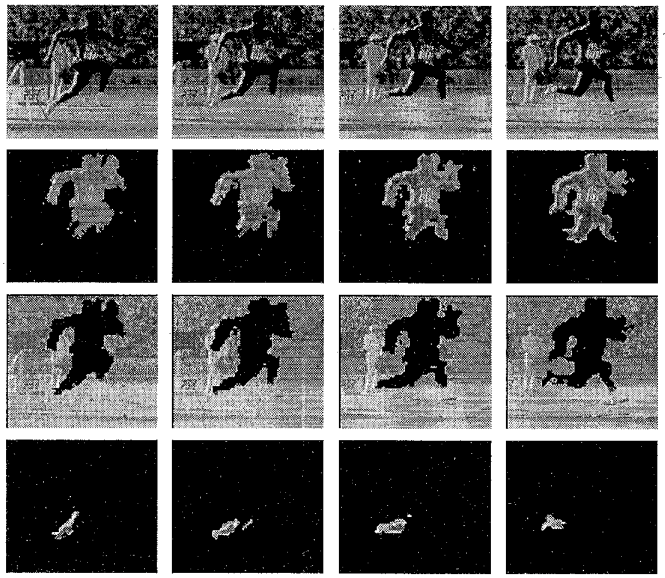

Figure 5: The first row shows an image sequence of Carl Lewis running. Notice that the background is moving to the left as the camera is panning to keep the runner in the center of the image, and therefore background subtraction would not work as an image segmentation technique. The original image size is 200 x 190, and image patches of size 3 x 3 is used to construct the partition graph. Each of the image patches are connected to others that are less than 5 superpixels and 3 image frames away. Row 2 to 4 show the motion segmentation produced by our algorithm. Note these regions found corresponds the runner in row 2, moving background in row 3, and the left lower leg in row 4. The left lower leg is segmented from the runner because it undergoes significant upward rotation in these seven image frames. By recursive cuts and by lowering the maximum allowed $Ncut$ value, the other moving limbs can be found.

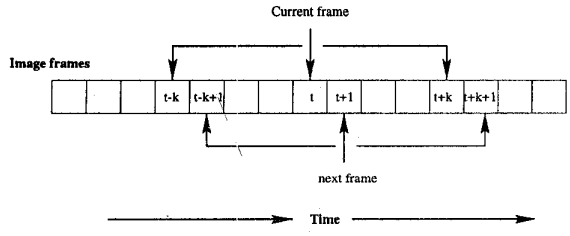

Figure 6: At any given time step t, only $\pm$ k image frames centered around t are used to construct the partition graph, for computing the segmentation and group correspondence. At the next time step $t+1$, image frame $t-k$ is dropped and $t+k+1$ is incorporated in our grouping algorithm.

each of the image frames:

$$A^t = \begin{bmatrix} A_{(t-k)(t-k)} & A_{(t-k)(t-k+1)} & \cdots & 0 \\ A_{(t-k+1)(t-k)} & A_{(t-k+1)(t-k+1)} & \vdots & \vdots \\ 0 & 0 & \cdots & A_{(t+k)(t+k)} \end{bmatrix}, \tag{8}$$

where each $\boldsymbol{A}_{ij}$ encodes the connections from nodes in frame i to nodes in frame j. Let $\mathbf{z}^t$ be an eigenvector for $\boldsymbol{A}^t$. $\mathbf{z}^t$ can also be broken into sub-vectors corresponding to each of the frames:

$$\mathbf{z}^t = \begin{bmatrix} \mathbf{z}^t_{t-k} \\ \mathbf{z}^t_{t-k+1} \\ \vdots \\ \mathbf{z}^t_{t+k} \end{bmatrix} \tag{9}$$

When the time step is advanced by one frame, the new $\boldsymbol{A}^{t+1}$ is related to the previous $\boldsymbol{A}^t$ by,

$$A^{t+1} = \begin{bmatrix} & & & 0 \\ & A^t(2:2k+1, 2:2k+1) & & \vdots \\ & & & A_{(t+k)(t+k+1)} \\ 0 & \cdots & A_{(t+k+1)(t+k)} & A_{(t+k+1)(t+k+1)} \end{bmatrix}. \tag{10}$$

Let $\mathbf{z}^{t+1}$ be the eigenvector for $\boldsymbol{A}^{t+1}$,

$$\mathbf{z}^{t+1} = \begin{bmatrix} \mathbf{z}^{t+1}_{t-k+1} \\ \vdots \\ \mathbf{z}^{t+1}_{t+k} \\ \mathbf{z}^{t+1}_{t+k+1} \end{bmatrix}. \tag{11}$$

Unless there is a major scene change at time $t+1$, we expect the first $2k$ frames of $\mathbf{z}^{t+1}$ to stay relatively the same. To initialize the component corresponding to the new frame, we can compute $\mathbf{z}^{t+1}_* = A^{t+1}\mathbf{z}$, where $\mathbf{z} = [\mathbf{z}^t_{t-k+1}, \cdots, \mathbf{z}^t_{t+k}, 0]^t$. Intuitively this amounts to interpolating from the eigenvectors from the previous frames to obtain a guess at the eigenvector for the frame $t+k+1$. $\mathbf{z}^{t+1}_*$ is then input to the Lanczos eigenvector solver to speed up the computation.

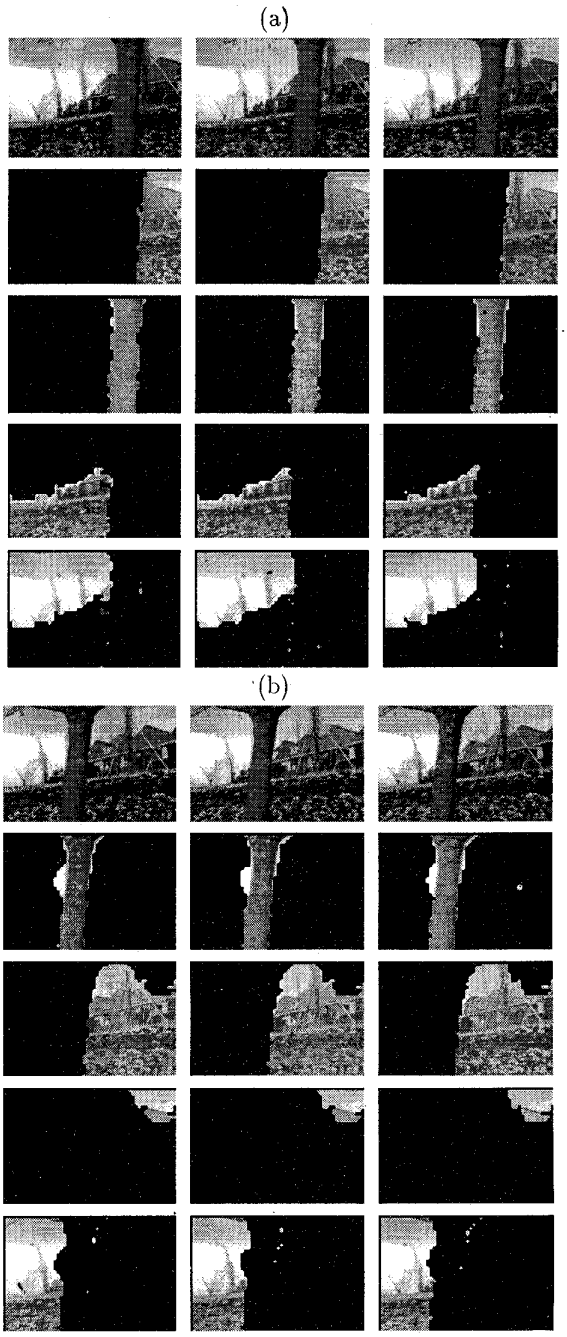

Figure 7: Subplot (a) shows the three of the first six frames of the "flower garden" sequence along with the segmentation. The original image size is 120 x 175, and image patches of size 3 x 3 are used to construct the partition graph. Each of the image patches are connected to others that are less than 5 superpixels and 3 image frames away. Subplot (b) shows the 15th to the 18th frame of the sequence and the motion segmentation using tracking algorithm with the sliding time window method.

To test our method, we used the standard "flower garden" sequence. In this experiment, we set the half-width of the time window, k, to 2 frames. Figure (7a) shows three of the first six frames of the image sequence along with the motion segmentation. In sliding time window updating mode, the cost of computing the generalized eigenvectors at the each of the new time steps amounts to only 20% of the cost in the batch mode. Figure (7b) shows the 15th to the 18th image frame along with the motion segmentation.

6 Conclusion

In this paper, we have developed a motion segmentation algorithm based on the *normalized cuts* graph partitioning method. We treat the image sequence as a three dimensional spatiotemporal data set and construct a weighted graph by taking each pixel as a node, and connecting pixels that are in the spatiotemporal neighborhood of each other. We define a motion profile vector at each image pixel which captures the probability distribution of the image velocity at that point. By defining a distance between motion profile at two pixels, we can assign a weight on the graph edge connecting them. Using *normalized cuts* we find the most salient partitions of the spatiotemporal volume formed by the image sequence. Each partition, which is in the form of a spatiotemporal volume, corresponds to a group of pixels moving coherently in space and time. We have also developed a recursive technique for segmenting and tracking long image sequences. Experimental results on various real image sequences are presented.

Acknowledgments: This research is supported by (ARO)DAAH04-96-1-0341, and an NSF Graduate Fellowship to J. Shi.

References

[1] E. Adelson and J. Bergen. Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am.*, 2(2):284–299, 1985.

[2] P. Anandan. A computational framework and an algorithm for the measurement of vision motion. *Int. J. of Computer Vision*, 2:283–310, 1989.

[3] Serge Ayer and Harpreet S. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *Int. Conf. Computer Vision*, pages 777–784, 1995.

[4] M. Black and P. Anandan. Robust dynamic motion estimation over time. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 296–302, 1991.

[5] R. Cipolla and A. Blake. Surface orientation and time to contact from image divergence and deformation. In *Second European Conference on Computer Vision*, pages 187–203, 1992.

[6] T. Darrell and A. Pentland. Robust estimation of a multi-layered motion representation. In *IEEE Workshop on Visual Motion*, pages 173–178, 1991.

[7] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.

[8] D. Fleet and A. Jepson. Computation of component image velocity from local phase information. *Int. J. of Computer Vision*, 5:77–104, 1990.

[9] D.J. Heeger. Optical flow using spatiotemporal filters. *Int. J. of Computer Vision*, 2:181–190, 1990.

[10] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[11] S. Hsu, P. Anandan, and S. Peleg. Accurate computation of optical flow by using layered motion representation. In *12th International Conference on Pattern Recognition*, pages A:743–746, 1994.

[12] A. Jepson and M. J. Black. Mixture models for optical flow computation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 760–761, 1993.

[13] J.J. Koenderink. Optical flow. *Vision Research*, 26(1):161–179, 1986.

[14] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proc. 7th Int. Joint Conf. on Art. Intell.*, pages 121–130, 1981.

[15] H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8:565–593, Sept. 1986.

[16] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 731–737, 1997.

[17] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[18] E. P. Simoncelli. *Distributed Representation and Analysis of Vision Motion.* PhD thesis, MIT Media Laboratory, 1993.

[19] P. H. S. Torr and D. W. Murray. Stochastic motion clustering. In *Proc. 3rd European Conf. on Computer Vision, Stockholm*, pages B:328–337, 1994.

[20] J.Y.A. Wang and E.H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, pages 625–638, 1994.

[21] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 520–526, 1997.

[22] 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. Computer Vision and Pattern Recognition*, pages 321–326, 1996.

[23] M. Wertheimer. Laws of organization in perceptual forms. In W.B. Ellis, editor, *A Sourcebook of Gestalt Psycychology(Partial translation)*, pages 71–88. Harcourt, Brace and Company, 1938.