Non-Negative Matrix Factorization of Partial Track Data for Motion Segmentation

Anil M. Cheriyadat[1,2] and Richard J. Radke[2]
[1]Computational Sciences and Engineering Division
Oak Ridge National Laboratory, Oak Ridge, TN 37831
[2]Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute, Troy, NY 12180

Abstract

This paper addresses the problem of segmenting low-level partial feature point tracks belonging to multiple motions. We show that the local velocity vectors at each instant of the trajectory are an effective basis for motion segmentation. We decompose the velocity profiles of point tracks into different motion components and corresponding non-negative weights using non-negative matrix factorization (NNMF). We then segment the different motions using spectral clustering on the derived weights. We test our algorithm on the Hopkins 155 benchmarking database and several new sequences, demonstrating that the proposed algorithm can accurately segment multiple motions at a speed of a few seconds per frame. We show that our algorithm is particularly successful on low-level tracks from real-world video that are fragmented, noisy and inaccurate.

1. Introduction

The automated analysis of dynamic scenes from video data requires efficient segmentation of multiple object motions. Such motions can be generated by independent objects, articulated parts of the same object, or the camera itself. Fast, accurate solutions are required for large volumes of data, e.g., from surveillance applications. The main contribution of this paper is a fast algorithm for motion segmentation that is highly robust to the noisy, missing, or partial data typical of real-world tracking algorithms. The proposed algorithm is computationally efficient (seconds or less per frame), and its speed and performance is independent of the number of underlying motions. Figure 1 illustrates results from our algorithm on several example video frames.

As discussed below, many previous motion segmentation methods are based on applying geometrical constraints to motion subspaces. Instead, our approach explores a different aspect of the point track data: the local velocity information associated with the point tracks. Our approach begins by computing a *velocity profile* for each point track, consisting of the local velocity magnitude and direction computed at each temporal instant. Our expectation is that all the point tracks associated with a single motion should have a similar velocity profile structure. We decompose each structure into a different non-negative combination of the same non-negative motion subcomponents. Our analysis illustrates that point tracks belonging to the same object motion tend to have high coherence in the non-negative weight space, and we exploit this coherence for motion segmentation using the N-cut graph cut framework proposed by Shi and Malik [15]. Our algorithm offers a consistent framework to handle both complete and incomplete motion data, where in the latter case discrimination between motion groups is automatically learned from partial motion data. As a result our algorithm is able to segment incomplete point track data without the need for point track reconstruction steps (e.g., as required in [13]).

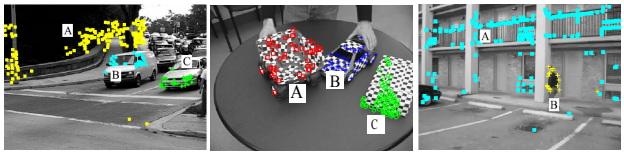

Figure 1. Frames from three of the many different sequences used to test our algorithm. Motion segmentation results are overlaid on the frames. The figures in this paper are best viewed in color.

We tested our algorithm on the Hopkins 155 benchmark dataset [17], which contains examples of independent, articulated, rigid, and degenerate motions, as well as on the dataset proposed by Sugaya and Kanatani [16]. We also tested our algorithm on several new and challenging video sequences containing occlusions, multiple object motions, and camera motion. We found that our algorithm performs particularly well on natural videos with

three or more motions, and those with a substantial amount of missing data. For videos of all the results, which more clearly illustrate the quality of the segmentations, please view the supplementary videos at `http://www.ecse.rpi.edu/~rjradke/nnmf/`.

2. Related Work

We assume we are given a set of P feature points tracked through F frames, where the location of the p^{th} point in the f^{th} frame is given by (x_f^p, y_f^p). To allow for missing data, we also define an indicator variable I_f^p that equals 1 if feature p was tracked in frame f, and 0 otherwise.

We can collect all the track information into a $2F \times P$ motion matrix M where the p^{th} column is given by

$$[x_1^p, \ldots, x_F^p, y_1^p, \ldots, y_F^p]^T. \quad (1)$$

Entries corresponding to missing data ($I_f^p = 0$) are assigned zeros in M and unused by our algorithm. The goal of motion segmentation is to determine a permutation of the columns of M to form $[M^1|M^2|\ldots|M^N]$, where the submatrix M^i is composed of point tracks associated with the i^{th} object motion. Previous solutions to this problem were based on methods including factorization [4, 7, 8], generalized PCA [18], statistical learning [16, 20], and minimum description length [13].

The basic approach behind several early methods exploited the observation that the trajectories associated with each motion lie in a subspace of dimension four or less [1, 4]. Factorization approaches [4, 7, 8] provided an elegant framework to partition the matrix M directly into maximally rank deficient submatrices M^i. However, these approaches were shown to degrade when the motions were not independent, or the input point tracks were noisy or incomplete. Zelnik-Manor and Irani [21] applied factorization on the matrix M^T with a different objective of clustering frames containing consistent shapes. They showed that their approach resulted in temporal segmentation of video frames (clustering rows of M).

The multi-stage learning method proposed by Sugaya and Kanatani [16], hereafter termed as MSL, combined subspace factorization with statistical learning. The algorithm starts with the subspace factorization approach proposed by Costeira and Kanade [4], and uses the EM algorithm to refine the motion segmentation results. While this method produced very good segmentations, it is generally impractical since it takes hours to converge.

The algebraic method proposed by Vidal and Hartley [18], hereafter termed as GPCA, fits a set of polynomials to the point tracks after their projection onto a 5-dimensional subspace. A basis for each motion subspace is obtained from derivatives of these polynomials, and different motions are segmented using spectral clustering on subspace angles. The method is computationally efficient. However, one of the drawbacks is that the proposed method does not scale well with number of motions. As noted by the original authors [18], the number of coefficients that need to be estimated grows exponentially with the number of subspaces and dimension of the subspace. Hence the amount of trajectory data that might be available in a real-time practical situation will limit GPCA's ability to segment a large number of motions.

The method proposed by Yan and Pollefeys [20], hereafter termed as LSA, also begins with projecting the point track data onto a lower dimensional linear normalized subspace. In contrast to the global fit subspace criterion pursued by [18], the LSA method seeks to fit a local subspace around each point. The motion similarity between a pair of point tracks is computed from the principal angles between the local subspaces occupied by these tracks. Segmentation is achieved through spectral clustering. Since the method relies on local subspace estimation, for motions with significant spatial overlap, this method will yield suboptimal performance.

Recently Rao et al. [13] proposed an algorithm, hereafter termed as ALC, to cluster the point tracks based on the principles of lossy minimum description length. They argue that subspace separation based on matrix rank minimization is challenging, and that similar segmentation of point tracks can be achieved by finding the set of point tracks that minimize the coding length required to describe the data up to a distortion parameter. The method uses an agglomerative scheme to find the minimum coding length. This involves running several iterations of the algorithm with different distortion settings to estimate the parameter yielding the minimum coding length; hence it is a computationally expensive procedure.

In real-world automatic tracking, scene dynamics, occlusions and tracker limitations can often result in tracks being highly fragmented, noisy and inaccurate. Many of the above algorithms have proposed different ways of handling these data irregularities. In the case of GPCA [18], power factorization is used to project missing data onto a 5-dimensional space. In the case of ALC [13], the algorithm utilizes the subspace constraints to reconstruct the fragmented and noisy data before the motion segmentation process. There is a clear need to develop a fast motion segmentation algorithm that can scale well with the number of motions and can handle partial or missing data efficiently. As illustrated in next sections our proposed algorithm is able to achieve these objectives in a consistent manner.

3. Proposed Algorithm

Broadly, our algorithm proceeds as summarized in Algorithm 1. We now discuss each step of our algorithm in detail.

Algorithm 1 Motion Segmentation

1. Compute velocity profile matrix V from track data
2. Apply non-negative matrix factorization to V to obtain subcomponent motions S and weights W
3. Compute affinity matrix A from W
4. Apply spectral clustering to A to generate N clusters

We now discuss each step of our algorithm in detail.

3.1. Point Track Velocity Profiles

The first step in our algorithm is to derive a velocity profile for each point track. This is simply based on the instantaneous magnitude and angle of the track computed as:

$$m_f^p = \sqrt{(x_{f+1}^p - x_f^p)^2 + (y_{f+1}^p - y_f^p)^2} \quad (2)$$

$$c_f^p = \frac{x_{f+1}^p - x_f^p}{\sqrt{(x_{f+1}^p - x_f^p)^2 + (y_{f+1}^p - y_f^p)^2}} + 1 \quad (3)$$

$$s_f^p = \frac{y_{f+1}^p - y_f^p}{\sqrt{(x_{f+1}^p - x_f^p)^2 + (y_{f+1}^p - y_f^p)^2}} + 1 \quad (4)$$

Note that c_f^p and s_f^p (the cosine and sine of the angle plus one respectively) are always in the range [0,2]. The non-negative representation of the point track velocity profiles ensures that the matrix V can be factored into two non-negative matrices. An indicator function $\tilde{I}_f^p$ derived from I_f^p denotes whether the velocity profile exists at each point/instant, to account for missing data.

We collect all velocity information into a non-negative $3(F-1)\times P$ velocity profile matrix V where the p^{th} column is given by

$$[m_1^p, \ldots, m_{F-1}^p, c_1^p, \ldots, c_{F-1}^p, s_1^p, \ldots, s_{F-1}^p]^T. \quad (5)$$

Unused zeros are entered into V in the case of missing data.

3.2. Factorization of Velocity Profiles

We next factor V using non-negative matrix factorization (NNMF), an elegant framework for describing the non-negative measurements as a non-negative linear combination of a set of basis vectors which can be thought of as "building parts" [10]. This parts-based representation of the data is quite different from the holistic data representation offered by factorization approaches like PCA and the SVD. Unlike such approaches, where basis vectors can be added or subtracted for reconstruction, NNMF only allows for addition. NNMF has found a wide range of applications in various areas including face recognition [10], medical data analysis [3], and microarray analysis [9]. Our intuition behind using NNMF for motion segmentation is that the non-negative weights used to combine different "parts" for each track should provide a good measure of data similarity. In the case of partial or missing data, the parts-based representation provided by NNMF offers a way to measure the similarity between partial data sets that cannot be compared directly (e.g., due to zero temporal overlap). Although other approaches such as PCA for missing data [6] exist for handling these data irregularities, our experimental analyses show that NNMF handles missing data gracefully without extra steps.

Let us initially assume that V is fully populated, i.e., $\tilde{I}_f^p = 1$ for all f and p. Formally, our objective is to find non-negative matrices $S \in \mathbb{R}_+^{3(F-1)\times r}$ and $W \in \mathbb{R}_+^{r\times P}$, with $r \ll F, P$, that minimize

$$\|V - SW\|_F^2 \quad (6)$$

Here, the matrix S contains the subcomponent "building parts" of the motions, and W contains the non-negative weighting of the parts for each point track. For the minimization, we apply the common approach of seeking the solution in an alternating least-squares manner [5]. That is, we alternate between fixing S and solving for W, and vice versa. While this procedure is not guaranteed to find a global minimum, we found it to converge quickly in practice and to result in high-quality segmentations. We apply the multiplicative update rules initially proposed by Lee and Seung [10], as given in Algorithm 2.

Algorithm 2 NNMF with Multiplicative Updates

Initialize $S \in \mathbb{R}^{3(F-1)\times r}$ and $W \in \mathbb{R}^{r\times P}$ as random positive matrices.
for n = 1 to T **do**
  $R = V \oslash (SW)$
  $S \leftarrow S \otimes (RW^T)$
  Normalize columns of S to have unit L_1 norm
  $R = V \oslash (SW)$
  $W \leftarrow W \otimes (S^T R)$
end for

Here, $\oslash$ and $\otimes$ represent matrix element-wise division and multiplication respectively. Lee and Seung showed that these rules result in a non-decreasing cost function. The constraint that the column vectors of the subcomponent matrix S should add to one makes sure that the "parts" derived from factorization are "comparable". In accordance with typical practice, we set T to a large number (1000); an alternative would be to terminate when the residuals stop changing significantly. To get the best result, we start with 10 random initializations of (S, W), do 10 iterations of updates for each, and proceed to convergence with the pair

having the minimum residual $\|V - SW\|_F^2$. To avoid numerical problems, exact zeros in V are replaced by machine precision ε.

The number of column vectors in S, denoted by r, corresponds to the number of "parts" that make up the velocity profiles. One possibility for choosing r is to use a model selection algorithm to determine the rank of the matrix V, as suggested by [19]. For our experiments on the Kanatani and Hopkins 155 sequences, we simply set the value of r to 3 and for our longer new video sequences we set r to 4.

Figure 2 shows frames from one of the video sequences used in our experiments. The feature points that are tracked are overlaid on the frame. Point tracks correspond to three different motion groups as indicated by the letters overlaid on the figure. Motion groups A, B and C correspond to track numbers 1-37, 38-111 and 112-548 respectively. The associated affinity matrix, computed as described in [15], from the non-negative weight vectors generated by our algorithm is shown next to the frame. The affinity matrix clearly indicates that the clusters in this case are tight and well-separated, indicating promise for the motion segmentation problem. The affinity matrix A is computed as:

$$A(i,j) = \exp\left(\frac{-\|w(i) - w(j)\|_2}{\sigma}\right) \tag{7}$$

where σ is the scale and $w(i)$ is the i^{th} column vector of W. We used σ in the range 0.01-0.03.

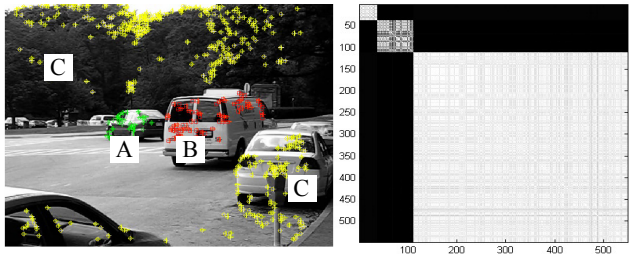

Figure 2. On the left a video frame from the Hopkins 155 [17] database containing multiple motions is shown. On the right the affinity matrix computed from W is shown. A brighter pixel means more similarity. The block diagonals correspond to different motions. For this example, our algorithm yielded 0% misclassification.

3.2.1 Handling Incomplete Data

For incomplete data cases, the cost function (7) must be modified to

$$\sum_{f=1}^{F-1}\sum_{p=1}^{P} \tilde{I}_f^p \|v_f^p - S_f W^p\|^2, \tag{8}$$

where $(v_f^p = (m_f^p, c_f^p, s_f^p)^T$ and S_f and W^p are the appropriate $3 \times r$ and $r \times 1$ submatrices, respectively.

The multiplicative update rules used here provide a simple but effective framework for handling missing data values. The matrix R computed during Algorithm 2 determines the ratios by which elements in the matrices S and W are tuned during each iteration. For the missing data cases, the elements of R corresponding to $\tilde{I}_f^p = 0$ are simply set to 1, meaning that they have no influence on the update.

To motivate the use of NNMF for partial data analysis we first illustrate it on a simulated missing data example. Consider the point tracks for the example sequence shown in Figure 2. In order to simulate missing data, for half of the tracks in each motion group we forced the tracks to have a long band (60-70%) of missing data. The bands of missing data were laid out in such a way that half of the tracks in each motion group have zero overlap with half of the tracks from other groups (Figure 3). Considering the inter- and intra-object occlusions in the scene and tracker limitations, it is not unusual for real-world point tracks to exhibit such data irregularities. Because of these data irregularities, neither direct comparison of the original tracks nor direct application of dimensionality techniques like PCA or SVD is possible here. We use the NNMF framework as described above to factorize these partial tracks. Through iterative steps, NNMF constructs its "building parts" and non-negative weights from the available partial data. The inherent similarity in the data is captured by the non-negative weights that are used to combine the "parts". The affinity matrix computed from the non-negative weights for this missing data case shows similar clustering behavior to the one derived for the complete data case in Figure 2. The results presented in Figure 3 clearly demonstrate the applicability of NNMF for clustering partial point track data.

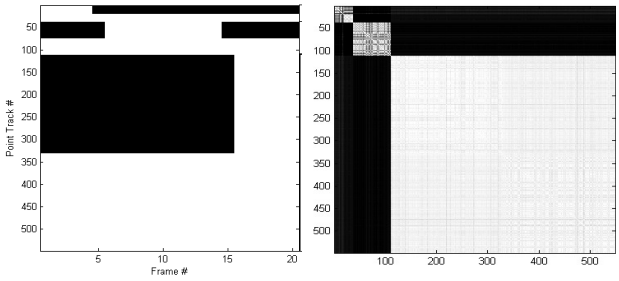

Figure 3. The mask indicating missing data (black pixels) for the example sequence is shown on the left. On the right the affinity matrix computed from the non-negative weight matrix W for this missing data case is shown, which is quite similar to the affinity matrix from Figure 2.

3.3. Spectral Clustering

Generally, we can apply any clustering technique to the column vectors of W to retrieve the N motion clusters.

To avoid the K-means random initialization which is part of spectral clustering [12], we use the iterative multi-stage spectral clustering algorithm proposed by Shi and Malik [15] to achieve stable segmentation results. From a graph cut perspective, our clustering strategy can be viewed as finding subgraphs representing different motions. Here, the column vectors of W form nodes in the graph, and the similarity A computed between column vectors of W form the edge weights. Clustering is achieved through a normalized-cut of this graph network into subgraphs representing different motions.

4. Experiments

In this section, we report experimental results on the test sequences proposed by Sugaya and Kanatani [16], the Hopkins 155 benchmark dataset [17], and several new and challenging video sequences. In the case of the Kanatani and Hopkins 155 sequences, the point tracks are given and were manually cleaned and associated with underlying motions. Thus, we can compute the classification error, given by the ratio of number of misclassified point tracks to the total number of point tracks. In our new videos, point tracks are automatically generated without cleaning, outlier rejection, or manual classification, and qualitative results are reported.

4.1. Kanatani Sequences

We first ran our algorithm on the sequences proposed by Kanatani [8]. Like MSL, GPCA and ALC, our algorithm achieves 0% misclassification. For comparison of these results with additional algorithms, see [16, 18]. Except for GPCA, the computational requirements for the other algorithms are much higher than ours. The average computation time taken by our algorithm on these sequences is 2.5 secs.

4.2. Hopkins 155 Database

The Hopkins 155 database consists of 155 motion sequences representing various motion conditions including independent, articulated, partially dependent, and degenerate motions. The database is subcategorized into different types, and we compared our algorithm on the *traffic* and *articulated* categories. The *traffic* set consists of 38 sequences of outdoor traffic scenes taken by a moving handheld camera. Most scenes contain degenerate motions, particularly linear and planar motions. The *articulated* set consists of 13 sequences displaying motions constrained by joints, heads, faces and other non-rigid motions. Refer to [17] for more details on the Hopkins 155 database. We tested our algorithm on this database and compared the performance with the motion segmentation results reported in [16, 18]. To obtain a fair comparison with previous work, we tested our algorithm separately on sequences with two motions (Table 1) and three motions (Table 2). As with the comparison algorithms, our algorithm is provided with the original point track data and the known number of motions as input (used in the spectral clustering phase).

We report the mean and median classification errors and the average computation time (Table 3) for the three video categories. We compare our algorithm with MSL, LSA, GPCA, and ALC, algorithms which are considered as the state-of-the-art and whose performance results are reported in [16, 18, 13]. Our results were obtained on an Intel dual core 2.4 GHz processor with 2 GB RAM.

Table 1. Misclassification percentages for sequences with two motions. The number of sequences for each category is given in parentheses.

Traffic (31)	MSL	LSA	GPCA	ALC	*Ours*
Mean	2.23%	5.43%	1.41%	1.59%	0.1%
Median	0.0%	1.48%	0%	1.17%	0%
Articulated (11)	MSL	LSA	GPCA	ALC	*Ours*
Mean	7.23%	4.10%	2.88%	10.7%	10%
Median	0.0%	1.22%	0%	0.95%	2.6%

Table 2. Misclassification percentages for sequences with three motions. The number of sequences for each category is given in parentheses.

Traffic (7)	MSL	LSA	GPCA	ALC	*Ours*
Mean	1.8%	25.07%	19.83%	7.75%	0.1%
Median	0%	23.79%	19.55%	0.49%	0%
Articulated (2)	MSL	LSA	GPCA	ALC	*Ours*
Mean	2.71%	7.25%	16.85%	21.08%	15%
Median	2.71%	7.25%	16.85%	21.08%	15%

Table 3. Average computation times for various algorithms

Method	MSL	LSA	GPCA	ALC	*Ours*
Time	19.6 hr	9.7 sec	0.72 sec	21 min	3 sec

As demonstrated by the results, our algorithm gives excellent results for the real-world *traffic* sequences, and comparable results with the other algorithms for the *articulated* sequences. The computation time required by our algorithm is on the order of seconds. Both LSA and GPCA have computation time on the order of seconds whereas MSL and ALC require higher computation time due to the iterative nature of the algorithms. We note that our proposed algorithm is based on a linear matrix factorization and hence might yield suboptimal results for sequences with "strong" camera rotations and background perspective effects. This was validated through our experimental analysis on such sequences fom Hopkins 155 *checkerboard* category which contained non-linear effects.

4.3. Missing Data Cases

We next tested and compared the performance of our algorithm for missing data cases we introduced into the Hop-

kins 155 sequences. In real-world automatic tracking, feature points frequently get occluded or the quality of the feature correspondence degrades during the course of tracking, forcing the tracker to abandon the point track. Hence, it is quite important for segmentation algorithms to robustly handle these data irregularities.

In order to test the ability of our algorithm to handle missing data, we generated random binary masks as shown in Figure 4. For each mask shown, black denotes missing data and white otherwise, and the width and height of the mask represents the number of point tracks and the number of total frames respectively. For each randomly selected point track, we randomly fix the *start* and *duration* of a feature point's successful tracking window. This mask generation process ensures a resulting point track set that contains a mix of a few complete tracks and many incomplete tracks with varying degrees of "incompleteness". We found this mask generation process to closely simulate automatically generated tracks in real world scenes. In our experiments, we generated masks that represent 25-35% missing data values. We used the masks to force missing data in both the *traffic* and *articulated* categories, and ran the publicly available ALC algorithm [13] and our algorithm on these sequences. For each motion data sequence from these categories, we applied the same mask before running both algorithms.

We were interested to compare the segmentation performance of our algorithm with the ALC approach, because the ALC algorithm uses an additional step to reconstruct the data before applying the segmentation algorithm. In contrast, our algorithm does not explicitly reconstruct the data but handles missing data through NNMF factorization. Our results (Table 4) illustrate that our segmentation algorithm is fast and robust in the presence of incomplete data, with only a slight loss in performance over the complete data scenario. Handling partial motion data is important for motion segmentation in real-time and real-world scenarios as illustrated by our next set of experiments.

Figure 4. One of the binary masks used for an example sequence is shown here. Here the rows and columns of the binary mask represent frames F and points P respectively.

4.4. New Video Sequences

In order to test the robustness of our algorithm to real-world, rather than provided/cleaned tracks, we generated several new video sequences that contain multiple motions and are substantially longer than most of the Kanatani/Hopkins sequences. The low-level features are automatically tracked over time using a hierarchical implementation [2] of the Kanade-Lucas-Tomasi optical flow algorithm [11]. No manual effort was made to correct or remove incorrect tracks, and point track generation is a continuous process. That is, at each frame new feature point tracks are initiated in addition to the existing ones. In contrast, all of the Kanatani/Hopkins sequences only use track points initiated in the first frame. The continuous approach ensures a sizable set of "good" point tracks representing different motions in the scene (e.g., producing more tracks when the camera pans significantly). On the other hand, the continuous point track generation process results in wide variations in the temporal extents of the tracks. A robust motion segmentation algorithm should produce good results in the presence of such data.

Table 4. Misclassification percentages for sequences with 20-30% missing data vs. complete data. The results for ALC complete data are reported in [13].

	Missing Data		Complete Data	
Traffic	ALC	*Ours*	ALC	*Ours*
Mean	5.77%	2.2%	2.77%	1.1%
Median	2.39%	0.5%	1.10%	0%
Average Time	13.9 min	5 sec	17.19 min[1]	5 sec
Articulated	ALC	*Ours*	ALC	*Ours*
Mean	18%	13%	13.71%	11%
Median	17%	11%	3.46%	3%
Average Time	8 min	3.6 sec	10.43 min[1]	3.6 sec

We tested our algorithm on four different video sequences we called *drinkingcoffee*, *racquetsession*, *panningcamera* and *rotatingcontainers*. The *drinkingcoffee* sequence contains two different motions, one caused by the person rotating in his chair and the other by the motion of the coffee cup which involves both rotation and translation. The *racquetsession* sequence contains two complex articulated motions generated by the movement of the person and of the racquet head. The *panningcamera* sequence contains two motions, one generated by the person walking and the other by the camera motion. In this sequence, as the camera pans across the scene, feature points appear and disappear at the edges of the frame. For these three sequences, as a result of continuous feature point extraction and tracking, the point tracks have wide variations in their temporal extent. The *rotatingcontainers* sequence contains four different motions: two containers displaying different rotations but the same (minimal) translational motion, a person leaning forward displaying looming/translational motion, and points in the background which are basically stationary. The binary mask indicating missing data for the *racquetsession* sequence is shown in Figure 5, illustrating the "incompleteness" and variations in the degree of temporal overlap.

[1]Complete data timings are reported from [13] and may not be comparable with the missing data timings (run on our PC)

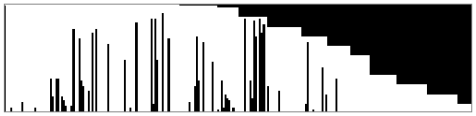

Figure 5. The binary mask obtained from the *racquetsession* sequence, indicating "incompleteness" and variations in track overlap. Some track pairs even have zero overlap. The top row corresponds to the first frame.

The automatically extracted tracks from these sequences and the true number of object motions were input to both our algorithm and the ALC algorithm. Segmentation results from our algorithm are overlaid on exemplar frames from these sequences as shown in Figure 6. The results illustrate that our algorithm is able to produce excellent segmentations on these challenging videos compared to the ALC algorithm, which resulted in incorrect segmentations on all sequences (Figure 6i-j) except *drinkingcoffee*. Table 5 gives details on the videos and timings.

In order to highlight the coherence of point tracks belonging to the same motion, we show in Figure 7 the affinity matrices computed from the non-negative weight vectors (column vectors of W). We have conducted additional experiments on a variety of video sequences and results are available at `http://www.ecse.rpi.edu/~rjradke/nnmf/`.

Table 5. Computation Time for Segmentation of New Videos

Sequence	*drinkingcoffee*	*racquetsession*	*camerapanning*	*rotatingcontainer*
Num of points	1682	407	52	135
Num of frames	49	98	56	290
ALC time	26 hr	20 min	9 min	2.3 min
Ours time	54 sec	36 sec	12 sec	27 sec

5. Conclusions and Future Work

We presented a new motion segmentation technique based on non-negative factorization of velocity profiles to achieve fast and robust motion segmentation from partial low-level feature point tracks. As opposed to previous motion segmentation algorithms that relied on geometrical subspace constraints on the positional information of point track data, the approach proposed in this paper uses the instantaneous velocity information extracted from the point tracks. The proposed approach is simple, fast, and can handle noisy and incomplete data under various motion conditions.

In future work, we plan to fuse the velocity-based motion segmentation approach with low-level object segmentation to distinguish multiple similarly moving objects. We also plan to find better ways of initializing the NNMF, and investigate whether different constraints on S and W produce better motion clusters. The NNMF technique could also be extended based on the ideas of Kernel PCA [14] to handle non-linear "parts". Finally, we plan to improve our clustering technique to automatically determine the number of underlying motions.

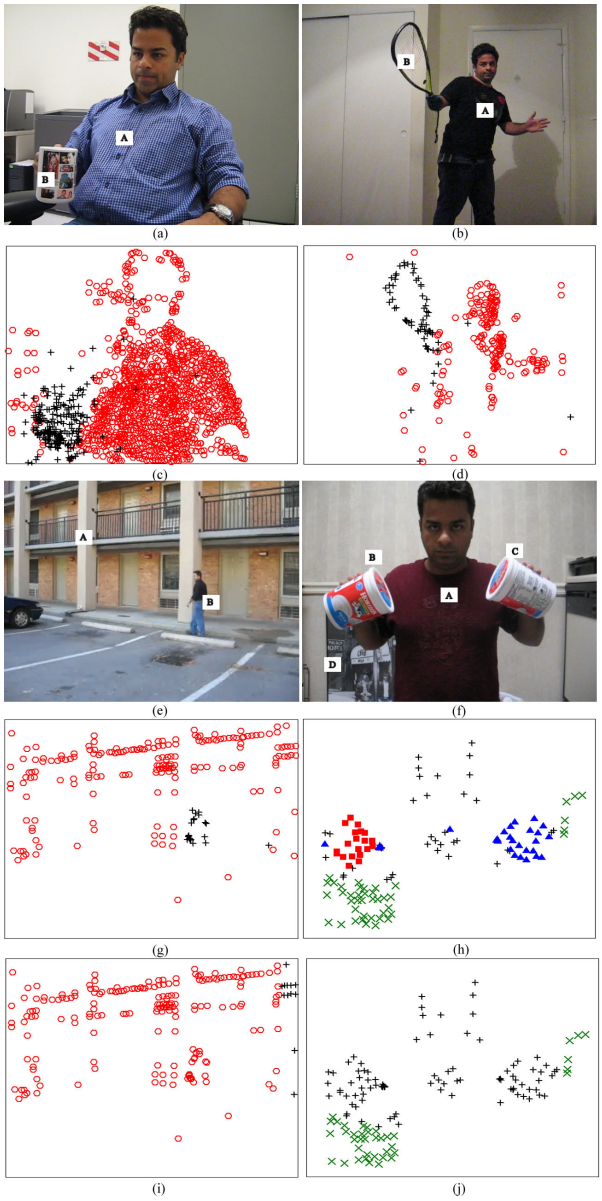

Figure 6. The segmentation results of our algorithm on new video sequences are shown here. (a-c) *drinkingcoffee*, (b-d) *racquetsession*, (e-g) *panningcamera*, (f-h) *rotatingcontainers*. Panels (i-j) are segmentation results from the ALC algorithm for *panningcamera* and *rotatingcontainers*. Except for segmentation on *drinkingcoffee*, the ALC algorithm yielded poor results on the other sequences.

6. Acknowledgements

Prepared by Oak Ridge National Laboratory[2], P.O. Box 2008, Oak Ridge, Tennessee 37831-6285, managed by UT-

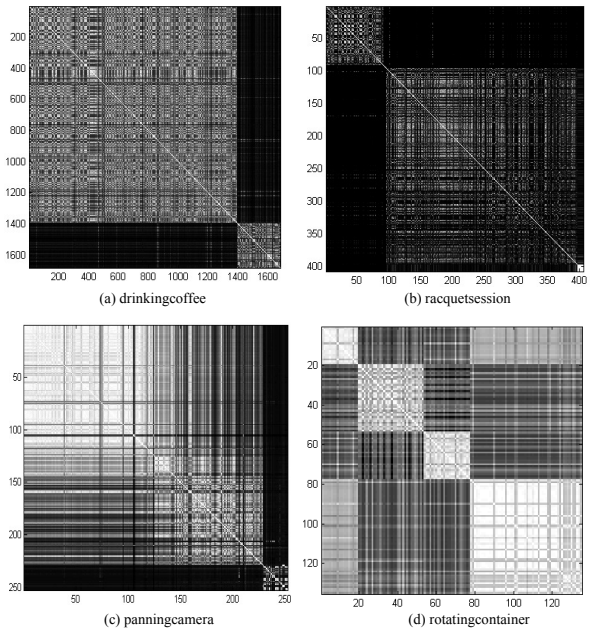

Figure 7. Affinity matrices computed from W for the new sequences. A brighter pixel means more similarity. The column vectors of W are sorted based on the clustering obtained by our algorithm. Block diagonals correspond to different object motions. The clustering accuracy is confirmed through visual inspection of video sequences.

Battelle, LLC for the U.S. Department of Energy under contract no. DEAC05-00OR22725.

References

[1] T. E. Boult and L. G. Brown. Factorization-based segmentation of motions. In *Proceedings of the IEEE Workshop on Motion Understanding*, pages 179–186, 1991.

[2] G. Bradski. OpenCV: Examples of use and new applications in stereo, recognition and tracking. In *Proc. of International Conference on Vision Interface*, 2002.

[3] Z. Chen, A. Cichocki, and T. M. Rutkowski. Constrained non-negative matrix factorization method for EEG analysis in early detection of Alzheimer disease. In *Proceedings of the IEEE International Conference Acoustics, Speech and Signal Processing*, 2006.

[4] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.

[2]This manuscript has been authored by employees of UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. Accordingly, the United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

[5] K. R. Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21:489–498, 1979.

[6] H.Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *Modelling from reality*, pages 3–39, 2001.

[7] N. Ichimura. Motion segmentation based on factorization method and discriminant criterion. *IEEE International Conference on Computer Vision*, pages 600–605, 1999.

[8] K. Kanatani. Motion segmentation by subspace separation and model selection. In *IEEE International Conference on Computer Vision*, pages 586–591, 2001.

[9] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 3(12):1495–1502, 2007.

[10] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[11] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[12] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *In Advances in Neural Information Processing Systems*, 2002.

[13] S. R. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[14] B. Scholkopf, A. Smola, and K. R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Technical Report No. 44, Max Planck Institute*, 1996.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(8):888–905, 2000.

[16] Y. Sugaya and K. Kanatani. Geometric structure of degeneracy for multi-body motion segmentation. In *Proceedings of Workshop on Statistical Methods in Video Processing*, 2004.

[17] R. Tron and R. Vidal. A benchmark for the comparison of 3-D motion segmentation algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[18] R. Vidal and R. Hartley. Motion segmentation with missing data by PowerFactorization and generalized PCA. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 310–316, 2004.

[19] R. Vidal, Y. Ma, and J. Piazzi. A new GPCA algorithm for clustering subspaces by fitting, differentiating and dividing polynomials. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[20] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European Conference on Computer Vision*, pages 94–106, 2006.

[21] L. Zelnik-Manor and M. Irani. Temporal factorization Vs Spatial factorization. In *Proceedings of the European Conference on Computer Vision*, 2004.