Semi-Nonnegative Matrix Factorization for Motion Segmentation with Missing Data[⋆]

Quanyi Mo and Bruce A. Draper

Colorado State University
{qmo,draper}@cs.colostate.edu

Abstract. Motion segmentation is an old problem that is receiving renewed interest because of its role in video analysis. In this paper, we present a Semi-Nonnegative Matrix Factorization (SNMF)method that models dense point tracks in terms of their optical flow, and decomposes sets of point tracks into semantically meaningful motion components. We show that this formulation of SNMF with missing values outperforms the state-of-the-art algorithm of Brox and Malik in terms of accuracy on 10-frame video segments from the Berkeley test set, while being over 100 times faster. We then show how SNMF can be applied to longer videos using sliding windows. The result is competitive in terms of accuracy with Brox and Malik's algorithm, while still being two orders of magnitude faster.

Keywords: Motion Segmentation, Semi-Nonnegative Matrix Factorization(SNMF), Missing Data.

1 Introduction

Motion segmentation is an old and well-studied problem in computer vision that is finding renewed importance because of its role in video analysis. Put simply, many video analysis techniques segment moving objects as a precondition for object, action or event recognition. Unfortunately, video segmentation can be complicated by multiple independent motions, transient occlusions among objects, changes in illumination, flexible and/or articulated objects, camera motions, and related factors. The segmentation challenge is to build a system that can identify parts of scenes that move consistently with each other over time.

In ECCV 2010, Brox and Malik [1] introduced a new public data set and experimental protocol for motion segmentation. Their data set, commonly called the Berkeley Motion Segmentation Dataset, contains multiple videos of varying lengths, with ground truth segmentations provided for some frames. To focus the challenge on segmentation, rather than interest point detection or tracking, they also provide sets of tracked interest points for every video, and an evaluation mechanism that scores point-based segmentations in terms of their overall error, average error, over-segmentation error, and density (in case other point tracks

[⋆] This work is supported by DARPA Contract W911NF-10-2-0066.

are used). The same paper also introduces a segmentation algorithm based on spectral clustering that establishes a performance baseline for the data set. They show that spectral clustering outperforms older, better known methods such as GPCA[2], LSA[3] ALC[4] and RANSAC, and they challenge other researchers to produce better algorithms.

This paper accepts Brox and Malik's challenge. Our first contribution is a motion segmentation algorithm based on Semi-Nonnegative Matrix Factorization (SNMF) with missing values. We show that SMNF is more accurate and over one hundred times faster than the algorithm by Brox and Malik when applied to 10-frame videos (i.e. when applied in evaluation mode 1 in [1]). Unfortunately, as videos get longer, point tracking fails more often because of occlusions, changes in lighting, and frame boundaries. As a result, the percent of missing values grows, and the likelihood of SNMF getting caught in an undesirable local minima increases. Our second contribution is a sliding window algorithm that uses SNMF to segment overlapping 10-frame windows of data and spectral clustering to piece together segments across windows. The result is an algorithm that can process arbitrarily long videos. On the Berkeley data set it is competitive with the algorithm by Brox and Malik (within two percent of overall error) while still being two orders of magnitude faster.

2 Previous Work

The motion segmentation problem has been studied widely by several research communities, and a full survey of the literature is beyond the current scope. For the purposes of this paper, we concentrate on two general methods. The first method approaches motion segmentation as clustering. These techniques use local information to evaluate trajectory similarities[3][5]. Brox and Malik's method falls into this group; they define the distance between trajectories as the maximum difference of their motion over time. This approach is shown to be effective in separating moving objects. However, these methods need to construct an affinity matrix for all trajectories. When the tracked trajectories become dense, the computational complexity increases quickly. For example, the largest video in the Berkeley motion segmentation dataset has $163,266$ tracks and requires about 30 hours to be processed. One of the objectives in this paper is to propose a faster algorithm to solve this trajectory clustering problem.

The other basic approach represents motion segmentation as subspace separation. This idea is that recognizable motion patterns often live in one or more low dimensional subspaces. Vidal et al[2] propose GPCA(Generalized Principle Component Analysis), an algebraic method to fit a set of polynomials to a point track after projection into a low dimensional subspace. Alternatively, matrix factorization methods can be used to construct low rank representations of motion data[4][6]. Our approach fits in this category. Many matrix factorization methods require that all trajectories be the same length. However, real world trajectory data always contains missing values due to occlusion, frame transition or tracking failure. While subspace methods are elegant and seem to catch the essence of motion patterns, dealing with missing data is a key problem to solve.

The previous method closest to ours is NMF (Non-Negative Matrix Factorization), as proposed by Cheriyadat et al[7]. They construct a velocity profile matrix by representing optical flow as velocity magnitudes and angles to fit the NMF framework. A clever modification is made on Lee et al's classic NMF method[8] to handle missing data. However, using representation of velocity magnitude and angle in a linear model hurts interpretability. In addition, their missing data handling mechanism is somewhat ad-hoc and lacks rigorous analysis. Matrix factorization with missing data is a famous non-convex problem. While the length of tracks and the proportion of missing data increases, almost all factorization methods fall into sub-optimal local minima. In contrast, we propose using matrix factorization methods over short time windows to get stable results and then propagating information along the timeline. The experimental results show that this strategy is effective in handling longer videos with more missing data. We also build our approach on SNMF[9] (rather than NMF), which allow us to use velocity information directly to build a more natural motion component representation.

3 Methods

The input to point-based video segmentation is a dense set of points extracted from frames in a video and tracked over time. The goal is to group together point tracks that correspond to independently moving objects. Without *a priori* knowledge, the definition of an independently moving object is somewhat arbitrary. For example, in a video of a person walking the intuitive result is to segment the tracks into two groups: one corresponding to points on the person, the other to points on the background. The person's arms and legs, however, may display motion that is significantly different from the motion of the torso. As a result, it is common to over-segment such videos into multiple regions corresponding to different body parts. Indeed, over-segmentation may even be desirable if the goal is to classify human actions. We therefore concentrate on minimizing errors in which two or more points from different objects are grouped together. Over-segmentation errors, in which a single object is split into multiple groups, are reported but considered less important as long as they do not become extreme.

3.1 SNMF for motion segmentation problem

In this paper, we propose a fast Semi-Nonnegative Matrix Factorization (SNMF) based motion segmentation algorithm for object-level segmentation of videos. To segment point tracks using SNMF, we represent motion data as a velocity history matrix. Given a video with F frames and P tracked point trajectories, the velocity history matrix X is a $2(F-1) \times P$ matrix:

$$X = \begin{bmatrix} \Delta x_1^1 & \Delta x_2^1 & \dots & \Delta x_P^1 \\ \Delta y_1^1 & \Delta y_2^1 & \dots & \Delta y_P^1 \\ \vdots & \vdots & \ddots & \vdots \\ \Delta x_1^{F-1} & \Delta x_2^{F-1} & \dots & \Delta x_P^{F-1} \\ \Delta y_1^{F-1} & \Delta y_2^{F-1} & \dots & \Delta y_P^{F-1} \end{bmatrix}$$

Here Δx , Δy are the x and y coordinates of the displacement vectors of the points. The subscripts denote the index of the point, while the superscripts denote the frame number. Hence Δx_n^m represents the displacement in x of point n from frame m to frame $m+1$. X is a $2(F-1) \times P$ matrix because the last frame F is not the starting point for any displacement vectors.

Semi-nonnegative matrix factorization (SNMF) factors X into two matrices F and G so as to minimize an error function $J(F, G)$:

$$J(F,G) = \min_{F,G} \|(X - FG^T)\|_F, \quad G \geq 0 \tag{1}$$

where X is the velocity history matrix defined above. F is a $2(F-1) \times r$ *component* matrix in which the r^{th} column is the r^{th} component. In essence, components represent trajectories. G is an $r \times P$ coefficients matrix. SMNF models the observed trajectories in X as a sum of components in F, with G providing the relative weights. The term *semi-nonnegative* reflects the constraint that while F is allowed to have negative components, G is constrained to be non-negative. In a low rank decomposition case, $r \ll min(F, P)$.

The non-negative constraint on G is based on the common observation that NMF methods can extract meaningful "parts" of ensemble data[8][10]. The non-negative constraint on coefficients restricts the linear combination on basis components to be "additive" rather than "substractive". In the motion segmentation scenario, the relative motion is the natural "part" to be separated from the ensemble motion data. Here the relative motion can be the foreground object motion relative from the camera, the relative motion between different objects, or even between the sub-parts of specific objects. By adding all the relative motion up we can recover the original ensemble motion data. Based on this consideration, we see SNMF as a promising framework for the motion segmentation problem.

In the case of missing data, we define a $2(F-1) \times P$ indicator matrix W that is 1 in the locations of valid data and 0 in the locations of missing values. We then seek to minimize:

$$J(F,G) = \min_{F,G} \|W \otimes (X - FG^T)\|_F, \quad G \geq 0 \tag{2}$$

where $\otimes$ denotes element-wise multiplication. One of the contributions of this paper is to extend SNMF to explicitly minimize $J(F,G)$ in the presence of missing data, as in the equation above.

In motion segmentation, columns of F are extracted components and columns of G are coefficients; F and G form a new representation for the trajectories. We set the number of columns of G to 3, to get a three dimensional compact representation for each trajectory.

Algorithm 1. SNMF with missing data

```
1: Initialize $F$ as a random matrix and $G$ as a random positive matrix respectively.
   Iteration for step 2 and step 3 until converge.
2: Updating $F$ by:
```

$$R = ((W \otimes X)G) \oslash ((W \otimes FG^T)G) \tag{3}$$

$$F = F \otimes R \tag{4}$$

```
3: Updating $G$ by:
```

$$R_1 = ((W^T \otimes X^T)F)^+ + ((W^T \otimes GF^T)F)^- \tag{5}$$

$$R_2 = ((W^T \otimes X^T)F)^- + ((W^T \otimes GF^T)F)^+ \tag{6}$$

$$G = G \otimes R_1 \oslash R_2 \tag{7}$$

3.2 SNMF with missing data

We propose a new algorithm for SNMF with missing data. The algorithm is an iterative updating algorithm that alternatively updates F and G to minimize $J(F,G)$ as given in equation 2. Please see Algorithm 1.

Here the $(\cdot)^+$ and $(\cdot)^-$ are the positive and negative parts of a matrix, defined as:

$$A^+ = (|A| + A)/2, \quad A^- = (|A| - A)/2 \tag{8}$$

Next we derive the update procedure. In every stage of the update procedure we alter one of F or G to reduce the cost function $J(F,G)$ while fixing the other. The cost function (equation 2 above) can be transformed as below:

$$\begin{aligned} J(F,G) =& \; \|W \otimes (X - FG^T)\|_F \\ =& \; Tr((W \otimes X)^T (W \otimes X) - 2(W^T \otimes X^T)(W \otimes FG^T) + (W^T \otimes GF^T)(W \otimes FG^T)) \\ =& \; Const. - 2\,Tr((W^T \otimes X^T)(W \otimes FG^T)) + Tr((W^T \otimes GF^T)(W \otimes FG^T)) \end{aligned} \tag{9}$$

The second item of (9) can be further transformed by

$$\begin{aligned} J_2 &= Tr((W^T \otimes X^T)(W \otimes FG^T)) \\ &= \sum_{k=1}^{n}((W^T \otimes X^T)(W \otimes FG^T))_{kk} \\ &= \sum_{k=1}^{n}\sum_{i=1}^{m}(W^T \otimes X^T)_{ki}(W \otimes FG^T)_{ik} \\ &= \sum_{i=1}^{m}\sum_{k=1}^{n}(W \otimes X \otimes W \otimes FG^T)_{ik} \\ &= Tr((W^T \otimes W^T \otimes X^T)FG^T) \end{aligned} \tag{10}$$

The optima of cost function $J(F,\cdot)$ is obtained at point where $\partial J/\partial F = 0$, so from (9),(10) we have

$$\frac{\partial J(F,\cdot)}{\partial F} = -2\ (W \otimes W \otimes X)G + \frac{\partial Tr((W^T \otimes GF^T)(W \otimes FG^T))}{\partial F} = 0 \tag{11}$$

Similar to (10), we derive the derivative of the third item of (9) as

$$\begin{aligned} \frac{\partial J_3}{\partial F_{pq}} &= \frac{\partial Tr((W^T \otimes GF^T)(W \otimes FG^T))}{\partial F_{pq}} \\ &= \frac{\partial \sum_{i=1}^{m}\sum_{k=1}^{n}(W \otimes W \otimes FG^T \otimes FG^T)_{ik}}{\partial F_{pq}} \\ &= \frac{\partial \sum_{i=1}^{m}\sum_{k=1}^{n} W_{ik}^2 (FG^T)_{ik}^2}{\partial F_{pq}} \end{aligned} \tag{12}$$

Only the pth row of FG^T are related to the F_{pq}, so we have

$$\begin{aligned} \frac{\partial J_3}{\partial F_{pq}} &= \frac{\partial \sum_{k=1}^{n} W_{pk}^2 (FG^T)_{pk}^2}{\partial F_{pq}} \\ &= 2\sum_{k=1}^{n} W_{pk}^2 (FG^T)_{pk} \frac{\partial (FG^T)_{pk}}{\partial F_{pq}} \\ &= 2\sum_{k=1}^{n} W_{pk}^2 (FG^T)_{pk} G_{kq} \end{aligned} \tag{13}$$

In matrices derivation form, that is

$$\frac{\partial J_3}{\partial F} = 2(W \otimes W \otimes FG^T)G \tag{14}$$

So finally we have

$$\frac{\partial J(F,\cdot)}{\partial F} = -2\ (W \otimes W \otimes X)G + 2(W \otimes W \otimes FG^T)G = 0 \tag{15}$$

It's worth noting that when X is a full data matrix (i.e. when. W is all 1s), $\partial J(F,\cdot)/\partial F$ is $-2XG+2FG^TG$, which leads to the standard least square update formulation[11]. To handle missing data, we modify the optimum condition (15) to be

$$F \otimes ((W \otimes FG^T)G) = F \otimes ((W \otimes X)G) \tag{16}$$

which leads to the fixed-point updating rule

$$F = F \otimes \frac{(W \otimes X)G}{(W \otimes FG^T)G} \tag{17}$$

Here the fraction is element-wise division, which was previously denoted as $\oslash$ above. It is a fixed point iteration that has limiting solution at the optimum of the cost function (2).

To derive the updating rule of G, we further solve the optimization problem with non-negative constraint on G. We combine the above analysis with [9], the original derivation of SNMF with complete data. We construct the Lagrangian function $L(G)$ as

$$L(G) = J(\cdot, G) + \sum_{j,k} \beta_{kj} G_{kj} \tag{18}$$

Here the $J(\cdot, G)$ follows definition (1). β_{kj} are Lagrangian multipliers that enforce non-negative. The analysis of $\partial J(\cdot, G)/\partial G$ is similar to that of $\partial J(F,\cdot)/\partial F$. The optimum condition is given by Kuhn-Tucker conditions:

$$\frac{\partial L(G)}{\partial G} = (-2\ (W^T \otimes X^T)F + 2(W^T \otimes GF^T)F) + \beta = 0 \tag{19}$$

Combine with complementary slackness condition $\beta \otimes G = 0$, we have

$$(-2\ (W^T \otimes X^T)F + 2(W^T \otimes GF^T)F) \otimes G = 0 \tag{20}$$

Although (20) is similar to (16), update similar to (17) can not be applied because of the non-negative constraint. Here we use an update rule similar to [9] as

$$G = G \otimes \frac{((W^T \otimes X^T)F)^+ + ((W^T \otimes GF^T)F)^-}{((W^T \otimes X^T)F)^- + ((W^T \otimes GF^T)F)^+} \tag{21}$$

Note that for any matrix A, $A = A^+ - A^-$ holds. So the cost function reaches its optimum (20) when (21) converges to limiting solution. G remains non-negative throughout all update stages.

It's worth noting that matrix factorization has inherit ambiguity that for any given factorization $X = FG^T$, there exists a family of equivalent factorization $X = FRR^{-1}G^T$. Here R can be any invertible matrix. To avoid this ambiguity, r consecutive rank-1 SNMF steps are performed for rank-r SNMF factorization. In each step we do rank-1 SNMF (2) in which both F and G are 1-column matrix. By normalizing F and multiplying back the normalization constant to G, the factorization ambiguity is avoided. The G obtained can be explained as "coefficients for the primary motion trend". Then the Residual $X^{'} = X - FG^T$ is feeded into the next step as input data matrix. The obtained r column coefficients G serves as the new compact r-dimensional trajectory representation.

Fig. 1 shows the extraction result for the proposed SNMF algorithm. The first row shows four sample frames from the Berkeley motion segmentation dataset. The second to fourth rows show the value of the first to the third column of G, which are extracted SNMF coefficients. In each sub-figure, we transform the magnitude of g^i to gray scale values. This illustration clearly shows the extracted coefficients separate foreground from background, as well as independently moving objects. The last row shows the actual segmentation performed by standard K-means clustering.

3.3 Segmentation Propagation

The proposed algorithm produces state-of-the-art results on 10 frame segments of videos while being two orders of magnitude faster than the algorithm by Brox and Malik. However, the accuracy drops quickly when the algorithm is used on longer frame sequences. This is because matrix factorization with missing data is a difficult non-convex problem, and no method is known that will always find the global optimum. The alternation optimization procedure we propose converges to local optima; as the portion of missing data increases and the search dimension becomes higher, the algorithm more and more frequently falls into local minima. We therefore propose a method called segmentation propagation. The outline of this method is listed in Algorithm 2.

The idea behind segmentation propagation is that the point tracker[12] sustains well. The majority of tracked points last longer than 10 frames. So while SNMF works well on short sequences of data, the segmentation information it extracts should be able to be propagated across sliding windows by the majority

Algorithm 2. Segmentation Propagation

```
1: Do short time SNMF segmentation in every sliding window.
   $N = n * t$,  $N$: total number of segmentation, $n$: number of segmentation per time
   window, $t$: number of time window
2: Construct $N \times N$ affinity matrix by adding up total segmentation co-occurrence
   within every track
3: Do Spectral Clustering to make $K$ segmentation groups
4: Reassign each segmentation with its group labels
5: Within every track of length $l$ ($l < t$ for partial tracks), vote for the most frequent
   label as track label
```

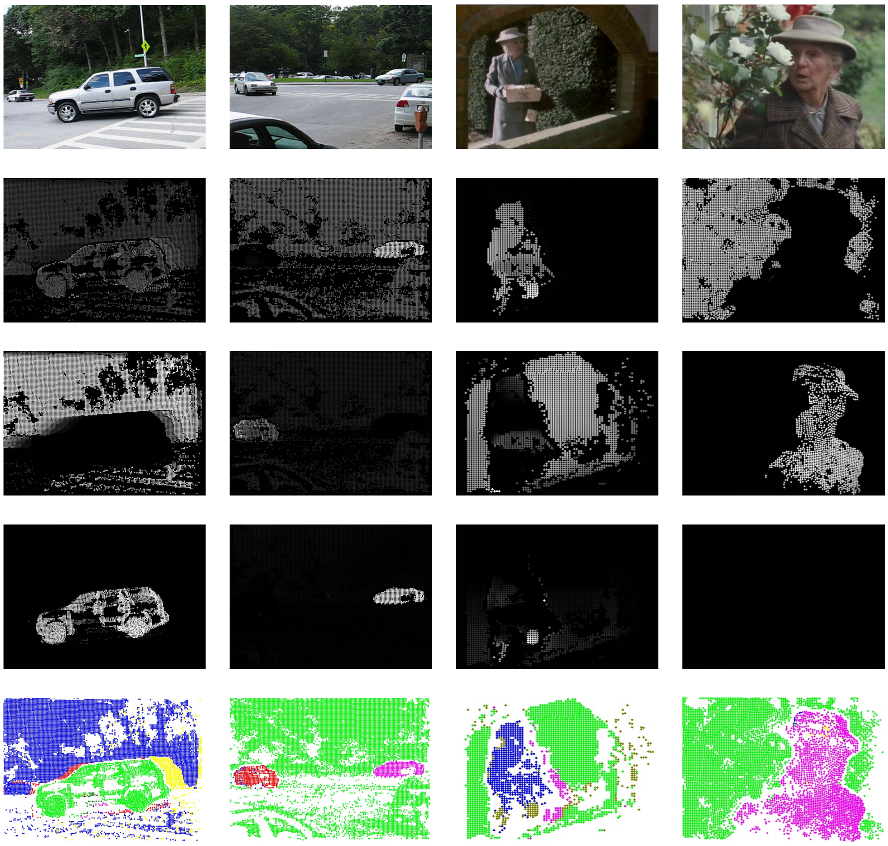

Fig. 1. The first row is four sample frame from the Berkeley motion segmentation dataset. The second to fourth row shows the value of the first to the third column of G, which are extracted SNMF coefficient. In each sub-figure, we transform the magnitude of g^i to the gray scale value.The last row of sub-figures shows the actual segmentation.

of interest points that are tracked through multiple time windows. If many points inside one time window form a segment and can be tracked into the next time window and form another segment, there is good chance that the two segments come from the same object in adjacent time windows. So we build an affinity matrix of all segments throughout the video by counting co-occurrences of segmentations related to each track. We then merge segments by standard spectral clustering. Both tracking and segmentation introduce error. It is common for one track to have different merged segmentation labels in different time interval. We decide the label of the whole track by voting with winner-take-all rule.

In step 1 of Algorithm 2, we segment the target video into multiple short time windows. We use 10 frame time windows throughout the experiments in this paper. In order to introduce better sustainability of trajectories, we start a 10-frame time window every 5 frames, so every trajectory appears in at least

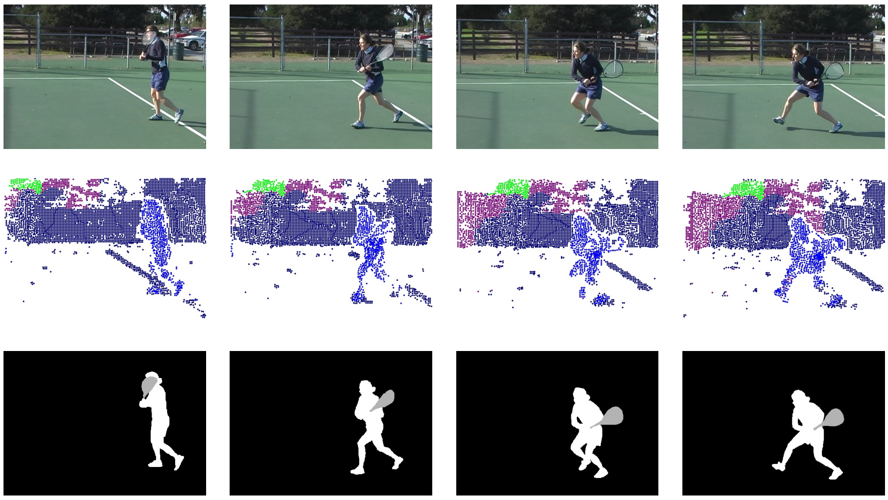

Fig. 2. Sample segmentation results on Tennis by Segmentation Propagation

two time windows. Hence we perform $F/5$ short-time SNMF segmentations in a video of length F.

By step 1 we have multiple initial segmentation labels (at least 2) for each track. In step 2, we construct an $N \times N$ affinity matrix A, where N is the total number of segments across time and $A[i, j]$ is the number of tracks that occur in both segments i and j. The co-occurrences do not have to be contiguous in time. Step 3 does standard spectral clustering based on this affinity matrix to group initial segmentation to K groups, where K is the number of final segments. In this paper we set $K = 11$. As frame length increases, this step becomes the most time consuming stage since the time complexity of spectral clustering is $O(N^3)$. In step 4, each initial segmentation is re-assigned to its corresponding group label.

After step 4, we get t labels for a trajectory that lasts t time windows. Ideally, all segments of a trajectory should be assigned to a single group. However both tracking and segmentation introduce error. This error can even propagate through the affinity graph in the clustering stage, so it is common for one trajectory to be assigned to multiple segment groups. In step 5 we simply take the most frequent assignment to be the final label of the trajectory. Assuming that SNMF correctly segments most time windows, voting allows the correct segmentations to overwhelm the error. In practice we do observe some short time SNMF segmentation errors, but most of the errors are suppressed by the winner-take-all voting strategy. Fig 2 shows sample segmentation results.

4 Experimental Evaluation

In this section we report our result on the Berkeley motion segmentation dataset [1], following the comparison protocol in [1]. Quantitative results are generated using tools publicly provided by the authors of [1]. We also report execution speeds to illustrate the efficiency of our method. We built our new SNMF algorithm within the framework of Python Matrix Factorization Library(PyML) [13].

The Berkeley motion segmentation dataset contains 26 video sequences, 12 of which were chosen from the Hopkins 155 database[14]. Annotation is provided for selected frames. Test methods are expected to run in four evaluation modes, evaluating the first 10, 50, and 200 frames, and evaluating all frames.

In our first experiment, we run the proposed SNMF algorithm directly on 10 frame segmentation(evaluation mode 1). In this case we only have 2 parameters. We set the number of SNMF bases $r = 3$, and the number of clusters $n = 6$. For the other 3 longer comparisons we run the SNMF algorithm with 10 frame windows beginning every 5 frames. The number of final segments in the segmentation propagating procedure is set to 11.

We first compare our method to the current state-of-the-art method proposed in [1]. To make a more complete comparison, we also use the performance numbers reported in [1] for other contemporary algorithms. They are: Generalized PCA (GPCA)[2], Local Subspace Affinity (LSA)[3], RANSAC, and Agglomerative Lossy Compression(ALC)[4]. Among them, ALC is the only method that can deal with incomplete tracks. Table 1 shows the comparison on all four evaluation modes on all 5 evaluation metrics: **Density** is the number of points on with segmentation label is reported. We report data on the same point trajectories as in [14], so we have the same densities as they do. **Overall error** is the percent of wrong assigned label over the total number of labeled points. **Average error** is similar to the overall error but is averaged across regions. This number are usually much higher than the overall error because a full penalty of 100% error will be reported for the region missed by segmentation. **Over-segmentation error** is defined as the average number of clusters merged in production above "precision" measure versus total number. The last index is **number of extracted objects with less than 10% error**. It is a measure of high quality object level segmentation.

From table 1 we can see that the proposed SNMF algorithm gets the lowest overall segmentation error on 10 frame comparisons. For the other 3 comparisons(see Table 2), our method(combined) has 1%-3% higher error rate compared to [1] but is still much better than ALC. In all 4 comparisons our method gets the lowest average segmentation error. Since the major source of average segmentation error is the penalty for missed ground truth objects, our method generally tends to produce fewer missing detections. In the number of extracted objects with less than 10% error comparison, which corresponding to high quality object level segmentation, SNMF stills wins the 10 frame comparison while falling behind with [1] in the other three cases. In general, our segmentation accuracy is comparable to that of [1].

While the proposed method produces competitive segmentation accuracy, it has a lower over-segmentation error rate than Brox and Malik's method. However, their method has two explicit over-segmentation/merge stages while we don't perform postpocessing. Also, their method uses spatial location of trajectories to regularize segmentation. In contrast, our method use pure motion information to perform the segmentation task.

Perhaps the most important advantage of our method is execution speed. Table 3 and table 4 provide comparisons of sample running speeds of the benchmarked algorithms. In the 10 frame comparison, we run SNMF algorithm only. In comparisons involving more frames, we run both SNMF and segmentation propagation algorithm. It is easy to see from table 3 and table 4 that our methods is over two orders of magnitude faster than [1]. In addition, our code is written in Python while Brox and Malik's code is C++, which gives us every expectation that our speed can be further improved. All the other algorithms in Table 3 are much slower, except for RANSAC, which cannot handle missing data.

Table 1. Evaluation results on 10 frame comparison of all 26 videos in the Berkeley motion segmentation. Legend: $\epsilon_{overall}$: overall error; $\epsilon_{average}$:average error; O.S: over-segmentation error; E.O.: extracted foreground region with less than 10% error

	Density	$\epsilon_{overall}$	$\epsilon_{average}$	O.S.	E.O.
First 10 frames(26 sequences)					
SNMF	3.32%	6.98%	17.45%	3.36	26
Brox & Malik[1]	3.32%	7.66%	25.33%	0.5	24
GPCA	2.98%	14.28%	29.44%	0.65	12
LSA	2.98%	19.69%	39.76%	0.92	6
RANSAC	2.98%	13.39%	26.11%	0.50	15
ALC corrupted	2.98%	7.88%	24.05%	0.15	26
ALC incomplete	3.34%	11.20%	26.73%	0.54	19

Table 2. Evaluation results on 50, 200, and all frames of all 26 videos in the Berkeley motion segmentation. Legend: $\epsilon_{overall}$: overall error; $\epsilon_{average}$:average error; O.S: over-segmentation error; E.O.: extracted foreground region with less than 10% error

	Density	$\epsilon_{overall}$	$\epsilon_{average}$	O.S.	E.O.
First 50 frames(15 sequences)					
our method(combined)	3.26%	9.93%	17.23%	7.20	8
Brox & Malik[1]	3.26%	6.91%	32.45%	0.46	9
ALC corrupted	1.53%	7.91%	42.13%	0.36	8
ALC incomplete	3.26%	16.42%	49.05%	6.07	2
First 200 frames(7 sequences)					
our method(combined)	3.43%	8.47%	25.75%	6.85	3
Brox & Malik[1]	3.43%	6.86%	32.03%	2.71	7
ALC corrupted	0.20%	0.00%	74.52%	0.40	1
ALC incomplete	3.43%	19.33%	50.98%	54.57	0
All available frames(26 sequences)					
our method(combined)	3.30%	7.41%	23.79%	7.34	23
Brox & Malik [1]	3.30%	6.52%	27.31%	2.07	29
ALC corrupted	0.99%	5.32%	52.76%	0.10	15
ALC incomplete	3.30%	14.93%	43.14%	18.80	5

Table 3. Speed comparison on the people1 sequence(first 10 frames) on 6 methods

	tracks	time
SNMF	15486	1.36s
Brox & Malik[1]	15486	497s
GPCA	12060	2963s
LSA	12060	38614s
RANSAC	12060	15s
ALC	957	22837s

Table 4. Speed comparison results on 4 different videos with different frame length

Sequence	People1	Cars4	Tennis	Marple7
Tracks	15486	17224	31706	163266
Frames	10	50	200	528
Brox et al's	497 sec	38 min	98 min	33 hr
Our method	1.36 sec	20.82 sec	132 sec	18 min

5 Conclusion

In this paper we present two algorithms that efficiently solve motion segmentation problem with dense, partial tracks. We develop a fast SNMF-based method that achieves state-of-the-art results on the Berkeley motion segmentation dataset for short time intervals. We then present a new approach to broadcast local time interval results along the timeline, achieving similar performance to the best known methods with two orders of magnitude speed up. In addition, the SNMF method itself is extended to handle missing data. In this work, only motion information is used. In the future, postprocessing methods will be explored to handle additional available information such as spatial location of trajectory and segments regularity to get even better results for object level segmentation in video.

References

1. Brox, T., Malik, J.: Object Segmentation by Long Term Analysis of Point Trajectories. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 282–295. Springer, Heidelberg (2010)
2. Vidal, R., Ma, Y.: A Unified Algebraic Approach to 2-D and 3-D Motion Segmentation. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3021, pp. 1–15. Springer, Heidelberg (2004)
3. Yan, J., Pollefeys, M.: A General Framework for Motion Segmentation: Independent, Articulated, Rigid, Non-rigid, Degenerate and Non-degenerate. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 94–106. Springer, Heidelberg (2006)
4. Rao, S., Tron, R., Vidal, R., Ma, Y.: Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE (2008)
5. Fan, Z., Zhou, J., Wu, Y.: Multibody grouping by inference of multiple subspaces from high-dimensional data using oriented-frames. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 91–105 (2006)
6. Elhamifar, E., Vidal, R.: Sparse subspace clustering. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2790–2797. IEEE (2009)

7. Cheriyadat, A., Radke, R.: Non-negative matrix factorization of partial track data for motion segmentation. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 865–872. IEEE (2009)
8. Lee, D., Seung, H., et al.: Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)
9. Ding, C., Li, T., Jordan, M.: Convex and semi-nonnegative matrix factorizations. IEEE Transactions on Pattern Analysis and Machine Intelligence 32, 45–55 (2010)
10. Hoyer, P.: Non-negative matrix factorization with sparseness constraints. The Journal of Machine Learning Research 5, 1457–1469 (2004)
11. Hartley, R., Schaffalitzky, F.: Powerfactorization: 3d reconstruction with missing or uncertain data. In: Australia-Japan Advanced Workshop on Computer Vision, vol. 74, pp. 76–85 (2003)
12. Sundaram, N., Brox, T., Keutzer, K.: Dense point trajectories by gpu-accelerated large displacement optical flow. In: Proceedings of the 11th European Conference on Computer Vision, pp. 438–451 (2010)
13. Thurau, C.: PyMF: Python matrix factorization library (2010), `http://code.google.com/p/pymf/`
14. Tron, R., Vidal, R.: A benchmark for the comparison of 3-d motion segmentation algorithms. In: IEEE Conference on Computer Vision and Pattern Recognition 2007, pp. 1–8. IEEE (2007)