Robust Trajectory Clustering for Motion Segmentation

Feng Shi[1], Zhong Zhou[1,*], Jiangjian Xiao[2], Wei Wu[1]
[1]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China
[2]Ningbo Industrial Technology Research Institute, CAS, China

Abstract

Due to occlusions and objects' non-rigid deformation in the scene, the obtained motion trajectories from common trackers may contain a number of missing or mis-associated entries. To cluster such corrupted point based trajectories into multiple motions is still a hard problem. In this paper, we present an approach that exploits temporal and spatial characteristics from tracked points to facilitate segmentation of incomplete and corrupted trajectories, thereby obtain highly robust results against severe data missing and noises. Our method first uses the Discrete Cosine Transform (DCT) bases as a temporal smoothness constraint on trajectory projection to ensure the validity of resulting components to repair pathological trajectories. Then, based on an observation that the trajectories of foreground and background in a scene may have different spatial distributions, we propose a two-stage clustering strategy that first performs foreground-background separation then segments remaining foreground trajectories. We show that, with this new clustering strategy, sequences with complex motions can be accurately segmented by even using a simple translational model. Finally, a series of experiments on Hopkins 155 dataset and Berkeley motion segmentation dataset show the advantage of our method over other state-of-the-art motion segmentation algorithms in terms of both effectiveness and robustness.

1. Introduction

Motion segmentation is not only an important preprocessing step for many computer vision applications, but also an essential issue in dynamic scenes understanding. In this paper, we focus on sparse methods in motion segmentation. That is, given a set of feature point trajectories tracked from a video, one seeks to cluster the trajectories according to their corresponding motions.

Due to containing long-term motion cues, feature trajectories are very suitable to be used to partition video into temporally consistent clusters [3]. However, their long duration also leads their sensitivity to object occlusions and tracking error. Therefore, the input trajectories to motion segmentation algorithm are often contaminated by missing and corrupted entries. How to get robust results from the low quality input is still a challenge in the field of motion segmentation. For solving this problem, some methods [7, 10, 13] are proposed to repair the pathological trajectories by computing their sparse or low-rank representation with respect to a dictionary formed by all other trajectories. However, this kind of algorithms has an inherent drawback that requires a sufficiently large set of complete and uncorrupted trajectories, thus it becomes unsuitable for the real sequences which may have a large number of missing and corrupted data. Another kind of motion segmentation algorithms [8, 4, 3, 9, 12, 11, 6] which does not require any completion of trajectories is recently proposed, where certain motion models are used for trajectory clustering. Thus, although it has significant advantage in handling incomplete trajectories, they may still fail on the sequences when the objects motion deviates the employed motion models in the algorithms.

Obviously, there is a need to develop a motion segmentation algorithm that can handle not only the missing and corrupted data but also complex motions both typically of real tracking process. The main contribution of this paper is a trajectory clustering algorithm that can achieve this objective in a consistent manner. The effectiveness and robustness of our method are mainly attributed to the proper exploitation of temporal and spatial characteristics from feature point trajectories. We notice that trajectories of feature points from most nature deforming objects are smooth and continuous. Therefore, it implies that such temporal smoothness of feature trajectories can be compactly represented by predefined basis vectors. Based on this analysis, we select DCT as predefined bases to approximate feature trajectories, and then design a non-linear optimization scheme that can effectively decompose the input trajectories into a set of DCT basis vectors and corresponding coefficients. The introduction of DCT results in a significant reduction in unknowns during the estimation with little in-

*Corresponding author. E-mail: zz@vrlab.buaa.edu.cn.

formation loss, and thus ensures the ability of estimated coefficients to capture inherent similarities between trajectories even with a large amount of missing data and corrupted noises.

We also notice that trajectories of foreground and background may have different characteristics in spatial distribution, that is, background trajectories ($i.e.$ static parts of a scene) are usually dispersed all over the scene while foreground trajectories ($i.e.$ moving objects) of the same motion are usually compact. Based on this observation, we present a two-stage clustering strategy which first separates foreground trajectories from background trajectories based on motion subspaces constraints and then divides the foreground trajectories into different partitions using spectral clustering. By this way, our method can even use a simple translational model to obtain highly robust segmentation results for those sequences with complicated rigid or nonrigid object motions.

Finally, we evaluate our method and state-of-the-art trajectory clustering algorithms on both the Hopkins 155 dataset [16] and the Berkeley motion segmentation dataset [3]. The experimental results demonstrate that our method significantly outperforms state-of-the-art methods in terms of both effectiveness and robustness. It is worth noting that our work is the first in trajectory clustering to conduct experiments on both Hopkins and Berkeley datasets that have quite different characteristics.

2. Related Work

During the past two decades, numerous trajectory clustering algorithms have been proposed for motion segmentation. In this field, the most important class of algorithms is multi-body factorization methods [5, 17, 13, 7, 10], and their underlying idea is using motion subspaces constraints, where the trajectories of the same motion can span a low-dimensional linear subspace and of different motions may distribute in different subspaces. Based on this, segmenting a video sequence containing various types of motion ($e.g.$ independent, articulated, rigid, non-rigid or any combination of them) can be cast as a subspace separation problem to be solved. Most early subspace-based methods, such as [5, 17], assume that all input trajectories are complete and do not contain gross errors. However, the tracking failure of feature points is very common in real-world automatic tracking, causing trajectories to have missing entries (incomplete trajectories) or some entries with gross errors (corrupted trajectories). To handle the two kinds of pathological trajectories, Rao $etal.$ [13] proposed Agglomerative Lossy Compression (ALC) that repairs a trajectory with missing or corrupted entries prior to subspace separation by computing its sparse representation with respect to all other trajectories. The method proposed by Elhamifar and Vidal [7], known as sparse subspace clustering (SSC), uses a similar strategy as ALC to handle pathological trajectories, and uses the sparse coefficients of trajectories to build the affinity matrix for spectral clustering. Liu $etal.$ [10] proposed an alternative approach that uses low-rank representation instead of sparse representation to correct the corruption in trajectories. The three methods, although have been successfully used to segment sequences with a certain amount of missing and corrupted data, have an inherent drawback, which requires an assumption that each motion should have a sufficiently large subset of complete and uncorrupted trajectories. If input data is highly fragmented or grossly corrupted, which are common phenomena in real-world tracking, the performance of this kind of methods will deteriorate drastically.

In recent years, a few trajectory clustering methods [8, 4, 3, 9, 12, 11, 6] which do not rely on motion subspaces constraints are proposed for motion segmentation. These methods usually utilize a motion model to measure similarities between all trajectories, and then perform a common clustering technique to segment trajectories. Among them, [4, 3, 9, 11] use the velocity information to represent trajectories similarities, thus are translational model-based methods. [8, 12, 6] use higher order motion models to measure trajectories similarities. Compared to multi-body factorization methods, the motion model-based methods have significant advantage in handling incomplete trajectories because they do not require any completion of the input trajectories. In [8, 3, 9, 12, 6], similarities between trajectories are computed only using their available entries. In [4, 11], an iterative optimization algorithm is proposed to decompose a velocity matrix computed from the incomplete trajectories, and the resulting components are used as representations of trajectories for clustering. A limitation of this kind of method is that it can only compute the similarity of trajectories based on the underlying motion model. As a result, it will often lead to poor performance when they were applied to segment sequences containing motions that deviate from their motion models.

3. Proposed Algorithm

In this paper, we suppose that trajectories of P feature points have been obtained by running some existing trackers, $e.g.$ the KLT tracker [14], on a video with F frames. Let $T(p) = (x_1^p, y_1^p, ..., x_F^p, y_F^p)^T$ denotes the p^{th} trajectory, where x_f^p and y_f^p are the X and Y coordinates of the p^{th} point at frame f. Our goal is to partition the P trajectories into different groups according to their corresponding motions. Broadly, our algorithm proceeds as summarized in Algorithm 1. We next discuss each step of our algorithm in detail.

Algorithm 1 Trajectory Clustering for Motion Segmentation

1.Project $T(p), p = 1, ..., P$ to low-dimensional representation $C(p), p = 1, ..., P$.
2.Exploit motion subspaces constraints to perform foreground-background separation on $C(p)$.
3.Apply spectral clustering to $C(p), p \in foreground$ to generate clusters in foreground.
4.Label $T(p), p = 1, ..., P$ according to the clustering result of $C(p), p = 1, ..., P$.

3.1. Matrix Factorization with Temporal Smoothness Constraint

To project the trajectory data into a lower dimensional space, where trajectories belonging to different motions can be distinguished more easily, we first form a measurement matrix $W \in R^{F\times 2P}$ by arranging the X and Y coordinates of all trajectories vertically:

$$W = \begin{pmatrix} x_1^1 & y_1^1 & \cdots & x_1^P & y_1^P \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_F^1 & y_F^1 & \cdots & x_F^P & y_F^P \end{pmatrix} \quad (1)$$

Assume rank(W) = r, W can be decomposed as: $W = BC$, where the columns of $B \in R^{F\times r}$ are the bases of the column space of W, and the $2p^{th} - 1$ and $2p^{th}$ columns of $C \in R^{r\times 2P}$ are the corresponding coefficients of the X and Y coordinates of $T(p)$. We then denote the p^{th} column of C by c^p, and form the low-dimensional representation of $T(p)$ with $C(p) = ((c^{2p-1})^T, (c^{2p})^T)^T$. As discussed in the next subsection, when appropriate bases are chosen, $C(p)$ can offer a reliable way to measure the similarity of trajectories. So, in the next subsection, we are going to perform clustering on $C(p), p = 1, ..., P$, and then to label the trajectories accordingly.

Generally speaking, the difficulty of obtaining desirable results by factorizing a matrix depends heavily on the quality of this matrix, which can be measured by the amount of its missing and corrupted data. Specific to our method, in the presence of considerable quantities of incomplete and corrupted trajectories from automatic feature tracking, it is challenging to decompose W into components to capture the inherent similarity of trajectories. To solve this problem, we propose to exploit an inherent property of most natural deforming objects, temporal smoothness, to introduce a constraint on the bases of trajectories.

In our setup, the temporal smoothness of $T(p), p = 1, ..., P$ suggests that the values in each column of W vary smoothly over time, thus can be considered as samples of a smooth signal. There are a number of predefined bases which can approximate smooth signals compactly, and among them, the DCT bases have been proved to be particularly suitable for motion trajectories [1, 2]. Figure 1 also shows that the trajectory bases learned from Hopkins 155 dataset[1] very closely resemble the DCT bases. We therefore use a linear combination of $d(r < d \ll F)$ DCT vectors to approximate the column bases of W. Then W can be factorized as:

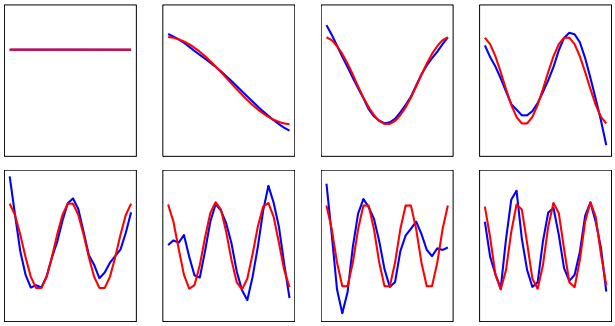

Figure 1. Comparison of the DCT bases (red) with the PCA-learned trajectory bases (blue) of the Hopkins 155 dataset. The first 8 DCT and PCA bases (ordered left-to-right, top-to-bottom) are plotted, and some of the bases are multiplied by -1 for better visual comparison.

$$W = BC = \Omega_d X C = \begin{pmatrix} \theta^1 & \cdots & \theta^d \end{pmatrix} XC \quad (2)$$

where θ^j denotes the j^{th} DCT basis, and the unknown factor X describes the bases of trajectories in DCT domain.

Obviously, the introduction of the DCT basis vectors results in a significant reduction in unknowns with almost no loss in representation when estimating $C(p)$, and thus corresponding numerical stability.

We denote the p^{th} column of W by $w^p \in R^F$, and use a F^p-dimension vector $\tilde{w}^p$ ($F^p \leq F$) to denote the observed entries in w^p. Then, we define $\Pi^p \in R^{F^p\times F}$ as a row-amputated identity matrix such that $\Pi^p\Omega_d X$ has the rows in $\Omega_d X$ that correspond to the rows of entries in $\tilde{w}^p$. Next, based on the L-2 distance between the observed and estimated trajectories, the following error function on the DCT coefficients matrix X and the trajectories coefficients matrix C is obtained:

$$f(X, C) = \frac{1}{2}\sum_{p=1}^{2P} \|\tilde{w}^p - \Pi^p\Omega_d X c^p\|_2^2 \quad (3)$$

For computing X and C, we design an iterative nonlinear optimization algorithm as described in Algorithm 2 to minimize Eq. (3). The optimization strategy of Algorithm 2 is: alternately update C with least squares method and X with Gaussian-Newton method until convergence. Eq. (4) and (5) in Algorithm 2 are of the form (see complete derivation in Appendix):

$$vec(C) = (\Psi^T\Psi)^{-1}\Psi^T vec(\tilde{W}) \quad (4)$$

[1]In the test, we choose sequences longer than 24 frames from the Hopkins 155 dataset, and perform PCA on the first 24 frames of these sequences to compute the trajectory bases.

Algorithm 2 Matrix Factorization with Temporal Smoothness Constraint

Input: $\delta = 10^{-4}$, $X = [I_r, 0]^T$.
Repeat
  Compute C with Eq. (4).
  Compute gradient (g) and Hessian (H) with Eq. (5).
  Repeat
    $\delta = \delta \times 10$.
    Compute ΔX with $vec(\Delta X) = (H + \delta I)^{-1} g$.
  Until $f(X - \Delta X, C) < f(X, C)$
  $X = X - \Delta X$, $\delta = \delta \times 10^{-2}$.
  Orthogonalize the columns of X.
Until convergence.

where Ψ is a block diagonal matrix which is formed by $\Pi^p \Omega_d X, p = 1, ..., 2P$, $vec(\tilde{W}) = ((\tilde{w}^1)^T, ..., (\tilde{w}^{2P})^T)^T$, $vec(C) = ((c^1)^T, ..., (c^{2P})^T)^T$.

$$g = -\sum_{p=1}^{2P} (J^p)^T r^p, H = \sum_{p=1}^{2P} (J^p)^T J^p \tag{5}$$

where $J^p = (c^p)^T \otimes (\Pi^p \Omega_d)$, $r^p = \tilde{w}^p - \Pi^p \Omega_d X c^p$, $\otimes$ is the Kronecker product.

3.2. Two-Stage Clustering

After the matrices X and C are computed, we intend to perform cluster analysis on $C(p), p = 1, ..., P$, and then to label $T(p), p = 1, ..., P$ accordingly.

Let S denote the product of X and C, and $s_i \in R^{2P}$ denote the i^{th} row of S. From Eq. (2), we have:

$$\begin{pmatrix} s_1^T & \cdots & s_d^T \end{pmatrix}^T = \begin{pmatrix} \theta^1 & \cdots & \theta^d \end{pmatrix}^T W \tag{6}$$

Then, the first row of S equals to

$$s_1 = (\theta^1)^T W = \sqrt{F} \left(\overline{w^1}, \overline{w^2}, \cdots, \overline{w^{2P-1}}, \overline{w^{2P}} \right) \tag{7}$$

where $(\overline{w^{2p-1}}, \overline{w^{2p}}) = (\sum_{f=1}^F x_f^p, \sum_{f=1}^F y_f^p)/F$ is the centroid of $T(p)$. Thus, s_1 can represent the average spatial location of $T(p), p = 1, ..., P$.

Subsequently, combining Eq. (7) with (2), we have:

$$\sum_{j=2}^{d} \theta^j s_j = W - \theta^1 s_1 = \frac{1}{F} \begin{pmatrix} \sum_{j=1}^F \overrightarrow{x}_{j1}^1 & \sum_{j=1}^F \overrightarrow{y}_{j1}^1 & \cdots & \sum_{j=1}^F \overrightarrow{x}_{j1}^P & \sum_{j=1}^F \overrightarrow{y}_{j1}^P \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum_{j=1}^F \overrightarrow{x}_{jF}^1 & \sum_{j=1}^F \overrightarrow{y}_{jF}^1 & \cdots & \sum_{j=1}^F \overrightarrow{x}_{jF}^P & \sum_{j=1}^F \overrightarrow{y}_{jF}^P \end{pmatrix} \tag{8}$$

where $\overrightarrow{x}_{ij}^p = x_j^p - x_i^p$, $\overrightarrow{y}_{ij}^p = y_j^p - y_i^p$ are respectively the displacements in X and Y coordinates of point p from frame i to j. Thus, $\sum_{j=2}^d \theta^j s_j^T$, and hence $(s_2, ..., s_d)^T$, can capture the translational information of $T(p), p = 1, ..., P$.

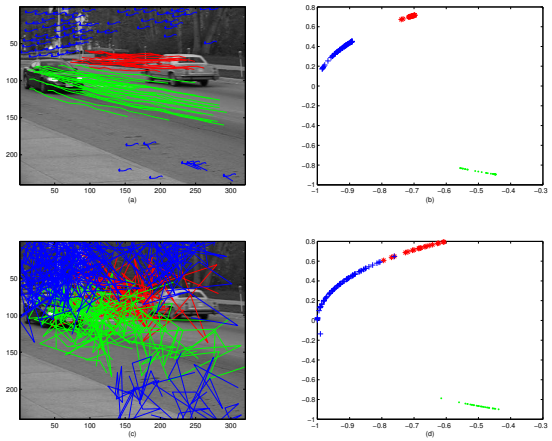

Figure 2. The intermediate results of our method on the sequence 'cars2_06'. (a)&(c) Original trajectories & trajectories with 50% missing and 100% corrupted entries (adding Gaussian noise with zero mean and variance $0.1\|W\|_F$). Three colors are employed to illustrate ground-truth segmentation of the trajectories. (b)&(d) Visualization of two-dimensional Laplacian eigenmaps projection of $C(p)$ of the trajectories in (a)&(c) respectively.

Next, utilizing SVD, we decompose S into $U \in R^{d \times r}$, $D \in R^{r \times r}$ and $V^T \in R^{r \times 2P}$, and reset the matrix B and C into $\Omega_d U D$ and V^T, respectively. From $C = (D^{-1} U^T) S$, it can be deduced that $C(p) = ((c^{2p-1})^T, (c^{2p})^T)^T$ is the weighted sum of spatial and translational information of $T(p)$, thus can be used to distinguish trajectories belonging to different translational motions. Figure 2 illustrates the computed S and $C(p), p = 1, ..., P$ in our method for the sequence 'cars2_06' from the Hopkins 155 dataset. The sequence contains three translational motions. It can be seen from Figure 2 that, by performing clustering on $C(p), p = 1, ..., P$, the three translational motions can be separated from each other clearly even with severe data missing and noises.

Now a question arises naturally: how to utilize $C(p), p = 1, ..., P$ to distinguish trajectories belonging to different non-translational motions?

To answer this question, we need to classify trajectories as foreground and background, which are respectively induced by the motions of camera and objects in the scene. Note that foreground and background trajectories have different spatial characteristics, that is, background trajectories are usually distributed over the scene while foreground trajectories belonging to the same motion are usually spatially close to each other. To show the differences of foreground and background trajectories resulted from their spatial characteristics, we plot the computed S and $C(p), p = 1, ..., P$ for the sequence '1R2RCR' of Hopkins 155 dataset in Figure 3. This sequence contains three

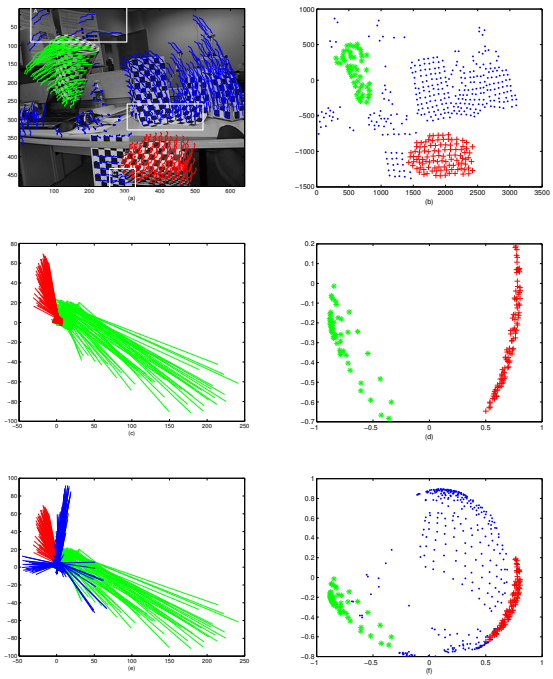

Figure 3. The intermediate results of our method on the sequence '1R2RCR' of Hopkins 155 dataset. (a) Input trajectories and ground-truth segmentation, red and green denote foreground clusters while blue denote background cluster. (b) Visualization of s_1 for all trajectories. (c)&(e) Visualization of $(s_2^T, ..., s_d^T)^T$ for foreground trajectories & foreground trajectories and background trajectories in rectangle 'A', 'B', and 'C'. (d)&(f) Visualization of two-dimensional Laplacian eigenmaps projection of $C(p), p \in foreground$ & $C(p), p = 1, ..., P$.

rotational motions which come from the camera and two foreground objects. It can be seen from Figure 3(b) and 3(c) that the spatial and translational information both exhibit tight and well-separated clusters for foreground trajectories. Consequently, as Figure 3(d) shows, clustering on $C(p), p = 1, ..., P$ can easily distinguish trajectories belonging to different non-translational foreground motions. In [3], Brox $etal.$ also indicated that translational model can offer a good approximation for spatially close points of the same non-translational motion. However, this argument is unsuitable for background trajectories since they are usually distributed in space. From Figure 3(e), we can see that the translational information of background trajectories in rectangle 'A', 'B', and 'C' approaches that of foreground trajectories. As a result, as Figure 3(f) shows, it is difficult to directly use $C(p), p = 1, ..., P$ to cluster trajectories in a video sequence with non-translational background motion.

To circumvent this problem, we have developed the following two-stage clustering, which first separates foreground from background based on motion subspaces constraints then segments foreground using spectral clustering, to segment $C(p), p = 1, ..., P$:

1. Compute affinity matrix A for $C(p), p = 1, ..., P$:

$$A(i, j) = exp(-\|C(i) - C(j)\|_2) \tag{9}$$

2. Apply spectral clustering to A to segment all trajectories into 2 clusters, and choose the one with lower dimension as background[2].

3. Iterate the following two steps until convergence:

   (a) Compute the bases of background subspace by performing SVD on the matrix formed by $C(p), p \in background : N = (\mu_1, \mu_2, \mu_3, \mu_4)$.

   (b) Compute projection error of all trajectories to background subspace: $\epsilon(p) = \|(I_{2r} - N(N)^+)C(p)\|_F$, then apply K-means to $\epsilon(p)$ to repartition all trajectories into foreground and background.

4. Compute projection error of foreground trajectories to foreground subspace, and reject the trajectories with projection error greater than a threshold λ as outliers.

5. Compute affinity matrix for $C(p), p \in foreground$ with Eq. (9), and apply spectral clustering to the affinity matrix to generate clusters in foreground.

In our two-stage clustering, step 2 provides a good initialization for the following iteration, thus leads to a superior convergence performance of the foreground_background segmentation (usually needs less than 10 iterations to converge). Figure 4 illustrates the segmentation results of our two-stage clustering on the sequence '1R2RCR' from Hopkins 155 dataset.

4. Experiments

In this section, we evaluate our method on both the Hopkins 155 dataset [16] and the Berkeley motion segmentation dataset [3] by comparing with state-of-the-art trajectory clustering algorithms. As most trajectory clustering algorithms [4, 7, 11, 13], we assume the cluster number is given. For the Berkeley dataset, the number is fixed to 5. In all runs of our experiments, we set the number of DCT bases $d = min(max(0.1 \times F, 15), 5)$, $\lambda = 0.5$. As to the rank of W, we first attempt different values of $r \in \{2, ..., d\}$ in our method, and use sum of projection error of each trajectory to its motion subspace, which is computed by SVD on all trajectories in the cluster it belonged, to measure segmentation quality and choose one with the fewest error as the best.

[2]The dimensions of the clusters are computed by SVD.

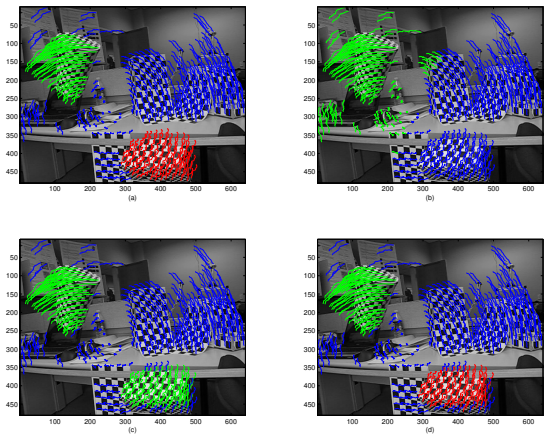

Figure 4. The segmentation results of our method on the sequence '1R2RCR'. (a) Input trajectories and ground-truth segmentation, red and green denote foreground clusters while blue denote background cluster. (b)&(c) Results of step 2&3 of our two-stage clustering algorithm. (d) Final result of our clustering algorithm.

Table 1. Misclassification rates (%) for the sequences with three motions in Hopkins 155 dataset.

Method	ALC	SSC	NNMF	MSMC	Ours'
Checkerboard, 26 sequences					
Mean	5.20%	2.97%	19.38%	8.30%	**2.44%**
Median	0.67%	0.27%	18.08%	0.93%	**1.29%**
Traffic, 7 sequences					
Mean	7.75%	0.58%	0.10%	0.17%	**0.05%**
Median	0.49%	0.00%	0.00%	0.00%	**0.00%**
Articulated, 2 sequences					
Mean	21.08%	**1.42%**	15.00%	2.13%	**1.60%**
Median	21.08%	0.00%	15.00%	2.13%	**1.60%**

4.1. Clean Data

First, we perform experiments on the full original Hopkins 155 dataset. The dataset consists of 120 sequences with 2 motions and 35 sequences with 3 motions which can be divided into three categories: checkerboard, traffic, and articulated. For each sequence, feature trajectories are obtained using an automatic tracker, and errors in tracking are manually corrected. Therefore, the motion sequences in this dataset can be considered as clean data without any corruption or missing entries. We run our method on all sequences of Hopkins 155 dataset and compute the average and median misclassification rates for each category of sequences. Our results are listed in Tables 1 and 2. For the purpose of comparison, we also list in Tables 1 and 2 the corresponding results of state-of-the-art motion segmentation algorithms: ALC [13], SSC [7], Non-Negative Matrix Factorization (NNMF) [4] and Multi-Scale Motion Clustering (MSMC) [6].

As Tables 1 and 2 show, among all of the algorithms compared, the overall performance of ours is the best. To be specific, our algorithm gives the most accurate segmentation results for checkerboard sequences with two and three motions and traffic sequences with three motions. For other sequences in this dataset, our results are just slightly inferior to the best performing algorithm, $i.e.$ SSC. We also note that NNMF and MSMC perform comparably to ALC, SSC, and our method for traffic and articulated sequences but significantly worse for checkerboard sequences, which are the most important components of the Hopkins dataset. We further divide the checkerboard sequences into two parts: 51 sequences without or with only translational background motions ($e.g.$ '1RT2TC')[3]; 53 sequences with non-translational background motions ($e.g.$ '1R2RCR').We then compute the average and median misclassification rates of our method, NNMF, and MSMC for the whole and the two parts of checkerboard sequences, and list these values in Table 3. From Table 3, it can be observed that non-translational background motion leads to significant deterioration of the performance of translational model-based NNMF. In MSMC, the use of a higher order motion model (homography matrix) can only help alleviate this problem to some extent, but cannot solve it completely. In contrast, our method, although also based on the translational motion model, can achieve low misclassification rates for the sequences with non-translational background motion as well, showing the effectiveness of foreground-background separation in our clustering algorithm.

Table 2. Misclassification rates (%) for the sequences with two motions in Hopkins 155 dataset.

Method	ALC	SSC	NNMF	MSMC	Ours'
Checkerboard, 78 sequences					
Mean	1.55%	1.12%	11.60%	3.62%	**0.71%**
Median	0.29%	0.00%	3.08%	0.00%	**0.00%**
Traffic, 31 sequences					
Mean	1.59%	**0.02%**	0.10%	0.66%	**0.05%**
Median	1.17%	0.00%	0.00%	0.00%	**0.00%**
Articulated, 11 sequences					
Mean	10.70%	0.62%	10.00%	2.66%	**0.96%**
Median	0.95%	0.00%	10.00%	0.00%	**0.00%**

Table 3. Misclassification rates (%) for the checkerboard sequences in Hopkins 155 dataset.

Method	NNMF	MSMC	Ours'
All, 104 sequences			
Mean	13.54%	4.79%	**1.14%**
Median	7.17%	0.00%	**0.00%**
Sequences without or with only translational			
Mean	5.70%	1.86%	**0.63%**
Median	0.95%	0.00%	**0.00%**
Sequences with non-translational			
Mean	21.69%	7.84%	**1.68%**
Median	19.36%	0.00%	**0.00%**

[3]For sequences without background trajectories ($e.g.$ '1R2RCR_g12'), only step 4 of our two-stage clustering is performed on.

4.2. Missing and Corrupted Data

In this subsection, two experiments are conducted to test the robustness of our method to missing and corrupted data in motion sequences. The first experiment is designed to examine how the segmentation accuracy of our method changes with increasing percentage of missing or corrupted entries in a sequence. In this experiment, we compare the performance of our method with that of ALC and NNMF, which both can deal with incomplete and corrupted trajectories. We refer to ALC designing to handle incomplete and corrupted trajectories as ALC-miss and ALC-corrupted, respectively. We choose three representative sequences from Hopkins 155 dataset for testing: '1RT2TC' (checkerboard), 'arm' (articulation), and 'cars2_06' (traffic). For each sequence, we remove a portion (from 0% to 80%) of the entries by randomly fixing the start and duration of each feature point's successful tracking window, and run our method, ALC, and NNMF on the resulting sequences. Then, in Figure 5(Top), the misclassification rates of our method, ALC, and NNMF on each sequence are plotted as a function of percentage of missing entries. Next, for each sequence, we randomly select a portion (from 0% to 100%) of the entries, and corrupt each selected entry by adding Gaussian noise with zero mean and variance $\lambda\|W\|_F$ (λ is randomly valued from 0.01 to 0.1). We then run our method, ALC, and NNMF on the resulting sequences, and plot their misclassification rates as a function of percentage of corrupted entries in Figure 5(Bottom).

From Figure 5, it can be seen that for clean sequences without any corruption or missing entries, all of the three algorithms can give near-perfect segmentation results. However, when increasing amounts of missing or corrupted data are introduced to a given sequence, the performance of ALC and NNMF degrades much faster than ours. Especially in the case of more than half of trajectories being abandoned or corrupted, our method significantly outperforms other methods. The different performances of the three algorithms are mainly due to the different ways of dealing with incomplete and corrupted trajectories. To ALC, the requirement of enough complete and uncorrupted trajectories for repairing the pathological trajectories limits its ability to robustly segment the sequences with a large number of missing or corrupted entries. As for our method and NNMF, although both use matrix factorization to handle pathological trajectories, our method exploits temporal smoothness of feature trajectories to compactly represent the bases of trajectories with the DCT basis vectors, resulting in a significant reduction in unknowns and corresponding robustness of handling missing and corrupted data.

In the second experiment, we use Berkeley motion segmentation dataset to test the performance of our method on motion sequences with real incomplete and corrupted trajectories. This dataset contains 26 video sequences. For each sequence, feature trajectories are obtained using tracker in [15], and no manual effort is made to correct or remove incorrect tracks. Therefore, the sequences in this dataset contain considerable quantities of missing and corrupted entries. Another feature of each sequence in this dataset is that the background area either remains static or undergoes mainly translational motion. An evaluation tool including 5 metrics: density (%), overall error (E_{Over},%), average error (E_{Ave},%), over-segmentation error (O.S.) and number of extracted objects (E.O.) was also provided by authors of this dataset. We compare our method with Brox&Malik [3], Max-Project [12], Semi-Nonnegative Matrix Factorization (SNMF) [11] and ALC, and show results in Table 4. As the results show, our method achieves the best overall performance among all of the compared methods. More specifically, we obtain the lowest over-segmentation error and can extract the most objects in the dataset (62 in total). For overall error and average error, our method is slightly inferior to the best performing state-of-the-art algorithms, $i.e.$ Max-project, but significantly superior to others. The superior performance on Berkeley motion segmentation dataset again demonstrates the robustness of our method to missing and corrupted data.

Table 4. Evaluation on the Berkeley motion segmentation dataset.

	Density	$E_{Overall}$	$E_{Average}$	O.S.	E.O.
All available frames					
Our method	**3.31%**	**4.72%**	**23.82%**	**1.35**	**33**
Brox&Malik	3.31%	6.82%	27.34%	1.77	27
Max-project	3.22%	**4.48%**	**22.34%**	1.84	31
SNMF	3.30%	7.41%	**23.79%**	7.34	23
ALC-miss	3.29%	14.93%	43.14%	18.80	5

5. Conclusion

This paper proposes a new trajectory clustering algorithm for motion segmentation which is highly robust to the missing and corrupted data typical of real-world tracking process. Differing from previous works, we propose using the DCT basis as temporal smoothness constraint to facilitate segmentation of incomplete and corrupted trajectories. Due to the optimality of DCT basis for representing motion trajectories, our method can estimate reliable low-dimensional representation of trajectories even with a large amount of missing data and corrupted noises. We next present a two-stage clustering strategy, which first separates foreground from background based on motion subspaces constraints and partitions the foreground trajectories into different clusters using spectral clustering, to segment the low-dimensional representation, and then label the trajectories accordingly. The advantage of the proposed two-stage clustering is that we can use a simple translational model to effectively handle sequences containing complex motions. Experiments on both Hopkins and Berkeley datasets show

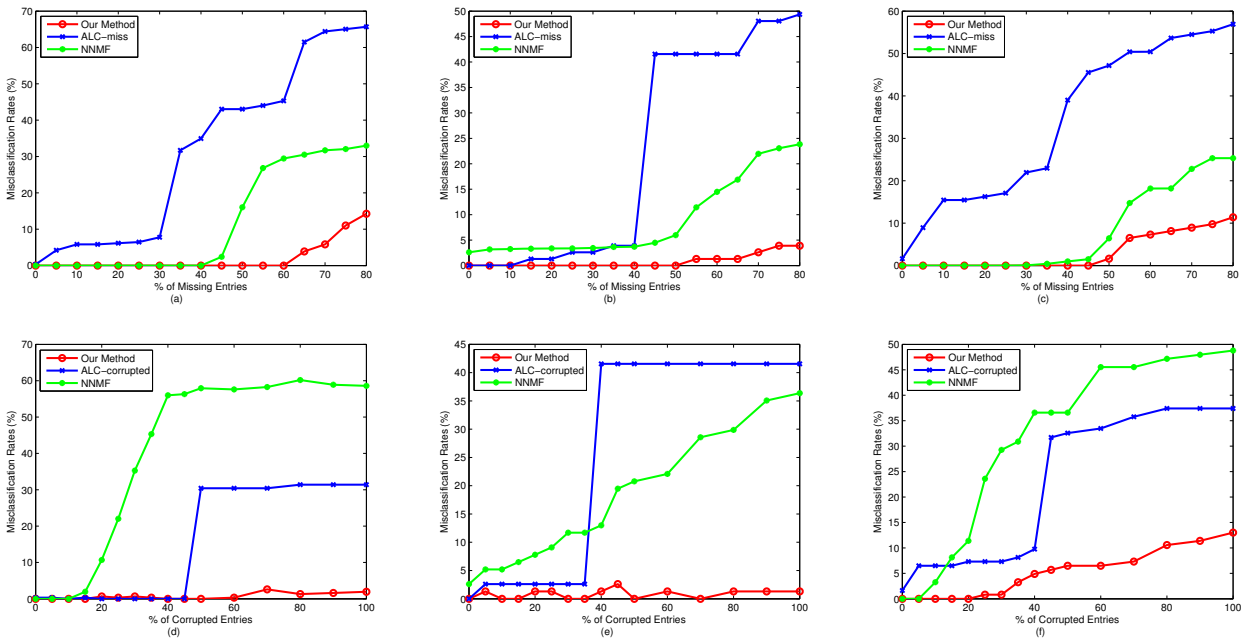

Figure 5. Top: The misclassification rates of our method, ALC, and NNMF on '1RT2TC' (a), 'arm' (b), and 'cars2_06' (c) when percentage of missing entries in the sequence increases from 0% to 80%. Bottom: The misclassification rates of our method, ALC, and NNMF on '1RT2TC' (d), 'arm' (e), and 'cars2_06' (f) when percentage of corrupted entries in the sequence increases from 0% to 100%.

the advantage of our method over other state-of-the-art trajectory clustering algorithms in terms of both effectiveness and robustness.

6. Acknowledgements

This work is supported by National 863 Program of China under Grant No.2012AA011803, Natural Science Foundation of China under Grant No.61170188 and No.61273276.

References

[1] I. Akhter, Y. Sheikh, S. Khan, and T. Kanade. Trajectory space: a dual representation for nonrigid structure from motion. *IEEE Trans. on PAMI*, 33(7):1442–1456, July 2011.

[2] I. Akhter, T. Simon, S. Khan, T. Simon, and Y. Sheikh. Bilinear spatiotemporal basis models. *ACM Trans. Graph*, 31(2), April 2012.

[3] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *Proc. ECCV*, volume 6315, pages 282–295, 2010.

[4] A. M. Cheriyadat and R. J. Radke. Non-negative matrix factorization of partial track data for motion segmentation. In *Proc. ICCV*, pages 865–872. IEEE, 2009.

[5] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 29(3):159–179, September 1998.

[6] R. Dragon, B. Rosenhahn, and J. Ostermann. Multi-scale clustering of frame-to-frame correspondences for motion segmentation. In *Proc. ECCV*, 2012.

[7] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *Proc. CVPR*, pages 2790–2797. IEEE, 2009.

[8] M. Fradet, P. Robert, and P. Perez. Clustering point trajectories with various life-spans. In *Proc. CVMP*, pages 7–14. IEEE, 2009.

[9] K. Fragkiadaki and J. Shi. Detection free tracking: exploiting motion and topology for segmenting and tracking under entanglement. In *Proc. CVPR*, pages 2073–2080. IEEE, 2011.

[10] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *Proc. ICML*, 2010.

[11] Q. Mo and B. A. Draper. Semi-nonnegative matrix factorization for motion segmentation with missing data. In *Proc. ECCV*, volume 7578, pages 402–415, 2012.

[12] P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *Proc. CVPR*, pages 614–621. IEEE, 2012.

[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 *Proc. CVPR*, pages 1–8. IEEE, 2008.

[14] J. Shi and C. Tomasi. Good features to track. In *Proc. CVPR*, pages 593–600. IEEE, 1994.

[15] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *Proc. ECCV*, volume 6311, pages 438–451, 2010.

[16] R. Tron and R. Vidal. A benchmark for the comparison of 3d motion segmentation algorithms. In *Proc. CVPR*, pages 1–8. IEEE, 2007.

[17] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, nonrigid, degenerate and non-degenerate. In *Proc. ECCV*, volume 3954, pages 94–106, 2006.