Online Motion Segmentation using Dynamic Label Propagation

Ali Elqursh Ahmed Elgammal
Rutgers University

Abstract

The vast majority of work on motion segmentation adopts the affine camera model due to its simplicity. Under the affine model, the motion segmentation problem becomes that of subspace separation. Due to this assumption, such methods are mainly offline and exhibit poor performance when the assumption is not satisfied. This is made evident in state-of-the-art methods that relax this assumption by using piecewise affine spaces and spectral clustering techniques to achieve better results. In this paper, we formulate the problem of motion segmentation as that of manifold separation. We then show how label propagation can be used in an online framework to achieve manifold separation. The performance of our framework is evaluated on a benchmark dataset and achieves competitive performance while being online.

1. Introduction

Since the early 20th century, gestalt psychologist have identified common fate as one of the most important cues for dynamic scene understanding. In the field of Computer Vision, this is reflected by the vast amount of literature on motion segmentation, video segmentation, and tracking. Specifically, motion segmentation deals with the problem of segmenting feature trajectories according to different motions in the scene, and is an essential step to achieve object segmentation and scene understanding.

Recent years have witnessed a large increase in the proportion of videos coming from streaming sources such as TV Broadcast, internet video streaming, and streaming from mobile devices. Unfortunately, most motion segmentation techniques are mainly offline and with a high computational complexity. Thus rendering them ineffective for processing videos from streaming sources. This highlights the need for novel online motion segmentation techniques.

There exist a plethora of applications that would benefit from online motion segmentation. For example, currently activity recognition is either restricted to videos captured from stationary cameras (where existing background subtraction techniques can be used to segment the different actors [18]), or restricted to process videos offline using motion segmentation techniques. This is complicated even further if, after processing, more data becomes available, with the only solution typically being to reprocess the entire video from the beginning.

Figure 1. Frames 40, and 150 from the marple7 sequence and the corresponding segmentation obtained by our online approach.

Another domain that would benefit from online motion segmentation is in that of 3D TV processing. A real-time motion segmentation would enable performing 2D-to-3D conversion and video re-targeting on the fly on viewers devices. Other applications include online detection and segmentation of moving targets, and visual surveillance from mobile platforms to name a few.

Many approaches for motion segmentation are based on the fact that trajectories generated from rigid motion and under affine projection spans a 4-dimensional subspace. First introduced by [16], this geometric constraint has been used extensively especially in the case of independent motion. Most notably in [4], the problem is reduced to sorting of a matrix called shape interaction matrix with entries that represent the likelihood of a pair of trajectories belonging to

the same object. In [11] the problem is reformulated as an instance of subspace separation, making the connection explicit.

Most motion segmentation approaches suffer from two problems. First, formulating the problem as that of factorizing a trajectory matrix has led many approaches to assume that trajectories span the entire frame sequence. To handle the case where parts of trajectories are missing, such approaches borrow ideas from matrix completion. However, this is only successful up to a limit, since it assumes that at least there exist some trajectories that span the entire frame sequences. Second, the affine camera assumption restricts the applicability of motion segmentation to those videos where the assumption is satisfied. For example, this is typically true when the depth of the scene is small but not otherwise. To overcome the later problem, we assume a general perspective camera instead of an affine camera. On the other hand, to overcome the former problem, we measure the similarity between trajectories using a metric that depends only on the overlapping frames.

We propose an approach that achieves online motion segmentation by segmenting a set of manifolds through dynamic label propagation and cluster splitting. Starting from an initialization computed over a fixed number of frames, we maintain a graph of pairwise similarity between trajectories in an online fashion. To move to the next frame we propagate the label information from one frame to the next using label propagation. The label propagation respects the computed graph structure while taking into account the previous labeling. To handle cases where new evidence suggests that one cluster comes from two differently moving objects, we evaluate each cluster and measure a normalized cut cost of splitting the cluster. This process is then repeated for each subsequent frame. Figure 1 shows frames 40 and 150 of the sequence *marple7* and the segmentation by our approach. Miss Marple is correctly tracked throughout the sequence even when affected by occlusion as in frame 40.

2. Contributions

Our paper has several contributions. First we show how trajectories belonging to a rigid object with smooth depth variation form a manifold of dimension 3. This generalizes the problem of affine motion segmentation from subspace separation (linear manifold segmentation) to that of (general) manifold segmentation. It also explains why previous approaches using spectral clustering methods produced superior results while using simpler models. Second, we show that the problem of online manifold segmentation can be cast in a label propagation framework using Markov Random walks.

3. Related Work

Approaches to motion segmentation (and similarly subspace separation) can be roughly divided into four categories: statistical, factorization-based, algebraic, and spectral clustering. Statistical methods alternate between assigning points to subspaces and reestimating the subspaces. For example, in [9] the Expectation-Maximization (EM) algorithm was used to tackle the clustering problem. Robust statistical methods, such as RANSAC [6], repeatedly fits an affine subspace to randomly sampled trajectories and measures the consensus with the remaining trajectories. The trajectories belonging to the subspace with the largest number of inliers are then removed and the procedure is repeated.

Factorization-based methods such as [16, 11, 10] attempt to directly factor a matrix of trajectories. These methods work well when the motions are independent. However, it is frequently the case that multiple rigid motions are dependent, such as in articulated motion. This has motivated the development of algorithms that handle dependent motion. Algebraic methods, such as GPCA [19] are generic subspace separation algorithms. They do not put assumptions on the relative orientation and dimensionality of motion subspaces. However, their complexity grows exponentially with the number of motions and the dimensionality of the ambient space.

Spectral clustering-based methods [21, 2, 12], use local information around the trajectories to compute a similarity matrix. It then use spectral clustering to cluster the trajectories into different subspaces. One such example is the approach by Yan et al [21], where neighbors around each trajectory are used to fit a subspace. An affinity matrix is then built by measuring the angles between subspaces. Spectral clustering is then used to cluster the trajectories. Similarly, sparse subspace clustering [5] builds an affinity matrix by representing each trajectory as a sparse combination of all other trajectories and then applies spectral clustering on the resulting affinity matrix. Spectral clustering methods represent the state-of-the-art in motion segmentation. We believe this can be explained because the trajectories do not exactly form a linear subspace. Instead, such trajectories fall on a nonlinear manifold.

With the realization of accurate trackers for dense long term trajectories such as [13, 15] there have been great interest in exploiting dense long term trajectories in motion segmentation. In particular, Brox et al. [2] achieves motion segmentation by creating an affinity matrix capturing similarity in translational motion across all pairs of trajectories. Spectral clustering is then used to over-segment the set of trajectories. A final grouping step then achieves motion segmentation. More recently, Fragkiadaki et al. [7] proposes a two step process that first uses trajectory saliency to segment foreground trajectories. This is followed by a two-stage spectral clustering of an affinity matrix computed

over figure trajectories. The success of such approaches can be attributed in part to the large number of trajectories available. Such trajectories help capture the manifold structure empirically in the spectral clustering framework. Our approach is also based on building an affinity matrix between all pairs of trajectories, however we process frames online and do not rely on spectral clustering. Deviating from the spectral clustering, is the idea of using nonlinear dimensionality reduction (NLDR) techniques followed by clustering to achieve motion segmentation [8].

Compared to previous methods, our method has several advantages. To our knowledge our approach is the first to achieve online motion segmentation, while spending a constant computation time per frame. We explicitly model trajectories as lying on a manifold, and thus are able to handle videos where the affine camera assumption is not satisfied. Our method also takes into account the entire history of the trajectory in computing the similarity matrix.

4. Basic Formulation of Motion Segmentation

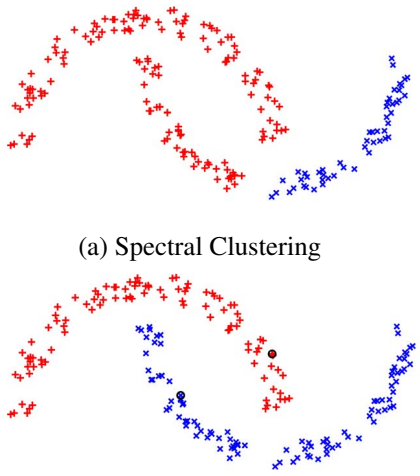
In this section we show how the problem of motion segmentation can be cast as a manifold segmentation problem. This is demonstrated in two steps. First, we show how trajectories in the three-dimensional space form a three-dimensional manifold. Next, we show how the projection of these trajectories to 2D image coordinates also form a three-dimensional manifold.

Let X be an open set of points in 3D comprising a single rigid object. Together with the Euclidean metric it forms a three-dimensional manifold. The motion of the object at times $t = 2, \ldots, F$ can be represented by rigid transformations $f_2(\mathbf{x}), \ldots f_F(\mathbf{x})$ respectively with $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is a 3D point in the camera coordinate system. The space of trajectories can be therefore defined by the set

$$\Gamma(f) = \{(\mathbf{x}_1, \ldots, \mathbf{x}_F) \in \mathbb{R}^{3F} : \mathbf{x}_i = f_i(\mathbf{x}_1)\ i \neq 1\},$$

with subspace topology. Let $\pi_1 : \mathbb{R}^{3F} \rightarrow \mathbb{R}^3$ denote the projection of a point $(\mathbf{x_1}, \ldots, \mathbf{x}_F) \in \mathbb{R}^{3F}$ onto the first factor. Let $\phi : \Gamma(f) \rightarrow X$ be the restriction of $\pi_1$to $\Gamma(f)$. Since $f_2, \ldots, f_F$ are continuous maps and ϕ is a restriction of a continuous map, ϕ is also continuous. It is also a homeomorphism because it has a continuous inverse. This implies that the space of trajectories is a manifold of dimension three.

Furthermore, we can show that projecting the 3D trajectories into the image coordinates also induces a manifold. Let $g(\mathbf{x}) = \frac{\mathbf{f}}{z}[x\, y]^T$ be the camera projection function that projects a point in the camera coordinate system to image coordinates, where $\mathbf{f}$ is the camera focal length. $g(\mathbf{x})$ is continuous at all points except at points with $z = 0$. Let $\Omega(f) = \Gamma(f) \backslash \{\mathbf{x}_1 \ldots \mathbf{x}_F : z_i \neq 0\}$ be the subset of $\Gamma(f)$ where all points satisfy this constraint, it follows that $G(\mathbf{x}_1, \ldots, \mathbf{x}_F) = (g(\mathbf{x}_1), \ldots, g(\mathbf{x}_F))$ is also a smooth continuous map over $\Omega(f)$. It is therefore easy to show that $G(\Omega)$ is also a manifold of dimension three.



(a) Spectral Clustering

(b) Label Propagation

Figure 2. Spectral Clustering vs Label propagation. Colors represent the different clusters and the black circle represent the supervised labels.

Note that even though we know that trajectories in image space form a manifold, we do not have an analytical manifold. However, under the assumption that the manifold is densely sampled, empirical methods can be used to model the manifold. In addition, note that each distinct motion in the scene will generate one manifold. In this paper we rely on label propagation and dense trajectory tracking to solve the manifold separation problem.

To see why label propagation is well suited for the manifold separation problem, consider the simple two moons example shown at the top Figure 2. Separating the two moons can be cast as a manifold separation problem. However, when applying spectral clustering on this example, due to the proximity, one cluster leaks over the other cluster. On the other hand, with proper initialization, label propagation is able to successfully segment the two moons. As explained in the next sections, the initialization comes from previous frames.

5. Approach

Starting from dense trajectories that are continuously extended and introduced, our approach continuously updates a segmentation of the trajectories corresponding to different motions. To achieve this, we start by explaining how the similarities (affinities) between trajectories can be updated in an online framework (Subsection 5.1). Next we introduce the necessary background on label propagation and show how it can be used to maintain a segmentation over dynamically changing manifolds (Subsection 5.2). Finally,

we propose two different methods to initialize our framework in (Subsection 5.3).

5.1. Online Affinity Computation

As identified by the previous section, trajectories belonging to a single object lie on a three-dimensional manifold. However, such manifolds are not static as they are a function of the motion of the object, which changes over time. To model such dynamic manifolds without resorting to resolving for each frame, we design a distance metric that can be computed incrementally. Such computation has to be done in time independent of the length of the trajectories. In addition, the metric must capture the similarity in spatial location and motion. The intuition is that if two trajectories are relatively close to each other and move similarly, then they are likely to belong to the same object. In this subsection we show how one such metric can be computed incrementally.

We start by introducing some notation. A trajectory $T_a = \{p_a^i = (x_a^i, y_a^i) : i \in A\}$ is represented as a sequence of points p_a^i that spans frames in the set A. For simplicity we reserve superscripts for frame references and subscripts for trajectory identification. The motion of a trajectory between frames i and j in the x and y direction is denoted by $u_a^{i:j} = x_a^j - x_a^i$ and $v_a^{i:j} = y_a^j - y_a^i$.

Given two trajectories T_a and T_b we define two distance metrics $d_M^{1:t}(T_a, T_b)$ and $d_S^{1:t}(T_a, T_b)$ representing the difference in motion and spatial location up to time t respectively. In a way similar to [2], the distances are defined as the supremum of distances over pairs of frames. The max function helps "remember" large differences in motion and spatial location. Formally,

$$d_M^{1:t}(T_a, T_b) = max_{\{i-\Delta, i\} \subset X} d_M^i(T_a, T_b), \quad (1)$$

$$d_S^{1:t}(T_a, T_b) = max_{i \in X} d_S^i(T_a, T_b), \quad (2)$$

where $X = \{x : x < t, x \in A \cap B\}$ is the set of overlapping frames up to time t and Δ is a user defined parameter, which controls the amount of smoothing used in computing motion difference. Distances over frames are defined as

$$d_M^i(T_a, T_b) = \frac{(u_a^{i-\Delta:i} - u_b^{i-\Delta:i})^2}{(\sigma_{Mu}^i)^2} + \frac{(v_a^{i-\Delta:i} - v_b^{i-\Delta:i})^2}{(\sigma_{Mv}^i)^2},$$

and

$$d_S^i(T_A, T_B) = \| p_a^i - p_b^i \| / \sigma_S^2.$$

$(\sigma_{Mu}^i)^2$, and $(\sigma_{Mu}^i)^2$ are two parameters that control the weighting of motion distances while σ_S^2 controls the weighting of the spatial distance. We compute $(\sigma_{Mu}^i)^2$, and $(\sigma_{Mu}^i)^2$ adaptively for each frame as the variance of $u_a^{i-\Delta:i}$ and $v_a^{i-\Delta:i}$ over all trajectories. For n trajectories, we can collect all pairwise frame distances into two $n \times n$ matrices $\Delta \mathbf{D}_M^t = [d_M^t(T_i, T_j)]$, and $\Delta \mathbf{D}_S^t = [d_S^t(T_i, T_j)]$. It follows that we can compute the total distance between trajectories up to time t, $\mathbf{D}^t$, incrementally from $\mathbf{D}^{t-1}$ and $\Delta \mathbf{D}^t$ by taking the maximum of the two. To convert distances to affinities $\mathbf{W}$ we use

$$\mathbf{W}^t = exp(-(\mathbf{D}_M^t + \mathbf{D}_S^t)). \quad (3)$$

5.2. Label Propagation

Preliminaries Here we briefly explain the machine learning machinery used in our online object segmentation approach. For further details and an overview about label propagation please refer to [3].

Given a graph G and a weight matrix W such that W_{ij} is the weight of the edge between nodes i and j, a simple idea for semi-supervised learning is to propagate labels on the graph. Formally, let Y_l denotes the labels for the labeled nodes. Furthermore, let $\hat{Y} = (\hat{Y}_l, \hat{Y}_u)$ denotes the estimated node labels, with $\hat{Y}_l$, and $\hat{Y}_u$ corresponding to the labeled and unlabeled nodes respectively. Y_l and $\hat{Y}$ are encoded using a one-hot encoding such that each row of Y is a vector with 1 at the location corresponding to the label of the node and zero otherwise.In order to estimate probabilities over labellings, [22] presents an algorithm that uses Markov random walks on the graph with transition probabilities from i to j defined by,

$$p_{ij} = \frac{W_{ij}}{\sum_k W_{ik}}.$$

In matrix form $P = D^{-1}W$, where $D_{ii} = \sum_j W_{ij}$. Given the partition of the nodes into labeled and unlabeled nodes, the matrix P can be written in matrix form as

$$P = \begin{bmatrix} P_{ll} & P_{lu} \\ P_{ul} & P_{uu} \end{bmatrix}.$$

The algorithm then works as follows: it assigns to a node x_i the probability of arriving to a positively labeled example, when performing a random walk starting from node i and until a label is found. This probability can be written as $P(y_{end} = 1|i) = \sum_{j=1}^n P(y_{end} = 1|j) p_{ij}$. In matrix form this can be written as

$$Y_u = (P_{ul}|P_{uu}) \begin{pmatrix} Y_l \\ Y_u \end{pmatrix}.$$

It can be shown that the fixed point solution is

$$Y_u = (I - P_{uu})^{-1} P_{ul} Y_l.$$

The label for a node i, z_i can be then obtained by

$$z_i = argmax_j \, y_{ij}.$$

Manifold Separation via Dynamic Label Propagation

Given the segmentation of trajectories at frame $t-1$ the goal is to obtain an updated labeling at frame t. To accomplish this we have two address several scenarios. First, new

trajectories may be introduced into the distance matrix, and second, motion information from existing trajectories may necessitate splitting or merging existing clusters. In addition, in the former case, new trajectories may belong to existing objects or they may belong to new objects. In this section, we describe how these two cases are handled in our framework.

First we address how new information can be integrated to update our cluster labels while assuming that no new objects has entered the scene. At each frame, we assume extended trajectories have a probability distribution over different segment labels. The inferered label is defined as the label with the maximum probability. One way to proceed is to use these as supervised labels and attempt to label new trajectories based on a classifier learned over the labeled samples. There are several problems with this solution. First, existing labels are not revised and thus errors cannot be recovered from. Second, classifier do not take into account the graph structure we have available in hand.

To solve these problems, we use labels for existing trajectories as *prior* knowledge and attempt to label both trajectories extended from previous frames and new trajectories while taking the graph structure into account. We attach to each node (trajectory) in the current frame a dongle node corresponding to the trajectory label in the previous frame. Let P_{uu} be the transition probabilities obtained from the affinity matrix W^t , the new transition matrix for the augmented graph is

$$P = \begin{bmatrix} P_{ll} & P_{lu} \\ P_{ul} & P'_{uu} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \eta I \\ I & (1-\eta)P_{uu} \end{bmatrix}.$$

Applying Markov random walks on this graph gives $Y_u = (I - (1-\eta)P_{uu})^{-1}Y_l$. Since the labeled nodes are actually the previous frame labels, this equation becomes

$$Y^t = (I - (1-\eta)P_{uu})^{-1}Y^{t-1}. \quad (4)$$

The parameter $0 \leq \eta \leq 1$ controls how strongly previous labels affect future labels. In the extreme case, when $\eta = 0$, the graph structure is the dominant term. Using label propagation in this manner effectively takes into account the uncertainty in previous labels. Applying the above iteration we get a new probability distribution for each node and the segment labels are obtained by

$$z_l = \text{argmax}_j \; y_{ij}$$

Although, it may seem like an overkill to re-apply label propagation for each incoming frame, there are two reasons why this is not the case. First, by using the previous labels as anchoring points we avoid leaking the labels across clusters such as in spectral clustering (Figure 2). Second, in practice equation (4) is solved iteratively [3] and by using the previous frame labels as an initial solution we are able to converge in a few iteration.

Once the new labels are obtained, we need to evaluate if a new object has entered the scene, or if an object that was previously not moving started moving. Note, that label propagation only propagates existing labels and does not introduce new ones. If a new object enters the scene, there must be an associated set of new trajectories. These new trajectories would inevitably receive some label by the label propagation, and introduce high intra-cluster variation within that associated cluster. Similarly, if an stationary object starts moving this will lower the affinities between the object trajectories and the rest of the trajectories in the same cluster. The task is therefore to go over the clusters one at a time and see if any of them requires splitting. This can be accomplished by computing an optimal binary cut using normalized cut and then evaluate the normalized cut cost. If the cost is above a threshold then we keep the cluster intact.

To perform the cut we use the method of [14]. First, we extract the sub-matrix W_c corresponding to the cluster to be evaluated. We then solve the generalized eigen-value problem $D_c - W_c = \lambda D_c y$. Next, we extract the eigen-vector corresponding to the second smallest eigen-value, and evaluate the normalized cut cost for different thresholds. The normalized cut cost is expressed as $\frac{y^T(D_c - W_c)y}{y^T D_c y}$, where y is the thresholded eigen-vector. The vector y that minimizes the normalized cut cost is then selected as the best cut. The normalized cut cost is a value between 0 and 1. In our approach, if the cut cost is bellow a threshold τ we split the cluster.

5.3. Initialization

Our framework assumes that for a frame at time t we know the labels of the trajectories at time $t-1$. There are two ways to boot strap the system. First, we could use an offline motion segmentation method, which does not depend on the affine camera assumption to generate the initial labels. Another approach is to assign all the trajectories a single label initially and allow our cluster splitting process to discover the number classes over time. In our experiments we take the later approach and show that even without initialization, our approach is able to detect the moving objects in the scene. In Figure3 we show how starting from an initial assignment of a single cluster, the cluster splitting process is able to segment the two persons in the scene.

6. Experiments

In this section we evaluate our algorithm on the Berkley motion segmentation dataset introduced in [2]. The dataset consists of 26 sequences that include rigid and articulated motion. The ground truth for the dataset is provided as frame annotations for 189 frames and the dataset comes

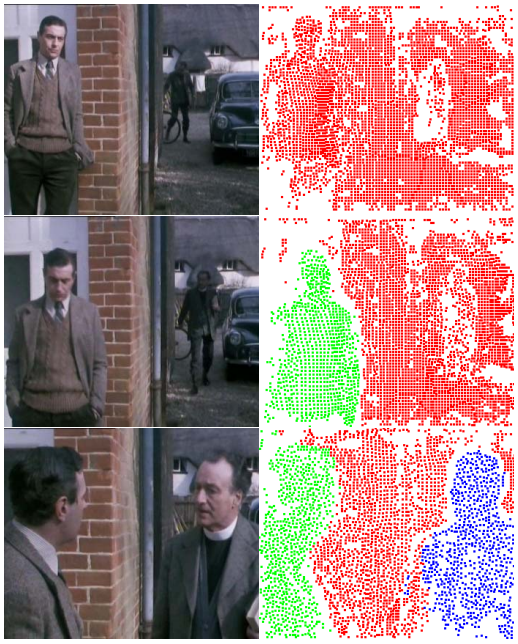


Figure 3. Initialization on the marple6 sequence. From top to bottom frames, 100, 160, 260 and their corresponding segmentations. At frame 1, all trajectories are given the same label and there is hardly any motion in the scene. After frame 100, the man leaning on the wall starts to move and is automatically segmented from the background cluster. Similarly, after frame 160 the person coming closer to the camera is also detected and popped out from the background.

with an evaluation tool. However, it is important to note that the evaluation tool was designed for offline algorithms. For instance, it assumes that each trajectory is assigned a single label throughout the sequence and will thus penalize a trajectory which is assigned an incorrect label at the beginning of a video sequence even if the label is corrected at a latter frame. Similarly, if an object is stationary and then moves, the approach is penalized for not segmenting the object while it is stationary. This puts our algorithm at a disadvantage, since it is impossible to detect motion before it occurs. In real applications this can be easily mitigated via a look ahead process, where the decision is delayed by letting the algorithm run several frames ahead. For the sake of consistency we report error measures using the same evaluation tool. More results are available in the supplemental materials.

Trajectory tracking is done using LDOF [15] but the approach is not limited to it. As demonstrated by the Middlebury benchmark [1], there now exists several real-time implementations of accurate optical flow that runs on the GPU. For example, [20].

The evaluation tool of [2] yields 5 measures for each sequence, which are then averaged across all sequences. The 5 measures are density, overall error, average error, over-segmentation error and the number of segments with less than 10% error which we abbreviate as *lt10*. The density measures the percentage of labeled trajectories to the total number of pixels. A higher number indicates better coverage of the image. Algorithms that require full trajectories over a sliding window reduces the density. The overall error is the total number of correctly labeled trajectories over the number of labeled trajectories. The tool automatically computes an assignment of clusters to ground truth regions and may assign several clusters to the same region. The average clustering error is the average of the ratio of mislabeled trajectories to the number of trajectories for each region. Since the tool may assign multiple segments to the same ground truth region, the tool also reports an over-segmentation error defined as the number of segments merged to fit the ground truth regions. Additionally the tool reports the number of regions covered with less than 10% error with one region subtracted per sequence to account for the background.

In our experiments we set the parameters to the following values; $\Delta = 3$, $\sigma_S = 300$, $\eta = 0.1$, and $\tau = 1 \times 10^{-3}$. The value for σ_S was set high enough to capture similarity between different parts of the background when a foreground object splits it into two disjoint regions. The remaining parameters were set empirically.

We compare our algorithm to the offline algorithms of [2], RANSAC, GPCA [19], and LSA [21]. The code for RANSAC, GPCA, and LSA was obtained from the Hopkins dataset [17]. It is important to note that [2] is used as a representative of offline spectral clustering algorithms. For the case where we run over more than 10 frames we compare with a baseline online algorithm that uses RANSAC over a sliding window. As noted in [2], other motion segmentation algorithms do not scale efficiently with number of trajectories. For example, over only 10 frames from the people1 sequence, GPCA takes 2963 seconds, and LSA [21] 38614 seconds. It is therefore infeasible to run these algorithms on a sliding window.

The RANSAC baseline is always given the total number of ground truth regions in the sequence. Even though increasing the window size may have improved the results, this would have been achieved by reducing the density drastically as it becomes harder to find trajectories that span the entire window. Figure 4 shows the effect of increasing the window size on the density and segmentation.

Table 1 presents the quantitative results of running our approach on the dataset introduced in [2]. We perform three sets of experiments. In the first, we compare our approach to [2], RANSAC, GPCA, and LSA over the first 10 frames while excluding the first frame. This experiment is designed to quantify the performance of the algorithm with respect to traditional motion segmentation algorithms that require the set of trajectories to span the entire sequence. In the second, we evaluate the approach over the first 200 frames.

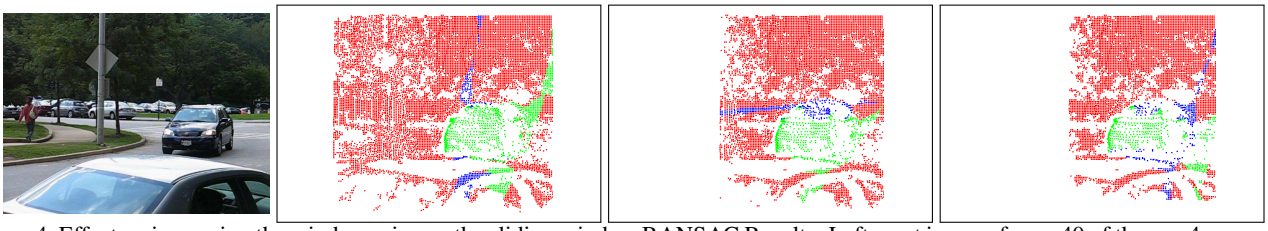


Figure 4. Effect on increasing the windows size on the sliding window RANSAC Results. Left most image: frame 40 of the cars4 sequence. Right three images: the segmentation with sliding window values of 10, 20, and 30 respectively. As the sliding window size increase, less trajectories span the entire window. (Best seen in color).

	Density	Overall Error	Average Error	Overseg	lt10
First 10 frames (26 sequences)					
Ours	3.43%	9.69%	29.93%	0.31	21
[2]	3.43%	7.49%	25.92%	0.46	20
RANSAC	3.37%	14.4%	29.87%	0.73	13
GPCA	3.37%	17.86%	28.64%	0.85	7
LSA	3.37%	19.69%	39.76%	0.92	6
Frames 50 - 200 frames (7 sequences)					
Ours	3.26%	6.77%	33.44%	2.57	6
[2]	3.43%	8.32%	37.29%	3.14	6
RANSAC	2.43%	28.3%	45.46%	1.42	0
All frames (26 sequences)					
Ours	3.22%	9%	32.89%	2.30	16
[2]	3.31%	6.82%	27.34%	1.77	27
RANSAC	2.28%	16.04%	42.6%	1.15	9

Table 1. Evaluation results on the Berkley Dataset

To avoid bias in the result due to initialization, we evaluate on ground truth frames starting on or after the 50th frame. This experiment is designed to quantify the performance of the algorithm on long sequences. Such sequences, represent the typical use case of an online algorithm. Finally, in the third set of experiments we evaluate over the entire set of sequences and ground truth annotation images.

Over 10 frames we out-perform GPCA, RANSAC, and LSA while achieving comparable results to results [2]. In fact, if we restrict ourselves to longer sequences we outperform [2] as can be seen in the second experiment. This indicates that our online approach outperforms traditional approaches while maintaining competitive accuracy. Comparing with RANSAC over longer sequences exposes the main problem with any online approach that is based on a sliding window. Information outside the sliding window is not remembered and it is therefore common to merge objects that were known to move differently.

Finally, over the entire dataset, we achieve online performance at the cost of slightly worser performance than [2]. These errors can possibly be further reduced if we employ a look-ahead process where the decision for a trajectory is delayed for several frames.

Algorithm	Tracks	Time (seconds)
Our method	4625	29
Brox et al. [2]	4699	19
GPCA	4625	1345
LSA	1012	1996

Table 2. Computation Time over 10 frames from marple1 sequence

track	dist	aff	lblprop	Total
34	1521	1169	403	3027

Table 3. Computation time over different stages on a single frame from *marple1*. The stages are: tracking (**track**) **,** distance matrix update (**dist**), affinity matrix (**aff**) and label propagation (**lblprop**). Times are in msec. The number of trajectories is 4427. Optical flow computation used in tracking is not included.

Figure 5 shows the result of applying our method to the *marple2* sequence. At frame 50, the method had already recovered from the bad initialization and is segmenting Miss Marple correctly. Finally as Miss Marple reappears from behind the column, our method re-detects the segment.

Table 2. compares the running time of different algorithms over the first 10 frames of the *marple1* sequence. Although the method of [2] uses 19 seconds for the first 10 frames, applying it on a sliding window would require 19 sec. per frame. On the other hand, our un-optimized Matlab implementation takes around 3 seconds per frame. Table 6 further shows that the computational time is dominated by updating the $n \times n$ distance matrix and computing the affinity matrix. We believe that real-time performance can be easily achieved since such operations can be easily parallelized on the GPU.

7. Conclusion

We showed how motion segmentation can be cast as a manifold separation problem. Based on this, we presented an approach that achieves online motion segmentation without sacrificing the accuracy of state of the art methods. By using label propagation on a dynamically changing graph, our approach is able to maintain labels and recover from errors as more information becomes available. Compared to offline approaches, we showed competitive results on a

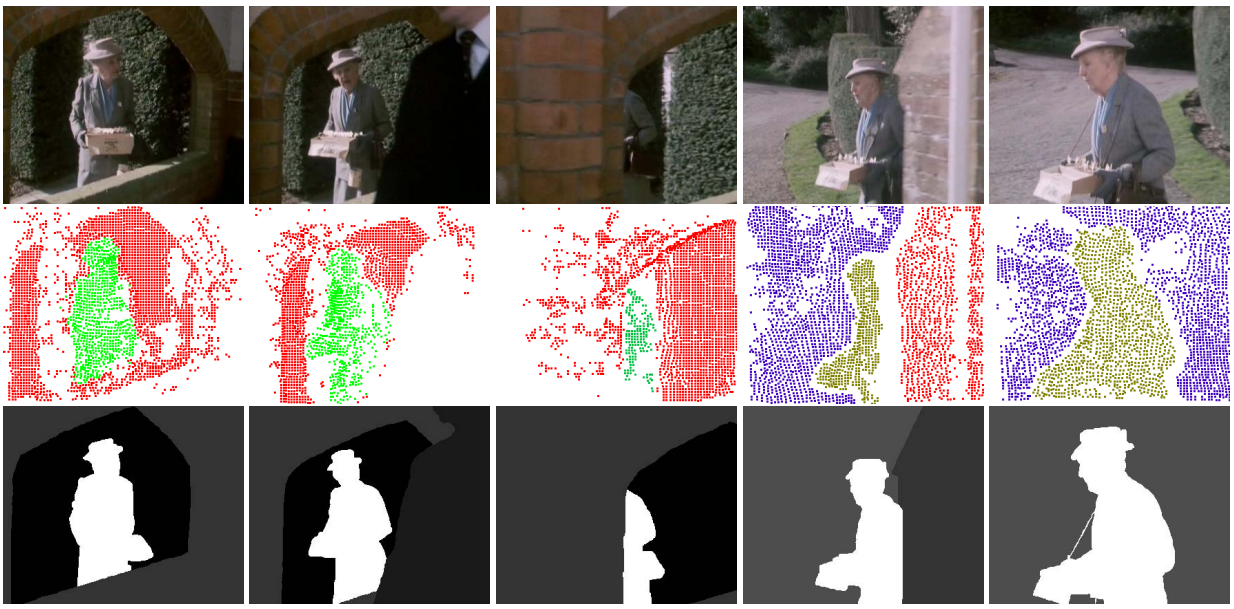


Figure 5. Result of our approach on the *marple2* sequence. First row: frames 50, 110, 135, 170, 200 of the sequence. Second Row: segmentation results. The third row shows ground truth frames associated with the frames. Starting from a incorrect initialization, our approach is able to automatically detect the person in the scene. Clusters maintain their labels under partial occlusion as can be seen with the background segment between frames 110 , 135. However, when a segment is totally occluded its label is lost and is assigned a new label once it is dis-occluded. (Best seen in color.)

benchmark dataset.

We are motivated in our approach by several applications that require online processing. For example, real-time motion segmentation can be used to perform video re-targeting on-the-fly on viewers devices. Even when the videos are available offline, processing movie-long videos would take in the order of weeks using existing offline approaches.

Acknowledgements This work was partly supported by the National Science Foundation award number 0923658.

References

[1] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A Database and Evaluation Methodology for Optical Flow. *IJCV*, 92(1):1–31, Nov. 2010. 6

[2] T. Brox and J. Malik. Object Segmentation by Long Term Analysis of Point Trajectories. In *ECCV*, pages 282–295, 2010. 2, 4, 5, 6, 7

[3] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. 4, 5

[4] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *ICCV*, pages 1071–1076. IEEE Comput. Soc. Press, 1995. 1

[5] E. Elhamifar and R. Vidal. Sparse subspace clustering. *CVPR*, pages 2790–2797, June 2009. 2

[6] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. 2

[7] K. Fragkiadaki and J. Shi. Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement. *CVPR*, pages 2073–2080, June 2011. 2

[8] A. Goh and R. Vidal. Segmenting Motions of Different Types by Unsupervised Manifold Clustering. *CVPR*, pages 1–6, June 2007. 3

[9] A. Gruber and Y. Weiss. Multibody factorization with uncertainty and missing data using the EM algorithm. *CVPR*, 1:707–714, 2004. 2

[10] N. Ichimura. Motion Segmentation Based on Factorization Method and Discriminant Criterion. In *ICCV*, volume 00, 1999. 2

[11] K. Kanatani. Motion segmentation by subspace separation and model selection. In *ICCV*, volume 2, pages 586–591. IEEE, 2001. 2

[12] P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *CVPR*, pages 614–621. Ieee, June 2012. 2

[13] P. Sand and S. Teller. Particle Video: Long-Range Motion Estimation Using Point Trajectories. *IJCV*, 80(1):72–91, May 2008. 2

[14] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *PAMI*, 22(8):888–905, 2000. 5

[15] N. Sundaram, T. Brox, and K. Keutzer. Dense Point Trajectories by GPU-Accelerated Large Displacement Optical Flow. In *ECCV*, pages 438–451, 2010. 2, 6

[16] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *IJCV*, 9(2):137–154, Nov. 1992. 1, 2

[17] R. Tron and R. Vidal. A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms. In *CVPR*, 2007. 6

[18] P. Turaga, R. Chellappa, and V. S. Subrahmanian. Machine recognition of human activities: A survey. *Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008. 1

[19] R. Vidal and R. Hartley. Motion segmentation with missing data using powerfactorization and GPCA. In *CVPR*, volume 2, pages 310–316, 2004. 2, 6

[20] A. Wedel, T. Pock, and C. Zach. An improved algorithm for TV-L 1 optical flow. *Statistical and Geometrical ...*, 1(x):23–45, 2009. 6

[21] J. Yan and M. Pollefeys. A General Framework for Motion Segmentation : Independent , Articulated , Rigid , Non-rigid , Degenerate and Non-degenerate. In *ECCV*, 2006. 2, 6

[22] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *ICML*, 2003. 4