Motion Segmentation by Model-Based Clustering of Incomplete Trajectories

Vasileios Karavasilis, Konstantinos Blekas, and Christophoros Nikou

Department of Computer Science, University of Ioannina,
PO Box1186, 45110 Ioannina, Greece
{vkaravas,kblekas,cnikou}@cs.uoi.gr

Abstract. In this paper, we present a framework for visual object tracking based on clustering trajectories of image key points extracted from a video. The main contribution of our method is that the trajectories are automatically extracted from the video sequence and they are provided directly to a model-based clustering approach. In most other methodologies, the latter constitutes a difficult part since the resulting feature trajectories have a short duration, as the key points disappear and reappear due to occlusion, illumination, viewpoint changes and noise. We present here a sparse, translation invariant regression mixture model for clustering trajectories of variable length. The overall scheme is converted into a Maximum A Posteriori approach, where the Expectation-Maximization (EM) algorithm is used for estimating the model parameters. The proposed method detects the different objects in the input image sequence by assigning each trajectory to a cluster, and simultaneously provides the motion of all objects. Numerical results demonstrate the ability of the proposed method to offer more accurate and robust solution in comparison with the mean shift tracker, especially in cases of occlusions.

Keywords: Motion segmentation, visual feature tracking, trajectory clustering, sparse regression.

1 Introduction

Visual target tracking is a preponderant research area in computer vision with many applications such as surveillance, targeting, action recognition from motion, motion-based video compression, teleconferencing, video indexing and traffic monitoring. Tracking is the procedure of generating inference about apparent motion given a sequence of images, where it is generally assumed that the appearance model of the target (e.g. color, shape, salient feature descriptors etc.) is known *a priori*. Hence, based on a set of measurements from image frames the target's position should be estimated.

Tracking algorithms may be classified into two main categories[1]: The first category is based on filtering and data association. It assumes that the moving object has an internal state, where the position of an object is estimated by combining the measurements with the model of the state evolution. Kalman filter [2] belongs to such methods which successfully tracks objects if the assumed type of motion is correctly modeled including cases of occlusion. Alternatively particle filters [3], including the condensation algorithm [4], are more general tracking methods without assuming specific type of densities. Finally there are methods relying on feature extraction and tracking using optical flow techniques [5]. A general drawback of this family of tracking algorithms is that the type of object's motion should be known and modeled correctly.

On the other hand, there are algorithms which are based on target representation and localization assuming for a probabilistic model for the object's appearance and the aim is to estimate it. More specifically, color or texture features of the object masked by an isotropic kernel are used to create a histogram. Then, the object's position is estimated by minimizing a cost function between the model's histogram and candidate histograms in the next image. A typical method in this category is the mean shift algorithm [1] and its extensions [6,7], where the object is supposed to be surrounded by an ellipse and the histogram is constructed from its internal pixel values. Also, algorithms based on the minimization of the differential Earth mover's distance [8,9] belong to this category.

Motion segmentation constitutes a significant application of tracking algorithms, which aims at identifying moving objects in a video sequence. It can be seen either as the post-processing step of a tracking algorithm, or as an assistive mechanism of the tracking algorithms by incorporating knowledge on the number of individual motions or their parameters. It has been considered in the framework of optical flow estimation, like in [10], where violations of brightness constancy and spatial smoothness assumptions caused by multiple motions are addressed and in [11], where affine flow is obtained by clustering the features into segments using the EM algorithm. Also, sparse features are clustered into groups and the number of groups is updated automatically over time in [12].

Trajectory clustering is also proposed in [13] where 3D trajectories are grouped using an agglomerative clustering algorithm and occlusions are handled by multiple tracking hypotheses. Finite mixtures of hidden Markov models (HMMs) were also employed [14] with parameter estimation obtained through the EM algorithm. Zappela *et al.* [15] project the space of the trajectories into a subspace of smaller dimensions and the clustering is performed by analyzing the eigenvalues of an affinity matrix, while in [16], overlapping trajectories are clustered and the resulting clusters are merged to cover large time spans.

Furthermore, spectral clustering approaches were also proposed, such as the method in [17], where the motions of the tracked feature points are modeled by linear subspaces and the approach in [18] where missing data from the trajectories are filled in by a matrix factorization method. Moreover, Yan *et al.* [19] estimate a linear manifold for every trajectory and then spectral clustering is employed to separate these subspaces. In [20], motion segmentation is accomplished by computing the shape interaction matrices for different subspace dimensions and combine them to form an affinity matrix that is used for spectral clustering.

Finally, many methods proposed independently rely on the separation of the image into layers. For example, in [21] tracking is performed in two stages: at first foreground extracted blobs are tracked using graph cut optimization and then pedestrians are associated with blobs and their motion is estimated by a Kalman filter.

In this work, we present a framework for visual target tracking based on clustering trajectories of key points. The proposed method creates trajectories of image features that correspond to Harris corner features [22]. However, key point tracking introduces an additional difficulty because the resulting feature trajectories have a short duration, as the key points disappear and reappear due to occlusion, illumination and viewpoint changes. Therefore, we are dealing with time-series of variable length. Motion segmentation is converted next into a clustering of these input trajectories, in a sense of grouping together feature trajectories that belong to the same object. For this purpose, we use an efficient regression mixture model, which has three significant features: a) Sparseness, b) it is allowed to be translated in measurement space and c) its noise covariance matrix is diagonal and not spherical as in most cases. The above properties are incorporated through a Bayesian regression modeling framework, where the Expectation-Maximization (EM) algorithm can be applied for estimating the model parameters. Special care is given for initializing EM where an interpolating scheme is proposed based on executing successively the k-means algorithm over the duration of trajectories. Experiments show the robustness of our method to occlusions and highlight its ability to discover better the objects motion in comparison with the state-of-the art mean shift algorithm [1].

The rest of the paper is organized as follows: the procedure of feature extraction and tracking in order to create the trajectories is presented in section 2. The trajectories clustering algorithm is presented in section 3, experimental results are shown in section 4 and a conclusion is drawn in section 5.

2 Extracting Trajectories

Trajectories are constructed by tracking points in each frame of the video sequence. The main idea is to extract some salient points from a given image and associate them with points from previous images. To this end, we employ the so called *Harris corners* [22] and standard optical flow for the data association step [23]. Let us notice that any other scale or affine invariant features [24,25] would also be appropriate. In this work we use Harris corner features due to their simplicity, as they rely on the eigenvalues of the matrix of the second order derivatives of the image intensities.

Let T be the number of image frames and $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1,\dots,N}$ be a list of trajectories with N being unknown beforehand. Each individual trajectory $\boldsymbol{y}_i$ consists of a set of 2D points, the time of appearance of its corner point into the trajectory, (i.e. the number of the image frame) and the optical flow vector of the last point in the list.

Initially, the list $\boldsymbol{Y}$ is empty. In every image frame, Harris corners are detected and the optical flow vector at each corner is estimated [5]. Then, each corner found in the current image frame is attributed to a trajectory that already exists

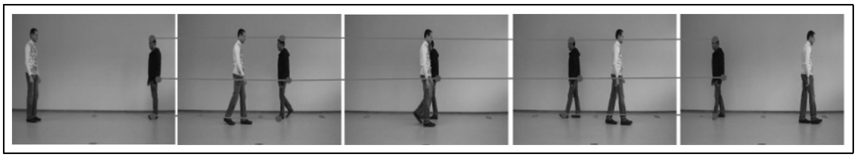


Fig. 1. Example of trajectories construction. The red dots represent the image key points and the green lines represent their trajectories. The figure is better seen in color.

or a new trajectory is created having with only one element, the corner under consideration. According to this scheme, three cases are possible:

- If any key point of the previous frame has an optical flow vector pointing out the key point under consideration, then the current corner is attributed to an existing trajectory. In this case, a trajectory follows the optical flow displacement vector, meaning that the corner is apparent in consecutive frames.
- If there is no such key point in the previous frame, we apply a window around the last corner which is similar to the current corner. If there are more than one similar corners then we choose the closet one.
- Otherwise, a new trajectory is constructed containing only the corner under consideration.

In Fig. 1, an intuitive example is presented where three corners are considered for demonstration purposes and five time instances are depicted. In the first frame, three corners are detected and three trajectories are created. In the second frame, the same corners are detected and associated with existing trajectories due to optical flow constraint. Next, one corner is detected and attributed to an existing trajectory due to optical flow constraints while the other two key points are occluded. During the fourth frame, the key point that was not occluded is also detected and attributed to an existing trajectory. One of the other two corner points that reappear is attributed to a trajectory due to local window matching. The other corner is not associated with any existing trajectory, so a new trajectory is created. In the last frame three corners are detected and associated with existing trajectories due to optical flow. Thus, four trajectories have been created, two of the same key point and another two of distinct key points.

Once the list $\boldsymbol{Y}$ is constructed trajectories of corner points belonging to the background are eliminated. This is achieved by removing the trajectories having small variance along the whole video sequence, according to a predefined-threshold value, as well as trajectories of small length (e.g. 1% of the number of frames). The complete procedure is described in the next algorithm 1.

3 Clustering Trajectories of Variable Length

Suppose we have a set of trajectories of N tracked feature points over T frames obtained from the previous procedure. The aim is to detect K independently

Algorithm 1. Trajectories construction algorithm

```
 1: function CreateTrajectories($\boldsymbol{Im}$)
 2:     Input: an image sequence $\boldsymbol{Im}$.
 3:     Output: a list of trajectories $\boldsymbol{Y}$.
 4:     $\boldsymbol{Y} = \emptyset$.
 5:     for every image $\{im^{(t)}\}_{t=1,\dots,T}$ do
 6:         Detect corners $\{c_l^{(t)}\}_{l=1,\dots,L^{(t)}}$ and estimate optical flow $\{f_l^{(t)}\}_{l=1,\dots,L^{(t)}}$ in
    each one.
 7:         for every corner $\{c_l^{(t)}\}_{l=1,\dots,L^{(t)}}$ detected in $im^{(t)}$ do
 8:             if $\boldsymbol{y}_i$ has its last corner $c_i^{last}$ in the image $t-1$ and it's optical flow $f_i^{last}$
    points to the current corner, i.e. $c_i^{last} + f_i^{last} \approx c_l^{(t)}$ then
 9:                 Insert $c_l^{(t)}$ into $\boldsymbol{y}_i$.
10:             else if $\boldsymbol{y}_i$ has its the window around it's last corner $c_i^{last}$ similar to the
    window around thw current corner $c_l^{(t)}$ then
11:                 Insert $c_l^{(t)}$ into $\boldsymbol{y}_i$.
12:             else
13:                 Insert a new trajectory $\boldsymbol{y}_i$ with only $c_l^{(t)}$ into $\boldsymbol{Y}$.
14:             end if
15:         end for
16:     end for
17:     Eliminate trajectory $\boldsymbol{y}_i$ with small variation in it's corners coordinates.
18: end function
```

moving objects in the scene by estimating labels on these points and classifying them into groups of different motions. Also, we want to estimate the characteristic motion of all objects.

A 2D trajectory $\boldsymbol{y}_i = (\boldsymbol{y}_i^{(1)}, \boldsymbol{y}_i^{(2)})$ consists of two directions: (1) horizontal and (2) vertical and is defined by a set of T_i points $\{(y_{i1}^{(1)}, y_{i1}^{(2)}), \dots, (y_{iT_i}^{(1)}, y_{iT_i}^{(2)})\}$, corresponding to the successive image positions $(t_{i1}, \dots, t_{iT_i})$ in the image sequence. That is important to note is that we deal with trajectories of variable length T_i, since occlusions or illumination changes may block the view of the objects in certain image frames.

Linear regression model constitutes a powerful platform for modeling sequential data that can be adopted in our case. In particular, we assume that a trajectory $\boldsymbol{y}_i^{(j)}$ of any direction $j = \{1, 2\}$ can be modeled through the following functional form:

$$\boldsymbol{y}_i^{(j)} = \Phi_i \boldsymbol{w}^{(j)} + d_i^{(j)} + \boldsymbol{e}_i^{(j)} \ , \tag{1}$$

where Φ_i is the design kernel matrix (common for both directions) of size $T_i \times T$, and $\boldsymbol{w} = (w_1, \dots, w_T)$ is the vector of the T unknown regression coefficients that must be estimated. In our case we have considered a design matrix Φ of size $T \times T$ having constructed with wavelets kernels. Thus, the matrix Φ_i is a block-matrix of Φ, which has only the T_i lines that corresponds to the sucessive T_i frames $(t_{i1}, \dots, t_{iT_i})$ of the i-th trajectory.

Also in the above equation, we assume an translation scalar term $d_i^{(j)}$ that allows for the entire trajectory to be translated as a unit [26]. Incorporating of such term results in a regression model that allows for arbitrary translations in measurement space. Finally, the error term $\boldsymbol{e}_i^{(j)}$ in the above formulation is a T_i-dimensional vector that is assumed to be zero-mean Gaussian and independent over time, i.e. $e_i \sim \mathcal{N}(0, \Sigma_i)$ with a diagonal covariance matrix $\Sigma_i^{(j)} = diag(\sigma_{t_{i1}}^{2^{(j)}}, \ldots, \sigma_{t_{iT_i}}^{2^{(j)}})^1$.

Under these assumptions, the conditional density of trajectory is Gaussian, i.e. $p(\boldsymbol{y}_i^{(j)}|\boldsymbol{w}^{(j)}, \Sigma_i^{(j)}, d_i^{(j)}) = \mathcal{N}(\Phi_i \boldsymbol{w}^{(j)} + d_i^{(j)}, \Sigma_i^{(j)})$. In our case we consider the scalar $d_i^{(j)}$ to be a zero-mean Gaussian variable with a variance $u^{(j)}$, i.e. $p(d_i^{(j)}) = \mathcal{N}(0, u^{(j)})$. We can further integrate out d_i to obtain the marginal density for $\boldsymbol{y}_i^{(j)}$ which is also Gaussian,

$$p(\boldsymbol{y}_i^{(j)}|\theta) = \int p(\boldsymbol{y}_i^{(j)}|\boldsymbol{w}^{(j)}, \Sigma_i^{(j)}, d_i^{(j)}) p(d_i^{(j)}) dd_i^{(j)} = \mathcal{N}(\Phi_i \boldsymbol{w}^{(j)}, \Sigma_i^{(j)} + u^{(j)}\mathbb{1}) \ , \tag{2}$$

where $\mathbb{1}$ is a matrix of 1's of size $T_i \times T_i$. The marginal density is implicitly conditioned on the parameters $\boldsymbol{\theta} = \{\boldsymbol{w}, u, \Sigma\}$.

In our study we consider a different functional regression model for every object k, as described by the set of model parameters $\boldsymbol{\theta}_k = \{\boldsymbol{\theta}_k^{(1)}, \boldsymbol{\theta}_k^{(2)}\}$, where $\boldsymbol{\theta}_k^{(j)} = \{\boldsymbol{w}_k^{(j)}, u_k^{(j)}, \Sigma_k^{(j)}\}$. Therefore, the task of discovering K objects becomes equivalent to clustering the set of N trajectories into K clusters. This can be described by the following regression mixture models:

$$p(\boldsymbol{y}_i|\boldsymbol{\Theta}) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{y}_i|\boldsymbol{\theta}_k) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{y}_i^{(1)}|\boldsymbol{\theta}_k^{(1)}) p(\boldsymbol{y}_i^{(2)}|\boldsymbol{\theta}_k^{(2)}) \ , \tag{3}$$

where we have assumed independence between that trajectories of both directions $(\boldsymbol{y}_i^{(1)}, \boldsymbol{y}_i^{(2)})$. In addition, π_k are the mixing weights satisfying the constraints $\pi_k \geq 0$ and $\sum_{k=1}^{K} \pi_k = 1$, while $\boldsymbol{\Theta}$ is the set of all the unknown mixture parameters, $\boldsymbol{\Theta} = \{\pi_k, \boldsymbol{\theta}_k\}_{k=1}^{K}$.

An important issue, when dealing with regression models is how to determine their order. Models of small order can lead to under-fitting, while larger order lead to curve over-fitting. Both cases may cause to serious deterioration of the clustering performance. Sparse modeling [27] offers a significant solution to this problem by employing models having initially many degrees of freedom than could uniquely be adapted given data. Sparse Bayesian regression can be achieved through a hierarchical prior definition over regression coefficients $\boldsymbol{w}_k^{(j)} = (w_{k1}^{(j)}, \ldots, w_{kT}^{(j)})^T$. In particular, we assume first that coefficients follows a zero-mean Gaussian distribution:

$$p(\boldsymbol{w}_k^{(j)}|\boldsymbol{\alpha}_k^{(j)}) = \mathcal{N}(\boldsymbol{w}_k^{(j)}|\boldsymbol{0}, \boldsymbol{A}_k^{-1^{(j)}}) = \prod_{l=1}^{T} \mathcal{N}(w_{kl}^{(j)}|0, \alpha_{kl}^{-1^{(j)}}) \tag{4}$$

1 Again Σ_i is a blosck matrix of a $T \times T$ diagonal covariance matrix that corresponds to the noise variance of T frames.

where $\boldsymbol{A}_k^{(j)}$ is a diagonal matrix containing the T elements of precisions $\boldsymbol{\alpha}_k^{(j)} = (\alpha_{k1}^{(j)}, \ldots, \alpha_{kT}^{(j)})^T$. We impose next a Gamma prior on these hyperparameters::

$$p(\boldsymbol{\alpha}_k^{(j)}) = \prod_{l=1}^{T} Gamma(\alpha_{kl}^{(j)}|a,b) \propto \prod_{l=1}^{T} \alpha_{kl}^{(j)^{a-1}} \exp(-b\alpha_{kl}^{(j)}) \ , \tag{5}$$

where a and b denote parameters that are a prior set to near zero values (e.g. $a = b = 10^{-6}$). The above two-stage hierarchical prior is actually a Student-t distribution [27]. This is a heavy tailed prior distribution that enforces most of the values $\alpha_{kl}^{(j)}$ to be large, thus the corresponding $w_{kl}^{(j)}$ are set zero and eliminated from the model. In this way the complexity of the regression models is controlled in an automatic and elegant way and over-fitting is avoided.

Now the clustering procedure has been converted into a Maximum-A-Posterior (MAP) estimation approach, in a sense of estimating the mixture model parameters that maximize the MAP log-likelihood function:

$$L(\boldsymbol{\Theta}) = \sum_{i=1}^{N} \log\{\sum_{k=1}^{K} \pi_k p(\boldsymbol{y}_i|\boldsymbol{\theta}_k)\} + \sum_{k=1}^{K}\sum_{j=1}^{2}\{\log p(\boldsymbol{w}_k^{(j)}|\boldsymbol{\alpha}_k^{(j)}) + \log p(\boldsymbol{\alpha}_k^{(j)})\} \ . \tag{6}$$

In this direction, the Expectation - Maximization (EM) algorithm [28] can be applied in order to MAP estimate the model parameters $\boldsymbol{\Theta}$. It iteratively performs two main stages: During the *E-step* the expected values of the hidden variables are calculated. In our case this includes the cluster labels of trajectories as given by the posterior probabilities:

$$z_{ik} = P(k|\boldsymbol{y}_i) = \frac{\pi_k p(\boldsymbol{y}_i|\boldsymbol{\theta}_k)}{\sum_{k'} \pi_{k'} p(\boldsymbol{y}_i|\boldsymbol{\theta}_{k'})} \ , \tag{7}$$

as well as the mean value of the translations $d_{ik}^{(j)}$ at any direction. The latter is obtained by using the fact that the posterior density of translations is also Gaussian:

$$p(d_{ik}^{(j)}|\boldsymbol{y}_i^{(j)}, k) \propto p(\boldsymbol{y}_i^{(j)}|\boldsymbol{\theta}_k^{(j)})p(d_{ik}^{(j)}) = \mathcal{N}(\hat{d}_{ik}^{(j)}, V_{ik}^{(j)}), \tag{8}$$

where

$$\hat{d}_{ik}^{(j)} = V_{ik}^{(j)} \left(\boldsymbol{y}_i^{(j)} - \Phi_i \boldsymbol{w}_k^{(j)}\right)^T \boldsymbol{\Sigma}_{ik}^{-1^{(j)}} \mathbf{1}_i \ \text{ and } \ V_{ik}^{(j)} = \left(\mathbf{1}_i^T \boldsymbol{\Sigma}_{ik}^{-1^{(j)}} \mathbf{1}_i + \frac{1}{u_k^{(j)}}\right)^{-1} , \tag{9}$$

where $\mathbf{1}_i$ is a T_i-length vector of 1's.

In the *M-step*, the maximization of the expected value of the complete log-likelihood function (Q-function) is performed. This leads to the following update rules [26], [29]:

$$\hat{\pi}_k = \frac{\sum_{i=1}^{N} z_{ik}}{N} , \tag{10}$$

$$\hat{\boldsymbol{w}}_k^{(j)} = \left[\sum_{i=1}^{N} z_{ik} \Phi_i^T \boldsymbol{\Sigma}_{ik}^{-1^{(j)}} \Phi_i + \boldsymbol{A}_k^{(j)}\right]^{-1} \sum_{i=1}^{N} z_{ik} \Phi_i^T \boldsymbol{\Sigma}_{ik}^{-1^{(j)}} (\boldsymbol{y}_i^{(j)} - \hat{d}_{ik}^{(j)}) , \tag{11}$$

$$\alpha_{kl}^{(j)} = \frac{1+2a}{\hat{w}_{kl}^{2^{(j)}} + 2b} \quad \forall\, l = 1, \ldots, T , \tag{12}$$

$$\hat{\sigma}_{kl}^{2^{(j)}} = \frac{\sum_{i=1}^{N} z_{ik} \left\{ \left(\boldsymbol{y}_{il}^{(j)} - [\Phi_i \hat{\boldsymbol{w}}_k^{(j)}]_l - \hat{d}_{ik}^{(j)} \right)^2 + V_{ik}^{(j)} \right\}}{\sum_{i=1}^{N} z_{ik}} , \quad \forall\, l = 1, \ldots, T \tag{13}$$

$$\hat{u}_k^{(j)} = \frac{\sum_{i=1}^{N} z_{ik} \left(\hat{d}_{ik}^{2^{(j)}} + V_{ik}^{(j)} \right)}{\sum_{i=1}^{N} z_{ik}} . \tag{14}$$

where $[.]_l$ indicates the l-th component of the T_i-dimensional vector that corresponds to time location t_{il}.

After convergence of the EM algorithm, two kinds of information are available: At first the cluster labels of the trajectories are obtained according to the maximum posterior probability (Eq. 7). Moreover, the motion of objects are obtained from the predicted mean trajectories per cluster, i.e. $\boldsymbol{\mu}_k = (\boldsymbol{\mu}_k^{(1)}, \boldsymbol{\mu}_k^{(2)}) = (\Phi \boldsymbol{w}_k^{(1)}, \Phi \boldsymbol{w}_k^{(2)})$.

3.1 Initialization Strategy

A fundamental issue when applying the EM algorithm, is its strong dependence on the initialization of the model parameters due to its local nature. Improper initialization may lead to reaching poor local maxima of the log-likelihood, a fact that may affect significantly the performance of the method. A natural way for initialization is by randomly selecting K samples from the set of input trajectories, one for each cluster. Then, we can obtain the least-squared solution for the regression coefficients. In addition, the noise variance Σ_k is initialized by a small percentage of the total variance of all trajectories equally for each clusters, while we set the mixing weights π_k equal to $1/K$. Finally, one step of the EM algorithm is executed for further refining these parameters and calculating the MAP log-likelihood function. Several such different trials are made and the optimum solution is selected according to the likelihood function as the initial state of the parameters.

However, the above scheme cannot be easily applied to our approach due to the large variability in length (T_i) of the input trajectories which brings a practical difficulty in obtaining the least-squared solution. For this reason we have followed an alternative initialization scheme based on interpolation among successive time steps. More specifically, starting from the first time we perform periodically (e.g. every $0.05T$ frames) the k-means clustering over the points $(y_{it}^{(1)}, y_{it}^{(2)})$ until the

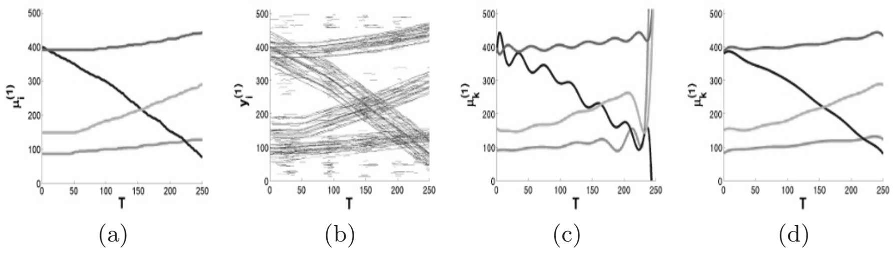


Fig. 2. The overall progress of our method to a experimental sequence of 250 images with $k = 4$ objects of different motion (a). First the input trajectories (b) are created, then an initial estimation of mean trajectories (c) are produced, and finally the predicted motion (d) is obtained

end of frame T. Then, the resulting K centers are associated with those of the previous time according to the minimum distance criterion. Finally, a linear interpolation (per cluster) is performed and thus the initial mean curves are produces used for estimating the initial values of the parameters. It must be noted that in cases where there is a large number of dense features representing the background, the initialization may diverge from the desired solution since the existence of a significant amount of outliers will affect the k-means solution. Even if during our experiments we have not faced with any such problem, treating this situation and eliminating this case still remains a future plan of study. An example of the proposed process is given in Fig. 2 adopted from a experimental data set, where both the initial interpolated trajectories and the final clustering solution are shown.

4 Experimental Results

We have studied the performance of our approach using both simulated and real examples. Some implementation details of our method are the following: At first, we have normalized spatial and temporal coordinates into the interval $[0, 1]$. Next, extracted trajectories with length less than 1% of the number of frames T were not taken into account. The same stands for those trajectories with variance less than 0.01. We have selected the mexican hat wavelet kernel $\phi(x) \propto \left(1 - \frac{x^2}{\sigma^2}\right) e^{\frac{-t^2}{2\sigma^2}}$. Experiments have shown that the method was not very sensitive to these kernel parameters. During our study we have set $\sigma = 0.3$. It must be noted that we have taken similar results for various values from the range $[0.1, 0.5]$.

Comparison has been made with the mean shift algorithm [1] which is a state of the art algorithm in visual tracking. For the mean shift algorithm, the images were represented in the RGB color space using histograms of 16 bins for each component. For initializing it we have manually selected the position and the size of each object in the first frame of the image sequence. After that, mean shift tracks the objects, using a distinct tracker for each target. The centers of

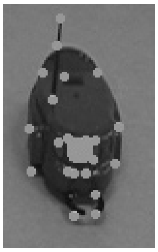


Fig. 3. Features are not uniformly distributed over the object and the center of gravity of the key points does not coincide with the center of gravity of the object. The small dots represent the features and the big dot represents their barycenter. The figure is better visualized in color.

the ellipses surrounding the targets are used to construct the mean trajectory of each object. This comparison favors mean shift in cases where the features are not uniformly distributed around the object, as the center estimated by the features may vary from the geometric center of the object (Fig. 3).

4.1 Experiments with Simulated Data Sets

The first series of experiments involves seven (7) simulated image sequences with spheres moving in predefined directions as shown in Fig. 4 and Fig. 5. Each sequence contains 130 frames of dimensions 512×512. About $1500 - 2000$ trajectories per problem were created with average length near 60 frames each. In sequences *Sim1* through *Sim5* all objects are visible during the whole sequence. In the next two experimental setups there is occlusion. In particular, in *Sim6* a sphere disappears while in *Sim7* a sphere disappears and reappears.

Since in our study we are aware of the ground truth, the performance of both tracking approaches was evaluated using two criteria:

- The mean squared error (MSE), measured in pixels, between the ground truth $\boldsymbol{r}$ and the estimated mean trajectories $\boldsymbol{\mu}$ as given by

$$MSE = \frac{1}{K \cdot T} \sum_{k=1}^{K} \sum_{j=1}^{2} ||r_k^{(j)} - \mu_k^{(j)}||^2 \ .$$

- The percentage of correctly classified trajectories (ACC). It must be noted that the input trajectories created by our method have been chosen also to evaluate mean shift algorithm. In particular, we assign every input trajectory to an object according to the smallest distance with the predicted mean trajectory.

Table 1 shows the results obtained by the proposed method and the mean shift algorithm. As it may be seen, both methods have comparable accuracy. However, our approach estimates better the true motion of the objects, as shown from the MSE criterion. Also in problem *Sim7*, mean shift fails to track the sphere that

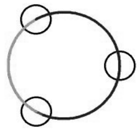
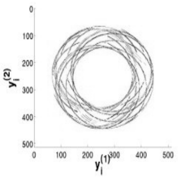
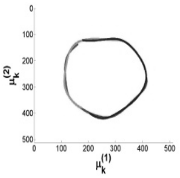
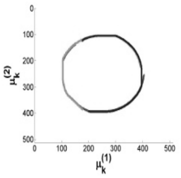
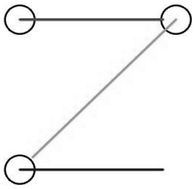
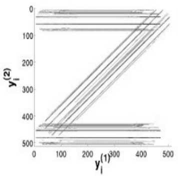
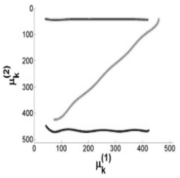
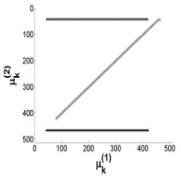
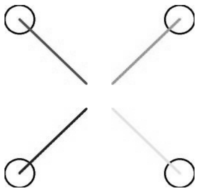
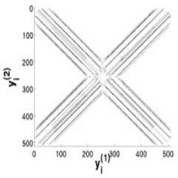
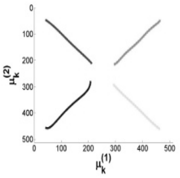
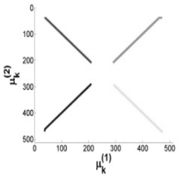
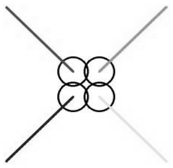
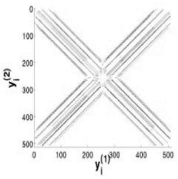
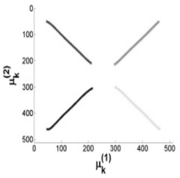
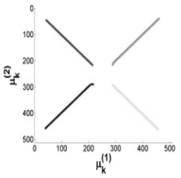
Problem	The objects with their true motion	Input trajectories	Estimated motion by our approach	Estimated motion by mean shift
Sim1				
Sim2				
Sim3				
Sim4				

Fig. 4. A part of the simulated data sets used in our experimental study. For each problem we give the real objects motion, the created input trajectories and the predicted motion as estimated by both compared approaches.

disappears and reappears and tracks the object only as long as it is visible. In the case of problems *Sim6* and *Sim7*, the frames in which a sphere is not visible are not taken into account for computing MSE. On the other hand, the proposed method correctly associates the two separated trajectories of the sphere.

4.2 Experiments with Real Data Sets

We have also studied our motion segmentation approach on five (5) real sequences shown in Fig. 6. Three of them (*Real1-3*) show mobile robots moving in various directions and two other sequences (*Real4-5*) contain two persons walking. In there, occlusions take place, as one person gets behind the other. All images are of size 512×512 pixels. During the set *Real1* ($T = 250$), the robots

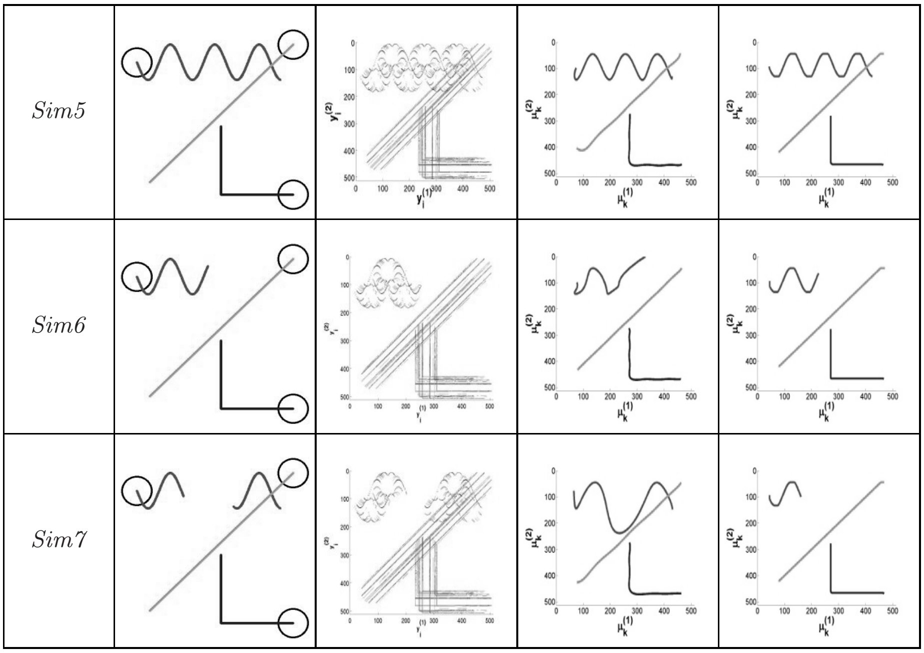


Fig. 5. A part of the simulated data sets used in our experimental study. For each problem we give the real objects motion, the created input trajectories and the predicted motion as estimated by both compared approaches.

Table 1. The performance of our approach and mean shift in terms of classification accuracy and mean squared error criteria

Problem	Our approach		Mean shift	
	MSE	ACC	MSE	ACC
Sim1	69	100%	121	100%
Sim2	10	99%	114	100%
Sim3	10	96%	114	99%
Sim4	15	97%	130	99%
Sim5	20	100%	118	100%
Sim6	29	100%	74	100%
Sim7	41	99%	lost	lost

are moving towards the borders of the image forming the vertices of a square. In *Real2* ($T = 680$), the robots are moving around the center of the image, forming a circle, while in *Real3* ($T = 500$), the robots are moving forward and backward. Finally, in *Real4* ($T = 485$) two persons are moving from the one side of the scene to the other and backwards, and in *Real5* ($T = 635$) the persons are not only moving from one side to the other many times but also they move forward and backward in the scene. In problems *Real4* and *Real5*, due to occlusions,

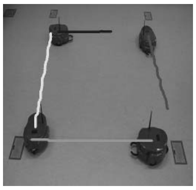
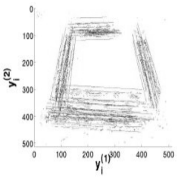
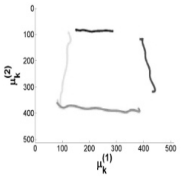
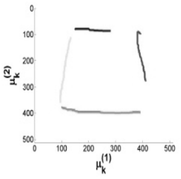
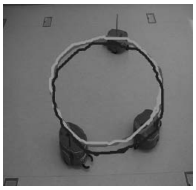
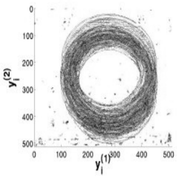
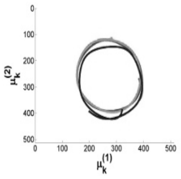
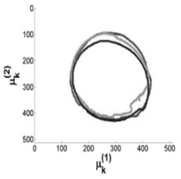
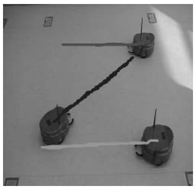
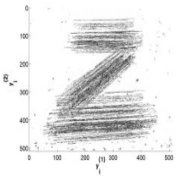
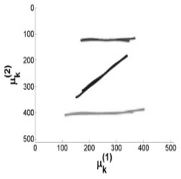
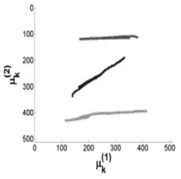
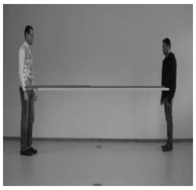
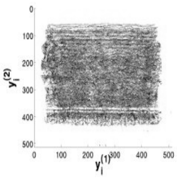
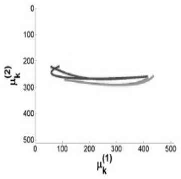
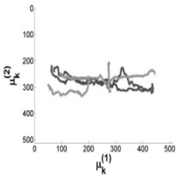
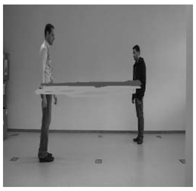
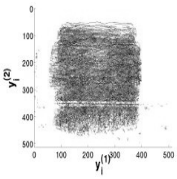
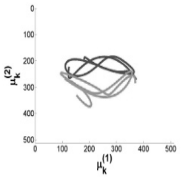
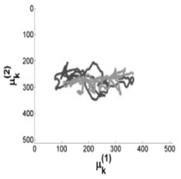
Problem	The objects with their true motion	Input trajectories	Estimated motion by our approach	Estimated motion by mean shift
Real1				
Real2				
Real3				
Real4				
Real5				

Fig. 6. The five real data set used in our experimental study. For each problem we give the real objects motion (chosen manually), the created input trajectories and the predicted motion as estimated by both comparative approaches.

the mean shift algorithm fails to track the objects while the proposed algorithm successfully follows them.

As ground truth is not provided for these sequences, only visual evaluation can be done. In problems *Real1-3* both algorithms produce approximately the same trajectories. On the contrary, in problems *Real4* and *Real5*, where we deal

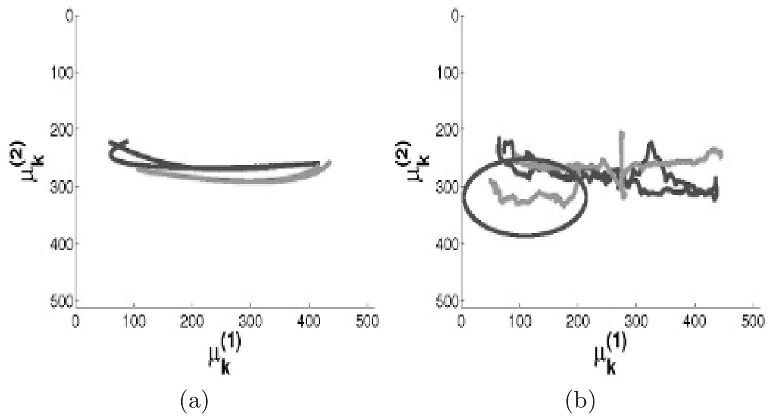


Fig. 7. Estimated trajectories for the sequence *Real4*. (a) Our method, (b) mean shift. The green (printed in light gray in black and white) trajectory in (b) corresponds to the person in black moving from the right side of the image to the left and backwards. The ellipse highlights the part of the trajectory where the person is lost, because mean shift fails to track the object due to occlusion. The figure is better visualized in color.

with articulated objects and occlusion, mean shift does not produce smooth trajectories and one object is lost. This is due to the change in the appearance of the target. When the person walks there are instances where both arms and legs are visible and instances where only one of them is present. In these cases, mean shift identifies the target with respect to its initial model and may produce abrupt changes in motion estimation. On the other hand, our method smooth out these effects through data association.

Looking in more detail the problem *Real4*, we can see the person in black disappears (because he gets behind the other person) twice during this sequence: at first, when he is moving from right to left, and then, as he is moving from left to right. Mean shift successfully follows the object before and after the first occlusion, but it fails to track it then the second one takes place. This is better depicted in Fig. 7(b) where the predicted trajectory, corresponding to the person in black, is shown in green. The part of the trajectory where the object is lost is highlighted by an ellipse. On the other hand, the proposed method successfully tracks the object in all frames (Fig. 7(a)).

5 Conclusion

We have presented a compact methodology for objects tracking based on model-based clustering trajectories of Harris corners extracted from a video sequence. Clustering is achieved through an efficient sparse regression mixture model that embodies special characteristics in order to handle trajectories of variable length, and to be translated in measurement space. Experiments have shown the abilities of our approach to automatically detect the motion of objects without any human interaction and also have demonstrated its robustness to occlusion and feature misdetection. Some directions for future study include an alternative strategy for initializing mixture model parameters during EM procedure especially in cases of occlusion, as well as to simultaneously estimate the number of objects K in the image sequences. Also, our willing is to study the performance of our method in other interesting computer vision applications, such as human action recognition [30], as well as to fully 3D motion estimation.

References

1. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(5), 564–577 (2003)
2. Cuevas, E., Zaldivar, D., Rojas, R.: Kalman filter for vision tracking. Technical Report B 05-12, Freier Universitat Berlin, Institut fur Informatik (2005)
3. Yang, M., Wu, Y., Hua, G.: Context-aware visual tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(7), 1195–1209 (2009)
4. Isard, M., Blake, A.: Condensation - conditional density propagation for visual tracking. International Journal of Computer Vision 29, 5–28 (1998)
5. Shi, J., Tomasi, C.: Good features to track. In: 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1994), pp. 593–600 (1994)
6. Wang, Z., Yang, X., Xu, Y., Yu, S.: Camshift guided particle filter for visual tracking. Pattern Recognition Letters 30(4), 407–413 (2009)
7. Zhoo, H., Yuan, Y., Shi, C.: Object tracking using SIFT features and mean shift. Computer Vision and Image Understanding 113(3), 345–352 (2009)
8. Zhao, Q., Tao, H.: Differential Earth Mover's Distance with its application to visual tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(5), 274–287 (2010)
9. Karavasilis, V., Nikou, C., Likas, A.: Visual tracking using the earth mover's distance between Gaussian mixtures and Kalman filtering. Image and Vision Computing 29(5), 295–305 (2011)
10. Black, M., Anandan, P.: The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. Computer Vision and Image Understanding 63(1), 75–104 (1996)
11. Wong, K.Y., Spetsakis, M.E.: Motion segmentation by EM clustering of good features. In: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW 2004), vol. 11, pp. 1–8. IEEE Computer Society, Los Alamitos (2004)
12. Pundlik, S.J., Birchfield, S.T.: Real-time motion segmentation of sparse feature points at any speed. IEEE Transactions on Systems, Man, and Cybernetics 38, 731–742 (2008)
13. Buzan, D., Sclaroff, S., Kollios, G.: Extraction and clustering of motion trajectories in video. In: International Conference on Pattern Recognition, pp. 521–524 (2004)
14. Alon, J., Sclaroff, S., Kollios, G., Pavlovic, V.: Discovering clusters in motion timeseries data. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 375–381 (2003)
15. Zappella, L., Llado, X., Provenzi, E., Salvi, J.: Enhanced local subspace affinity for feature-based motion segmentation. Pattern Recognition 44, 454–470 (2011)

16. Zeppelzauer, M., Zaharieva, M., Mitrovic, D., Breiteneder, C.: A novel trajectory clustering approach for motion segmentation. In: Boll, S., Tian, Q., Zhang, L., Zhang, Z., Chen, Y.-P.P. (eds.) MMM 2010. LNCS, vol. 5916, pp. 433–443. Springer, Heidelberg (2010)
17. Lauer, F., Schnorr, C.: Spectral clustering of linear subspaces for motion segmentation. In: Proc. of the 12th IEEE Int. Conf. on Computer Vision, ICCV 2009 (2009)
18. Julià, C., Sappa, A., Lumbreras, F., Serrat, J., López, A.: Motion segmentation from feature trajectories with missing data. In: Martí, J., Benedí, J.M., Mendonça, A.M., Serrat, J. (eds.) IbPRIA 2007. LNCS, vol. 4477, pp. 483–490. Springer, Heidelberg (2007)
19. 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)
20. Kim, J.H., Agapito, L.: Motion segmentation using the Hadamard product and spectral clustering. In: Proceedings of the 2009 International Conference on Motion and Video Computing, WMVC 2009, pp. 126–133. IEEE Computer Society, Los Alamitos (2009)
21. Masoud, O., Papanikolopoulos, N.P.: A novel method for tracking and counting pedestrians in real-time using a single camera. IEEE Transactions on Vehicular Technology 50(5), 1267–1278 (2001)
22. Harris, C., Stephens, M.: A combined corner and edge detection. In: Proceedings of The Fourth Alvey Vision Conference, pp. 147–151 (1988)
23. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI 1981), pp. 674–679 (1981)
24. Lowe, D.G.: Distinctive image features from Scale-Invariant keypoint. International Journal of Computer Vision 60(2), 91–110 (2004)
25. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Gool, L.V.: A comparison of affine region detectors. International Journal of Computer Vision 65(1/2), 43–72 (2005)
26. Gaffney, S.: Probabilistic curve-aligned clustering and prediction with regression mixture models. PhD thesis, Department of Computer Science, University of California, Irvine (2004)
27. Tipping, M.: Sparse Bayesian Learning and the Relevance Vector Machine. Journal of Machine Learning Research 1, 211–244 (2001)
28. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. J. Roy. Statist. Soc. B 39, 1–38 (1977)
29. Blekas, K., Nikou, C., Galatsanos, N., Tsekos, N.: A regression mixture model with spatial constraints for clustering spatiotemporal data. Intern. Journal on Artificial Intelligence Tools 17(5), 1023–1041 (2008)
30. Oikonomopoulos, A., Patras, I., Pantic, M., Paragios, N.: Trajectory-based representation of human actions. In: Huang, T.S., Nijholt, A., Pantic, M., Pentland, A. (eds.) ICMI/IJCAI Workshops 2007. LNCS (LNAI), vol. 4451, pp. 133–154. Springer, Heidelberg (2007)