A novel framework for motion segmentation and tracking by clustering incomplete trajectories

Vasileios Karavasilis¹, Konstantinos Blekas, Christophoros Nikou *

Department of Computer Science, University of Ioannina, PO Box 1186, 45110 Ioannina, Greece

A R T I C L E   I N F O
Article history:
Received 1 September 2011
Accepted 31 July 2012
Available online 21 August 2012

Keywords:
Motion segmentation
Visual feature tracking
Trajectory clustering
Sparse regression modeling

A B S T R A C T
In this paper, we present a framework for visual object tracking based on clustering trajectories of image key points extracted from an image sequence. The main contribution of our method is that the trajectories are automatically extracted from the image 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 their motion. Numerical results demonstrate the ability of the proposed method to offer more accurate and robust solutions in comparison with other tracking approaches, such as the mean shift tracker, the camshift tracker and the Kalman filter.

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 making 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, using a set of measurements from image frames the target's position is estimated.

Tracking algorithms can be classified into two main categories [1, 2]. The first one 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 [3] belongs to such family which can be successfully applied when the assumptions made about the model of the motion are adequately satisfied, including cases of occlusion. Alternatively, particle filters [4], including the condensation algorithm [5], are more general tracking methods without assuming any specific type of densities.

Finally, there are methods relying on feature extraction and tracking using optical flow techniques [6]. A general drawback of this family is that the type of object's motion must be known and modeled correctly.

On the other hand, there are approaches which are based on target representation and localization and assume a probabilistic model for the object's appearance in order to estimate its location. More specifically, color or texture features of the object masked by an isotropic kernel are used to create a histogram [1]. Then, the object's position is estimated by minimizing a cost function between the model's histogram and candidate histograms of the next image. A characteristic method in this category is the mean shift algorithm [1] and its extensions [7, 8], where the object is supposed to be surrounded by an ellipse while the histogram is constructed from its internal pixel values. Also, algorithms based on the minimization of the differential Earth mover's distance [9, 10] belong to this category.

The above methods track only one object at a time. Other methods that simultaneously track many objects have also been proposed [11, 12]. In [13], multiple objects are tracked by using Graph Cuts [14] over some observations (i.e. possible locations of the object) which are extracted. Another approach is to employ level-sets to represent each object to be tracked [15, 16] which may also be useful to handle the case of multiple object tracking [17, 18] by optimally grouping regions whose pixels have similar feature signatures. An application in vehicle tracking is presented in [19] where multiple vehicles are tracked by initially assigning
Motion segmentation constitutes a significant application of tracking algorithms, which aims at identifying moving objects in an image 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 to the number of individual motions or their parameters. It has been considered as an optical flow estimation \cite{23} where violations of brightness constancy and spatial smoothness assumptions are addressed. In \cite{24}, an alternative scheme is used where small textured patches with uniform optical flow are detected and clustered into layers, each one having an affine flow. Likewise in \cite{25}, image features are clustered into groups and the number of groups is updated automatically over time in.

Trajectories of image key-points extracted from an image sequence have also been used for tracking. Grouping 3D trajectories is proposed in \cite{26} using an agglomerative clustering algorithm where occlusions are handled by multiple tracking hypotheses. Finite mixtures of Hidden Markov Models (HMMs) were also employed in \cite{27} where training is made using the EM algorithm. Zappella et al. \cite{28} assume that trajectories belonging to different objects lie in different subspaces, thus, the segmentation can be obtained by grouping together all the trajectories that generate these subspaces. The grouping is obtained by the eigenvectors of an affinity matrix which contains the pairwise distances between trajectories computed in the corresponding subspaces. The number of motions can be estimated based on the number of eigenvalues of the symmetric normalized Laplacian matrix. In \cite{29}, a two stage procedure is proposed: Initially, an iterative clustering scheme is applied that groups temporally overlapping trajectories with similar velocity direction and magnitude. Next, the clusters created are merged each other covering larger time spans.

Furthermore, spectral clustering approaches were also proposed, such as in \cite{30} where the motions of the tracked feature points are modeled by linear subspaces, and the approach in \cite{31} where missing data from the trajectories are filled in by a matrix factorization method. Moreover, Yan and Pollefeys \cite{32} estimate a linear manifold for every trajectory and then spectral clustering is employed to separate these subspaces. In \cite{33}, motion segmentation is accomplished by computing the shape interaction matrix of individual motions or their parameters. It has been considered as an optical flow estimation \cite{23} where violations of brightness constancy and spatial smoothness assumptions are addressed. In \cite{24}, an alternative scheme is used where small textured patches with uniform optical flow are detected and clustered into layers, each one having an affine flow. Likewise in \cite{25}, image features are clustered into groups and the number of groups is updated automatically over time in.

Trajectories of image key-points extracted from an image sequence have also been used for tracking. Grouping 3D trajectories is proposed in \cite{26} using an agglomerative clustering algorithm where occlusions are handled by multiple tracking hypotheses. Finite mixtures of Hidden Markov Models (HMMs) were also employed in \cite{27} where training is made using the EM algorithm. Zappella et al. \cite{28} assume that trajectories belonging to different objects lie in different subspaces, thus, the segmentation can be obtained by grouping together all the trajectories that generate these subspaces. The grouping is obtained by the eigenvectors of an affinity matrix which contains the pairwise distances between trajectories computed in the corresponding subspaces. The number of motions can be estimated based on the number of eigenvalues of the symmetric normalized Laplacian matrix. In \cite{29}, a two stage procedure is proposed: Initially, an iterative clustering scheme is applied that groups temporally overlapping trajectories with similar velocity direction and magnitude. Next, the clusters created are merged each other covering larger time spans.

Furthermore, spectral clustering approaches were also proposed, such as in \cite{30} where the motions of the tracked feature points are modeled by linear subspaces, and the approach in \cite{31} where missing data from the trajectories are filled in by a matrix factorization method. Moreover, Yan and Pollefeys \cite{32} estimate a linear manifold for every trajectory and then spectral clustering is employed to separate these subspaces. In \cite{33}, motion segmentation is accomplished by computing the shape interaction matrix of 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 \cite{34}, 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 novel framework for visual target tracking based on model-based clustering trajectories of key points (Fig. 1). The proposed method creates trajectories of image features (e.g. Harris corner \cite{35}). However, key point tracking introduces an additional difficulty since 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 then converted into a clustering problem 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) sparse properties over the regression coefficients, (b) it is allowed to be translated in measurement space and (c) its noise covariance matrix is diagonal and not spherical as it is commonly used. The above properties are incorporated through a Bayesian regression modeling framework, where the EM algorithm can be applied for estimating the model parameters. Special care is given for initializing the EM algorithm 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 \cite{1}.

A preliminary version of this work was presented in \cite{36}. Here, we made a more extended evaluation of the proposed method. The experimental results involve comparisons with more methods, the evaluation of the method using image sequences of the Hopkins 155 dataset \cite{37} and the analysis of the algorithm using more features for the generation of trajectories.

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 conclusions is drawn in Section 5.

2. Extracting trajectories

Trajectories are constructed by tracking points in each frame of the image 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 \cite{35} and standard optical flow for the data association step \cite{38}. Let us notice that any other scale or affine invariant features \cite{39,40} 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 $Y = \{\mathbf{y}_i\}_{i=1...N}$ be a list of trajectories with $N$ being unknown beforehand. Each individual trajectory $\mathbf{y}_i$ 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 $Y$ is empty. In every image frame, Harris corners are detected and the optical flow vector at each corner is estimated \cite{6}. Then, each corner found in the current image frame is attributed to a trajectory that already exists, or a new trajectory is created having 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 closest one.
- Otherwise, a new trajectory is constructed containing only the corner under consideration.

In Fig. 2, 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 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 trajectories corresponding to the same key point and two additional trajectories involving two distinct key points.

Once the list $Y$ is created, the trajectories of the corner points that belong to the background are eliminated. This is achieved by removing the trajectories having a small variance, according to a predefined-threshold value, as well as the trajectories of small length (e.g. 1% of the number of frames). The complete procedure is described in Algorithm 1.
Algorithm 1. Trajectories construction algorithm

1: function CREATETRAJECTORIES(|Im|)
2: Input: an image sequence |Im|.
3: Output: a list of trajectories Y.
4: Y = ∅.
5: for every image |Im(t)| in 1...T do
6: Detect corners \( \{c_{\text{init}}^{(t)}\}_{t=1,...,T} \) and estimate optical flow \( \{f_{\text{flow}}^{(t)}\}_{t=1,...,T} \) in each one.
7: for every corner \( \{c_{\text{init}}^{(t)}\}_{t=1,...,T} \) detected in |Im(t)| do
8: if \( y_i \) has its last corner \( c_{\text{init}} \) in the image \( t-1 \) and its optical flow \( f_{\text{flow}} \) points to the current corner, i.e., \( c_{\text{init}} + f_{\text{flow}} \approx c_{\text{flow}} \) then
9: Insert \( c_{\text{flow}} \) into \( y_i \).
10: else if \( y_i \) has its the window around its last corner \( c_{\text{init}} \) similar to the window around the current corner \( c_{\text{flow}} \) then
11: Insert \( c_{\text{flow}} \) into \( y_i \).
12: else
13: Insert a new trajectory \( y_i \) with only \( c_{\text{init}} \) into Y.
14: end if
15: end for
16: end for
17: Eliminate trajectory \( y_i \) with small variation in its corners coordinates.
18: end function

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 of tracking is to detect \( K \) independently moving objects in the scene and simultaneously estimate their characteristic motion. This can be seen as a clustering problem.

A 2D trajectory \( y_i = (y_{i1}, y_{i2}) \) consists of two directions: (1) horizontal and (2) vertical. It is defined by a set of \( T_i \) points \( \{y_{i1}, y_{i2}, ..., y_{i1}, y_{i2}\} \), corresponding to \( T_i \) image positions \( (t_1, ..., t_{T_i}) \) of the image sequence. It is important to note 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 \( y_i \) of any direction \( j = (1,2) \) can be modeled through the following functional form:

\[
y_{ij} = \Phi_i w_{ij} + d_{ij} + e_{ij}.
\]

where \( \Phi_i \) is the design kernel matrix (common for both directions) of size \( T_i \times T_i \) and \( w = \{w_1, ..., w_T\} \) is the vector of the \( T \) unknown regression coefficients.

At first, we assume that the input space consists of the time instances \( (t_1, ..., t_T) \) which correspond to the \( T \) frames of the image sequence. Having that in mind, Eq. 1 is the standard linear regression model [41] with a global translation term added, where \( y_{ij} \) is a vector of length \( T_i \) containing the values of the \( i \)th observation of the horizontal \((j = 1)\) and vertical \((j = 2)\) directions. The \( i \)th design matrix \( \Phi_i \) is generated by keeping the \( T_i \) rows of the global design matrix \( \Phi \) that correspond to time instances \( (t_1, ..., t_{T_i}) \), where the

ith key point exists. The global matrix \( \Phi \) is a kernel matrix of size \( T \times T \) with elements calculated by a kernel function, i.e. \( \Phi(k, l) = k(t_k, t_l) \), where \( t_k, t_l \in \{t_1, ..., t_T\} \). In our work, we considered the mexican hat wavelet kernel \( k(t_k, t_l) = \frac{2}{\sqrt{\pi}} \left[ \frac{4t_k^2}{t_l^2} \right] e^{-\frac{t_k^2}{t_l^2}} \), though any other kernel function could be used (e.g. Gaussian).

Also, in the above equation, we assume a translation scalar term \( d_{ij} \) that allows for the entire trajectory to be translated globally [42], see Fig. 3. Incorporating such term results in a regression model that allows for arbitrary translations in measurement space. In our case, the features are distributed around the edges of the objects and the trajectories of the key points are translated in order to be aligned with the trajectory of the center of gravity of the object. Finally, the error term \( e_{ij} \) in the above formulation is a \( T_i \)-dimensional vector that is assumed to be zero-mean Gaussian and independent over time:

\[
e_i \sim \mathcal{N}(0, \Sigma_i). \tag{2}
\]

with a diagonal covariance matrix \( \Sigma_i = \text{diag}(\sigma_{i1}^2, ..., \sigma_{i2}^2) \). More specifically, \( \Sigma_i \) is a block matrix of a \( T \times T \) diagonal covariance matrix that corresponds to the noise variance of \( T \) frames.

Under these assumptions, the conditional density of a trajectory is also Gaussian:

\[
p(y_i | w_i, \Sigma_i) = \mathcal{N}(0, \Sigma_i).
\]  \tag{3}

Moreover, we consider the scalar \( d_{ij} \) to be a zero-mean Gaussian variable with variance \( w_{ij} \):

\[
d_{ij} \sim \mathcal{N}(0, w_{ij}). \tag{4}
\]

Thus, we can further integrate out \( d_{ij} \) to obtain the marginal density for \( y_i \) which is also Gaussian:

\[
p(y_i | \theta) = \int p(y_i | w_i, \Sigma_i, d_{ij}) p(d_{ij}) \, dd_{ij} = \mathcal{N}(\Phi_i w_i, \Sigma_i + w_{ij}I).
\]  \tag{5}

where \( t \) is a matrix of 1’s of size \( T_i \times T_i \). The marginal density is implicitly conditioned on the parameters \( \theta = (w, \Sigma, \Sigma_i) \).

In our study we consider that every object \( k \) can be described by a unique functional regression model, as given by the set parameters \( \theta_k = \{\theta_k^1, \theta_k^2\} \), where \( \theta_k^1 = (w_k^1, u_k^1, \Sigma_k^1) \), that fits to all trajectories belong to this object. Therefore, the objective is to dividing the set of \( N \) trajectories into \( K \) clusters. This can be described by the following regression mixture model:

\[
p(y_i | \Theta) = \sum_{k=1}^{K} \pi_k p(y_i | \theta_k) = \sum_{k=1}^{K} \pi_k p(y_i | \theta_k^1) p(y_i^2 | \theta_k^2). \tag{6}
\]

where \( \pi_k \) is the mixing weights satisfying the constraints \( \pi_k \geq 0 \) and \( \sum_{k=1}^{K} \pi_k = 1 \), while \( \Theta \) is the set of all the unknown mixture parameters, \( \Theta = (\pi, \theta_k)_{k=1}^{K} \).

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 [43] 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 \( w_{ij}^k = (w_{ij}^1, ..., w_{ij}^k)^T \). In particular, we assume first that coefficients follow a zero-mean Gaussian distribution:
where $A_{ik}^{(0)}$ is a diagonal matrix containing the $T$ elements of precisions $\varphi_{ik}^{(0)} = (\varphi_{ik}^{(0)}, \ldots, \varphi_{ik}^{(0)})^T$. We impose next a Gamma prior on these hyperparameters:

$$p(x_{ik}^{(0)}) = \prod_{i=1}^{T} \Gamma(a, b) \times \prod_{l=1}^{T} \exp(-b x_{ik}^{(0)}), \quad (8)$$

where $a$ and $b$ denote parameters that are a priori set to values near zero. The above two-stage hierarchical prior is actually a Student's $t$-distribution [43]. This is a heavy tailed prior distribution that enforces most of the values $x_{ik}^{(0)}$ to be large, thus the corresponding $w_{ik}^{(0)}$ are set to zero and eliminated from the model. In this way, the complexity of the regression model is controlled in an automatic and elegant way and over-fitting is avoided.

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

$$L(\Theta) = \sum_{t=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k p(y_t|\theta_k) \right\} + \sum_{k=1}^{K} \sum_{l=1}^{T} \left\{ \log p\left(w_{ik}^{(0)}|\varphi_{ik}^{(0)}\right) + \log p\left(x_{ik}^{(0)}\right) \right\}. \quad (9)$$

The model can be trained using the Expectation–Maximization (EM) algorithm [44] that iteratively performs two main stages: the E-step where 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:

$$z_{ik} = P(k|y_t) = \frac{\pi_k p(y_t|\theta_k)}{\sum_{l=1}^{L} \pi_l p(y_t|\theta_l)}, \quad (10)$$

as well as the mean value of the translations $d_{ik}^{(0)}$ at any direction. The latter is obtained by using the fact that the posterior density of translations is also Gaussian:

$$p(d_{ik}^{(0)}|y_t, k) \propto p(y_t|\theta_k) p(d_{ik}^{(0)}), \quad (11)$$

where

$$d_{ik}^{(0)} = \sum_{t=1}^{N} \sum_{k=1}^{K} \left\{ \sum_{t=1}^{T} \sum_{l=1}^{T} \left( y_t = \phi_{ik}^{(0)} \right) \right\},$$

and

$$\mu_{ik}^{(0)} = \left[ \sum_{t=1}^{T} \sum_{l=1}^{T} \left( y_t = \phi_{ik}^{(0)} \right) \right] \mu_{ik}^{(0)},$$

where $\mu_{ik}^{(0)}$ and $\varphi_{ik}^{(0)}$ are the mean and precision of the $i$th component of the $T$-dimensional vector $\phi_{ik}^{(0)}$.

During the M-step, the maximization of the expected value of the complete log-likelihood function (referred as $Q$-function in the machine learning bibliography [42]) is performed:

$$Q(\theta', \theta^{(t)}) = \sum_{t=1}^{N} \sum_{k=1}^{K} \sum_{l=1}^{T} \left\{ \log \pi_k + \sum_{t=1}^{T} \left[ \frac{1}{2} \log |\varphi_{ik}^{(0)}| + \frac{1}{2} \sum_{l=1}^{T} \frac{1}{\varphi_{ik}^{(0)}} \right] \right\}$$

$$+ \sum_{t=1}^{T} \sum_{l=1}^{T} \left[ \frac{1}{2} \log \left| \mu_{ik}^{(0)} - \mu_{ik}^{(0)} \right|^2 + \frac{1}{2} \mu_{ik}^{(0)} \right]$$

Maximizing (13) with respect to the model parameters leads to the following update rules [42,45]:

$$\pi_k = \frac{\sum_{t=1}^{T} z_{ik}}{N}, \quad (14)$$

$$w_{ik}^{(0)} = \left( \sum_{t=1}^{T} \sum_{l=1}^{T} \left( y_t = \phi_{ik}^{(0)} \right) \right)^{-1} \sum_{t=1}^{T} \sum_{l=1}^{T} \left( y_t = \phi_{ik}^{(0)} \right) \left( \phi_{ik}^{(0)} - \mu_{ik}^{(0)} \right), \quad (15)$$

$$x_{ik}^{(0)} = \frac{1 + 2a}{\sum_{l=1}^{L} + 2b} \quad \forall \; l = 1, \ldots, T, \quad (16)$$

$$\varphi_{ik}^{(0)} = \frac{\sum_{t=1}^{T} \sum_{l=1}^{T} \left( y_t = \phi_{ik}^{(0)} \right) \left( \phi_{ik}^{(0)} - \mu_{ik}^{(0)} \right) \left( \phi_{ik}^{(0)} - \mu_{ik}^{(0)} \right)}{\sum_{t=1}^{T} \sum_{l=1}^{T} \left( y_t = \phi_{ik}^{(0)} \right)} \quad \forall \; l = 1, \ldots, T, \quad (17)$$

$$\mu_{ik}^{(0)} = \frac{\sum_{t=1}^{T} \sum_{l=1}^{T} \left( y_t = \phi_{ik}^{(0)} \right)}{\sum_{t=1}^{T} \sum_{l=1}^{T} \left( y_t = \phi_{ik}^{(0)} \right)}, \quad (18)$$

where $\mu_{ik}^{(0)}$ indicates the $i$th component of the $T$-dimensional vector that corresponds to time location $t_{ik}$.

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 value (Eq. (10)). Moreover, the motion of objects are obtained from the predicted mean trajectories per cluster, i.e. $\mu_{ik}^{(0)} = \left( \mu_{ik}^{(1)}, \mu_{ik}^{(2)} \right) = \left( \phi_{ik}^{(1)}, \phi_{ik}^{(2)} \right)$.
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-squares solution for initializing the regression coefficients. In addition, the noise variance $R_k$ is initialized by a small percentage of the total variance of all trajectories equally for each clusters, while we set the mixing weights $p_k$ 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 executed and the solution with the maximum MAP likelihood function is selected for initializing model 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 a more robust initialization scheme based on the interpolation among successive time steps. More specifically, starting from the first time step we perform periodically (e.g. at every 5$T_i$ frames) the $k$-means clustering over the points $(y_{i1}^{(1)}, y_{i2}^{(1)})$. 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 resulting $K$ curves are used for initializing the parameters of the $K$ clusters. It must be noted that in cases where there is a large number of features representing the background, the initialization may diverge from the desired solution since the existence of a significant amount of outliers affects the $k$-means solution. Even if during our experiments we have not faced with any such problem, treating with this situation still remains a future plan of study. An example of the proposed process is given in Fig. 4 adopted from an experimental data set, where both the initial interpolated trajectories (Fig. 4c) and the final clustering solution are shown (Fig. 4d).

4. Experimental results

We have evaluated the performance of our approach using both simulated and real examples. Demonstration videos are available at http://www.cs.uoi.gr/~vkaravas/motion_segmentation_and_tracking. Some implementation details of our method are the fol-
The small dots represent the features and the big dot represents their barycenter. Gravity of the key points does not coincide with the center of gravity of the object. In this figure legend, the reader is referred to the web version of this article. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The first series of experiments involves seven (7) simulated image sequences with spheres moving in predefined directions as shown in Figs. 6 and 7. Each image sequence contains 130 frames of size 512 × 512. The value of $N$ varies from 1500 to 2000 trajectories per case, with average length around $T = 60$ frames each trajectory. In the first five problems, no occlusions are simulated (all objects are visible in every frame). In the rest two problems, occlusion is happened, where in Sim6 a sphere disappears while in Sim7 a sphere disappears and reappears.

Since we are aware of the ground truth, the performance of all tracking approaches was evaluated using two criteria:

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

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

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

The depicted results are presented in Table 1. As it is obvious, we took comparable results in the first three approaches Sim1, Sim2 and Sim3. However, in the fourth problem (Sim4), our approach and mean shift successfully track the objects, while camshift and the Kalman filter are failed. This is happened in the case of camshift due to the fact that the objects are overlapping in the initial frame, and after some iterations the result is an ellipse with its center located at the center of the image and whose size is increased in order to contain all the objects inside it. Moreover, Kalman filter fails because it uses camshift to obtain the measurements. On the other hand, mean shift tracks the objects due to the fact that in this implementation we decided not to integrate a scale change, as in camshift. In problem Sim5, camshift and Kalman filter fail again, because two objects (the one with the green trajectory and the one with the blue trajectory in Fig. 7) pass near each other and camshift makes the target ellipse bigger in order to include both objects. In problem Sim6 one object suddenly disappears (the one with the red in Fig. 7). Finally, in problem Sim7 the object disappears and reappears in another position after some time. Mean shift fails to track the sphere that disappears and reappears and tracks the object only as long as it is visible. Camshift and Kalman filter, as they take into account scale changes, increase the size of the ellipse and track a nearby object. On the other hand, the proposed method correctly associates the two separated trajectories of the sphere. Finally, it must be noted that in the case of both problems Sim6 and Sim7, the frames in which a sphere is not visible are not taken into account for computing the MSE criterion.

4.2. Experiments with real data sets

We have also evaluated our motion segmentation approach using five (5) real datasets shown in Fig. 8 containing images of size 512 × 512. All of these videos were created in our laboratory and they may be downloaded at http://www.cs.uoi.gr/~vkaravas/motion_segmentation_and_tracking. The first three of them (Real1–3) show mobile robots moving in various directions: In Real1 ($T = 250$), the robots are moving towards the borders of the image forming the vertices of a square. In Real2 ($T = 680$), the...
Table 6. Comparative results with four artificial datasets. For each problem we give the true object motion, the created input trajectories and the estimated motion by all approaches.

<table>
<thead>
<tr>
<th>Simulated datasets (A)</th>
<th>Sim1</th>
<th>Sim2</th>
<th>Sim3</th>
<th>Sim4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True motion</strong></td>
<td><img src="image1" alt="True motion" /></td>
<td><img src="image2" alt="True motion" /></td>
<td><img src="image3" alt="True motion" /></td>
<td><img src="image4" alt="True motion" /></td>
</tr>
<tr>
<td><strong>Input trajectories</strong></td>
<td><img src="image5" alt="Input trajectories" /></td>
<td><img src="image6" alt="Input trajectories" /></td>
<td><img src="image7" alt="Input trajectories" /></td>
<td><img src="image8" alt="Input trajectories" /></td>
</tr>
<tr>
<td><strong>Our approach</strong></td>
<td><img src="image9" alt="Our approach" /></td>
<td><img src="image10" alt="Our approach" /></td>
<td><img src="image11" alt="Our approach" /></td>
<td><img src="image12" alt="Our approach" /></td>
</tr>
<tr>
<td><strong>Mean shift</strong></td>
<td><img src="image13" alt="Mean shift" /></td>
<td><img src="image14" alt="Mean shift" /></td>
<td><img src="image15" alt="Mean shift" /></td>
<td><img src="image16" alt="Mean shift" /></td>
</tr>
<tr>
<td><strong>Camshift</strong></td>
<td><img src="image17" alt="Camshift" /></td>
<td><img src="image18" alt="Camshift" /></td>
<td><img src="image19" alt="Camshift" /></td>
<td><img src="image20" alt="Camshift" /></td>
</tr>
<tr>
<td><strong>Kalman filter</strong></td>
<td><img src="image21" alt="Kalman filter" /></td>
<td><img src="image22" alt="Kalman filter" /></td>
<td><img src="image23" alt="Kalman filter" /></td>
<td><img src="image24" alt="Kalman filter" /></td>
</tr>
<tr>
<td>Simulated datasets (B)</td>
<td>Sim5</td>
<td>Sim6</td>
<td>Sim7</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td><strong>True motion</strong></td>
<td><img src="image" alt="Sim5 True Motion" /></td>
<td><img src="image" alt="Sim6 True Motion" /></td>
<td><img src="image" alt="Sim7 True Motion" /></td>
<td></td>
</tr>
<tr>
<td><strong>Input trajectories</strong></td>
<td><img src="image" alt="Sim5 Input Trajectories" /></td>
<td><img src="image" alt="Sim6 Input Trajectories" /></td>
<td><img src="image" alt="Sim7 Input Trajectories" /></td>
<td></td>
</tr>
<tr>
<td><strong>Our approach</strong></td>
<td><img src="image" alt="Sim5 Our Approach" /></td>
<td><img src="image" alt="Sim6 Our Approach" /></td>
<td><img src="image" alt="Sim7 Our Approach" /></td>
<td></td>
</tr>
<tr>
<td><strong>Mean shift</strong></td>
<td><img src="image" alt="Sim5 Mean Shift" /></td>
<td><img src="image" alt="Sim6 Mean Shift" /></td>
<td><img src="image" alt="Sim7 Mean Shift" /></td>
<td></td>
</tr>
<tr>
<td><strong>Camshift</strong></td>
<td><img src="image" alt="Sim5 Camshift" /></td>
<td><img src="image" alt="Sim6 Camshift" /></td>
<td><img src="image" alt="Sim7 Camshift" /></td>
<td></td>
</tr>
<tr>
<td><strong>Kalman filter</strong></td>
<td><img src="image" alt="Sim5 Kalman Filter" /></td>
<td><img src="image" alt="Sim6 Kalman Filter" /></td>
<td><img src="image" alt="Sim7 Kalman Filter" /></td>
<td></td>
</tr>
</tbody>
</table>

*Fig. 7.* Comparative results with three artificial datasets. For each problem we give the true objects motion, the created input trajectories and the estimated motion by all approaches.
robots are moving around the center of the image forming a circle, and finally in Real3 \((T = 500)\), the robots are moving forward and backward. The rest two datasets \((Real4–5)\) show two persons walking, and so occlusion take place as one person gets behind the other. In particular, in Real4 \((T = 485)\) two persons are moving from one side of the scene to the other and backwards, while in Real5 \((T = 635)\) the persons additionally move forward and backward in the scene.

As ground truth is not provided in these cases we have evaluated our approach only visually. In problems \(Real1–3\) all algorithms produce approximately the same trajectories. On the contrary, in the cases of Real4 and Real5 problems, where we deal with articulated objects and occlusion, mean shift and camshift fail to properly track the persons and one of them is lost. Moreover, the trajectory of the center of the ellipse (which represents the object in mean shift and camshift) is not smooth. This is due to the change in the appearance of the target. When the person walks, there are frames where both arms and legs of a person are visible and instances where only one of them is present. In these cases, mean shift and camshift may produce abrupt changes in motion estimation because the person in the frame has changed its appearance with respect to its appearance in the first frame (which is used to initialize the model representing the object). On the other hand, our method is more accurate and produces a smooth trajectory.

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 image 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 person before and after the first occlusion, but it fails to track it when the second one takes place. Camshift fails to track the person after the first occlusion, but it follows it after the second occlusion when the person turns back. The Kalman filter successfully follows the person. This is better depicted in Fig. 9 where the predicted trajectory, corresponding to the person in black, is shown in green. The part of the trajectory where the person is lost is highlighted by an ellipse. Mean shift (Fig. 9b) follows the correct person from right to left and from left until the middle of the image when the second occlusion takes place. Then it follows the other person which moves from the middle of the image to the left. Camshift (Fig. 9c) follows the person until the first occlusion at the middle of the image. Then, it follows the other person which moves from the middle to the right and from the right to the middle. After the second occlusion is occurred, it finds the correct person again which moves from the middle to the right. On the other hand, the proposed method successfully tracks the object in all frames (Fig. 9a).

### 4.3. Experiments using the Hopkins 155 dataset

We have also used the Hopkins 155 dataset [37] to evaluate our approach, where we have selected the traffic subset that consists of 31 scenarios with two motions and seven scenarios with three motions. For each video, the extracted trajectories and the ground truth are provided. Here we must highlight that these trajectories have been manually corrected or filled by the researchers that constructed the dataset [37], so each trajectory has a value for every frame. Some representative frames of the traffic subset are presented in Fig. 10.

In Tables 2 and 3, the performance of our algorithm is presented in terms of the classification error. For comparison purposes, we also summarize the results obtained by 15 other methods as they are presented in the web page of the dataset (http://www.vision.jhu.edu/motion.php#results). More specifically, the average value of the classification error as calculated by our method over the corresponding dataset is shown in Table 2. For the rest of the compared methods, we present the minimum, the maximum, the mean and the median values for the average classification error over all the 15 state of art methods. This means that we have included the average classification error of each of the compared methods in the computation of each statistic (min, max, mean, median). As it may be observed, our method provides satisfactory results compared to all other methods. This is also confirmed by the ranking information, shown in the last column of Table 2. In the case of two motions, our method is ranked first among the 15 compared algorithms. Let us notice that the method sparse subspace clustering (SSC) [47] provides a similar average error. This is indicated in the last column of Table 2, showing how many methods provide the same error in average. In the case of three motions, the performance of our method is ranked in the second place behind SSC, whose average error is 0.58%.

Similar in spirit statistics are shown in Table 3 concerning the median classification error. In that case, our method accurately classifies more than half of the time series which is also the case for 7 out of 15 methods for the problems involving two motions. For the three motions problems, where the complexity increases, our algorithm is also ranked at the first place along with other three methods, namely multi stage learning (MSL) [48], local linear manifold clustering with projection to a space of dimension 5 (LLMC 5) [49] and SSC [47].

### 4.4. Experiments using other key point descriptors

In Section 2 we described how to create trajectories from an image sequence and as an example we used Harris corners [35] in order to detect salient image features and optical flow [38] to associate them between images. Apart from Harris corners, numerous interest-point detectors have been proposed in the computer vision literature, such as the scale invariant feature transform (SIFT) [39], the maximally stable extremal regions (MSERs) [50], the Hessian matrix-based affine features [40] and the speeded-up robust features (SURFs) [51]. They mainly differ to the level of the tradeoff between repeatability and complexity [40,52]. The SIFT features for example are highly repeatable but require a large computational cost. This is why SURF key point detectors have gained increasing interest, as they are faster (they are based on box filters and integral images [53]).

In order to study the consistency of the proposed method we have evaluated it in terms of the technique used for generating trajectories. Table 4 summarize the comparative results on the seven simulated datasets using Harris corners, SIFT and SURF features. For every case we give also the number of the generated trajectories. It may be observed that when the trajectories are computed using Harris corners a smaller MSE is obtained while trajectories computed by SIFT and SURF detectors have approximately similar errors but in any case higher than Harris corners. In terms of accuracy, all of the methods exhibit, in average, comparable rates.
### Real datasets

<table>
<thead>
<tr>
<th></th>
<th>Real1</th>
<th>Real2</th>
<th>Real3</th>
<th>Real4</th>
<th>Real5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True motion</strong></td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>Input trajectories</strong></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>Our approach</strong></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>Mean shift</strong></td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>Camshift</strong></td>
<td><img src="image21.png" alt="Image" /></td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
<td><img src="image25.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>Kalman filter</strong></td>
<td><img src="image26.png" alt="Image" /></td>
<td><img src="image27.png" alt="Image" /></td>
<td><img src="image28.png" alt="Image" /></td>
<td><img src="image29.png" alt="Image" /></td>
<td><img src="image30.png" alt="Image" /></td>
</tr>
</tbody>
</table>

**Fig. 8.** Comparative results with five real datasets. For each problem we give the true objects motion (chosen manually), the created input trajectories and the estimated motion by all approaches.
Fig. 9. Estimated trajectories for the dataset Real4. (a) Our method, (b) mean shift, (c) camshift. The green (printed in light gray in black and white) trajectory in (b) and (c) 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 or camshift fails to track the object due to occlusion. The figure is better visualized in color. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 10. Representative frames of the Hopkins 155 dataset. The feature points are marked using different colors in order to denote the cluster they belong to. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Statistics on the average of classification error for the traffic subset of the Hopkins 155 dataset.

<table>
<thead>
<tr>
<th></th>
<th>Our approach (%)</th>
<th>Other approaches (%)</th>
<th>Our Rank</th>
<th>In tie</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min (%)</td>
<td>Max (%)</td>
<td>Mean (%)</td>
<td>Median (%)</td>
</tr>
<tr>
<td>Two motions</td>
<td>0.02</td>
<td>0.02</td>
<td>5.74</td>
<td>2.63</td>
</tr>
<tr>
<td>Three motions</td>
<td>0.98</td>
<td>0.58</td>
<td>27.02</td>
<td>9.53</td>
</tr>
</tbody>
</table>

Table 3
Statistics on the median of classification error for the traffic subset of the Hopkins 155 dataset.

<table>
<thead>
<tr>
<th></th>
<th>Our approach (%)</th>
<th>Other approaches (%)</th>
<th>Our Rank</th>
<th>In tie</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min (%)</td>
<td>Max (%)</td>
<td>Mean (%)</td>
<td>Median (%)</td>
</tr>
<tr>
<td>Two motions</td>
<td>0.00</td>
<td>0.00</td>
<td>1.55</td>
<td>0.51</td>
</tr>
<tr>
<td>Three motions</td>
<td>0.00</td>
<td>0.00</td>
<td>34.01</td>
<td>7.22</td>
</tr>
</tbody>
</table>

Table 4
The performance of the different key point extraction methods in terms of classification accuracy (ACC) and mean squared error (MSE).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Harris corners</th>
<th>SIFT</th>
<th>SURF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Trajectories</td>
<td>MSE</td>
<td>ACC (%)</td>
</tr>
<tr>
<td>Sim1</td>
<td>3820</td>
<td>69</td>
<td>100</td>
</tr>
<tr>
<td>Sim2</td>
<td>1516</td>
<td>10</td>
<td>99</td>
</tr>
<tr>
<td>Sim3</td>
<td>2348</td>
<td>20</td>
<td>96</td>
</tr>
<tr>
<td>Sim4</td>
<td>2346</td>
<td>15</td>
<td>97</td>
</tr>
<tr>
<td>Sim5</td>
<td>1954</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>Sim6</td>
<td>1351</td>
<td>29</td>
<td>100</td>
</tr>
<tr>
<td>Sim7</td>
<td>1485</td>
<td>41</td>
<td>99</td>
</tr>
</tbody>
</table>
which indicates a larger number of trajectories, provided for example by SURF, does not necessarily ensure a better result.

5. Conclusions

In this study we have presented a compact methodology for objects tracking based on model-based clustering trajectories of Harris corners extracted from an image sequence. Clustering is achieved through an efficient sparse regression mixture model that embodies efficient 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 demonstrated its robustness to occlusion and feature misidentification.

The main advantage of our method with respect to Kalman filter is that the former handles both tracking and motion segmentation while the latter only tracks the target. Also, Kalman filter should be provided with the motion model while the method proposed here-in needs as input only the number of the objects to be tracked. Moreover, our method may be applied without the knowledge of the full trajectories by using only time series data up to the current time instant if this is imposed by the application. Finally, linear regression model can be applied in order to predict the next state [41].

Some directions for future study include an alternative strategy for initializing regression mixture model parameters during EM procedure especially in cases of occlusion, as well as a mechanism to simultaneously estimate the number of objects K in the image sequence. Also, our willing is to study the performance of our method in other interesting computer vision applications, such as human action recognition [54] and fully 3D motion estimation [30].

References
