Computer Vision and Image Understanding 116 (2012) 1135–1148

Contents lists available at SciVerse ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

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

Vasileios Karavasilis [1], Konstantinos Blekas, Christophoros Nikou *

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

ARTICLE INFO

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

ABSTRACT

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

☆ This paper has been recommended for acceptance by Y. Aloimonos.

* Corresponding author.
E-mail addresses: vkaravas@cs.uoi.gr (V. Karavasilis), kblekas@cs.uoi.gr (K. Blekas), cnikou@cs.uoi.gr (C. Nikou).

[1] V. Karavasilis is funded by the Greek State Scholarships Foundation.

1077-3142/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.cviu.2012.07.004

each pixel either to background, or foreground and applying next Kalman filter to estimate the vehicle position and associate each foreground pixel with a single object. In cases of articulated objects (e.g. pedestrians) stereo depth information have been used in order to acquire a 3D articulated structure per object and then estimate the 3D trajectory of each object [20]. In any case, combining multiple object representations could make the tracking procedure more robust [21,22].

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 [23] where violations of brightness constancy and spatial smoothness assumptions are addressed. In [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 [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 [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 [27] where training is made using the EM algorithm. Zappella et al. [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 [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 [30] where the motions of the tracked feature points are modeled by linear subspaces, and the approach in [31] where missing data from the trajectories are filled in by a matrix factorization method. Moreover, Yan and Pollefeys [32] estimate a linear manifold for every trajectory and then spectral clustering is employed to separate these subspaces. In [33], 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 [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* [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 [1].

A preliminary version of this work was presented in [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 [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* [35] and standard optical flow for the data association step [38]. Let us notice that any other scale or affine invariant features [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 $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1,\ldots,N}$ 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 [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

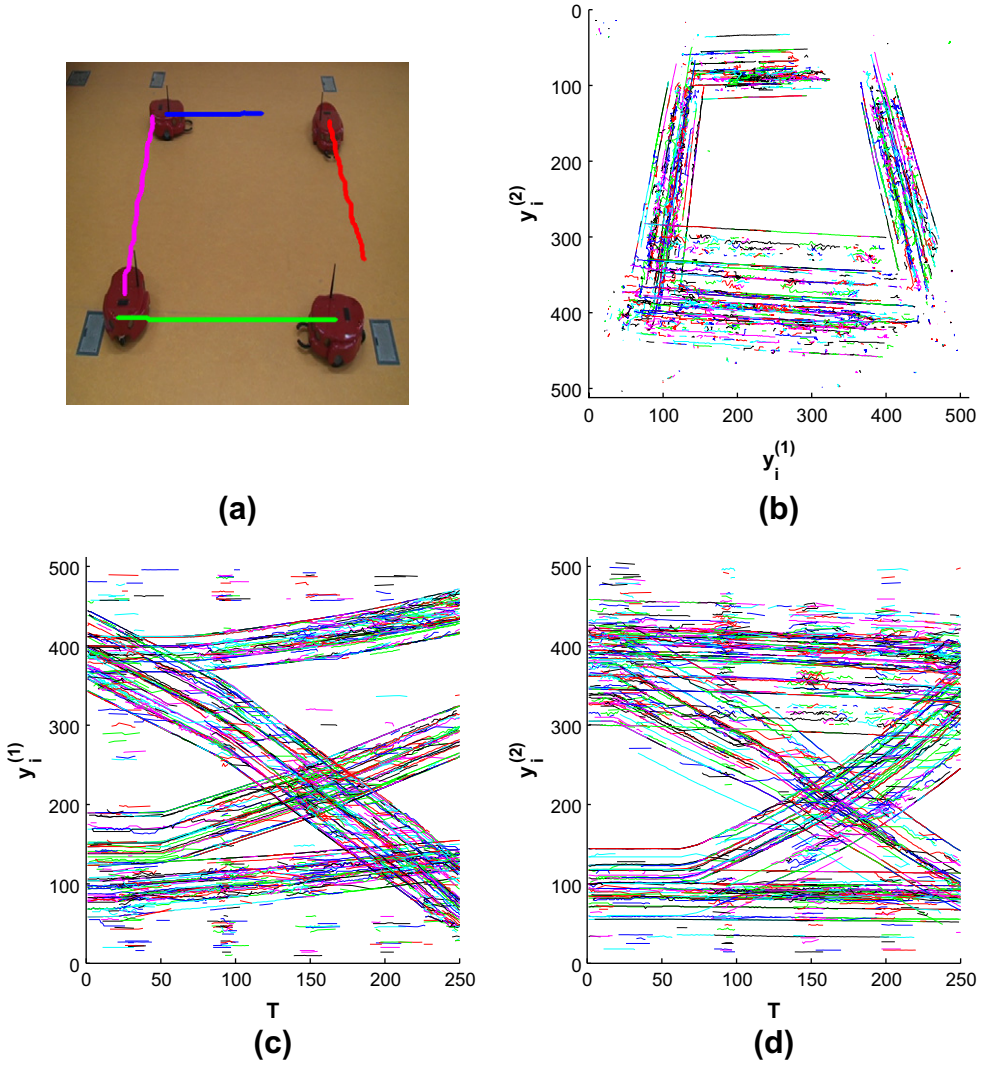

Fig. 1. Trajectories extracted from an image sequence. (a) The first frame of the image sequence showing four robots and their mean trajectories. The group of robots perform a square-like movement. (b) The trajectories of the features extracted from the image sequence. The two axes represent the horizontal and vertical coordinates. (c) The horizontal trajectories along time. (d) The vertical trajectories along time. Notice that there is a large number of short and incomplete trajectories because the features disappear and reappear during the image sequence due to illumination changes and the distance of the object from the camera.

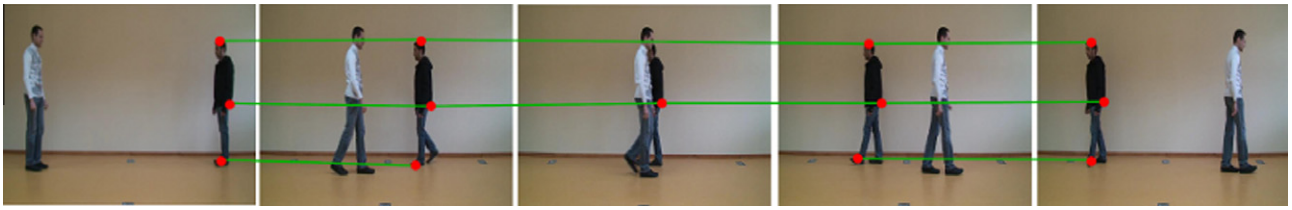

Fig. 2. Example of trajectories construction. The red dots represent the image key points and the green lines represent their trajectories. The figure is better seen in color. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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 trajectories corresponding to the same key point and two additional trajectories involving two distinct key points.

Once the list $\boldsymbol{Y}$ 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: $Y = \emptyset$.
5: **for** every image $\{im^{(t)}\}_{t=1,\ldots,T}$ **do**
6: Detect corners $\{c_l^{(t)}\}_{l=1,\ldots,L^{(t)}}$ and estimate optical flow $\{f_l^{(t)}\}_{l=1,\ldots,L^{(t)}}$ in each one.
7: **for** every corner $\{c_l^{(t)}\}_{l=1,\ldots,L^{(t)}}$ detected in $im^{(t)}$ **do**
8: **if** y_i has its last corner c_i^{last} in the image $t-1$ and its 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 y_i.
10: **else if** y_i has its the window around its last corner c_i^{last} similar to the window around the current corner $c_l^{(t)}$ **then**
11: Insert $c_l^{(t)}$ into y_i.
12: **else**
13: Insert a new trajectory y_i with only $c_l^{(t)}$ 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 $\boldsymbol{y}_i = \left(\boldsymbol{y}_i^{(1)}, \boldsymbol{y}_i^{(2)}\right)$ consists of two directions: (1) horizontal and (2) vertical. It is defined by a set of T_i points $\left\{\left(y_{i1}^{(1)}, y_{i1}^{(2)}\right), \ldots, \left(y_{iT_i}^{(1)}, y_{iT_i}^{(2)}\right)\right\}$, corresponding to T_i image positions $(t_{i1}, \ldots, t_{iT_i})$ 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 $\boldsymbol{y}_i^{(j)}$ 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 $\boldsymbol{w} = (w_1, \ldots, w_T)$ is the vector of the T unknown regression coefficients.

At first, we assume that the input space consists of the time instances $(t_1, \ldots, t_T)$ 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 $\boldsymbol{y}_i^{(j)}$ is a vector of length T_i containing the values of the ith observation of the horizontal ($j = 1$) and vertical ($j = 2$) directions. The ith design matrix Φ_i is generated by keeping the T_i rows of the global design matrix Φ that correspond to time instances $(t_{i1}, \ldots, t_{iT_i})$, where the ith key point exists. The global matrix Φ 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 $t_k, t_l \in \{t_1, \ldots, t_T\}$. In our work, we considered the mexican hat wavelet kernel $k(t_k, t_l) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}}\left(1 - \frac{(t_l - t_k)^2}{\sigma^2}\right)e^{\frac{-(t_l - t_k)^2}{2\sigma^2}}$, though any other kernel function could be used (e.g. Gaussian).

Also, in the above equation, we assume a translation scalar term $d_i^{(j)}$ 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 $\boldsymbol{e}_i^{(j)}$ in the above formulation is a T_i-dimensional vector that is assumed to be zero-mean Gaussian and independent over time:

$$e_i \sim \mathcal{N}(0, \Sigma_i), \tag{2}$$

with a diagonal covariance matrix $\Sigma_i^{(j)} = diag\left(\sigma_{t_{i1}}^{2^{(j)}}, \ldots, \sigma_{t_{iT_i}}^{2^{(j)}}\right)$. More specifically, Σ_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:

$$p\left(\boldsymbol{y}_i^{(j)} \middle| \boldsymbol{w}^{(j)}, \Sigma_i^{(j)}, d_i^{(j)}\right) = \mathcal{N}(\Phi_i \boldsymbol{w}^{(j)} + d_i^{(j)}, \Sigma_i^{(j)}). \tag{3}$$

Moreover, we consider the scalar $d_i^{(j)}$ to be a zero-mean Gaussian variable with variance $u^{(j)}$:

$$d_i^{(j)} \sim \mathcal{N}(0, u^{(j)}). \tag{4}$$

Thus, we can further integrate out d_i to obtain the marginal density for $\boldsymbol{y}_i^{(j)}$ which is also Gaussian:

$$\begin{aligned} p\left(\boldsymbol{y}_i^{(j)} \middle| \theta\right) &= \int p\left(\boldsymbol{y}_i^{(j)} \middle| \boldsymbol{w}^{(j)}, \Sigma_i^{(j)}, d_i^{(j)}\right) p\left(d_i^{(j)}\right) dd_i^{(j)} \\ &= \mathcal{N}\left(\Phi_i \boldsymbol{w}^{(j)}, \Sigma_i^{(j)} + u^{(j)} \mathbb{1}\right), \end{aligned} \tag{5}$$

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 $\theta = \{\boldsymbol{w}, u, \Sigma\}$.

In our study we consider that every object k can be described by a unique functional regression model, as given by the set parameters $\theta_k = \left\{\theta_k^{(1)}, \theta_k^{(2)}\right\}$, where $\theta_k^{(j)} = \left\{\boldsymbol{w}_k^{(j)}, u_k^{(j)}, \Sigma_k^{(j)}\right\}$, 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:

$$p(\boldsymbol{y}_i | \boldsymbol{\Theta}) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{y}_i | \theta_k) = \sum_{k=1}^{K} \pi_k p\left(\boldsymbol{y}_i^{(1)} \middle| \theta_k^{(1)}\right) p\left(\boldsymbol{y}_i^{(2)} | \theta_k^{(2)}\right), \tag{6}$$

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

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 $\boldsymbol{w}_k^{(j)} = \left(w_{k1}^{(j)}, \ldots, w_{kT}^{(j)}\right)^T$. In particular, we assume first that coefficients follows a zero-mean Gaussian distribution:

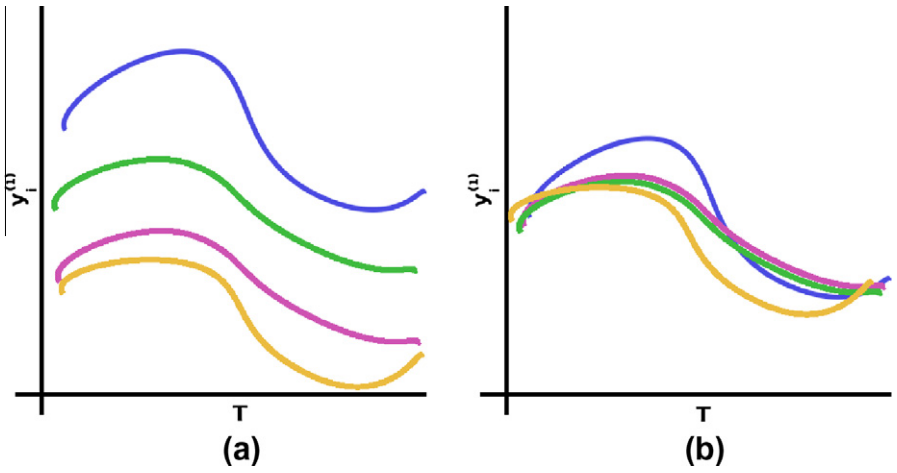

Fig. 3. The effect of the translation parameter. (a) A set of trajectories. (b) Alignment of the trajectories.

$$p\left(\boldsymbol{w}_k^{(j)}|\boldsymbol{\alpha}_k^{(j)}\right)=\mathcal{N}\left(\boldsymbol{w}_k^{(j)}|\boldsymbol{0},\boldsymbol{A}_k^{-1(j)}\right)=\prod_{l=1}^{T}\mathcal{N}\left(w_{kl}^{(j)}|0,\alpha_{kl}^{-1(j)}\right), \tag{7}$$

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

$$p(\boldsymbol{\alpha}_k^{(j)})=\prod_{l=1}^{T}Gamma\left(\alpha_{kl}^{(j)}|a,b\right)\propto\prod_{l=1}^{T}\alpha_{kl}^{(j)^{a-1}}\exp\left(-b\alpha_{kl}^{(j)}\right), \tag{8}$$

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 $\alpha_{kl}^{(j)}$ to be large, thus the corresponding $w_{kl}^{(j)}$ 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:

$$L(\boldsymbol{\Theta})=\sum_{i=1}^{N}\log\left\{\sum_{k=1}^{K}\pi_k p(\boldsymbol{y}_i|\boldsymbol{\theta}_k)\right\}+\sum_{k=1}^{K}\sum_{j=1}^{2}\left\{\log p\left(\boldsymbol{w}_k^{(j)}|\boldsymbol{\alpha}_k^{(j)}\right)+\log p\left(\boldsymbol{\alpha}_k^{(j)}\right)\right\}. \tag{9}$$

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:

$$z_{ik}=P(k|\boldsymbol{y}_i)=\frac{\pi_k p(\boldsymbol{y}_i|\boldsymbol{\theta}_k)}{\sum_{k'}\pi_{k'}p(\boldsymbol{y}_i|\boldsymbol{\theta}_{k'})}, \tag{10}$$

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:

$$p\left(d_{ik}^{(j)}\middle|\boldsymbol{y}_i^{(j)},k\right)\propto p\left(\boldsymbol{y}_i^{(j)}\middle|\boldsymbol{\theta}_k^{(j)}\right)p\left(d_{ik}^{(j)}\right)=\mathcal{N}\left(\hat{d}_{ik}^{(j)},V_{ik}^{(j)}\right), \tag{11}$$

where

$$\hat{d}_{ik}^{(j)}=V_{ik}^{(j)}\left(\boldsymbol{y}_i^{(j)}-\Phi_i\boldsymbol{w}_k^{(j)}\right)^T\boldsymbol{\Sigma}_{ik}^{-1(j)}\boldsymbol{1}_i\ \text{and}\ V_{ik}^{(j)}=\left(\boldsymbol{1}_i^T\boldsymbol{\Sigma}_{ik}^{-1(j)}\boldsymbol{1}_i+\frac{1}{u_k^{(j)}}\right)^{-1}, \tag{12}$$

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

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:

$$Q(\theta^t,\theta^{t-1})=\sum_{i=1}^{N}\sum_{k=1}^{K}z_{ik}\left\{\log\pi_k+\sum_{j=1}^{2}-\frac{1}{2}\log\left|\Sigma_k^{(j)}\right|-\frac{\left(\boldsymbol{y}_i^{(j)}-\boldsymbol{\mu}_k^{(j)}\right)^T\Sigma_k^{(j)}\left(\boldsymbol{y}_i^{(j)}-\boldsymbol{\mu}_k^{(j)}\right)}{2}\right\}+\sum_{k=1}^{K}\sum_{j=1}^{2}-\frac{1}{2}\log\left|A_k^{(j)}\right|-\frac{\boldsymbol{w}_k^{(j)^T}A_k^{(j)}\boldsymbol{w}_k^{(j)}}{2}+\sum_{k=1}^{K}\sum_{j=1}^{2}\sum_{l=1}^{T}\log\alpha_{kl}^{(j)^{a-1}}-b\alpha_{kl}^{(j)}. \tag{13}$$

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

$$\hat{\pi}_k=\frac{\sum_{i=1}^{N}z_{ik}}{N}, \tag{14}$$

$$\hat{\boldsymbol{w}}_k^{(j)}=\left[\sum_{i=1}^{N}z_{ik}\Phi_i^T\boldsymbol{\Sigma}_{ik}^{-1(j)}\Phi_i+\boldsymbol{A}_k^{(j)}\right]^{-1}\sum_{i=1}^{N}z_{ik}\Phi_i^T\boldsymbol{\Sigma}_{ik}^{-1(j)}\left(\boldsymbol{y}_i^{(j)}-\hat{d}_{ik}^{(j)}\right), \tag{15}$$

$$\alpha_{kl}^{(j)}=\frac{1+2a}{\hat{w}_{kl}^{2(j)}+2b},\quad\forall\, l=1,\ldots,T, \tag{16}$$

$$\hat{\sigma}_{kl}^{2(j)}=\frac{\sum_{i=1}^{N}z_{ik}\left\{\left(\boldsymbol{y}_{il}^{(j)}-\left[\Phi_i\hat{\boldsymbol{w}}_k^{(j)}\right]_l-\hat{d}_{ik}^{(j)}\right)^2+V_{ik}^{(j)}\right\}}{\sum_{i=1}^{N}z_{ik}},\quad\forall\, l=1,\ldots,T, \tag{17}$$

$$\hat{u}_k^{(j)}=\frac{\sum_{i=1}^{N}z_{ik}\left(\hat{d}_{ik}^{2(j)}+V_{ik}^{(j)}\right)}{\sum_{i=1}^{N}z_{ik}}. \tag{18}$$

where $[\cdot]_l$ indicates the lth 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 value (Eq. (10)). Moreover, the motion of objects are obtained from the predicted mean trajectories per cluster, i.e. $\boldsymbol{\mu}_k=\left(\boldsymbol{\mu}_k^{(1)},\boldsymbol{\mu}_k^{(2)}\right)=\left(\Phi\boldsymbol{w}_k^{(1)},\Phi\boldsymbol{w}_k^{(2)}\right)$.

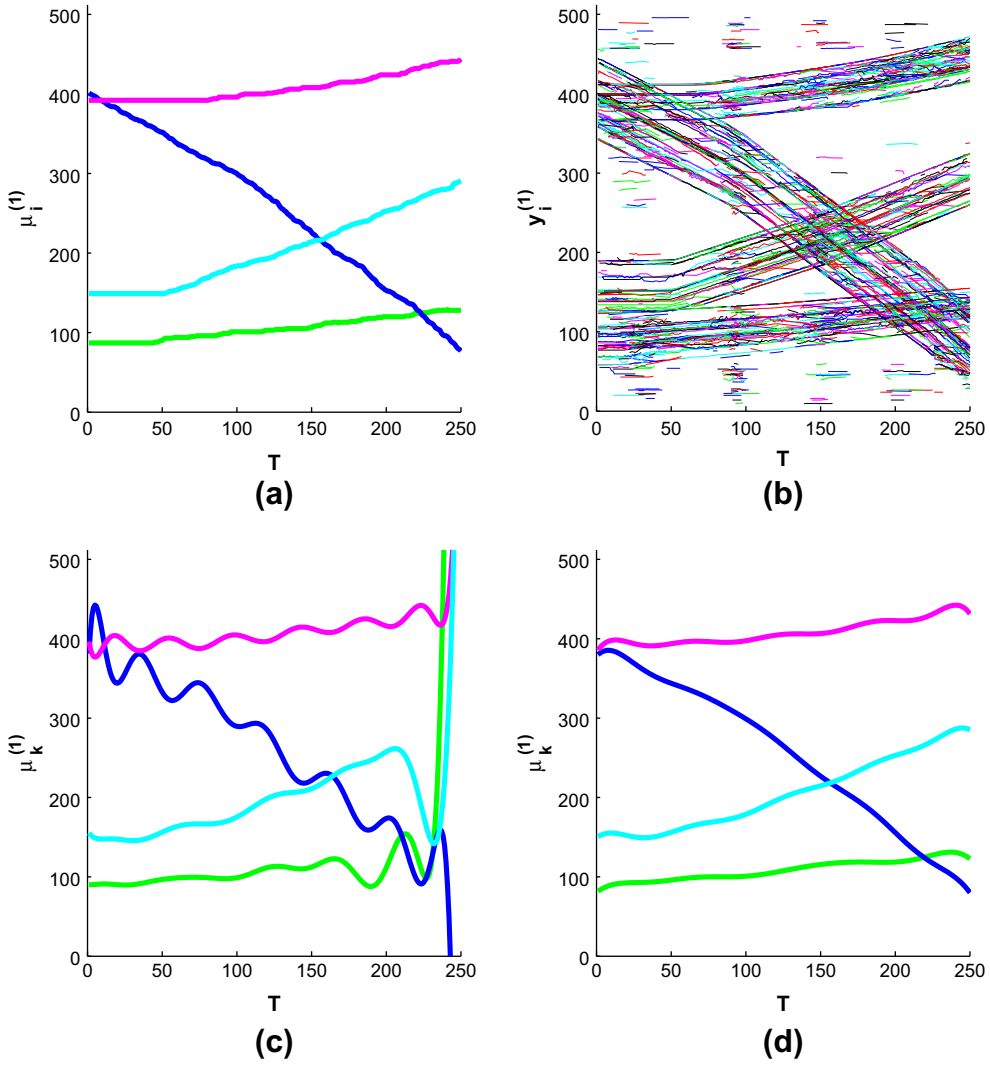

Fig. 4. The overall progress of our method applied to an experimental image sequence of 250 images with $k = 4$ objects with different motions. (a) Real trajectories, (b) input trajectories, (c) initial estimation of mean trajectories using the proposed technique, and (d) the estimated trajectories after EM convergence.

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 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 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 5T% frames) the k-means clustering over the points $\left(y_{it}^{(1)}, y_{it}^{(2)}\right)$. 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-

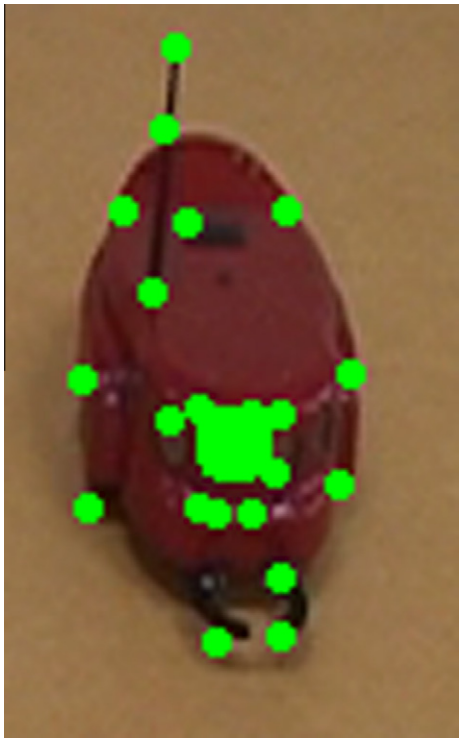

Fig. 5. Features are not uniformly distributed over the object and the center of gravity of the key points does not coincide with the center of gravity of the object. The small dots represent the features and the big dot represents their barycenter. The figure is better visualized in color. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

lowing: At first, we have normalized spatial and temporal coordinates into the interval [0,1] while the extracted trajectories either with length less than 1% of the number of frames T, or with variance less than 0.01 were not taken into account. We have selected the mexican hat wavelet kernel $k(t_k,t_l)=\frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}}\left(1-\frac{(t_l-t_k)^2}{\sigma^2}\right)e^{\frac{-(t_l-t_k)^2}{2\sigma^2}}$ for the design kernel matrix Φ, where the scalar parameter took the value of $\sigma = 0.3$. Experiments have shown that the performance of our approach is not very sensitive to this parameter, since we took similar results for values in the range of [0.1,0.5].

Comparison has been made with the mean shift algorithm [1], the camshift algorithm [46] and the Kalman filter [3], which are established algorithms 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 the camshift algorithm, the hue component of HSV color space was used to construct a 16-bin histogram. For the Kalman filter, camshift was used in order to provide measurements (the position and the size of the object). For initializing all these algorithms, we have manually selected the position and the size of each object in the first frame of the image sequence. Then, the objects are tracked, using a distinct tracker per each target. The centers of the ellipses surrounding the targets are used to construct the mean trajectory of each object. This comparison favors these algorithms 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. 5).

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 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 $\boldsymbol{r}$ and the estimated mean trajectories $\boldsymbol{\mu}$ as given by

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

- 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 thee 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[2] 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

[2] For interpretation of color in Fig. 7, the reader is referred to the web version of this article.

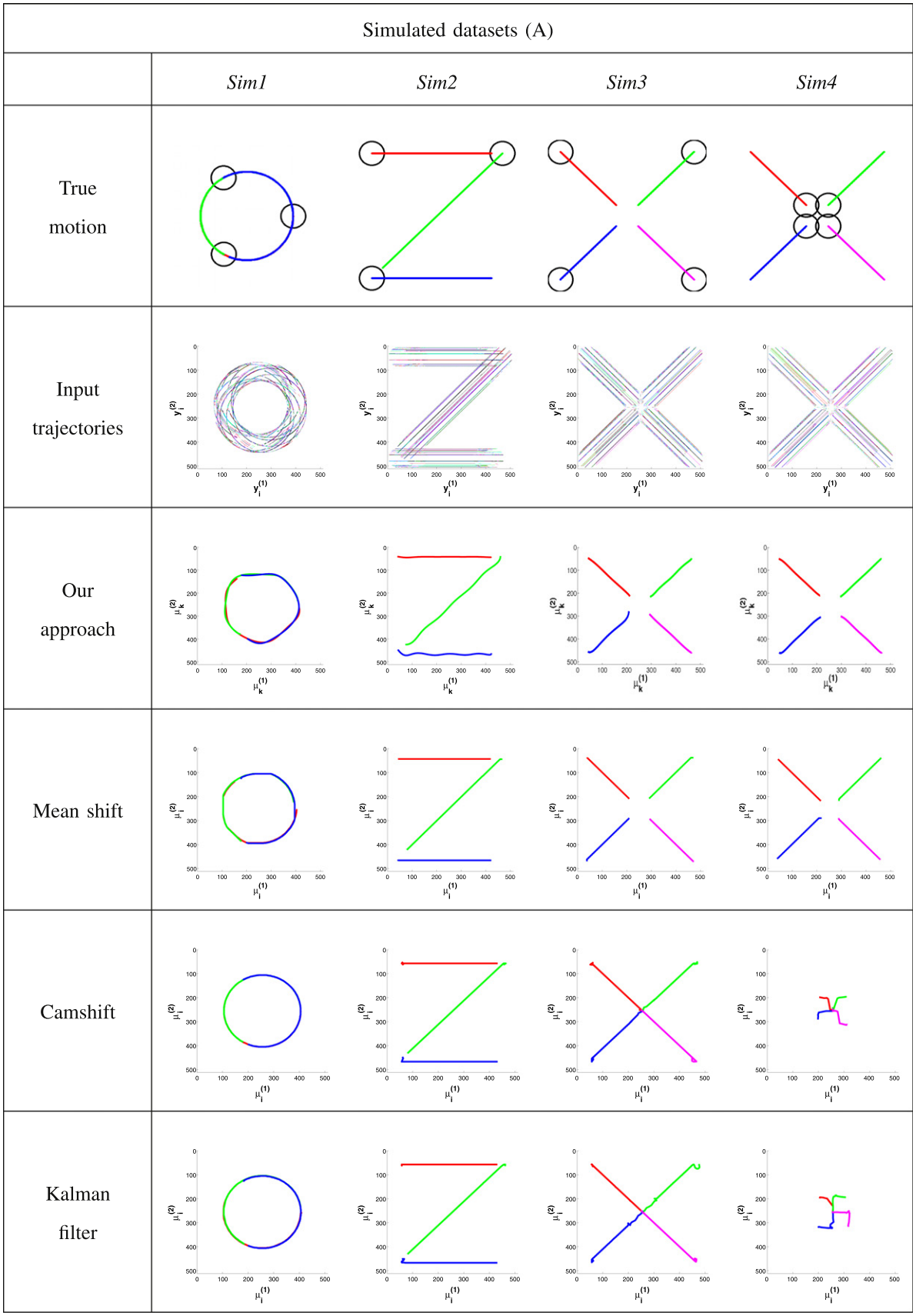

Fig. 6. Comparative results with four artificial datasets. For each problem we give the true objects motion, the created input trajectories and the estimated motion by all approaches.

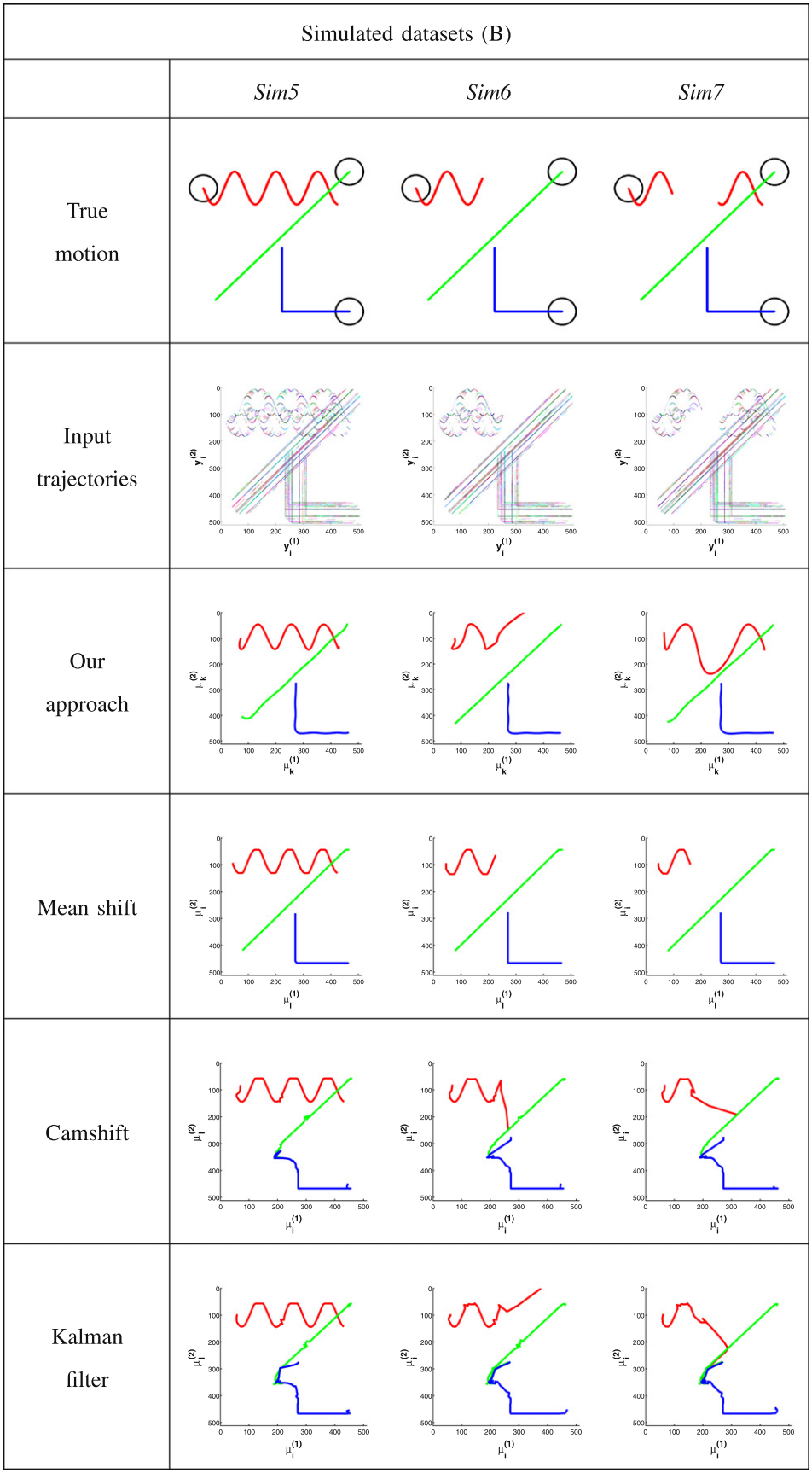

Fig. 7. Comparative results with three artificial datasets. For each problem we give the true objects motion, the created input trajectories and the estimated motion by all approaches.

Table 1
The performance of the compared methods in terms of classification accuracy (ACC) and mean squared error (MSE).

Problem	Our approach		Mean shift		Camshift		Kalman	
	MSE	ACC (%)	MSE	ACC (%)	MSE	ACC (%)	MSE	ACC (%)
Sim1	69	100	121	100	4	96	0	100
Sim2	10	99	114	100	55	100	54	100
Sim3	10	96	114	99	202	95	224	95
Sim4	15	97	130	99	Lost	Lost	Lost	Lost
Sim5	20	100	118	100	Lost	Lost	Lost	Lost
Sim6	29	100	74	100	Lost	Lost	Lost	Lost
Sim7	41	99	Lost	Lost	Lost	Lost	Lost	Lost

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 change 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 difference of Gaussian key points (SIFT DoG or SIFT for simplicity) [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

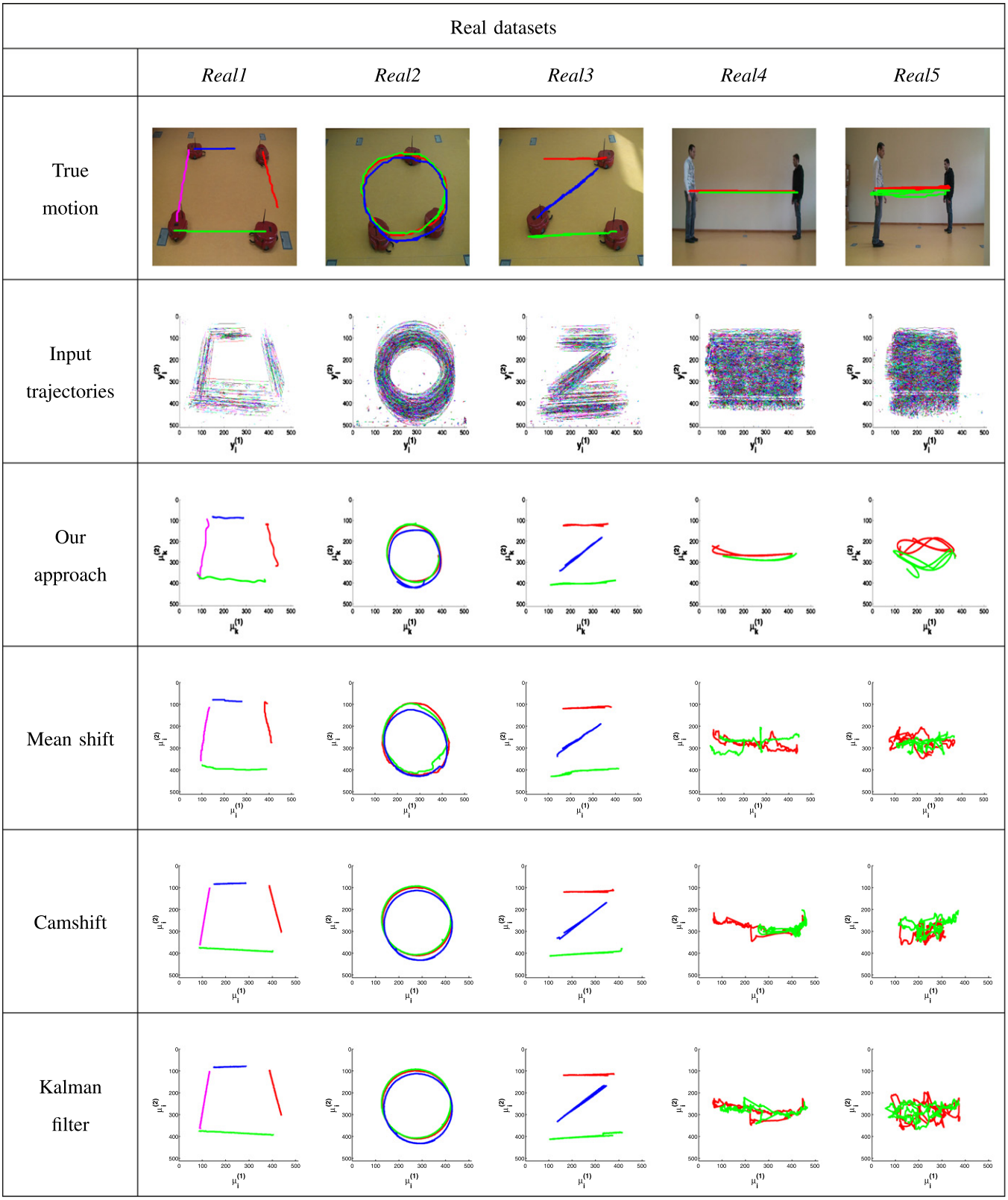

Fig. 8. Comparative results with five real datasets. For each problem we give the true objects motion (chosen manually), the created input trajectories and the estimated motion by all approaches.

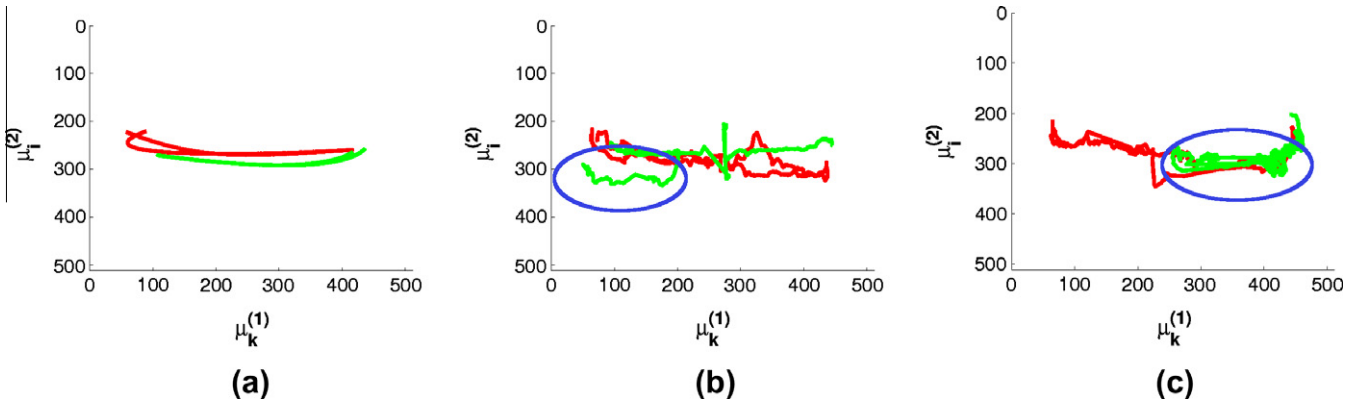

Fig. 9. Estimated trajectories for the dataset *Real4*. (a) Our method, (b) mean shift, (c) camshift. The green (printed in light gray in black and white) trajectory in (b) and (c) corresponds to the person in black moving from the right side of the image to the left and backwards. The ellipse highlights 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.

	Our approach (%)	Other approaches (%)				Our Rank	In tie
		Min (%)	Max (%)	Mean (%)	Median (%)		
Two motions	0.02	0.02	5.74	2.63	2.23	1/15	1
Three motions	0.98	0.58	27.02	9.53	8.00	2/15	–

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

	Our approach (%)	Other approaches (%)				Our Rank	In tie
		Min (%)	Max (%)	Mean (%)	Median (%)		
Two motions	0.00	0.00	1.55	0.51	0.21	1/15	7
Three motions	0.00	0.00	34.01	7.22	2.06	1/15	3

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

Problem	*Harris corners*			SIFT			SURF		
	#Trajectories	MSE	ACC (%)	#Trajectories	MSE	ACC (%)	#Trajectories	MSE	ACC (%)
Sim1	3820	69	100	973	130	100	6352	107	99
Sim2	1516	10	99	1146	138	98	3453	139	99
Sim3	2348	10	96	2296	172	98	7708	104	99
Sim4	2346	15	97	2319	122	98	7758	32	100
Sim5	1954	20	100	811	110	99	4848	130	100
Sim6	1351	29	100	646	142	99	3693	191	99
Sim7	1485	41	99	689	168	100	4244	155	99

which indicates a larger number of trajectories, provided for example by SURF, does not necessary 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 misdetection.

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 herein 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

[1] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Trans. Pattern Anal. Mach. Intell. 25 (5) (2003) 564–577.
[2] A. Yilmaz, O. Javed, M. Shah, Object tracking: a survey, ACM Comput. Surv. 38 (4) (2006) 1–45.
[3] E. Cuevas, D. Zaldivar, R. Rojas, Kalman Filter for Vision Tracking, Freier Universitat Berlin, Institut fur Informatik, Tech. Rep. B 05-12, 2005.
[4] M. Yang, Y. Wu, G. Hua, Context-aware visual tracking, IEEE Trans. Pattern Anal. Mach. Intell. 31 (7) (2009) 1195–1209.
[5] M. Isard, A. Blake, Condensation – conditional density propagation for visual tracking, Int. J. Comput. Vision 29 (1998) 5–28.
[6] J. Shi, C. Tomasi, Good features to track, in: 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR94), 1994, pp. 593–600.
[7] Z. Wang, X. Yang, Y. Xu, S. Yu, Camshift guided particle filter for visual tracking, Pattern Recognit. Lett. 30 (4) (2009) 407–413.
[8] H. Zhoo, Y. Yuan, C. Shi, Object tracking using SIFT features and mean shift, Comput. Vision Image Understand. 113 (3) (2009) 345–352.
[9] Q. Zhao, H. Tao, Differential earth mover's distance with its application to visual tracking, IEEE Trans. Pattern Anal. Mach. Intell. 32 (5) (2010) 274–287.
[10] V. Karavasilis, C. Nikou, A. Likas, Visual tracking using the earth mover's distance between Gaussian mixtures and Kalman filtering, Image Vision Comput. 29 (5) (2011) 295–305.
[11] A.A. Argyros, M.I.A. Lourakis, Real-time tracking of multiple skin-colored object with a possibly moving camera, in: Proceedings of the European Conference on Computer Vision (ECCV), 2004, pp. 368–379.
[12] D. Beymer, K. Konolige, Real-time tracking of multiple people using continuous detection, in: Proceedings of International Conference on Computer Vision (ICCV99), 1999.
[13] A. Bugeau, P. Perez, Track and cut: simultaneous tracking and segmentation of multiple objects with graph cuts, EURASIP J. Image Video Process. (2008). No. ID:317278.
[14] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Trans. Pattern Anal. Mach. Intell. 23 (11) (2001) 1222–1239.
[15] A. Mansouri, Region tracking via level set PDEs without motion computation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 947–961.
[16] D.P. Mukherjee, S. Member, N. Ray, S.T. Acton, S. Member, Level set analysis for leukocyte detection and tracking, IEEE Trans. Image Process. 13 (2004) 562–572.
[17] E. Horbert, K. Rematas, B. Leibe, Level-set person segmentation and tracking with multi-region appearance models and top-down shape information, in: Proceedings of International Conference on Computer Vision (ICCV11), 2011, pp. 1–8.
[18] C. Bibby, I. Reid, Real-time tracking of multiple occluding objects using level sets, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10), 2010, pp. 1307–1314.
[19] D.R. Magee, Tracking multiple vehicles using foreground, background and motion models, Image Vision Comput. 22 (2) (2004) 143–155.
[20] S. Gammeter, A. Ess, T. Jaeggli, K. Schindler, B. Leibe, L. van Gool, Articulated multibody tracking under egomotion, in: European Conference on Computer Vision (ECCV'08), LNCS, Springer, 2008, pp. 816–830.
[21] B. Han, S.-W. Joo, L.S. Davis, Probabilistic fusion tracking using mixture kernel-based Bayesian filtering, in: Proceedings of International Conference on Computer Vision (ICCV07), 2007, pp. 1–8.
[22] F. Moreno-Noguer, A. Sanfeliu, D. Samaras, Dependent multiple cue integration for robust tracking, IEEE Trans. Pattern Anal. Mach. Intell. 30 (4) (2008) 670–685.
[23] M. Black, P. Anandan, The robust estimation of multiple motions: parametric and piecewise-smooth flow fields, Comput. Vision Image Understand. 63 (1) (1996) 75–104.
[24] K.Y. Wong, M.E. Spetsakis, Motion segmentation by EM clustering of good features, in: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04), CVPRW '04, vol. 11, IEEE Computer Society, 2004, pp. 1–8.
[25] S.J. Pundlik, S.T. Birchfield, Real-time motion segmentation of sparse feature points at any speed, IEEE Trans. Syst., Man, Cyber. 38 (2008) 731–742.
[26] D. Buzan, S. Sclaroff, G. Kollios, Extraction and clustering of motion trajectories in video, in: International Conference on Pattern Recognition, 2004, pp. 521–524.
[27] J. Alon, S. Sclaroff, G. Kollios, V. Pavlovic, Discovering clusters in motion time-series data, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003, pp. 375–381.
[28] L. Zappella, X. Llado, E. Provenzi, J. Salvi, Enhanced local subspace affinity for feature-based motion segmentation, Pattern Recognit. 44 (2011) 454–470.
[29] M. Zeppelzauer, M. Zaharieva, D. Mitrovic, C. Breiteneder, A novel trajectory clustering approach for motion segmentation, Adv. Multimedia Model. 5916 (2010) 433–443.
[30] F. Lauer, C. Schnorr, Spectral clustering of linear subspaces for motion segmentation, in: Proceedings of the 12th IEEE International Conference on Computer Vision, ICCV'09, 2009.
[31] C. Julià, A. Sappa, F. Lumbreras, J. Serrat, A. López, Motion segmentation from feature trajectories with missing data, in: Proceedings of the 3rd Iberian Conference on Pattern Recognition and Image Analysis, Part I, IbPRIA '07, 2007, pp. 483–490.
[32] J. Yan, M. Pollefeys, A general framework for motion segmentation: independent, articulated, rigid, non-rigid, degenerate and non-degenerate, in: Proceedings of the European Conference on Computer Vision, ECCV'06, 2006.
[33] J.-H. Kim, L. Agapito, Motion segmentation using the Hadamard product and spectral clustering, in: Proceedings of the 2009 International Conference on Motion and Video Computing, WMVC'09, IEEE Computer Society, 2009, pp. 126–133.
[34] O. Masoud, N.P. Papanikolopoulos, A novel method for tracking and counting pedestrians in real-time using a single camera, IEEE Trans. Veh. Technol. 50 (5) (2001) 1267–1278.
[35] C. Harris, M. Stephens, A combined corner and edge detection, in: Proceedings of the Fourth Alvey Vision Conference, 1988, pp. 147–151.
[36] V. Karavasilis, K. Blekas, C. Nikou, Motion segmentation by model-based clustering of incomplete trajectories, in: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML'11-PKDD'11), Athens, Greece, 5–9 September, 2011.
[37] R. Tron, R. Vidal, A benchmark for the comparison of 3D motion segmentation algorithms, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07), vol. 1, 2007.
[38] B. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81), 1981, pp. 674–679.
[39] D.G. Lowe, Distinctive image features from scale-invariant keypoint, Int. J. Comput. Vision 60 (2) (2004) 91–110.
[40] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L.V. Gool, A comparison of affine region detectors, Int. J. Comput. Vision 65 (1/2) (2005) 43–72.
[41] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[42] S. Gaffney, Probabilistic Curve-aligned Clustering and Prediction with Regression Mixture Models, Ph.D. Dissertation, Department of Computer Science, University of California, Irvine, 2004.
[43] M. Tipping, Sparse bayesian learning and the relevance vector machine, J. Mach. Learn. Res. 1 (2001) 211–244.
[44] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Statist. Soc. B 39 (1977) 1–38.
[45] K. Blekas, C. Nikou, N. Galatsanos, N. Tsekos, A regression mixture model with spatial constraints for clustering spatiotemporal data, Int. J. Artif. Intell. Tools 17 (5) (2008) 1023–1041.
[46] G.R. Bradski, Computer vision face tracking for use in a perceptual user interface, Intel Technol. J. (Q2) (1998).
[47] E. Elhamifar, R. Vidal, Sparse subspace clustering, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09), 2009, pp. 2790–2797.

[48] Y. Sugaya, K. Kanatani, Multi-stage optimization for multi-body motion segmentation, IEICE Trans. Inform. Syst. E87-D (7) (2004) 1935–1942.
[49] A. Goh, R. Vidal, Segmenting motions of different types by unsupervised manifold clustering, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07), 2007.
[50] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions, in: British Machine Vision Conference (BMVC'02), vol. 1, 2002, pp. 384–393.
[51] H. Bay, T. Tuytelaars, L.V. Gool, SURF: speeded up robust features, in: Proceedings of the European Conference on Computer Vision (ECCV'06), 2006, pp. 404–417.
[52] V. Chandrasekhar, D. Chen, A. Lin, G. Takacs, S. Tsai, N.-M. Cheung, Y. Reznik, R. Grzeszczuk, B. Girod, Comparison of local feature descriptors for mobile visual search, IEEE Signal Process. Mag. 28 (4) (2011) 61–76.
[53] P. Viola, M. Jones, Robust real-time face detection, Int. J. Comput. Vision 57 (2004) 137–154.
[54] A. Oikonomopoulos, I. Patras, M. Pantic, N. Paragios, Trajectory-based representation of human actions, in: Proceedings of the ICMI 2006 and IJCAI 2007 International Conference on Artificial Intelligence for Human Computing, vol. 4451, 2007, pp. 133–154.