

On-line trajectory clustering for anomalous events detection

C. Piciarelli, G.L. Foresti

*Department of Mathematics and Computer Science, University of Udine
Via delle Scienze 206, 33100 Udine, ITALY
Tel. +39 0432 558423, Fax +39 0432 558499*

Abstract

In this paper we propose a trajectory clustering algorithm suited for video surveillance systems. Trajectories are clustered on-line, as the data are collected, and clusters are organised in a tree-like structure that, augmented with probability information, can be used to perform behaviour analysis, since it allows the identification of anomalous events.

Key words: trajectory clustering, on-line clustering, behaviour analysis

1 Introduction

Video surveillance systems generally have a very complex structure, since they must span different levels of abstraction, from the low-level detection of moving objects in video streams to the high-level behaviour analysis; it is not surprising that very few complete systems have been proposed so far (Collins et al., 2000; Haritaoglu et al., 2000). In particular, while there are countless works on the low-level image analysis part (such as change detection and object tracking), few have addressed the high level semantic interpretation of the scene (Ivanov and Bobick, 2000; Minnen et al., 2003). One of the possible reasons could be the lack of a proper link between the two processing levels, in which the raw data are collected and organized in a more abstract representation, useful for activity analysis. Even if some works show that it is possible to infer high-level analysis directly from the low-level data (Ivanov and Bobick, 2000) we still believe that a “middle level” could give some benefits. In this

Email address: {piccia,foresti}@dimi.uniud.it (C. Piciarelli, G.L. Foresti).

paper we propose one of the possible approaches in moving towards high-level analysis, based on trajectory clustering.

Trajectory clustering has been addressed in several works, for a survey see Liao (2005). Johnson and Hogg (1996) proposed an algorithm that statistically models the spatial distribution of trajectories using vector quantization. New trajectories are represented as sequences of vectors and are clustered using two competitive learning neural networks, one that finds the sequence of vectors that best represent a trajectory and the second to cluster those sequences. Stauffer and Grimson (2000) use again vector quantization, but the clusters are identified by a hierarchical analysis of the vector co-occurrences in the trajectories. Sclaroff, Kollios et al. proposed two different algorithms for track clustering, one based on a similarity measure called Longest Common Subsequence (LCSS) (Buzan et al., 2004) and one using a probabilistic approach based on Hidden Markov Models (Alon et al., 2003). HMM were used also by Porikli (2004), which find clusters using eigenvector analysis on the HMM parameter space. Lin et al. (2004) propose a hierarchical version of the k-means algorithm well suited for time series based on wavelet analysis. Makris and Ellis (2005) propose a method for modelling paths similar to the one proposed in this article, but their approach requires an initial off-line learning step, and only the classification step is performed on-line.

In this paper we extend our preliminary work (Piciarelli and Foresti, 2005) on on-line trajectory clustering. On-line clustering is about clustering computed as the incoming data are acquired, in opposition to off-line approaches like many of the works previously cited. Our main aim is to avoid the classical two-step clustering (data collection and off-line processing) since we want to use clustering information for on-line behaviour analysis. We also propose a tree-like structure to represent clusters that can be used for many purposes, for example detection of anomalies in the trajectories of moving objects.

2 Cluster representation, updating and matching

The proposed method groups trajectories in clusters in order to detect common patterns of activity. The trajectories are acquired by a multi-camera system, described in (Micheloni et al., 2003). The multi-camera approach allows a robust detection of trajectories, being less sensitive to occlusions and ambiguities than single-camera systems.

As previously stated, we require that the clustering algorithm performs on-line, as the data are acquired, without the need of two-step processing, in which data analysis is performed off-line after a phase of data acquisition. The clustering procedure should also be dynamic, in the sense that clusters must

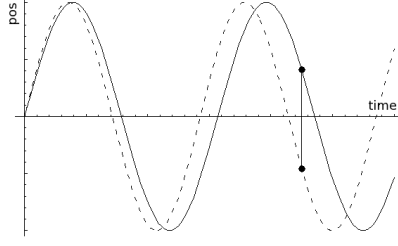


Fig. 1. The Euclidean distance between the positions of two objects at the same time can be great even if the trajectories are very similar each other, due to time shifts.

evolve through time, in order to match the changing patterns in the monitored environment. For example, if the system is used to monitor a parking lot, it is reasonable to assume that the common trajectories early in the morning, when people arrive in the parking lot, are much different from the ones detected in the evening, when people leave.

Every clustering algorithm must address three main problems:

- choose a suitable way to represent data;
- define a distance or similarity measure between trajectories and clusters;
- define a cluster updating method.

Many trajectory clustering systems deal with trajectory representation using data reduction techniques, most notably vector quantization (Johnson and Hogg, 1996; Stauffer and Grimson, 2000), component analysis (Biliotti et al., 2005; Hayashi et al., 2002) or DFT or wavelet coefficients (Lin et al., 2004). This step is mandatory in off-line systems, since the memorization of a large number of trajectories waiting for off-line processing would lead to an excessive amount of data. In the proposed on-line algorithm however only clusters and the currently developing trajectories need to be kept in memory, so we can afford a simple, yet very useful, explicit representation of the data. Each trajectory T_i is represented by a list of vectors t_{ij} representing the spatial position of the object along the x and y axes at time j :

$$T_i = \{t_{i1} \dots t_{in}\} \quad \text{where} \quad t_{ij} = \{x_{ij}, y_{ij}\} \quad (1)$$

Note that the position data also implicitly code temporal information, since the t_i vectors are acquired at fixed time intervals. Clusters are again represented as a list of vectors:

$$C_i = \{c_{i1} \dots c_{im}\} \quad \text{where} \quad c_{ij} = \{x_{ij}, y_{ij}, \sigma_{ij}^2\} \quad (2)$$

where σ_{ij}^2 is an approximation of the local variance of the cluster i at time j .

In order to check if a trajectory matches a given cluster, a distance or similarity measure must be defined. The usual Euclidean distance is not adequate,

since it performs poorly in presence of time shifts, as depicted in figure 1. Possible alternatives are Dynamic Time Warping (Salvador and Chan, 2004) or the Longest Common Subsequence (Buzan et al., 2004) but they have some drawbacks (computational complexity, inability to start the computation until the trajectory has come to end) so we propose a new distance measure. Given a trajectory $T = \{t_1 \dots t_n\}$ and a cluster $C = \{c_1 \dots c_m\}$ the distance measure adopted is defined as

$$D(T, C) = \frac{1}{n} \sum_{i=1}^n d(t_i, C) \quad (3)$$

where

$$d(t_i, C) = \min_j \left(\frac{\text{dist}(t_i, c_j)}{\sqrt{\sigma_j^2}} \right) \quad j \in \{ \lfloor (1 - \delta)i \rfloor \dots \lceil (1 + \delta)i \rceil \} \quad (4)$$

and $\text{dist}(t_i, c_j)$ is the usual Euclidean distance. The distance of a trajectory from a cluster is thus a mean of the normalized distances of every point of the trajectory from the nearest point of the cluster found inside a sliding temporal window centered in j . The temporal window has a variable, increasing size, in order to permit matching under accumulating temporal differences. The window is clipped in the range $(1 \dots m)$ and, if it completely falls outside this range, the distance D is set to ∞ . Note that the values $d(t_i, C)$ can be computed as the track develops, and can be used to detect if the track is moving away from the cluster.

Finally, when a trajectory matches a cluster, the cluster need to be updated. If $c_j = \{x, y, \sigma^2\}$ is the cluster element nearest to the trajectory element $t_i = \{\hat{x}, \hat{y}\}$, then c_j is updated as follows:

$$\begin{aligned} x &= (1 - \alpha)x + \alpha\hat{x} \\ y &= (1 - \alpha)y + \alpha\hat{y} \\ \sigma^2 &= (1 - \alpha)\sigma^2 + \alpha[\text{dist}(t_i, c_j)]^2 \end{aligned} \quad (5)$$

with $\alpha \in [0, 1]$ being an appropriate update rate, where higher values cause a faster fit of the clusters to the newly detected data. In this way each cluster is a dynamic approximation of the mean and variance of the trajectories that matched it, with an exponentially decreasing weight for the older trajectories.

3 Building trees of clusters

Since our final goal is to analyse behaviours from the trajectory data, we have structured the clustering information in such a way that can be useful for probabilistic reasoning about trajectories. In particular we want to detect and explicitly represent the shared prefixes of clusters, where a *prefix* is just

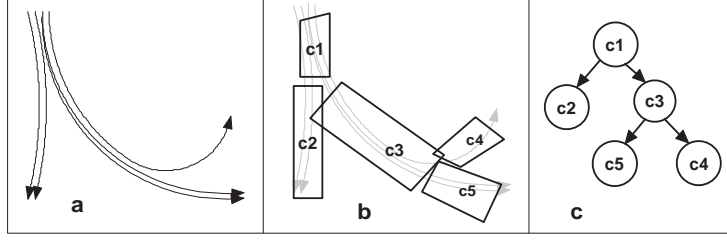


Fig. 2. Representing trajectories as a tree of clusters.

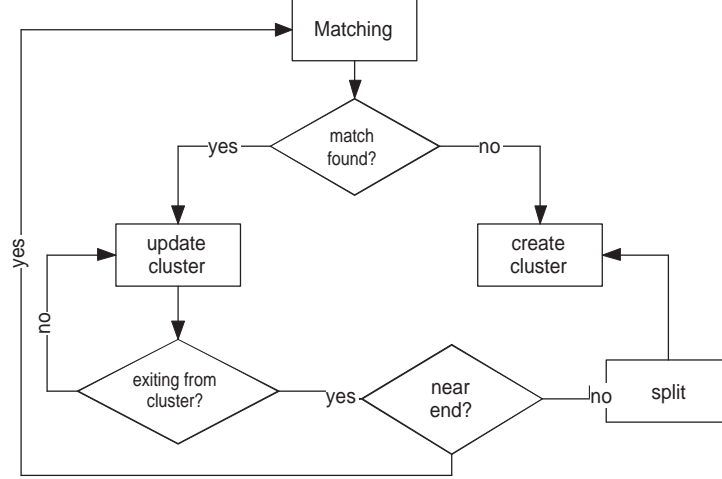


Fig. 3. The tree-building process.

the starting piece of a cluster. This model of shared prefixes allows to make predictions on the possible future movements of an object (and this is why we are not interested, for example, in shared *suffixes*). A group of trajectories is thus represented no more by a single cluster, but by groups of clusters organized in a tree-like structure, as shown in figure 2.

The whole process of tree creation is subdivided in two steps: tree building and tree maintenance. The tree building phase creates the trees of clusters and can be performed as the data are acquired, without the need to wait for the end of the trajectory. The main steps of the tree-building phase are summarized in the flow chart of figure 3.

When a trajectory is detected, its partial distances from all the root nodes of the trees is computed. If no clusters are sufficiently near to the trajectory, a new cluster is created; this implies the creation of a new tree consisting of a single node. If instead a match is found, the cluster starts to be immediately updated using equations 5. As the cluster is being updated, its distance $d(t_i, C)$ (see equation 4) is monitored in order to detect if the trajectory is moving away from the cluster. If this distance keeps growing for an excessive amount of time, we state that the trajectory is no more matching the cluster. This could happen for two reasons: the trajectory could be going beyond the end

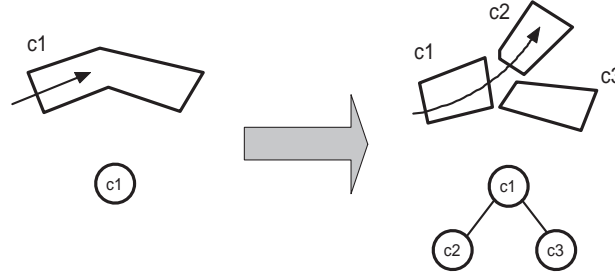


Fig. 4. The splitting procedure.

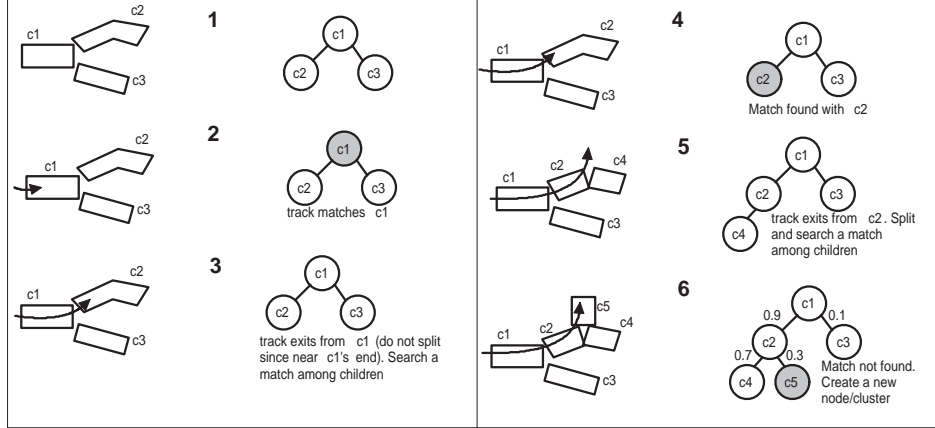


Fig. 5. 6-steps example representing the full match/update/split process.

of the cluster, or it could be exiting far from cluster's end. In the latter case a split is performed, consisting in the subdivision of the initial cluster in two subclusters, with the necessary tree adjustments as shown in figure 4.

When a trajectory exits from a cluster, a new matching step is performed. If a split was done, the trajectory automatically matches the newly created cluster, while in the other case the match is searched among all the children of the just-left cluster, and the whole process is iteratively repeated. An example of the match/update/split procedure is shown in figure 5.

When the system has fully detected a trajectory, the tree maintenance phase can be performed. This step is needed to increase the robustness of the process and guarantees the consistency of the trees. The first maintenance step is merging. For each tree, all the nodes at the same levels are compared in order to detect those clusters that are sufficiently near each other (the cluster-to-cluster distance is a simple variation of the trajectory-to-cluster distance defined in equation 3). All those clusters are merged together in a new, single cluster, which is a weighted mean of all the merged clusters and inherits all their children. The merge procedure is needed because clusters can become too near each another due to cluster updating, but it is also useful to make the process more robust. In fact one of the problems of the algorithm presented so far is that the cluster variance must be initialized to a fixed value, thus it

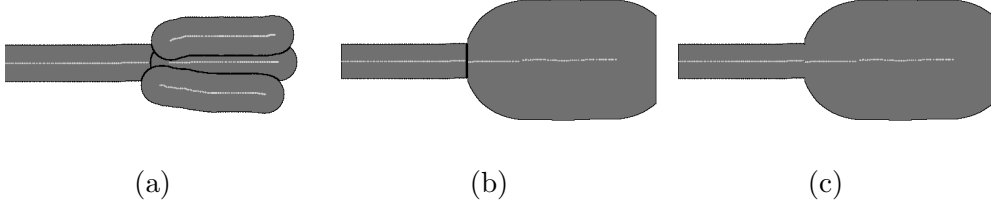


Fig. 6. The merge and concatenation procedures. (a) a tree of clusters; (b) the merge procedure applied to the leaf nodes; (c) concatenation of the two resulting clusters into one.

should be tuned depending on the kind of trajectories expected. If the variance is initialized with a too low value, the system cannot detect large clusters and it will model them as a separate set of clusters. This can be avoided if we allow the merge procedure to happen more easily when a large number of clusters is detected. To obtain this result, when clusters are compared to find merges, their variance is multiplied by a scaling factor linearly increasing with the number of clusters. Using this approach, in presence of many clusters their variance is increased and their distance (which is normalized using the variance, see equation 4) decreases, thus allowing the merge procedure to be used.

Two other maintenance procedures are concatenations and pruning. Concatenation is used when a node in the tree has a single child (this could happen after a merging step) and concatenates that cluster with its child, creating a single node from two. Pruning consists in cutting away the branches that have not been updated for a long time; this is useful in all those environments in which clusters dynamically change over time.

4 Behaviour analysis

The proposed algorithm was developed as a starting point for a behaviour identification system based on trajectory analysis. While we are currently working on explicit modelling of anomalous behaviours, we have already used the trajectory clustering algorithm to perform statistical anomaly detection. In this approach an anomaly is simply defined as an event which happens rarely. Of course an anomalous event is not necessarily a dangerous one, but we can assume that dangerous events are generally anomalous, thus justifying the use of this approach in surveillance systems.

Anomaly detection can be obtained if we associate to each cluster the number of times it was matched. This way the arcs in each tree (and the trees themselves) can be labelled with probabilistic information: given a node t , the arc to its child c_i will have the probability $\#c_i / \sum_j \#c_j$, where $\#c_i$ is the number

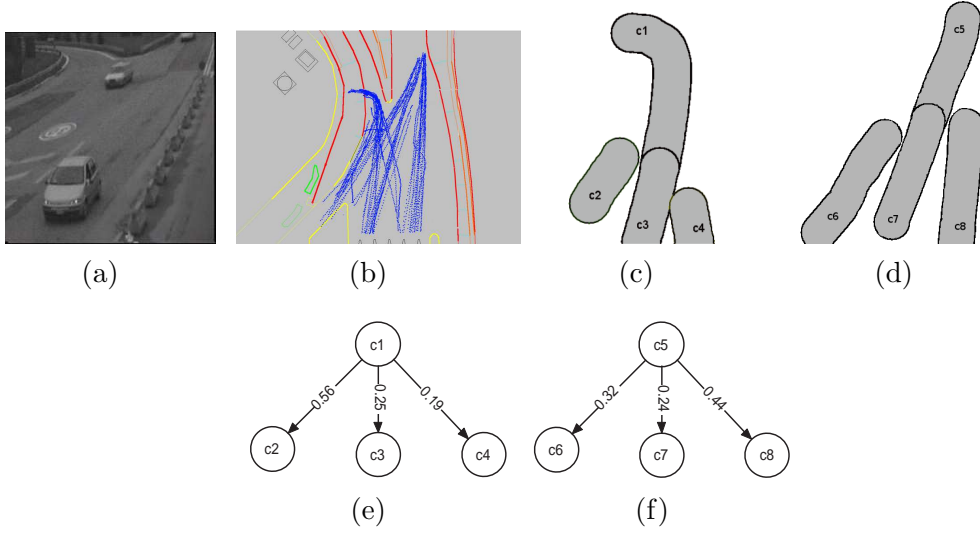


Fig. 7. Real world example of trajectory clustering. (a) Traffic on a highway; (b) trajectories acquired in the last 13 minutes; (c) (d) detected clusters for the cars coming from the upper-left and upper-right part of the image; (e) (f) probability tree showing relations between the clusters of figure (d) (e).

of times the cluster c_i was matched and the index j cycles over all the children of t (the same approach can be applied to entire trees, if we consider them as children of a single main root node). Using this approach, the probability of a path is just the product of the probabilities associated to every arc in the tree traversed by the path from the root node to the ending cluster. An anomalous trajectory can thus be defined simply as a trajectory that matches a path in the tree of clusters with low probability. Figure 7 shows an example of this probability-based approach applied to a real world environment, summarizing the traffic behaviour on a highway. Figure (a) shows the monitored environment, while figure (b) represents the 67 trajectories detected during the last 13 minutes. In figures (c) and (d) the detected clusters are shown, respectively for the cars coming from the upper-left and upper-right zone of the scene. Figures (e) and (f) finally show the two probability-labelled trees modelling the relations between clusters.

5 Experimental results

In this section we present two different tests to measure the clustering algorithm performances and an example of the system applied to real-world data. All the experiments were done using the values $\alpha = 0.05$, $\delta = 0.5$ (see equations 4 and 5). The initial variance is set to $\sigma^2 = 1225$ and a point is considered to match a cluster if it falls in the 2σ range from the nearest point of the cluster inside the temporal window.

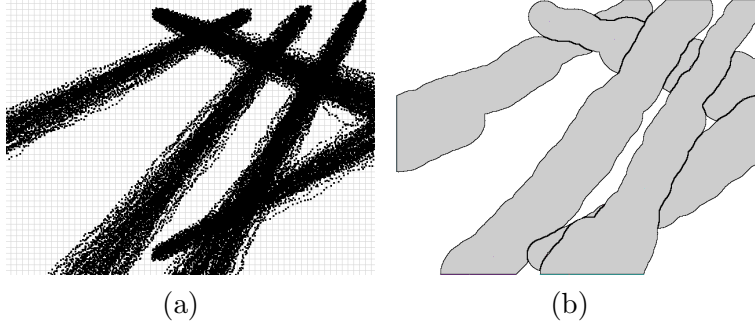


Fig. 8. Test sets and clustering results. (a) Trajectories from one of the 5-clusters data sets and (b) the clusters computed by the proposed algorithm.

The first test was aimed to measure the quality of the core clustering procedures, without taking into consideration the building of trees. For this purpose, groups of clusters without common prefixes have been automatically computer generated. We have considered the cases with 3, 4 and 5 clusters in the scene, and for each of them three different test sets were generated. Each test set contains the specified number of clusters, each one composed by 100 trajectories. Figure 8(a) shows for example the first test set with 5 clusters. On such test sets, the system should ideally detect the correct number of clusters represented by single-node trees; however, it may happen that a group of trajectories is represented by a multi-node tree because the merging procedure did not work. This surely is a clustering error, but we believe it should be considered less severe than the missed detection (or a false detection) of a full tree. The results are shown in table 1, in which we report both the number of single- and multi-node trees. In all the cases except one the correct number of trees was detected, even if three tests lead to the identification of multi-node trees.

In order to test the tree building and maintenance algorithm, we have taken into consideration six reference trees (which are shown in the first column of table 2) and, for each one of them, we asked three people to manually draw on a computer a set of 20 trajectories that could be modelled by the reference tree. The trajectories were then clustered with the proposed algorithm, and finally the reference and the computed trees were compared. A distance measure between reference and computed trees is calculated using the well-known tree edit distance (Bille, 2003), where each single editing operation (insertion, deletion) has an unitary cost. We have normalized the distance dividing it by the number of nodes in the reference tree. Table 2 shows the reference trees, the computed trees and their normalized tree distance. As can be seen, the merging procedure sometimes merges two clusters that are meant to be separated, as in test 5, subjects 1 and 3, and sometimes does not merge two clusters that should go together, as in test 4, subject 2, but the results are anyway encouraging.

	Single-node trees	multi-node trees	total trees
3 clusters			
set 1	3	0	3
set 2	3	0	3
set 3	3	0	3
4 clusters			
set 1	3	1	4
set 2	3	1	4
set 3	2	2	4
5 clusters			
set 1	5	0	5
set 2	4	0	4
set 3	5	0	5

Table 1
Experimental results for the cluster building algorithm.

























Reference tree	Subject 1	Subject 2	Subject 3
	 0.0	 0.0	 0.0
	 0.5	 0.75	 0.0
	 0.0	 0.0	 0.0
	 0.0	 0.28	 0.0
	 0.33	 0.0	 0.33
	 0.0	 0.12	 0.0

Table 2
Experimental results for the tree building algorithm.

Finally, figure 9 shows an example of the system running on real-world data, clustering car trajectories while monitoring a highway entrance. The original sequence is 27 minutes long, over which 108 trajectories were detected; some frames of the original sequence are shown in (a). Figures (b)–(f) depict the evolution of the trajectory model after 9, 30, 60, 90 and 108 trajectories. The image on the left represents the detected trajectories, while the one on the right shows the clusters, with arrows at the beginning of each tree and in the junctions, labelled with the probabilities described in section 4. Figure (b) was chosen because it is the first one in which a tree is composed of more than one node. The tree on the right has an high probability of being matched (0.86) and leads to two different children clusters, with probability 0.56 and 0.44. In figure 9(c) a new cluster appears, matching the cars coming from the top-right zone of the image, which is then extended, as can be seen in figure (d), and finally subdivided in two small branches at the end, representing the two main directions chosen by the cars while entering in the highway gates (figures (e) and (f)). Note that the clusters built after 30 trajectories already are a good general model for the activity detected in the scene, thus the on-line procedure can be considered reliable even after few processed trajectories.

6 Conclusions

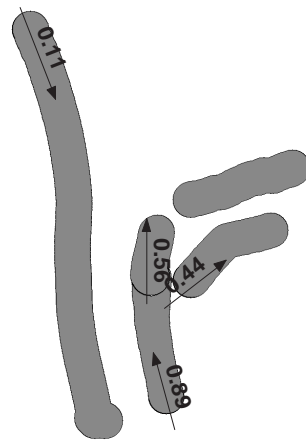
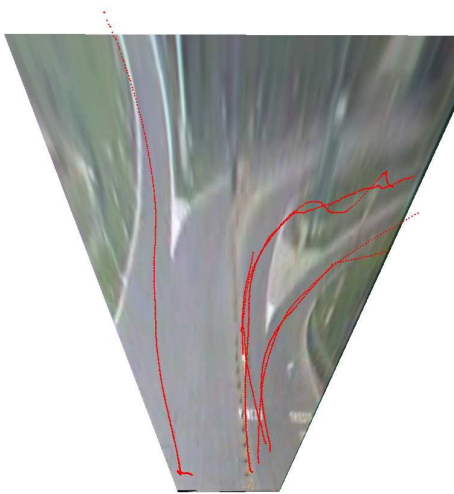
In this work we have presented an on-line trajectory clustering method. Avoiding the typical 2-step approach based on off-line data collection and analysis and on-line classification, the proposed method builds the clusters as the data are acquired by the tracking system. The clusters are organized in a tree-like structure which models the relations between clusters. The probabilistic data associated to the tree branches can be used to make predictions on the possible future developments of a trajectory or to calculate the global probability of a trajectory given the current model. We plan to use this data to move towards high-level scene understanding for surveillance purposes, since a trajectory probabilistic analysis can be useful to define and detect anomalous events that do not match the current most common behaviours.

References

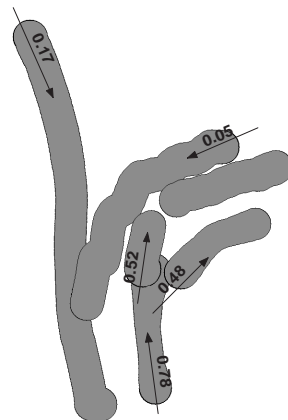
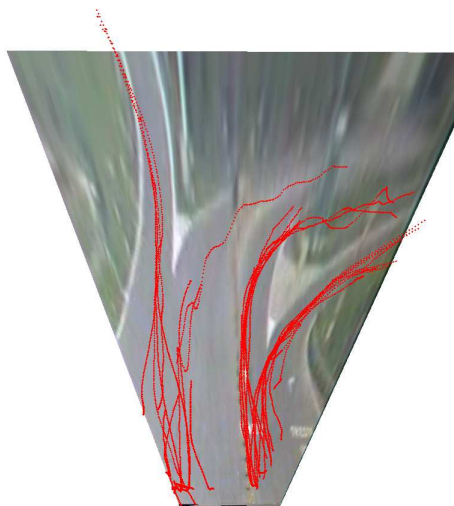
- Alon, J., Sclaroff, S., Kollios, G., Pavlovic, V., 2003. Discovering cluster in motion time-series data. In: Proc. Computer Vision and Pattern Recognition. Madison, WI, USA, pp. 375–381.
- Biliotti, D., Antonini, G., Thiran, J., 2005. Multi-layer hierarchical clustering of pedestrian trajectories for automatic counting of people in video se-



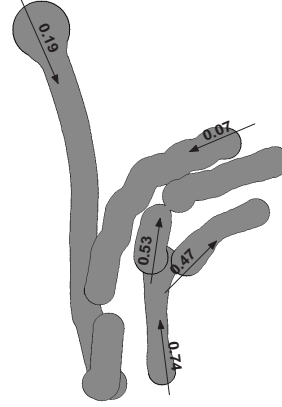
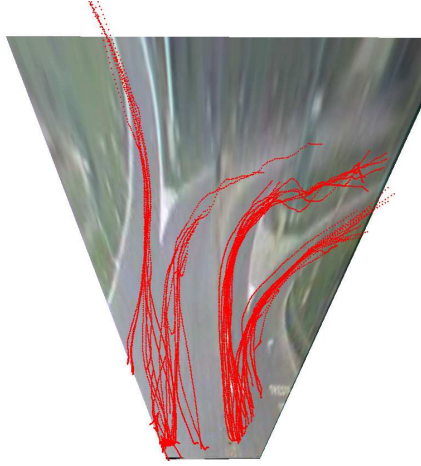
(a)



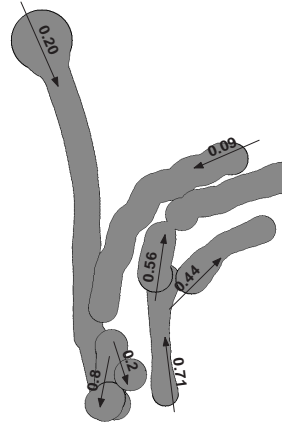
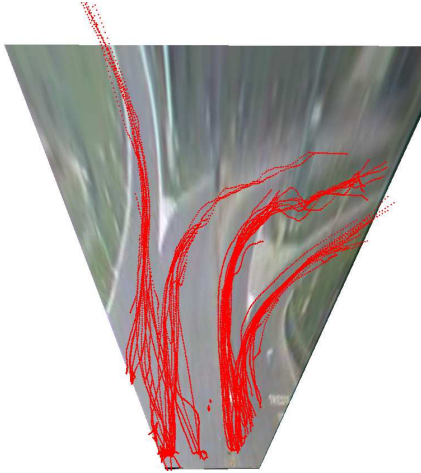
(b)



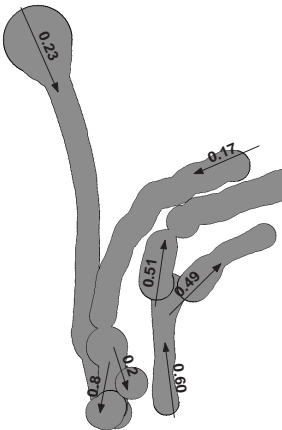
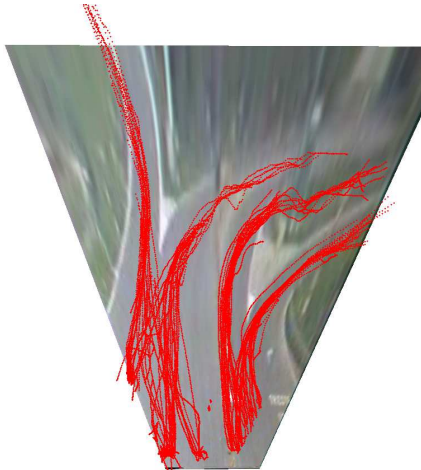
(c)



(d)



(e)



(f)

Fig. 9. Clustering the car trajectories on a highway. (a) The original sequence and the trajectories and clusters detected after (b) 9, (c) 30, (d) 60, (e) 90 and (f) 108 trajectories.

- quences. In: IEEE Workshop on Motion and Video Computing. Breckenridge, CO, USA, pp. II/50–57.
- Bille, P., 2003. Tree edit distance, alignment distance and inclusion. Tech. Rep. TR-2003-23, The IT University of Copenhagen.
- Buzan, D., Sclaroff, S., Kollios, G., 2004. Extraction and clustering of motion trajectories in video. In: Proc. International Conference on Pattern Recognition. Cambridge, UK, pp. 521–524.
- Collins, R., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., 2000. A system for video surveillance and monitoring. Tech. Rep. CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University.
- Haritaoglu, I., Harwood, D., Davis, L., 2000. w^4 : Real-time surveillance of people and their activities. IEEE Trans. on Pattern Analysis and Machine Intelligence 22 (8), 809–830.
- Hayashi, A., Nakasima, R., Kanbara, T., Suematsu, N., 2002. Multi-object motion pattern classification for visual surveillance and sports video retrieval. In: Proc. International Conference on Vision Interface. Calgary, Canada, pp. 101–108.
- Ivanov, Y., Bobick, A., 2000. Recognition of visual activities and interactions by stochastic parsing. IEEE Trans. on Pattern Analysis and Machine Intelligence 22 (8), 852–872.
- Johnson, N., Hogg, D., 1996. Learning the distribution of object trajectories for event recognition. Image and Vision Computing 14 (8), 609–615.
- Liao, T. W., 2005. Clustering of time series data – a survey. Pattern Recognition 38 (11), 1857–1874.
- Lin, J., Vlachos, M., Keogh, E., Gunopulos, D., 2004. Iterative incremental clustering of time series. In: Proc. Extending Database Technology. Crete, Greece, pp. 106–122.
- Makris, D., Ellis, T., 2005. Learning semantic scene models from observing activity in visual surveillance. IEEE Trans. on Systems, Man, and Cybernetics — Part B: Cybernetics 35 (3), 397–408.
- Micheloni, C., Foresti, G., Snidaro, L., 2003. A cooperative multicamera system for video-surveillance of parking lots. In: Proc. IEEE Intelligent Distributed Surveillance Systems. London, UK, pp. 5/1–5.
- Minnen, D., Essa, I., Starner, T., 2003. Expectation grammars: leveraging high-level expectations for activity recognition. In: Computer Vision and Pattern Recognition. pp. II–626–632.
- Piciarelli, C., Foresti, G., 2005. Toward event recognition using dynamic trajectory analysis and prediction. In: Proc. Imaging for Crime Detection and Prevention. London, UK, pp. 131–134.
- Porikli, F., 2004. Clustering variable length sequences by eigenvector decomposition using HMM. In: Proc. International Workshop on Structural and Syntactic Pattern Recognition. Lisbon, Portugal, pp. 352–360.
- Salvador, S., Chan, P., 2004. Fastdtw: Toward accurate dynamic time warping in linear time and space. In: KDD Workshop on Mining Temporal and

Sequential Data. Seattle, WA, USA, pp. 70–80.
Stauffer, C., Grimson, W., 2000. Learning patterns of activity using real-time tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 22 (8), 852–872.