Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition Letters 27 (2006) 1835–1842
Pattern Recognition Letters
www.elsevier.com/locate/patrec

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

Available online 21 April 2006

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 organized 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.

Keywords: 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 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

* Corresponding author. Tel.: +39 0432 558420; fax: +39 0432 558499.
E-mail addresses: piccia@dimi.uniud.it (C. Piciarelli), foresti@dimi.uniud.it (G.L. Foresti).

0167-8655/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2006.02.004

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 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:

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

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:

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

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

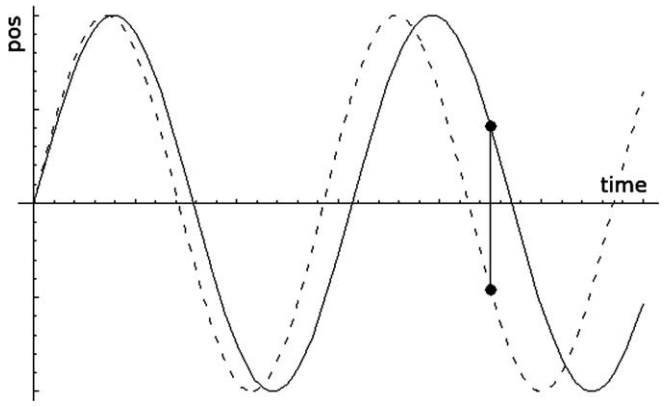

Fig. 1. The Euclidean distance between the positions of two objects at the same time can be great even if the trajectories are very similar each other, due to time shifts.

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 Fig. 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

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

where

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

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:

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