Trajectory-based anomalous event detection

Claudio Piciarelli, Student Member, IEEE, Christian Micheloni, Member, IEEE
and Gian Luca Foresti, Senior Member, IEEE

Abstract—During the last years the task of automatic event analysis in video sequences has gained an increasing attention among the research community. The application domains are disparate, ranging from video surveillance to automatic video annotation for sport videos or TV shots. Whatever the application field, most of the works in event analysis are based on two main approaches: the former based on explicit event recognition, focused on finding high-level, semantic interpretations of video sequences, the latter based on anomaly detection. This paper deals with the second approach, where the final goal is not the explicit labelling of recognized events, but the detection of anomalous events differing from typical patterns. In particular, the proposed work addresses anomaly detection by means of trajectory analysis, an approach with several application fields, most notably video surveillance and traffic monitoring. The proposed approach is based on single-class support vector machine clustering, where the novelty detection SVM capabilities are used for the identification of anomalous trajectories. Particular attention is given to trajectory classification in absence of a-priori information on the distribution of outliers. Experimental results prove the validity of the proposed approach.

Index Terms—Anomaly detection, event analysis, support vector machines, trajectory clustering.

I. INTRODUCTION

In the last years, several works have been proposed on event analysis. Partial surveys on this topic can be found in the reviews by Buxton [1] and Hu et al. [2] in the field of video surveillance. Even though many different approaches have been developed, they can generally be classified in two categories, often referred to with the terms explicit event recognition and anomaly detection.

In the explicit event recognition approach, the system has an explicit knowledge of the events that must be identified. Once an event is detected, it can be properly labelled with a semantic description. The fundamental part of an explicit recognition system is thus an a-priori knowledge base, where all the information about the recognisable events is stored, and the system behaves as a “parser” matching the incoming data with predefined templates found in the knowledge base. Explicit event recognition has its primary strength in its expressiveness, since it allows the modelling of heterogeneous events by labelling them with high-level semantic descriptions, thus giving a real interpretation of the analysed scene. The main drawback of this approach is the need of a prior modelling for each possible instance of any event of interest. In real-world scenarios with uncontrolled and uncooperative environments, this could imply an exponential growth of the number of models as the scene complexity increases. Because of the nature of the explicit event recognition task, it is not surprising that most works on this topic are based on stochastic parsers for the identification of known patterns of atomic events, as in the works by Ivanov and Bobick [3], Minnen et al. [4] or Moore and Essa [5]. Similar to a parser-based approach is the work of Vu, Brémond and Thomnat [6], even though in their case the language used to express complex events is not a full grammar, but rather a set of sub-events subject to temporal and logical constraints. Other works are based on the integration of prior knowledge and low level data acquired by video processing (change detection, object tracking etc.) as in the works by Ayers and Shah [7] or Medioni et al. [8]. Wada and Matsuyama proposed a work in which raw image data are directly used to infer behaviour recognition, without intermediate processing as object detection or tracking [9]. Other works use a-priori defined knowledge base partially modelled using machine learning techniques rather than being manually defined as in a stochastic grammar. Most of these works rely on the use of hidden Markov models for event recognition, as in the works by Oliver et al. [10], Galata et al. [11], Brand and Kettnaker [12], Bashir et al. [13]. Finally, other works are based on more general stochastic models, such as Bayesian networks, as in Hongeng et al. [14] or Moënne-Loccot et al. [15].

The other popular approach to event analysis is based on anomaly detection. In this case, the system does not require a prior knowledge base about the events that should be recognised. On the contrary, the event models are automatically built by the system itself in an unsupervised way, as the data are acquired. The models have a probabilistic nature, in the sense that the system is able to detect the most frequent patterns of activity occurred in the scene; the output of the event analysis process is no more an explicit description of the detected event (this would require a supervised learning), but only a measure of probability of that event, allowing to focus on the most rare ones (anomalies). A commonly used approach in this case is based on the clustering of the trajectories of the detected moving objects; the obtained clusters are then used as a normality model for anomaly detection. For a survey on trajectory (and more generally time series) clustering, see [16]. The work by Johnson and Hogg [17] was probably one of the first researches in this direction, using vector quantization for the compact representation of trajectories and multi-layer neural networks for the identification of common patterns. The same
authors later published another work on the same topic [18], where the probability density function of common patterns was approximated with a mixture of multivariate Gaussian densities. Another notable work on trajectory analysis has been done by Stauffer and Grimson [19], where vector-quantized trajectories are analysed in terms of co-occurrence matrices of vector prototypes. Makris and Ellis [20], [21] proposed a method for labelling the scene with topological information which is then used within a Bayesian approach to detect anomalous trajectories. Piciarelli and Foresti [22] proposed a trajectory clustering algorithm especially suited for on-line anomaly detection, which is a main issue in surveillance systems. Other relevant works on trajectory analysis for event detection include the ones by Porikli [23], which is based on eigenvalue decomposition of affinity matrices in order to find the correct number of clusters and compute conformity scores for anomaly detection; Hu et al. [24] use a hierarchical clustering of trajectories depending on spatial and temporal information; Lou et al. [25] use dynamic clustering techniques for traffic analysis; Morris and Hogg [26] analyse trajectories in order to detect interactions with known objects in the environment; Wang et al. [27] define an ad-hoc trajectory similarity measure and an associated clustering technique; Dee and Hogg [28] detect anomalies using inexplicability scores, a measure of how much a trajectory can be explained in terms of simple goals.

In this work, we address the problem of trajectory analysis for anomaly detection using support vector machines. Although support vector machines have been a widely used tool for classification and clustering problems and can explicitly address the problem of anomaly (or novelty) detection [29], only few trajectory clustering papers rely on SVM, such as [30]. In a preliminary work, we proposed an anomalous trajectory detection technique based on single-class SVM [31] and we faced the problem of choosing the best training parameters for optimal clustering. The proposed solution, however, had an high computational complexity; in this work we propose an alternative, faster technique for SVM training in presence of outliers. Moreover, we present further experimental results proving the validity of our approach.

The paper is structured as follows. In the next section, a short review of the support vector machine theory is given, in order to clarify some concepts that will be recalled later in the paper. Section III discusses the possible applications of support vector machine techniques to trajectory analysis, and some preliminary results are given. Section IV addresses the problem of tuning SVM in presence of an unknown number of outliers, which is a common scenario in practical applications. Finally, section V gives some experimental results proving the validity of the proposed approach.

II. SUPPORT VECTOR MACHINES THEORY

Support Vector Machines are a set of classification and regression techniques based on the concepts of statistical learning theory and risk minimization, initially proposed by Vapnik et al. [32] and later extended to the non-linear case with the introduction of kernel methods [33]. Single-class support vector machines [29] are a particular type of SVM well suited for anomaly detection tasks, since they consider the training data as elements drawn from a single probability distribution, whose support has to be found while possibly discarding outliers. This approach can naturally be applied to anomalous trajectory detection: the detected trajectories are initially transformed in fixed-dimension feature vectors; then the training data are clustered using a single-class feature vectors, thus detecting the hypervolume in the feature space containing all the normal trajectories. Identifying anomalous trajectories is just a matter of checking if a new trajectory falls outside the computed hypervolume. In order to better understand the remaining sections of this work, a brief description of the SVM theory is here given. For further details on support vector machines, see [34], [35].

The main idea at the basis of support vector machine theory arises from the application of statistical learning theory results to linear classifiers. Let us consider the case of two-class hyperplane classifiers in some dot product space $\mathcal{H}$:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0, \quad \mathbf{w}, \mathbf{x} \in \mathcal{H}, b \in \mathbb{R}$$

(1)

corresponding to the decision function

$$f(\mathbf{x}) = \text{sgn}((\mathbf{w} \cdot \mathbf{x}) + b)$$

as depicted in figure 1. Statistical learning theory states that the optimal classifier can be found by maximizing the distance between the two dashed hyperplanes in the direction of $\mathbf{w}$ (the margin).

![Fig. 1. Optimal classification hyperplane separating two classes. The optimal classification function is found by maximizing the distance between the two dashed hyperplanes in the direction of $\mathbf{w}$ (the margin).](image)

$$\min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w}||^2$$

subject to $y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \quad i = 1, \ldots, m$

where $m$ is the number of training data and $y_i \in \{-1, +1\}$ is the expected label for each element. The problem can be solved introducing the Lagrangian multipliers $\alpha_i \geq 0$ and setting to zero the partial derivatives of the Lagrangian
w.r.t. the primal variables w and b, thus leading to the dual optimization problem

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$  \hspace{1cm} (2)

subject to $\alpha_i \geq 0$, $i = 1, \ldots, m$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

which can be solved using standard quadratic optimization techniques. The decision function is now defined as

$$f(x) = \text{sgn} \left( \sum_{i=1}^{m} y_i \alpha_i (x \cdot x_i) + b \right)$$  \hspace{1cm} (3)

where $\alpha_i$ are solutions of (2) and $b$ can be found knowing that, for any x lying on the margin borders, the following equation holds:

$$y_i (w \cdot x_i + b) - 1 = 0$$

The points lying on the margin borders are called Support Vectors and they are associated to nonzero values of $\alpha_i$. This implies that the final solution (3) is sparse, being defined only in terms of a small subset of the data used in the training step.

The capability of solving only linear classification problems and the definition in a dot product space are the main drawbacks of this approach. Both the problems can be addressed by defining a map $\Phi : \mathcal{X} \rightarrow \mathcal{H}$ from the nonempty set of the original input data $\mathcal{X}$ to a dot product space $\mathcal{H}$ (feature space) where the problem has a linear solution. The fundamental observation is that, in the dual problem (2) and in the decision function (3), the only operations performed in $\mathcal{H}$ are dot products. The classifier can thus be trained without doing any explicit computation in $\mathcal{H}$ if an explicit formula for dot products in $\mathcal{H}$ is given. This formula is traditionally called kernel, defined as $k(x, x') = \Phi(x) \cdot \Phi(x')$. Using kernels, the decision function (3) becomes

$$f(x) = \text{sgn} \left( \sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b \right)$$  \hspace{1cm} (4)

where $\alpha$ is a solution of the optimization problem

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$  \hspace{1cm} (5)

subject to $\alpha_i \geq 0$, $i \in \{1 \ldots m\}$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

Similar techniques can also be applied to non-classification tasks such as anomaly detection. Given a set of unlabelled measures generated according to an unknown probability distribution $P$, single-class support vector machines are used to estimate the support of a particular quantile of $P$. The goal is to find an appropriate region in the feature space $\mathcal{X}$ containing most of the data drawn from $P$, possibly leaving outliers outside this region. In the SVM framework this can be obtained by searching for a decision hyperplane in the feature space $\mathcal{H}$ which maximizes its distance from the origin, while only a small fraction of data (the outliers) falls between the hyperplane and the origin. This can be expressed in terms of the constrained minimization problem

$$\min_{w,\xi,b} \frac{1}{2} \|w\|^2 + \frac{1}{\nu n} \sum_i \xi_i - b$$  \hspace{1cm} (6)

subject to $w \cdot \Phi(x_i) \geq b - \xi_i$, $\xi_i \geq 0$

where $x_i \in \mathcal{X}, i \in \{1 \ldots n\}$ are training data in the data space $\mathcal{X}$, $\Phi : \mathcal{X} \rightarrow \mathcal{H}$ is the function mapping vectors $x_i$ in the feature space $\mathcal{H}$ and $(w \cdot \Phi(x_i)) - b = 0$ is the decision hyperplane in $\mathcal{H}$. In the minimisation process, outliers are linearly penalized by the slack variables $\xi_i$, whose weight is controlled by the parameter $\nu \in (0,1]$. Introducing the Langrangian multipliers $\alpha_i$, the minimisation problem (6) can be reformulated in its dual form

$$\min_{\alpha} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)$$  \hspace{1cm} (7)

subject to $0 \leq \alpha_i \leq 1/(\nu n)$

$$\sum_i \alpha_i = 1$$

Values $\alpha_i$ can be found by solving the problem with standard quadratic programming methods; $w$ is given by

$$w = \sum_i \alpha_i \Phi(x_i)$$  \hspace{1cm} (8)

and, for any vector $\Phi(x_i)$ with $\alpha_i \neq 0$, the following equations hold

$$b - \xi_i = (w \cdot \Phi(x_i)) = \sum_j \alpha_j k(x_i, x_j)$$  \hspace{1cm} (9)

with $\xi_i > 0$ for outliers and $\xi_i = 0$ for support vectors lying on the decision plane. The decision function in the data space $\mathcal{X}$ is thus defined as

$$f(x) = \text{sign}( (w \cdot \Phi(x)) - b) = \text{sign}( \sum_i \alpha_i k(x, x_i) - b)$$  \hspace{1cm} (10)

The solution is again sparse since $\alpha_i = 0$ for any $x_i$ lying within the support region identified by the decision function: the majority of the training vectors do not contribute to the definition of the decision function. For the other vectors (support vectors), $0 < \alpha_i < 1/(\nu n)$ if the vector lies on the decision hyperplane, and $\alpha = 1/(\nu n)$ if the vector is an outlier.

The parameter $\nu$ has an intuitive meaning since it tunes the number of acceptable outliers. From (6) it can be seen that, when $\nu \rightarrow 0$, the outliers’ penalization factors grow to infinity, thus leading to a hard margin solution, where no outliers are allowed at all. On the other hand, if $\nu = 1$, the constraints in (7) allow a single solution in which all the $\alpha_i$ are saturated to the maximum value $1/n$, thus all the vectors are outliers. More specifically, it can be proven that $\nu$ is an upper bound for the fraction of outliers and a lower bound for the fraction of support vectors (e.g. with $\nu = 0.1$ at least 10% of the training vectors will be support vectors, and at most 10% will be outliers).
III. SVM TRAJECTORY CLUSTERING

In this section, we will explore the possibility of using support vector machines for event detection systems. The basic idea is to extract the trajectories of moving objects from video sequences, and cluster together groups of trajectories sharing similar features while leaving out the anomalous ones; testing a new trajectory for anomaly detection will thus be only a matter of comparing the new trajectory with the cluster model. Since a single class (the normal trajectories) is searched for and the aim is to find outliers for that class, single-class support vector machines seem to be well-suited for this problem, due to their ability to detect a region in the data space enclosing the training data while possibly excluding outliers.

First of all, a proper input space must be defined. Trajectories are generally represented by variable-length sequences of 2D coordinates, but the most commonly used kernels are defined on fixed-dimension spaces. A possible choice would be to define an ad-hoc kernel working on sequences, and some efforts have recently been done in this direction, e.g. with the definition of string kernels [36]; however in this work we have chosen to work with standard kernels. This implies that trajectories must be represented by fixed-dimension feature vectors, something we obtained by trajectory subsampling (all the experiments in this work have been performed by evenly subsampling trajectories into a list of 16 2D coordinates). Trajectories are also smoothed with a running-average filter before subsampling in order to remove the noise. Note that in this case we consider only spatial features, but other features could also be included, as long as the resulting feature vector has a fixed size. As an example, one of the last experimental results in section V includes temporal information.

Regarding the choice of an appropriate kernel, it has been proven that all the traditionally used kernels (Gaussian, polynomial, sigmoidal) have very similar performances given that the proper parameters are chosen [34]. Since we are working with spatial features, the Gaussian kernel seems a natural and intuitive choice, since it explicitly takes in consideration the Euclidean distance between feature vectors:

$$k(x_1, x_2) = \exp \left( -\frac{\|x_1 - x_2\|^2}{2\sigma^2} \right) \quad (x_1, x_2) \in \mathcal{X} \times \mathcal{X}$$

The choice of $\sigma$ heavily influences the final result since it represents the “scale factor” at which the data should be clustered. However, the scale factor is more related to the type of data (e.g. vehicle trajectories in a specific road, etc.) rather than on specific instances of data sets: $\sigma$ could be estimated only once for each given scenario, without the need of re-estimation when new data sets from the same scenario need to be processed. The choice of the SVM parameter $\nu$ is more critical, since it depends on the specific data set and thus cannot be estimated a-priori. In the next section, we propose a method to avoid an explicit choice of $\nu$ by automatically identifying the outliers in the training data.

A. Preliminary tests

We will now present some preliminary experimental results on trajectory clustering with support vector machines. The first test is based on labelled data sets, where training trajectories are grouped in several known clusters. Multi-class support vector machines are trained using some training sets, and their generalization capabilities are evaluated on the corresponding test sets. Being a supervised learning problem, this classification approach cannot be directly applied to anomaly detection, and it has mainly been performed to confirm the validity of the choices described in the section about trajectory representation and choice of kernel. If the test succeeds in separating different trajectories in different clusters, this implies that linear separability of the data in the feature space $\mathcal{H}$ is possible. This is not obvious, since $\mathcal{H}$ depends both on the initial feature space and on the kernel used; a success in this test does not imply good results in the final proposed method, but a failure would be a strong indicator that the choices described in the previous section could not lead to good results, whatever SVM technique is used. Tests have been done with increasing number of classes, ranging from a minimum of 2 to a maximum of 10. For each case, 50 different training and test sets have been randomly generated, each one with 100 trajectories in each class, for a total of 900 data sets. Each training set has been used for the training phase of a multi-class SVM with a Gaussian kernel ($\sigma = 0.5$), whose performances have been measured on the corresponding test set. Results are shown in table I, reporting the total number of misclassifications in each run of 50 tests. The good results (with a worst case of 0.04% of classification errors) confirm that the combination of a Gaussian kernel with subsampling-based spatial trajectory representation can be a good choice for trajectory analysis with support vector machines.

<table>
<thead>
<tr>
<th># clust.</th>
<th>Classification errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0/10000 (0%)</td>
</tr>
<tr>
<td>3</td>
<td>0/15000 (0%)</td>
</tr>
<tr>
<td>4</td>
<td>0/20000 (0%)</td>
</tr>
<tr>
<td>5</td>
<td>0/25000 (0%)</td>
</tr>
<tr>
<td>6</td>
<td>0/30000 (0.02%)</td>
</tr>
<tr>
<td>7</td>
<td>0/35000 (0%)</td>
</tr>
<tr>
<td>8</td>
<td>0/40000 (0.02%)</td>
</tr>
<tr>
<td>9</td>
<td>0/45000 (0.04%)</td>
</tr>
<tr>
<td>10</td>
<td>0/50000 (0%)</td>
</tr>
</tbody>
</table>

TABLE I. EXPERIMENTAL RESULTS WITH SUPERVISED SVM CLASSIFICATION.
positives (normal trajectories misdetected as anomalies). The number of false positives is stable in all the tests, while the true positives slowly decrease with the increasing number of clusters, since it becomes more and more difficult to detect a potentially anomalous trajectory among many different types of normal ones. In any case, detections results seem promising, especially with few clusters in the scene.

<table>
<thead>
<tr>
<th># clust.</th>
<th>Classification results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True positives</td>
</tr>
<tr>
<td>1</td>
<td>497/500 (99.4%)</td>
</tr>
<tr>
<td>2</td>
<td>487/500 (97.4%)</td>
</tr>
<tr>
<td>3</td>
<td>467/500 (93.4%)</td>
</tr>
<tr>
<td>4</td>
<td>457/500 (91.4%)</td>
</tr>
<tr>
<td>5</td>
<td>452/500 (90.4%)</td>
</tr>
<tr>
<td>6</td>
<td>447/500 (89.4%)</td>
</tr>
<tr>
<td>7</td>
<td>446/500 (89.2%)</td>
</tr>
<tr>
<td>8</td>
<td>443/500 (88.6%)</td>
</tr>
<tr>
<td>9</td>
<td>428/500 (85.6%)</td>
</tr>
<tr>
<td>10</td>
<td>437/500 (87.4%)</td>
</tr>
</tbody>
</table>

TABLE II

EXPERIMENTAL RESULTS WITH UNSUPERVISED SVM CLUSTERING.

IV. OUTLIER DETECTION

The major problem in trajectory clustering with single-class support vector machines lies in the choice of the parameter $\nu$. This parameter is directly linked to the number of outliers in the training data, since it is an upper bound for the fraction of outliers and a lower bound for the fraction of support vectors. For example, if it is known that 10% of the training data will be outliers, this would imply that $\nu$ must necessarily be greater than 0.1. However, in practical trajectory clustering applications, there generally is no prior knowledge on the amount of outliers, and thus no bounds can be given.

Since an arbitrary choice of $\nu$ would lead to sub-optimal results, this parameter should be properly estimated. There is some literature on the estimation of support vector machine parameters [37], but these works are generally based on cross-validation or similar techniques that search the parameter space for a configuration that minimises the generalization error. This approach can be used in supervised learning problems, where labelled training data are available for validation, but cannot be directly applied to unsupervised, single-class clustering techniques. In this section, we describe an approach to single-class support vector machine tuning not relying on labelled data; note that labelled data will actually be used for performance evaluation, but they are not involved in the algorithm itself. The proposed method will entirely avoid the problem of estimating a proper value for $\nu$ by automatically detecting outliers before applying SVM clustering techniques.

In a preliminary work we already proposed a technique for optimal $\nu$ estimation [31], but it is affected by strong computational requirements since, for each classification problem, $n$ support vector machines must be trained, with $n$ proportionally increasing with the required precision of the system. In this paper, we propose an alternative and more efficient approach requiring only a single SVM training by avoiding an explicit computation of $\nu$. The main idea is that the outliers are by definition few in number and with an underlying distribution different from the normal data. We thus hypothesize that outliers can be detected by observing the shrink of the hypervolume enclosing all the normal data as we change the training conditions, for example by varying the value of $\nu$ ranging from 0 to 1 (as we did in our previous work) or directly by removing from the training set the support vectors defining the boundaries of the classification function (the approach used in this paper). More precisely, given a support region initially forced to include all the data, its hypervolume in the feature space would shrink rapidly when removing real outliers, while it would shrink slowly when removing normal data, as shown in the 2D toy example of figure 2. The detection of the point of change in shrinking speed can thus lead to an optimal support region discarding only real outliers. The support region volume however cannot be easily computed in the data space $X$, and thus we are forced to perform the computations in the feature space $H$. Recall that $H$ could be extremely complex or even unknown, but there is an easy way to compute dot products in $H$ by means of kernels.

From now on, we will suppose to work with a normalised kernel, this is a kernel $k$ such that $k(x,x) = 1, x \in X$. Note that the Gaussian kernel chosen in the previous section is always normalised, since

$$k_{Gaussian}(x,x) = \exp \left( -\frac{||x-x||^2}{2\sigma^2} \right) = \exp(0) = 1$$

Moreover, any other kernel $k'(x_1,x_2) = \Phi(x_1) \cdot \Phi(x_2)$ has a normalised version $k$ defined as

$$k(x_1,x_2) = \frac{\Phi(x_1) \cdot \Phi(x_2)}{||\Phi(x_1)|| \cdot ||\Phi(x_2)||} = \frac{\Phi(x_1) \cdot \Phi(x_2)}{\sqrt{\Phi(x_1) \cdot \Phi(x_1)} \cdot \sqrt{\Phi(x_2) \cdot \Phi(x_2)}} = \frac{k'(x_1,x_2)}{\sqrt{k'(x_1,x_1) \cdot k'(x_2,x_2)}}$$

The use of a normalised kernel influences the distribution of the data projected in $H$, since

$$||\Phi(x)||^2 = \Phi(x) \cdot \Phi(x) = k(x,x) = 1, \ x \in X$$

and thus all the feature vectors in $H$ lie on the surface of an hypersphere with unitary radius, and the decision hyperplane cuts the hypersphere so that all the non-outlier data lie on the resulting hyperspherical cap (figure 3). With normalized kernels it is easy to compute the angle $\theta$ between two feature vectors $\Phi(x_1), \Phi(x_2), x_1 \in X$ lying on the hypersphere, since

$$\Phi(x_1) \cdot \Phi(x_2) = ||\Phi(x_1)|| ||\Phi(x_2)|| \cos(\theta) = \cos(\theta)$$

$$\theta = \arccos(\Phi(x_1) \cdot \Phi(x_2)) = \arccos(k(x_1,x_2))$$

(11)

(as a side note, observe that, if $k(x_1,x_2) \geq 0$ such as in the case of the Gaussian kernel, then $\cos(\theta) > 0$ for any $\Phi(x_1), \Phi(x_2)$, and thus all the data lie in the same orthant).

Let us now consider the centre $c$ of the hyperspherical cap containing the non-outlier data. Since vector $w$ is perpendicular to the decision hyperplane (see equation 1), the centre $c$ is defined as

$$c = \frac{w}{||w||}$$
Using (11) it is possible to compute the angle between \( c \) and a generic outlier \( x_{out} \) (see figure 3), defined as

\[
\theta_{out} = \arccos(c \cdot \Phi(x_{out}))
\]

\[
= \arccos\left(\frac{w \cdot \Phi(x_{out})}{\|w\|}\right)
\]

Using (9), this leads to

\[
\theta_{out} = \arccos\left(\frac{b - \xi_{out}}{\|w\|}\right)
\]

Using (8) and (9) and knowing the values \( \alpha_i \), obtained as a solution of (7), \( \theta_{out} \) can thus be defined as

\[
\theta_{out} = \arccos\left(\frac{\sum \alpha_i k(x_{out}, x_i)}{\sqrt{\sum \alpha_i \xi_{out}} \cdot \sqrt{\sum \alpha_i \xi_{out}}}ight)
\]

(12)

We now observe that, in the degenerate case of \( \nu = 1 \), the constraints in (7) lead to the solution \( \alpha_i = 1/n \), and thus all the data are considered outliers and the decision function (10) reduces to a Parzen window estimate of the underlying distribution. In this case, from (8) it follows that \( w/\|w\| \) points to the mean of the feature vectors in \( \mathcal{H} \); this can be intuitively seen in figure 2, where the support region converges toward the data centre as \( \nu \) approaches to 1.

![Figure 2. Support regions found by training a single-class support vector machine with different values of \( \nu \). The largest region is obtained with \( \nu = 10^{-7} \); the regions shrink as \( \nu \) approaches to 1.](image)

The proposed outlier detection technique can now be defined. Given a set of training data, a single-class support vector machine is ideally trained using \( \nu = 1 \) (we do not really need to compute the solution, since we already know it will be \( \alpha_i = 1/n \)); then, for each training vector \( x_i \), the angle \( \theta_i \) between \( \Phi(x_i) \) and the data centre \( c \) is calculated using (12). Note that (12) is defined only for outliers, but in this case it can be applied to any data vector, since any vector is by definition an outlier when \( \nu = 1 \). The considerations made at the beginning of this section on support region hypervolumes can now be applied to the values of \( \theta \): we can assume that the majority of the data will have similar, small values of \( \theta \) since, if the classification works correctly, they are enclosed in the same hyperspherical cap of the non-outlier data, while few elements will have large values of \( \theta \), being outliers falling out of the searched distribution. We identify these elements by plotting the \( \theta \) values sorted in ascending order (see figure 5): the plot will have a high-curvature “elbow” separating the normal data, characterized by a slow growth in the values of \( \theta \), from the outliers, characterized by a final quick growth of the plot. The elbow can be identified by searching for the point of the plot with the largest distance from a line connecting the two extrema of the plot itself. The elbow point has an associated \( \theta_{th} \) angle (see again figure 5) that can be used as a threshold in order to separate normal data from anomalies: outliers in the training set can be detected by finding the elements \( x_i \) such that \( \theta_i > \theta_{th} \).

At this point there is no need to explicitly find a good \( \nu \) value for outlier removal, since we have already detected which data should not be considered during the support vector machine training step. A single-class support vector machine can thus be trained with a near-to-zero value for \( \nu \) on the training data set where the supposed outliers are removed; the small \( \nu \) will enforce the decision function to enclose all the data previously classified as non-outliers. The final classification function will not be influenced by the presence of outliers, since they have been explicitly detected and removed from the training set.

The process can thus be summarized as follows: first compute the angle \( \theta \) for each element in the training data set, then sort the obtained values and remove the outliers automatically detected by searching for the elbow point in the plot of \( \theta \). Finally, train an hard-margin single-class support vector machine on the remaining data. A single SVM training is needed, thus greatly reducing the computational complexity if compared to our preliminary work [31]. The overall computational time depends on the time required for the training of the support vector machine, which is \( O(m^3) \), where \( m \) is the number of elements in the training set, if no particular optimizations are used and can be improved up to \( O(m) \) with proper approximation techniques [38]. Computing the \( \theta \) values can be done in \( O(m^2) \), while sorting them and finding the bend points is \( O(m \log m) \). Once the training step is completed, checking a new trajectory involves a standard SVM test, which is a very fast operation since it is linear in
the number $n_{sv}$ of support vectors, and typically $n_{sv} \ll m$.

V. Experimental results

In order to test the validity of the proposed approach, experimental results on both synthetic and real-world data sets are here given\(^1\). Before focusing on automatic outlier discovery, we want to check if the angle $\theta$, as defined in eq. 12, is a good measure for outlier detection. In order to perform this test, we compared the proposed method with another simple yet very effective outlier detection technique, based on the concept of discords [39], [40]: a discord is defined as the trajectory maximizing its Euclidean distance from the nearest neighbour in the data space. The discord distance $d$ has the same interpretation of $\theta$ in our work: the higher the value, the more probable is for a trajectory to be an outlier. We randomly generated 1000 datasets, each one containing 260 trajectories: 250 trajectories grouped in 5 main clusters, plus 10 outliers. For each dataset we computed both $\theta$ and $d$, and we checked if the outliers really fall within the 10 trajectories with highest $\theta$ or $d$ respectively. Experimental results show that the performances of the two approaches are comparable, with a 3.70% error obtained with the proposed method and 2.96% error using discords. The use of discords has its main advantages in computational efficiency (in [39] a linear algorithm for discord discovery is proposed) and in the lack of tuning parameters, while the results of our method proved to be dependant on a good choice of $\sigma$ ($\sigma = 0.5$ has been used in the previous test). On the other hand, our method has the advantage of generalization, since the analysis of previously unseen data drawn from the same probability distribution of the training data can be done without training again the system. Moreover, the major advantage of the proposed method relies in its ability of automatically detecting the number of outliers, while discord-based methods as proposed in [39] can only detect the $k$ most anomalous trajectories, without giving any hint on a good choice of $k$.

1) Synthetic data: In the rest of this section, we will test the automatic outlier detection capabilities of the proposed approach. In order to compare the proposed approach with a standard single-class support vector machine with fixed $\nu$ training, 50 experimental cases were considered by randomly generating training sets with 100 normal trajectories grouped in two clusters, in which anomalous trajectories were added, ranging from 1 to 10 anomalies; results have been averaged over 10 runs for each test case. For each training set, two SVM were trained, one using the proposed method and one with $\nu = 0.2$. Both the support vector machines use a Gaussian kernel with $\sigma = 2$. Results are shown in figure 4 and are measured in terms of anomalies and normal data correctly identified in the training sets. Note that no test sets are used here, since we are not measuring the generalization capabilities of the proposed method, but only its ability to remove outliers from the training data sets. As it can be seen, the fixed $\nu$ approach initially has a large number of false positives (normal trajectories classified as anomalous) since $\nu = 0.2$ is an overestimated choice for a small amount of outliers. If we increase the number of real outliers, we face the opposite problem, with low true positives rates. On the other hand, the proposed approach always has a good true positives detection rate (more than 90%) with a negligible false positive rate.

Previous results show how the proposed method can handle outliers in the training data better than the naive approach, however they do not prove that the system has good generalization performances when classifying previously unseen patterns. We thus considered 100 different experimental cases, each one with a different number of groups of 100 normal trajectories (ranging from 1 to 10) and outliers (from 1 to 10). For each test, 10 different training/test sets were created, for a total of 2000 data sets. Single-class support vector machines have been trained using the proposed method on the training sets and the results were measured on the test sets (for an example, see figure 5). Note that outliers in the test sets are drawn from the same distribution used for outliers in the corresponding training sets, in order to ease the detection of anomalous training patterns being included in the normality class. Tables III and IV respectively show the generalization results of the proposed approach in terms of false positives and true positives. As it can be seen, the true positives detection always gives good results, with an average of 92% of correct detections. Both the true positives and false positives results do not seem to be noticeably influenced by the increasing number of outliers, while there is a slight performance degradation in true positive detection when increasing the number of groups of normal trajectories.

Since the last experiment did not show any appreciable variation in the results with low numbers of outliers (from 1 to 10 outliers in each training/test set), another test has been performed in order to measure the robustness of the proposed method in presence of large amounts of outliers. We considered the case of three clusters of normal data, each one containing 100 trajectories, where an increasing number of outliers has been added, ranging from 1 to 300 (in the last case the data sets thus contain 50% outliers). Figure 6 shows the plot of the true positives detected by the proposed approach; as it can be seen, the system has a high true positives detection rate even in presence of large amounts of outliers. The maximum number of detected false positives was 2.

2) Real-world data: The previously described experimental results were obtained on synthetic data in order to ease the evaluation of the algorithm performances. We now conclude this section with some examples of applications to real-world data, even though in this case there is no easy way to collect ground truth information and thus it is not possible to give objective performance measurements. We considered the case of an urban road, monitored by a static colour camera (figure 8(a)). The acquired video data consist in a 51-minutes-long sequence at 320 × 240 resolution, during which 1430 vehicles were detected by means of multi-gaussian background-subtraction techniques as in [19] and tracked using a combination of Kalman-based and CamShift trackers. We simulated the presence of two forbidden U-turns obtained by merging the trajectories of vehicles moving in opposite directions. Moreover, the trackers are not error-free, and thus

\(^1\)The datasets used in this section are available at http://avires.dimi.uniud.it/papers/trclust
Fig. 4. Comparative results for trajectory clustering with increasing amounts of outliers in the training set. (a) true positives (correctly detected outliers), and (b) false positives (normal trajectories misclassified as outliers) with standard single-class SVM, $\nu = 0.2$ and with the proposed method.

![Graph](image)

**Fig. 5.** Anomalous trajectory detection. (a) Training set, 300 normal trajectories are grouped in three main clusters and are perturbed with 10 outliers; (b) test set, with 10 outliers correctly detected; (c) sorted values of $\theta$.

![Graph](image)

<table>
<thead>
<tr>
<th>Anomalies</th>
<th>Clusters in the training set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3.10%</td>
</tr>
<tr>
<td>2</td>
<td>1.70%</td>
</tr>
<tr>
<td>3</td>
<td>2.70%</td>
</tr>
<tr>
<td>4</td>
<td>2.30%</td>
</tr>
<tr>
<td>5</td>
<td>3.00%</td>
</tr>
<tr>
<td>6</td>
<td>3.30%</td>
</tr>
<tr>
<td>7</td>
<td>1.90%</td>
</tr>
<tr>
<td>8</td>
<td>1.00%</td>
</tr>
<tr>
<td>9</td>
<td>2.70%</td>
</tr>
<tr>
<td>10</td>
<td>2.70%</td>
</tr>
</tbody>
</table>

**Table III**

| False positives results for the proposed method, expressed in terms of normal trajectories misclassified as anomalies over the total amount of normal trajectories in each data set. Results have been averaged over 10 runs.

The acquired data are affected by noise, especially in the form of broken trajectories due to lost objects. The acquired trajectories have been split in a train and a test set, each one containing 715 trajectories, and the proposed method has been applied using $\sigma = 3$. Since the trajectory shapes are quite simple, we have chosen to subsample each trajectory using 8 points, thus leading to feature vectors composed of 16 elements (the $x$ and $y$ coordinates for each subsampled point). The results on the test set are shown in figures (b) and (c). As it can be seen, the system has correctly recognized as anomalies the two U-turn trajectories (bold lines); moreover, the broken trajectories generated by errors in the tracking system are marked as anomalies as well, since they significantly differ in shape from the normal data.

In order to test the system on other publicly available datasets, we also applied the proposed outlier detection algorithm to the trajectory datasets used in [41] and [42]. The dataset from [41] consists in 160 trajectories of moving persons acquired in an indoor environment. 152 trajectories are grouped in 4 main clusters, the remaining 8 are outliers. Figure 7(a) shows the results obtained by our system: all the 8 anomalous trajectories have been correctly detected, although also two false positives are present. The dataset used in [42] is made up of trajectories extracted from infrared surveillance.
In this paper we proposed a technique for anomalous event detection by means of trajectory analysis. The trajectories are subsampled to a fixed-dimension vector representation and clustered with a single-class support vector machine. The presence of outliers in the training data has led us to a novel technique for automatic detection of the outliers based on geometric considerations in the SVM feature space. The experimental results have proven the validity of the proposed approach.

VI. CONCLUSIONS

In this paper we proposed a technique for anomalous event detection by means of trajectory analysis. The trajectories are subsampled to a fixed-dimension vector representation and clustered with a single-class support vector machine. The presence of outliers in the training data has led us to a novel technique for automatic detection of the outliers based on geometric considerations in the SVM feature space. The experimental results have proven the validity of the proposed approach.

REFERENCES


<table>
<thead>
<tr>
<th>Anomalies</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>true positives</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>50</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>100</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>150</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>200</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>250</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>300</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

TRUE POSITIVES RESULTS FOR THE PROPOSED METHOD, EXPRESSED IN TERMS OF ANOMALOUS TRAJECTORIES CORRECTLY CLASSIFIED AS OUTLIERS OVER THE TOTAL AMOUNT OF ANOMALOUS TRAJECTORIES IN EACH DATA SET. RESULTS HAVE BEEN AVERAGED OVER 10 RUNS.

![Fig. 6. True positives results with high amounts of outliers both in the training and in the test set and with 300 normal trajectories grouped in three clusters.](image-url)
Fig. 7. Test results on publicly available datasets, the black thick trajectories are outliers. (a) dataset from [41]; (b) dataset from [42].

Fig. 8. The proposed method applied to real-world data. (a) top-view of an urban road; (b) normal trajectories and (c) anomalous trajectories detected in the test set (bold lines: the U-turn trajectories added to the test set).


[40] E. Keogh, J. Lin, and A. Fu, “Hot SAX: efficiently finding the most unusual time series subsequence,” in IEEE Int. Conf. on Data Mining, Houston, TX, USA, 2005, pp. 226–233.
