Anomalous trajectory detection using support vector machines

Article · September 2007

DOI: 10.1109/AVSS.2007.4425302 · Source: IEEE Xplore

CITATIONS
13

READS
110

2 authors:

Claudio Piciarelli
University of Udine
53 PUBLICATIONS **905** CITATIONS
SEE PROFILE

G.L. Foresti
University of Udine
323 PUBLICATIONS **4,267** CITATIONS
SEE PROFILE

ANOMALOUS TRAJECTORY DETECTION USING SUPPORT VECTOR MACHINES

C. Piciarelli, G. L. Foresti

Department of Mathematics and Computer Science
University of Udine
Udine, Italy

Abstract

One of the most promising approaches to event analysis in video sequences is based on the automatic modelling of common patterns of activity for later detection of anomalous events. This approach is especially useful in those applications that do not necessarily require the exact identification of the events, but need only the detection of anomalies that should be reported to a human operator (e.g. video surveillance or traffic monitoring applications). In this paper we propose a trajectory analysis method based on Support Vector Machines; the SVM model is trained on a given set of trajectories and can subsequently detect trajectories substantially differing from the training ones. Particular emphasis is placed on a novel method for estimating the parameter ν, since it heavily influences the performances of the system but cannot be easily estimated a-priori. Experimental results are given both on synthetic and real-world data.

1. Introduction

In the last years, computer vision researchers started to address the topic of high-level event detection and semantic understanding of video sequences. Starting from the low-level processing methods developed up to now, the new aim is to move toward a more abstract interpretation of the scene, enabling the identification of actions, events, activities. These tasks can be achieved by explicit, template-based modelling of the activities to be recognized and some interesting works have been proposed in this direction (e.g. [21, 6]), however this approach has some inherent limits, especially regarding the ability of being able to recognize only activities that were defined a-priori in the knowledge base. In many practical applications of event analysis (e.g. video surveillance, traffic monitoring etc.), the detection of the exact event is not a critical issue, while it is more useful the identification of anomalous behaviours that should be notified to human operators. The loss in semantic expressiveness (we do not know what happened, but only that something anomalous happened) is balanced by the greater flexibility, due to the absence of a-priori models. In this approach, a model of common events is automatically built from training data (and possibly updated online), and the detection is performed by comparing new events with the model, in order to evaluate their degree of anomaly.

Trajectories have proven to be good features for this kind of analysis, thus leading to several works on trajectory clustering for anomaly detection [9]. A work on modelling the spatial distribution of trajectories using vector quantization was first proposed by Johnson and Hogg [7]; later VQ has been used also by Stauffer and Grimson [19] combined with hierarchical clustering. Many works are based on the use of techniques already popular in other time-series analysis problems such as Hidden Markov Models: Porikli uses eigenvector analysis in the HMM feature space [14]; other works using HMM are the ones proposed by Alon et al.[1] or by Saunier and Sayed [15]. Another common approach is to represent trajectories in a fixed-size feature space where standard clustering algorithms can be applied: Hu et al.[5] resample trajectories to a fixed length and group them using a hierarchical spatial/temporal fuzzy k-means clustering; Lin et al.[10] use hierarchical k-means on trajectories' wavelet coefficients. Other works were focused on clustering techniques directly applied to trajectory data without the need of a fixed-length representation, most notably by Makris and Ellis [11] and Piciarelli and Foresti [13].

In this paper we follow the fixed-size representation approach to explore the possibility of trajectory clustering for anomaly detection using single-class support vector machines. SVMs have in fact proven to be a good tool for data clustering with solid mathematical foundations, we thus believe that they can successfully be used also in trajectory analysis. In the next sections we briefly recall some basic points on single-class SVM and discuss how to apply this technique to trajectories. We also propose a novel technique for the estimation of the parameter ν and conclude with experimental results on both synthetic and real-wolrd data.

978-1-4244-1696-7/07/$25.00

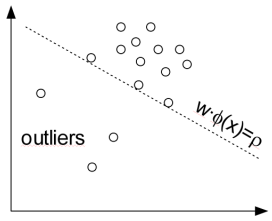

Figure 1: *Linear separation of the outliers in feature space.*

2. SVM for anomaly detection

The main idea at the basis of Support Vector Machines was first proposed by Vapnik et al. [20], where the theories on risk minimization are used to solve linear classification problems with margin maximization. However it's only with the introduction of kernels that SVMs gained greater popularity [2], thanks to their ability to solve non-linear problems. After that, many new works based on the SVM idea were proposed, as ν-SVM [17] and single-class SVM [16]; for further details see [12, 18].

Assuming some basic familiarity with SVM concepts, in this section we will focus on a particular kind of support vector machine that will be used in the rest of this paper, the single-class ν-SVM, which is a natural choice for novelty detection applications. Given a set of unlabelled measures generated according to an unknown distribution P, single-class ν-SVM are used to estimate the support of a particular quantile of P. The goal is thus to find an appropriate region in the input 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 $\mathcal{F}$ with maximized distance from the origin, with the constraint that only a fraction of the data should fall in the region between the plane and the origin, as shown in figure 1.

Following SVM theory, this approach can be formulated as a quadratic minimization problem

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi},\rho} \ & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{\nu n}\sum_i \xi_i - \rho \\ \text{subject to } \ & \boldsymbol{w}\cdot\Phi(\boldsymbol{x}_i) \geq \rho - \xi_i, \quad \xi_i \geq 0 \end{aligned} \tag{1}$$

where $\boldsymbol{x}_i \in \mathcal{X}, i \in [1 \dots n]$ are n training data in the input space $\mathcal{X}$, $\Phi : \mathcal{X} \to \mathcal{F}$ is the function mapping vectors $\boldsymbol{x}_i$ in the feature space $\mathcal{F}$ and $((\boldsymbol{w}\cdot\Phi(\boldsymbol{x})) - \rho)$ is the decision hyperplane in $\mathcal{F}$. In the minimization process, outliers are linearly penalized by the slack variables ξ_i, whose weight is controlled by the parameter $\nu \in (0, 1]$. Introducing the Langrangian multipliers α_i the problem can be formulated in its dual form

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \ & \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ \text{subject to } \ & 0 \leq \alpha_i \leq 1/(\nu n) \\ & \sum_i \alpha_i = 1 \end{aligned} \tag{2}$$

where $k(\boldsymbol{x}_i,\boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j)$ is a kernel function. Values α_i can be found solving the problem with standard quadratic programming methods; $\boldsymbol{w}$ is given by

$$\boldsymbol{w} = \sum_i \alpha_i \Phi(\boldsymbol{x}_i) \tag{3}$$

and ρ is found considering that, for any vector $\Phi(\boldsymbol{x})$ lying on the decision hyperplane (this is, a support vector), the following equations hold

$$\rho = (\boldsymbol{w}\cdot\Phi(\boldsymbol{x})) = \sum_i \alpha_i k(\boldsymbol{x},\boldsymbol{x_i}) \tag{4}$$

The decision function in the input space $\mathcal{X}$ is thus defined as

$$\begin{aligned} f(\boldsymbol{x}) &= \operatorname{sign}((\boldsymbol{w}\cdot\Phi(\boldsymbol{x})) - \rho) \\ &= \operatorname{sign}\Big(\sum_i \alpha_i k(\boldsymbol{x},\boldsymbol{x_i}) - \rho\Big) \end{aligned} \tag{5}$$

The solution is sparse since $\alpha_i = 0$ for any $\boldsymbol{x}_i$ lying within the support region identified by the decision function: the majority of the training vectors thus do not contribute to the definition of the decision function. For the other vectors (called *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 ν has an intuitive meaning since it tunes the number of acceptable outliers. From equation 1 it can be seen that, when $v \to 0$, 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 of equation 2 allow a single solution in which all the α are saturated to the maximum value $1/n$, thus all the vectors are outliers. More specifically, it can be proven that ν 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).

3. Anomalous trajectory detection

Trajectories have always been considered good features for anomaly detection, especially in scenarios in which the moving objects generally follow few pre-defined paths (e.g.

traffic surveillance). We thus want to build a model of common trajectories by detecting a cluster grouping the majority of trajectories sharing common features while leaving out the anomalous ones. The single-class ν-SVM discussed above seems to be a suitable tool to create this model, thanks to its ability to detect a region in the input space enclosing the training data while possibly excluding outliers. However, two main problems must be faced, namely i) data representation, and ii) parameter tuning.

Finding a suitable way to represent trajectories is a critical step since SVMs cannot be easily applied to raw trajectories. The main problem is that commonly used kernels require a fixed-size input space, while trajectories are generally represented by a variable-length sequence of 2D coordinates $(x_1, y_1), \ldots, (x_m, y_m)$. Two solutions are possible: to develop a specialized kernel for trajectories or use standard ones on fixed length vectors of trajectory features. While we plan to investigate the first approach in the future, in this work we will use a fixed-length representation. Note that "fixed length" here is referred to the size of the feature space in which trajectories are projected for SVM training and classification, and not to the size of the original trajectories; trajectories can thus have different lengths. This representation should take into account two requirements:

- adequate expressive power, to avoid loss of important information about trajectories.
- robustness to noise, since the raw data acquired by sensors (e.g. video-based tracking systems) are often noisy;

Since we do not need any particular invariant about the trajectories (e.g. for translation, total length, etc.) we have chosen to directly use the trajectory coordinates as feature data, thus each trajectory is evenly subsampled to a 32-coordinates list that is directly used as a feature vector. This choice of course limit our expressive power since, for any fixed number n, there exists a complex enough trajectory that cannot be properly approximated by n samples, but this number has proved to be suitable for all of our test sets, and of course it can anyway be tuned to fit specific requirements. Robustness to noise is another factor to consider, since naive subsampling can extract noisy data. To avoid this problem trajectories are first smoothed with a running-average filter whose size is set to 5% the total length of the trajectory. Finally, in order to normalize the data, we center and rescale each feature vector by subtracting the mean and dividing by the standard deviation of the whole training set.

Parameter tuning is the second aspect to consider. First of all a proper kernel must be chosen. Since our feature vectors represent spatial coordinates, a Gaussian kernel seems to be a natural and intuitive choice, because it is directly

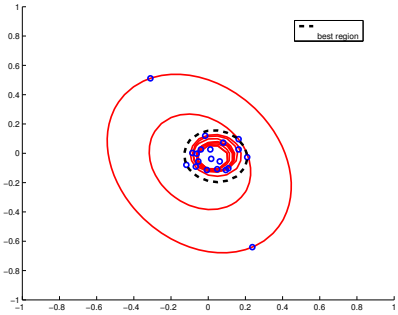

Figure 2: *Support regions with different values of ν.*

based on the euclidean distance between vectors:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\Big(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}{2\sigma^2}\Big) \qquad \forall (\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathcal{X} \times \mathcal{X} \tag{6}$$

The choice of σ heavily influences the final results since it represents a "scale factor" at which the data should be clustered, but it is mostly dependent on the nature of data. The parameter can be empirically estimated, but once found the best one for a given scenario, it does not need to be changed. The choice of ν is more critical, since it cannot be estimated a-priori (this would imply to know the outliers ratio in the data) and can change through time in online systems. The next section faces the problem of ν estimation.

4. ν estimation

As discussed in the previous sections, the parameter $\nu \in (0, 1]$ is related to the number of outliers we want to leave outside the support region; in particular, ν is an upper bound for the fraction of outliers and a lower bound for the fraction of support vectors. However in typical applications in which trajectory analysis is needed, there is no a-priori knowledge on the expected number of outliers, thus ν must be estimated. Moreover, if the training is performed on-line as the data are acquired (incremental SVMs exist for this case [8, 3]) there is no guarantee that this parameter will stay fixed, since the outlier ratio could change, thus requiring a periodic modification of the ν parameter.

Our method for ν estimation is based on the simple consideration that outliers are by definition few and distributed far from the group of "normal" data. If we change the ν parameter from a near-to-zero value (no outliers, all the data lie in the support region) to progressively increasing values, the support region *area* should have a noticeable shrink when outliers are left out of the support region while should decrease slowly when normal data start to be classified as outliers, as in the 2D example of figure 2. Unfortunately we cannot easily compute the support region area in the input space $\mathcal{X}$, but some considerations can be done in the feature

space $\mathcal{F}$. If the kernel is normalized[1], this is $k(x,x) = 1$, then $\|\Phi(\boldsymbol{x})\|^2 = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}) = k(x,x) = 1, x \in \mathcal{X}$. This means that all the feature vectors lie on the surface of an hypersphere with radius 1 in the feature space $\mathcal{F}$, and the decision hyperplane is cutting the hypersphere such that the majority of the data lies on the resulting hyperspherical cap. Inspired by the work of Desobry et al.[4] we also note that, given two vectors $\Phi(\boldsymbol{x}_1)$ and $\Phi(\boldsymbol{x}_2)$, $\boldsymbol{x}_i \in \mathcal{X}$, it is possible to calculate the angle θ subtending the arc $\overset{\frown}{\Phi(\boldsymbol{x}_1)\Phi(\boldsymbol{x}_2)}$ since

$$\Phi(\boldsymbol{x}_1) \cdot \Phi(\boldsymbol{x}_2) = \|\Phi(\boldsymbol{x}_1)\|\|\Phi(\boldsymbol{x}_2)\| \cos(\theta) = \cos(\theta) \quad (7)$$

$$\theta = \arccos(\Phi(\boldsymbol{x}_1) \cdot \Phi(\boldsymbol{x}_2)) = \arccos(k(\boldsymbol{x}_1, \boldsymbol{x}_2)) \quad (8)$$

We now want to compute the angle θ between the centre $\boldsymbol{c} = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ of the spherical cap and a generic point $\Phi(\boldsymbol{p})$ lying on the intersection of the hypersphere and the hyperplane (this is a non-saturated support vector), as shown in figure 3. Since $\boldsymbol{p}$ is a support vector, we know that $\boldsymbol{w} \cdot \Phi(\boldsymbol{p}) = \rho$ and thus

$$\begin{aligned} \theta &= \arccos(c \cdot \Phi(\boldsymbol{x})) \\ &= \arccos\Big(\frac{\boldsymbol{w} \cdot \Phi(\boldsymbol{p})}{\|\boldsymbol{w}\|}\Big) = \arccos\Big(\frac{\rho}{\|\boldsymbol{w}\|}\Big) \end{aligned} \quad (9)$$

where ρ can be computed as shown in section 2 and, from equation 3, $\|\boldsymbol{w}\|$ is given by

$$\|\boldsymbol{w}\|^2 = \boldsymbol{w} \cdot \boldsymbol{w} = \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \quad (10)$$

$\boldsymbol{K}$ being the kernel matrix such that $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Similar processes lead us to the equation for the angle γ between the centres $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ of two different SVMs:

$$\begin{aligned} \gamma &= \arccos\Big(\frac{\boldsymbol{w}_1 \cdot \boldsymbol{w}_2}{\|\boldsymbol{w}_1\|\|\boldsymbol{w}_2\|}\Big) \\ &= \frac{\boldsymbol{\alpha}_1^T \boldsymbol{K} \boldsymbol{\alpha}_2}{\sqrt{\boldsymbol{\alpha}_1^T \boldsymbol{K} \boldsymbol{\alpha}_1}\sqrt{\boldsymbol{\alpha}_2^T \boldsymbol{K} \boldsymbol{\alpha}_2}} \end{aligned} \quad (11)$$

It is now possible to compute the *overlap angle* β, that is the angle subtending the intersection area of the two spherical caps in the direction $\boldsymbol{c}_1 - \boldsymbol{c}_2$:

$$\beta = \min(\theta_1, \gamma + \theta_2) - \max(-\theta_1, \gamma - \theta_2) \quad (12)$$

Our initial hypothesis on the variations of the support region's area can now be reformulated in terms of overlap angles. We train several SVMs on the same data by varying ν in the range $(0, 1]$; we also record the overlap angle β_ν between the current SVM and the first one (corresponding to the hard-margin case) every time the number of outliers increases. Figure 4 shows the recorded overlap angles for the data shown in figure 2; it is possible to note how the overlap angle decreases quickly at the beginning, when the first outliers are discarded, while decreases slowly when "normal" data start to be classified as outliers. We can thus define the best ν as the one corresponding to the point A in figure 4, chosen as the farthest point from the line merging the first and last elements in the plot. Figure 2 shows the best support region detected with this method. Note that the procedure works only if at least one outlier exists; if the training data set does not contain any outliers the plot decreases uniformly. This situation can be easily detected by checking that the measured distance is below a given threshold; in this case we can safely assign to ν the smallest possible value.

[1] It's always possible to obtain a normalized version k' of a kernel k: $k'(x_1, x_2) = k(x_1, x_2)/\sqrt{k(x_1, x_1)k(x_2, x_2)}$

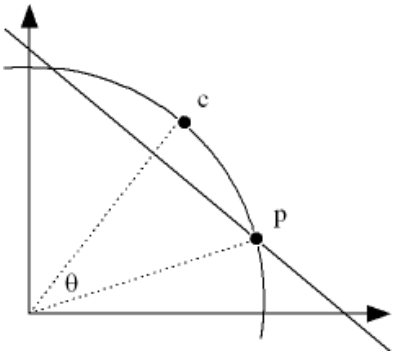

Figure 3: *Angles in the feature space $\mathcal{F}$.*

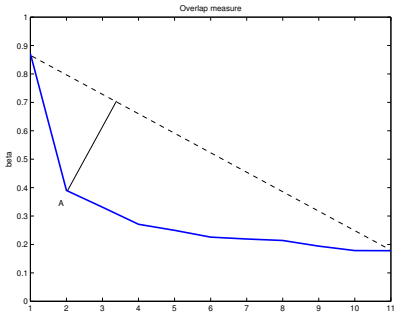

Figure 4: *Overlap angles with increasing values of ν.*

5. Experimental results

To evaluate the performances of the proposed algorithm we have tested it both on synthetic and real-world data. The first experiment with synthetic data is oriented to check if the SVM clustering can effectively detect anomalous trajectories, without taking in consideration the problem of ν tuning. A set of 30 trajectories shown in figure 5(a) has been hand-drawn and used as training set for a SVM with gaussian kernel, $\sigma = 5$. Since the training set does not con-

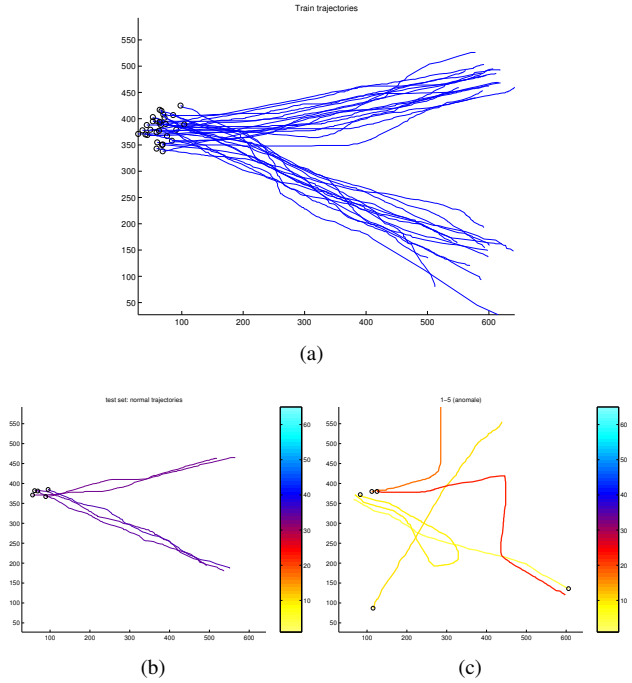

Figure 5: *Test with synthetic data: training set (a), normal (b) and anomalous (c) test set.*

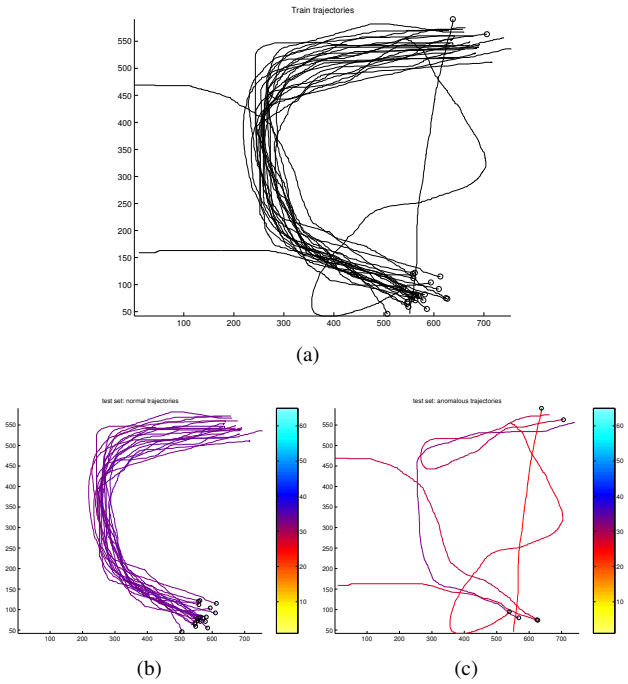

Figure 6: *Test with synthetic data: training set (a), normal (b) and anomalous (c) elements of the training set.*

tain outliers, we have fixed ν at a very small value, thus forcing the algorithm to consider all the trajectories as non-outlier vectors. A set of 10 trajectories, 5 normal and 5 anomalous, have been then classified using the SVM. Results in figure (b) and (c) show that the SVM has correctly classified the two groups.

In order to test the ν estimation algorithm, we trained a SVM on a train set of 25 synthetic trajectories, 5 of them being outliers, and we used the same set also for training. We used again a gaussian kernel with $\sigma = 5$. As a result, the best ν is 0.30, which effectively separates well the normal from the anomalous trajectories (see figure 6).

We also tested the proposed method with real-world data. We used a vehicle trajectories data set[2], composed of 1211 trajectories; 500 trajectories were used as training set (see figure 7). The optimal ν was 0.28. Figures 7(b) 7(c) show the 20 most anomalous trajectories in the test set: the results seem reasonable since the vast majority of the trajectories have a north-south orientation (or vice-versa), thus the few east-west trajectories are considered anomalies if compared with the model built from observations.

6. Conclusions

In this paper we have proposed an anomalous trajectory detection method based on support vector machines; single-

[2]dataset available at `http://ngsim.camsys.com/`

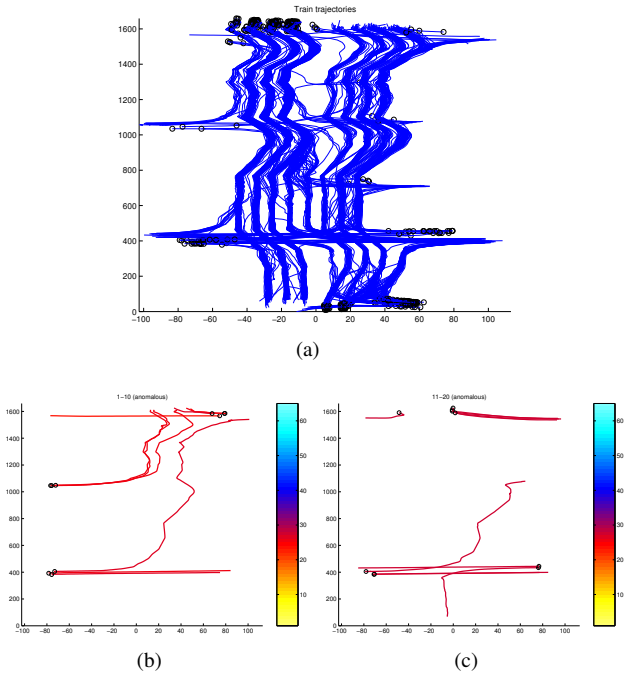

Figure 7: *Test with real-world data: training set (a), and the 20 most anomalous trajectories in the test set (b) (c).*

class ν-SVM are used to build a normality model used to detect anomalies. We also proposed a novel method for estimating the best value for parameter ν. As a future work we are planning to check if the performances can be further increased by using a kernel specially tuned for trajectories.

Acknowledgements

This work was partially supported by the Italian Ministry of University and Scientific Research within the framework of the project "Ambient Intelligence: event analysis, sensor reconfiguration and multimodal interfaces"(2007-2008) and by the European FP6 Project HAMLeT "Hazardous Material Localisation & Person Tracking" (SEC6-SA-204400)

References

[1] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering cluster in motion time-series data. In *Computer Vision and Pattern Recognition*, pages 375–381, Madison, WI, USA, 2003.

[2] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *5th Annual ACM Workshop on Computational Leraning Theory*, pages 144–152, 1992.

[3] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, pages 409–415, 2000.

[4] F. Desobry, M. Davy, and C. Doncarli. An online kernel change detection algorithm. *IEEE Transactions on Signal Processing*, 53(8):2961–2974, 2005.

[5] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1450–1464, 2006.

[6] Y.A. Ivanov and A.F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[7] N. Johnson and D.C. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8):609–615, 1996.

[8] P. Laskov, C. Gehl, S. Krüger, and K.R. Müller. Incremental support vector learning: analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909–1936, 2006.

[9] T. W. Liao. Clustering of time series data – a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[10] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. Iterative incremental clustering of time series. In *9th International Conference on Extending Database Technology*, pages 106–122, Crete, Greece, 2004.

[11] D. Makris and T.J. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Trans. on Systems, Man, and Cybernetics — Part B: Cybernetics*, 35(3):397–408, 2005.

[12] K.R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2), 2001.

[13] C. Piciarelli and G.L. Foresti. On-line trajectory clustering for anomalous events detection. *Pattern Recognition Letters*, 27:1835–1842, 2006.

[14] F.M. Porikli. Trajectory pattern detection by hmm parameter space features and eigenvector clustering. In *8th European Conference on Computer Vision*, Prague, CZ, 2004.

[15] N. Saunier and T. Sayed. Clustering Vehicle Trajectories with Hidden Markov Models. Application to Automated Traffic Safety Analysis. In *International Joint Conference on Neural Networks*, Vancouver, 2006.

[16] B. Schölkopf, , J. Platt, J. Shave-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. Technical Report TR 87, Microsoft Research, Redmon, WA, 1999.

[17] B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

[18] J. Shave-Taylor and N. Cristianini. *Kernel methods for Pattern Analysis*. Cambridge University Press, 2004.

[19] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[20] V. Vapnik and A. Lerner. Pattern recognition using generalized prortrait method. *Automation adn Remote Control*, 24:774–780, 1963.

[21] V.T. Vu, F. Brmond, and M. Thonnat. Automatic video interpretation: A novel algorithm for temporal scenario recognition. In *8th International Joint Conference on Artificial Intelligence*, pages 9–15, Acapulco, MEX, 2003.