Learning and Classification of car trajectories in road video by string kernels

Luc Brun, Alessia Saggese, Mario Vento

Learning and Classification of car trajectories in road video by string kernels

Luc Brun[1], Alessia Saggese[2] and Mario Vento[2]
[1]*GREYC UMR CNRS 6072 ENSICAEN - Université de Caen Basse-Normandie, 14050 Caen, France*
[2]*Dipartimento di Ingegneria Elettronica e Ingegneria Informatica, University of Salerno, I -84084 Fisciano (SA), Italy*
luc.brun@ensicaen.fr, {asaggese, mvento}@unisa.it

Keywords: Abnormal Trajectories Recognition, Abnormal Behaviors Recognition, Clustering.

Abstract: An abnormal behavior of a moving vehicule or a moving person is characterized by an unusual or not expected trajectory. The definition of exptected trajectories refers to supervised learning where an human operator should define expected behaviors. Conversely, definition of usual trajectories, requires to learn automatically the dynamic of a scene in order to extract its typical trajectories. We propose, in this paper, a method able to identify abnormal behaviors based on a new unsupervised learning algorithm. The original contributions of the paper lies in the following aspects: first, the evaluation of similarities between trajectories is based on string kernels. Such kernels allow us to define a kernel-based clustering algorithm in order to obtain groups of similar trajectories. Finally, identification of abnormal trajectories is performed according to the typical trajectories characterized during the clustering step. The method has been evaluated on a real dataset and comparisons with other state-of-the-arts methods confirm its efficiency.

1 INTRODUCTION

In the last decades we have witnessed a growing need for security in many public environments, which has lead to a proliferation in the number of control systems and, consequently, in the presence of acquisition peripherals. In particular, cameras represent a suitable solution for their relative low cost of maintenance, the possibility of installing them virtually everywhere and, finally, the ability to analyze complex events.

For these reasons, a deep analysis has been recently conducted in order to realize control systems able to automatically generate alarms. Most of researches recently conducted in the field of behavior analysis has focused on the recognition of simple activities (*i.e.* running, waving, jumping) in high resolution videos, by exploiting the details of human body (Aggarwal and Ryoo, 2011). The main problem in such an approach lies in the fact that in a lot of real applications detailed information related, for instance, to the pose or to the clothing colors of people are not available.

As a matter of fact, in these situations, objects are in a far-field or video has a low-resolution: the only information that a video analytic system is reliably able to extract is a noisy trajectory. This consideration has recently drawn the scientific community to store (d'Acierno et al., 2012a) (d'Acierno et al., 2012b) and analyze moving objects' trajectories in order to understand their behaviors, identifying abnormal ones (Piciarelli et al., 2008)(Acampora et al., 2012). In fact, in a lot of real contexts trajectories provide the control system with enough elements to detect an anomalous behavior inside a scene: think, as an example, to a person who moves in the opposite direction of a crowd or who follows a path that the system has never seen.

The architecture of a system for behavior understanding is usually based on the following steps: *learning phase* and *operating phase*. The learning phase aims at defining rules or at extracting prototypes of normal trajectories. The definition of rules is strongly dependent on the environment and, at the same time, on the knowledge that the human operator has about the possible (ab)normal behaviors (Dee and Hogg, 2004). On the other hand, the extraction of prototypes for (ab)normal trajectories can be performed by following one of this two models: supervised and unsupervised. Techniques trained in *supervised* mode (Zhou et al., 2007) assume the availability of a training data set with labeled instances of normal as well as abnormal trajectories. However, such an approach has a significant drawback: abnormal instances are usu-

ally far fewer compared to normal ones in the training set, so implying that the prototypes extracted for abnormal trajectories are not accurate and representative. Furthermore, it is impossible to predict all abnormal behaviors inside a complex scenario.

Techniques operating in *unsupervised* mode (Morris and Trivedi, 2011) do not require labeled data since they make the implicit assumption that normal instances are far more frequent than abnormal ones. An unsupervised learning phase makes the control system context-independent and can be easily applied in different real environments, since it does not use human knowledge. This is a very important and not negligible feature, since it allows the system to autonomously understand dynamics within a scene.

For all these reasons, we propose an unsupervised approach, where an abnormal trajectory refers to something that the control system has never (or rarely) seen. However, a system that raises an alarm for each trajectory which has not been seen before risks to generate too many false alarms: the system needs to identify a normal trajectory as one *enough* similar to one or more trajectories that the system already knows. For this purpose, we propose a learning phase based on the following steps, as depicted in Figure 1a:

- **Trajectory extraction**: the tracking algorithm detailed in (Di Lascio et al., 2012) is applied in order to extract moving objects' trajectories from a video for a long time period.
- **Trajectories representation**: the scene is partitioned into zones according to the distribution of trajectories; starting from this, each trajectory is represented as a sequence of symbols, according to the zones crossed in the scene.
- **Trajectories similarity**: similarity between two trajectories is evaluated by using a kernel-based method. The main advantage in this choice lies in the fact that we may combine these kernels with a large class of clustering and machine learning algorithms, which can be expressed using only scalar product between input data.
- **Clustering**: Given the kernel, a novel clustering algorithm is applied in order to extract clusters of trajectories inside the scene. Each cluster encodes a type of normal trajectories, dynamically extracted from the scene.

Once extracted the prototypes of *normal* trajectories, the control system can start the operating phase, depicted in Figure 1b: for each detected abnormal trajectory, it raises an alarm. In particular we propose to subdivide the operating phase in the following steps:

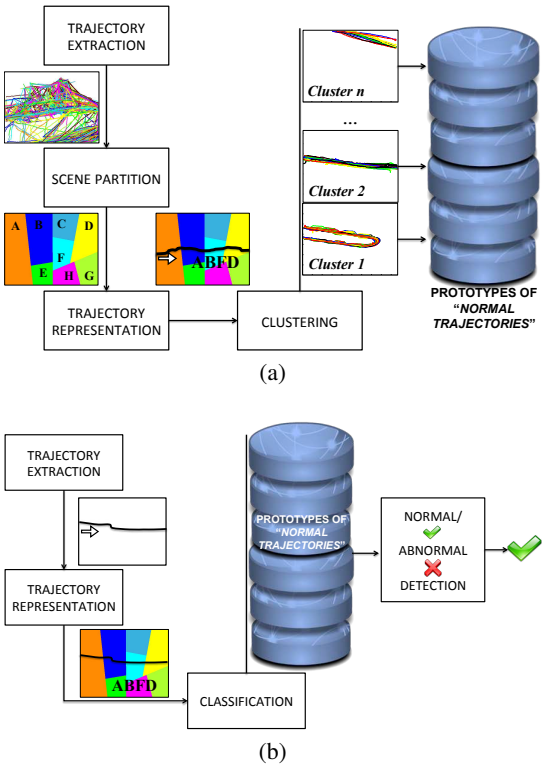


Figure 1: Learning phase (a) and operating phase (b).

- **Trajectory extraction**: the trajectory is extracted from a video by using the tracking algorithm detailed in (Di Lascio et al., 2012).
- **Trajectory representation**: the extracted trajectory is represented as a sequence of symbols.
- **Classification**: the trajectory is compared with the prototypes of each cluster and a similarity value is obtained for each comparison.
- **Decision**: the computed similarity values are processed; if such similarities are sufficiently high the trajectory is considered normal (✓), otherwise it is considered abnormal (✗). In this way, the proposed system is able to identify both rare and atypical trajectories: the former refer to something that does not appear in the training set (or only rarely appears); the latter consider all those trajectories differing in a slightly but significant way from a group of normal trajectories.

In this paper we focus on the learning phase: Subsection 2.1 shows the algorithm used to adaptively partition the scene into zones, while trajectories representation is presented in Subsection 2.2. Some details about the metric used to evaluate the similarity are provided in Subsection 2.3, while Section 3 provides a description of the proposed clustering algo-

rithm. Furthermore, Section 4 shows the approach used to verify if a novel trajectory is normal or abnormal. Experimental results, which confirm the efficiency of the proposed method, are finally presented in Section 5.

2 THE LEARNING PHASE

A trajectory t can be seen as a sequence of k spatio-temporal points $p^i = [p^i_x, p^i_y, p^i_t]$: $t =< p^1, p^2, ..., p^k >$. This representation has two main drawbacks: first, a trajectory results in a very large amount of data to be managed; second, row data are more sensible to noise and tracking errors, and thus a filtering of each trajectory is needed before use. Furthermore, if a system considers the similarity between row data, it can introduce non relevant differences between trajectories. For example, many trajectories on a garden path may be considered as similar independently of the exact position of people on the path.

For this reason, a common representation of a trajectory consists in a reduced sequence of symbols, namely a string, aiming to preserve only the discriminant information and to reduce the space required to store trajectories.

The discriminant information to be preserved is strongly influenced by the aim of the system: as a matter of fact, in order to verify, for instance, if a person is moving in the opposite direction of a crowd or if a vehicle is driving on the emergence line on the highway, the most discriminant feature is the sequence of zones crossed by the moving object. Such scenarios can be labeled as *constrained*: the moving objects are expected to follow given paths within the scene.

From these considerations, we need to partition the scene into a set of zones, hence associating a single symbol to a sequence of points and eliminating non discriminant information. The criterion adopted to subdivide the scene certainly influences the performance of the entire system. As a matter of fact, on one hand it strongly affects the time needed for the computation of similarity between trajectories; on the other hand, it could decrease the reliability of the system if the chosen number of zones is not sufficiently representative of the scene. The simplest way could be to divide the scene using a fixed-size uniform grid, as in (Papadopoulos, 2008). The main drawback of an uniform grid, however, is that each zone has an uneven statistics, causing only a suboptimal statistical segmentation of trajectories. Furthermore, it is evident that the distribution of trajectories in the scene highlights region of interests, in which the major parts of trajectories lie and for which we need an higher level of detail.

In order to overcome these limitations, we propose an adaptive method aimed at minimizing the mean error made when assimilating a trajectory to its zone (Section 2.1). As a consequence of this partitioning criterion, areas in the scene in which most of trajectories lie are represented with an higher number of zones.

2.1 Scene Partitioning

Initially, the scene is represented by a single zone Z_1. The scene partitioning algorithm aims at dividing Z_1 into a fixed number N of zones. The main idea behind our algorithm is to exploit the distribution of the training set by taking into account the density, as in the clustering algorithm proposed in (Brun and Trémeau, 2002). Each zone $(Z_i)_{i\in\{1,...,N\}}$ is represented by using its statistical properties (mean, major axis and covariance matrix); the proposed method works by recursively splitting a selected zone by a set of planes (*cutting planes*) at a chosen position (*cutting position*). In the following, more details about this algorithm will be provided since an enhanced kernelized version of it will be introduced in Section 3.

Let p be a generic point in the scene. We define *f(p)* as the number of trajectories passing through p in the image.

Using function $f(.)$ and our zone's representation, we define the statistical property of a zone Z_i (cardinality $|Z_i|$, Mean μ_i and Covariance Cov_i):

$$\begin{aligned} |Z_i| &= \sum_{p\in Z_i} f(p) \qquad \mu_i = \frac{\sum_{p\in Z_i} f(p)p}{|Z_i|} \\ Cov_i &= \frac{\sum_{p\in Z_i} f(p)p \bullet p^t}{|Z_i|} - \mu_i \bullet \mu_i^t. \end{aligned} \tag{1}$$

A *splitting strategy* is based on the definition of the following steps:

- The selection of the next cluster to be split;
- The selection of the *cutting axis* (*i.e.* the direction normal to each *cutting plane*);
- The selection of the *cutting position* (*i.e.* the location of the cutting plane along the cutting axis).

Since the set of all possible partitions into N zones is too large for an exhaustive enumeration, an heuristic needs to be applied. In particular, we decided to use the following heuristics, detailed in (Braquelaire and Brun, 1997).

Splitting Strategy: Each zone Z_i is recursively split into two sub-zones until the N final zones are obtained. This bipartitioning strategy generates a *tree-structured vector quantization*, as shown in Figure 2. Each leaf of the tree represents a zone of the scene.

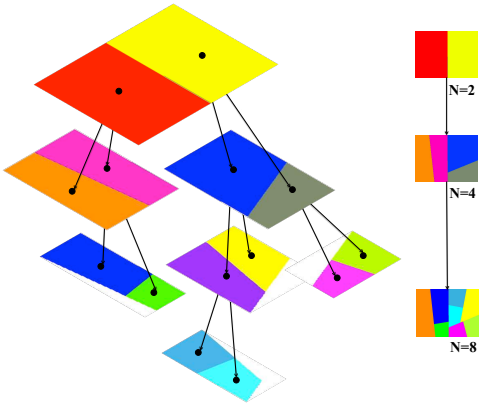


Figure 2: Tree-structured vector quantization obtained recursively partitioning the scene into $N = 8$ zones.

Cluster Selection: At each iteration, a single leaf of the *tree-structured vector quantization* is selected and then split into two zones. Therefore, the choice of the zone to be split plays a crucial role. Our algorithm attempts to minimize the total squared error TSE induced by the partition $P = \{Z_1, ..., Z_N\}$:

$$TSE(P) = \sum_{i=1}^{N} SE(Z_i), \tag{2}$$

where $SE(Z_i)$ is the squared error of one zone Z_i computed as follows:

$$SE(Z_i) = \sum_{p \in Z_i} ||\mu_i - p||^2. \tag{3}$$

In order to minimize $TSE(P)$, the algorithm splits the zone Z with the maximum squared error, since its contribution to the $TSE(P)$ is the largest. It has been shown in (Wan et al., 1988) that this strategy corresponds to a good compromise between the computational time and the final quantization error.

Cutting Axis: Given a zone Z_i to be split, we need to determine the location of the *cutting plane*. As in (Braquelaire and Brun, 1997), we decide to split along the axis with the greatest variance, namely the major axis.

Cutting Position: Once chosen the *cutting axis*, we need to select the optimal cutting position $t^\star$ to subdivide the zone Z_i into two sub-zone $Z_i^{1\star}$ and $Z_i^{2\star}$. In particular, we choose the value able to maximize the decrease of the total squared error induced by the split:

$$SE(Z_i) - \left[SE(Z_i^1) + SE(Z_i^2)\right] \tag{4}$$

Let m and M be respectively the minimum and the maximum projections of Z_i on the cutting axis. It can be proved (Braquelaire and Brun, 1997) that the maximization of Equation 4 can be reached by computing the optimal cutting position $t^\star$ as follows:

$$t^\star = \arg \max_{t \in [m,M]} \left[\frac{\delta(t)}{1-\delta(t)} ||\mu_i - \mu_i^1||^2 \right], \tag{5}$$

where μ_i and μ_i^1 denote respectively the mean of Z_i and $Z_i^{1\star}$ and $\delta(t)$ is defined as $\delta(t) = \frac{|Z_i^1|}{|Z_i|}$. It is worth noting that if $\delta(t) = 1/2$, the zone is divided into two sub-zones with the same cardinality.

An example of the output of the proposed scene partitioning method is provided in Figure 3: starting from the trajectory distribution in Figures 3a and 3b, we show the partition of the scene by using different values of $N = \{20, 50\}$.

2.2 Trajectory representation

Once partitioned the scene into zones, a trajectory can be segmented into l segments: $t = \{< s_1, ..., s_l >\}$, where the j-th segment is defined as the sequence of points lying in the same zone Z_k: $s_j = \{p_i \in < p_a, ...p_b > | p_i \in Z_k\}$.

Finally, the operator $\alpha(\bullet)$ allows us to map the j-th segment into a symbol of our alphabet, each symbol identifying the passing through a zone. It means that the trajectory t can be now defined as $t = \{< \alpha(s_1), ..., \alpha(s_l) >\}$.

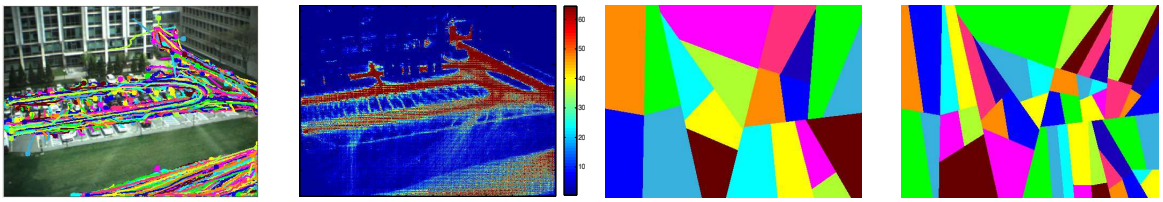
Once obtained the information about the zones crossed in the scene, additional features are extracted in order to improve the reliability of the proposed system. For each segment, information about the speed v and the shape s of the trajectories are taken into account by means of the operator $\theta(\bullet)$. In particular, we use the Bernstein Polynomial Approximation *to model each trajectory into a zone:*

$$s = \sum_{i=0}^{c_s} = a_i^s \cdot t_i \quad v = \sum_{i=0}^{c_v} = a_i^v \cdot t_i. \tag{6}$$

In our experiment, c_s is bounded by 3 and c_v is bounded by 2. The operator $\theta(\bullet)$ is then computed as the vector composed by the a_i values: $\theta(\bullet) = [a_1^s, ..., a_{c_s}^s, a_1^v, ..., a_{c_v}^v]$.

Thanks to this representation, each trajectory can be seen as $t = \{< \alpha(s_1), ..., \alpha(s_l) >, < \theta(s_1), ..., \theta(s_l) >\}$.

Although the operator $\alpha(\bullet)$ gives to our method the most important contribution, the shape of the trajectory is an important feature, since it contributes to distinguish trajectories lying in the same zones but with very different shapes. We can think, for instance, to a person which moves with a very irregular shape but following a common path.



(a) (b) (c) (d)

Figure 3: Partition of the scene starting from the training set depicted in (a) represented by the frequency map in (b). The quantization algorithm is applied with different values of N: (c) N=*20*, (d) N=*50*.

2.3 Trajectories similarity

The complexity and the different typology of information to take into account to represent a trajectory result in a complex strategy to verify the similarity between trajectories. In fact, we need to manage a string for the position and a sequence of numerical values for the speed and the shape, respectively obtained by means of the $\alpha(\bullet)$ and the $\theta(\bullet)$ operators.

In the last years, a lot of different methods based on dynamic programming have been proposed in order to evaluate the similarity between two sequences. These algorithms are based on similarity criteria such as the Dynamic-Time-Warping (DTW) score (Shimodaira et al., 2002), the Smith Waterman algorithm (Saigo et al., 2004) and the edit-distance (Neuhaus and Bunke, 2006). However, the main problem lies in the fact that, although these methods are able to compute a similarity value, they do not define a metric. In order to solve these problems, we propose a novel similarity metric based on kernels: the main advantage is that the problem can then be formulated in an implicit vector space on which statistical methods for pattern analysis can be applied.

Furthermore, thanks to this choice, is it possible to evaluate the similarity between sequences of symbol with different length, so avoiding to force the representation of trajectories to a vector of features with a fixed dimension, as in (Piciarelli et al., 2008).

In particular, we construct our kernel starting from the Fast Global Alignment Kernel (FGAK) proposed in (Cuturi, 2011). The main idea of all global alignment kernels is to measure the similarity between two sequences by summing up scores obtained from local alignments with gaps of the sequences.

An alignment between two sequences $x = \{x_1,...,x_n\}$ and $y = \{y_1,...,y_m\}$ of length n and m respectively is a pair of increasing integral vectors (π_1,π_2) of length $p < n+m$, such that $1 = \pi_1(1) \leq ... \leq \pi_1(p) = n$ and $1 = \pi_2(1) \leq ... \leq \pi_2(p) = m$, with unary increments and no simultaneous repetitions. Let $A(n,m)$ be the set of all the possible alignments between the two time series of lengths n and m.

The global alignment kernel (GAK) is defined as:

$$k_{GA}(x,y) = \sum_{\pi \in A(n,m)} \prod_{i=1}^{|\pi|} k(x_{\pi_1(i)}, y_{\pi_2(i)}). \quad (7)$$

It can be shown (Cuturi, 2011) that k_{GA} is a positive definite kernel if k and $k/(1+k)$ are positive definitive kernels. Furthermore, the GAK avoids the diagonal dominance of the Gram matrix. Diagonal dominance is an undesirable property, since it implies that all the points in a training set are nearly orthogonal to each other in the corresponding feature space.

Starting from the representation of our trajectories, we need to define a kernel which is able to properly combine all the different features related to a trajectory. In particular, we defined the following kernels.

Toeplitz Kernel: In order to speed up the computation of the kernel, we use the triangular kernel for integers, also known as Toeplitz kernel:

$$w(i,j) = \left(1 - \frac{|i-j|}{T}\right), \quad (8)$$

where T is the order of the kernel. The main advantage in the use of the triangular kernel is that it allows to only consider a smaller subset of alignments.

Dirac Kernel: In order to evaluate the similarity between two strings $\alpha(x)$ and $\alpha(y)$ encoding the sequences of zones respectively traversed by trajectories x and y, we use a dirac kernel $\delta(\alpha(x_i),\alpha(y_i))$, defined as:

$$\delta(\alpha(x_i),\alpha(y_i)) = \begin{cases} 0 & \text{if } \alpha(x_i) \neq \alpha(y_i) \\ 1 & \text{if } \alpha(x_i) = \alpha(y_i) \end{cases} \quad (9)$$

The Dirac Kernel is combined with the Toeplitz Kernel so obtaining:

$$k_Z(x_i,y_j,i,j) = w(i,j) \bullet \delta(\alpha(x_i),\alpha(y_j)). \quad (10)$$

Weighted Dirac Kernel: The main lack of this similarity evaluation lies in the fact that the proximity

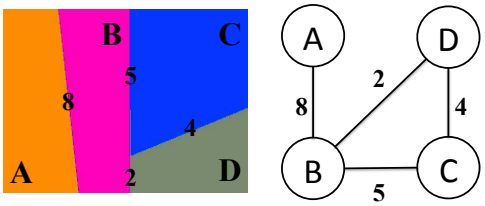


Figure 4: Graph-based representation of the scene. Each zone is represented by a vertex and each border between two zones is represented by an edge. The weight of each edge is determined by the length of the corresponding border.

of two zones is not considered. In order to overcome this limitation by taking into account adjacency relationships between zones, a weighted version of the dirac kernel is also exploited:

$$k_{WZ}(x_i,y_j,i,j) = w(i,j) \bullet \delta_w(\alpha(x_i),\alpha(y_j)). \quad (11)$$

Zones are mapped into a non-oriented weighted graph $G = \{V,E,w\}$, whose vertices $V = \{V_1,...,V_N\}$ identify zones and whose edges $E = \{E_1,...,E_L\}$ identify proximity of two zones. Each edge is associated to a weight e_{v_1,v_2}, identifying the number of pixels separating two zones. An example is shown in Figure 4:

$$\delta_w(\alpha(x_i),\alpha(y_i)) = \begin{cases} 0 & \text{if } \alpha(x_i) \neq \alpha(y_i) \\ & \text{and } e_{\alpha(x_i),\alpha(y_i)} \notin E \\ e^I_{\alpha(x_i),\alpha(y_i)} & \text{if } e_{\alpha(x_i),\alpha(y_i)} \in E \\ 1 & \text{if } \alpha(x_i) = \alpha(y_i) \end{cases} \quad (12)$$

where $e^I_{\alpha(x_i),\alpha(y_i)}$ is a normalized version of $e_{\alpha(x_i),\alpha(y_i)}$, obtained by dividing $e_{\alpha(x_i),\alpha(y_i)}$ by the length of the longest zone's border.

Speed and Shape Kernel: The evaluation of the similarity related to the velocity and to the shape is based on the following kernel:

$$k_{SS}(\theta(x_i),\theta(y_i)) = e^{-\phi_\sigma(\theta(x_i),\theta(y_i))}, \quad (13)$$

where

$$\phi_\sigma(\theta(x_i),\theta(y_i)) = \frac{1}{2\sigma^2}||\theta(x_i)-\theta(y_i)||^2 + \log\left(2 - e^{-\frac{|\theta(x_i)-\theta(y_i)||^2|}{2\sigma^2}}\right). \quad (14)$$

This last kernel is used instead of the Gaussian one in order to guarantee the p.d. of k_{GA} (Cuturi, 2011). The combination of these two last kernels is defined as:

$$k_{(W)ZSS}(x_i,y_j,i,j) = k_{(W)Z}(\alpha(x_i),\alpha(y_j)) \bullet k_{SS}(\theta(x_i),\theta(y_i)). \quad (15)$$

Starting from Equation 7, products of any of the 4 kernels (k_Z, k_{WZ}, k_{ZSS} and k_{WZSS}) can be considered to obtain the final kernel k_{GA}. Finally, a normalization of the kernel is performed in order to normalize kernel's values in the interval $[0,1]$. Therefore, the final normalized kernel k^N_{GA} is:

$$k^N_{GA}(x_i,y_j,i,j) = \frac{k_{GA}(x_i,y_j,i,j)}{\sqrt{k_{GA}(x_i,x_i,i,i) * k_{GA}(y_j,y_j,j,j)}}. \quad (16)$$

3 CLUSTERING

From a general point of view, the goal of a clustering algorithm is to find a fixed number N_C of groups that are both homogeneous and well separated, that is, trajectories within the same group should be similar and entities in different groups dissimilar. In our context, we aim at exploiting a clustering algorithm in order to obtain a set of prototypes of normal trajectories.

In the last decades, a lot of graph-based clustering algorithms (Foggia et al., 2007)(Schaeffer, 2007)(Foggia et al., 2008), like Spectral Clustering or Cut-Clustering, have been exploited. Although these techniques seem to provide good results, they do not allow to readily verify if a novel trajectory belongs to a cluster, that is our main objective.

In order to avoid this restriction, k-means approach and its derivative methods are most frequently used. In particular, the Kernel k-mean (Dhillon et al., 2004) is a generalization of the standard k-means algorithm: the input data are mapped into a higher dimensional feature space through a non-linear transformation and then k-means is applied in feature space. In this way, this algorithm allows to separate the non linearly separable clusters. However, the main problem of such an approach consists in the initialization, which strongly influences the performance of this method since the algorithm converges to the local minimum closest to the initial condition.

In (Tzortzis and Likas, 2009) an improved version of the basic Kernel k-means, the Global Kernel k-means, has been proposed. The main idea is that a near-optimal solution with k clusters can be obtained by starting with a near-optimal solution with $k-1$ clusters and initializing the kth cluster appropriately based on a local search. During the local search, N initializations are tried, where N is the size of the data set. The $k-1$ clusters are always initialized to the $k-1$-clustering problem solution, while the kth cluster for each initialization includes a single point of the data set. The solution with the lowest clustering error is kept as the solution with k clusters. Since the optimal solution for the 1-clustering problem is known,

the above procedure can be applied iteratively to find a near-optimal solution to the M-clustering problem. It is clear that the main drawback of the Global Kernel k-means is the high computational cost, as experimental results conducted will show in Section 5.

In order to overcome these limitations, we introduce a novel and efficient kernelized clustering algorithm. The proposed clustering algorithm is based on the splitting methods presented in Section 2.1: the cluster with the maximum squared error is selected and then split into two different clusters along the major axis. However, the main novelty refers to the kernelization of the considered algorithm.

Thanks to the chosen heuristics, the partitioning of the space into N_C clusters is performed by a sequence of $N_C - 1$ iterations. It is an important and not negligible feature, since a lot of recently proposed clustering algorithms (Tzortzis and Likas, 2009) are very expensive from a computational point of view, as we will show in Section 5.

Given the cluster C containing all the trajectories belonging to the training set, let us consider a generic cluster $C_t \subset C$. This cluster is encoded by the following vector of $\mathbb{R}^{|C|}$:

$$\Lambda_{C_t}^i = \begin{cases} 1 & \text{if } i \in C_t \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Finally, K is the Gramm Matrix of the training set, defined by $K_{ij} = (k_{GA}^N(s_i, s_j))_{ij}$.

As mentioned in Section 2.1, our clustering algorithm builds a sequence of partitions $P_1, \ldots, P_n$ of C, with $P_1 = \{C\}$ and P_n enconding a partition of C into n clusters. The following heuristics have been selected for each step:

Cluster Selection: For each iteration k, the cluster C_t of P_k with the maximum squared error $SE(C_t)$ is selected.

$$SE(C_t) = \sum_{s \in C_t} ||\psi_s - \mu_t||^2 = |C_t| - \frac{1}{|C_t|} \Lambda_{C_t}^t K \Lambda_{C_t}, \tag{18}$$

where ψ_s is the projection of the string s in the Hilbert space implicitly defined by k_{GA}^N. Equation 18 may be evaluated in $O(|C_t|^2)$, with $|C_t|$ denoting the cardinality of C_t A.1.

Cutting Axis: Once selected the cluster C_t, we need to chose a cutting plane to obtain clusters C_t^1 and C_t^2: the optimum cutting plane aims at minimizing $SE(C_t^1) + SE(C_t^2)$.

In particular, we decided to cut the cluster along the major axis, obtained by means of a Kernel PCA (Schölkopf et al., 1998). The Gramm Matrix K is first diagonalized and the centered matrix $K' = (K - 1_{C_t} \cdot K - K \cdot 1_{C_t} + 1_{C_t} \cdot K \cdot 1_{C_t})$ is obtained, with 1_{C_t} being a $|C_t|$ by $|C_t|$ matrix for which each element takes value $1/|C_t|$.

The first eigenvector α, satisfying: $\lambda\alpha = K'\alpha$ is then computed. For each trajectory s, the projection of ψ_s on the major axis α is obtained by:

$$< \alpha, \psi_s >= \sum_{i=1}^{|C_t|} \alpha_i k(s_i, s). \tag{19}$$

Thanks to the computed projections, trajectories belonging to C_t are ordered along the major axis in order to obtain the optimum cutting position.

Cutting Position: Given the selected cluster C_t and its cutting axis, the cutting position $t^\star$ is computed in the range $[m, M]$, being m and M respectively the minimal and the maximal projection of C_t on the major axis by means of Equation 5.

It can be shown A.2 that $||\mu - \mu_t||^2$ can be computed as follows:

$$||\mu - \mu_t||^2 = \frac{1}{|C|^2} e^t K e - \frac{2}{|C_t||C|} e^t K \Lambda_{C_t} + \frac{1}{|C_t|^2} \Lambda_{C_t}^t K \Lambda_{C_t}, \tag{20}$$

where $|C|$ denotes the cardinality of cluster C.

We can note that this operation must be performed for each point in the range $[m, M]$. Since it requires multiple matrix multiplications, it results in a high computational cost. Let p denotes the next trajectory to add to C_t in order to obtain C_{t+1} ($C_{t+1} = C_t \cup \{p\}$). It can be shown A.2 that $||\mu - \mu_{t+1}||^2$ can be efficiently updated from $||\mu - \mu_t||^2$. In particular, the first term $\frac{1}{|C|^2} e^t K e$ is constant since it does not depend on the partition induced by t. Let be $\Lambda_{C_{t+1}} = \Lambda_{C_t} + \delta_p$, being δ_p the vector of zeros containing a single 1 at position p. The second term $e^t K \Lambda_{C_{t+1}}$ of equation 20 may be defined iteratively as follows:

$$e^t K \Lambda_{C_{t+1}} = e^t K \Lambda_{C_t} + \sum_{i=1}^{n} k(i, p). \tag{21}$$

Note that the second term of equation 21 may be precomputed. Equation 21 is thus evaluated in constant time. Finally, the last term $\Lambda_{C_{t+1}}^t K \Lambda_{C_{t+1}}$ becomes:

$$\Lambda_{C_{t+1}}^t K \Lambda_{C_{t+1}} = \Lambda_{C_t}^t K \Lambda_{C_t} + 2 \sum_{i \in C_t} k(i, p) + k(p, p). \tag{22}$$

Using equations 21 and 22, we significantly reduce the computational cost of our algorithm: as a matter of fact, the evaluation of $\|\mu - \mu_{t+1}\|^2$ only requires to compute, for each iteration, values $\sum_{i \in C_t} k(i, p)$ and $k(p, p)$ this last term being equal to 1 since we use a normalized kernel.

Stop Condition: In our context, the clustering algorithm is used to initialize the system during its unsupervised learning phase. It means that the number

of clusters can not be fixed a priori, since the system has not any knowledge about the environment. For this reason, we choose to use as stop condition a lower bound on the mean squared error (MSE) made when assimilating one trajectory to its cluster:

$$MSE(C_t) = \frac{SE(C_t)}{|C_t|}. \quad (23)$$

In this way, the system does not need knowledge of the human operator about the environment, but is able to determine the optimum number of clusters starting from the distribution of trajectories.

4 OPERATING PHASE

The operating phase aims at identifying abnormal behaviors according to the set of typical trajectories determined during the learning phase (Section 3). In particular, our algorithm evaluates the distance between a trajectory t_s and all cluster's centers $C_1, ...C_{N_C}$ obtained during the learning phase. The cluster with the closest mean from t_s is selected as the potential typical trajectory followed by t_s. An additional test should then be performed in order to determine if t_s belongs to this closest cluster. According to this last test t_s is classified as normal (it belongs to one cluster encoding typical trajectories) or an alert is raised and t_s is classified as an abnormal behavior.

Classification: Les s denote the string associated to t_s and Ψ_s the projection of s into the Hilbert space encoded by one of our kernel. The squared distance between Ψ_s and the mean μ_t of a cluster C_t is defined by:

$$\begin{aligned} d_t^2(\mu_t, \Psi_s) &= < \mu_t, \mu_t > + < \Psi_s, \Psi_s > -2 < \mu_t, \Psi_s > \\ &= 1 + 1 - 2 < \mu_t, \Psi_s > \\ &= 2(1 - < \mu_t, \Psi_s >) \\ &= 2\left(1 - \frac{1}{|C_t|} \sum_{s_i \in C_t} k(s, s_i)\right). \end{aligned} \quad (24)$$

Decision: Let C_t^* denote the cluster with the closest center (μ_t^*) determined according to equation 24. Since our clustering algorithm always split clusters according to their axis of greatest variance, we consider that the covariance matrix of each cluster is approximately diagonal. In this case, a threshold on the Gaussian probablity that string t_s belongs to C_t^* is approximated by comparing the squared distance $d^2(\mu_t^*, \Psi_s)$ with a multiple of the squared error of C_t^*:

$$d^2(\mu_t^*, \Psi_s) \leq \alpha * MSE(C_t^*). \quad (25)$$

Conversely to the parameter ν of one class SVM (Cortes and Vapnik, 1995), a high value of α provides a better generalization but may increases the number of false positive in the test determining the classification to C_t^*.

5 EXPERIMENTAL RESULTS

The proposed method has been validated on the MIT trajectories dataset (Wang et al., 2011), a standard and freely available dataset composed by 40.453 trajectories obtained from a parking lot scene within five days. Starting from the entire dataset D, a subset D^* of trajectories belonging to vehicles (10.335) has been manually extracted by an expert (see Figure 3a) and the proposed system has been evaluated. The experiments have been conducted on a MacBook Pro equipped with Intel Core 2 Duo running at 2.4 GHz.

The experimentation has been conducted into three different steps: the first aims to validate the proposed kernels, highlighting advantages and disadvantages of each by means of a visual interpretation. The second is a quantitative and qualitative experimentation conducted over the clustering algorithm, aimed at confirming the efficiency of the proposed method if compared with other state of the art methods. Finally, a misclassification matrix is presented, in order to confirm the efficiency of the proposed classification method.

Experimentation over kernels: As for the first step, it is interesting to note that kernels induce metrics. This means that a quantitative comparison of different kernels is not possible outside a regression or classification task with a ground truth which is not available for this data set. We thus only perform a qualitative evaluation based on visual comparisons. Such an experiment has been performed over the dataset D^*, aiming at extracting the M most similar trajectories with respect to the one shown in Figure 5a (M=30).

The parameters of the kernels that we need to tune are respectively T (for the Toeplitz Kernel) and σ (for the Speed and Shape Kernel). In particular, in (Cuturi, 2011) is shown that the best result can be achieved by using the following parameters:

- The *Bandwidth* σ is set to a multiple of a simple estimate of the median distance K of different points observed in different time-series of our training set, scaled by the square root of the median length of time-series in the training set. In particular, in (Cuturi, 2011) is suggested to try $\sigma \in \{0.1, 1, 10\} \cdot K$, where

$$K = median||\theta(x_i) - \theta(y_i)|| \sqrt{median(||x||)} \quad (26)$$

and to use higher multiples (*e.g.* 2,5). Being $K = 60$ in our dataset, we choose $\sigma = 2 * 60 = 120$.

- The *Triangular* parameter T can be set to a reasonable multiple of the median length, *e.g.* 0.2 or 0.5. Being the median length equal to 5 in our dataset, we chose $T = 3$.

An example of our different kernels at work is shown in Figure 5. It is worth noting that the Weighted Dirac Kernel (Figure 5c) s less sensitive than the Dirac Kernel (Figure 5b) to small variations in the position of trajectories. It is a very desirable property, especially in the operating phase, since it allows to reduce the number of false positive. On the contrary, the combination of Weighted Dirac Kernel, Gaussian and Toeplitz ones (Figure 5d) increases the reliability of the evaluation by taking into account the shape of trajectories.

Experimentation over clustering: As for the second step, a qualitative evaluation of the proposed method over the dataset D^* is shown in Figure 6, where some of the obtained clusters are depicted in a tree structure.

In order to have a quantitative measure, we compute the C-index (Hubert and Schultz, 1976). It is defined as:

$$C = \frac{S - S_{min}}{S_{max} - S_{min}}, \tag{27}$$

where S is the sum of distances over all pairs of objects form the same cluster, n is the number of those pairs and S_{min} is the sum of the n smallest distances if all pairs of objects are considered. Likewise S_{max} is the sum of the n largest distances out of all pairs. The C-index ranges from 0 to 1 and the optimum value is 0. The above mentioned index has been used to compare the proposed method with other state of the art approaches. In particular, we considered the traditional Kernel k-mean with its two improved versions, the Global Kernel k-means (Tzortzis and Likas, 2009) and the Fast Global Kernel k-means (Tzortzis and Likas, 2009).

A comparison in terms of C-Index and computational cost is shown in Tables 1a and 1b. Note that the results of the Kernel k-mean (in terms of C-index and time) is obtained by taking the minimal c-index and overall time over 200 different trials, in order to limit the dependency of the results from a particular initialization.

The value that we obtained, equal to 0.0580 if the similarity value is computed by using the Dirac Kernel, while it is equal to 0.0799 when using a Weighted Dirac Kernel instead of a simple Dirac Kernel. In both cases, the efficiency of our technique is confirmed, since the proposed method outperforms the other considered approaches. Furthermore, the computational cost is not too expensive if we consider that this algorithm does not need to work in real time, since it is only used at the start-up of the system.

Method (Dirac Kernel)	C-Index	Time (secs)
Proposed Method	**0.0580**	947.18
K-Means	0.3907	**15.946**
Global K-Means	0.1282	$3.45 \cdot 10^4$
Fast Global K-Means	0.1877	67.882

(a)

Method (Weighted Dirac Kernel)	C-Index	Time (secs)
Proposed Method	**0.0799**	1055.98
K-Means	0.4203	**12.783**
Global K-Means	0.1376	$3.45 \cdot 10^4$
Fast Global K-Means	0.2868	75.201

(b)

Table 1: Comparison of the proposed method with other state of the art approaches, in terms of C-index and computational cost. The similarity between trajectories is computed by using the Dirac Kernel (a) and the Weighted Dirac Kernel (b).

		Predicted Class	
		Normal	**Abnormal**
GT	**Normal**	84.10%	21.40%
	Abnormal	15.90%	76.60%

(a)

		Predicted Class	
		Normal	**Abnormal**
GT	**Normal**	85.30%	7.10%
	Abnormal	14.70%	92.90%

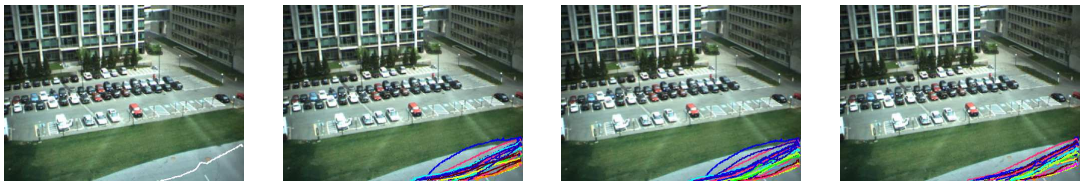
(b)

Table 2: Misclassification Matrix obtained by using Dirac Kernel (a) and Weighted Dirac Kernel (b).

Experimentation over operating phase: The dataset D^* has been divided into three folds and one of these has been used for the learning phase. The remaining two folds have been mixed with the remaining trajectories ($D \setminus D^*$) and are used to test the system. The tests have been performed by computing the similarity between trajectories by using the Dirac Kernel and the Speed and Shape Kernel.

The confusion matrix obtained is depicted in Table 2. The results, for a fixed value of α ($\alpha = 2$), show that the Weighted Dirac Kernel provides a better generalization than the Dirac Kernel, without paying in terms of false positive errors.

Furthermore, it is worth pointing out that in this case it has not been possible to compare our approach with other state-of-the-art methods since the results



(a) (b) (c) (d)

Figure 5: Most similar trajectories to the one shown in (a), obtained by using Dirac Kernel (b), Weighted Dirac Kernel (c), Speed and Shape Kernel (d).

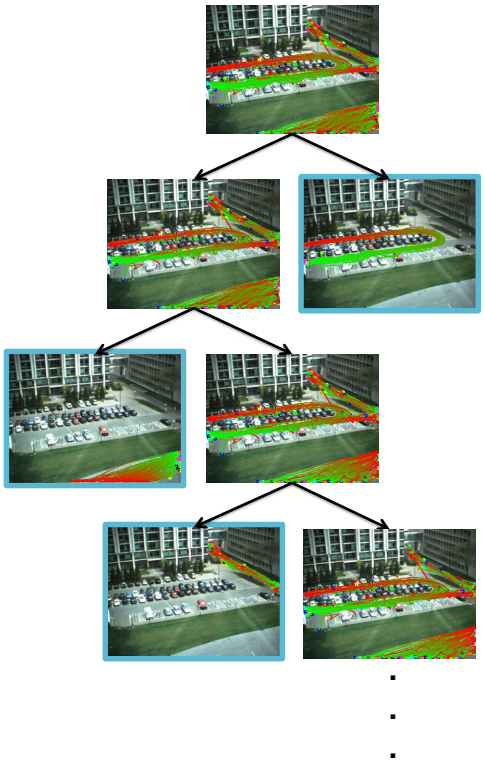


Figure 6: A part of the tree obtained by using the proposed clustering method. The color of each trajectory highlights its direction: it starts in green and progressively turns its color into red.

on the considered dataset are not available. As a matter of fact, a lot of recently proposed methods have been tested over owner datasets, not always available.

In any case, starting from the obtained results, which are sufficiently good for most practical applications, we can enforce the effectiveness of the method by drawing some considerations about the nature of the errors; first of all, as shown in Figure 7a, we can see a trajectory labelled as abnormal in our dataset, but classified as genuine by the system. This happens because they refer to vehicle's trajectories partially located in the grass (labelled as abnormal in the dataset) and at the same time very similar in shape and close to others that avoid the grass just for a few centimeters. The system, based for its nature on shape similarity, has no chance to give a correct answer. It is worth noting that this error cannot be recovered by any unsupervised system based on similarity evaluation; only the introduction of information about the boundaries of the areas can provide a system the possibility of discriminating among very close trajectories on the basis of possible boundary crosses. Another similar situations occurs in Figure 7b, where the vehicle tries to park, but because of car place lack, fails, consequently leaving the parking by exiting after a complete turn. In this case, the description of the trajectory follows a regular and normal path, except for a very limited stretch, reproducing the same typology of error occurring in the previous case.

Opposite kinds of error occur in Figure 8, in which some trajectories, normal with respect to their semantic and consequently labelled in this way, are wrongly classified as abnormal by the system because their short dimension, probably caused by a tracking algorithm error, do not allow to associate them with a sufficient reliability to a cluster containing normal trajectories.

In conclusions, the yet sufficiently high performance are even more acceptable at the light of the fact that a rather high percentage of errors cannot be recovered by using unsupervised methods, as their elimination would require further information possibly given by a supervised approach. However, we could think, as future work, the introduction of a mixed solution based both on clustering and boundary-constraints so as to catch the advantages of both these approaches, even at the cost of introducing a little more heavy a priory knowledge about the scene to be processed.

6 CONCLUSIONS

We propose a system able to identify abnormal trajectories without the human operator explicitly defining the rules. It has been achieved by introduc-

(a) (b)

Figure 7: Abnormal trajectories classified as normal.

(a) (b)

Figure 8: Normal trajectories classified as abnormal.

ing an unsupervised method able to deduce properties of a scene from a set of trajectories. Starting from a set of normal trajectories acquired by means of a video analysis system, our method represent each trajectory by a sequence of symbols associated to relevant features of trajectories (crossed zones, shape and speed in each zone). This quantization is obtained by partitioning the scene into a fixed number of adaptive zones. Similarity between trajectories is evaluated by means of a fast alignment global kernel. Trajectories are then grouped into homogenous clusters encoding normal trajectories. The classification into (ab)normal trajectories is performed by taking advantaging on the statistical properties of the clusters. Experiments have been performed on a real dataset and the obtained results, compared with other state of the art methods, confirm the efficiency of the proposed approach. However, a deeper analysis will be conducted in order to confirm the efficiency of the entire approach if compared with other state-of-the art methods. Furthermore, some more reliable classifiers will be exploited in order to increase the overall dependability of the system.

A APPENDIX

In the following, the formal proofs of the computation of the squared error and of the cutting position of the clustering algorithm are provided.

A.1 Squared Error

Let be:

$$b_s^i = \begin{cases} 1 - \frac{1}{|C_t|} & \text{if } i = s \\ -\frac{1}{|C_t|} & \text{if } i \in C_t \wedge i \neq s \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

$$\delta_s^i = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

$$b_s = -\frac{1}{|C_t|}\Lambda_{C_t} + \delta_s. \tag{30}$$

$$\begin{aligned} SE(C_t) &= \sum_{s \in C_t} ||\psi_s - \mu_t||^2 \qquad (31)\\ &= \sum_{s \in C_t} ||\psi_s - \frac{1}{|C_t|}\sum_{i \in C_t}\psi_i||^2 \\ &= \sum_{s \in C_t} ||\sum_{i \in C_t} b_s^i \psi_i||^2 \\ &= \sum_{s \in C_t}\sum_{i,j \in C_t} b_s^i b_s^j < \psi_i, \psi_j > \\ &= \sum_{s \in C_t} b_s^t K b_s \\ &= \sum_{s \in C_t} \left(-\frac{1}{|C_t|}\Lambda_{C_t} + \delta_s\right)^t K \left(-\frac{1}{|C_t|}\Lambda_{C_t} + \delta_s.\right) \\ &= \sum_{s \in C_t} \frac{1}{|C_t|^2}\Lambda_{C_t}^t K \Lambda_{C_t} - \frac{2}{|C_t|}\Lambda_{C_t}^t K \delta_s + \delta_s K \delta_s \\ &= \frac{1}{|C_t|}\Lambda_{C_t}^t K \Lambda_{C_t} - \frac{2}{|C_t|}\sum_{s \in C_t}\Lambda_{C_t}^t K \delta_s + \sum_{s \in C_t} k(s,s) \\ &= \frac{1}{|C_t|}\Lambda_{C_t}^t K \Lambda_{C_t} - \frac{2}{|C_t|}\Lambda_{C_t}^t K \left(\sum_{s \in C_t}\delta_s\right) + \sum_{s \in C_t} k(s,s) \\ &= -\frac{1}{|C_t|}\Lambda_{C_t}^t K \Lambda_{C_t} + \sum_{s \in C_t} k(s,s) \\ &= |C_t| - \frac{1}{|C_t|}\Lambda_{C_t}^t K \Lambda_{C_t} \end{aligned}$$

A.2 Cutting Position

Let be:

$$\mu = \frac{1}{|C|}\sum_{i \in C}\Psi_i \tag{32}$$

$$\mu_t = \frac{1}{|C_t|}\sum_{i \in C_t}\Psi_i = \frac{1}{|C_t|}\sum_{i \in C}\Lambda_{C_t}^i \Psi_i. \tag{33}$$

$$b_i = \frac{1}{|C|} - \frac{\Lambda_{C_t}^i}{|C_t|}; \tag{34}$$

$$b = \frac{1}{|C|}e - \frac{\Lambda_{C_t}^i}{|C_t|}, \tag{35}$$

being e a vector of one values.

It can be shown that $||\mu-\mu_t||^2$ can be computed as follows:

$$
\begin{aligned}
||\mu-\mu_t||^2 &= ||\sum_{i\in C}\frac{\Psi_i}{|C|}-\frac{\Lambda^i_{C_t}\Psi_i}{|C_t|}||^2 \qquad (36)\\
&= ||\sum_{i\in C}\left(\frac{1}{|C|}-\frac{\Lambda^i_{C_t}}{|C_t|}\right)\Psi_i||^2\\
&= ||\sum_{i\in C} b_i\Psi_i||^2\\
&= \sum_{i,j\in C} b_i b_j < \Psi_i,\Psi_j >\\
&= \sum_{i,j\in C} b_i b_j \cdot k(s_i,s_j)\\
&= b^t K b\\
&= \left(\frac{1}{|C|}e-\frac{\Lambda^i_{C_t}}{|C_t|}\right)^t K \left(\frac{1}{|C|}e-\frac{\Lambda^i_{C_t}}{|C_t|}\right)\\
&= \frac{1}{|C|^2}e^tKe-\frac{2}{|C_t||C|}e^tK\Lambda_{C_t}+\frac{1}{|C_t|^2}\Lambda^t_{C_t}K\Lambda_{C_t}.
\end{aligned}
$$

Furthermore, starting from $||\mu-\mu_t||^2$, $||\mu-\mu_{t+1}||^2$ can be quickly computed. In particular, the second term $e^tK\Lambda_{C_{t+1}}$ is:

$$
\begin{aligned}
e^tK\Lambda_{C_{t+1}} &= e^tK(\Lambda_{C_t}+\delta) \qquad (37)\\
&= e^tK\Lambda_{C_t}+e^tK\delta_p\\
&= e^tK\Lambda_{C_t}+\sum_{i=1}^{n}k(i,p).
\end{aligned}
$$

Finally, the last term $\Lambda^t_{C_{t+1}}K\Lambda_{C_{t+1}}$ becomes:

$$
\begin{aligned}
\Lambda^t_{C_{t+1}}K\Lambda_{C_{t+1}} &= (\Lambda_{C_t}+\delta_p)^t K(\Lambda_{C_t}+\delta_p) \qquad (38)\\
&= \Lambda^t_{C_t}K\Lambda_{C_t}+2\Lambda^t_{C_t}K\delta_p+\delta^t_pK\delta_p\\
&= \Lambda^t_{C_t}K\Lambda_{C_t}+2\sum_{i\in C_t}k(i,p)+k(p,p).
\end{aligned}
$$

REFERENCES

Acampora, G., Foggia, P., Saggese, A., and Vento, M. (2012). Combining neural networks and fuzzy systems for human behavior understanding. In *to appear in Proceedings of the "IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)"*.

Aggarwal, J. and Ryoo, M. (2011). Human activity analysis: A review. *ACM Comput. Surv.*, 43(3):16:1–16:43.

Braquelaire, J. P. and Brun, L. (1997). Comparison and optimization of methods of color image quantization. *IEEE Transactions on Image Processing*, 6(7):1048–1052.

Brun, L. and Trémeau, A. (2002). *Digital Color Imaging Handbook*, chapter 9 : Color quantization, pages 589–637. Electrical and Applied Signal Processing. CRC Press.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20:273–297.

Cuturi, M. (2011). Fast global alignment kernels. In Getoor, L. and Scheffer, T., editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 929–936, New York, NY, USA. ACM.

d'Acierno, A., Leone, M., Saggese, A., and Vento, M. (2012a). An efficient strategy for spatio-temporal data indexing and retrieval. In *to appear in Proceedings of the "International Conference on Knowledge Discovery and Information Retrieval (KDIR)"*.

d'Acierno, A., Leone, M., Saggese, A., and Vento, M. (2012b). A system for storing and retrieving huge amount of trajectory data, allowing spatio-temporal dynamic queries. In *to appear in Proceedings of the "IEEE Conference on Intelligent Transportation Systems (ITSC)"*.

Dee, H. and Hogg, D. (2004). Detecting inexplicable behaviour. In *In: Proceedings of the British Machine Vision Conference, The British Machine Vision Association*, pages 477–486.

Dhillon, I. S., Guan, Y., and Kulis, B. (2004). Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 551–556. ACM.

Di Lascio, R., Foggia, P., Saggese, A., and Vento, M. (2012). Tracking interacting objects in complex situations by using contextual reasoning. In Csurka, G. and Braz, J., editors, *VISAPP (2)*, pages 104–113. SciTePress.

Foggia, P., Percannella, G., Sansone, C., and Vento, M. (2007). Assessing the performance of a graph-based clustering algorithm. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4538 LNCS:215–227.

Foggia, P., Percannella, G., Sansone, C., and Vento, M. (2008). A graph-based algorithm for cluster detection. *IJPRAI*, 22(5):843–860.

Hubert, L. and Schultz, J. (1976). Quadratic assignment as a general data analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29(2):190–241.

Morris, B. and Trivedi, M. (2011). Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(11):2287 –2301.

Neuhaus, M. and Bunke, H. (2006). Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10):1852 – 1863.

Papadopoulos, A. N. (2008). Trajectory retrieval with latent semantic analysis. In *Proceedings of the 2008 ACM*

symposium on Applied computing, SAC '08, pages 1089–1094, New York, NY, USA. ACM.

Piciarelli, C., Micheloni, C., and Foresti, G. (2008). Trajectory-based anomalous event detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1544–1554.

Saigo, H., Vert, J.-P., Ueda, N., and Akutsu, T. (2004). Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689.

Schaeffer, S. (2007). Graph clustering. *Computer Science Review*, 1(1):27–64.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319.

Shimodaira, H., ichi Noma, K., Nakai, M., and Sagayama, S. (2002). Dynamic time-alignment kernel in support vector machine. In *Advances in Neural Information Processing Systems (NIPS2002)*, volume 14(2), pages 921–928. MIT Press.

Tzortzis, G. F. and Likas, A. C. (2009). The global kernel k-means algorithm for clustering in feature space. *Trans. Neur. Netw.*, 20(7):1181–1194.

Wan, S. J., Wong, S. K. M., and Prusinkiewicz, P. (1988). An algorithm for multidimensional data clustering. *ACM Trans. Math. Softw.*, 14(2):153–162.

Wang, X., Ma, K. T., Ng, G.-W., and Grimson, W. E. (2011). Trajectory analysis and semantic region modeling using nonparametric hierarchical bayesian models. *Int. J. Comput. Vision*, 95:287–312.

Zhou, Y., Yan, S., and Huang, T. (2007). Detecting anomaly in videos from trajectory similarity analysis. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1087 –1090.