A clustering algorithm of trajectories for behaviour understanding based on string kernels

Luc Brun, A. Saggese, Mario Vento

A clustering algorithm of trajectories for behaviour understanding based on string kernels

Luc Brun
GREYC UMR CNRS 6072
ENSICAEN - Université de Caen Basse-Normandie
14050 Caen, France
luc.brun@ensicaen.fr

Alessia Saggese and Mario Vento
Department of Electronic and Information Engineering
University of Salerno
I -84084 Fisciano (SA), Italy
{asaggese, mvento}@unisa.it

***Abstract*—This work aims to identify abnormal behaviors from the analysis of humans or vehicles' trajectories. A set of normal trajectories' prototypes is extracted by means of a novel unsupervised learning technique: the scene is adaptively partitioned into zones by using the distribution of the training set and each trajectory is represented as a sequence of symbols by taking into account positional information (the zones crossed in the scene), speed and shape. The main novelties of this work are the following: first, the similarity between trajectories is evaluated by means of a kernel-based approach. Furthermore, we define a novel and efficient kernel-based clustering algorithm, aimed at obtaining groups of normal trajectories. The proposed approach has been compared with state-of-the-art methods and it clearly outperforms all the other considered techniques.**

I. 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 [1]. 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 [2] [3] and analyze [4] moving objects' trajectories in order to understand their behaviors, identifying abnormal ones. 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 [5]. On the other hand, the extraction of prototypes for (ab)normal trajectories can be performed by following one of this two models [6]: supervised and unsupervised. Techniques trained in *supervised* mode [7] 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 usually 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 [8] 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 [9] 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:

- **Trajectory extraction**: the trajectory is extracted from a video by using the tracking algorithm detailed in [9].
- **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 II-A shows the algorithm used to adaptively partition the scene into zones, while trajectories representation is presented in Subsection II-B. Some details about the metric used to evaluate the similarity are provided in Subsection II-C, while Section III provides a description of the proposed clustering algorithm. Experimental results, which confirm the efficiency of the proposed method, are finally presented in Section IV.

II. THE PROPOSED METHOD

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.

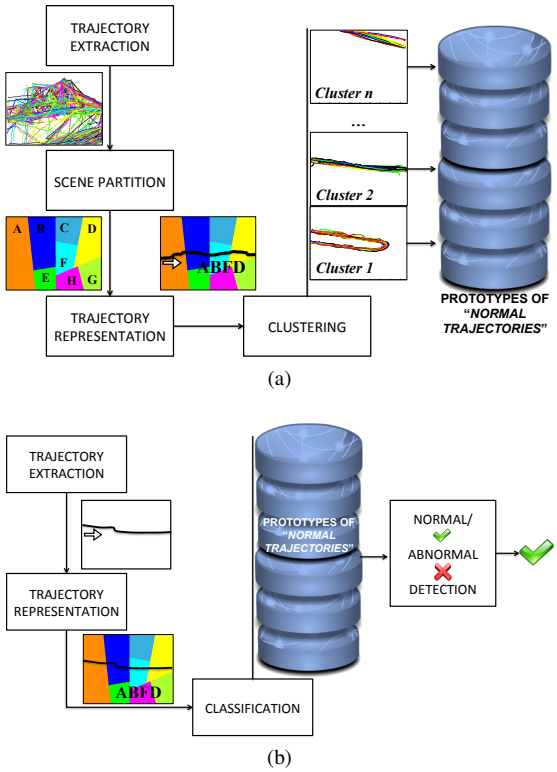


Fig. 1: Learning phase (a) and operating phase (b).

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. 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 II-A). As a

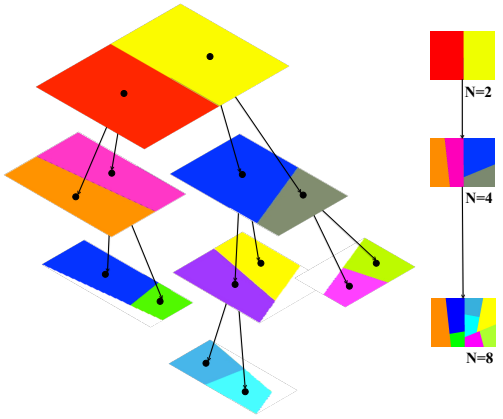


Fig. 2: Tree-structured vector quantization obtained recursively partitioning the scene into $N = 8$ zones.

consequence of this partitioning criterion, areas in the scene in which most of trajectories lie are represented with an higher number of zones.

A. 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 [10]. Each zone $(Z_i)_{i\in\{1,\dots,N\}}$ 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 III.

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 [11].

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.

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 $P = \{Z_1, \dots, Z_N\}$:

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

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 [12] 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 [11], 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 [11] that the maximization of Equation 4 can be reached by computing the optimal cutting position $t^\star$ as follows:

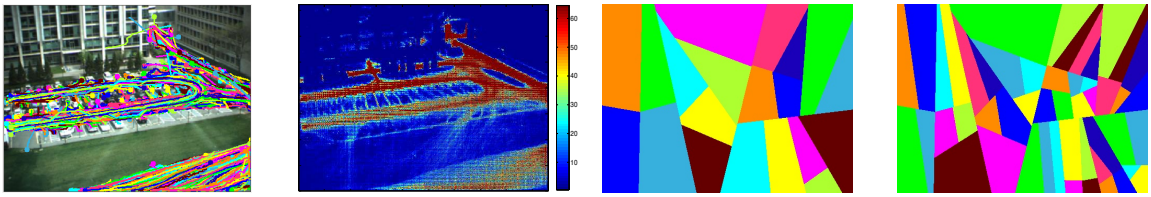
$$t^\star = \arg\max_{t\in[m,M]} \left[\frac{\delta(t)}{1-\delta(t)} ||\mu_i - \mu_i^1||^2\right], \tag{5}$$

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 $N = \{20, 50\}$.

B. Trajectory representation

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



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

Fig. 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*.

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.

C. 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 [13], the Smith Waterman algorithm [14] and the edit-distance [15]. 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.

In particular, we construct our kernel starting from the Fast Global Alignment Kernel (FGAK) proposed in [16]. 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:

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

It can be shown [16] 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:

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

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:

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

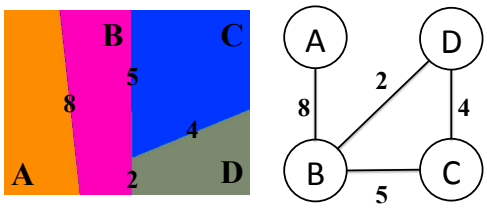


Fig. 4: Graph-based representation of the scene. Each zone is represented by a vertex and each border between two zones is represented by an edge. The weight of each edge is determined by the length of the corresponding border.

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 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} [16]. 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:

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

III. 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. The clustering can be performed with different aims; for instance, in [17] a density-based clustering algorithm has been considered in order to discover common sub-trajectories by means of a partition-and-group framework. As our method, this algorithm performs a sub-sampling of trajectories. However, our method considers all possible sub-samplings of two trajectory when computing their similarity (Eq. 7) while [17] performs one sub sampling based on a MDL criterion. We indeed consider that sub-sampling should be based on trajectory comparisons rather than on an a priori compromise between preciseness and conciseness. Moreover, [17] aims to determine clusters of trajectories roughly following a same line for a period while our method aims to determine clusters of globally similar trajectories up to some short parts.

In the last decades, a lot of graph-based clustering algorithms [18]–[20], 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 [21] 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 [22] 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 IV.

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 II-A: 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 [22] are very expensive from a computational point of view, as we will show in Section IV.

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 $\mathbb{R}^{|C|}$:

$$\Lambda^i_{C_t} = \begin{cases} 1 & \text{if } i \in C_t \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

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

As mentioned in Section II-A, 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.

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

where ψ_s is the projection of the string s in the Hilbert space implicitly defined by k^N_{GA}. Equation 18 may be evaluated in $O(|C_t|^2)$, with $|C_t|$ denoting the cardinality of C_t [23].

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

In particular, we decided to cut the cluster along the major axis, obtained by means of a Kernel PCA [24]. 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:

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

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 [23] that $||\mu - \mu_t||^2$ can be computed as follows:

$$||\mu - \mu_t||^2 = \frac{1}{|C|^2} e^t K e - \frac{2}{|C_t||C|} e^t K \Lambda_{C_t} + \frac{1}{|C_t|^2} \Lambda^t_{C_t} K \Lambda_{C_t}, \quad (20)$$

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 [23] 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:

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

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

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

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.

IV. Experimental Results

The proposed method has been validated on the MIT trajectories dataset [25], 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 two different steps: the former aims to validate the proposed kernels, highlighting advantages and disadvantages of each by means of a visual interpretation. The latter 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.

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). 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.

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 [26]. It is defined as:

$$C = \frac{S - S_{min}}{S_{max} - S_{min}}, \tag{24}$$

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 [22] and the Fast Global Kernel k-means [22].

A comparison in terms of C-Index and computational cost is shown in Tables Ia and Ib. 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 I: 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).

V. Conclusions

We propose 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. 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.

Acknowledgment

This research has been partially supported by A.I.Tech s.r.l. (a spin-off company of the University of Salerno, www.aitech-solutions.eu).

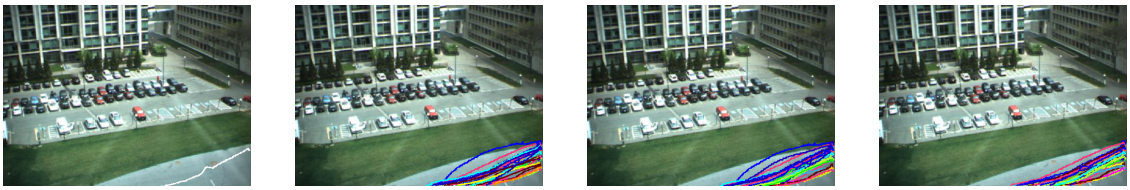
References

[1] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," *ACM Comput. Surv.*, vol. 43, no. 3, pp. 16:1–16:43, Apr. 2011.

[2] A. d'Acierno, M. Leone, A. Saggese, and M. Vento, "A system for storing and retrieving huge amount of trajectory data, allowing spatio-temporal dynamic queries," in *Proceedings of the "IEEE Conference on Intelligent Transportation Systems (ITSC)"*, 2012, pp. 989–994.

[3] ——, "An efficient strategy for spatio-temporal data indexing and retrieval," in *Proceedings of the "International Conference on Knowledge Discovery and Information Retrieval (KDIR)"*, 2012, pp. 227–232.

[4] G. Acampora, P. Foggia, A. Saggese, and M. Vento, "Combining neural networks and fuzzy systems for human behavior understanding," in *Proceedings of the "IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)"*, 2012, pp. 88–93.



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

Fig. 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).

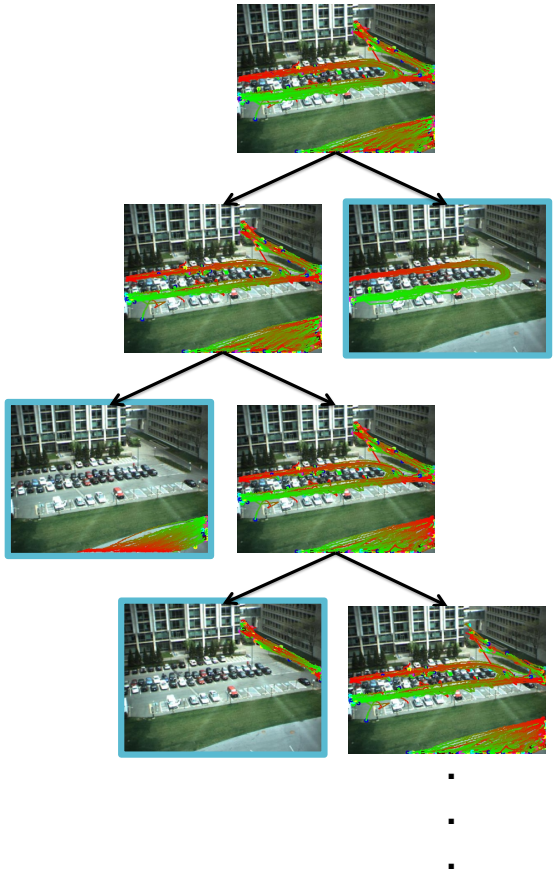


Fig. 6: A part of the tree obtained by using the proposed clustering method. The color of each trajectory highlights its direction: it starts in green and progressively turns its color into red.

[5] H. Dee and D. Hogg, "Detecting inexplicable behaviour," in *In: Proceedings of the British Machine Vision Conference, The British Machine Vision Association*, 2004, pp. 477–486.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[7] Y. Zhou, S. Yan, and T. Huang, "Detecting anomaly in videos from trajectory similarity analysis," in *Multimedia and Expo, 2007 IEEE International Conference on*, july 2007, pp. 1087 –1090.

[8] B. Morris and M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 11, pp. 2287 –2301, nov. 2011.

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

[10] L. Brun and A. Trémeau, *Digital Color Imaging Handbook*, ser. Electrical and Applied Signal Processing. CRC Press, 2002, ch. 9 : Color quantization, pp. 589–637.

[11] J. P. Braquelaire and L. Brun, "Comparison and optimization of methods of color image quantization," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 1048–1052, july 1997.

[12] S. J. Wan, S. K. M. Wong, and P. Prusinkiewicz, "An algorithm for multidimensional data clustering," *ACM Trans. Math. Softw.*, vol. 14, no. 2, pp. 153–162, Jun. 1988.

[13] H. Shimodaira, K. ichi Noma, M. Nakai, and S. Sagayama, "Dynamic time-alignment kernel in support vector machine," in *Advances in Neural Information Processing Systems (NIPS2002)*, vol. 14(2). MIT Press, Dec 2002, pp. 921–928.

[14] H. Saigo, J.-P. Vert, N. Ueda, and T. Akutsu, "Protein homology detection using string alignment kernels," *Bioinformatics*, vol. 20, no. 11, pp. 1682–1689, Jul. 2004.

[15] M. Neuhaus and H. Bunke, "Edit distance-based kernel functions for structural pattern classification," *Pattern Recognition*, vol. 39, no. 10, pp. 1852 – 1863, 2006.

[16] M. Cuturi, "Fast global alignment kernels," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ser. ICML '11, L. Getoor and T. Scheffer, Eds. New York, NY, USA: ACM, June 2011, pp. 929–936.

[17] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '07. New York, NY, USA: ACM, 2007, pp. 593–604. [Online]. Available: http://doi.acm.org/10.1145/1247480.1247546

[18] P. Foggia, G. Percannella, C. Sansone, and M. Vento, "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)*, vol. 4538 LNCS, pp. 215–227, 2007.

[19] S. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, Aug. 2007.

[20] P. Foggia, G. Percannella, C. Sansone, and M. Vento, "A graph-based algorithm for cluster detection," *IJPRAI*, vol. 22, no. 5, pp. 843–860, 2008.

[21] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '04. ACM, 2004, pp. 551–556.

[22] G. F. Tzortzis and A. C. Likas, "The global kernel k-means algorithm for clustering in feature space," *Trans. Neur. Netw.*, vol. 20, no. 7, pp. 1181–1194, Jul. 2009.

[23] L. Brun, A. Saggese, and M. Vento, August 2012. [Online]. Available: http://mivia.unisa.it/zope/mivia/research/clustering/appendix.pdf

[24] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[25] X. Wang, K. T. Ma, G.-W. Ng, and W. E. Grimson, "Trajectory analysis and semantic region modeling using nonparametric hierarchical bayesian models," *Int. J. Comput. Vision*, vol. 95, pp. 287–312, December 2011.

[26] L. Hubert and J. Schultz, "Quadratic assignment as a general data analysis strategy," *British Journal of Mathematical and Statistical Psychology*, vol. 29, no. 2, pp. 190–241, 1976.