

# Dynamic Scene Understanding for behavior analysis 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

Dept. of Information Eng., Electrical Eng. and Applied Mathematics

University of Salerno

I - 84084 Fisciano (SA), Italy

{asaggese, mvento}@unisa.it

**Abstract**—This work aims at dynamically understanding the properties of a scene from the analysis of moving object trajectories. Two different applications are proposed: the former is devoted to identify abnormal behaviors, while the latter allows to extract the  $k$  most similar trajectories to the one hand-drawn by an human operator. A set of normal trajectories' models 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. Experimentations, conducted over three standard datasets, confirms the efficiency of the proposed approach.

## I. INTRODUCTION

In the last years we have assisted to a dramatic decrease in the cost of videocameras, which has encouraged the use of such a device in different applicative domains, ranging from video surveillance to marketing purposes. As for the former, most of existing systems still need human intervention: however, a study recently conducted by Security Solutions Magazine confirms that *after 12 minutes of continuous video monitoring, a guard will often miss up to 45% of screen activity. After 22 minutes of video, up to 95% is overlooked*. This consideration has strongly encouraged the definition of systems able to automatically define common behaviors and identify dangerous ones. On the other hand, the analysis of videos is very useful for evaluating marketing strategies by understanding shoppers' behavior: as a matter of fact, the identification of flow patterns, as well as the measurement of alley attractiveness, can be used in a shopping center in order to target appropriate strategies for customers.

In both the above mentioned contexts, the key point is the understanding of *activity patterns*, which are *the underlying hidden process that dictates how objects move in a scene* [1]. In other words, the analysis of moving objects trajectories can allow to understand common behaviors inside a specific context.

It is worth pointing out that most of researches recently conducted in the field of behavior analysis and scene under-

standing has focused on the recognition of simple activities (*i.e.* running, waving, jumping) in high resolution videos, by exploiting the details of human body [2]. However in real applications, devoted to video surveillance or marketing purposes, detailed information related to the pose of persons are not available. As a matter of fact, objects are often in a far-field or video has a low-resolution, so implying that 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 [3] and analyze [4] moving objects' trajectories in order to understand their behaviors and identify abnormal ones. As a matter of fact, the trajectory is a very discriminant feature, since the movement of objects in a scene is not random, but instead have an underlying structure which can be exploited to build some models.

The learning of activity patterns is a very challenging problem, which has interested a lot of scientists in the last years. Starting from a preliminary study [5] [6], in this paper we propose a novel and efficient method for dynamically understanding the activity patterns inside a scene starting from the analysis of moving objects trajectories. A set of normal trajectories' models is extracted by means of a novel unsupervised learning technique: in particular, once extracted moving object trajectories [7] from a video, the scene is adaptively partitioned into zones by using the distribution of the training set; 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 corresponding to normal behaviors. Finally, two different applications are proposed: an abnormal behavior framework, which allows human operator to be informed of abnormal behaviors occurring in the scene; on the other hand, a query by sketch algorithm, which makes possible an efficient search of the  $k$  most similar trajectories to the one hand drawn by the human operator without analyzing the entire dataset. An overview of the proposed approach is shown in Figure 1.

The paper is organized as follows: in Section II the different

techniques recently considered for dynamic scene understanding are presented; a description of our method is proposed in Sections III, IV and V: in particular, the algorithm for partitioning the scene is detailed in Section III-A; Section III-B describes the representation of the trajectories and finally the similarity measure is introduced in Section III-C. Section IV is devoted to the proposed clustering algorithm and Section V to the behavior analysis and query by sketch applications. Finally, the experimental evaluation of our method is presented in Section VI, before concluding the paper in Section VII. Proofs related to our method, together with some additional results, are shown in the electronic annex attached to this paper.

## II. STATE OF THE ART

In the last decades, several methods have been proposed in order to face with the fascinating but at the same time challenging problems of behavior analysis and dynamic scene understanding.

Activity patterns learning usually requires the following two steps: preprocessing of raw trajectories and clustering. In particular, the former aims at producing a trajectory representation which is suitable for the chosen clustering algorithm. Two approaches are typically used: a normalization, which guarantees all trajectories having the same length (for instance by zero padding [8]), or a dimensionality reduction, which allows to represent a trajectory into a lower dimensional space, computationally less expensive [9].

It is clear that the chosen representation strongly influences the definition of a metric encoding the similarity between trajectories. A common strategy consists to represent a trajectory by means of a single features vector and to evaluate the similarity by using one of the common distances: Hausdorf [10], Mahalanobis [11] or Bhattacharyya [12].

In [13] trajectories are represented as a sequence of symbols and the Edit Distance has been exploited in order to evaluate the similarity between them, while in [14] a Dynamic Time Warping based approach is considered. It is a template-based dynamic programming matching technique able to measure similarity between two time series by finding an optimal match. It is conceptually simple and provides robust performance in recovering different speeds and scale variations.

In [15] trajectories are modeled and classified within a state transition matrix by a continuous chain of Hidden Markov Models (HMMs). The main idea in this case is that the observations are defined to be similar in terms of common similarity to a model, expressed through the likelihood function. The main benefit in this choice lies in the ability of the system to cope with the so-called uneven sampling instances, which are non-uniform temporal sampling between consecutive points. The main drawback, however, is that in general a large amount of data is needed to avoid the over-fitting during the HMM training step.

We can note that the main problem of most of the above mentioned techniques lies in the fact that they either induce an important loss of information from trajectories to vectors or that they cannot readily be related to an inner product space

by an isometric embedding hence closing the door to most of statistical tools used in pattern recognition.

As for the clustering step, it aims at discovering prototypical activities starting from the analysis of preprocessed trajectories; furthermore, each clustering algorithm requires that a proper similarity measure has been defined. As summarized in [16], different typologies of clustering have been recently exploited for dynamic scene understanding: hierarchical agglomerative or divisive techniques, graph cuts and spectral methods. Although these techniques seem to provide good results, they do not allow to readily verify if a novel trajectory belongs to a cluster: this is a very important and not negligible feature, since we are interested in recovering normal and abnormal trajectories by verifying if the single trajectory belongs or not to a given cluster.

In order to avoid this restriction,  $k$ -means approach and its derivative methods are most frequently used. In particular, the Kernel  $k$ -mean [17] 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 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 [18] 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  $k$ th 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  $k$ th 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 its high computational cost.

Starting from the limitations of existing approaches, 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, we define a novel and efficient kernelized clustering algorithm, able to perform a partition into  $k$  clusters by only performing  $k - 1$  iterative bisections.

## III. THE PROPOSED METHOD

A trajectory  $t$  can be seen as a sequence of  $k$  spatio-temporal points  $p^i = [p_x^i, p_y^i]$ :  $t = \langle p^1, p^2, \dots, p^k \rangle$ . 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 and the dynamic of the scene: 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 an highway, the most discriminant feature is the sequence of zones crossed by the moving object. Such scenarios can be labeled as *constrained*: 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. In 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 [19]. 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 III-A). 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.

#### 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  $L$  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 [20]. Each zone  $(Z_i)_{i \in \{1, \dots, L\}}$  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 chosen positions (*cutting positions*). In the following, more details about this algorithm will be provided since an enhanced kernelized version of it will be introduced in Section IV.

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  $f(\cdot)$  and our zone's representation, we define the statistical property of a zone  $Z_i$  (cardinality  $|Z_i|$ , Mean  $\mu_i$  and Covariance  $Cov_i$ ):

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

The main idea is to minimize the loss of information induced by the mapping of the set of points in each zone to the mean of the zone they belong to. This property can be achieved by minimizing the total squared error  $TSE$  induced by the partition  $P = \{Z_1, \dots, Z_L\}$ :

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

being  $SE(Z_i)$  the squared error of the  $i$ th zone  $Z_i$  computed as follows:

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

Since the set of all possible partitions into  $L$  zones is too large for an exhaustive enumeration, an heuristic needs to be applied. In particular, we decided to use the following heuristic, detailed in [21]. Each zone  $Z_i$  is recursively split into two sub-zones until the  $L$  final zones are obtained. This *splitting strategy* is based on the definition of the following steps (Figure 5):

- 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).

This bipartitioning strategy generates a *tree-structured vector quantization*, as shown in Figure 2, where 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. In order to minimize  $TSE(P)$ , the algorithm splits the zone  $Z$  with the maximum squared error, since its contribution to  $TSE(P)$  is the largest. It has been shown in [22] 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 [21], we decide to split along the axis with the greatest variance, namely the major axis.

**Cutting Position:** The optimal cutting position  $t^*$  defines the position of the hyperplane along the cutting axis, which allows to subdivide the zone  $Z_i$  into two sub-zone  $Z_i^{1*}$  and  $Z_i^{2*}$ . In particular, we choose the value able to maximize the decrease of the total squared error induced by the split:

$$SE(Z_i) - [SE(Z_i^{1*}) + SE(Z_i^{2*})] \quad (4)$$

Let  $m$  and  $M$  be respectively the minimum and the maximum projections of  $Z_i$  on the cutting axis. It can be proved [21] that the maximization of Equation 4 can be reached by computing the optimal cutting position  $t^*$  as follows:

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

where  $\mu_i$  and  $\mu_i^1$  denote respectively the mean of  $Z_i$  and  $Z_i^{1*}$  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  $L = \{20, 50\}$ .

### 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  $Z_k$ :  $s_j = \{p_i \in < p_a, \dots, p_b > \mid p_i \in Z_k\}$ .

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 defined as  $t = \{< \alpha(s_1), \dots, \alpha(s_l) >\}$ .

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:

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

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:  $\theta(s) = [a_1^s, \dots, a_{c_s}^s, a_1^v, \dots, a_{c_v}^v]$ .

Thanks to this representation, each trajectory is encoded by  $t = \{< \alpha(s_1), \dots, \alpha(s_l) >, < \theta(s_1), \dots, \theta(s_l) >\}$ .

Although the operator  $\alpha(\bullet)$  gives to our method the most important contribution, the shape of the trajectory contributes to distinguish trajectories lying in the same zones but with very different shapes. The definition of a proper similarity value (Section III-C) allows to weight both types of information.

### 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 kernels based on dynamic programming have been built, namely the Dynamic-Time-Warping (DTW) score [23], the Smith Waterman algorithm

[24] or the edit-distance [25]. However, as stated in [26], the main problem lies in the fact that, although these methods are able to compute a distance, it is not easy to translate them into positive definite kernels and then to define a metric, which is an important requirement of kernel machines in the training phase. 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 [27]. 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  $x = \{x_1, \dots, x_n\}$  and  $y = \{y_1, \dots, y_m\}$  of length  $n$  and  $m$  respectively is a pair of increasing integral vectors  $(\pi_1, \pi_2)$  of length  $p < n + m$ , such that  $1 = \pi_1(1) \leq \dots \leq \pi_1(p) = n$  and  $1 = \pi_2(1) \leq \dots \leq \pi_2(p) = m$ , 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 [27] 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  $k$  which is able to properly combine all the different features related to a trajectory. In particular, we defined the following kernels.

**Triangular Kernel:** In order to speed up the computation of the kernel, we use the triangular kernel for integers, also known as Toeplitz kernel [27]:

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

where  $T$  is the order of the kernel. The main idea behind the introduction of the triangular kernel is to consider only a small but feasible subset of all the possible alignments induced by the GAK. As a matter of fact, if the two time-series length differ by more than  $T$ , their kernel value is equal to 0.

**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  $\delta(x_i, y_i)$ , defined as:

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

The Dirac Kernel is combined with the Toeplitz Kernel so

obtaining:

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

**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:

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

Zones are mapped into a non-oriented weighted graph  $G = (V, E, w)$ , whose vertices  $V$  identify zones and whose edges  $E$  identify zones' adjacencies (Figure 4). Each edge is associated to a weight  $w(e_{ij})$ , defined as the number of pixels separating zones  $i$  and  $j$  divided by the double length of the longest zone's border. The maximal value of  $w(e_{i,j})$  is thus equal to  $1/2$ . Given this graph encoding, the function  $\delta_w(\bullet, \bullet)$  is then defined as follows:

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

**Speed and Shape Kernel:** The evaluation of the similarity related to the velocity and to the shape is based on the following kernel:  $k_{SS}(x_i, y_i) = e^{-\phi_\sigma(\theta(x_i), \theta(y_i))}$ , where:

$$\phi_\sigma(\theta(x_i), \theta(y_i)) = \frac{1}{2\sigma^2} \|\theta(x_i) - \theta(y_i)\|^2 + \log \left( 2 - e^{-\frac{\|\theta(x_i) - \theta(y_i)\|^2}{2\sigma^2}} \right) \quad (13)$$

This last kernel is used instead of the Gaussian one in order to guarantee the p.d. of  $k_{GA}$  [27]. The combination of these two last kernels is defined as:

$$k_{(W)ZSS}(x_i, y_j, i, j) = k_{(W)Z}(x_i, y_j, i, j) \bullet k_{SS}(x_i, y_i). \quad (14)$$

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  $k_{GA}^N$  is:

$$k_{GA}^N(x_i, y_j, i, j) = \frac{k_{GA}(x_i, y_j, i, j)}{\sqrt{k_{GA}(x_i, x_i, i, i) * k_{GA}(y_j, y_j, j, j)}}. \quad (15)$$

#### IV. CLUSTERING

The proposed clustering algorithm is based on the splitting methods presented in Section III-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  $N$  clusters is performed by a sequence of  $N - 1$  iterations. It is an important and not negligible feature, since a lot of recently proposed clustering algorithms [18] are very expensive from a computational point of view, as we will show in Section VI.

Given the cluster  $T$  containing all trajectories belonging to

the training set (*i.e.* the root of the tree), let us consider an arbitrary order on  $T = (t_1, \dots, t_{|T|})$  and a generic cluster  $C \subset T$ . This cluster is encoded by the vector  $\Lambda_C^t$  of  $\mathbb{R}^{|T|}$ , which is 1 if  $i \in C$  and 0 otherwise. Finally,  $K$  is the Gram Matrix of the training set, defined by  $K_{ij} = (k_{GA}^N(s_i, s_j))_{ij}$ .

As mentioned in Section III-A, our clustering algorithm builds a sequence of partitions  $P_1, \dots, P_n$  of  $C$ , with  $P_1 = \{T\}$  and  $P_N$  encoding a partition of  $T$  into  $N$  clusters. The following heuristics have been selected for each step; a simple example is shown in Figure 5.

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

$$SE(C) = \sum_{s \in C} \|\psi_s - \mu\|^2 = |C| - \frac{1}{|C|} \Lambda_C^t K \Lambda_C, \quad (16)$$

where  $\psi_s$  is the projection of the string  $s$  in the Hilbert space implicitly defined by  $k_{GA}^N$ . Equation 16 may be evaluated in  $O(|C|^2)$ , with  $|C|$  denoting the cardinality of  $C$  (Electronic Annex, Appendix B).

**Cutting Axis:** Once selected the cluster  $C$ , we need to chose a cutting plane in order to partition the cluster  $C$  into two clusters  $C_t$  and  $(C \setminus C_t)$ ; the optimum cutting plane aims at minimizing  $SE(C_t) + SE(C \setminus C_t)$  (Figure 5(b)).

A traditional heuristic to minimize  $SE(C_t) + SE(C \setminus C_t)$  consists in splitting  $C$  along its axis with the greatest variance, *i.e.* the major axis. This last axis can be obtained by means of a Kernel PCA [28]. Let be  $K_C$  the Gram Matrix associated to the cluster  $C$ . It is first diagonalized and the centered matrix  $\tilde{K}_C$  is obtained:

$$\tilde{K}_C = (K_C - 1_C \cdot K_C - K_C \cdot 1_C + 1_C \cdot K \cdot 1_C) \quad (17)$$

with  $1_C$  being a  $|C|$  by  $|C|$  matrix for which each element takes value  $1/|C|$ .

The largest eigenvector  $\alpha$  of  $\tilde{K}_C$ , satisfying:  $\lambda \alpha = \tilde{K}_C \alpha$  is then computed. For each trajectory  $s$ , the projection of  $\psi_s$  on the major axis  $v$  of  $C$  is obtained by:

$$\langle v, \psi_s \rangle = \sum_{i=1}^{|C|} \alpha_i k(s_i, s). \quad (18)$$

**Cutting Position:** Thanks to the computed projections, trajectories belonging to  $C$  are ordered along the major axis and the computation of the optimum cutting position  $t^*$  can be done in the range  $[m, M]$ , being  $m$  and  $M$  respectively the minimal and the maximal projection of  $C$  on the major axis by means of Equation 5 (Figure 5(c)).

It can be shown (Electronic Annex, Appendix C) that  $\|\mu - \mu_t\|^2$  can be computed as follows:

$$\|\mu - \mu_t\|^2 = \frac{1}{|C|^2} \Lambda_C^t K \Lambda_C - \frac{2}{|C_t| |C|} \Lambda_C^t K \Lambda_{C_t} + \frac{1}{|C_t|^2} \Lambda_{C_t}^t K \Lambda_{C_t}, \quad (19)$$

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 (Electronic Annex, Appendix C) that  $\|\mu - \mu_{t+1}\|^2$  can be efficiently updated from  $\|\mu - \mu_t\|^2$ . In particular, the first term  $\frac{1}{|C|^2} \Lambda_C^t K \Lambda_C$  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  $\delta_p$  the vector of zeros containing a single 1 at position  $p$ . The second term  $\Lambda_{C_{t+1}}^t K \Lambda_{C_{t+1}}$  of equation 19 may be defined iteratively as follows:

$$\Lambda_{C_{t+1}}^t K \Lambda_{C_{t+1}} = \Lambda_{C_t}^t K \Lambda_{C_t} + \sum_{i \in C} k(i, p). \quad (20)$$

Note that the second term of equation 20 may be precomputed. Equation 20 is thus evaluated in constant time. Finally, the last term  $\Lambda_{C_{t+1}}^t K \Lambda_{C_{t+1}}$  becomes:

$$\Lambda_{C_{t+1}}^t K \Lambda_{C_{t+1}} = \Lambda_{C_t}^t K \Lambda_{C_t} + 2 \sum_{i \in C} k(i, p) + k(p, p). \quad (21)$$

Using equations 20 and 21, 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:  $MSE(C_t) = SE(C_t)/|C_t|$ . 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.

As for the computational cost of the proposed clustering algorithm, it can be shown (Electronic Annex, Appendix A) that each iteration of the algorithm can be performed in  $O(|C|^{2.376})$ .

## V. APPLICATIONS

In order to validate the proposed method, we propose in the following two different applications: the former aims at recognizing abnormal trajectories, while the latter aims at identifying the  $k$  most similar trajectory to the one interactively drawn by the user. More details will be provided respectively in Subsections V-A and V-B.

### A. Abnormal Detection

This phase aims at identifying abnormal behaviors according to the set of typical trajectories determined during the learning phase, detailed in Sections III and IV. In particular, our algorithm evaluates the distance between a trajectory  $t_s$  and all the cluster's centroids  $C_1, \dots, C_N$  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:** Let  $s$  denote the string associated to  $t_s$  and  $\psi_s$  the projection of  $s$  into the Hilbert space encoded by our kernel. The distance between cluster  $C$  and  $\psi_s$  is computed by using a Kernel Mahalanobis Distance  $d_{RC}$ . This choice is justified by the following two reasons: first, we consider the distribution of trajectories inside each cluster to be nearby gaussian, because of the construction of the tree; furthermore, as experimentally evaluated in [29], this distance is advantageous for problems with high class overlap and nonlinear pattern distributions in a kernel-induced feature space.

In particular,  $d_{RC}$  can be also applied to singular gram matrices, since it is based on a previously regularization of the covariance matrix, which prevents it to be singular. Let be  $\tilde{K}_{reg}$  the regularized covariance matrix, obtained as:  $\tilde{K}_{reg} = \tilde{K} + \alpha \cdot I_{|C|}$ ,  $\alpha = |C| \cdot \sigma^2$  being  $\tilde{K}$  and  $\sigma^2 > 0$  respectively the Centered Gram Matrix and a parameter to be chosen, set in our experiments to 1 (as suggested in [29]).  $I_{|C|}$  is the  $|C|$  by  $|C|$  identity matrix.

Let be  $\mathbf{k}_s = [k(s, s_1), \dots, k(s, s_{|C|})]^T$  and  $\tilde{\mathbf{k}}_s = [\tilde{k}(s, s_1), \dots, \tilde{k}(s, s_{|C|})]^T$  respectively the vector of kernel values of  $s$  on the training data and its centered version; according to Equation 17, the generic  $\tilde{k}(s, s_i)$  can be computed as follows:

$$\tilde{k}(s, s_i) = k(s, s_i) - \frac{2}{|C|} \cdot \mathbf{1}_{|C|}^T \cdot \mathbf{k}_s + \frac{1}{|C|^2} \cdot \mathbf{1}_{|C|}^T \cdot K \cdot \mathbf{1}_{|C|}. \quad (22)$$

The distance between  $C$  and  $\psi_s$  can be computed as:

$$d_{RC}^2(\psi_s, C) = \frac{1}{\sigma^2} (\tilde{k}(s, s) - \tilde{\mathbf{k}}_s^T \cdot \tilde{K}_{reg}^{-1} \cdot \tilde{\mathbf{k}}_s). \quad (23)$$

**Decision:** Let  $C^*$  denote the cluster with the lowest distance  $d_{RC}(\psi_s, C^*)$  determined according to Equation 23. A threshold on the probability that string  $t_s$  belongs to  $C^*$  is obtained by comparing the squared distance  $d_{RC}^2(\psi_s, C^*)$  with a fixed threshold  $\alpha$ :  $d_{RC}^2(\psi_s, C^*) \leq \alpha$ . Conversely to the parameter  $\nu$  of one class SVM, a high value of  $\alpha$  provides a better generalization but may increases the number of false positive in the test determining the classification to  $C^*$ .

### B. Query by Sketch

A generic query by sketch aims at retrieving the  $k$  most similar trajectories to the one hand-drawn by the user. In particular, one of the fastest way to solve this kind of problem is to use a  $k$ -d tree [30], a space-partitioning data structure for organizing points in a  $k$ -dimensional space. A  $k$ -d tree is a generalization of a binary tree: the *root* of the tree represents the entire set; each nonterminal node has two *sons* (or *successor nodes*), each representing the two sub-sets induced by the partitioning. The terminal nodes (the leaves of the tree) represent mutually exclusive small subsets of data records, which collectively form a partition of the record space. The main advantage in the use of such a data structure is that a nearest-neighbor query (NN), as well as a  $k$ -NN query, can be answered in a logarithmic time.

The main idea exploited in this paper is that the tree built during the previous step can be considered as a  $k$ - $d$  tree: as a matter of fact, the generic cluster  $C$  is partitioned into two clusters  $C_1$  and  $C_2$ , once properly chosen a cutting axis and a cutting position; it means that, once generated the tree, a  $k$ -NN search can be easily performed without the need to analyze all the trajectories belonging to the dataset.

In order to further speedup the search, two main improvements have been conducted on the tree's building, with respect to the technique detailed in Section IV: the computation of the cutting axis and the stop condition.

**Cutting Axis:** Although being the best known heuristic, the computation of the major axis, and in particular the projection of trajectories on the major axis, is a really expensive operation if a large amount of queries has to be considered: in fact, given the cluster  $C$ , it can be evaluated in  $O(C)$ . However, it is worth pointing out that, whereas the size of the dataset increases, each trajectory can be considered as an axis and the one holding the greatest variance can well approximate the major axis:

$$\begin{aligned} \text{var}(s) &= \frac{1}{|C|} \sum_{j=1}^{|C|} (\langle \psi_j, \psi_s \rangle - \langle \mu, \psi_s \rangle)^2 \\ &= \frac{1}{|C|} \sum_{j=1}^{|C|} k^2(s, s_j) - \left( \frac{1}{|C|} \sum_{j=1}^{|C|} k(s, s_j) \right)^2. \end{aligned} \quad (24)$$

The main advantage in this choice lies in the fact that the projection on the cutting axis of the generic trajectory  $t_i$  can be very quickly computed as the similarity between  $t_i$  and  $t_s^*$ , in  $O(1)$ . Furthermore, as we will show in Section VI-C, the results of the two strategies, major axis and axis with the greatest variance, are still comparable.

**Stop Condition:** Our aim is to minimize the number of trajectories to be analyzed. In particular, the best case arises when only a single leaf of the tree needs to be analyzed. This is mainly why we decided to choose as a stop condition the maximum number of trajectories in each leaf, set to  $k$ .

**Algorithm:** The algorithm defined for solving a query by sketch problem is shown in Figure ?? . It is described as a recursive procedure, where the first invocation passes the root of the built tree as argument. If the node under investigation is a leaf, then all trajectories belonging to it are evaluated and only the  $k$  most similar ones are maintained by using the priority queue  $pq$  (lines 10-12).

If the node under investigation is not a leaf, the son which contains the query *bestSon* is found (lines 14-22) and the procedure is recursively called over it (lines 24-26). Furthermore, the algorithm has to verify if it is necessary to also invoke the procedure over the other child *otherSon* (lines 28-30). This test, also known as *bound overlap ball* test (from now on *bob* test) can be performed by verifying if the hyperplane delimiting the two clusters (*bestSon* and *otherSon*) is not intersected by the ball centered at query position  $x$  and whose *radius* is equal to the maximum distance to the  $k$  trajectories contained in  $pq$ .

In order to clarify this concept, a simple example is shown in Figure 6. In particular, Figure 6a shows a tree obtained by the proposed clustering algorithm with its representation on the plane (Figures 6b, 6c). The red cross represents the query trajectory  $x$ , while the yellow ones refer to the stored trajectories. Finally, the red circle identifies the ball centered at query position  $x$  and whose *radius* is equal to the maximum distance to the  $k$  trajectories contained in  $pq$ . We can note that in Figure 6b the boundary delimiting the two children clusters (identified respectively by the number 4 and 5 in Figure 6a) are not intersected by the ball, while it clearly happens in Figure 6c. It means that only in this last case the algorithm needs to also exploit the trajectories contained in the *otherSon*.

## VI. EXPERIMENTAL RESULTS

This section is devoted to detail the obtained experimental results. A description of the considered datasets and a discussion on the setup of the parameters of our method will be provided respectively in Subsections VI-A and VI-B ; in Subsection VI-C we will show the performance of the clustering algorithm, while in Subsection VI-D we will focus on the classification method. Finally, in Subsection VI-E more detail about the results obtained by using the proposed method for solving query by sketch will be proposed.

### A. Datasets

Three different well-known datasets (Figure 8), acquired in very different environments, have been used in order to test the proposed system:

**MIT Trajectory dataset** [31] is 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, a subset of trajectories belonging to vehicles (10,335) has been manually extracted by an expert.

**Edinburgh Informatics Forum Pedestrian Database** [32] consists of a set of detected targets of people walking through the Informatics Forum, the main building of the School of Informatics at the University of Edinburgh. In particular, we focused on the 7146 trajectories acquired on August, 2010. As already said for the MIT Trajectory Dataset, it has been manually labeled and approximately the 90% has been evaluated as normal.

**MIT Train Station dataset** [33] has been acquired into the New York Grand Central Station; the video, compressed into 1.1 GB AVI file, lasts 33:20 minutes and it is composed by 50,010 frames (25 fps) among which 42,821 moving objects trajectories have been automatically extracted and filtered in order to remove the noise, resulting in 12,414 trajectories.

It is worth pointing out that the trajectories used in all the above mentioned datasets have not been manually extracted by a human operator, but instead by means of different kinds of tracking algorithms. This is a very important consideration, since it allows us to confirm that the statistical nature of the proposed method allows it to also deal with the typical errors of tracking algorithms.

## B. Parameters Setup

This subsection is devoted to justify the choice of the free parameters in the proposed method. The first parameter to be defined is the *number of zones*  $|Z|$  used to partition the scene. It plays a fundamental role since a very small number of zones does not give enough informative content to represent trajectories (think, as an example, if we just use a single zone). On the other hand, a very large number of zones would avoid the system to be enough general to deal with rare but normal trajectories and at the same time would make the system too much sensitive to the error of the tracking phase. However, in Section VI-D we will show that the exact number of zones does not significantly influence the performance of the system, and the range  $|Z| \in \{20, \dots, 40\}$  is a good compromise for all the considered datasets.

The parameters of the kernels that we need to tune are respectively  $T$  (for the Triangular Kernel) and  $\sigma$  (for the Speed and Shape Kernel). In particular, in [27] it is shown that the best results can be achieved by using the following parameters:

- The *Bandwidth*  $\sigma$  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 [27] it is suggested to try  $\sigma \in \{0.1, 1, 10\} \cdot K$ , where  $K = \text{median}(\|\theta(x_i) - \theta(y_i)\| \sqrt{\text{median}(\|x\|)})$  and to use higher multiples (e.g. 2, 5).
- The *Triangular* parameter  $T$  can be set to a reasonable multiple of the median length, e.g. 0.2 or 0.5.

Finally, a proper lower bound on the MSE should be selected as stop condition: since this value strongly depends on the real environment we are analyzing and then cannot be fixed a priori, in Section VI-C we will show how the performance of the clustering algorithm varies by using different thresholds.

## C. Clustering: experimental results

The three above mentioned datasets have been used in order to evaluate the performance of the proposed clustering algorithm. The experiments have been conducted on a MacBook Pro equipped with Intel Core 2 Duo running at 2.4 GHz.

Two different evaluations have been carried out: the former is a quantitative evaluation, performed in terms of C-index and computational cost. The latter is a qualitative evaluation, aiming at visually confirming the effectiveness of the proposed method by showing some of the most representative clusters obtained by using the proposed method. Finally, a comparison both in terms of C-index and computational cost will be carried out by considering other state of the art approaches.

**Quantitative Evaluation:** the C-index [34] is often used in order to evaluate the performance of the clustering algorithms. It is defined as:

$$C = \frac{S - S_{\min}}{S_{\max} - S_{\min}}, \quad (25)$$

where  $S$  is the sum of distances over all pairs of objects from 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.

Figure 9 shows the performance of the proposed algorithm, both in terms of computational cost and C-index for the three different considered datasets. In particular, the graphics in the first row confirm that the time needed to perform the clustering is linear with the number of clusters, which is in turn strongly dependent in our method on the chosen threshold. Furthermore, we can also note that the greatest variance based approach is a good approximation of the major axis based approach: in fact, it is, as expected, less expensive than the major axis based clustering but the C-index score is in practice still comparable. This consideration is confirmed by the second row of Figure 9, where it is shown how the C-index varies with different stop condition criteria (and then with different thresholds). As a matter of fact, we can note that the two methods are comparable in all the considered datasets.

The second row of Figure 9 also gives very important information concerning the optimum threshold value. In fact, we can note that it strongly depends on the particular dataset and on the homogeneity of the trajectories belonging to it, so it can not be established a priori. It is worth pointing out that, depending on the particular application context, it is possible to fix the threshold by minimizing the C-index. The main drawback is that it requires a significant increase in the time needed to train the system: being not stable the trend of the C-index depending on the threshold, it is necessary to run the system with different threshold and to choose the optimum one. In general, we can state that a *good* value is 0.5, being a proper tradeoff between the time needed for the clustering and the quality of the results. For this reason, all experiments conducted in the next Sections will be performed by considering the aforementioned value.

**Qualitative Evaluation:** a qualitative evaluation of the algorithm is provided in Figure 10 and in Figures E.3, E.4 and E.5 of the Electronic Annex. In particular, some representative clusters for each dataset are depicted. The color of each trajectory highlights its direction: it starts in black and progressively turns its color into red. As confirmed by the previous quantitative evaluation, the quality of the clustering algorithm is strongly related to the particular environment: as a matter of fact, the first dataset mainly contains vehicles trajectories: the trajectories are much more compact, being the scenario a constraint one, so implying that all the clusters are very homogeneous (see Figure 10(a)). On the other hand, in the other datasets the trajectories belong to people which are free to move inside a square, without a street which forces them to follow a given path. Although in this case the problem is much more difficult, the result obtained by the proposed clustering algorithm are really promising, as shown in Figures 10(b) and 10(c).

**Comparison:** in order to confirm the effectiveness of the proposed approach, a comparison with the following algorithms has been carried out, in terms of C-index and compu-

tational cost: in particular, the traditional Kernel  $k$ -means and two improved versions, the Global Kernel  $k$ -means and the Fast Global Kernel  $k$ -means [18], are considered. A standard implementation of the Kernel  $K$ -means has been used and can be found in [35], while as for the other two considered methods, the code has been made available by the authors. It is worth pointing out that a very important disadvantage of the Kernel  $k$ -means lies in the fact that the performance are also influenced by the random initialization needed to initiate the system. For this reason, in order to limit the dependency of the results from the particular initialization, all the results presented in this section are obtained by taking the minimal C-index and the corresponding time over 300 different trials.

The results for the different datasets are shown in Table ?? . In particular, we fixed the threshold to 0.5 for the proposed method and the so obtained numbers of clusters have been considered as input for the other state-of-the-art approaches.

Even if the proposed method is slower than the traditional Kernel  $k$ -Means and the Fast Global Kernel  $k$ -Means, it clearly outperforms both of these in terms of C-Index over all the three considered datasets. It is worth pointing out that in general the low performance of the above mentioned approaches is mainly due to the so-called problem of the *curse of dimensionality* [36]: when the dimensionality increases, the volume of the space increases so fast that the available data becomes sparse. It implies that the amount of data needed to obtain a statistically sound and reliable result grows exponentially with the dimensionality. Furthermore, also the distance functions lose their usefulness: as a matter of fact, the distance between any two points in a given dataset converges and then the discrimination of the nearest and farthest point in particular becomes meaningless. This issue can be overcome by reducing the dimensionality of the space [37]: the PCA is applied and only the first  $N$  components, being  $N$  the number of clusters, are considered. Finally, the traditional  $k$ -means is used in order to find out the clusters. Although this approach reveals to be better than the Kernel  $k$ -Means, the proposed method still outperforms it, both in terms of C-index and time needed to perform the clustering. This is mainly due to the fact that the proposed method reduces the dimensionality of the data without paying it in terms of information loss: as a matter of fact, at each iteration all the information pertaining a given cluster are analyzed in order to find its major axis.

#### D. Anomaly Detection: experimental results

The classification algorithm has been tested over two of the considered datasets, namely the MIT Trajectory dataset and the Edinburgh Informatics Forum Pedestrian Database. The absence of the MIT Train Station dataset is mainly due to the fact that this step requires a preliminary labeling of the trajectories into normal and abnormal, which makes sense only if abnormal trajectories can be identified in the dataset.

In both cases, given the entire dataset  $D$ , the normal trajectory dataset  $D^*$  has been randomly partitioned 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. Finally, a cross-validation strategy has been adopted in order to obtain the results presented in Table II and further detailed in the Electronic Annex (Appendix D). These results present the area under curve (AUC) of the Receiver Operating Characteristic (ROC) curves computed on the on the MIT Trajectory dataset and the Edinburgh Informatics Forum Pedestrian Database using both our traditional Dirac Kernel ( $k_{ZSS}$ ) and Weighted Dirac kernel ( $k_{WZSS}$ ) with an increasing number of zones (from 10 to 60).

Starting from the obtained results, the following consideration can be done: the number of zones  $|Z|$  does not strongly influence the performance of the proposed system. As a matter of fact, we can note that by using a Dirac Kernel in the MIT dataset (Table II, line 1 and Figure D.1(a) of the Electronic Annex) we achieve comparable performance with  $|Z|$  ranging from 10 to 40; the performance decreases with an higher number of zones ( $|Z|$  ranging from 50 to 70). This is mainly due to the fact that the system in this case pays in terms of generalization and it is not able to correctly classify those normal trajectories only slightly different from the ones included in the dataset. This consideration is also confirmed by the result obtained on the Edinburgh Trajectories Dataset (Table II, line 2 and Figure D.2(a) of the Electronic Annex), where the optimum number of zones ranges from 30 to 50.

Furthermore, we can note that in this case a low number of zones ( $|Z| = 10$ ) does not guarantee good performance. This is mainly due to the different trajectories' distribution of the two considered datasets. In fact we observe in Figure 8 that the trajectories in the MIT trajectories dataset are located in a limited area of the image and are much more compact than in the Edinburgh Informatics Pedestrian Database, where people can freely move along the entire image. For this reason, although the proposed scene partitioning algorithm is able to optimize the partition of the space, the limited number of zones seems to not give enough informative content because of the homogeneous distribution of the dataset.

Finally, we can note that the introduction of the Weighted Dirac Kernel only slightly improves the performance in both the considered datasets. However, its main advantage lies in the fact that this similarity evaluation is less sensitive to the different number of zones thanks to its nature: as a matter of fact, it also guarantees good performance with a low, as well as an high number of zones, since it also considers the proximity of zones. Furthermore, it is able to provide a better generalization than the Dirac Kernel, without paying in terms of false positive errors. This consideration is confirmed by the analysis of the AUCs, reported in Table II: the improvement between the best and the worst case by using the Dirac Kernel is higher than 20%; for instance, for the MIT Trajectories dataset, where we obtain  $\frac{0.88-0.67}{0.88} = 0.23$ . On the other hand, it is lower than 7% by using the Weighted Dirac Kernel, where, for instance, we obtain on the same dataset  $\frac{0.92-0.86}{0.92} = 0.06$ .

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; as we will show in the following, most of the errors can be discussed, being strongly related to ambiguous interpretations of the trajectories also for a human operator. An example is shown in Figure 11a: the trajectory in red is labeled as abnormal in the ground truth, since it refers to a vehicle's trajectory partially located in the grass (or to an error of the tracking phase as well); our method, as well as any other kinds of methods based on shape and position similarities, has no chance to give a correct answer and classifies such a trajectory as abnormal, since it is very similar to those normal which avoid the grass just for a few centimeters. Only the introduction of areas boundaries could make the system able to provide a correct answer by boundary cross detection.

Similar situation occurs in Figure 11b, where the vehicle tries to park, but because of place lack, leaves out after a complete turn. In this case, the description of the trajectory, manually labeled as abnormal, follows a regular and normal path, except for a very limited stretch, reproducing the same typology of error occurring in the previous case.

In general, it is worth pointing out that it is a really difficult task, also for a human operator, to distinguish normal and abnormal trajectories: in fact, both the above mentioned behaviors are ambiguous and could be considered normal as well as abnormal. For instance, the last situation could be considered an error of the tracking phase and could be labeled as normal also by a human operator.

Opposite kinds of error occur in Figures 11c and 11d. In this case, the two trajectories are manually labeled as normal with respect to their semantic, but can also refer to tracking errors because of their very short lengths. The system in such a situations has not sufficient information and then is not able to reliably associate the two trajectories to any cluster containing normal trajectories.

In conclusion the performance, yet acceptable for many practical application, can be considered even better at the light of the above considerations.

#### E. Query by sketch: experimental results

In order to confirm the effectiveness of the proposed method, two different evaluations will be provided in this section, a quantitative and a qualitative one. Both these evaluations have been performed over the three datasets described in Section VI-A.

The quantitative evaluation has been carried out by evaluating the improvement of the performance if compared with a brute-force approach. Among the entire datasets, 3000 trajectories have been randomly selected from each dataset for testing and the remaining have been used for training. In particular, the number of the trajectories used to build the  $k$ -d tree has been progressively increased in order to verify the performance of the proposed method with different size, as shown in Figure 12. A  $k$  nearest neighbor search is performed and the number of trajectories analyzed by the algorithm, expressed in percentage with respect to the size of the considered dataset, have been evaluated. The results are

shown in Figure 12: as evident, the improvement is much more significant with big datasets: for instance, if we are interested in discovering the most similar trajectory ( $k = 1$ ) in a dataset composed by 3000 trajectories, we only have to analyze 24% of it (approximately 700 trajectories) in the MIT Trajectories dataset (Figure 12a) and 11% of it (approximately 300 trajectories) in the Edinburgh and the MIT Train Station dataset (Figure 12b), resulting in a significant improvement of the overall performance.

Furthermore, a simple Graphical User Interface has been designed in order to allow the user to solve a query by sketch. The user simply draw a trajectory and chose the number of trajectories (the  $k$  value) he is interested in. Examples, one for each considered dataset, are provided in Figure 13. In particular, the first row shows the trajectories sketched by the user for the different datasets, while in the second one the result of the algorithm is depicted. The obtained results highlight the effectiveness of the proposed method over very different datasets, and then over different application fields.

## VII. CONCLUSIONS

We propose an unsupervised method able to deduce properties of a scene from a set of normal trajectories. Starting from a set of normal trajectories acquired by means of a video analysis system, our method represents 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. Furthermore, two different applications have been proposed: the former aims at identifying abnormal behaviors, while the latter allows to solve the queries by sketch by extracting the  $k$  most similar trajectories to the one hand-drawn by the human operator. Experiments have been performed over three different 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](http://www.aitech-solutions.eu)) and by the Università Italo Francese - Université Franco Italienne (UIF - UFI) (<http://www.universite-franco-italienne.org>).

## REFERENCES

- [1] B. Morris and M. Trivedi, "A survey of vision-based trajectory learning and analysis for surveillance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1114–1127, 2008.
- [2] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," *ACM Comput. Surv.*, vol. 43, no. 3, pp. 16:1–16:43, Apr. 2011.
- [3] A. d'Acerno, 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.

- [4] A. Sodemann, M. Ross, and B. Borghetti, "A review of anomaly detection in automated surveillance," *IEEE Trans. Syst. Man, Cybern. C*, vol. 42, no. 6, pp. 1257–1272, 2012.
- [5] L. Brun, A. Saggese, and M. Vento, "A clustering algorithm of trajectories for behaviour understanding based on string kernels," in *Proceedings of the 2012 SITIS Conference*. IEEE, 2012, pp. 267–274.
- [6] —, "Learning and classification of car trajectories in road video by string kernels," in *VISAPP*, 2013.
- [7] R. Di Lascio, P. Foggia, G. Percannella, A. Saggese, and M. Vento, "A real time algorithm for people tracking using contextual reasoning," *Comput Vis Image Und*, 2013.
- [8] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1450–1464, 2006.
- [9] S. Atev, G. Miller, and N. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 647–657, sept. 2010.
- [10] *Similarity based vehicle trajectory clustering and anomaly detection*, vol. 2, 2005.
- [11] R. R. Sillito and R. B. Fisher, "Semi-supervised learning for anomalous trajectory detection," in *BMVC*, M. Everingham, C. J. Needham, and R. Fraile, Eds. British Machine Vision Association, 2008.
- [12] A. Prati, S. Calderara, and R. Cucchiara, "Using circular statistics for trajectory shape analysis," in *CVPR*, june 2008, pp. 1–8.
- [13] J.-W. Hsieh, S.-L. Yu, and Y.-S. Chen, "Motion-based video retrieval by trajectory matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, pp. 396–409, 2006.
- [14] U. Gaur, B. Song, and A. Roy-Chowdhury, "Query-based retrieval of complex activities using strings of motion-words," in *Motion and Video Computing, 2009. WMVC '09. Workshop on*, dec. 2009, pp. 1–8.
- [15] H.-Y. Cheng and J.-N. Hwang, "Integrated video object tracking with applications in trajectory-based event detection," *J. Vis. Commun. Image Represent.*, vol. 22, no. 7, pp. 673–685, Oct. 2011. [Online]. Available: <http://dx.doi.org/10.1016/j.jvcir.2011.07.001>
- [16] B. Morris and M. Trivedi, "Learning trajectory patterns by clustering: Experimental studies and comparative evaluation," in *CVPR*, 2009, pp. 312–319.
- [17] G. K. D. De Vries and M. Van Someren, "Machine learning for vessel trajectories using compression, alignments and domain knowledge," *Expert Syst. Appl.*, vol. 39, no. 18, pp. 13 426–13 439, Dec. 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.eswa.2012.05.060>
- [18] 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.
- [19] 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, no. 3, pp. 287–312, Dec. 2011. [Online]. Available: <http://dx.doi.org/10.1007/s11263-011-0459-6>
- [20] 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.
- [21] J. P. Braquelair 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.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] M. Neuhaus and H. Bunke, "Edit distance-based kernel functions for structural pattern classification," *Pattern Recognition*, vol. 39, no. 10, pp. 1852 – 1863, 2006.
- [26] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui, "A kernel for time series based on global alignments," *CoRR*, vol. abs/cs/0610033, 2006.
- [27] M. Cuturi, "Fast global alignment kernels," in *ICML*, L. Getoor and T. Scheffer, Eds. New York, NY, USA: ACM, June 2011, pp. 929–936.
- [28] 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.
- [29] B. Haasdonk and E. Pekalska, "Classification with kernel mahalanobis distance classifiers," in *GfKI*, 2008, pp. 351–361.
- [30] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975. [Online]. Available: <http://doi.acm.org/10.1145/361002.361007>
- [31] 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.
- [32] B. Majecka, "Statistical models of pedestrian behaviour in the Forum," Ph.D. dissertation, School of Informatics, University of Edinburgh, 2009.
- [33] B. Zhou, X. Wang, and X. Tang, "Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents," in *CVPR*, 2012, pp. 2871–2878.
- [34] 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.
- [35] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.
- [36] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 1, pp. 1:1–1:58, Mar. 2009. [Online]. Available: <http://doi.acm.org/10.1145/1497577.1497578>
- [37] C. Ding and X. He, "K-means clustering via principal component analysis," in *ICML*. ACM, 2004, pp. 29–.

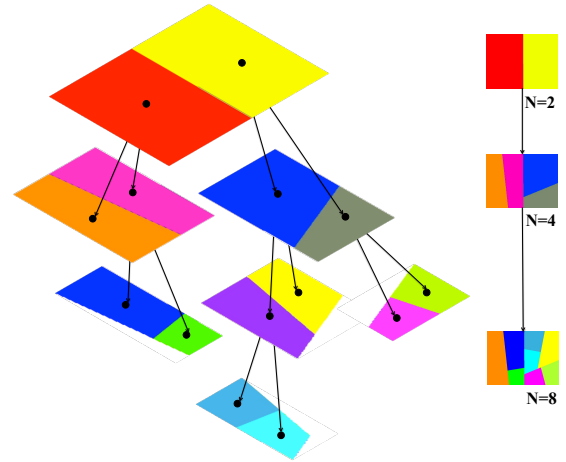


Fig. 2: Example of a tree-structured vector quantization obtained recursively partitioning the scene into  $L = 8$  zones.

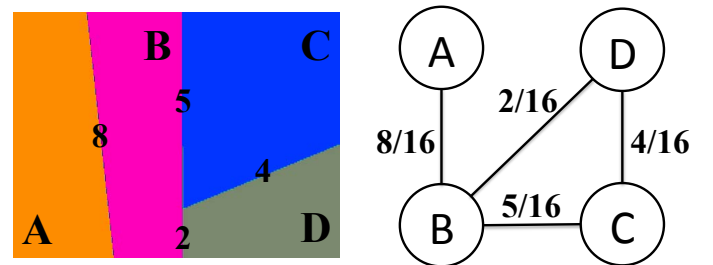


Fig. 4: Graph-based representation of the scene. Each zone is represented by a vertex and each border between two zones is represented by an edge. The weight of each edge is determined by the length of the corresponding border.

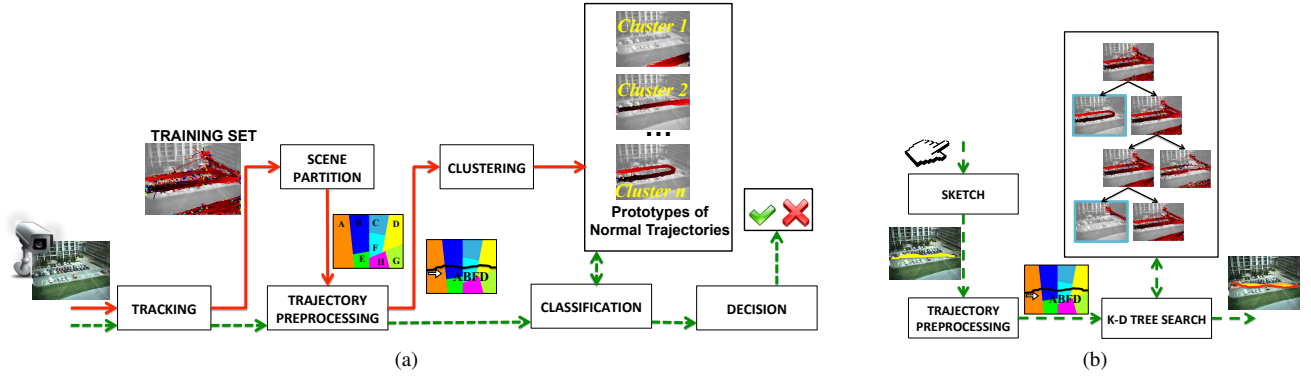


Fig. 1: Learning phase (red solid arrows) and operating phase (green dotted arrows) for abnormal recognition (a) and for queries by sketch (b).

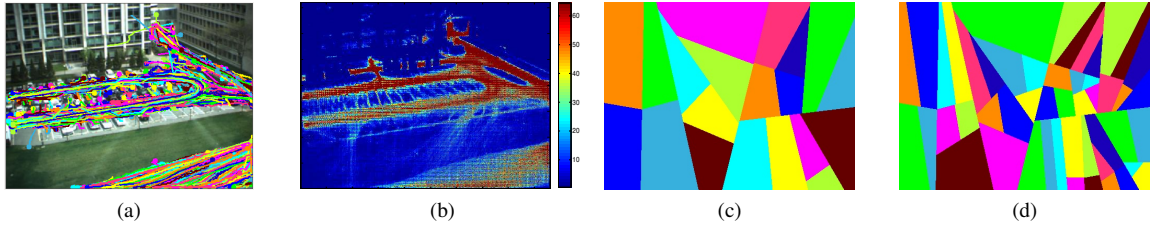


Fig. 3: Partition of the scene starting from the training set depicted in (a) represented by the frequency map in (b). The quantization algorithm is applied with different values of  $L$ : (c)  $L=20$ , (d)  $L=50$ .

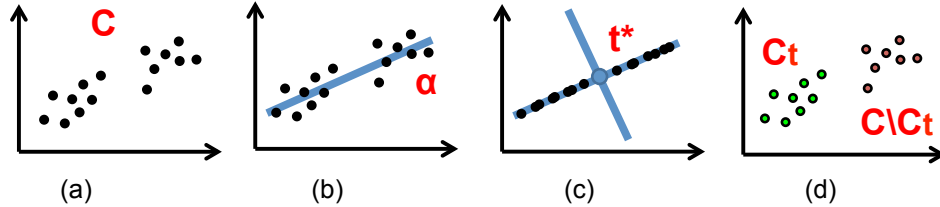


Fig. 5: Simple example of the proposed clustering algorithm: once selected the cluster  $C$  (a), the cutting axis and the cutting position are computed ((b) and (c) respectively) and the new clusters  $C_t$  and  $(C \setminus C_t)$  are obtained.

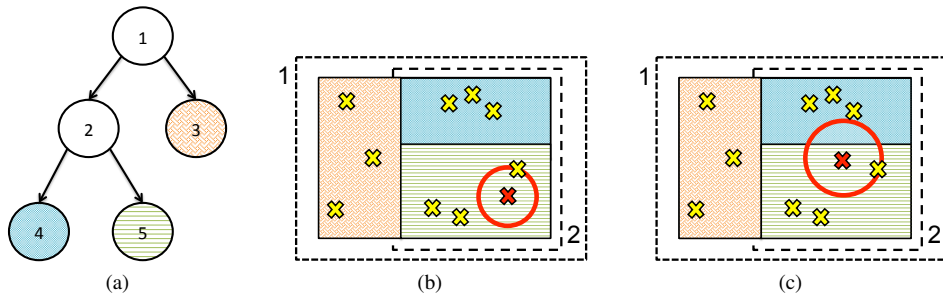


Fig. 6: A tree (a) represented by its geometric boundaries (b)(c). In (b) the *bob* test has success, while in (c) it fails.

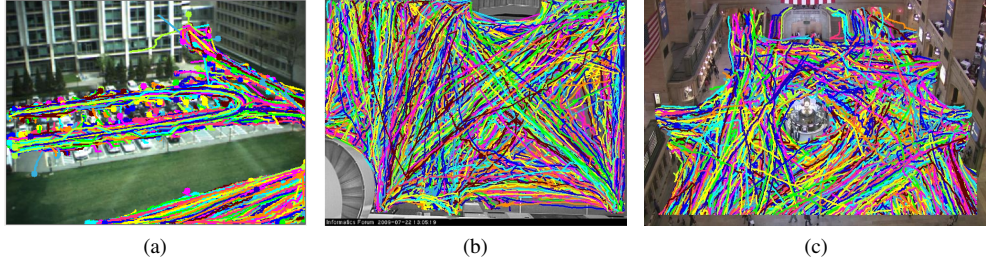


Fig. 8: MIT Trajectory dataset (a), Edinburgh Informatics Forum Pedestrian Database (b) and MIT Train Station dataset (c).

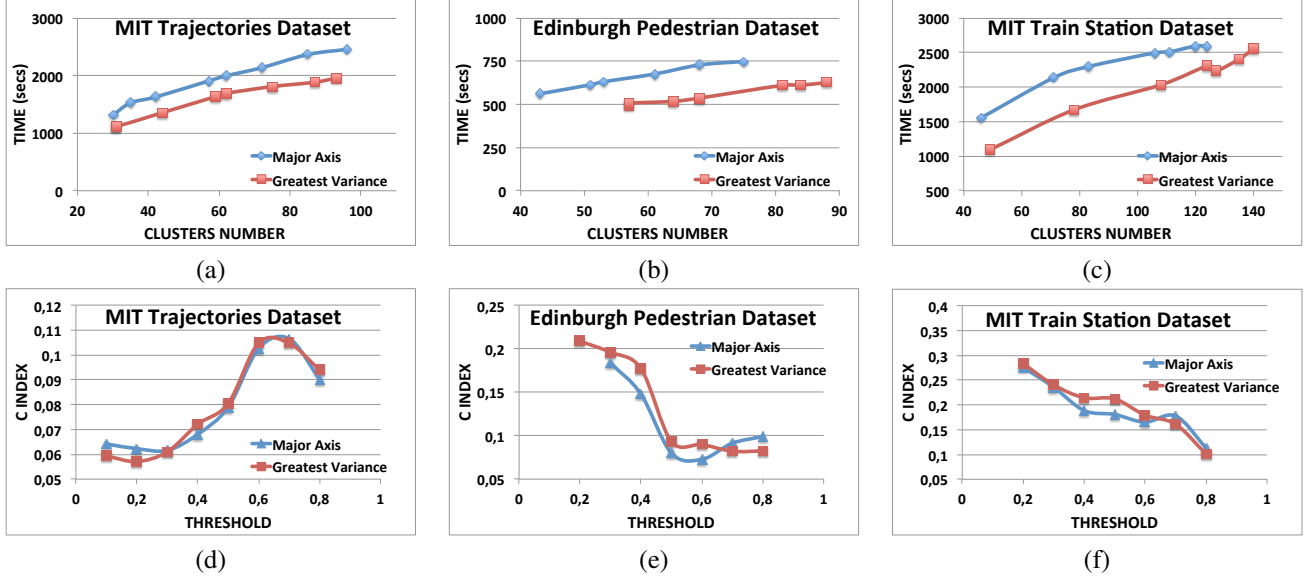


Fig. 9: Performance of the proposed clustering algorithm for the different datasets (MIT Trajectory Dataset (a,d), Edinburgh Informatics Forum Pedestrian Database (b,e) and MIT Train Station dataset (c,f)) in terms of computational costs and c-index.

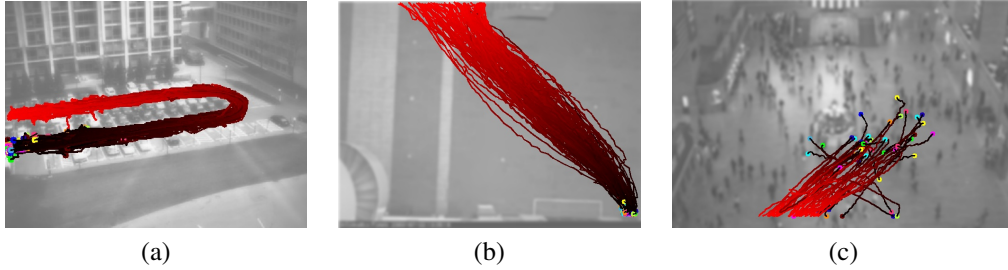


Fig. 10: A cluster for the three considered datasets: MIT Trajectory Dataset (a), Edinburgh Informatics Forum Pedestrian Database (b) and MIT Train Station dataset (c).

|                     | 10 zones | 20 zones    | 30 zones    | 40 zones    | 50 zones | 60 zones |
|---------------------|----------|-------------|-------------|-------------|----------|----------|
| MIT Dirac           | 0.87     | <b>0.88</b> | 0.85        | 0.87        | 0.74     | 0.67     |
| EDIN Dirac          | 0.72     | 0.73        | 0.84        | <b>0.87</b> | 0.86     | 0.77     |
| MIT Weighted Dirac  | 0.87     | 0.85        | <b>0.92</b> | 0.86        | 0.89     | 0.86     |
| EDIN Weighted Dirac | 0.79     | <b>0.85</b> | <b>0.85</b> | 0.82        | 0.79     | 0.80     |

TABLE II: AUC of the MIT Trajectories and of the Edinburgh Pedestrian datasets obtained by varying the  $\alpha$  parameter and by using the Dirac Kernel and its weighted version.

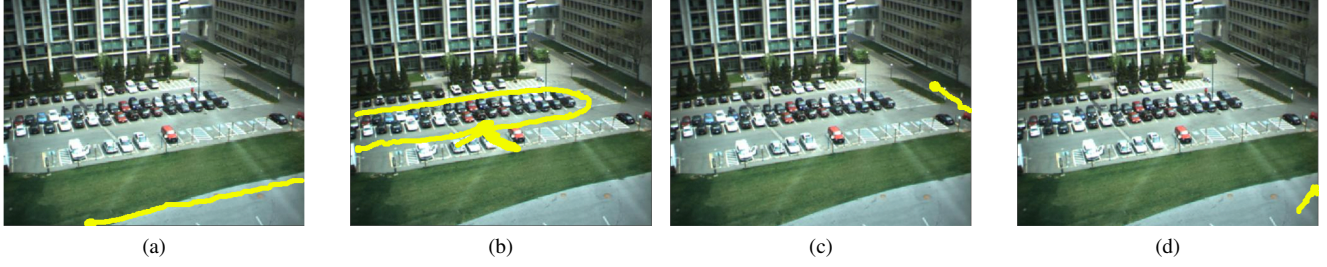


Fig. 11: Abnormal trajectories classified as normal (a) (b) and normal trajectories classified as abnormal (c)(d).

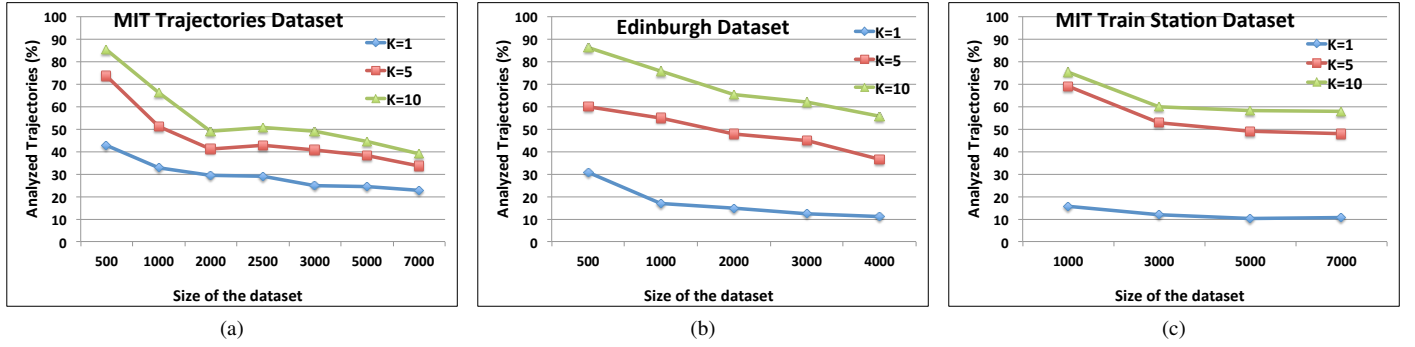


Fig. 12: Query by sketch: percentage of trajectories that need to be analyzed in the MIT Trajectory dataset (a), in the Edinburgh Informatics Forum Pedestrian Database (b) and in the MIT Train Station dataset (c).

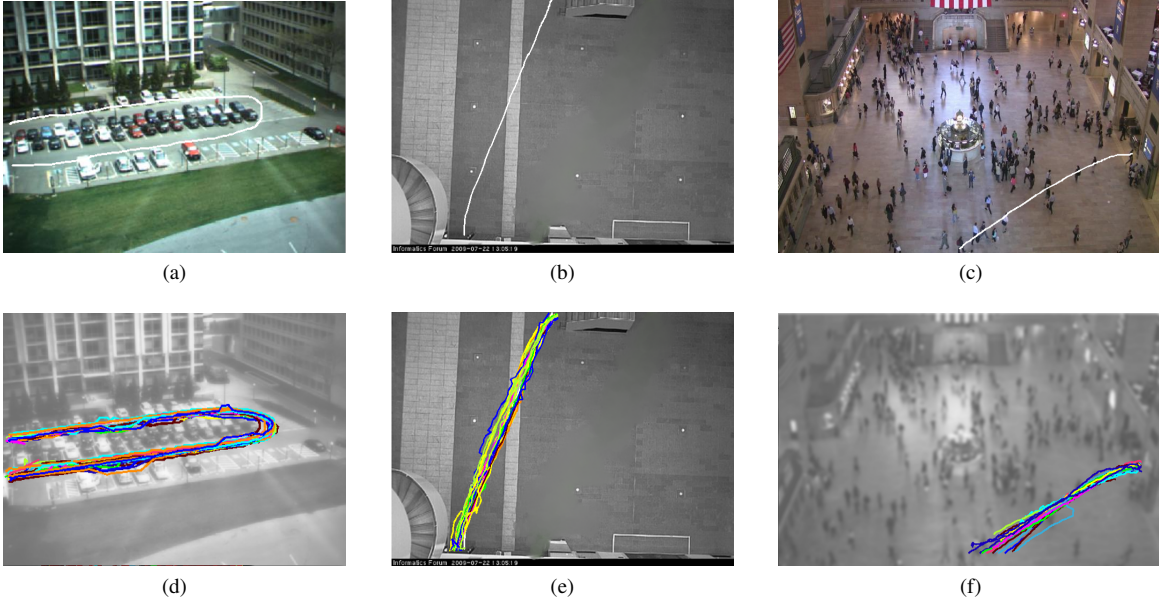


Fig. 13: The trajectories sketched by the user are shown in the first row, while the obtained results (with  $k = 20$ ) are in the second row for the different datasets: MIT Trajectory Dataset (a,d), Edinburgh Informatics Forum Pedestrian Database (b,e) and MIT Train Station dataset (c,f).

# Dynamic Scene Understanding for behavior analysis based on string kernels

## ELECTRONIC ANNEX

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

Alessia Saggese and Mario Vento  
Dept. of Information Eng., Electrical Eng. and Applied Mathematics  
University of Salerno  
I - 84084 Fisciano (SA), Italy  
{asaggese, mvento}@unisa.it

### Abstract

This document is the electronic annex of the paper *Dynamic Scene Understanding for behavior analysis based on string kernels*, by Luc Brun, Alessia Saggese and Mario Vento.

### APPENDIX

In this section we summarize the complexity analysis of the proposed clustering algorithm. As for the *cluster selection* step, the algorithm need to find the cluster with the maximum squared error, so resulting (in the worst case) in a complexity of  $O(N/2)$ , being  $N$  the number of clusters built until the current iteration. As a matter of fact, the squared error of a new cluster  $C_t$  can be simply computed during its creation without additional costs: in particular the second terms  $(\Lambda_{C_t}^T K \Lambda_{C_t})$ , which is the most expensive one, is evaluated during the computation of the cutting position, as shown in Equation 21 of the paper. On the other hand, the squared error of  $(C \setminus C_t)$  can be simply computed as the difference between  $SE(C)$  and  $SE(C_t)$ .

As for the *cutting axis* computation, it mainly requires two different steps: the computation of the centered matrix and the computation of the first eigenvector. In both cases, the operations can be evaluated in  $O(|C|^{2.376})$  using the Coppersmith Winograd algorithm [1]. Furthermore, the sort of trajectories' projections on the cutting axis is performed using an Heapsort algorithm, which requires  $O(|C| \log |C|)$  in the worst case.

Finally, the computation of the *cutting position* only requires  $O\left(\frac{|C| \cdot (|C| + 1)}{2}\right)$ : in fact, we need to compute, for each trajectory  $p$  of  $C$ , the sum of similarities between  $p$  and all trajectories belonging to  $C_t$  ( $\sum_{i \in C_t} k(i, p)$ ). Since the size of  $C_t$  increases by 1 at each iteration, the complexity of this step is equal to the sum of integers up to  $|C|$ .

In other words, each iteration of the algorithm can be performed in

$$O(N) + O(|C|^{2.376}) + O(|C| \log |C|) + O\left(\frac{|C| \cdot (|C| + 1)}{2}\right) = O(|C|^{2.376}). \quad (1)$$

Let be:

$$b_s^i = \begin{cases} 1 - \frac{1}{|C|} & \text{if } i = s \\ -\frac{1}{|C|} & \text{if } i \in C \wedge i \neq s \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$\delta_s^i = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$b_s = -\frac{1}{|C|} \Lambda_C + \delta_s. \quad (4)$$

$$\begin{aligned}
SE(C) &= \sum_{s \in C} \|\Psi_s - \mu_t\|^2 \tag{5} \\
&= \sum_{s \in C} \left\| \Psi_s - \frac{1}{|C|} \sum_{i \in C} \Psi_i \right\|^2 \\
&= \sum_{s \in C} \left\| \sum_{i \in C} b_s^i \Psi_i \right\|^2 \\
&= \sum_{s \in C} \sum_{i, j \in C} b_s^i b_s^j \langle \Psi_i, \Psi_j \rangle \\
&= \sum_{s \in C} b_s^t K b_s \\
&= \sum_{s \in C} \left( -\frac{1}{|C|} \Lambda_C + \delta_s \right)^t K \left( -\frac{1}{|C|} \Lambda_C + \delta_s \right) \\
&= \sum_{s \in C} \frac{1}{|C|^2} \Lambda_C^t K \Lambda_C - \frac{2}{|C|} \Lambda_C^t K \delta_s + \delta_s K \delta_s \\
&= \frac{1}{|C|} \Lambda_C^t K \Lambda_C - \frac{2}{|C|} \sum_{s \in C} \Lambda_C^t K \delta_s + \sum_{s \in C} k(s, s) \\
&= \frac{1}{|C|} \Lambda_C^t K \Lambda_C - \frac{2}{|C|} \Lambda_C^t K \left( \sum_{s \in C} \delta_s \right) + \sum_{s \in C} k(s, s) \\
&= -\frac{1}{|C|} \Lambda_C^t K \Lambda_C + \sum_{s \in C} k(s, s) \\
&= |C| - \frac{1}{|C|} \Lambda_C^t K \Lambda_C
\end{aligned}$$

Let be:

$$\mu = \frac{1}{|C|} \sum_{i \in C} \Psi_i \tag{6}$$

$$\mu_t = \frac{1}{|C_t|} \sum_{i \in C_t} \Psi_i = \frac{1}{|C_t|} \sum_{i \in C} \Lambda_{C_t}^i \Psi_i. \tag{7}$$

$$a^i = \frac{\Lambda_C^i}{|C|} - \frac{\Lambda_{C_t}^i}{|C_t|}; \tag{8}$$

$$a = \frac{\Lambda_C}{|C|} - \frac{\Lambda_{C_t}}{|C_t|}. \tag{9}$$

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

$$\begin{aligned}
\|\mu - \mu_t\|^2 &= \left\| \sum_{i \in T} \frac{\Lambda_C^i \Psi_i}{|C|} - \frac{\Lambda_{C_t}^i \Psi_i}{|C_t|} \right\|^2 \tag{10} \\
&= \left\| \sum_{i \in T} \left( \frac{\Lambda_C^i}{|C|} - \frac{\Lambda_{C_t}^i}{|C_t|} \right) \Psi_i \right\|^2 \\
&= \left\| \sum_{i \in T} a^i \Psi_i \right\|^2 \\
&= \sum_{i, j \in T} a^i a^j \langle \Psi_i, \Psi_j \rangle \\
&= \sum_{i, j \in T} a^i a^j \cdot k(s_i, s_j) \\
&= a^t K a \\
&= \left( \frac{\Lambda_C}{|C|} - \frac{\Lambda_{C_t}}{|C_t|} \right)^t K \left( \frac{\Lambda_C}{|C|} - \frac{\Lambda_{C_t}}{|C_t|} \right) \\
&= \frac{1}{|C|^2} \Lambda_C^t K \Lambda_C - \frac{2}{|C_t| |C|} \Lambda_C^t K \Lambda_{C_t} + \frac{1}{|C_t|^2} \Lambda_{C_t}^t K \Lambda_{C_t}.
\end{aligned}$$

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

$$\begin{aligned}\Lambda_C^t K \Lambda_{C_{t+1}} &= \Lambda_C^t K (\Lambda_{C_t} + \delta_p) \\ &= \Lambda_C^t K \Lambda_{C_t} + \Lambda_C^t K \delta_p \\ &= \Lambda_C^t K \Lambda_{C_t} + \sum_{i \in C} k(i, p).\end{aligned}\tag{11}$$

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

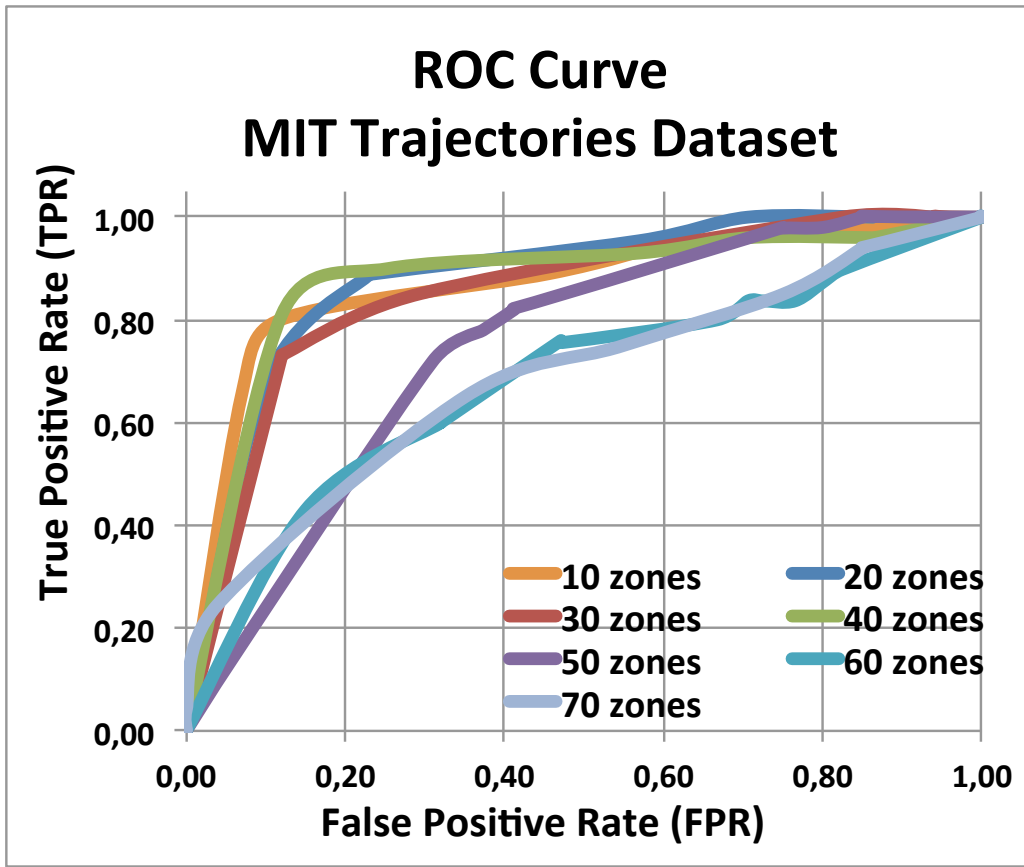
$$\begin{aligned}\Lambda_{C_{t+1}}^t K \Lambda_{C_{t+1}} &= (\Lambda_{C_t} + \delta_p)^t K (\Lambda_{C_t} + \delta_p) \\ &= \Lambda_{C_t}^t K \Lambda_{C_t} + 2 \Lambda_{C_t}^t K \delta_p + \delta_p^t K \delta_p \\ &= \Lambda_{C_t}^t K \Lambda_{C_t} + 2 \sum_{i \in C_t} k(i, p) + k(p, p).\end{aligned}\tag{12}$$

Figures 1 and 2 show ROC curves respectively obtained on the MIT Trajectory dataset and the Edinburgh Informatics Forum Pedestrian Database using a traditional Dirac Kernel (Figures 1a and 2a) and a weighted Dirac kernel (Figures 1b and 2b).

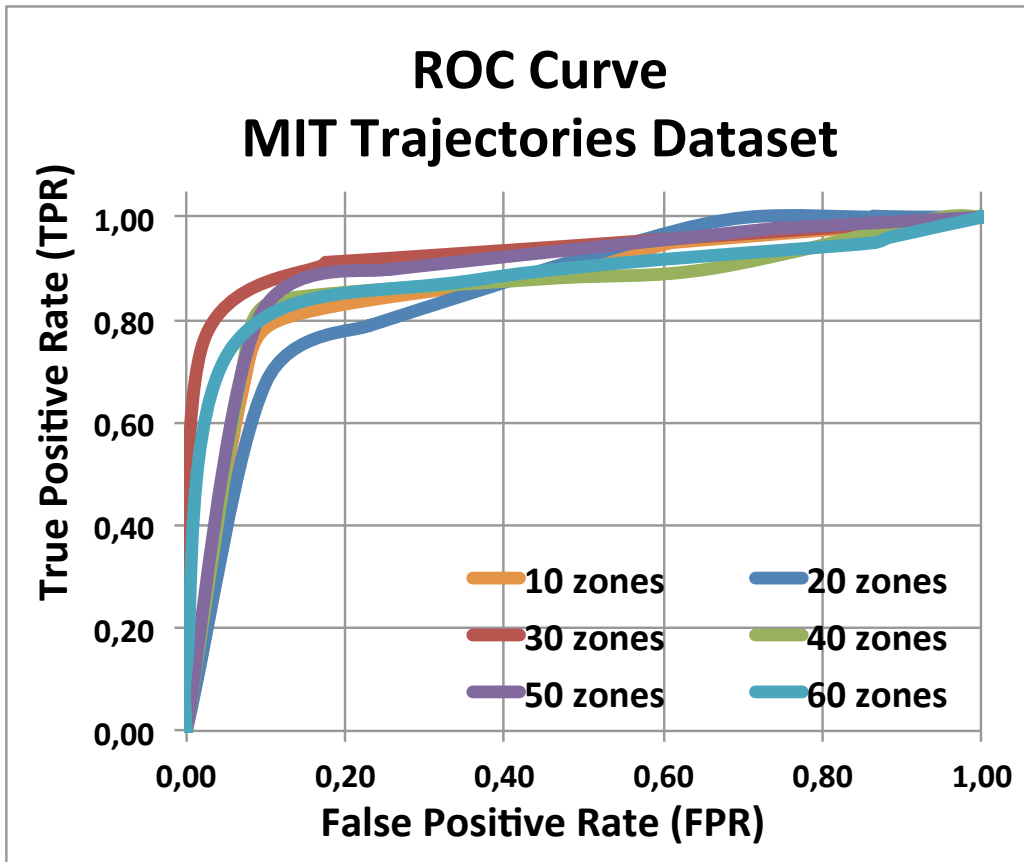
Some of the representative clusters obtained by applying the proposed clustering algorithm over the three considered datasets, MIT Trajectory Dataset, Edinburgh Informatics Forum Pedestrian Database and MIT Train Station dataset, are shown respectively in Figure 3, 4 and 5. The color of each trajectory highlights its direction: it starts in green and progressively turns its color into red.

#### REFERENCES

- [1] J. Demmel, I. Dumitriu, and O. Holtz, "Fast linear algebra is stable," *Numer. Math.*, vol. 108, no. 1, pp. 59–91, Oct. 2007.

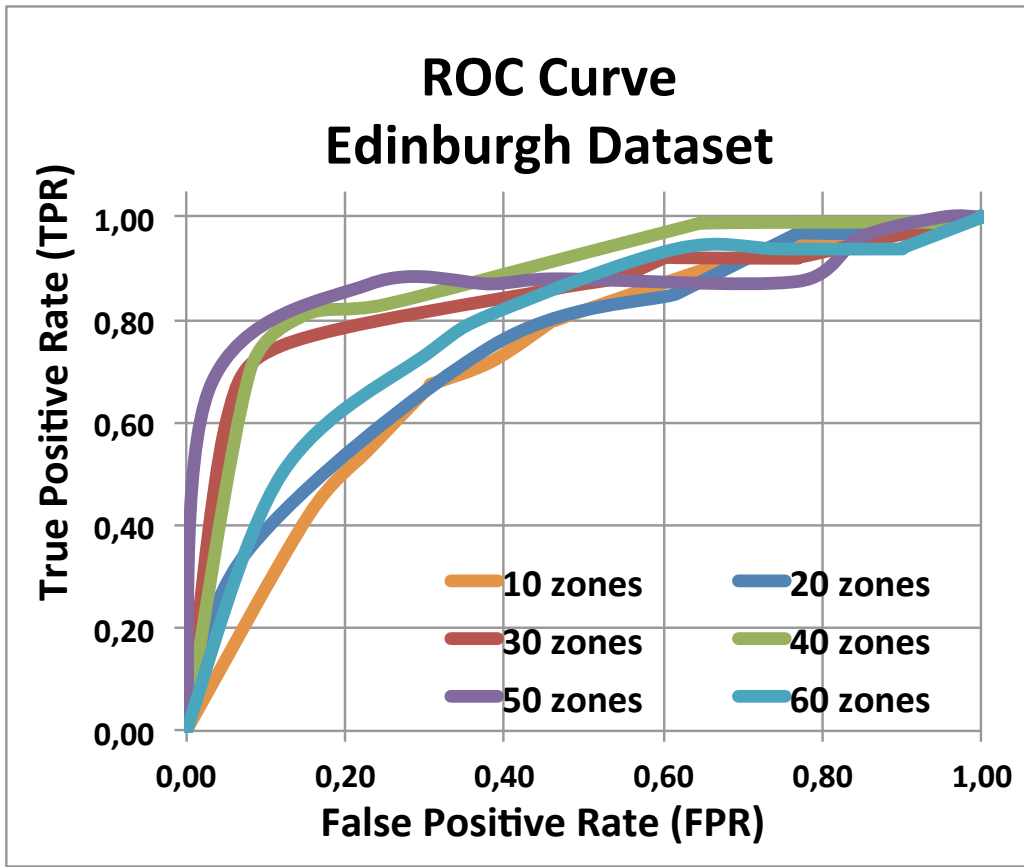


(a)

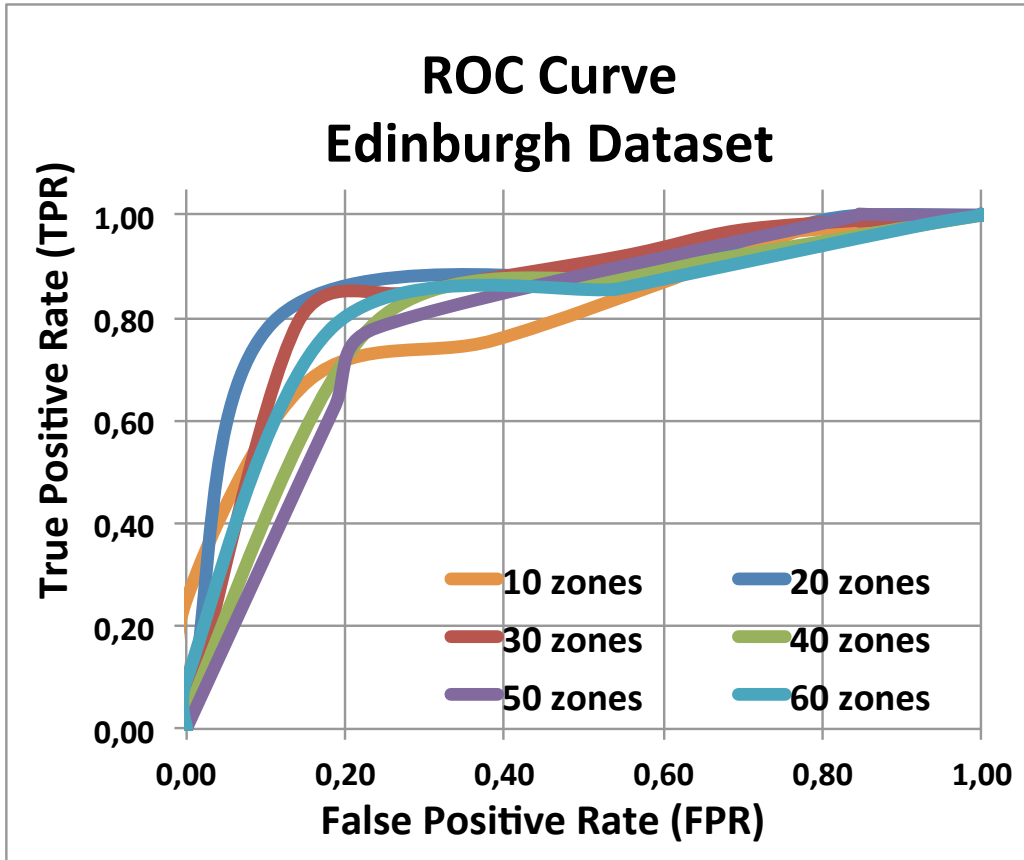


(b)

Fig. 1: ROC Curves of the MIT Trajectories dataset obtained by varying the  $\alpha$  parameter. In (a) the Dirac Kernel has been used, while in (b) its weighted version has been considered.



(a)



(b)

Fig. 2: ROC Curves of the Edinburgh Pedestrian dataset obtained by varying the  $\alpha$  parameter. In (a) the Dirac Kernel has been used, while in (b) its weighted version has been considered.

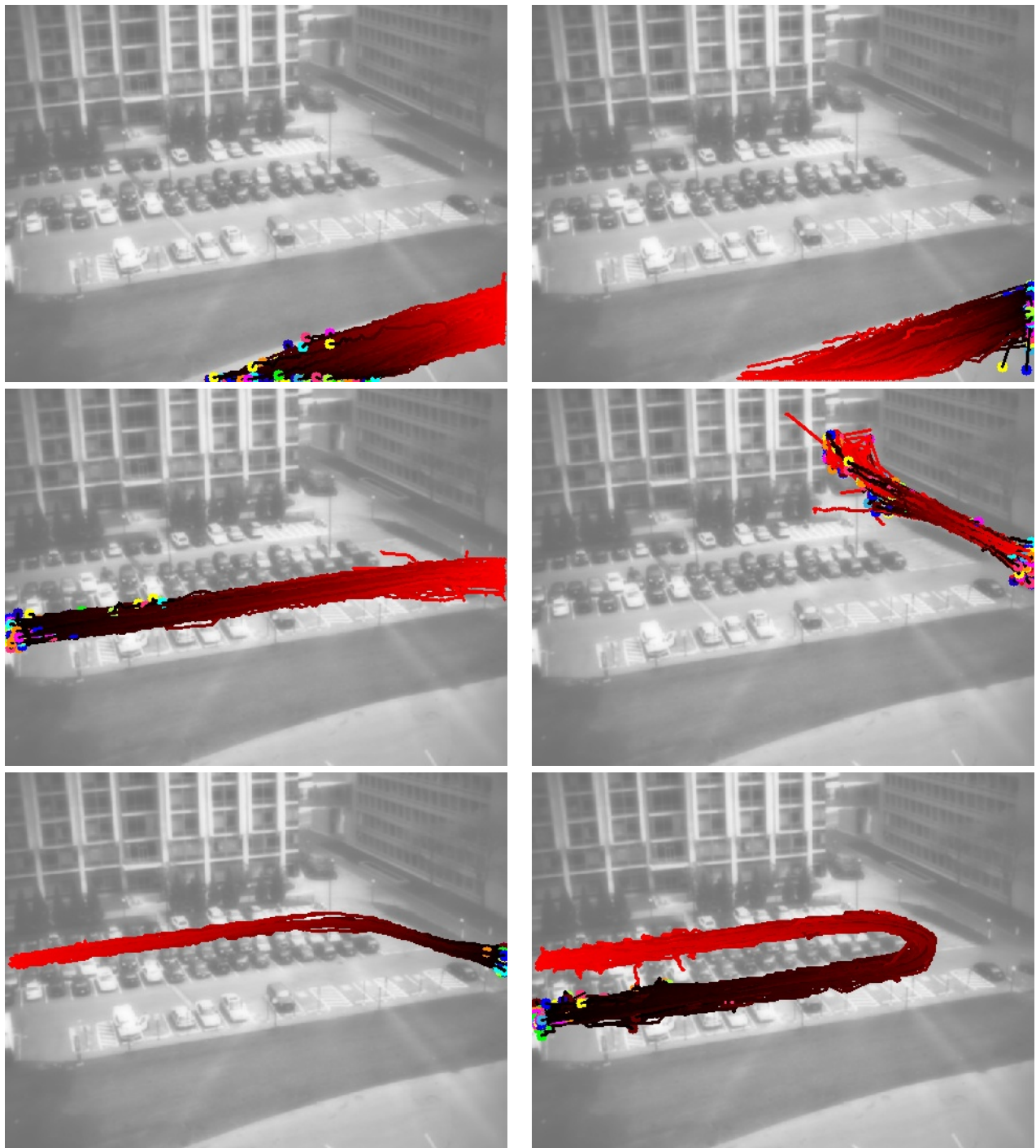


Fig. 3: Some of the representative clusters for the MIT Trajectory Dataset.

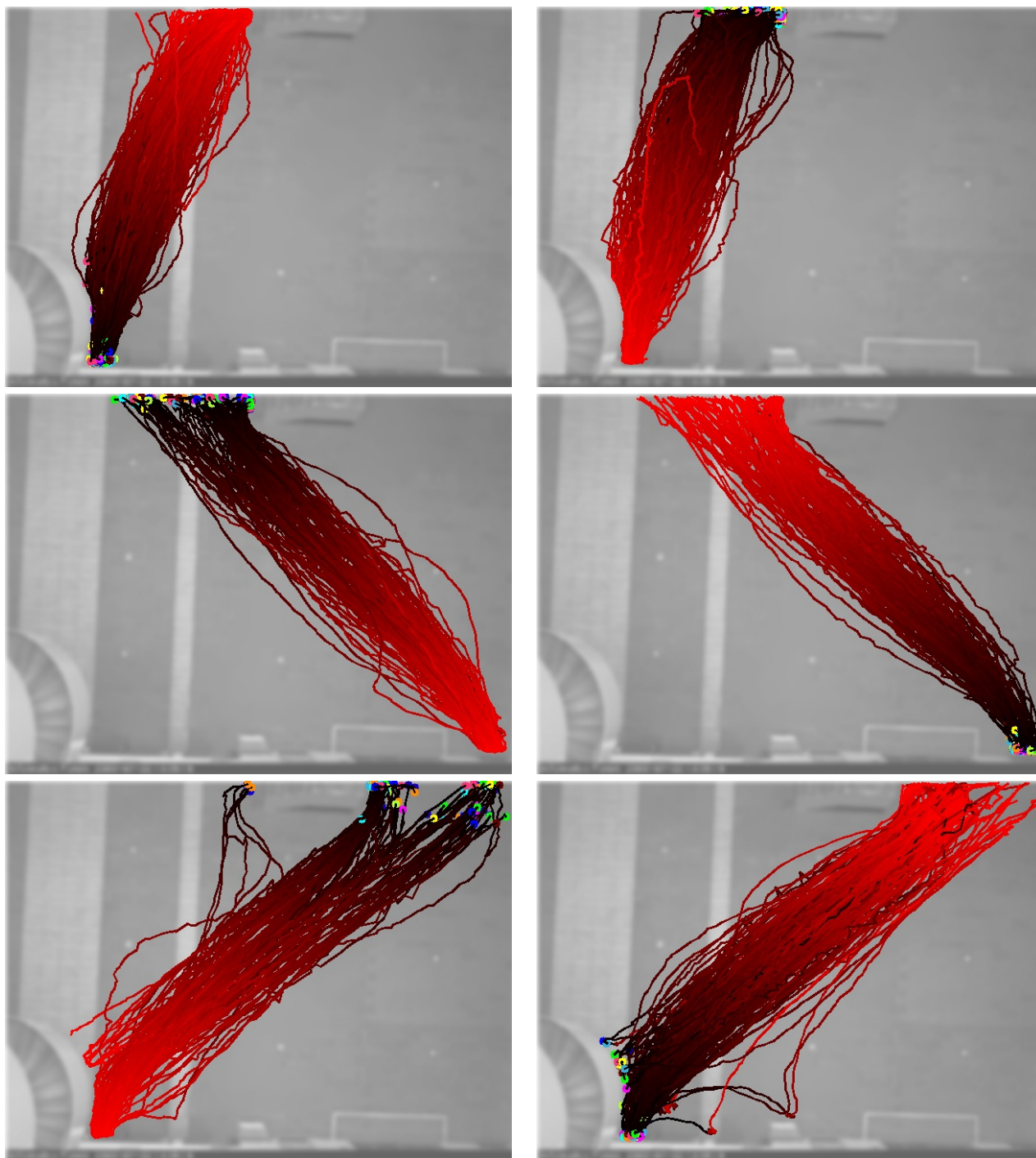


Fig. 4: Some of the representative clusters for the Edinburgh Informatics Forum Pedestrian Database.

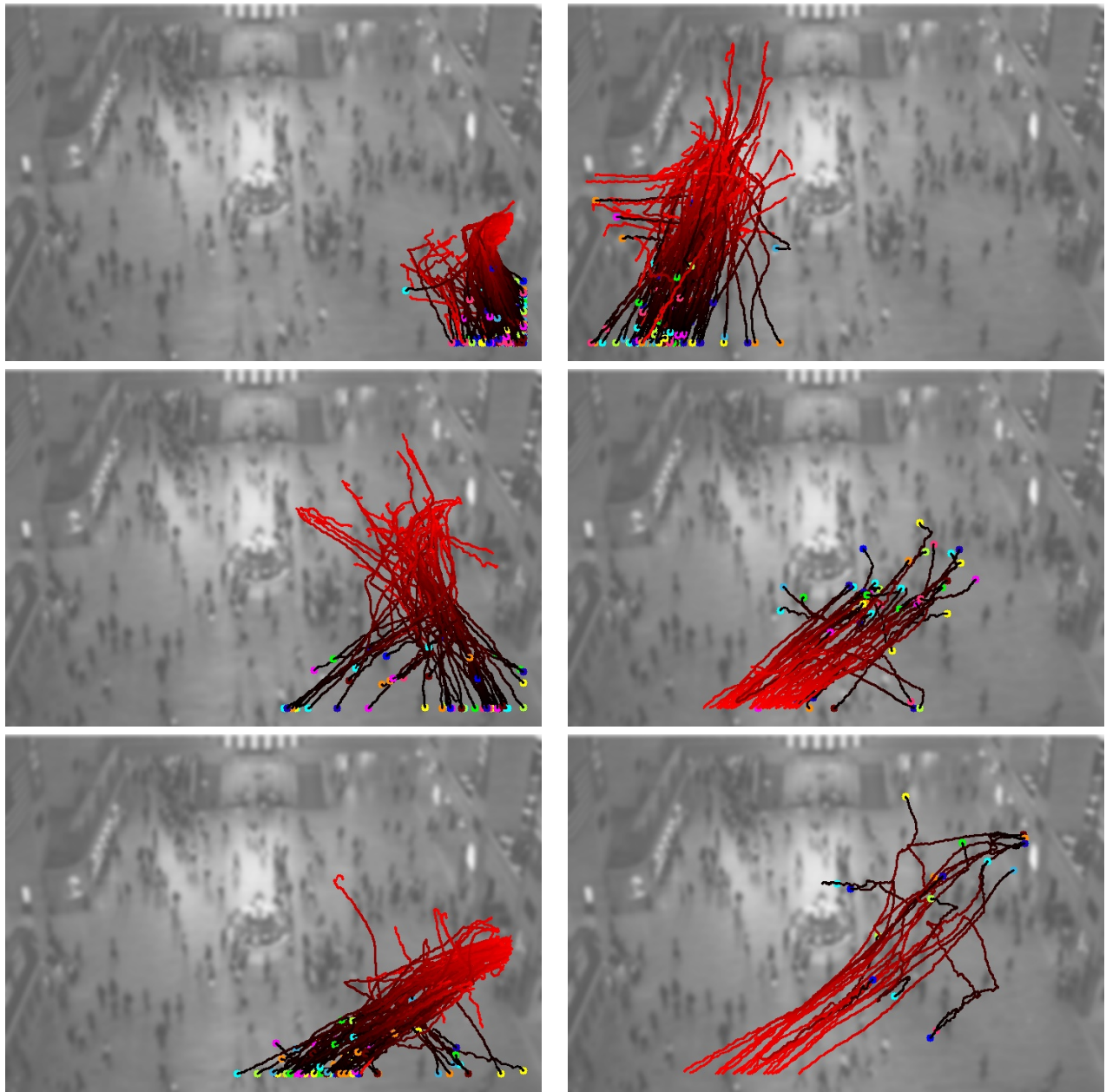


Fig. 5: Some of the representative clusters for the MIT Train Station dataset.