
***Università degli Studi di Salerno***

Dipartimento di Ingegneria dell'Informazione, Ingegneria Elettrica e Matematica Applicata

Dottorato di Ricerca in Ingegneria dell'Informazione
XII Ciclo – NS

Doctorat de l'université de Caen Basse-Normandie

UNIVERSITÀ ITALO FRANCESE


PHD THESIS

# Detecting and indexing moving objects for Behavior Analysis by Video and Audio Interpretation


CANDIDATE: **ALESSIA SAGGESE**

SUPERVISORS: **PROF. MARIO VENTO**
**PROF. LUC BRUN**

COORDINATORS: **PROF. ANGELO MARCELLI**
**PROF. FRÉDÉRIC JURIE**


Academic Year 2012 – 2013

# Contents

*Computers are incredibly fast, accurate and stupid;*
*humans are incredibly slow, inaccurate and brilliant;*
*together they are powerful beyond imagination.*
-Albert Einstein-

# Chapter 1

# Introduction

In the last decades we have assisted to a growing need for security in various public environments. According to a study recently conducted by the European Security Observatory, one half of the entire population is worried about the crime and requires a law enforcement to get more protection.

This consideration has lead the diffusion of cameras and microphones, which are a suitable solution for their relative low cost of maintenance, the possibility of installing them virtually everywhere and, finally, the capability of analyzing more complex events. However, the main limitation of traditional audio-video surveillance systems lies in the so called *psychological overcharge issue* of the human operators responsible for security; this issue causes a decrease in their capabilities to analyze raw data flows from multiple sources of multimedia information: indeed, as stated by a study conducted by Security Solutions magazine, "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" [1].

For these reasons, it would be really useful to design an *intelligent* surveillance system, able to support the human operator in his control task by properly interpreting audio and video signals with the aim of identifying events of interest. Let's think to a train station or to an airport where people are fighting: their

movements become disordered and screams can be heard in the environment. An intelligent system detects this dangerous behavior and alerts the users in charge of monitoring the environment, allowing a prompt intervention of the competent authorities.

It is worth pointing out that a system able to automatically understand events occurring in a scene would be useful even in other application fields, oriented for instance to marketing purposes: in fact, the shopper movement patterns can be captured and properly analyzed by an *intelligent* system in order to take better operational decisions for sales, marketing and staffing and to maximize the customer satisfaction.

This thesis introduces a novel video and audio analysis framework, able to meet the above mentioned needs by analyzing the low-level data acquired by video and audio peripherals and generating high-level information, in terms of events the human operator is interested in. Audio and video information, complementary for their nature, are used and properly combined in order to face different situations: in fact, a video analysis system is not feasible for every context, although it is able to provide detailed information about the scene; think, as an example, to a public toilette where it is obvious impossible to install a camera or to environments with too many obstacles, where a full covering would result in a large expense.

Different typologies of events can be recognized by the proposed system. On the one hand, screams, gunshots and broken glasses are evaluated by interpreting audio data; note that a visual interpretation of such events would be really difficult, also for a human operator. On the other hand, video analysis allows to interpret more complex typologies of events, that can not be analyzed by only using audio inspection: for instance, a person crossing an area of interest or several areas of interests in a given sequence, a vehicle crossing an highway in the wrong side, a person moving in the opposite direction of a crowd or loitering in a forbidden area, the typical paths of person inside a mall and so on.

Another important task of the proposed framework is to store

all the detected events so that the human operator can easily and quickly recover such events if necessary. In this way, the human operator can interact with the proposed system in different ways: on the one hand, he is alerted as soon as a dangerous situation is detected (for instance, a gun shot, a person loitering in a forbidden area or a vehicle crossing a street in the wrong side); on the other hand, he can easily interact with the proposed system for recovering information of interest; a typical example is the following: suppose that an aggression took place at the airport, near gate 10, at 11am. The human operator may be interested in identifying all the persons passing between 10:50am and 11:10am in the gate 10, so to very quickly identifying the person of interest.

The realization of a similar system is as fascinating as challenging; the complexity of the problem can be also confirmed if we analyze the scientific papers of the last years. No systems able to integrate audio and video information for security purposes have been proposed; furthermore, the analysis of behaviors through video as well as audio inspection is a very young field, and the scientific community is very active since a definitive solution to the problem has not been found yet.

This thesis aims to bridge over these considerations: it introduces a novel and efficient system for analyzing behaviors through a proper combination of video and audio analysis, by joining together different knowledge and skills, ranging from pattern recognition and computer vision to databases and signal processing.

## 1.1 Overview

An example of the proposed system at work is shown in Figure 1.1: considering the large amount of data to be managed, the human operators do not see the bag left unattended in one of the monitor. The aim of the proposed system in this case is to identify the anomaly and immediately alert the operators for a prompt intervention. Furthermore, all the acquired data are stored in order to keep knowledge of the abnormal events for future efficient

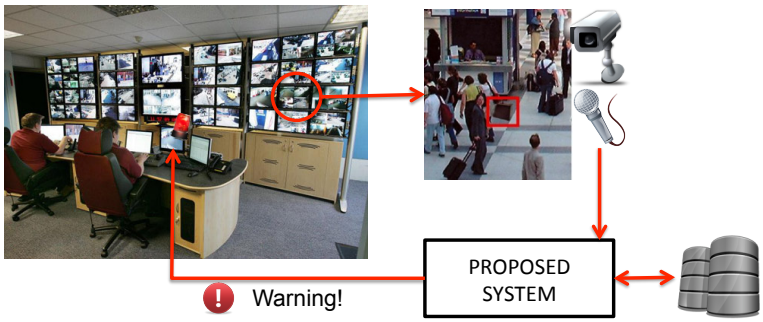



**Figure 1.1** The proposed system at work: as soon as an event of interest (bag left unattended in the airport in the particular example) is detected, the human operator is alerted.

analysis.

The main idea behind the proposed approach lies in the fact that the movement of the people (more in general of the objects) populating a scene is not random, but instead is determined by the associated behaviors. It means that analyzing moving objects trajectories would help in recognizing those events the human operator is interested in.

Figure 1.2 illustrates the overall architecture of the framework proposed in this thesis, whose modules cover the considered requirements. First, the system has to detect objects of interest populating the scene, for instance persons, vehicles or animals, (*Detection*) and extract information about their trajectories as well as their appearance, namely the class (*i.e.* person, bag, vehicle etc.), the real size or the dominant color (*Tracking*).

As soon as an abnormal behavior is detected, an alert needs to be sent to the human operator. As mentioned before, both audio and video information are used and properly combined: on the one hand, the trajectories extracted during the previous tracking phase are analyzed and abnormal ones, associated to dangerous events, are recognized according to a previously learnt set of normal paths (*Visual Behavior Analysis*). On the other hand, audio events of interest, such as gun shots, screams and broken glasses

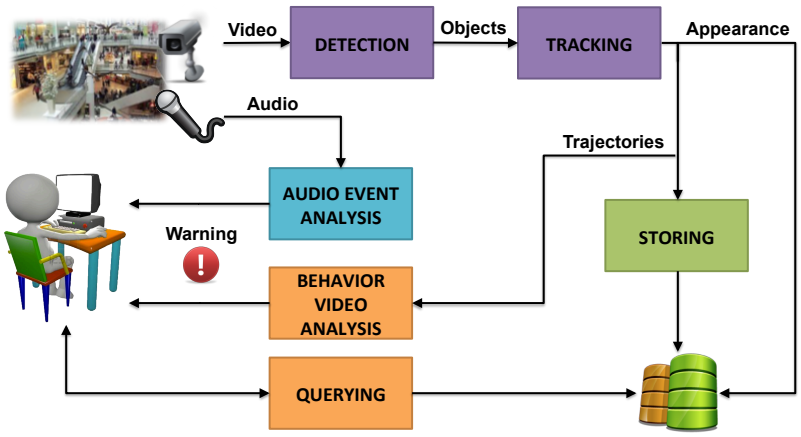


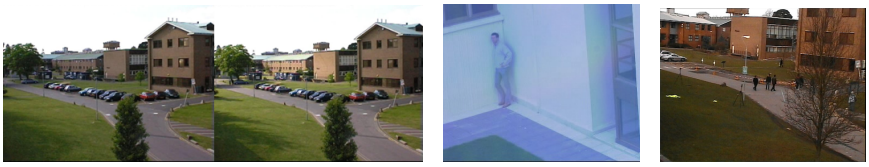

**Figure 1.2** Architecture of the proposed system.

are recognized by analyzing the audio stream acquired by a microphone (*Audio Event Analysis*).

Finally, this large amount of data needs to be properly stored (*Storing*) in a way that the human operator can efficiently interact with the system by submitting different typologies of queries (*Querying*), whose parameters are defined only at query time (that is exactly in the moment the query is thought). For instance, the objects crossing a given area in a given time interval (dynamic spatio temporal query) or the objects following a particular trajectory, similar to the one manually drawn by the user (query by sketch).

A brief description of the above mentioned modules is provided in the following:

- **Detection**: this module is devoted to detect moving objects of interest inside the scene. This is a very complex task to achieve, due to the several kinds of situations that need to be properly managed. A few examples are shown in Figure 1.3: in (a) the same scene taken at different time instant is reported (light change). Consider that a lot of differences in terms of colors, due to the relative position of the sun, are evident. In (b) we can note that the color of the person is



(a) (b) (c)

**Figure 1.3** Typical situations causing detection errors: light changes (a), camouflage (b) and waving trees (c).

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

**Figure 1.4** Typical situations causing tracking errors: an occlusion between three persons (a) and between a person and a pole (b). Typical errors of the detection phase the tracking has to deal with: merge (c), split (d) and undetected object (e). The yellow boxes represent the output of the detection.

very similar to the one of the wall (camouflage), consequently making very difficult to distinguish and then identify him. Finally, in (c) a tree moved by the wind is shown; note that, although it is a moving object, it can not be considered as an object of interest and the system has to filter out this kind of patterns. In other words, only the moving objects of interest needs to be identified and successively tracked. More details about the detection phase will be provided in Section 3.1.

- **Tracking**: this module extracts information about moving objects trajectories and appearance, namely the class (*i.e.* person, bag, vehicle etc.), the real size or the dominant color.

One of the most challenging problem is related to the *occlusions*, happening when an object of interest is partially or totally covered by another object in the scene. This issue is determined by the fact that tracking algorithms analyze a three-dimensional world by using bi-dimensional images. However, in the above mentioned systems it is required that a person has to be tracked (and its trajectory has to be extracted) even if it is partially or completely hidden by another person moving in the scene (Figure 1.4a) or by a static object (Figure 1.4b). Think to a wall that only allows to see a small part of a person (his head or his legs), making this task extremely demanding. Note that this is a very challenging task also for a human operator, who should consider the path of each person populating the scene even if this is partially or totally hidden.

Furthermore, the tracking phase needs to face with the problems introduced by the detection phase, such as merged objects (Figure 1.4c), objects split in different pieces (Figure 1.4d) or undetected objects (Figure 1.4e).

More details about the proposed tracking algorithm will be provided in Section 3.1.

- **Visual Behavior Analysis**: the aim of this module is to alert the human operator as soon as an abnormal behavior occurs. In particular, it processes trajectories extracted during the tracking phase and it is able to identify abnormal trajectories, associated to potential dangerous behaviors, according to previously learnt set of normal and typical paths. It is evident that the system has to be robust enough to deal with the errors that typically occur during the tracking phase, related for instance to broken trajectories. Furthermore, an high level of generalization is required in order to avoid to wrongly classify as abnormal those normal trajectories that only rarely occur. More details about this step will be provided in Sections 3.2 and 3.4.1: in particular, the former will focus on the learning phase while the latter on

the operative phase.

- **Audio Event Analysis**: this module is in charge of detecting abnormal behaviors, such as gun shots, screams and broken glasses, by analyzing the audio stream acquired by means of a microphone. One of the main open issue of this kind of algorithms lies in the fact that the properties characterizing different events of interest might be evident at very diverse time scales: compare, for instance, an impulsive sound like a gunshot with a sustained sound, like a scream, that can have a duration of several seconds. Furthermore, in real applications there is often the problem that sounds of interest are superimposed to significant background sounds; thus it might be difficult to separate noise to be ignored from the useful sounds to be recognized. In Section 4 a detailed description of the proposed method will be provided.

- **Storing**: this module stores the large amount of information previously acquired, in terms of both events and trajectories. Its aim is to improve the overall performance of the system during the retrieval phase. This module needs to face with different problems: the former pertains the huge amount of data to be efficiently managed. As a matter of fact, in real scenarios it is required that millions of trajectories, extracted from different cameras, must be stored and that on this wide database a human operator can submit complex queries involving spatial and temporal data, as well as semantic one. Moreover, it is worth noting that the events can be simply stored by using available relational datasets, while trajectories are much more complex data to deal with, thus customized and enhanced solutions need to be properly defined. The Storing module will be deeply described in Section 3.3.

- **Querying**: human operator can interact with the system by submitting different typologies of queries, whose parameters are defined only at query time, concerning both spatio-

temporal and semantic information. Examples of queries that the user can submit involve the objects crossing a given area in a given time interval (*dynamic spatio temporal query*) or the objects following a particular trajectory, similar to the one hand drawn by the user (*query by sketch*). More details related to these interactions will be provided respectively in Sections 3.4.3 and 3.4.2.

It is worth pointing out that all the modules of the proposed system, besides their own problems, are joined by the following consideration: a common challenge lies in the fact that all these operations need to be performed in real time. It means that meanwhile the video and the audio are acquired (usually with a frame rate of 25 frames per second and a bit rate of 128 Kbit per second respectively) the system has to perform all the required operations (detection of objects, tracking of their movements, storage and analysis of their trajectories, detection of audio events and finally allow an interaction with the user), without showing delay in the response time.

Although increasing the complexity of the overall system, this consideration is very important, since it allows the proposed approach to be used in real applications.

## 1.2 Organization

This thesis consists of the following six chapters:

- **Chapter 1** presents an overview of this research, by analyzing the problem and briefly highlighting the novelties of the proposed approach.

- **Chapter 2** provides an overview of the state of the art methods, by deeply investigating approaches recently proposed for tracking moving objects, analyzing their behaviors and storing useful information respectively in Sections 2.1, 2.2 and 2.3. The chapter is concluded by Section 2.3, which

analyzes the methods for audio recognition proposed in the last years.

- **Chapter 3** analyzes the modules for extracting semantic information starting from the analysis of the videos; in particular, Section 3.1 details the proposed tracking algorithm; in Section 3.2 the method defined for dynamically understanding a scene and then to identify typical behaviors is proposed, while Section 3.3 details the strategy adopted for indexing and storing trajectories data. This chapter is concluded by Section 3.4, where the queries that the user is allowed to submit are defined.

- **Chapter 4** is devoted to analyze the method proposed for recognizing sound events of interest.

- **Chapter 5** analyzes the results obtained by the proposed modules. In particular, five different standard datasets, detailed in the following of this thesis, have been used in order to prove the effectiveness of the proposed algorithms.

- **Chapter 6** draws some conclusions and describes future works.

# Chapter 2

# State of the Art

The framework analyzed in Chapter 1 is really challenging: as briefly analyzed previously, each part composing the system is characterized by its intrinsic problems. This is mainly why in recent literature there is no author facing with the whole problem, but instead the trend is to focus over a single module, or in general over a few modules (for instance Detection and Tracking). This focus is even more pronounced if we analyze the number of papers where information coming from video and audio sources are combined [2].

For the above mentioned reason, in this chapter the state of the art of each module is separately analyzed, so to better highlight novelties and advances with respect to the state of the art introduced in this thesis.

Considering the large amount of methods (more than 2000) proposed in the last years, this thesis analyzes the state of the art starting from the surveys, summarized in Table 2.1, and by reporting some of the most important approaches for each module. In particular, Table 2.1 shows a significantly high number of review papers focusing on Visual Behavior Analysis. It is mainly due to the fact that this concept covers a lot of different kinds of *behaviors*, ranging from actions and activities to anomaly detection. In Section 2.2 more information will be provided in order to better understand this aspect. On the other hand, as highlighted in Table

| | People | Vehicles |
|---|---|---|
| **Detection** | [3] [4] | [5] [6] [7] |
| **Tracking** | [8] | [5] [6] |
| **Visual Behavior Analysis** | [3] [9] [10] [11] [12] [13] [13] [14] [15] | [14] [16] [17] [6] |
| **Trajectories Storing** | [18] [19] | |
| **Audio Event Analysis** | - | |

**Figure 2.1** Survey papers published in the last years partitioned by topic (Detection, Tracking, Behavior Analysis, Trajectories Storing and Audio Event Analysis) and by applications field (oriented to people or vehicles).

2.1, no survey has been proposed about audio-surveillance methods, since only recently the community has started investigating on this topic.

This chapter is organized as follows: methods introducing a tracking algorithm are summarized in Section 2.1; approaches for trajectories-based behavior analysis and trajectories storing are presented respectively in Section 2.2 and 2.3. Finally, the state of the art of the audio surveillance is analyzed in Section 2.4.

## 2.1 Trajectories extraction

The tracking problem is deceptively simple to formulate: given a video sequence containing one or more moving objects, the desired result is the set of trajectories of these objects. This result can be achieved by finding the best correspondence between the objects tracked until the previous frames and the ones identified by the detection phase (from now on the *blobs*) at the current frame, as shown in Figure 2.2. The ideal situation is shown in Figure 2.2a, where each person is associated to a single blob.

Unfortunately, in real world scenarios, there are several issues that make this result far from being easy to achieve: the detection may split a person in several blobs, as shown in Figure 2.2b, or

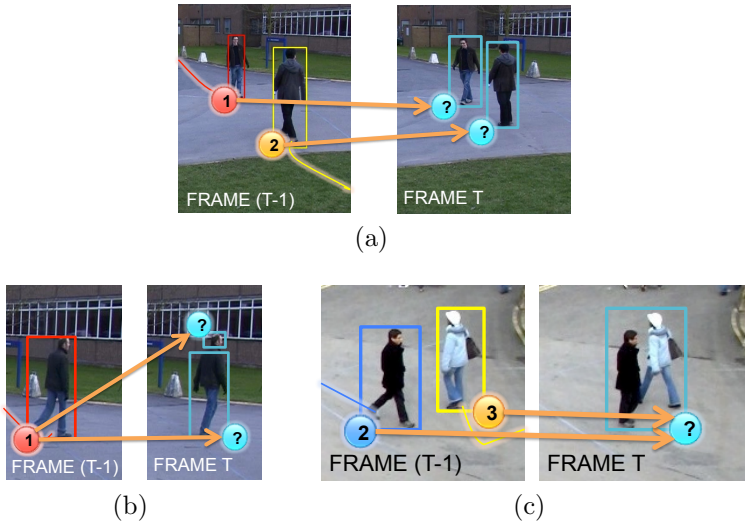



**Figure 2.2** Each image is composed as follows: on the left the objects tracked until the previous frame $t-1$, on the right the blobs identified during the detection step at the current frame $t$. The aim of the tracking algorithm is to perform the best association, identified by the orange arrows: in (a) the ideal situation is shown, in (b) the detection splits the person in two parts while in (c) the persons are merged in a single blob because of an occlusion pattern.

an occlusion pattern may merge two or more persons in a single blob, as shown in Figure 2.2c. It is evident that in such situations more complex associations need to be managed.

Because of these difficulties, many tracking algorithms have been proposed in the last years, but the problem is still considered open.

Tracking algorithms can be divided into two main categories, as summarized in Figure 2.4: the former category (off-line) contains the methods [20][21][22] which extract small but reliable pieces of trajectories, namely the tracklets, instead of entire trajectories. Once all the tracklets are available, the system needs a post-processing step aimed at linking the ones belonging to the same individual for extracting the final trajectories. An example is shown in Figure 2.3a. Although such a strategy significantly increases the final reliability in the extraction of the trajectories, it

(a) (b)

**Figure 2.3** In (a) the tracklets extracted at the time instant $t$ by using off-line tracking algorithms. The entire trajectories will be produced only when all the tracklets will be available. In (b) trajectories extracted at the same time instant by on-line algorithms.

can not be used in on-line applications for behavior analysis, since the human operator should wait for the alert a lot of time, so risking that the abnormal situation can not be properly managed.

The latter category (on-line) contains all those algorithms performing the analysis on-line, so that at each time instant $t$ the entire trajectory of each object is available until $t$ and ready to be used, as shown in Figure 2.3b. In this category, two main strategies have been proposed up to now:

- Detection and tracking: the tracking is performed after an object detection phase; in particular, objects are detected in each frame using a priori model of the objects or some form of change detection: differences from a background model, differences between adjacent frames, motion detection through optical flow and so on. Algorithms following this strategy are usually faster than the ones belonging to the other strategy, but they have to deal also with the errors of the detection phase as spurious and missing objects, objects split into pieces, multiple objects merged into a single detected *blob*.

- Detection-by-Tracking: detection and tracking are performed

at once, usually on the basis of an object model that is dynamically updated during the tracking.

Some examples of algorithms following the first strategy (detection and tracking) are [23] and [24]: the criterion used to find a correspondence between the evidence at the current frame and the objects at the previous one is based on the overlap of the areas. Overlap-based methods work well with high frame rates and when objects do not move very fast, but might fail in other conditions. Positional information, obtained by taking advantage of the Kalman filter, is also used in [25] and [26]. In the former, only the distance between the detected blob and the predicted position is considered; on the contrary, in the latter the appearance information is taken into account by means of a smoothed 4D color histogram.

Dai *et al.* [27] have proposed a method able to track pedestrians by using shape and appearance information extracted from infrared images. The method may have some problems when objects quickly change their appearance or during occlusions.

The method proposed in [28] formulates the tracking problem as a bipartite graph matching, solving it with the well-known Hungarian algorithm. It recognizes an occlusion, but is able to preserve the object identities only if the horizontal projection of the detected blob shows a separate mode. The Hungarian algorithm is also used in [29] and [30] in order to solve the assignment problem. In general, its main drawback lies in the polynomial time needed for the computation, which prevents these tracking methods from managing crowded scenarios.

The method by Pellegrini *et al.* [31] tries to predict the trajectories on the scene using a set of behavior models learned using a training video sequence. The method is very effective for repetitive behaviors, but may have some problems for behaviors that do not occur frequently.

Several recent methods [32][33][34] use the information from different cameras with overlapping fields of view in order to perform the occlusion resolution. The data provided by each camera are usually combined using a probabilistic framework to solve am-

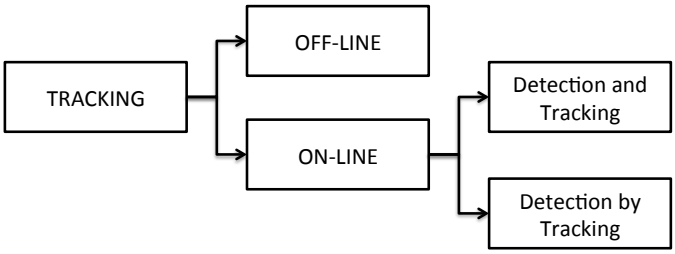



**Figure 2.4** Trajectories extraction: the state of the art methods.

biguities. These methods, although increasing the reliability of the entire system, are limited to situations where multiple cameras can be installed; furthermore, most of the methods adopting this approach requires a full calibration of each camera, which could make the deployment of the system more complicated.

On the other hand, methods belonging to the second strategy (Detection-by-Tracking) are computationally more expensive, and often have problems with the initial definition of the object models, that in some cases have to be provided by hand. The paper by Comaniciu *et al.* [35] proposes the use of Mean Shift, a fast, iterative algorithm for finding the centroid of a probability distribution, in order to determine the most probable position of the tracked target. It requires a manual selection of objects being tracked in the initial frame, and deals only with partial occlusions. This consideration implies that the method can not be applied in the proposed framework. Tao *et al.* [36] have proposed a method based on a layered representation of the scene, that is created and updated using a probabilistic framework. Their method is able to deal with occlusions, but is extremely computational expensive, requiring up to 30–40 seconds per frame. The method by Bhuvaneswari and Abdul Rauf [37] uses edge-based features called *edgelets* and a set of classifiers to recognize partially occluded humans; the tracking is based on the use of a Kalman filter. The method does not handle total occlusions, and, because of the Kalman filter, it works better if people are moving with uniform direction and speed. The method proposed by Han *et al.* [38] detects and tracks objects by

using a set of features, assigned with different confidence levels. The features are obtained by combining color histograms and gradient orientation histograms, which give a representation of both color and contour. The method is not able to handle large scale changes of target objects. The method by Yogameena *et al.* [39] uses a skin color model to detect and then track faces in the scene. The method is able to deal with crowded scenes where the persons are dressed with very similar attire, but it works only as long as the face of each person remains clearly visible.

Several recent methods [40] [41] [42] [43] [44] [45] [46] [47] [48] have investigated the use of Particle Filters, a tool based on the approximate representation of a probability distribution using a finite set of samples, for solving the tracking problem in a Bayesian formulation. Particle Filters look very promising, since they make tractable a very general and flexible framework, which can incorporate in the probability distribution also information related to the past history of objects. However, the computational cost is still too high for real-time applications, especially with multiple occluding targets, with a processing time ranging from 0.5 to 15 seconds per frame.

A recent, promising trend in tracking algorithms is the use of machine learning techniques. As an example, the method by Song *et al.* [49] improves the ability of tracking objects within an occlusion by training a classifier for each target when the target is not occluded. These individual object classifiers are a way of incorporating the past history of the target in the tracking decision. However, the method assumes that each object enters the scene un-occluded; furthermore, it is based on the Particle Filters framework, and so it is computationally expensive. Another example is the method by Wang *et al.* [50] that uses *manifold learning* to build a model of different pedestrian postures and orientations; this model is used in the tracking phase by generating for each object of the previous frame a set of candidate positions in the current frame, and choosing the closer candidate according to the model.

The high computational effort required by Detection-by-Tracking

algorithms is the main limitation for using such strategy in the proposed system. On the other hand, it is needed that at each time instant the entire trajectory is available in order to properly interpret the event associated to it, so making also unfeasible off-line algorithms. For these reasons, the tracking algorithm proposed in this thesis is based on the first of the above mentioned strategy (Detection and Tracking): it assumes that an object detection based on background subtraction generates its input data. However, one of the main limitations of the existing algorithms using a Detection and Tracking strategy lies in the fact that they make their tracking decisions by comparing the evidence at the current frame with the objects known at the previous one; all the objects are processed in the same way, ignoring their past history that can give useful hints on how they should be tracked: for instance, for objects stable in the scene, information such as their appearance should be considered more reliable. To exploit this idea, the proposed algorithm adopts an object model based on a set of scenarios, in order to deal differently with objects depending on their recent history and conditions; scenarios are implemented by means of a Finite State Automaton, that describes the different states of an object and the conditions triggering the transition to a different state. The state is used both to influence which processing steps are performed on each object, and to choose the most appropriate value for some of the parameters involved in the processing.

It is worth pointing out that another important advantage deriving by the use of a strategy based on Detection and Tracking is that the variables characterizing the tracking process are explicitly defined and easily understandable, and so can be used in the definition and in the manipulation of the state; an approach of the second kind (Detection by Tracking), especially if based on machine learning, would have hidden at least part of the state, and thus the history of the objects would not have been explicitly manageable through a mechanism such as the Finite State Automaton.

Furthermore, although a limited a priori knowledge about the objects of interest is required, in order to differentiate between


(a) (b) (c)

**Figure 2.5** Behavior analysis can be performed at different layers: gestures (a), actions (b) and activites (c).

single objects and groups, the method can be rapidly adapted to other application domains by providing a small number of object examples of the various classes. Finally, the spatio-temporal continuity of moving objects is taken into account using a graph-based approach. Thanks to this novel approach, our system is able to simultaneously track single objects and groups of objects, so significantly increasing the overall reliability of the proposed approach.

More details about the proposed tracking algorithm will be provided in Section 3.1.

## 2.2 Visual Behavior Analysis

The visual behavior analysis can be performed at different levels; although it is not possible to identify a strict partition among them, a common taxonomy identifies the following three layers: *gestures*, *actions* and *activities*. A simple example for each kind of behavior is shown in Figure 2.5. Gestures are elementary movements of a single body part; often, this terms just refers to hand motions. Think, as an example, to a person which uses hand gestures to control the computer mouse or keyboard functions [51]. An action is a more complex behavior of a single person that may be composed of multiple gestures organized temporally, such as walking, drinking, or speaking [52]. Finally, an activity is the

most complex behavior and is associated to the overall movement of the objects inside the scene, namely to its trajectory. An activity can involve a single object (a vehicle crossing the highway in the wrong side) or a group of objects (two persons are fighting) [14].

In real applications devoted to video surveillance detailed information related to the pose of persons is not available: 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. On the other hand, we can note that the analysis of moving objects trajectories can allow to understand common behaviors inside a specific context: it is due to the fact that the movement of objects in a scene is not random, but instead has an underlying structure which can be exploited to build some models. In particular, in the contexts mentioned before the key point is the understanding of *activity patterns*, which are *the underlying hidden process that dictates how objects move in a scene*. In this thesis we will focus on the activity pattern recognition and analysis: in particular, once learnt the activity patterns, the system is able to compute how much a new trajectory is similar to a pattern and then to recognize abnormal trajectories if not enough similar to a given pattern.

The process of learning activity patterns requires that each trajectory is properly represented (pre-processing) and that the *typical* patterns are computed. In particular, a typical strategy consists in extracting these typical patterns in an unsupervised way, so to make the entire system more general and able to be easily adapted to different environments. For this reason, a clustering algorithm needs to be defined.

Note that preprocessing and clustering steps are very related each others, since the preprocessing step 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 [53]) in order to easily verify if two trajectories are similar or not, or a dimensionality reduction, which allows

to represent a trajectory into a lower dimensional space, computationally less expensive (for instance by vector quantization [54], string quantization [55][56] or principal component analysis and wavelet transform [57]).

It is clear that the chosen representation strongly influences the definition of a metric encoding the similarity between trajectories. For instance, in [58] raw data are used and the similarity is evaluated by using a Time-Delay Neural Network. Although in this case the preprocessing step is avoided, the performance of the control system strongly depends on the accuracy of the extracted trajectories and a large amount of data is needed during the learning phase.

A common strategy consists in representing a trajectory by means of a single features vector and then evaluating the similarity by using one of the common distances: Hausdorf [59], Mahalanobis [60] or Bhattacharyya [61]. In [55], [62], [63] and [64] trajectories are represented as a sequence of symbols and statistical machine learning approaches are exploited. In particular, in [55], [62] and [63] the similarity between trajectories is evaluated by using the Edit Distance, while in [64] by a Dynamic Time Warping (DTW) based approach. DTW 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. The main problem of the above mentioned techniques lies in the fact that, although being able to compute a distance, are not able to easily define a metric.

In [65], [66] and [67] 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 observations are defined to be similar in terms of common similarity to a model, expressed through the likelihood function. The main benefit of this approach 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.

As for the clustering step, it aims at discovering prototypical activities starting from the analysis of preprocessed trajectories; furthermore, each clustering algorithm requires the definition of a proper similarity measure. As summarized in [68], different typologies of clustering have been recently exploited for dynamic scene understanding: hierarchical agglomerative or divisive techniques, graph cuts and spectral methods. The main limitation in the above mentioned algorithms lies in the fact that they do not allow to readily verify if a novel trajectory belongs to a cluster, an then do not allow to understand if a trajectory, and then the associated behavior, has to be considered normal or abnormal.

To avoid this restriction, $k$-means approach and its derivative methods are most frequently used. In particular, the Kernel $k$-means [69] 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 the resulting feature space. In this way, this algorithm allows to separate non linearly separable clusters. In [70] 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.

As regarding the trajectories pre-processing and the similarity measure, a common limitation of the above mentioned approaches lies in the fact that either cause an important loss of information from trajectories to vectors or that the similarity or dissimilarity

measures, such as the string edit distance, used by these methods do not induce as a metric, and then most of the traditional statistical tools used in pattern recognition (for instance many efficient clustering algorithms) can not be applied.

On the other hand, regarding clustering algorithms, two main drawbacks can be found in the considered algorithms: first, the initialization of the $k$-means based approaches strongly influences the performance of these methods, since the algorithms converge to the local minimum closest to the initial cluster's centroids. Furthermore, the high computational effort required for the activity patterns extraction closes the door the the application of such algorithms for our purposes.

Starting from the limitations of existing approaches, I decide to represent the trajectories by using a string based approach, in order to reduce the dimensionality without losing positional and other useful information. Furthermore, I propose a novel similarity metric based on kernels: the main advantage is that the problem can be formulated in an implicit vector space on which statistical methods for pattern analysis can be applied. Furthermore, a novel and efficient kernelized clustering algorithm, able to perform a partition into $k$ clusters by only performing $k-1$ iterative bisections, has been defined in this thesis. The proposed method for behavior analysis will be detailed in Section 3.2.

## 2.3 Storing and Retrieval

Another relevant consequence of the scenario we are working on is the possibility of storing a huge amount of trajectories extracted from the scene, each annotated with the properties of the associated object; this is related to the important perspective to efficiently extract synthetic data by suited queries using jointly geometric and temporal information: geometric information refers to the spatial area where the event of interest occurs, while temporal one pertains the time interval associated to the event of interest. For instance, if we are interested in investigated on the theft which

took place in the line 4 of the train station at 9pm, we need to find all the objects, and then all the trajectories stored in our database, crossing the portion of the scene where line 4 lies in the selected temporal interval (8:50pm - 9:10pm).

However, only a modest attention has been devoted to systems able to store and also retrieve off-line moving object trajectories, able to cope with very large amount of trajectory data and sufficiently general to deal with the needs of different application domains. This is an important and not negligible feature, especially when considering crowded real world scenarios. In these cases it is required that millions of trajectories must be stored and that, on this wide database, the user must be able to submit complex queries involving geometric and temporal data.

A few tries have been given in this sense in the last years; for instance, the method proposed in [71] aims at counting the number of vehicles crossing the scene at a given time instant. It is achieved by partitioning the scene into a fixed number of zones and by indexing the trajectory data using a multimedia augmented transition network model and a novel *multimedia input strings*. A similar approach has been used in [72], where the spatial structures of the monitored environment (like its division into regions corresponding to traffic lanes, curb, exit lane, etc.) are directly stored into the database.

The main drawback of the above mentioned methods is related to the fact that the reduction of the computational cost is obtained at the expense of the staticity of the query's schema. It is required that the parameters (both geometrical and temporal) included in queries are pre-determined and cannot be consequently chosen by the user at query time; this implies that the user is forced to use a set of pre-definite queries and the addition of another query typology requires heavy operations by the system designer, involving a system stop and a database recompiling.

An example of query supported by the above mentioned systems can be given by asking to the system to retrieve all those vehicles flowing on the different lanes. The limitation becomes evident when we would like to determine, for example, the vehicles

running exactly over the center line (so occupying two half lanes): in this case it is necessary to define at query time the area in which to find vehicles (a rectangle centered on the line dividing the two lanes).

Although the systems above cited are good examples of the potentiality of spatio-temporal queries in a given domain, a step towards their geometrical generalization could significantly improve their usability; in fact, they suffer from the fact that parameters characterizing the queries are mostly pre-determined and cannot be chosen by the user at query time. Their rationale is to have a system architecture devised and optimized for supporting a bunch of queries, each one referring to a given spatial area; this is sufficient to solve the corresponding retrieving problem, but the capability of choosing at query time (i.e. exactly when the query is thought) the area in which we are interested in is neglected.

The main contributions to this problem can be obtained by browsing the literature coming from the database field: while objects' meta-informations can be easily stored using standard tables of well known relational DBMSs, spatially enabled DBMSs can be profitably used to store trajectories represented as a sequence of 3D points. In particular, indexing moving objects databases has been an active research area in the recent past and several solutions have been proposed. [18] and [19] survey many accessing strategies, proposed in the last two decades, which are able to index the past and the current position, as well as methods supporting queries related to the future. A widely adopted solution for bi-dimensional spatial indexing is based on R-Trees [73], which hierarchically organize geometric bi-dimensional data representing each object by using its *MBR* (Minimum Bounding Rectangle), an expression of the object's maximum extents in its coordinate system. The conceptual simplicity of an R-Tree and its resemblance to widely adopted standard B-Trees, allowed the developers to easily incorporate such a solution in spatial enabled DBMS [74] in order to support spatial query optimization and processing.

Starting from the original R-Tree structure, several improved versions have been proposed: STR-trees [75] extend R-tree with a

different insert/split algorithm, while the characteristics of spatio-temporal data are captured by two access methods (STR-tree and TB-tree).

When objects' movements are constrained, for example on a network of connected road segments, a bidimensional R-tree can be used to index the static network's segments. In this case, each leaf contains a segment and a pointer to a monodimensional R-Tree that indexes the time intervals of objects' movements, as for FNR-Tree [76]. MON-tree [77] extends the FNR-tree by modeling the constrained network as a set of junctions and routes; a bidimensional R-tree is used to index polylines' bounding boxes while, for each polyline, another bidimensional R-tree indexes the time dimension of the objects within the polyline. PARINET [78] has been designed for historical data in constrained networks and models the network as a graph; trajectories are partitioned according to the graph partitioning theory. This method has been extended to handle continuous indexing of moving objects [79].

When dealing with real applications for indexing and querying large repositories of trajectories, the size of MBRs can be reduced by segmenting each trajectory and then indexing each sub-trajectory by using R-Trees; such an approach is described, for example, in [80], where a dynamic programming algorithm is presented for the minimization of the I/O for an average size query. SETI [81] segments trajectories and groups sub-trajectories into a collection of *spatial partitions*; queries run over the partitions that are most relevant for the query itself. TrajStore [82] co-locates on a disk block (or in a collection of adjacent blocks) trajectory' segments by using an adaptive multi-level grid; thanks to this method, it is possible to answer a query by only reading a few blocks.

All the above cited methods, even presenting efficient solutions from different perspectives, are typically not supported in the available commercial products that make use of very efficient spatial indexes that, unfortunately, are typically restricted to the bidimensional case. For instance, PostGIS [83], the well known open source spatial extension of the PostgreSQL DBMS, even support-

ing three (and four)-dimensional data, only recently introduced the possibility of indexing 3D data providing a first support for those queries involving intersection between two geometries. As a consequence, there is a strong interest in those methods which, even using well established 2D solutions, allow to solve the indexing problem in a multi-dimensional space.

In this thesis we propose to index 3D trajectories by turning the 3D representations into a set of 2D schemes, so as to make it possible the use of well established and optimized available 2D indexing solutions. In particular, the proposed solution is optimized to solve Range Spatial Queries assumed to be Dynamic (DRSQs), i.e. are formulated in any their part at query time, so allowing the user the important potentiality of extracting from the database non only predefined information but the data he is interest in at any moment of system's use. The contribution of this thesis also includes the definition of different typologies of 3D queries, so general to hopefully cover most of the applicative needs and that can be formulated as a combination of two or more DRSQs.

## 2.4 Audio Analysis

Audio analysis has been traditionally focused on the recognition of speech [84, 85] and speaker identification [86, 87]. In recent years several researchers have proposed audio-based systems for the automatic detection of abnormal or dangerous events for surveillance purposes. Such systems can be an inexpensive addition to existing video surveillance infrastructures, where video analytic solutions are used; in fact, many IP cameras are already predisposed to connect to a microphone, making available an audio stream together with the video stream. Furthermore, as mentioned before, there are some events that have a very distinctive audio signature, but are not so easy to spot on a video: for instance, a gunshot, or a person screaming. For these reasons, in the recent years the research community has shown a growing interest towards those audio surveillance applications able to reliably identify those events

of interest happening in the environment. In particular, most of the approaches recently proposed focuses on the recognizing of screams, gunshots and broken glasses.

In [88] Clavel et al. propose a method for gunshot detection, that operates by dividing the audio stream into 20 milliseconds frames, and computing for each frame a vector with such features as short-time energy, Mel-Frequency Cepstral Coefficients (MFCC) and spectral statistical moments. The vectors are classified using a Gaussian Mixture Model (GMM). Then, the final decision is taken over 0.5 seconds intervals using a Maximum A Posteriori (MAP) decision rule. Vacher et al. in [89] also adopt a GMM classifier, with wavelet-based cepstral coefficients as features, for the detection of screams and broken glass. Rouas et al. [90] use MFCC features and a combination of the GMM and Support Vector Machine (SVM) classifiers for detecting screams in outdoor environments. Their method uses an adaptive thresholding on sound intensity for limiting the number of false detections. Gerosa, Valenzise et al. [91, 92] propose a system for the detection of gunshots and screams which specifically address the ambient noise problem. Their method uses two parallel GMM classifiers trained to separate screams from noise and gunshots from noise. The paper by Ntalampiras et al. [93] proposes a two stage classifier: the first stage is used to discriminate between vocal sounds (such as screams and normal speech) and impulsive sounds (such as gunshots or explosions); then specific second stage GMM classifiers are activated, using different features for the two kinds of sound, to provide the final classification of the sound. In [94], the same authors explore techniques for novelty detection with application to acoustic surveillance of abnormal situations. In [95], Rabaoui et al. address the problem of reducing the effect of the environmental noise on the classification results by defining a novel and sophisticated dissimilarity measure, combined with a pool of one-class Support Vector Machine (SVM) classifiers. Conte et al. [96] present a method with two classifiers that operate at different time scales; the method uses a quantitative estimation of the reliability of each classification to combine the classifier decisions

and to reduce the false detections by rejecting the classifications that are not considered sufficiently reliable. Chin and Burred [97] propose a system in which the audio is represented as a sequence of symbols, each corresponding to a spectral shape observed over a 10 milliseconds window. Then, they apply to these sequences the Genetic Motif Discovery, a technique introduced for the analysis of gene sequences, in order to discover sub-sequences that can be used to recognize the audio events of interest. The algorithm is able to consider sub-sequences of different lengths for different classes, and the sub-sequences may contain *wildcard* elements that can be used to skip variable symbols due to the background noise.

However, one of the open problems in the design and implementation of a reliable and general audio event detector lies in the fact that the properties characterizing the different events of interest might be evident at very diverse time scales: compare, for instance, an impulsive sound like a gunshot with a sustained sound, like a scream, that can have a duration of several seconds. It is not easy to find a set of features that can accommodate both kinds of situation. Also, in real applications there is often the problem that the sounds of interest are superimposed to a significant level of background sounds [98]; thus it might be difficult to separate the noise to be ignored from the useful sounds to be recognized.

A strategy often applied in other fields, ranging from textual documents retrieval to human actions recognition and video-based object detection, is the *bag of words* approach. The datum to be classified is represented by detecting the occurrence of local, low-level features (*words*) and constructing a vector whose dimensionality corresponds to the number of possible words, and whose elements are indicators of the presence of the corresponding words, or a count of their occurrences. For instance, in text characterization the low-level features are the actual natural language words of a document (after removing suffixes, articles etc.), and the whole document is represented by a (high dimensional) vector of word occurrences; such vectors are then classified using traditional Pattern Recognition. For the extension of these approaches to Computer

Vision, the words are replaced either by small fixed-size image patches, or by salient points (e.g. SIFT features). In these cases, since the space of the possible words is huge (theoretically infinite), a quantization is performed using a training set; the result is a *codebook* that allows to associate each low-level feature with one word chosen from a finite set. In audio analysis, the bag of words approach has been recently applied to music hashing and retrieval [99] and to music classification [100].

In this thesis we introduce a novel audio event detection system for an audio surveillance application able to face the problems mentioned before by employingw a *bag of words* classifier. The proposed system uses a two level description of the audio stream: first-level features are computed on a very short time interval, and are somewhat analogous to the words of a text. Second-level features characterize a longer time interval, and are constructed by means of a learning process, on the basis of the actual sounds to be recognized. Finally, a classifier is trained on second level features, so as to learn which of them are significant for the recognition of a particular class of events and which ones are irrelevant. This architecture is thus able to work on a longer time scale, but still remains able to give the right weight to short relevant sounds. Furthermore, the presence of background noise has a reduced impact because the classifier can learn to ignore the second level features that are due to the background.

# Chapter 3

# From Pixels to Behaviors

In this chapter we describe the approach proposed in this thesis for extracting by visual inspection semantic information associated to moving objects' behavior. In particular, Section 3.1 introduces the problem, by focusing on method I propose for the extraction of moving objects' trajectories directly from the video data. In Section 3.2 I show that the method proposed in this thesis is able to process trajectories in order to discover *typical* paths inside a scene. The discussion continues in Section 3.3, where the method introduced for efficiently storing and indexing trajectories is detailed. Finally, in Section 3.4 the different interactions allowed to the user are presented.

## 3.1 Trajectories Extraction

As introduced in Section 2.1, the trajectories extraction module is in charge of extracting moving objects trajectories starting from the analysis of raw videos. A general overview is depicted in Figure 3.1: two main steps are required, namely the detection and the tracking.

The aim of the detection step is to obtain the list of *blobs*, being each blob a connected set of foreground pixels; in order to achieve this aim, the detection module first finds foreground pixels by comparing the frame with a background model; then

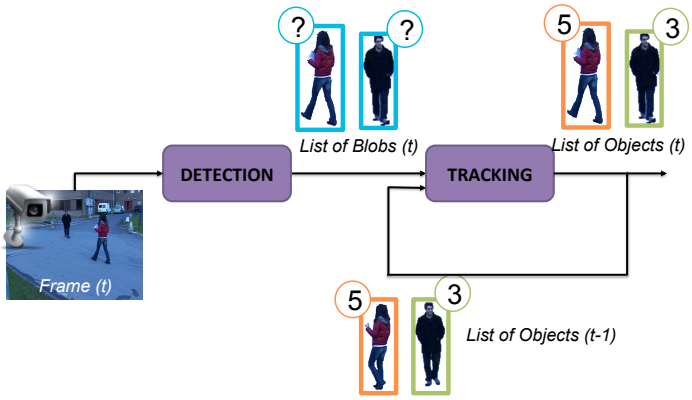



**Figure 3.1** Architecture of a generic tracking system based on a background subtraction algorithm: the list of blobs is extracted by the detection module and is analyzed by the tracking algorithm in order to update the information associated to the list of objects populating the scene.

foreground pixels are filtered to remove noise and other artifacts (e.g. shadows) and are finally partitioned into connected components, namely the *blobs*.

The tracking algorithm receives as input the set of blobs detected at each frame and produces a set of *objects*. An *object* is any real-world entity the system is interested in tracking. Each object has an *object model*, containing such information as

- the object class (*e.g.* a person or a vehicle) (Subsection 3.1.2.3),
- state (Subsection 3.1.2.2),
- size, position and predicted position, trajectory and appearance (Subsection 3.1.2.5).

A *group object* corresponds to multiple real-world entities tracked together; if a group is formed during the tracking (*i.e.* it does not enter in the scene as a group), its object model maintains a reference to the models of the individual objects of the group.

The task of the tracking algorithm is to associate each blob to the right object, so as to preserve the identity of real-world objects across the video sequence; the algorithm must also create

new object models or update the existing ones as necessary. As highlighted in Section 2.1 (see Figure 2.2), this problem is far for being simple because of occlusions, split or merge patterns that may happen because of the perspective flattening introduced by the use of a single camera or by errors of the detection step. The algorithm defined in this thesis is able to efficiently deal with the above mentioned problems: in particular, we used as baseline the detection algorithm proposed in [101], while a novel tracking algorithm has been introduced.

For the sake of completeness, in Subsection 3.1.1 only a brief description regarding the detection module will be provided. It is worth pointing out that the proposed tracking algorithm, detailed in Subsection 3.1.2, is able for its nature to deal with any kind of detection algorithm.

### 3.1.1 Detection

Detection algorithms can be classified in the two following main categories:

- *derivative algorithms*: working by comparing adjacent frames of the video, under the assumption that foreground objects correspond to rapidly changing areas, while the background is either static or slowly changing;

- *background subtraction algorithms*: the current frame of the video is compared with a background model, that is a (usually compact) representation of the set of the possible images observable when the scene does not contain foreground objects.

As for the first class, the main problem lies in the fact that the algorithms consider the changing parts of the image as foreground [102]. This yields two kinds of problems: on one hand, sometimes parts of a foreground object (even large parts) do not appear to change, either because the object remains motionless momentarily, or because it has a uniform color and texture, and

so its motion determines a pixel change only at its borders (*foreground aperture*). On the other hand, sometimes the pixel values of background areas do change, for instance due to lighting variations, or small uninteresting movements of objects that should be considered static (e.g., tree leaves moved by the wind). In this case, false foreground objects would be detected by a derivative algorithm.

To avoid these problems, the most common approach is based on a background subtraction strategy: benefits coming from this choice are paid in terms of algorithmic complexity. In fact, the background model must be initialized and, more important, continuously kept up to date to reflect changes in the observed scene.

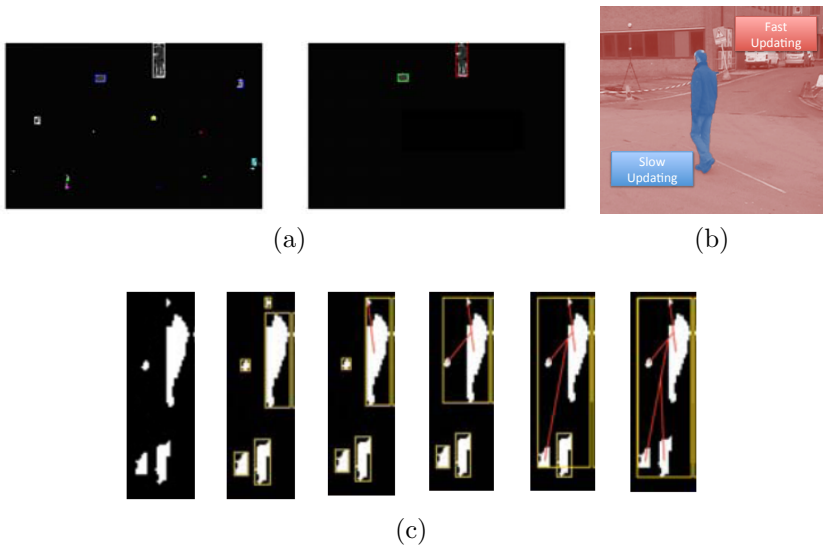
In particular, three different categories can be identified [102]:

- *reference image models*: represent the background as a single image; the comparison between the background model and the current frame is performed by computing the distance in the color space between the corresponding pixel values; pixels whose distance from the background is above a threshold are assigned to foreground;

- *probabilistic models*: represent the background as a probability distribution; the comparison between the background model and the current frame is performed by computing the probability that each pixel is generated according to the background distribution; pixels whose probability is below a threshold are assigned to foreground;

- *neural models*: represent implicitly the background by means of the weights of a neural network suitably trained on a set of uncluttered frames; the network learns how to classify each pixel into background and foreground.

It is worth pointing out that there is no large consensus in the scientific community on which background subtraction method gives the best results; in [102] a comparison involving algorithms belonging to different categories have been performed and the obtained results confirm that [101] ranks at the top positions, at-




(a) (b)

(c)

**Figure 3.2** Some of the heuristics applied by the detection algorithm used in this thesis: (a) Broken Object Recovery, (b) Background Maintenance Algorithm and (c) Noise Filtering.

taining a high value of the performance indices on all the datasets used for the experimentations. For the above mentioned reason, in this thesis it is used the algorithm proposed in [101].

A generic background subtraction algorithm is composed by the following steps:

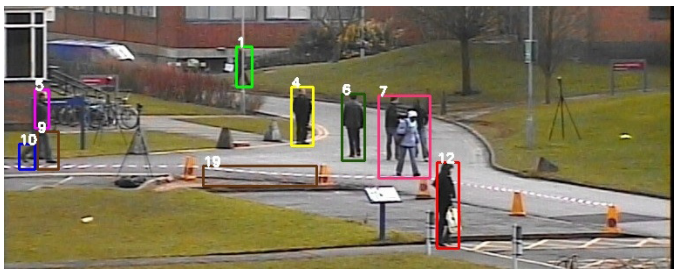
- pixel segmentation, which produces a foreground pixel mask from input frames, obtained by thresholding the absolute difference between the current image frame and the background image;
- morphological filtering, which is applied to the foreground mask;
- blob segmentation, which identifies semantically separated groups of pixels and localizes them by a connected components labeling algorithm.

Furthermore, in [101] the following heuristics have been applied in order to make the system more robust in real environments:

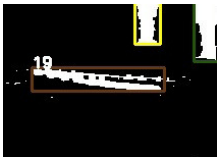
- *Adaptive Threshold*: the threshold used during the pixel segmentation step is adapted based on brightness changes of the scene;
- *Noise Filtering*: the spurious blobs identified during the blob segmentation step are filtered according to their dimensions and density with respect to the bounding box area (see Figure 3.2b);
- *Background Maintenance Algorithm*: the background is updated by using two different speeds, depending on the region; a simple example is shown in Figure 3.2a: for background pixels (red in the figure) the new values are updated very quickly in order to guarantee an accurate updating of the background if some variations occur; for detected objects region (blue in the figure) a very slow update policy is needed in order to avoid including wrong information in the background.
- *Broken Object Recovery*: blobs size is analyzed and blobs are merged, depending on their height and their reciprocal positions (see Figure 3.2c).

It is worth pointing out that in real cases the detection phase produces some common errors, which tracking algorithms need to deal with, as shown in Figure 3.3:

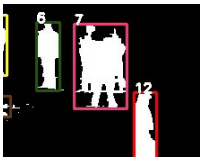
- *spurious blobs*, *i.e.* blobs not corresponding to any object; they can be caused by lighting changes, movements either of the camera or the background, and other transient changes that the detection algorithm was not able to filter out. An example is the blob identified by number 19, shown in Figure 3.3b. In such situation the background updating algorithm is not fast enough to compensate for the movement of the ribbon caused by the wind.



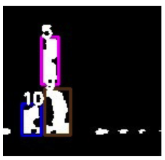

(a)




(b) spurious blob




(c) merged objects




(d) split object

**Figure 3.3** Typical problems of the detection phase. (a) A frame with the detected blobs, showing a spurious blob (number 19), objects merged into a single blob (number 7) and an object split into multiple blobs (numbers 5, 9 and 10). (b), (c), (d) a close-up of the detected foreground mask for the detection problems.

- *ghost blobs*, *i.e.* blobs appearing where there was an object previously considered as part of the background, that has moved away (*e.g.* if a parked car starts moving, a blob is wrongly found in that position);

- *missing blobs*, *i.e.* undetected objects, for instance objects too similar to the background behind them;

- *split blobs*, *i.e.* objects divided into multiple blobs. Figure 3.3d shows an example of an object split into three different blobs, identified by numbers 5, 9 and 10, because of camouflage. The camouflage is the typical problem occurring when the pixel characteristics of a foreground object are too similar to the background to be discerned, as happens when a person is wearing clothes having similar colors to the background.

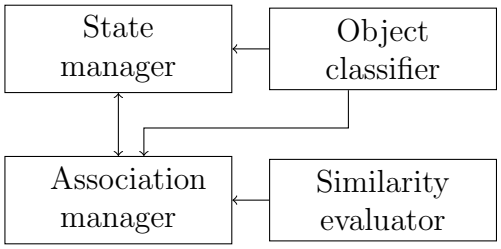



**Figure 3.4** An overview of the proposed tracking system.

- *merged objects*, *i.e.* multiple objects merged into a single blob; it is caused by partial or total occlusions between multiple persons. The algorithm must also handle this kind of situations, ensuring that object identities are not lost across the occlusion. An example of multiple occlusion is shown in Figure 3.3c, where three different individual objects are merged into a single blob, identified by number 7.

At this point it should be clear that the detection step typically introduces different typologies of errors that need to be properly managed by the moving objects tracking algorithm, so making such a task very difficult to achieve. However, we will show that the algorithm proposed in this thesis, described in the following section, reveals to be very robust with respect to errors and it is able to work with partial and total occlusions.

### 3.1.2 Tracking Algorithm

The main lack of most of existing algorithms lies in the fact that they make their tracking decisions by comparing the evidence at the current frame with the objects known at the previous one; all objects are managed in the same way, ignoring their past history that can give useful hints on how they should be tracked: for instance, for objects, stable in the scene, information such as their appearance should be considered more reliable.

To exploit this idea, the proposed algorithm adopts an object model based on a set of scenarios in order to deal differently with

objects depending on their recent history and conditions; the scenarios are implemented by means of a Finite State Automaton (FSA), that describes the different states of an object and the conditions triggering the transition to a different state. The state is used both to influence which processing steps are performed on each object, and to choose the most appropriate value for some of the parameters involved in the processing.

Although a limited a priori knowledge about objects of interest is required, in order to differentiate between single objects and groups, the proposed method can be rapidly adapted to other application domains by providing a small number of object examples of the various classes.

Figure 3.4 gives an overview of the modules composing the tracking system and their interdependencies:

- the *state manager*, which maintains and updates an instance of the FSA for each object;
- the *association manager*, which establishes a correspondence between objects and blobs, solving split events and performing occlusion reasoning;
- the *object classifier*, which assigns objects to a set of predefined classes; the object class is used both during the update of the FSA state and during the association between objects and blobs to solve split/merge cases;
- the *similarity evaluator*, which computes a similarity measure between objects and blobs, considering position, size and appearance; this similarity is used during the association between objects and blobs.

The above modules share a set of objects models, which, as previously said, contain all the relevant information about each object.

Figure 3.5 shows an outline of the tracking algorithm. The algorithm operates at the arrival of each new frame, receiving as inputs the existing object models and the blobs discovered by the

```
procedure Tracking(obj_models, blobs)
  Classify(blobs)
  S := ComputeSimilarity(obj_models, blobs)
  FindAssociations(obj_models, blobs, S)
  UpdateModels(obj_models, blobs)
  UpdateState(obj_models)
end procedure
```

**Figure 3.5** The structure of the tracking algorithm.

detection phase for the current frame; for the first frame of the sequence, the existing object models are initialized as an empty set. The output of the algorithm is a set of updated object models, which possibly includes new objects. The steps of the algorithm are the following:

- first, the classifier is applied to the current blobs, which are annotated with the information on the assigned class;
- then, the algorithm computes the similarity between each object and each blob, activating the similarity evaluator; the similarity information is kept in a similarity matrix `S`;
- at this point, the algorithm is ready to perform the association between objects and blobs, including the split/merge and occlusion reasoning;
- on the basis of the associations found, the algorithm updates the models for each object; if new objects are detected, their models are created at this step;
- finally, the state manager updates the FSA states, using the previous state and the information gathered by the previous steps and stored in the object models.

In the following subsections, more details are provided for each module of the system.


**Figure 3.6** Problems with entering objects. (a) Blobs across three adjacent frames. (b) The tracking performed without considering the object history; in this case a tracking algorithm would not be able to distinguish this situation from two separate objects joining a group, and so attempts to preserve the separate identities of the two objects 1 and 2.

#### 3.1.2.1 Common problems

In this section we examine some typical problems of people detection, in order to see how they can be solved by incorporating information about the history of objects.

One of the most frequently encountered issue is related to objects entering the scene, which have a very unpredictable appearance during the first frames of their life. Figure 3.6 shows a typical example, in which the person is split by the detection phase into two different blobs (*i.e.* legs and arms). The problem here is that after a few frames the parts appear to merge forming a new group of objects; since the occlusion resolution requires that object identities are preserved within a group, the tracking algorithm would continue to keep track of two separate objects (labeled 1 and 2 in the figure). To solve this problem, the tracking algorithm has to use different rules for dealing with merging objects when they are just entering or when they are stable within the scene.

Missing blobs are another typical problem affecting the detection; they can be caused either by camouflage, occurring when the foreground object is very similar to the background, or when occlusions between moving objects arise. The latter case is really difficult to deal with as information about the occluded part is totally missing and consequently to be restored by suited reasoning.

Figure 3.7 shows an example of the above mentioned problem: the person that passes behind the tree is detected in the first


**Figure 3.7** Problems with totally occluded objects. (a) The detection output across three adjacent frames. (b) The tracking performed without considering the object history; in this case a tracking algorithm would see the blob 2 as a new object in the scene, since in the previous frame there was no corresponding object.

and in the third frames of the sequence, but not in the second one. Thus, the tracking algorithm would find at the third frame a blob having no corresponding object in the previous frame, and would assign it to a newly created object, if it does not keep some memory about an object even when it is not visible in the scene. On the other hand, the tracking algorithm should not preserve information about objects that are truly leaving the scene: doing so it would risk to reassign the identity of an object that has left the scene to a different object that is entering from the same side.

Other issues are related to objects occluding each other, forming a group. Figure 3.8a illustrates a problem connected with the stability of group classification: in the first frame, the two persons in the group are perfectly aligned, and so a classifier would not be able to recognize that the object is a group. On the other hand, in the following frames the object is easier to recognize as such. Thus, in order to obtain a reliable classification the tracking algorithm has to wait that the classifier output becomes stable, before using it to take decisions.

Another problem related to groups is the loss of the identities of the constituent objects. An example is shown in Figure 3.8b, where objects 1 and 2 first join a group and then are separated again. When objects become separated, the tracking algorithm would incorrectly assign them new identities, if the original ones were not preserved and associated with the group. Note that in


**Figure 3.8** Occlusion related problems. In (a) the appearance of a group of people change from being identical to a single person to being clearly a group; if the classifier uses a classification result obtained in the first frame, it will continue to track the group considering it as an individual object. In (b) the system would not be able to correctly track the individual objects during all their life as a group if it attempts to exploit the uniformity of the motion, since the group can have strong changes in direction.

this case it would not have been possible to simply keep tracking separately the two objects using some kind of motion prediction until the end of the occlusion, because as a group the objects have performed a drastic change of trajectory (a 180° turn).

The analysis conducted in this Section about the typical problems in a real-world setting shows that in a lot of situations a tracking system cannot be able to correctly follow objects without additional information about their history. In the next subsection, we will see how the proposed FSA is able to provide this information.

#### 3.1.2.2 Object state management

The state manager has the task of maintaining and updating the FSA state of each object; the FSA state embodies the relevant information about the past history of the object, which can be used by the other parts of the tracking system. What pieces of information are actually relevant depends somewhat on the specific

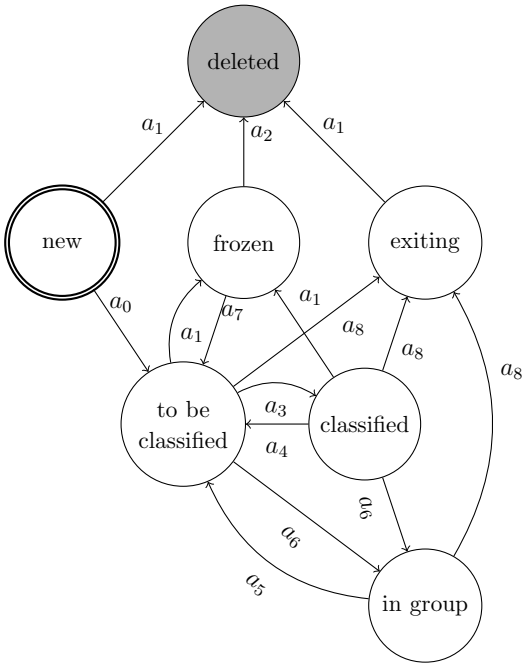



**Figure 3.9** The state diagram of the object state manager.

application; different problems may require the algorithm to keep different information in order to deal appropriately with them, and so may require an entirely different FSA.

Although we present only a single formulation of the FSA, the methodology remains general and easily extendable to other cases, since the knowledge about the states and the transitions between them is declaratively specified in the automaton definition, and not hidden within procedural code.

In order to deal with the issues discussed in Section 3.1.2.1, we propose a state manager based on the Finite State Automaton $\mathcal{A}$ depicted in Figure 3.9. It can be formally defined as:

$$\mathcal{A} = \langle S, \Sigma, \delta, s_0, F \rangle \tag{3.1}$$

where $S = \{s_0, \ldots, s_m\}$ is the set of the states; $\Sigma = \{a_0, \ldots, a_m\}$ is the set of the transition conditions, *i.e.* the conditions that may determine a state change; $\delta : S \times \Sigma \rightarrow S$ is the state-transition

| Id | Description |
|---|---|
| $s_0$ | new |
| $s_1$ | to be classified |
| $s_2$ | classified |
| $s_3$ | frozen |
| $s_4$ | in group |
| $s_5$ | exiting |
| $s_6$ | deleted |

(a)

| Id | Description |
|---|---|
| $a_0$ | obj is completely within the scene |
| $a_1$ | obj disappears from the scene |
| $a_2$ | obj does not reappear in the scene for a time $T_d$ |
| $a_3$ | obj classification is the same for two frames |
| $a_4$ | obj classification changes |
| $a_5$ | obj leaves the group |
| $a_6$ | obj occludes with one or more objects |
| $a_7$ | obj reappears inside the scene |
| $a_8$ | obj is not completely within the scene |

(b)

| | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ | $a_8$ |
|---|---|---|---|---|---|---|---|---|---|
| $s_0$ | $s_1$ | $s_6$ | - | - | - | - | - | - | - |
| $s_1$ | - | $s_3$ | - | $s_2$ | - | - | $s_4$ | - | $s_5$ |
| $s_2$ | - | $s_3$ | - | - | $s_1$ | - | $s_4$ | - | $s_5$ |
| $s_3$ | - | - | $s_6$ | - | - | - | - | $s_1$ | - |
| $s_4$ | - | - | - | - | - | $s_1$ | - | - | $s_5$ |
| $s_5$ | - | $s_6$ | - | - | - | - | - | - | - |

(c)

**Figure 3.10** The Finite State Automaton. (a) The set $S$ of the states. (b) The set $\Sigma$ of the transition conditions. (c) The state-transition function $\delta$; for entries shown as '-', the automaton remains in the current state.

function; $s_0 \in S$ is the initial state and $F \subset S$ is is the set of final states.

The proposed Finite State Automaton states and transitions are shown in Table 3.10. In particular, the set of states $S$ is shown in Table 3.10.a; we choose $s_0$ as initial state, since each object enters the scene by appearing either at the edge or at a known entry region (*e.g.* a doorway). Furthermore we choose $s_5$ as final state, since each object necessarily has to leave the scene. The set $\Sigma$ of transition conditions and the state-transition function $\delta$ are shown respectively in Table 3.10b and 3.10c.

It is worth noting that each state has been introduced in order to correctly solve one of the issues described earlier, as we will detail below. So it is possible to extend the FSA with the addition of other states and transitions, in order to deal with some other problems that should arise in a specific application context.

The meaning of the states and the conditions triggering the transitions are detailed below:


**Figure 3.11** Output of the tracking algorithm based on the proposed FSA, when applied to the problems discussed in Section 3.1.2.1. (a) The entering object is correctly recognized as a single object, and not a group. (b) The object identity is preserved when the person passes behind the tree. (c) A group initially classified as a single person is correctly handled when classification becomes stable. (d) The group object maintains the constituent object identities.

- *new* ($s_0$): the object has been just created and is located at the borders of the frame; if it enters completely, and so does not touch the frame borders ($a_0$), it becomes *to be classified*; otherwise, if it leaves the scene ($a_1$), it immediately becomes *deleted*.

  The introduction of *new* state solves the problem related to the instability of the entering objects, since it makes the system aware of such scenario and then capable to react in the best possible way, as shown in Figure 3.11a. Moreover, this state allows the algorithm to quickly discard spurious objects due to detection artifacts, since they usually do not persist long enough to become *to be classified*.

- *to be classified* ($s_1$): the object is completely within the scene, but its class is not yet considered reliable; if the classifier assign the same class for at least two frames ($a_3$), it becomes *classified*; if the association manager detects that the object has joined a group ($a_6$), it becomes *in group*; if the object disappears ($a_1$), it becomes *frozen*; if the object is leaving

the scene, *i.e.* it is not completely within it ($a_8$), it becomes *exiting*.

The *to be classified* state solves the issues of the objects entering the scene as group, discussed in Figure 3.8a. Thanks to this state, the class of the object is only validated when the system is sure about them. An example is shown in Figure 3.11c. Note that objects class is very important, since it makes the association manager able to take the correct decisions about the resolution of split and merge patterns.

- *classified* ($s_2$): the object is stable and reliably classified; if the classifier assigns a different class ($a_4$), it becomes *to be classified*; if the association manager detects that the object has joined a group ($a_6$), it becomes *in group*; if the object disappears ($a_1$), it becomes *frozen*; if the object is leaving the scene, then it is not completely within it ($a_8$), it becomes *exiting*.

  The distinction between *classified* and *to be classified* objects is used by the association manager when reasoning about split objects and group formation.

- *frozen* ($s_3$): the object is not visible, either because it is completely occluded by a background element, or because it has left the scene; if the object gets visible again ($a_7$), it becomes *to be classified*; if the object remains suspended for more than a time threshold $T_d$ ($a_2$), it becomes *deleted*; currently we use $T_d = 1$ sec.

  The *frozen* state avoids that an object is forgotten too soon when it momentarily disappears, as it happens in Figure 3.7.

- *in group* ($s_4$): the object is part of a group, and is no more tracked individually; its object model is preserved to be used when the object will leave the group; if the association manager detects that the object has left the group ($a_5$), it becomes *to be classified*; if the object is located at the borders of the frame ($a_8$), it becomes *exiting*.

The *in group* state has the purpose of keeping the object model even when the object cannot be tracked individually, as long as the algorithm knows it is included in a group. Thanks to this state, the proposed method is able to correctly solve the situation shown in Figure 3.8b, related to group objects.

- *exiting* ($s_5$): the object is located at the borders of the frame; if it disappears from the scene ($a_1$), it becomes *deleted*.

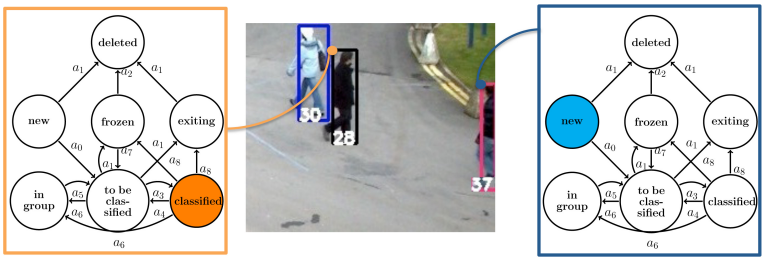
  The *exiting* objects differ from the *frozen* ones because of the system have not to preserve their identity, since their are leaving the scene;

- *deleted* ($s_6$): the object is not being tracked anymore; its object model can be discarded.

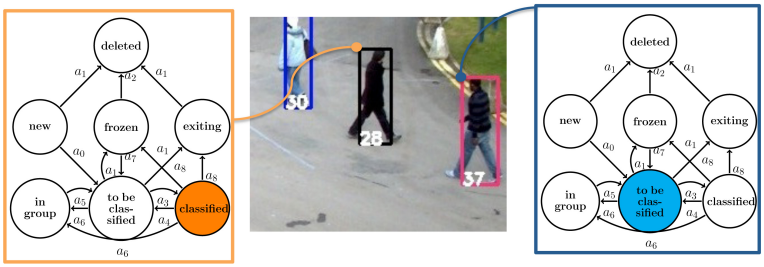
Figures 3.12 and 3.13 show a detailed example of how the object state management works.

#### 3.1.2.3 Object classification

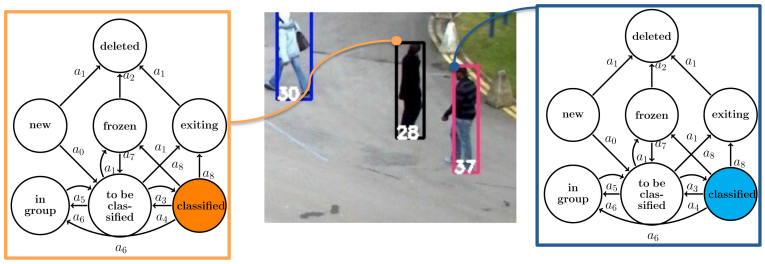
The tracking system needs an object classifier to determine if a blob corresponds to a group, an individual object, or an object part. In particular, two classes of individual objects have been considered in this thesis: person and baggage. We adopt a multi-class Support Vector Machine (SVM) classifier using the Histogram of Oriented Gradients (HOG) [103] as descriptor. HOG descriptor, which has already proved to be very effective for pedestrian detection, allows to describe the patterns by using the distribution of local intensity gradients or edge directions: the image is partitioned into *cells* and a local 1-D histogram of gradient directions or edge orientations over the pixels of each cell is computed. The accuracy of the descriptor is improved by contrast-normalizing the local histograms: a measure of the intensity across a larger region of the image, a *block*, is computed and it is used to normalize all the cells within the block. Such normalization makes the descriptor invariant in changes in illumination and shadowing. In Figure 3.14 we show the description of different entities using HOG.




*1.* The object 37 enters the scene as a *new* object while the object 28 is a *classified* object.

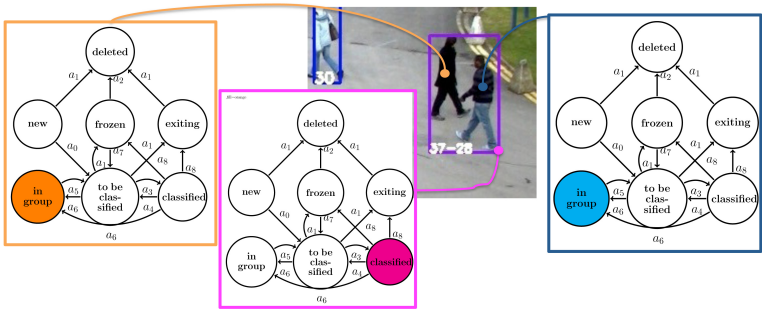



*2.* The object 37 becomes a *to be classified* object since it completely enters the scene.

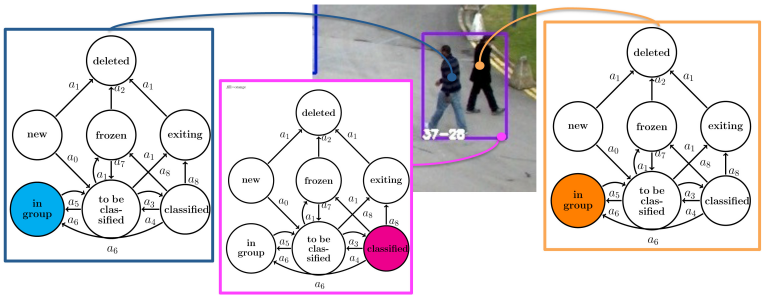



*3.* The classification as a person of the object 37 becomes reliable, then it becomes *classified*.

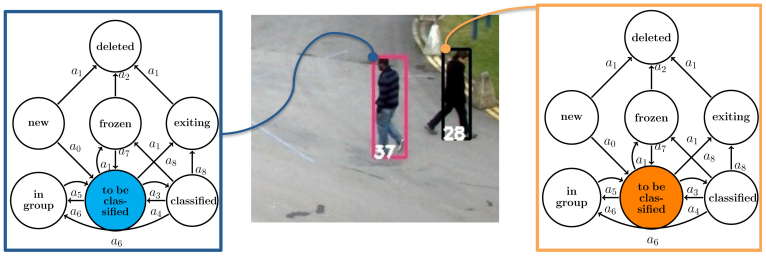
**Figure 3.12** Evolution of the states of the FSA of each object along a short frame sequence (part 1). The second part is in Figure 3.13.




*4.* After a few frames, the association manager shows an occlusion between objects 37 and 28. Both these objects become *in group* and a new group object is created.

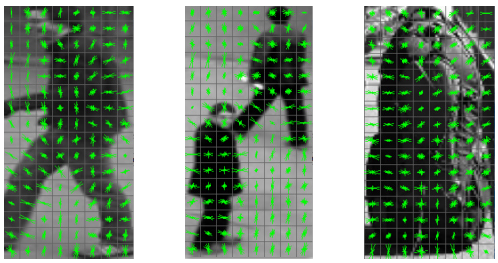



*5.* The objects 37 and 28 do not leave the group, then they do not change their states.




*6.* Finally, the group object splits and both the objects 37 and 28 become *to be classified*.

**Figure 3.13** Evolution of the states of the FSA of each object along a short frame sequence (part 2). The first part is in Figure 3.12.



(a) (b) (c)

**Figure 3.14** Examples of different entities by HOG descriptors: (a) a single person, (b) a small group of persons and (c) a backpack.

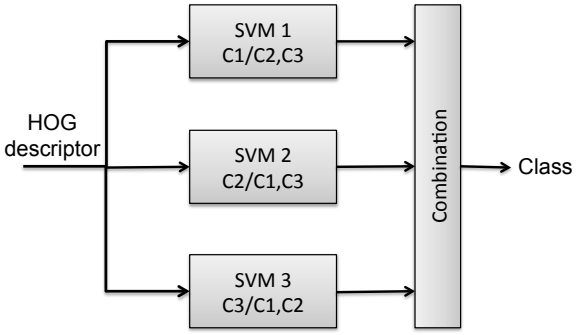



**Figure 3.15** Multi class SVM: the $i$-th classifier is labeled considered as positive samples the ones belonging to the $i$-th class, while as negative samples the ones belonging to the other classes.

Once extracted the features vector, a multi-class SVM has been applied. Being our problem multi-class, a $N$ *one-against-rest* strategy has been considered (see Figure 3.15); in particular N different classifiers, one for each class, are constructed. The $i$-th classifier is trained on the whole training data set in order to classify the members of $i$-th class against the rest. It means that the training set is relabeled: the samples belonging to the $i$-th class are labeled as positive examples, while samples belonging to other classes are labeled as negative ones. During the operating phase, a new object is assigned to the class with the maximum distance from the margin.

At this point one can note that object evolution is not dependent on its class (*e.g.* group or individual object), but only on its actual state. As a matter of fact, only object information is related to object class, while object state only determines the reliability of such information. In particular, for individual object we have information about appearance and shape: we consider the area and the perimeter of an object, its color histograms and its real dimensions, namely width and height, both obtained using an Inverse Perspective Mapping. Moreover we have information about the observed and predicted position of the object centroid. The predicted position is obtained using an extended 2D Position-Velocity (PV) Kalman Filter, whose state vector is:

$$\xi = \left[x_c, y_c, w, h, \dot{x}_c, \dot{y}_c, \dot{w}, \dot{h},\right] \tag{3.2}$$

where $(x_c, y_c)$ is the centroid of the object, $w$ and $h$ are the width and the height of the object minimum bounding box in pixels, $(\dot{x}_c, \dot{y}_c)$ and $(\dot{w}, \dot{h})$ are respectively the velocity of the object centroid and the derivative of the minimum bounding box size. It is worth noting that such a PV Kalman Filter is very effective when the object motion is linear and the noise has a Gaussian distribution.

*Group* objects contain also information about occluded objects. In this way the system can continue to track the *in group* objects when they leave the group.

#### 3.1.2.4 Association management

The aim of this module is to determine the correspondence between the set of blobs $B = \{b_1, ..., b_n\}$ and the set of objects $O = \{o_1, ..., o_m\}$, and properly update the information about the annotated objects: the current position is added to the trajectory, the appearance model updated and the state properly recomputed; this is made also in presence of detection errors: objects split into more parts, objects merged together or objects hidden by foreground elements.

To this aim, the system uses a graph based approach by taking into account the spatio-temporal information of each object and at

the same time reduces the computational cost needed to perform all the possible association: this is done by taking into account spatio-temporal information of each object; see details in Sections 3.1.2.4.2 and 3.1.2.4.1.

The associations between blobs and objects is represented by a matrix $T = \{t_{ij}\}$, where:

$$t_{ij} = \begin{cases} 0 & \text{if object } o_i \text{ is not associated to blob } b_j \\ 1 & \text{if } o_i \text{ is associated to } b_j \\ -1 & \text{if the pair } o_i, b_j \text{ hasn't been evaluated} \end{cases} \tag{3.3}$$

It can be obtained by evaluating the similarity matrix $S = \{s_{ij}\}$, being $s_{ij}$ the index of similarity between the blob $b_i$ and the object $o_j$. More details about the computation of this matrix will be provided in Section 3.1.2.5.

In simple situations, there is a one-to-one correspondence: the single blob $b_i$ is associated to the single object $o_i$, as shown in Figure 3.20a. However, in presence of split or merge, the association manager needs to take into account more complex associations (one-to-many, many-to-one, and even many-to-many).

To this purpose, we evaluate the similarity matrix over an augmented set of blobs $B^I$ and objects $O^I$ in order to take into account all those above mentioned situations:

$$B^I = B \cup B_d; \quad O^I = O \cup O_d. \tag{3.4}$$

$B_d$ and $O_d$ are respectively the set of derived blobs and derived objects, virtually created at the current frame; their introduction allows the system to simulate all the possible splits and occlusions occurring in real scenarios, so as to take the best possible decision in terms of association between one or more blobs and one or more objects.

In order to clarify this, consider two objects $o_1$ and $o_2$ meeting in the scene: at the frame $t-1$ the algorithm correctly tracks these objects one by one. At the frame $t$, a merge occurs and the detection phase detects a single blob $b_A$ (instead of two). The similarity $s_{A1}$ between the blob $b_A$ and the object $o_1$ and the similarity $s_{A2}$

between the blob $b_A$ and the object $o_2$ are very low, and then a simple association manager could fail. Thanks to the introduction of the derived object $o_{1\cup2}$, composed by the union of the objects $o_1$ and $o_2$, a merge situation is simulated. In this way, the similarity between the blob $b_A$ and the derived object $o_{1\cup2}$ is very high, and then the association can be correctly performed. The states of $o_1$ and $o_2$ are updated to *in group* and their information are stored inside the *group* object $o_{1\cup2}$, whose state is initialized to *classified*. In this way, the system is able to correctly track the two objects as a single group object.

**3.1.2.4.1 Derived Blobs and Objects Creator** This modules generates the sets $B_d$ and $O_d$ of derived blobs and objects. The most simple, but also the most inefficient way, to perform this task is to evaluate all the possible combinations of $k$ blobs (objects), with $k = 1...n(m)$.

$$B_d = \{b_1b_2, ..., b_1b_n, ..., b_1b_2b_n, ...\}; \tag{3.5}$$

$$O_d = \{o_1o_2, ..., o_1o_m, ..., o_1o_2o_m, ...\}. \tag{3.6}$$

The inclusion of all the combinations determines a very high computational cost and an explosion of the size of the similarity matrix; note that the number of possible combinations $C$ for $n$ blobs and $m$ objects is:

$$C = \sum_{i=1}^{n} \binom{n}{i} \cdot \sum_{j=1}^{m} \binom{m}{j}. \tag{3.7}$$

Consider that if the scene is populated by only 10 objects and the detection phase finds 10 blobs, we need to verify more than 1000 possible associations (1274). In order to decrease the number of associations to be evaluated, we propose to exploit the spatio-temporal continuity of the tracked objects so as to select only the subset of feasible combinations.

In particular, the following heuristics have been considered to obtain the sets $B_d$ and $O_d$:

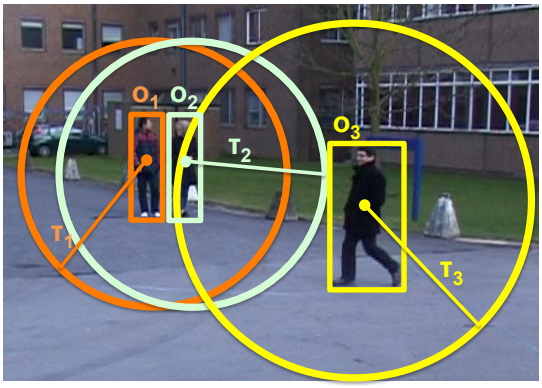



**Figure 3.16** Heuristic for pruning the derived blobs.

- The distance between the centroids of the blobs (objects respectively) composing a derived blob (object) has to be less than an adaptive threshold $\tau$. $\tau$ is dynamically chosen and depends on the maximum velocity of objects representing the maximum displacement (in pixels) of an object between two frames. This value is strongly related to the position of the object inside a scene: once fixed the maximum real velocity of an object inside the scene, the maximum distance $d_{max}$ in pixel can be computed by means of the IPM algorithm: $\tau = \alpha \cdot d_{max}$, being $\alpha$ a weight set in our experiments to 2, which determines the area where find out possible split and merge of blobs. An example is depicted in Figure 3.17: the only derived objects that the system can evaluate are $o_1o_2$ and $o_2o_3$, so discarding the pair $o_1o_3$, whose distance between the centroids is over the thresholds $\tau_1$ and $\tau_3$.

- The reciprocal position of the blobs (objects) is taken into account. For instance, starting from the situation depicted in Figure 3.17a, only the combination $b_1b_2b_3$ (Figure 3.17b), $b_1b_2$ (Figure 3.17c) and $b_2b_3$ (Figure 3.17d) can be accepted, while the association $b_1b_3$ (Figure 3.17e) doesn't make sense, since it is an impossible merge, implicitly including $b_2$. Note that in this very simple situation, often recurring also in real

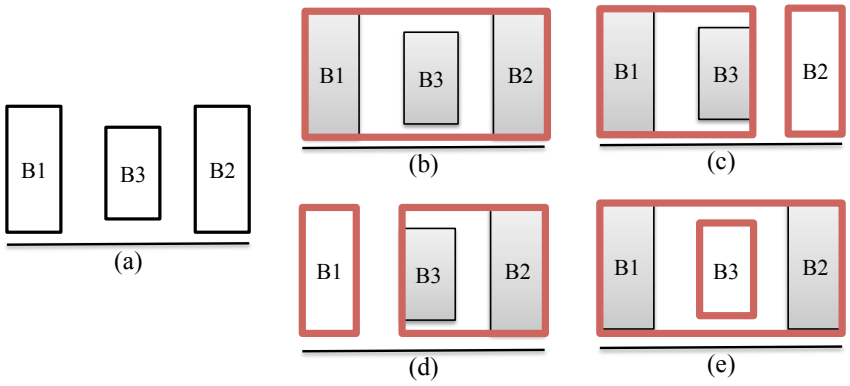



**Figure 3.17** Examples of feasible (b,c,d) and unfeasible (e) merge of the boxes in (a) for the composition of derived boxes.

scenarios, we are able to obtain a 25% decrease in the number of association to deal with; furthermore, it does not affect the reliability of the system, since it only discards unfeasible associations.

The main steps of the algorithm for obtaining the set of derived blobs $B_d$ are shown in Figure 3.18. A similar algorithm applies for derived objects $O_d$.

The algorithm starts by computing the distance between the boxes belonging to $B$, whose relative centroids distance is less than $\tau$, so obtaining the weighted undirected graph $G = \{V, E, w\}$; the vertices $V = \{V_1, V_2, ..., V_n\}$ are associated to blobs, while the edges $E = \{E_1, E_2, ..., E_k\}$ to the distances between the blobs. Each edge is associated to a weight $e_{i,j}$ corresponding to the distance between the blob $b_i$ and the blob $b_j$. Note that if the distance between two blobs is under $\tau$, then the corresponding edge does not belong to $E$.

For each blob $b_i$, the shortest path to reach all the other blobs belonging to $B$ and not yet explored is computed by using the Dijkstra's algorithm; it is implemented by a min priority queue with a Fibonacci heap [104], so that the global computational complexity of the algorithm is $O(|E| + |V| \cdot \log |V|)$.

For each found shortest path, a new derived blob $b_d$ is created,

```
procedure B_d = FindDerivedFeasibleBlobs(Blobs B)
 B_d={ }
 d := ComputeDistancesBetweenCentroids(B)
 graph := ComputeGraph(B,d)
 foreach b_i in B
    foreach b_j in B, j > i
        path := ComputeShortestPath(graph, b_i, b_j)
        b_d := CreateBlob(graph, path)
        B_d := AddBlob(graph,b_d)
end procedure
```

**Figure 3.18** The algorithm for finding out the derived feasible blobs (objects).

composed by the blobs corresponding to the vertices crossed during the path. An example is shown in Figure 3.19: in this case, the method reduces by 40% the number of derived objects (the combinations $b_2b_3$, $b_2b_4$, $b_1b_4$ and $b_2b_3b_4$ are not generated).

**3.1.2.4.2 The algorithm** The algorithm operates in two distinct phases, as shown in Figure 3.21: in the first one, it finds the correspondence for stable objects (*i.e.* objects in the *to be classified*, *classified* or *frozen* state); in the second phase it tries to assign the remaining blobs to objects in the *new* state, possibly creating such objects if necessary. The motivation for this distinction is that the management of split-merge and occlusions can be performed only for stable objects, since for new ones the system would not have enough information to do it in a reliable way.

During the first phase, the algorithm starts by choosing the maximum element $s^*_{ij}$ of the matrix, corresponding to the blob $b_i$ and to the object $o_j$ and records the corresponding associations in $T$. At each step, the algorithm selects the pair $b_i$–$o_j$ such that $s_{ij}$ is the maximum value corresponding to $t_{ij} = -1$ and records the obtained associations.

The following situations can occur:

***one-to-one correspondence***: $b_i \in B$, $o_j \in O$ (see Figure 3.20a). The associations corresponding to the blob $b_i$ and to the object $o_j$ need to be updated, together with all the derived blobs

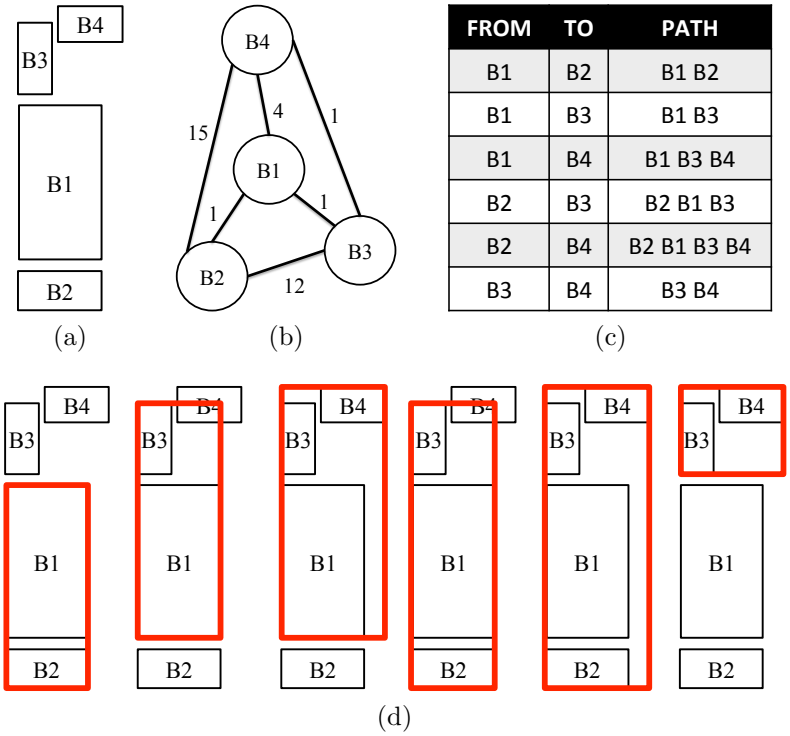
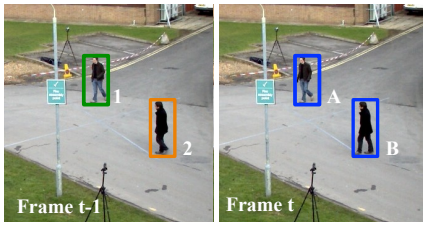



**Figure 3.19** An example of the algorithm in charge of creating derived blobs (objects) for a split pattern: starting from the detected blobs in (a), the algorithm creates the graph in (b) and finds the shortest paths in (c). The output of the algorithm is in (d).

and objects containing $b_i$ and $o_j$; in particular:

$$\begin{aligned} &t_{i,j} = 1; \\ &t_{z,j} = 0 \quad \forall\, b_z \in B^I,\ b_z \neq b_i; \\ &t_{z,k} = 0 \quad \forall\, b_z \supset b_i,\ \forall\, o_k \in O^I; \\ &t_{i,k} = 0 \quad \forall\, o_k \in O^I,\ o_k \neq o_j; \\ &t_{z,k} = 0 \quad \forall\, o_k \supset o_i,\ \forall\, b_z \in B^I; \end{aligned} \tag{3.8}$$

***one-to-many correspondence***: $b_i \in B$, $o_j \in O_d$ (see Figure 3.20b). In this case the occlusion pattern has to be solved. The associations corresponding to the blob $b_i$ and to the object $o_j$ need

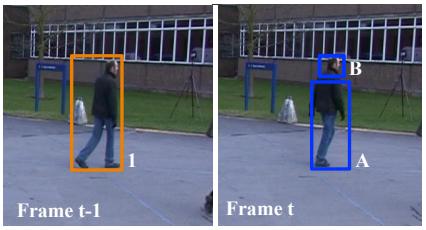



| | Obj 1 | Obj 2 | Obj 1 U 2 |
|---|---|---|---|
| Blob A | 0,96 | 0 | 0,32 |
| Blob B | 0 | 0,93 | 0,26 |
| Blobs A U B | 0,25 | 0,31 | 0,63 |

(a)

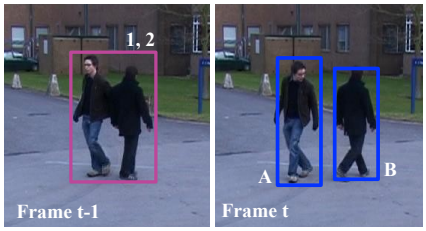

| | Obj 1 | Obj 2 | Obj 1 U 2 |
|---|---|---|---|
| Blob A | 0,48 | 0,53 | 0,86 |

(b)



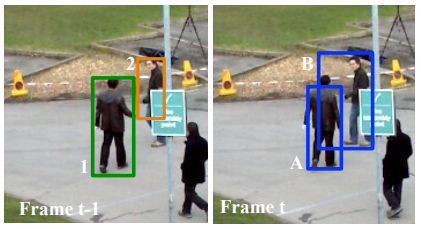

| | Obj 1 |
|---|---|
| Blob A | 0,65 |
| Blob B | 0,25 |
| Blobs A U B | 0,89 |

(c)




| | Obj 1,2 |
|---|---|
| Blob A | 0,38 |
| Blob B | 0,45 |
| Blobs A+B | 0,89 |

(d)




| | Obj 1 | Obj 2 | Obj 1 U 2 |
|---|---|---|---|
| Blob A | 0,59 | 0,20 | 0,43 |
| Blob B | 0,62 | 0,53 | 0,71 |
| Blobs A U B | 0,43 | 0,27 | 0,88 |

(e)


| | Obj 1 | Obj 2 | Obj 1 U 2 |
|---|---|---|---|
| Blob A | 0,90 | 0,15 | 0,23 |
| Blob B | 0,62 | 0,89 | 0,18 |
| Blobs A U B | 0,25 | 0,27 | 0,91 |

(f)

**Figure 3.20** Different kinds of associations: one-to-one association (a), one-to-many association (b), many-to-one associations (c,d), many-to-many associations (e,f).

to be updated, together with all the associations of the objects composing the derived object $o_j$:

$$\begin{aligned} & t_{i,j} = 1; \\ & t_{z,j} = 0 \quad \forall\ b_z \in B^I,\ b_z \neq b_i; \\ & t_{z,k} = 0 \quad \forall\ b_z \supset b_i,\ \forall\ o_k \in O^I; \\ & t_{i,k} = 0 \quad \forall\ o_k \in O^I,\ o_k \neq o_j; \\ & t_{z,k} = 0 \quad \forall\ o_k \subset o_j,\ \forall\ b_z \in B^I; \end{aligned} \tag{3.9}$$

Furthermore, the state of $o_j$ is initialized to *classified* while the state of the objects composing $o_j$ is updated to *in group*; finally, an instance of each object is stored inside the derived object, which is classified as a *group* object. Starting from this frame, the object $o_j$ will be tracked as a group until it will split in the next frames.

***many-to-one correspondence***: $b_i \in B_d$, $o_j \in O$. In this case a split problem has to be solved. Two different situations have to be taken into account; if $o_j$ is classified as a person (see Figure 3.20c), then the associations corresponding to the blob $b_i$ and to the object $o_j$ need to be updated, together with all the associations of the blobs composing the derived blob $b_i$:

$$\begin{aligned} & t_{i,j} = 1; \\ & t_{z,j} = 0 \quad \forall\ b_z \in B^I,\ b_z \neq b_i; \\ & t_{z,k} = 0 \quad \forall\ b_z \subset b_i,\ \forall\ o_k \in O^I; \\ & t_{i,k} = 0 \quad \forall\ o_k \in O^I,\ o_k \neq o_j; \\ & t_{z,k} = 0 \quad \forall\ o_k \supset o_i,\ \forall\ b_z \in B^I; \end{aligned} \tag{3.10}$$

A different situation occurs when the object $o_j$ is classified as a group (see Figure 3.20d), since the split refers to the end of an occlusion pattern. The system, by exploiting the set of occluded objects (1 and 2 in Figure 3.20d) and their history, does not solve the split but it is able to correctly associate each blob to the most similar object. A similarity matrix based approach is exploited in this case in order to find the best association between occluded objects and blobs.

***many-to-many correspondence***: $b_i \in B_d$, $o_j \in O_d$ (see Figure 3.20e). This situation arises when a merge and a split contemporaneously happen. In this case we decide to create a group object, so to avoid any kind of decision which could be reveal too risky. In this scenario, the associations corresponding to the blob $b_i$ and to the object $o_j$ need to be updated, together with all the associations of the blobs composing the derived blob $b_i$ and the objects composing the derived object $o_j$:

$$\begin{aligned} & t_{i,j} = 1; \\ & t_{z,j} = 0 \quad \forall\ b_z \in B^I,\ b_z \neq b_i; \\ & t_{z,k} = 0 \quad \forall\ b_z \subset b_i,\ \forall\ o_k \in O^I; \\ & t_{i,k} = 0 \quad \forall\ o_k \in O^I,\ o_k \neq o_j; \\ & t_{z,k} = 0 \quad \forall\ o_k \subset o_i,\ \forall\ b_z \in B^I; \end{aligned} \tag{3.11}$$

Furthermore, as in the *one-to-many correspondence*, the state of $o_j$ is initialized to *classified* while the state of the objects composing $o_j$ is updated to *in group* and their information are stored inside the derived object, which is classified as a *group* object.

A many-to-many correspondence can also arise when two or more objects are very near each other: as a matter of fact, the similarity between the derived object and the derived blob can be only slightly higher than the similarity between the single objects and the single blobs (see Figure 3.20f). If it happens, each object is associated to the blob with the higher similarity.

During the second phase, only the one-to-one association is performed. For this reason, the algorithm follows a similar scheme, except that it considers only the objects in the *new* state, and does not create derived blobs and derived objects. Moreover, the similarity matrix is built using less features than in the first phase since we have experimentally verified that only the position information is sufficiently reliable for such objects. At the end of this phase, any remaining unassigned blobs are used to create new annotated object, initialized to the *new* state.

```
procedure TrackingAlgorithm (annotated_objs, blobs)
 AssocStableObjs (annotated_objs, blobs)
 pending_blobs := SearchPendingBlobs(blobs)
 unassoc_objs := SearchUnassocObjs(annotated_objs)
 AssocInstableObjs (unassoc_objs, pending_blobs)
 unassoc_blobs := SearchPendingBlobs(pending_blobs)
 CreateObjFromPendingBoxes (annotated_objs,unassoc_blobs)
 UpdateObjectsState(annotated_objs)
end procedure

procedure AssocInstableObjs (annotated_objs, blobs)
 sim_mat := ObjInstableSimMatrix (annotated_objs, blobs)
 foreach obj in annotated_objs:
         (best_boxes, best_objs) := BestAssoc(sim_mat)
 end
end procedure

procedure AssocStableObjs (annotated_objs, blobs)
 derived_objs:=CreateDerivedFeasibleObjs(annotated_objs)
 derived_blobs := CreateDerivedFeasibleBlobs(blobs)
 all_objs := JoinObjs(annotated_objs, derived_objs)
 all_blobs := JoinBlobs(blobs, derived_blobs)
 sim_mat := ObjStableSimMatrix (all_objs, all_blobs)
 foreach obj in all_objs:
        (best_boxes, best_objs) := BestAssoc(sim_mat)
 end
end procedure
```

**Figure 3.21** Structure of the algorithm for stable and unstable objects associations.

#### 3.1.2.5 Similarity evaluation

As already mentioned, the similarity matrix is used to match one or more blobs with one or more objects.

In order to measure the similarity between an object $o_i$ and a blob $b_j$, the tracking system uses an index based on three kinds of information: the position, the shape and the appearance:

$$s_{ij} = \sqrt{\frac{\alpha_p \cdot (s_{ij}^p)^2 + \alpha_s \cdot (s_{ij}^s)^2 + \alpha_a \cdot (s_{ij}^a)^2}{\alpha_p + \alpha_s + \alpha_a}} \tag{3.12}$$

As described below, $s_{ij}$ values identify similarity metrics and $\alpha$ values are weights chosen according to the state of the object and the association management phase. In particular:

- $s_{ij}^p$ is the position similarity index, computed as the distance between the estimated centroid of an object $o_i$ and the centroid of a blob $b_j$;

- $s_{ij}^s$ is the shape similarity index between an object $o_i$ and a blob $b_j$;

- $s_{ij}^a$ is the appearance similarity index between an object $o_i$ and a blob $b_j$, based on color histograms;

- $\alpha_p$, $\alpha_s$ and $\alpha_a$ are the weights of position, shape and appearance similarity index respectively;

All $\alpha$ values have been chosen by experimentation over a training set. Namely, in the first phase, selected values for objects in the *to be classified* and *classified* state are $\alpha_p = \alpha_s = \alpha_a = 1$ while for objects in the *in group* state selected values are $\alpha_s = \alpha_a = 1$; $\alpha_p = 0$ since in this context shape and appearance similarity perform better than position one. Finally, in the second phase that evaluates *new* objects, we choose to consider the only reliable feature, namely the position. Thus selected $\alpha$ values are $\alpha_s = \alpha_a = 0$; $\alpha_p = 1$.

For the position, as already seen, the system uses a Kalman filter, based on a uniform velocity model, to predict the coordinates of the object centroid at the current frame. The predicted coordinates are compared with the blob centroid, using Euclidean distance, in order to obtain for each object $o_i$ and each blob $b_j$ the distance $d_{ij}$. The position similarity index is then computed as:

$$s_{ij}^p = 1 - d_{ij}/d_{\max} \tag{3.13}$$

where $d_{\max}$ is a normalization factor depending on the maximum velocity of objects representing the maximum displacement of an object between two frames.

For characterizing the shape similarity, the system uses the real height and the area of the blob and of the object model; in particular if we denote as $\Delta h_{ij}$ the relative height difference

between $o_i$ and $b_j$, and as $\Delta A_{ij}$ the relative area difference, the considered shape similarity index is:

$$s_{ij}^{s} = 1 - \sqrt{\frac{|\Delta A_{ij}| + (\Delta h_{ij})^2}{2}} \tag{3.14}$$

Finally, as a representation of the appearance we have used the color histograms computed separately for the upper half and for the lower half of the object or blob (*Image Partitioning*). We have experimented with several criteria for comparing the histograms, and we have found that the most effective value is the $\chi^2$ distance:

$$q_{ij} = \frac{1}{M} \sum_{k} \frac{\left(h_i^o(k) - h_j^b(k)\right)^2}{h_i^o(k) + h_j^b(k)} \tag{3.15}$$

where index $k$ iterates over the bins of the histogram, $h_i^o$ is the histogram of object $o_i$, $h_j^b$ is the histogram of blob $b_j$, and $M$ is the number of bins. The appearance similarity index is:

$$s_{ij}^{a} = 1 - \sqrt{\frac{\left(q_{ij}^{up}\right)^2 + \left(q_{ij}^{low}\right)^2}{2}}. \tag{3.16}$$

where $q_{ij}^{up}$ is the value of $q_{ij}$ computed using only the upper half of the object/blob, and $q_{ij}^{low}$ is the value computed using only the lower half.

#### 3.1.2.6 Conclusion

In this section I presented the novel tracking algorithm introduced in this thesis, able to overcome many of the problems induced by the object detection phase, as well as to deal with total or partial occlusions. In order to confirm these characteristics, the algorithm has been tested over two standard datasets, namely PETS and IS-SIA Soccer. As I will show in Section 5.1, the obtained results confirm the effectiveness of the proposed approach in very different environments. Furthermore, this algorithm participated to an international competition on surveillance applications, PETS

2013 Contest [105], and the obtained results confirms that it is a very promising algorithm. Furthermore its low computational cost makes this algorithm well suited for real-time applications for behavior analysis.

## 3.2 Visual Behavior Analysis

Once trajectories have been extracted (by using the algorithm detailed in Section 3.1), the proposed system has to analyze them in order to detect dangerous behaviors and then alert the human operator. This responsibility is assigned to the Visual Behavior Analysis module: this module is able to identify during a preliminary learning step typical behaviors occurring in the scene by a clustering algorithm, especially designed for this purpose. One of the main advantages lies in the fact that this step is performed into an unsupervised way, without requiring any knowledge of the human operator about the particular environment.

The extracted clusters will be used during the operating phase; in particular, the following interactions with the system are allowed to the human operator: an abnormal behavior framework, which allows the human operator to be informed of abnormal behaviors occurring in the scene, and a query by sketch application, 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.

The learning step will be analyzed in this section, while the proposed applications (abnormal behavior framework and query by sketch) will be analyzed in Section 3.4.

### 3.2.1 Preliminaries

As mentioned before, the aim of a tracking algorithm is to extract the trajectory of objects populating a scene. A generic trajectory $T^k$ can be represented as a sequence of $n$ three dimensional points:

$$T^k =< P_1^k, P_2^k, ..., P_n^k >, \tag{3.17}$$

being the generic point a triple containing the spatial information $(x_i^k, y_i^k)$ of the object at the time instant $t_i^k$:

$$P_i^k = (x_i^k, y_i^k, t_i^k) \quad \forall i \in \{1, ..., n\}. \tag{3.18}$$

The analysis of raw data for behavior analysis has two main drawbacks: first, raw data are more sensible to noise and tracking errors, and thus a filtering of each trajectory is needed before use. Second, if a system considers the similarity between raw 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.

As mentioned in Section 2.2, this problem is often deal with by representing the trajectory as a reduced sequence of symbols. In this thesis in particular I decided to represent a trajectory as a string, aiming to preserve only the discriminant information and to reduce the space required to store it.

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 a highway, the most discriminant feature is the sequence of zones crossed by the moving object.

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 and at the same time efficient way could be to partition the scene using a fixed-size uniform grid [106]. 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, where most of trajectories lies and for which we need a 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 3.2.3). As a consequence of this partitioning criterion, areas in the scene where most of trajectories lies are represented with a higher number of zones.

### 3.2.2 Overview

Before entering in the details of the proposed approach, let's briefly summarize the main steps required during the learning phase, as well as the motivation behind the choice of an unsupervised approach.

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 normal(or abnormal) behaviors. On the other hand, The extraction of prototypes for (ab)normal trajectories can be performed by following one of the following two strategies: supervised and unsupervised. Techniques trained in *supervised* mode 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 real scenario because of its intrinsic complexity.

Techniques operating in *unsupervised* mode [107] do not require labeled data since they make the implicit assumption that normal instances are far more frequent than abnormal ones. The main advantage of 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.

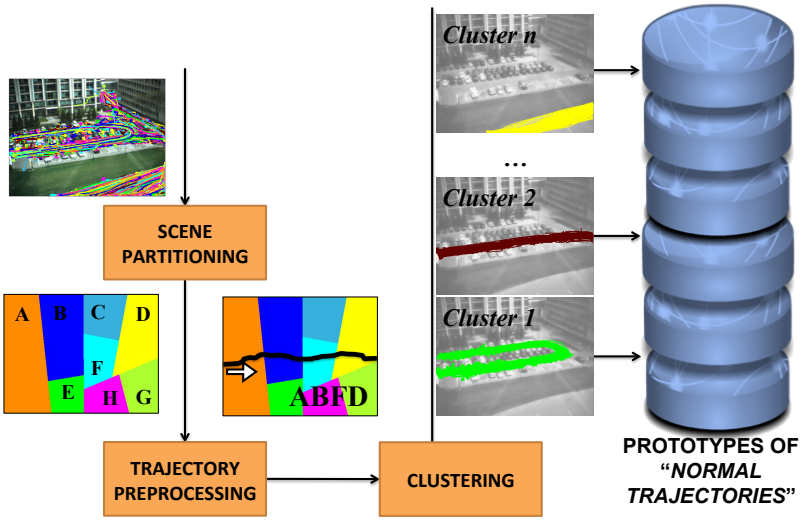



**Figure 3.22** Learning phase for abnormal recognition.

This is a very important and not negligible feature, since it allows the system to autonomously understand typical patterns within a scene, so avoiding the boring and expensive labour of labeling a large amount of trajectories data.

In order to make this step human-independent, I 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 a model of normal trajectories that the system already knows.

The method we propose is based on the following steps, as shown in Figure 4.1:

- **Scene Partitioning**: in order to reduce the large amount of data to be managed, the scene is partitioned into zones and the trajectories are represented by considering the zones crossed during their life: in particular, the zones are dynamically defined according to the distribution of the extracted trajectories, so that areas where most of the trajectories lie

are represent by an higher number of zones. The scene partitioning algorithm introduced in this thesis is detailed in Subsection 3.2.3.

- **Trajectories preprocessing**: each trajectory is represented as a sequence of symbols, namely a string. Each symbol contains those discriminant features (related to position, in terms of crossed zones, speed and shape in each zone) required for discriminate two different trajectories (see Section 3.2.4). The similarity between two trajectories is evaluated by using a kernel-based method. The main advantage in this choice lies in the fact that it is possible to combine the kernel with a large class of clustering and machine learning algorithms, which can be expressed using only scalar product between input data (see Subsection 3.2.5).

- **Clustering**: given the kernel, a novel tree-based clustering algorithm is applied in order to extract clusters of trajectories inside the scene, each corresponding to a typical normal path. The choice of an unsupervised method is justified by the fact that in this way the knowledge of the human operator about the particular environments is not necessary. The proposed clustering method is detailed in Subsection 3.2.6.

### 3.2.3 Scene Partitioning

In this section we will introduce the method proposed in this thesis for partitioning the scene into a fixed number of zones. As previously mentioned, its aim is to represent the areas where most of trajectories lies with an higher number of zones, in order to increase the discriminative power of the string-based representation.

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 [108]. Each zone $(Z_i)_{i\in\{1,\dots,L\}}$ is represented by using its statistical properties (mean,

```
procedure PartitionScene
        (List<Trajectory> LT, Scenario S, ZonesNumber N)

 current_zones_number = 0
 Tree <Zone> TZ <- InitializeTree(S)

 while (current_zones_number < N)
        TZ <- DefineStatisticalProperty(TZ, LT)
        Z <- SelectZoneToSplit(TZ)
        c_a <- SelectCuttingAxis(Z)
        c_p <- SelectCuttingPosition(Z, c_a)

        [Z1, Z2] <- CutZone(Z, c_a, c_p)
        TZ <- Add_Leafs (TZ, Z, Z1, Z2)

        current_zones_number = current_zones_number+1
 end
end procedure
```

**Figure 3.23** The structure of the scene partitioning algorithm.

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*). The algorithm is briefly summarized in Figure 3.38. In the following, more details about this algorithm will be provided since an enhanced kernelized version of it will be introduced in Subsection 3.2.6.

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 $\mu_i$ and Covariance $Cov_i$):

$$\begin{aligned} |Z_i| &= \sum_{P \in Z_i} f(P) \qquad \mu_i = \frac{\sum_{P \in Z_i} f(P) P}{|Z_i|} \\ Cov_i &= \frac{\sum_{P \in Z_i} f(P) P \cdot P^t}{|Z_i|} - \mu_i \cdot \mu_i^t. \end{aligned} \tag{3.19}$$

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 objective can be achieved by

minimizing the total squared error $TSE$ induced by the partition $\rho = \{Z_1, ..., Z_L\}$:

$$TSE(\rho) = \sum_{i=1}^{L} SE(Z_i), \tag{3.20}$$

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

$$SE(Z_i) = \sum_{P \in Z_i} ||\mu_i - P||^2 \cdot f(P). \tag{3.21}$$

Since the set of all possible partitions into $L$ zones is too large for an exhaustive enumeration for its very high computational cost, being

$$\frac{1}{L!} \sum_{l=0}^{L} (-1)^{L-l} \binom{L}{l} l^L, \tag{3.22}$$

an heuristic needs to be applied. In particular, we decided to use the following heuristic, detailed in [109]. 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:

- 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 3.24, 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. The main idea is to find the best compromise between the computational time required by this operation and the final quantization error. As shown in [110], this objective can be reached

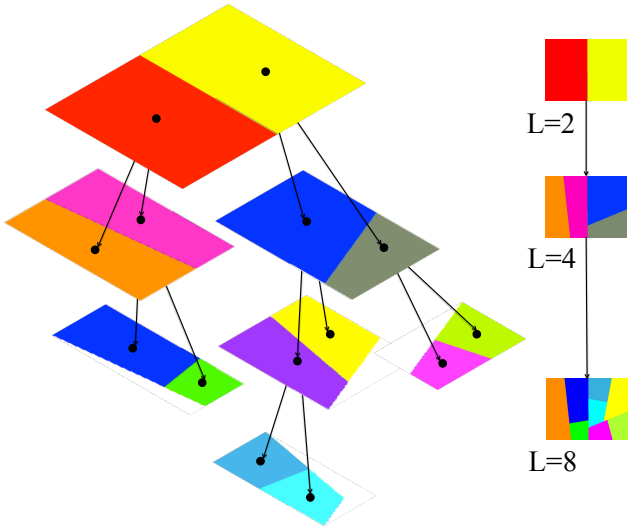



**Figure 3.24** Example of a tree-structured vector quantization obtained recursively partitioning the scene into $L = 8$ zones.

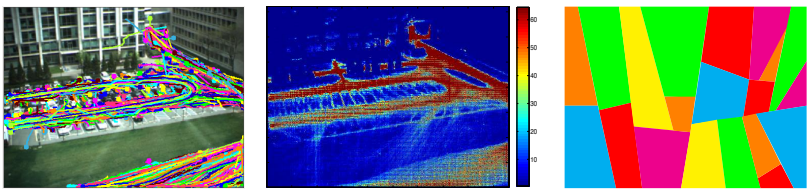
by minimizing $TSE(\rho)$: for this reason, the algorithm splits the zone $Z$ with the maximum squared error, since its contribution to $TSE(\rho)$ is the largest.

**Cutting Axis**: Given a zone $Z_i$ to be split, we need to determine the location of the *cutting plane*. As in [109], we decide to split along the axis with the greatest variance, namely the major axis.

**Cutting Position**: The optimal cutting position $T^\star$ 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:

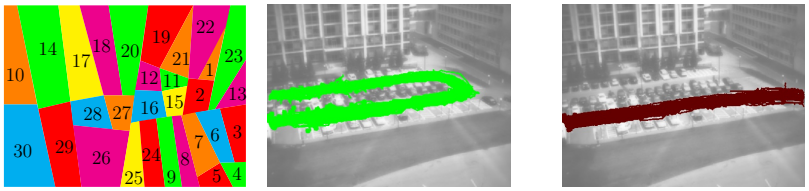
$$SE(Z_i) - \left[SE(Z_i^1) + SE(Z_i^2)\right] \tag{3.23}$$

Let $m$ and $M$ be respectively the minimum and the maximum projections of $Z_i$ on the cutting axis. It can be proved [109] that the maximization of Equation 3.23 can be reached by computing



(a) (b) (c)

**Figure 3.25** 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 *L=20* (c).




(a) (b) (c)

**Figure 3.26** Partition of the scene with *L=30* (a). Examples of two clusters: (b) composed by 30-29-28-27-16-15-(2)-(21)-11-12- 20-18-17-14-10 and (c) by 30-29-28-27-(20)-16-15-2-23-13. Symbols in brackets are the ones occurring only in a few trajectories.

the optimal cutting position $T^\star$ as follows:

$$T^\star = \arg \max_{T \in [m,M]} \left[ \frac{\delta(T)}{1-\delta(T)} ||\mu_i - \mu_i^1||^2 \right], \tag{3.24}$$

where $\mu_i$ and $\mu_i^1$ denote respectively the mean of $Z_i$ and $Z_i^1$ and $\delta(T)$ is defined as $\delta(T) = \frac{|Z_i^1|}{|Z_i|}$. Note 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 3.25: starting from the trajectory distribution in Figures 3.25a and 3.25b, we show the partition of the scene by using $L = 20$.

### 3.2.4 Trajectory representation

Once partitioned the scene into zones, a trajectory can be segmented into $l$ segments: $T = \{< S_1, ..., S_l >\}$, where the $j$-th segment $S_j$ is defined as the sequence of points lying in the same zone $Z_k$: $S_j = \{< P_a, ...P_b > | P_i \in Z_k\}$.

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 $T = \{< \alpha(S_1), ..., \alpha(S_l) >\}$.

In order to clarify this concept and to emphasize the importance of the chosen partitioning strategy, an example is shown in Figure 3.26. In particular, two different clusters and the corresponding representations are reported.

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 $\theta(\bullet)$. The introduction of these information is very important, since it allows to distinguish those trajectories which are similar only in terms of position: think, as an example, to two vehicles crossing the highway with a different speed, namely 120 and 250 km/h. It is evident that the last behavior should be identified as an abnormal one.

In particular, we use the Bernstein Polynomial Approximation to model each trajectory into a zone; the polynomial degrees are bounded by $c_s = 3$ and $c_v = 2$ for the polynomials representing respectively $s$ and $v$. The operator $\theta(\bullet)$ is then computed as the vector composed by the obtained coefficient $a_i^s$ and $a_i^v$: $\theta(S) = [a_1^s, ..., a_{c_s}^s, a_1^v, ..., a_{c_v}^v]$.

Thanks to this representation, each trajectory is encoded by $T = \{< \alpha(S_1), ..., \alpha(S_l) >, < \theta(S_1), ..., \theta(S_l) >\}$.

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

3.2.5) allows to weight both types of information.

### 3.2.5 Trajectory similarity

The definition of a similarity measure is a very complex task, since of course it depends on the particular application field we are interested in. In this thesis we are interested in evaluating the similarity between trajectories: intuitively, two trajectories can be considered similar to each other if are close enough, have the same direction and spend the same time inside the scene. We can note that the representation introduced in Section 3.2.4 is able to take into account all the required information. On the other hand, it is evident that this representation results in a complex strategy to verify the similarity between trajectories.

A very promising technique often used for evaluating the similarity between strings or time series in different contexts is the application of kernel-based approaches. The main idea behind such approach is to map the data into a high dimensional feature space, where each coordinate corresponds to one feature of the data items. The main advantage lies in the fact that kernel functions enable to operate in the feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space (the well known kernel trick). This operation is in general computationally cheaper than the explicit computation of the coordinates, making such approaches especially suited for real-time applications.

In the last years a family of similarities based on dynamic programming has been taken into account to construct kernels for evaluating the similarity between strings or time series and has been successfully applied to different applications fields, ranging from bioinformatics to text-processing (for instance the Dynamic-Time-Warping [111] and the Smith Waterman algorithm [112]). However, as stated in [113], 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 define

a metric, which is an important requirement of kernel machines. In order to face these problems, we propose a novel similarity metric based on kernels: the main advantage is that the problem can 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 [114]. The main idea of 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 $(\pi_1, \pi_2)$ 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 two time series of lengths $n$ and $m$.

The global alignment kernel (GAK) is defined as:

$$k_{GA}(x, y) = \sum_{\pi \in A(n,m)} \prod_{i=1}^{|\pi|} k(x_{\pi_1(i)}, y_{\pi_2(i)}). \tag{3.25}$$

It can be shown [114] 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 [114]:

$$w(i,j) = \left(1 - \frac{|i-j|}{O}\right)_+, \tag{3.26}$$

where $O$ is the order of the kernel and $+$ refers to the fact that $w(i,j) = 0$ if $|i-j| \geq O$. 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 $O$, 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:

$$\delta(x_i, y_i) = \begin{cases} 0 & \text{if } \alpha(x_i) \neq \alpha(y_i) \\ 1 & \text{if } \alpha(x_i) = \alpha(y_i) \end{cases} \tag{3.27}$$

The Dirac Kernel is combined with the Toeplitz Kernel obtaining:

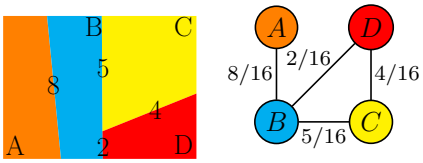
$$k_Z(x_i, y_j, i, j) = w(i,j) \cdot \delta(x_i, y_j). \tag{3.28}$$

**Weighted Dirac Kernel**: The main lack in using the traditional Dirac Kernel lies in the fact that the proximity of two zones is not considered. In order to overcome this limitation a weighted version of the dirac kernel which takes into account adjacency relationships between zones is also exploited:

$$k_{WZ}(x_i, y_j, i, j) = w(i,j) \cdot \delta_w(x_i, y_j). \tag{3.29}$$

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 3.27). 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:

$$\delta_w(x_i, y_i) = \begin{cases} 0 & \text{if } \alpha(x_i) \neq \alpha(y_i) \text{ and } e_{x_i,y_i} \notin E \\ w(e_{x_i,y_i}) & \text{if } e_{x_i,y_i} \in E \\ 1 & \text{if } \alpha(x_i) = \alpha(y_i) \end{cases} \tag{3.30}$$




(a) (b)

**Figure 3.27** A scene and the relative graph based representation. 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.

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

$$\phi_\sigma(\theta(x_i), \theta(y_i)) = \frac{1}{2\sigma^2}||\theta(x_i) - \theta(y_i)||^2 + \log\left(2 - e^{-\frac{||\theta(x_i)-\theta(y_i)||^2}{2\sigma^2}}\right). \tag{3.31}$$

This last kernel is used instead of the Gaussian one in order to guarantee that $k_{GA}$ is positive definite (p.d.) [114]. The combination of these two last kernels is defined as:

$$k_{(W)ZSS}(x_i, y_j, i, j) = k_{(W)Z}(x_i, y_j, i, j) \bullet k_{SS}(x_i, y_i). \tag{3.32}$$

Starting from Equation 3.25, 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 is performed in order to normalize kernel's values in the interval $[0, 1]$. Therefore, the final normalized kernel $k^N_{GA}$ is:

$$k^N_{GA}(x_i, y_j, i, j) = \frac{k_{GA}(x_i, y_j, i, j)}{\sqrt{k_{GA}(x_i, x_i, i, i) * k_{GA}(y_j, y_j, j, j)}}. \tag{3.33}$$

It is worth pointing out that the particular combination of kernels to be chosen depends on the general aim of the application we are interested in; suppose, for instance, that our objective is

to identify the vehicles crossing the highway in the wrong side: in this case, we may compare the trajectory to be tested with some prototypes encoding normal trajectories by only considering a similarity based on Dirac Kernel (or its weighted version); on the other hand, the Speed and Shape Kernel is sufficient alone if we are only interested in verifying if a vehicles is crossing a street with a very high speed, if compared with the usual speed of vehicles in that place.

### 3.2.6 Clustering algorithm

An essential step of the proposed approach is the understanding of typical patterns inside a scene. As a matter of fact, once extracted some *prototypes* of normal trajectories, abnormal trajectories can be easily found as those not enough similar to the extracted prototypes. The learning of normal prototypes is performed by using a novel kernelized clustering algorithm, based on the splitting methods presented in Subsection 3.2.3: 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 [70] are very expensive from a computational point of view, as we will show in Section 5.2.

Given the cluster $R$ containing all trajectories belonging to the training set (*i.e.* the root of the tree), let us consider an arbitrary order on $R = (T_1, ..., T_{|R|})$ and a generic cluster $C \subset R$. This cluster is encoded by the vector $1_C$ of $\mathbb{R}^{|R|}$, where $1_C(i)$ is equal to 1 if $i \in C$ and 0 otherwise. Finally, $K$ is the Gram Matrix of the training set, defined by $K_{ij} = (k^N_{GA}(s_i, s_j, i, j))$.

As mentioned in Section 3.2.3, our clustering algorithm builds a sequence of partitions $\rho_1, \ldots, \rho_N$ of $C$, with $\rho_1 = \{R\}$ and $\rho_N$ encoding a partition of $R$ into $N$ clusters. The following heuristics have been selected for each step; a simple example is shown in

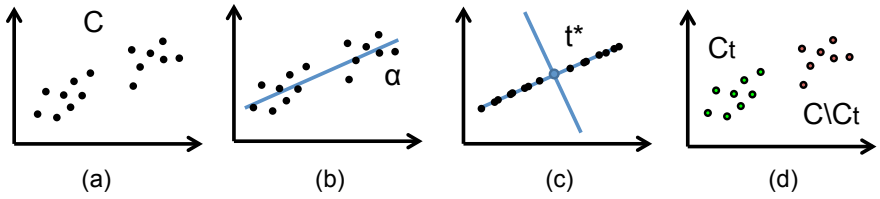



**Figure 3.28** Simple example of the proposed clustering algorithm: once selected the cluster $C$ (a), the cutting axis and the cutting position are computed ((b) and (c) respectively) and the new clusters $C_t$ and $(C \backslash C_t)$ are obtained.

Figure 3.28.

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

$$SE(C) = \sum_{s \in C} ||\psi_s - \mu||^2 = |C| - \frac{1}{|C|} 1_C^t K 1_C, \tag{3.34}$$

where $\psi_s$ is the projection of the string $s$ in the Hilbert space implicitly defined by $k_{GA}^N$. Equation 3.34 may be evaluated in $\mathcal{O}(|C|^2)$, with $|C|$ denoting the cardinality of $C$ (Appendix A).

**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 $(C \backslash C_t)$; the optimum cutting plane aims at minimizing $SE(C_t) + SE(C \backslash C_t)$.

A traditional heuristic to minimize $SE(C_t) + SE(C \backslash C_t)$ consists in splitting $C$ along its axis with the greatest variance, *i.e.* the major axis (Figure 3.28b). This last axis can be obtained by means of a Kernel PCA [115]. 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:

$$\tilde{K_C} = (K_C - \mathbf{\Gamma_C} \cdot K_C - K_C \cdot \mathbf{\Gamma_C} + \mathbf{\Gamma_C} \cdot K \cdot \mathbf{\Gamma_C}) \tag{3.35}$$

with $\mathbf{\Gamma_C}$ being a $|C|$ by $|C|$ matrix for which each element takes value $1/|C|$.

The eigenvector $\boldsymbol{\alpha}$ of $\tilde{K_C}$ associated to the largest eigenvalue $\lambda$ is then computed . Such a vector satisfies: $\lambda \boldsymbol{\alpha} = \tilde{K_C} \boldsymbol{\alpha}$ is then

computed. For each trajectory $s$, the projection of $\psi_s$ on the major axis $\nu$ of $C$ is obtained by:

$$< \nu, \psi_s >= \sum_{i=1}^{|C|} \alpha_i k(s_i, s). \tag{3.36}$$

**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 $T^\star$ 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 3.36 (Figure 3.28c).

If $\mu$ and $\mu_t$ denote respectively the means of $C$ and $C_t$, it can be shown (Appendix A) that $||\mu - \mu_t||^2$ can be computed as follows:

$$||\mu-\mu_t||^2 = \frac{1}{|C|^2}1_C^t K 1_C - \frac{2}{|C_t||C|}1_C^t K 1_{C_t} + \frac{1}{|C_t|^2}1_{C_t}^t K 1_{C_t}. \tag{3.37}$$

Note that the number of points to be processed, contained in the range $[m, M]$, is equal to $|C|$, which corresponds to the number of trajectories projected on the cutting axis. 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 (Appendix A) that $||\mu - \mu_{t+1}||^2$ can be efficiently updated from $||\mu - \mu_t||^2$. In particular, the first term $\frac{1}{|C|^2}1_C K 1_C$ is constant since it does not depend on the partition induced by $T$. Let be $1_{C_{t+1}} = 1_{C_t} + \delta_p$, being $\delta_p$ the vector of zeros containing a single 1 at position $p$. The second term $1_C K 1_{C_{t+1}}$ of Equation 3.37 may be defined iteratively as follows:

$$1_C K 1_{C_{t+1}} = 1_C K 1_{C_t} + \sum_{i \in C} k(i, p). \tag{3.38}$$

Note that the second term of Equation 3.38 may be precomputed. Equation 3.38 is thus evaluated in constant time. Finally, the last term $1_{C_{t+1}}^t K 1_{C_{t+1}}$ becomes:

$$1_{C_{t+1}}^t K 1_{C_{t+1}} = 1_{C_t}^t K 1_{C_t} + 2\sum_{i \in C_t} k(i, p) + k(p, p). \tag{3.39}$$

Using Equations 3.38 and 3.39, 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**: It should be clear that each obtained cluster encodes a prototype of normal trajectories: it implies that in real environments the number of different kinds of typical behaviors (and then the number of clusters) can not be fixed a priori, and then a more sophisticated strategy needs to be defined. In particular the following considerations have been considered: on the one hand, two clusters need to encode different behaviors (and then have to contain *different* trajectories). On the other hand, the risk to have clusters with a very low number of trajectories should be avoided. For these reasons, 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.

The output of the proposed clustering algorithm on the dataset shown in Figure 3.25a is depicted in Figure 3.29.

#### 3.2.6.1 Complexity analysis

In this section we summarize the complexity analysis of the proposed clustering algorithm. As for the *cluster selection* step, the algorithm needs 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 $(1^t_{C_t} K 1_{C_t})$, which is the most expensive one, is evaluated during the computation of the cutting position, as shown in Equation 3.37. On the other hand, the squared error of $(C \backslash C_t)$ can be simply computed as the difference between

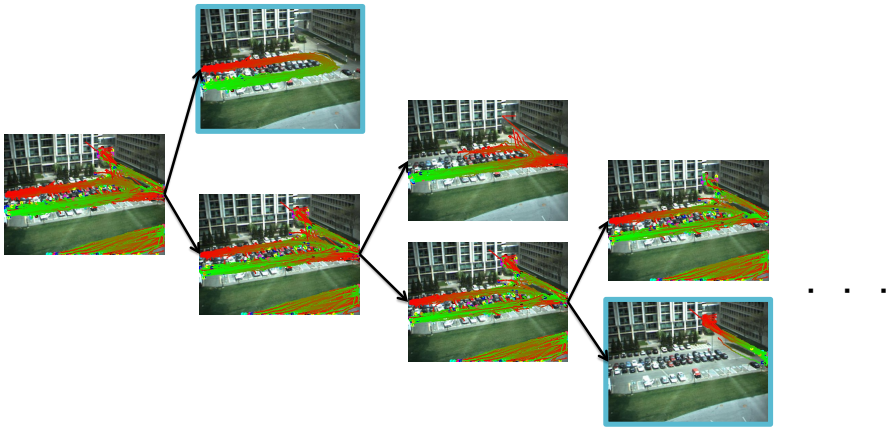


**Figure 3.29** A part of the tree obtained by applying the proposed clustering method over the MIT Trajectory Dataset. The color of each trajectory highlights its direction: it starts in green and progressively turns its color into red.

$SE(C)$ 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 [116]. 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 $C_t$ $\left(\sum_{i \in C_t} k(i,p)\right)$. 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 per-

formed in

$$O(N) + O(|C|^{2.376}) + O(|C| \log |C|) + O\left(\frac{|C| \cdot (|C| + 1))}{2}\right) = O(|C|^{2.376}). \quad (3.40)$$

### 3.2.7 Conclusion

In this section I analyzed the algorithm proposed in this thesis for learning typical trajectory patterns inside a scene by using a novel string-kernel based approach: each trajectory is represented as a string by combining different typologies of information concerning position, speed and shape; the similarity between trajectories is evaluated by defining a novel string kernel metric, which is used by the clustering algorithm proposed in this thesis. Each cluster encodes a kind of *prototype* of normal trajectories.

The tree obtained by the proposed clustering algorithm will be used for a double aim: on the one hand, it allows to recover abnormal trajectories if not enough similar to the clusters obtained during the learning phase (see Section 3.4.1). On the other hand, the tree is used for solving into a very efficient way the queries by sketch submitted by the human operator; in fact, the tree allows to find the most $k$ similar trajectories to the one hand drawn by the user without searching in the entire dataset (see Section 3.4.2).

A deep experimentation has been conducted over three standard datasets; as it will be shown in Section 5.2, the obtained results, compared with state of the art methods, confirm the robustness and the effectiveness of the proposed approach, both in terms of accuracy and computational cost.

## 3.3 Indexing and Storing Engine

Once extracted, the raw trajectories need to be properly indexed and stored in order to be efficiently retrieved.

The main contribution of the proposed approach is to index 3D trajectories by turning the 3D representations into a set of

2D schemes, so as to make it possible the use of well established and optimized available 2D indexing solutions. In particular, the proposed solution is optimized to solve Range Spatial Queries assumed to be Dynamic (*DRSQs*), i.e. are formulated in any their part at query time, so allowing the user the important potentiality of extracting from the database non only predefined information but the data he is interest in at any moment of system's use. The contribution of the method also includes the definition of different typologies of 3D queries, so general to hopefully cover most of the applicative needs and that can be formulated as a combination of two or more DRSQs. Finally, we describe how the retrieving efficiency can be significantly increased using a properly designed segmentation algorithm aimed at improving the selectivity of the proposed indexing strategy.

In this section I will focus on the storing engine: after a brief introduction over spatial databases in Subsection 3.3.1, some preliminary concepts will be provided in Subsection 3.3.2. The indexing engine and the segmentation strategy will be detailed respectively in Subsection 3.3.3 and 3.3.4. Finally, the physical representation scheme designed for this purpose will be discussed in Subsection 3.3.5.

More information on the retrieval engine devoted to the interactions with the user will be discussed in Section 3.4, and in particular in Subsection 3.4.3.

### 3.3.1 Spatial Databases

A spatial database defines special data types for geometric objects and allows to store geometric data in regular database tables. In particular, the following three basic geometries, shown in Figure 3.30, can be defined: points, linestrings and polygons.

The main advantage in the use of such databases for managing the above defined spatial entities lies in the fact that special functions and indexes for querying and manipulating that data using something like Structured Query Language (SQL) are provided [83]. Furthermore, such functions are optimized for answering spa-

tial queries: how far two points differ, whether points fall within a spatial are of interest and so on. A few examples are shown in Figure 3.31: for instance, function *Overlaps* verifies if two given polygons share space without being completely contained by each other, while function *Crosses* verifies if geometries (for instance a polygon and a linestring) have some, but not all, interior points in common.

On the other hand, the most efficient functions available in spatial databases are based on Minimum Bounding Rectangles (MBRs), which play a critical role when two geometries interact. As a matter of fact, it is possible to efficiently verify the interactions between geometries by checking the interactions between the corresponding bounding boxes. It is worth pointing out that this kind of comparison is made by taking advantage on the indices and then without the need of extracting geometries from the database, as in the cases shown in Figure 3.31.

On the other hand, it is clear that the geometries whose MBRs overlap do not necessarily overlap. In order to clarify this concept, an example is shown in Figure 3.32: on the left two geometries are shown; on the right the corresponding MBRs are depicted. We can note that, although the geometries do not intersect, their MBR do it. The reason why this concept is highlighted will be clarified in the following.

Any details about spatial datasets are out of the scope of this thesis. Our aim in this section is to highlight the big advantages in the use of such strategies, in order to better justify the proposed method. The reader can refer to the specialized literature for more details [83].

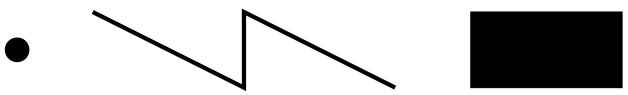


**Figure 3.30** Basic geometries of a spatial database: point, linestring and polygon.

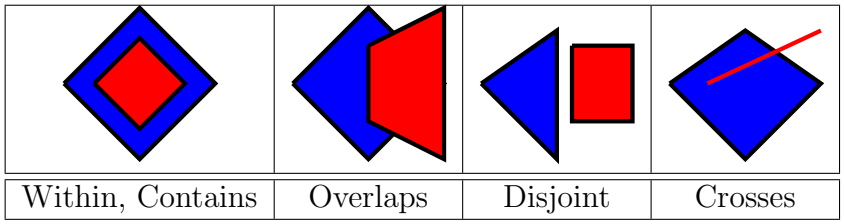



**Figure 3.31** Some of the basic spatial operations allowed by spatial databases.

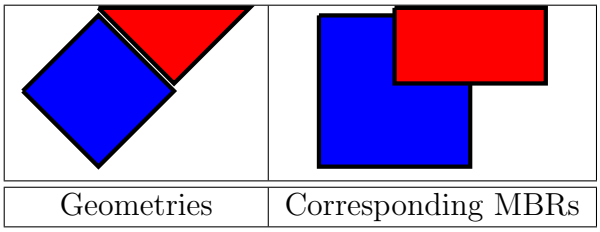



**Figure 3.32** The geometries do not intersect, while the corresponding MBRs do it.

### 3.3.2 Preliminaries

In Section 3.2.1 (Equation 3.17) I have shown that a trajectory can be seen as a sequence of three dimensional points. According to the line-segment model, we can further assume the trajectory as approximated by a polyline, each segment being the linear interpolant between two positions sampled at consecutive time instants.

Once defined the representation of trajectories, we need to exploit, from a geometric point of view, the proposed range spatial query (*DRSQ*). Its aim is to detect all those trajectories passing through a given spatial area $A$ in a given time interval $[t_s; t_e]$; the area $A$, assumed to be rectangular, can be fully identified by two points, respectively the *top-left* and the *bottom-right*, in the $xy$ plane: $P_m^{xy} = (x_{min}; y_{min})$ and $P_M^{xy} = (x_{max}; y_{max})$. Starting from this consideration, each DRSQ can be thus associated to a query box $Q$:

$$Q = \{(x_{min}; y_{min}; t_s); (x_{max}; y_{max}; t_e)\}. \tag{3.41}$$

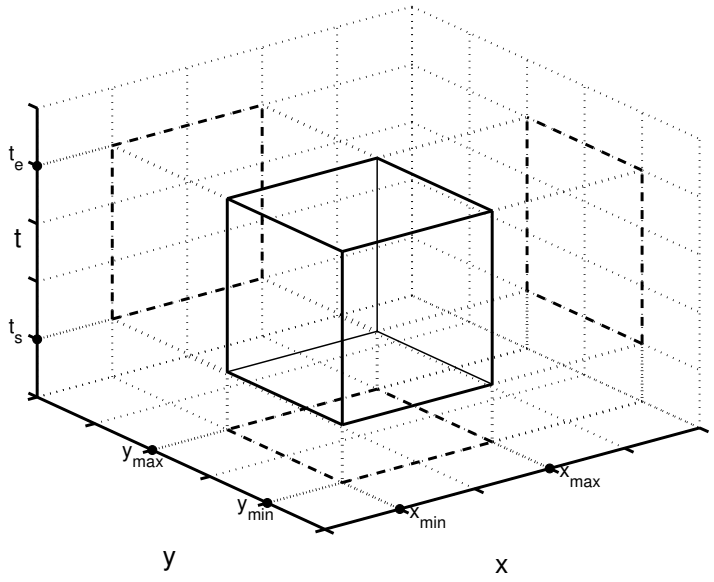



**Figure 3.33** Geometric interpretation of a *DRSQ*.

In other words, the temporal dimension extends the rectangular area $A$ in the 3D space (see Figure 3.33).

In order to solve a DRSQ, we have to select all the trajectories having at least one segment falling in the query box $Q$; for the generic trajectory $T^k$, it means to iteratively check this condition for each segment of $T^k$, being each segment identified by two consecutive points $\big((x_i^k, y_i^k, t_i^k), (x_{i+1}^k, y_{i+1}^k, t_{i+1}^k)\big)$, and to stop the process as soon as a segment intersecting the query box $Q$ is detected. The worst case arises when the trajectory does not intersect the query box: in this case, all the trajectory's segments have to be processed before asserting that the trajectory does not intersect the query box, resulting in a large amount of checks to do.

In turn, the problem of determining if a segment intersects $Q$ can be solved using the method proposed by Cohen and Sutherland for identifying that portion of a picture lying inside a given region, namely the clipping algorithms [117]; for the sake of completeness, let's briefly summarize their method into a 2D space.

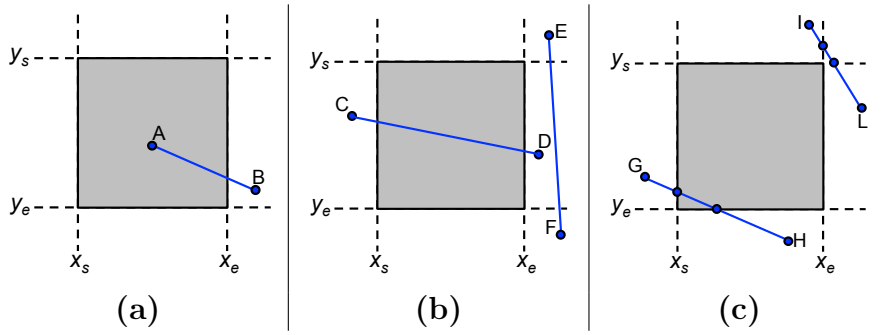



**Figure 3.34** The Cohen Sutherland clipping algorithm at work: in (a) and (b) the segment can be trivially analyzed, while in (c) a further processing is needed.

The geometric plane is subdivided into nine areas by extending the edges of the query box: if at least one of the segment endpoints lies inside the box, the intersection is trivially verified (see segment AB in Figure 3.34a); if, on the contrary, both the endpoints lie outside the query box, we check the position of the endpoints with respect to the query area: in some cases the intersection can be still trivially verified, as for CD and EF in Figure 3.34b, otherwise the segment is split at its intersection points and each obtained sub-segment is in turn inspected (as in the case of the segment GH and IL in Figure 3.34c). This clipping algorithm can be easily extended for dealing with 3D trajectories by considering 27 spatial regions, rather than 9. Although the aforementioned algorithm also provides the sub-segment lying inside the box, for simplicity in this discussion from now on we will indicate with the term *clipping* the boolean operation aimed at verifying if a segment intersects a box.

Note that the use of this kind of brute-force approach to solve a DRSQ is a very expensive operation, since it requires to preliminary extract each trajectory from the database and successively verify if at least one segment of each trajectory intersects $Q$ by applying the aforementioned strategies. In real applications the number of trajectories is too high to apply a similar approach, and more efficient approaches, coming from the spatial database field,

are thus mandatory: it implies that suitable indexing strategies are necessary to reduce the number of trajectories to be extracted and clipped. In the following sections the proposed solution will be detailed; it is mainly based on the assumption that the actual 3D problem can be solved in terms of three 2D problems, so that potentialities of well established bi-dimensional spatial indexes can be fully exploited.

Before leaving this section it is important to introduce the following symbols that we will use in the rest of this thesis. Given a query box $Q$, $Q|_{xy}$ ($Q|_{xt}$, $Q|_{yt}$) represents the projection of $Q$ on the $xy$ (respectively $xt$ and $yt$) coordinate plane. Furthermore, given a trajectory $T^k$ and its 3D MBR $B^k$, $B^k|_{xy}$ ($B^k|_{xt}$, $B^k|_{yt}$) is the projection of $B^k$ on the $xy$ (respectively $xt$ and $yt$) coordinate plane.

### 3.3.3 Indexing Engine

Before entering into details of the proposed approach, it is worth observing that if a trajectory $T^k$ intersects a query box $Q$, then $B^k$ will intersect $Q$ as well:

$$T^k \cap Q \neq \emptyset \Rightarrow B^k \cap Q \neq \emptyset \tag{3.42}$$

Equation 3.42 represents a necessary but not sufficient condition, as the opposite is clearly not true. It means that if the right part of the equation is satisfied, we can not assume that the trajectory intersects the query box. On the contrary, if the right part is not verified, we can confirm that the trajectory does not intersect the query box (see Figure 3.35(d)). This is a very important and not negligible consideration, since it allows to diminish the computational burden of the extraction of trajectories from the database, assuming the set of trajectories satisfying the right part of Equation 3.42 as the candidate set of trajectories to be extracted and clipped.

Another point to be considered is that the intersection between two boxes (the trajectory' MBR and the query box) can be easily verified using, when available, 3D spatial indexes, as briefly

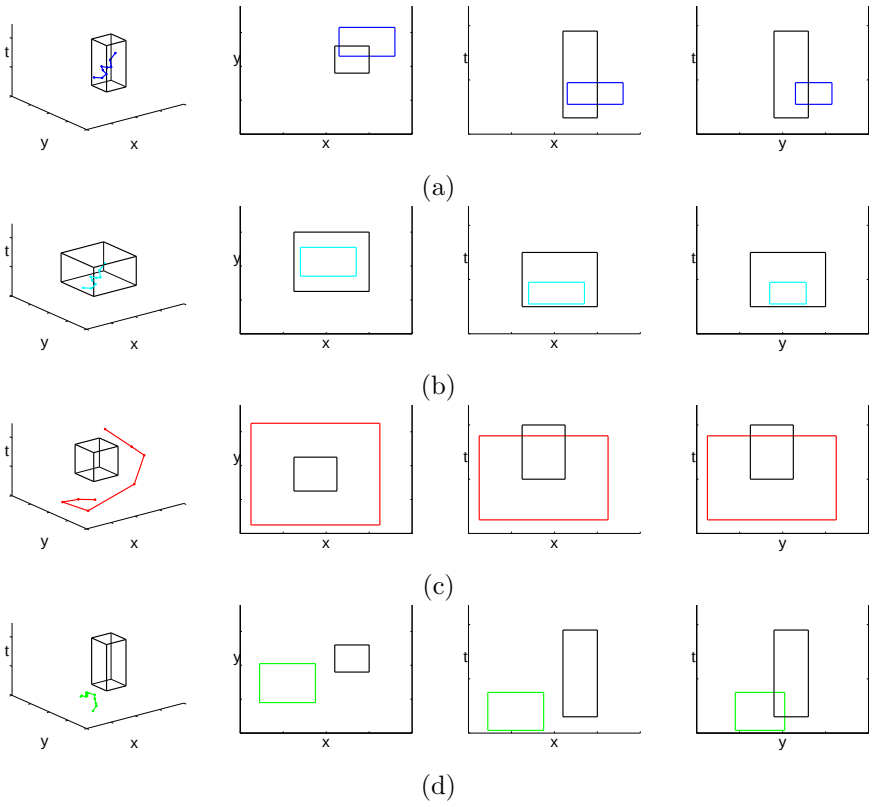



**Figure 3.35** Each row shows a trajectory and a query box, together with the corresponding projections on the 2D planes. In (a) the trajectory intersect the query box; in (b) it is completely contained in the query box; in (c) the trajectories does not intersect the query box although all the 2D projections do it; in (d) the trajectory does not intersect the query box.

explained in Section 3.3.1.

Furthermore, it is possible to observe that if and only if $B^k$ and $Q$ intersect (in the 3D space), all their respective projections on the coordinate planes intersect too (see Figure 3.35(a)):

$$B^k \cap Q \neq \emptyset \Leftrightarrow \left\{ \begin{array}{c} B^k|_{xy} \cap Q|_{xy} \neq \emptyset \\ B^k|_{xt} \cap Q|_{xt} \neq \emptyset \\ B^k|_{yt} \cap Q|_{yt} \neq \emptyset \end{array} \right\} \tag{3.43}$$

Unlike the previous condition, this is a necessary and sufficient

one. This is a very important feature, since Equation 3.43 allows to formalize the algorithm to solve a DRSQ using well established 2D indexes. In fact, assuming that for each trajectory $T^k$, we store and index $B^k|_{xy}$, $B^k|_{xt}$, and $B^k|_{yt}$, we can efficiently find, using one of the well established 2D indexes, three sets of trajectories $\Gamma_{xy}$, $\Gamma_{xt}$ and $\Gamma_{yt}$ defined as:

$$\Gamma_{xy} = \{T : B|_{xy} \cap Q|_{xy} \neq \emptyset\} \tag{3.44}$$

$$\Gamma_{xt} = \{T : B|_{xt} \cap Q|_{xt} \neq \emptyset\} \tag{3.45}$$

$$\Gamma_{yt} = \{T : B|_{yt} \cap Q|_{yt} \neq \emptyset\}, \tag{3.46}$$

Note that the generic set $\Gamma_{ab}$ contains all the trajectories whose projections on the plane $ab$ intersect the projection of the query box on the same plane. It is evident that the same trajectory can be contained in more than one set. Finally, the set $\Theta$ of the trajectories candidate to be clipped in the 3D space can be obtained by identifying all those trajectories contained in all the above defined sets. $\Theta$ is therefore defined as:

$$\Theta = \Gamma_{xy} \cap \Gamma_{xt} \cap \Gamma_{yt}. \tag{3.47}$$

In this way, we have turned a 3D problem (the one of establishing if $T^k$ intersects $Q$) into three 2D problems, namely to determine if the projection of $B^k$ intersects the projection of $Q$ on the same plane, for each of the three planes $xy$, $xt$ and $yt$.

It is important to underline that, although we are interesting in solving a 3D problem in this particular domain, Equations 3.42 and 3.43 can be easily generalized. This is a very interesting and useful result, since it makes the proposed method general and potentially able to deal with any kind of n-dimensional data.

The set of candidate trajectories to be clipped can be further reduced thanks to the following intuition: if the bounding box trajectory is completely contained in the query box, the condition expressed in Equation 3.42 becomes sufficient (see Figure 3.35(b)):

$$B^k \subset Q \Rightarrow T^k \cap Q \neq \emptyset. \tag{3.48}$$

Furthermore, as for the condition expressed in Equation 3.43, the problem can be easily re-conducted and solved into the bi-dimensional space: $B^k$ is completely contained in $Q$ in the 3D space if and only if all their respective projections on the coordinate planes are contained too:

$$B^k \subset Q \Leftrightarrow \left\{ \begin{array}{c} B^k|_{xy} \subset Q|_{xy} \\ B^k|_{xt} \subset Q|_{xt} \\ B^k|_{yt} \subset Q|_{yt} \end{array} \right\} \tag{3.49}$$

Consequently, we can state that:

$$\left\{ \begin{array}{c} B^k|_{xy} \subset Q|_{xy} \\ B^k|_{xt} \subset Q|_{xt} \\ B^k|_{yt} \subset Q|_{yt} \end{array} \right\} \Rightarrow T^k \cap Q \neq \emptyset. \tag{3.50}$$

The aforementioned consideration allows to select all those trajectories which surely satisfy the query, without the need to be clipped, and then without the need to be extracted from the database.

According to the considerations stated until this moment and independently on how the projection of trajectories are represented and indexed in the database, the selection of the trajectories intersecting a given query box can be carried out by three distinct successive steps, as shown in Figure 4.1:

- *Min-selection*: the determination of all the trajectories $T^i$ contained into the database $D$, whose bounding box $B^i$ is completely contained inside the query box $Q$:

$$PS = \left\{ T^i \in D | B^i \subset Q \right\}. \tag{3.51}$$

  It is worth noting that trajectories satisfying Equation 3.50 can be obtained by only working on their bounding box and checking that the latter is included in $Q$; the computational burden deriving from the need of extracting the trajectories from the database and successively checking that each its segment is included in $Q$ can be avoided, making this phase very effective from a computational point of view.

- *Max-selection*: the determination of all the trajectories $T^i$, not yet selected in the previous step, whose bounding box projections on the coordinate axes intersect the query box $Q$:

$$MS = \left\{ T^i \in D/PS | B^i \cap Q \neq \emptyset \right\}. \tag{3.52}$$

  This phase, as the previous one, is computationally cheap as it works by using the trajectories by means of their representation in terms of bounding box. Unfortunately, the max-selection, as clarified by Equation 3.42, selects not only trajectories which satisfy the query box, but also a set of other trajectories (from now on denoted as *false positive*) falling outside the query box but having the projections $B^k|_{xy}$, $B^k|_{xt}$ and $B^k|_{yt}$ overlapped with the corresponding query box projections $Q|_{xy}$, $Q|_{xt}$ and $Q|_{yt}$.

- *Clipping*: the selection among the trajectories selected in the previous step of all those trajectories really intersecting the query box (since the condition about the bounding boxes is only a necessary condition):

$$C = \left\{ T^i \in MS | T^i \cap Q \neq \emptyset \right\}. \tag{3.53}$$

  This is the most expensive step, since it necessarily requires the extraction of each trajectory $T^i$ from the dataset and the iteratively check of each segment.

The final set of trajectories $T$ satisfying our query will be composed both by the trajectories belonging to $PS$, selected during the *min-selection* step without the need to be extracted, and the trajectories $C$, selected by means of the *max selection* and the *clipping* steps:

$$T = PS \cup C. \tag{3.54}$$

It is important to underline that a 3D indexing has been recently implemented in some of the available spatial databases (like, for instance, Postgis), so allowing to verify the intersection between two boxes, namely the query box and the bounding box

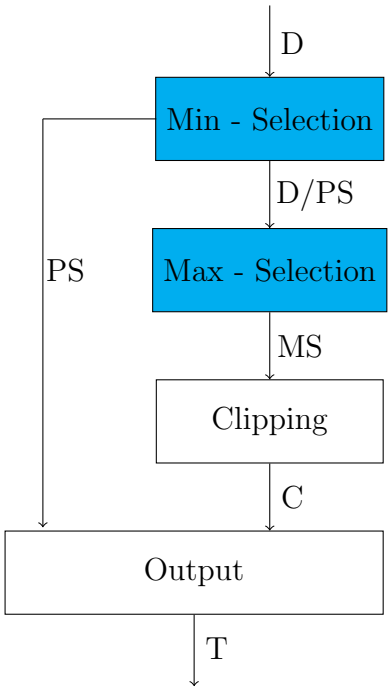



**Figure 3.36** An overview of the proposed method. Note that the Min and the Max Selection Steps work directly on the indexes.

trajectory, directly into the 3D space ($B^k \cap Q \neq \emptyset$), without the need to decompose the problem into three bi-dimensional spaces. However, on the other side, no solution has been proposed for verifying if a 3D box is completely within another one ($B^k \subset Q$). It means that an efficient solution can be reached only by decomposing the problem into simpler problems and by using off-the-shelf 2D operators.

### 3.3.4 Trajectory Representation Scheme

It is clear at this point that the retrieval performance of the proposed approach strongly depends on the capability of the system to reduce the number of trajectories to be clipped, and then to increase the number of trajectories satisfying Equations 3.43 and

3.51, namely the selectivity of the indexes. Furthermore, it is simple to imagine that the longer are the trajectories in the database the higher will be the size of the corresponding bounding boxes. The rationale of the proposed approach is to preliminarly segment each trajectory into smaller pieces, so as that, in the whole, the min-selection, max-selection and clipping steps are computationally more convenient.

The segmentation algorithm recursively segments a trajectory into smaller units (called trajectory units); the generic trajectory unit of the trajectory $T^k$ is identified by the ordinal number of its starting point and end point respectively, say $r$ and $s$, and denoted with $T^k_{r,s}$.

Let us now analyze how the algorithms proceeds: it starts from a trajectory unit (note that at the beginning an input trajectory $T^k$ can be seen as a trajectory unit and denoted with $T^k_{0,n}$). Each trajectory unit is then recursively split until any trajectory units has a length less than a given fixed threshold. Consideration about the determination of the optimal value of the threshold will be presented in the following.

A greedy criterion, aiming to maximize the selectivity of the indices, has been considered to split a generic trajectory unit; the selectivity can be informally defined as the suitability of the indexing method to select only those trajectories really satisfying the query, and then to reduce the number of false positive.

The selectivity of the indexing method can be achieved by splitting the input trajectory unit $T^k_{r,s}$ on the axis having the highest relative projection, as shown in Figure 3.37; to this aim we preliminarily calculate its bounding box projections $B^k_{r,s}|_x$, $B^k_{r,s}|_y$ and $B^k_{r,s}|_t$ on the three axes normalized by the maximum relative size. To this concern, we point out that each axis is limited to a range of values: $||x||$ and $||y||$ are the physical coordinates of the images while $||t||$ refers to the width of the time period.

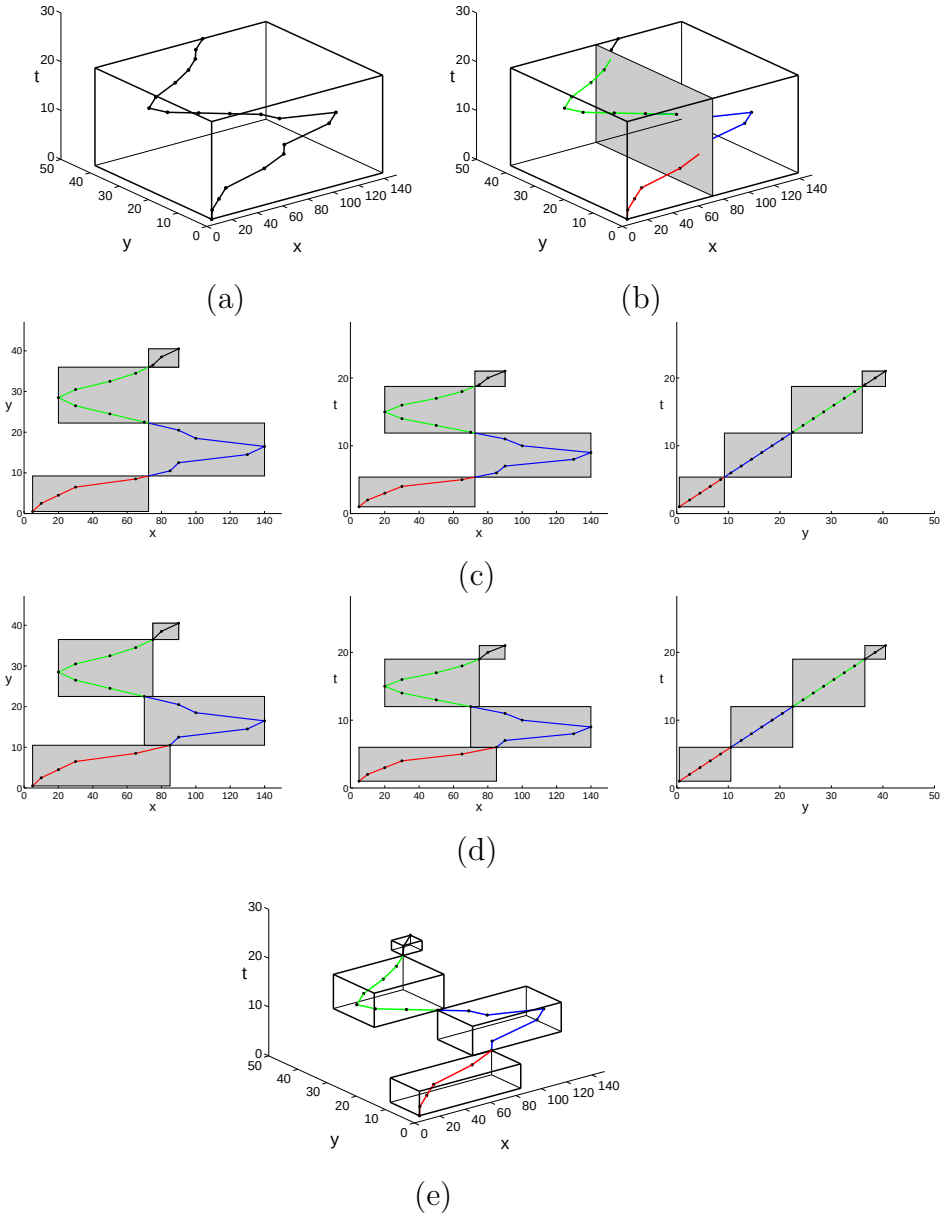



**Figure 3.37** An overview of the segmentation algorithm.

Let be:

$$O_x = \frac{||B^k_{r,s}|_x||}{||x||} \tag{3.55}$$

$$O_y = \frac{||B^k_{r,s}|_y||}{||y||} \tag{3.56}$$

$$O_t = \frac{||B^k_{r,s}|_t||}{||t||}. \tag{3.57}$$

Our algorithm selects the split axis $\xi$ as the one maximizing the corresponding $O_k$, with $k = \{x, y, t\}$:

$$\xi = \arg\max_k(O_k) \tag{3.58}$$

Once determined the split axis, the split value is selected as the average value of the projection of the trajectory unit on $\xi$;

$$\xi^* = \frac{max_\xi(T^k_{r,s}|_\xi) - min_\xi(T^k_{r,s}|_\xi)}{2} \tag{3.59}$$

The obtained value $\xi$ in a 3D plane allows to obtain the plane $\xi = \xi^*$ (from now on called segmenting plane) that intersects the considered input trajectory unit into at least one point; generally the number of intersections may be higher, so making it possible its split into several pieces, as highlighted in Figure 3.37, each one representing a smaller split unit.

The segmenting plane divides the 3D space into two half-3D planes: the trajectory is segmented by this plane and some of its pieces will fall into one of these two half-plane and other pieces on the other. Each piece now becomes a different trajectory unit, ready to be split again if the stop criterion of the Algorithm 3.38 is not satisfied.

It is worth pointing out that an intersection with the trajectory unit may happen not precisely at the junction of two successive segments. As it is important to remain still unchanged the original segments composing a trajectory, it is necessary in these cases to choose the junction closest to the intersection with the the plane

```
procedure SplitTrajectory(Trajectory T)
 for each Unit U in Trajectory T
   if(StopConditionsAreNotReached(U))
        ξ <- MaximumCoordinateAxis(O_x, O_y, O_t)
        ξ* <- ComputeSplitValue(ξ, U)
        P <- IntersectionPoints(U,ξ,ξ*)
        List<U> <- Split(U, P )
        List<U> <- UpdateSegmentedTrajectory(T, List<U>)
   end
  end
end procedure
```

**Figure 3.38** The structure of the segmentation algorithm.

$\xi = \xi^*$: so, instead to introduce a new point (the one relative to the intersection), the segmentation algorithm, by means of the procedure `Intersection points`, chooses for each segment intersecting the segmenting plane the closest point. Figure 3.37 illustrates this step: in c) are reported the points of the intersections and in d) the chosen ones.

The algorithm will end as soon as a stop condition, evaluated by the procedure `Stop Conditions Are Not Reached`, is verified. In particular, for each segment the areas occupied by the bounding box projection over all the planes are computed (respectively $A(B^k|_{xy})$, $A(B^k|_{xt})$ and $A(B^k|_{yt})$) and the obtained values are normalized over the corresponding projections of the entire volume ($A(V|_{xy})$, $A(V|_{xt})$ and $A(V|_{yt})$). Note that the volume is computed by considering the maximum extent of the trajectories contained in the dataset. If at least one of the computed ratios is lower than a given threshold ($AreaMin$), then the procedure is stopped.

### 3.3.5 Physical Representation Scheme

In the following, a detailed description of the physical schema of the database will be provided. For each trajectory, an instance in the table *Object*, depicted in Figure 5.2, is associated. This table contains all the additional information associated to the appearance of an object: its class (person, animal, vehicle and so on) and its dominant color. Both these information are extracted during

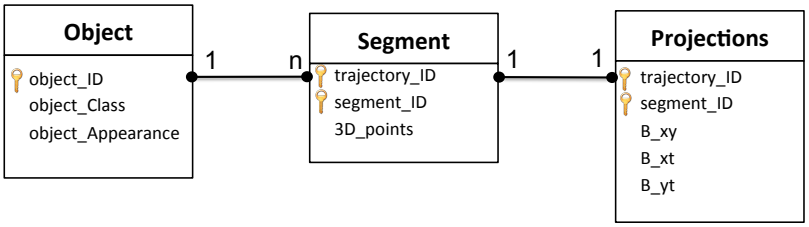



**Figure 3.39** The physical schema of the database.

the tracking phase. It is worth pointing out that any other useful and available information extracted by the tracking algorithm can be very easily stored in the table *Object*: for instance, the plate of the vehicle or its brand if available and so on.

According to the indexing method that we proposed, we need to store, for each acquired trajectory $T^k$, all its component units: a generic unit $T^k_{r,s}$ is represented by means of two typologies of information: the sequence of consecutive points and the bounding boxes on the coordinate planes $B^k_{r,s}|_{xy}$, $B^k_{r,s}|_{xt}$ and $B^k_{r,s}|_{yt}$.

These spatial objects are represented by exploiting the geometrical entities, efficiently stored in PostGIS through a Well-known binary (WKB) representation [118]; it is a markup language standardized from the Open Geospatial Consortium (OGC) in the ISO/IEC 13249-3:2011 standard (Information technology – Database languages – SQL multimedia and application packages – Part 3: Spatial). WKB, based on a binary format, permits geography data to be exchanged between a client application and an SQL database in a very efficient way. In particular, for our purposes we consider two kind of geometries: the *ST_LineString* is a path between locations which takes the form of an ordered sequence of two or more points; on the contrary, the *ST_Polygon* is a representation of an area. The outer boundary of the polygon is represented by a ring, which is a *ST_LineString* closed and simple. Since our polygon is a box, it can be univocally identified through its bottom-right and top-left points.

These information are stored in the tables *Segment* and *Projections*: in particular, the former contains the sequence of points *3d-points*, stored as a three-dimensional *ST_LineString*. The lat-

ter contains the three projections $B^k_{r,s}|_{xy}$, $B^k_{r,s}|_{xt}$ and $B^k_{r,s}|_{yt}$ stored as a PostGis *ST_Polygon*.

During the operating phase, the human operator defines a new query box $Q$: three new *ST_Polygon*s ($Q\_xy$, $Q\_xt$ and $Q\_yt$) are initialized, one for each coordinate plane.

The preselection step uses of the PostGis operator '@', which takes advantage on the spatial indices to verify if a box is completely within another one. A simplified example of the considered query is shown in the following:

```
SELECT Projections.trajectory_ID,
       Projections.segment_ID
FROM Projections p
WHERE p.B_{xy} @ Q_{xy} and
       p.B_{xt} @ Q_{xt} and
       p.B_{yt} @ Q_{yt};
```

During the max selection step, the PostGis '&&' operator is used in order to efficiently verify if a box intersects another one.

A very simple example for the max-selection step, without considering any kind of information about the vehicle appearance, is shown in the following:

```
SELECT Projections.trajectory_ID,
       Projections.segment_ID
FROM Projections p
WHERE
       Q_{xy} && p.B_{xy} and
       Q_{xt} && p.B_{xt} and
       Q_{yt} && p.B_{yt};
```

### 3.3.6 Conclusion

In this section the approach proposed in this thesis for storing and retrieving moving objects trajectories has been detailed. Starting from geometrical intuitions, it has been shown how to decompose the n-dimensional problem into a set of (n-1)-dimensional ones, and in particular how to decompose the three-dimensional problem into a set of bi-dimensional ones for managing three-dimensional trajectories data. This consideration has allowed to profitably use existing spatial databases, whit their well established bi-dimensional indexes, in order to significantly improve the

retrieving performance. As I will show in Section 5.3, the proposed method has been tested both over real and synthetic datasets and its performance, compared with a state-of-the art approach, confirm its effectiveness.

## 3.4 Interactions with the user

This section details the different typologies of interactions allowed to the user:

- **Anomaly detection**: as soon as an abnormal behavior occurs in the scene, the human operator is alerted. This interaction is made possible thanks to a previous learning phase, introduced in Section 3.2, which allows to learn some models corresponding to normal behaviors. More details will be provided in Subsection 3.4.1.
- **Query by sketch**: the human operator is allowed to draw a trajectory and to extract from a given set of trajectories the $k$ most similar ones. A detailed description of the algorithm defined for this purpose in detailed in Subsection 3.4.2.
- **Spatio Temporal Queries**: as in the previous case, the human operator can select at query time the queries parameter; in particular, spatial and temporal information can be defined by the operator. In Subsection 3.4.3 this kind of interaction will be described.

### 3.4.1 Anomaly Detection

Once extracted the prototypes of *normal* trajectories according to the algorithm defined in Section 3.2, the control system can start the operating phase, devoted to identify abnormal behaviors. In particular the operating phase is composed by the following steps:

- **Trajectory preprocessing**: the extracted trajectory is represented as a sequence of symbols according to the representation strategy detailed in Subsection 3.2.4.

- **Classification**: the distance between a trajectory $t_s$ and all the cluster's centroids $C_1, ...C_N$ obtained during the learning phase is evaluated. The cluster with the closest mean from $t_s$ is selected as the potential typical trajectory followed by $t_s$.

- **Decision**: An additional test is 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 (✗).

It is worth pointing out that 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.

Let's enter into details of the different steps required by the proposed method.

**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 [119], 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 previous regularization of the covariance matrix, which prevents it to be singular. Let be $\tilde{K}_{reg}$ the regularized covariance matrix, obtained as:

$$\tilde{K}_{reg} = \tilde{K} + \alpha \cdot I_{|C|}, \quad \alpha = |C| \cdot \sigma^2 \tag{3.60}$$

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 [119]). $I_{|C|}$ is the $|C|$ by $|C|$ identity matrix.

Let be $\mathbf{k}_s = [k(s, s_1), ..., k(s, s_{|C|})]^t$ and $\tilde{\mathbf{k}}_s = [\tilde{k}(s, s_1), ..., \tilde{k}(s, s_{|C|})]^t$ respectively the vector of kernel values of $s$ on the training data and its centered version; according to Equation 3.35, the generic $\tilde{k}(s, s_i)$ can be computed as follows:

$$\tilde{k}(s, s_i) = k(s, s_i) - \frac{1}{|C|}\left(\mathbf{1}_C^t \cdot \mathbf{k}_s + K \cdot \mathbf{1}_C(i)\right) + \frac{1}{|C|^2} \cdot \mathbf{1}_C^t \cdot K \cdot \mathbf{1}_C. \tag{3.61}$$

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

$$d_{RC}^2(\psi_s, C) = \frac{1}{\sigma^2}\left(\tilde{k}(s, s) - \tilde{k}_s^t \cdot \tilde{K}_{reg}^{-1} \cdot \tilde{k}_s\right). \tag{3.62}$$

**Decision:** Let $C^*$ denote the cluster with the lowest distance $d_{RC}(\psi_s, C^*)$ determined according to Equation 3.62. A threshold on the probability that the string $t_s$ belongs to $C^*$ is obtained by comparing the square distance $d_{RC}^2(\psi_s, C^*)$ with a fixed threshold $\alpha$:

$$d_{RC}^2(\psi_s, C^*) \leq \alpha. \tag{3.63}$$

Conversely to the parameter $\nu$ of one class SVM, a high value of $\alpha$ provides a better generalization but may increase the number of false positive in the test determining the classification to $C^*$.

### 3.4.2 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 [120], 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 subsets induced by the partitioning. The terminal nodes (the leaves of the tree) represent mutually exclusive small subsets of data

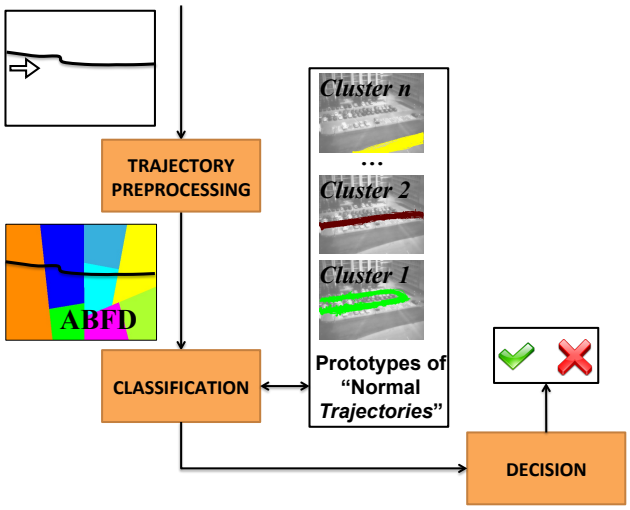



**Figure 3.40** Overview of the algorithm defined for anomaly behavior recognition.

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 logarithmic time.

The main idea exploited in this thesis 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, as shown in Figure 3.41.

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 Subection 3.2.6: 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

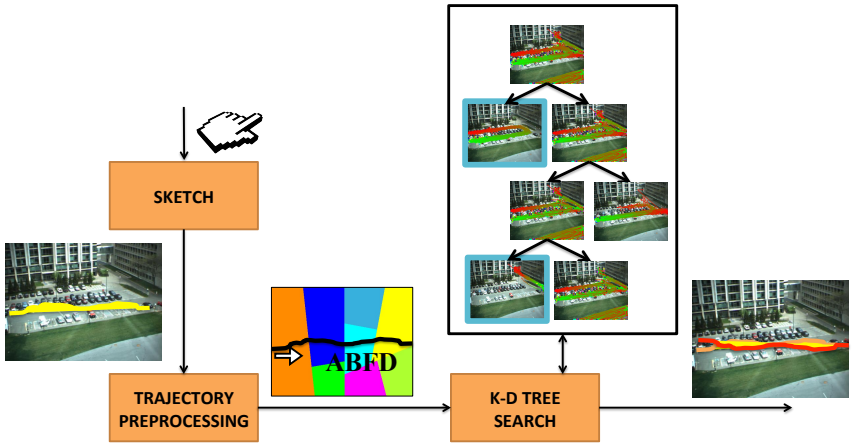



**Figure 3.41** Overview of the algorithm defined for answering queries by sketch.

cluster $C$, the projection on an input data (a query) on the major axis 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:

$$var(s) = \frac{1}{|C|}\sum_{j=1}^{|C|}(< \psi_j, \psi_s > - < \mu, \psi_s >)^2 \tag{3.64}$$
$$= \frac{1}{|C|}\sum_{j=1}^{|C|}k^2(s, s_j) - \left(\frac{1}{|C|}\sum_{j=1}^{|C|}k(s, s_j)\right)^2 .$$

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 5.2.4, 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

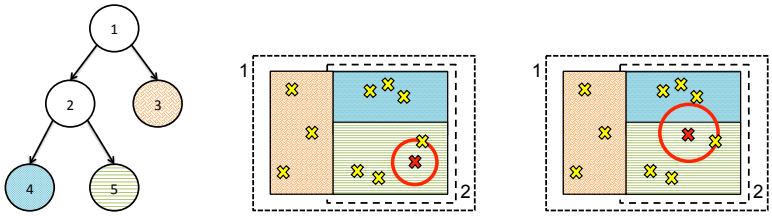
```
1  bool KNN( /*Input*/ Cluster c, Trajectory x, int k,
2            /*Input - Output*/ double &radius,
3            double &minDist, PriorityQueue &pq){
4   if (c.isLeaf == true){
5    radius <- updateKNN(c, x, pq);
6    return (radius<=minDist);
7   }
8
9   t <- ProjectOnTheCuttingAxis(c, x);
10  minDist <- min( minDist, |c.topt - t| );
11
12  if (t < c.topt){
13   bestSon <- c.leftSon;
14   otherSon <- c.rigthSon;
15  }else{
16   bestSon <- c.rigthtSon;
17   otherSon <- c.leftSon;
18  }
19
20  bob <- KNN(bestSon,x,k,radius,minDist,pq);
21  if (bob == true)
22   return true;
23
24  if (|c.topt - t| <= radius ){
25   bob <- KNN(otherSon,x,k,radius,minDist,pq);
26   return bob;
27  }
28
29  return false
30 }
```

**Figure 3.42** The structure of the algorithm for K- Nearest Neighbor Search.

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 3.42. 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* (Figure 3.42, lines 4-6).

If the node under investigation is not a leaf, the son which contains the query *bestSon* is found (lines 12-17) and the procedure is recursively called over it (lines 20-22). Furthermore, the algo-




(a) (b) (c)

**Figure 3.43** A tree (a) represented by its geometric boundaries (b)(c). In (b) the *bob* test has success, while in (c) it fails.

rithm has to verify if it is necessary to also invoke the procedure over the other child *otherSon* (lines 24-26). This test, also known as *bound overlap ball* test (hereinafter *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 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 3.43. In particular, Figure 3.43a shows a tree obtained by the proposed clustering algorithm with its representation on the plane (Figures 3.43b, 3.43c). 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 3.43b the boundary delimiting the two children clusters (identified respectively by the number 4 and 5 in Figure 3.43a) are not intersected by the ball, while it clearly happens in Figure 3.43c. It means that only in this last case the algorithm needs to also exploit the trajectories contained in the *otherSon*.

### 3.4.3 Spatio Temporal Queries

In Section 3.3 we have deeply analyzed the method proposed for indexing and storing trajectories, optimized for solving a *DRSQ* query. It should be clear that the aim of a *DRSQ* is to an-

swer queries involving spatio-temporal information: *find the objects crossing a given spatial area in a given time interval.*

From a more formal point of view, a *DRSQ* can be defined as a function:

$$DRSQ : (Q, D) \rightarrow (T \subseteq D) \tag{3.65}$$

in which $Q$ is the query box, $D$ is the entire set of trajectories stored into the database and $T$ is the resulting subset of trajectories satisfying the *DRSQ* query.

In order to better clarify the importance of the proposed *DRSQ* in real applications, a few examples are needed; for instance, in the context of traffic management systems, this kind of queries can be used for different retrieval purposes. Typical interests of a human operator are:

- *find the number of vehicles passing by a given toll-house each six hours during the last two days.*
- *find all the vehicles passing by highway I-55 and then by Interstate 60 from 2 to 4 pm today.*
- *find all the vehicles passing by an area (defined at query time) inside San Peter's Square on 14th March 2011 in the morning.*

More specific information can be obtained by augmenting this kind of queries with additional information about objects' appearance:

- *find the number of red trucks leaving highway I59 from exit 14 running overlapped with the middle lane (this area being defined at query time) between 9-12 am yesterday*;
- *verify whether a given motorcycle was walking through a level crossing between 6-8 pm yesterday.*

All these above mentioned queries can be easily and efficiently derived from the DRSQ. As a matter of fact, the *DRSQ* query can also be used to:

- count the objects satisfying the query; in this case the output will consist in a count value $c$:

$$DRSQ_{COUNT} : (Q, D) \rightarrow (c) \tag{3.66}$$

- verify whether a given object has passed through a given area at a given time interval; therefore the object's ID must also be given in input and the result has a boolean form:

$$DRSQ_{ID} : (Q, ID, D) \rightarrow \{true, false\} \tag{3.67}$$

- search or count the objects having specific appearance characteristics, in which case an appearance vector $A$ must also be provided and the output can consist of a count value $c$ or the subset of trajectories $T$ satisfying the query:

$$DRSQ_{A} : (Q, A, D) \rightarrow c \vee (T \subseteq D) \tag{3.68}$$

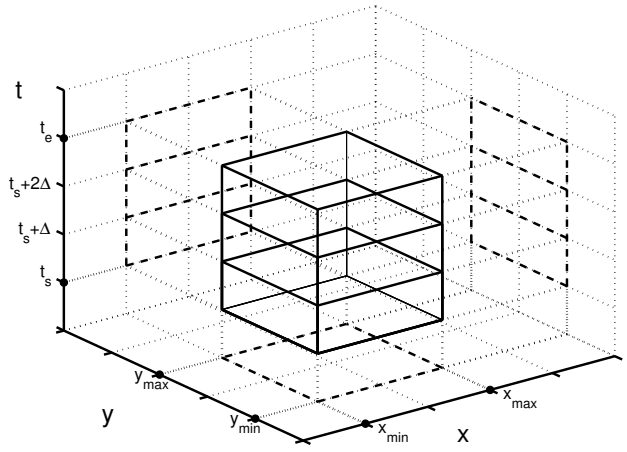
The *DRSQ* query can be also extended in order to analyze the traffic flow, or in general to provide statistic in the observed scene; from a geometrical point of view this new kind of query, hereinafter named as Flow-DRSQ *(F-DRSQ)*, can be introduced for analyzing the traffic flow in the observed scene; it allows to retrieve information of the type:

- *find the number of vehicles passing in a given area (dynamically defined at query time) on highway S14 each hour from 8 am to 6 pm yesterday*;
- *find the number of vehicles passing by a given toll-house each six hours during the last two days.*

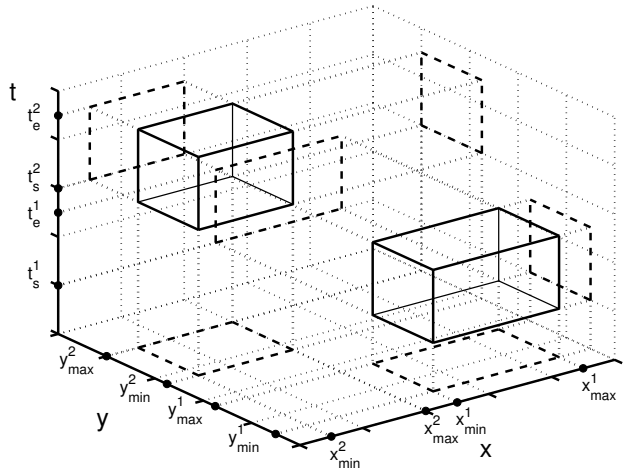
From a more formal point of view, it can be seen as the application of various *DRSQ* queries so as to obtain results at fixed time intervals, as shown in Figure 3.44a.

A *F-DRSQ* query can be defined as a function:

$$F - DST : (Q, n, D, A) \rightarrow (c_1, c_2, ..., c_n) \tag{3.69}$$




(a)




(b)

**Figure 3.44** Geometric interpretation of a *Flow-DRSQ* (a), and of a *2-DRSQ* (b).

in which $Q$ is the 3D query box, $n$ is the number of fixed intervals, $D$ is the entire set of trajectories and $A$ is an optional parameter identifying the appearance; $c_1, c_2, ..., c_n$ are the $n$ resulting count values.

Finally, a further type of query, called from now on Multi-

DRSQ *(M-DRSQ)*, can be defined to retrieve more complex and structured information; examples of this query type are:

- *find the number of vehicles that have passed by a given intersection and then have exited by a given offramp two days ago*;
- *find all the yellow vehicles passing by highway I-55 and then by Interstate 60 from 2 to 4 pm today.*

As Figure 3.44b shows, this query typology can be seen as an application of two (or more) DRSQ queries, typically having temporally successive query boxes.

$$M - DRSQ : (Q_1, ..., Q_n, D, A) \rightarrow (T \subseteq D) \tag{3.70}$$

in which $Q_1, ..., Q_n$ are the $n$ 3D query boxes, $A$ is the optional appearance information, $D$ is the entire set of trajectories and $T$ is the subset containing the trajectories intersecting all the $n$ query boxes and, thus, satisfying the *M-DRSQ* query.

### 3.4.4 Conclusion

In this section we analyzed the different typologies of interactions allowed to the human operator by the proposed system. On the one hand, an alert is sent as soon as an abnormal behavior occurs (Section 3.4.1); on the other hand, the human operator can dynamically interact with the system by submitting different typologies of queries, whose parameters are specified only at query time, which involve both semantical (Section 3.4.2) and spatio-temporal information (Section 3.4.3). A deeper experimentation has been conducted in Chapter 5 in order to confirm the effectiveness of the introduced interactions: in particular, in Section 5.2.5 the results concerning the anomaly detection algorithm are presented, while Sections 5.2.6 and 5.3.3 analyze the different typologies of queries, respectively concerning semantical and spatio-temporal information.

# Chapter 4

# From Audio Signals to Events of Interest

In this chapter we will focus on the module devoted to audio event detection; given $M$ classes of sounds of interest $C_1, \ldots, C_M$ (such as gunshot, scream, glass breaking, etc.), each represented by a finite set of examples, and an audio stream, the goal of the system is to find if (and where) there are occurrences of the sounds of interest within the stream. The audio stream usually contains other sounds not belonging to the classes of interest, that are considered as *background* sounds; we will indicate as $C_0$ the class containing all the background sounds.

The input audio signal is first divided into small frames and for each frame a feature vector is calculated. Then the feature vectors extracted from the audio clips of the training set are quantized by using a vector quantization technique, namely the K-means clustering algorithm, in order to build an artificial dictionary of *aural words*. *Aural words* are considered as small perceptual units of hearing, whose distribution over a time interval allows to characterize the type of sound. The detection of abnormal audio events is performed on a time window of $m$ seconds that is forward shifted on the audio stream by one third of its length. For each interval, a histogram of the occurrences of the aural words is constructed and is used as a descriptor of the sound to be classified. It is worth

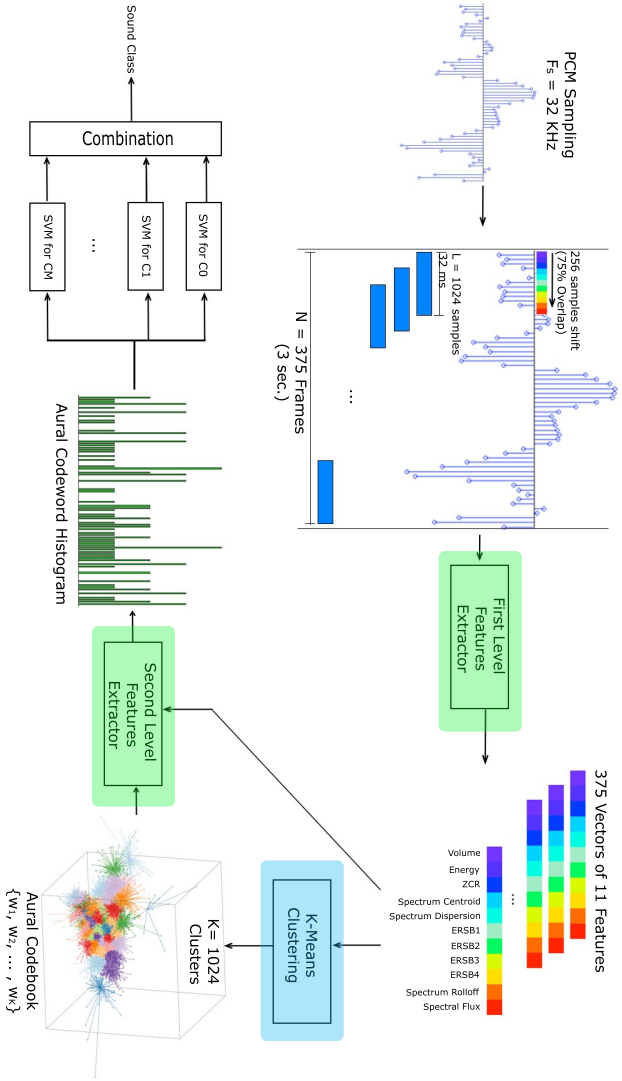



**Figure 4.1** System architecture of the proposed method. The modules used in both the training and the operative phases are shown in green, while the blue module is used only during the training phase. The values of the parameters $F_s$, $L$, $N$ and $K$ used for the experimental validation are also reported.

noting that the order of the aural words and their position within an interval is not important and it does not affect the detection results.

The architecture of the proposed system is shown in Fig. 4.1. The first-level and second-level feature extractor modules are used both in the training phase and in the testing phase. The K-means clustering module, instead, is involved only in the training phase in order to define a reference codebook for the construction of the second-level feature vectors. Finally, a multiclass SVM classifier is adopted to learn *bag of aural words* representations of the target events.

## 4.1 First-level features

In contrast to video signals, in which a scene can persist even for several seconds, an audio signal might show huge variations within a few milliseconds. Thus, the input audio stream, originally sampled at a rate $F_s = 32KHz$, is segmented into groups of $N$ partially overlapping frames of duration $T_F$. The choice of $T_F$ is influenced by two contrasting effects: if the value is too short, the frame will be unable to accurately represent low-frequency components of the sounds. Conversely, if it is too long the frame will not represent adequately shot-time changes in the audio signal. We have experimented with several values of $T_F$, and we have found that a reasonable compromise is achieved with $T_F = 32msec$. Every frame is built by forward shifting the frame window by $T_F/4$ and contains $L = 1024$ PCM samples.

For each frame, a first-level feature vector is computed. In particular, we have considered a set of 11 features from the literature on audio event detection belonging to the category of *spectral features*, namely spectral centroid, spectral spread, spectral rolloff, spectral flux [121, 122], of *energy features*, namely total energy, sub-band energy ratios (for 4 sub-bands), volume [121, 123] and of *instantaneous temporal feature*, namely the zero-crossing rate [121, 122]. In the following, we provide a brief description of the

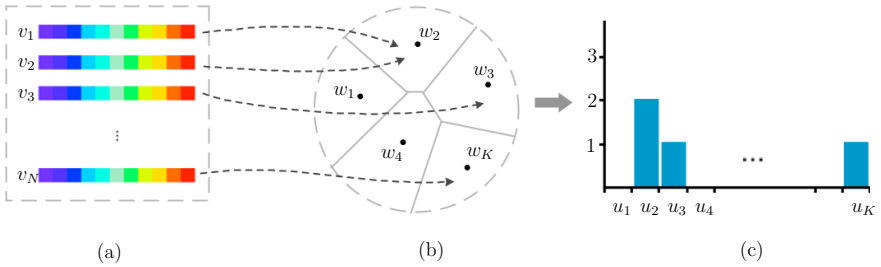



**Figure 4.2** Second-level feature vectors construction. The interval to be classified is segmented into $N$ frames of 32 milliseconds that are represented as first-level feature vectors (a). For each vector $v_i$ the codebook is searched for the nearest aural word $w_j$ (b). Then a histogram of the occurrences of the aural words is computed (c). In the example, the vectors $v_1$ and $v_2$ have $w_2$ as the nearest aural word in the codebook. Thus, the second bin of the histogram has a value equal to 2. In the same way, aural word $w_3$ has only one close vector, resulting in a value equal to 1 for the third bin of the histogram. Aural words $w_1$ and $w_4$, instead, have no occurrences.

first-level features.

**Spectral centroid and spectral spread**: In digital signal processing, the spectral centroid (SC) and the spectral spread (SS) are measures for characterizing the distribution of the frequency components of a signal. The spectral centroid is defined as the *center of mass* of the spectrum and is computed as follows:

$$SC = \frac{\sum_{k=1}^{L_F} k \frac{F_s}{L_F} |X(k)|}{\sum_{k=1}^{L_F} |X(k)|}, \tag{4.1}$$

while the spectral spread is computed as the dispersion of the frequency components of the signal around the centroid:

$$SS = \sqrt{\frac{\sum_{k=1}^{L_F} \left[k \frac{F_s}{L_F} - SC\right]^2 |X(k)|}{\sum_{k=1}^{L_F} |X(k)|}}, \tag{4.2}$$

where $L_F = 2048$ and $|X(k)|$ are the length and the module of the $FFT$ of the input signal $x(n)$, respectively.

**Spectral rolloff**: The spectral rolloff is a measure of the skewness of the spectrum and is defined as the frequency $f_{ro}$ at which

the percentage $P$ of the spectral components of the signal is at lower frequency. In our case, we consider $P = 90$ and determine the value $f_{ro}$ from the following relation:

$$\sum_{k=1}^{f_{ro}} |X(k)| = \frac{P}{100} \sum_{k=1}^{L_F} |X(k)| \,. \tag{4.3}$$

**Spectral flux**: The spectral flux (SF) indicates how quickly the spectral information of a signal is changing in time and it is computed as the squared-difference between the spectra of two consecutive audio frames according to the following equation:

$$SF = \sum_{k=1}^{L_F} \left[ |X_n(k)| - |X_{n-1}(k)| \right]^2 , \tag{4.4}$$

where $X_n$ is the FFT of the current audio frame and $X_{n-1}$ is the FFT of the immediately preceding frame.

**Energy ratios in sub-bands**: The energy ratios in sub-bands ($ERSB$) give a rough approximation of the energy distribution of the spectrum. We divided the spectrum of the signal into four sub-bands. For each sub-band we computed the ratio between the energy contained in that sub-band and the overall energy of the audio frame as reported in Eq. 4.5. The values of the bounds of the considered sub-band intervals are given in Eq. 4.6.

$$ERSB_n = \frac{\sum_{k=K_{n_1}}^{K_{n_2}} |X(k)|^2}{\sum_{k=1}^{L_F} |X(k)|^2}, \tag{4.5}$$

where

$$[K_{n_1}, K_{n_2}] = \begin{cases} [1, 630], & n = 1 \\ [631, 1720], & n = 2 \\ [1721, 4400], & n = 3 \\ [4401, 16000], & n = 4 \end{cases} . \tag{4.6}$$

**Volume and energy**: We calculate the volume feature (V) as the root mean square (RMS) of the amplitude value of the samples

in an audio frame:

$$V = \sqrt{\frac{1}{L}\sum_{k=1}^{L} x(k)^2}, \tag{4.7}$$

while the energy (E) is the squared-sum of the amplitude value of the audio samples:

$$E = \sum_{k=1}^{L} x(k)^2. \tag{4.8}$$

**Zero crossing rate**: The zero crossing rate (ZCR) is the rate of the sign-changes along a frame and is especially used to characterize percussive sounds and environmental noise. For a frame $x(n)$ of $L$ samples, the ZCR is computed as follows:

$$ZCR = \frac{1}{2L}\sum_{k=1}^{L-1} \left| sgn[x(k+1)] - sgn[x(k)] \right| \tag{4.9}$$

## 4.2 Second-level features (Aural words)

In order to derive a finite set of *aural words* that play the role of the words in a textual document, we have performed a quantization of the vector space of the first-level features using the well known K-Means clustering algorithm during the training phase of the system. Since this algorithm requires as a parameter the desired number of clusters $K$, a grid search was conducted to find the value that maximizes the final classification accuracy and we determined $K = 1024$.

It is worth noting that the method only requires unlabeled samples for performing the clustering. Thus for the training set it is not necessary to have a ground truth with a granularity of a single frame; this can be a significant advantage over other methods, greatly reducing the human labor time required to train the event detection system on a new set of sounds.

The output of the K-means algorithm is the set $CB$ of the $K$ centroids of the clusters, which constitutes the *codebook* of the

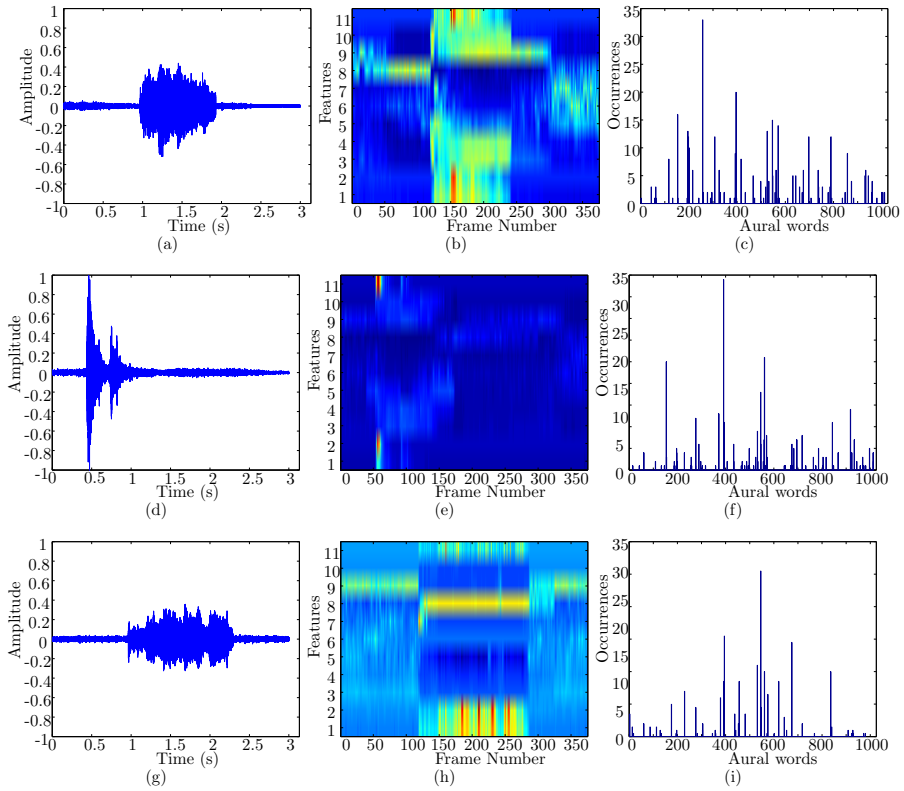



**Figure 4.3** Bag of aural words representation of 3 seconds intervals. In the first column a broken glass (a), a gunshot (d) and a scream (g) events; in the second column the corresponding first-level representations (b, e, h); in the third column the histograms of the aural words (c,f,i).

system:

$$CB = \{w_1, \ldots, w_K\} \tag{4.10}$$

Conceptually, an entry $w_i$ in the codebook can be thought as an elementary word that can be detected in the input data to be classified. We call *aural words* the entries in the set $CB$, to emphasize the fact that they are related to atomic, perceptual units of hearing, and not to linguistic units.

In the same way as the topic of a document cannot be inferred from a single word, but for a larger body of text it can be reason-

ably estimated in many cases by considering the presence or the absence of a certain number of relevant words, we assume that a single aural word is not sufficient to classify a sound event, but the presence or the absence of certain specific words over a longer time interval may lead to a reliable classification. Thus, in order to perform the classification, we compute a second-level feature vector. In Fig. 4.2, a sketch of the process of construction of the second-level feature vectors is shown. First, the input audio stream is segmented into sets of $N = 375$ partially overlapping frames, each corresponding to time intervals of 3 seconds. The value of $N$ is chosen so that an interval covers a time scale sufficient to recognize both impulsive and sustained sounds. For each frame in an interval, the first-level feature vectors $v_i$, (with $i = 1, \ldots, N$), are computed.

Then, for each feature vector $v_i$, the codebook is searched for the word $w_j$ that is closest to $v_i$, as shown in Fig. 4.2b. Let us denote as $b_i$ the index of such a word within the codebook:

$$b_i = \arg\min_j D(v_i, w_j), j = 1, \ldots, K, \tag{4.11}$$

where $D(v_i, w_j)$ is the Euclidean distance between the $i$-th vector of the interval and the $j$-th word of the codebook.

Finally, the second-level feature vector $U = (u_1, \ldots, u_K)$ is defined as follows:

$$u_j = \sum_{i=1}^{N} \delta\left(b_i, j\right), j = 1, \ldots, K \tag{4.12}$$

where $\delta(\cdot)$ is the Kronecker delta.

Thus, the second-level feature vector is the histogram of the occurrences of the aural words detected in the interval.

For instance, in Fig. 4.2 the vectors $v_1$ and $v_2$ have the aural word $w_2$ as the closest word in the codebook. Thus, the value of the bin $u_2$ of the histogram is equal to 2 (Fig. 4.2c). It means that, in the time interval to be classified, the aural word $w_2$ is present twice. In the same way, the words $w_3$ and $w_N$ are present once,

while the words $w_1$ and $w_4$ have no occurrences in the considered interval.

It is worth noting that the position of the aural words within an interval is not taken into account, as in our context it is important only the detection of the target audio event rather than the exact position. It can be considered an advantage for the building of the model of the target events, because it is not necessary to create different models for sounds that happen at different position within an interval. For instance, two gunshots, which happen respectively at the beginning and at the end of an interval, can be modeled with the same histogram of aural words.

For the experimental evaluation of the proposed approach, we considered the time window of 3 seconds that slides on the audio clip to be analyzed by steps of one second. Two consecutive time windows are thus overlapped for 2/3 of their length.

Examples of the *bag of aural words* representation of the sounds in Fig. 4.3a, 4.3d and 4.3g are respectively shown in Fig. 4.3c, 4.3f and 4.3i. On one hand, it is worth noting that the first-level representation of the sounds (Fig. 4.3b, 4.3e and 4.3h) allows to characterize the short-time properties of the events with respect to the background noise. On the other hand, the histograms of the aural words effectively describe longer time-scale intervals in which different kinds of event can occur.

## 4.3 The classifier

The second-level feature vectors are used to train a SVM classifier [124], using a labeled training set, with a ground truth defined at a time scale corresponding to an interval.

The choice of the SVM classifier is motivated by the ability of this algorithm to find a hyperplane separating the classes to be recognized that is maximally stable, in the sense that it maximizes the margin between the decision boundary and the training samples, so as to avoid overfitting on small training sets. We have used the original, linear version of the SVM, and not the kernelized

one, since it provided satisfactory results in our experiments.

The SVM (like other classifiers based on discriminant analysis, but differently from distance-based classifiers like the Nearest Neighbor) is able to construct a decision function that gives only a subset of the input features a non zero weight. In this way it can learn which are the aural words that are really discriminant for the events of interest, and ignore the others.

Since SVM is a binary classifier (i.e. it works on a two-classes problem), the proposed system is designed so as to have several SVM instances operating in parallel.Namely, we have a pool of $M + 1$ 1-vs-all SVM classifiers (where $M$ is the number of the classes to be recognized). The $i$-th classifier (with $i = 0, \ldots, M$) is trained using as positive examples the samples from class $C_i$ and as negative examples all the samples from the other classes. In the classification phase, given a second-level feature vector $U$, the $i$-th classifier produces an output $s_i$ that indicates the score of the feature vector for such classifier. Then, a combination rule (Eq. 4.13) decides the class of the vector $U$. If at least one classifier yields an output above a threshold $\tau$, the vector is assigned to the class that corresponds to the SVM that gives the maximum score (which might be the background class $C_0$, and so no event is reported). If all classifiers give a negative score, the vector $U$ is assigned to background class $C_0$. Formally, the output class $C$ is defined as follow:

$$C = \begin{cases} C_0, & if\ s_i < \tau\ \forall i = 0, \ldots, M \\ \arg\max_i s_i, & otherwise. \end{cases} \tag{4.13}$$

## 4.4 Conclusions

In this section I detailed the module proposed in this thesis for analyzing audio streams in order to detect events of interest, namely screams, gunshots and broken glasses. The effectiveness and the robustness of the proposed approach with respect to different environmental conditions will be confirmed in Section 5.4, devoted to the experimentation. Since no publicly available datasets have

been proposed up to now by the scientific community for audio surveillance applications, in this thesis we also introduce a novel dataset for benchmarking purposes, detailed in Section 5.4.1. The results obtained on this dataset have been compared with a state of the art approach, confirming the very promising performance of the proposed method.

# Chapter 5

# Experimental Results

In this chapter the results obtained by testing the different modules proposed in this thesis will be summarized. In particular:

- Section 5.1 analyzes the results obtained by the proposed tracking method, detailed in Section 3.1, over two different standard datasets, namely PETS 2010 Dataset and ISSIA Soccer Dataset, both acquired in real environments.

- Section 5.2 details the results obtained by the visual behavior analysis module, introduced in Sections 3.2 and 3.4. Three datasets, acquired in very different environments and involving both people and vehicles, have been considered, namely MIT Trajectory dataset, Edinburgh Informatics Forum Pedestrian Database and MIT Train Station dataset.

- In Section 5.3 the module devoted to store and allow spatio temporal queries, introduced in Sections 3.3 and 3.4, is tested and the results are presented. The test has been conducted both over the standard MIT Trajectory dataset and on a synthetic dataset, generated in order to stress the system.

- Finally, in Section 5.4 the results of the tests conducted over the algorithm for recognizing audio events of interest (introduced in Chapter 4) are presented. Since at our knowledge

| View | Camera Model | Resolution | Frame Rate |
|---|---|---|---|
| 1 | Axis 223M | 768x576 | 7 |
| 3 | Axis 233D | 768x576 | 7 |
| 4 | Axis 233D | 768x576 | 7 |
| 5 | Axis 223M | 720x576 | 7 |
| 6 | Axis 223M | 720x576 | 7 |
| 7 | Canon MV1 | 720x576 | 7 |
| 8 | Canon MV1 | 720x576 | 7 |

**Figure 5.1** PETS Dataset: Camera Specification [125].

there are not standard datasets, in this thesis a novel standard dataset for benchmarking purposes have been proposed and made publicly available.

## 5.1 Tracking Algorithm

In this section we will show the results of the tracking method, detailed in Section 3.1.2. In particular, in Subsection 5.1.1 we introduce the standard datasets used for the experimentation. In Subsection 5.1.2 the parameters selected for the experimentation are justified, while in Subsections 5.1.3 and 5.1.4 a quantitative and qualitative evaluation is respectively provided. Finally, the computational cost of the algorithm is show in Subsection 5.1.5.

### 5.1.1 Datasets

In order to assess the performance of the method with respect to the state of the art, we have used two different datasets: the publicly available PETS 2010 dataset [125], currently used by many research papers, and the ISSIA Soccer Dataset [126].

**PETS 2010 Dataset**: it has been recorded at Whiteknights Campus, University of Reading, UK in 2009. It is composed by three datasets: S1 concerns person count and density estimation, S2 addresses people tracking and S3 involves flow analysis and

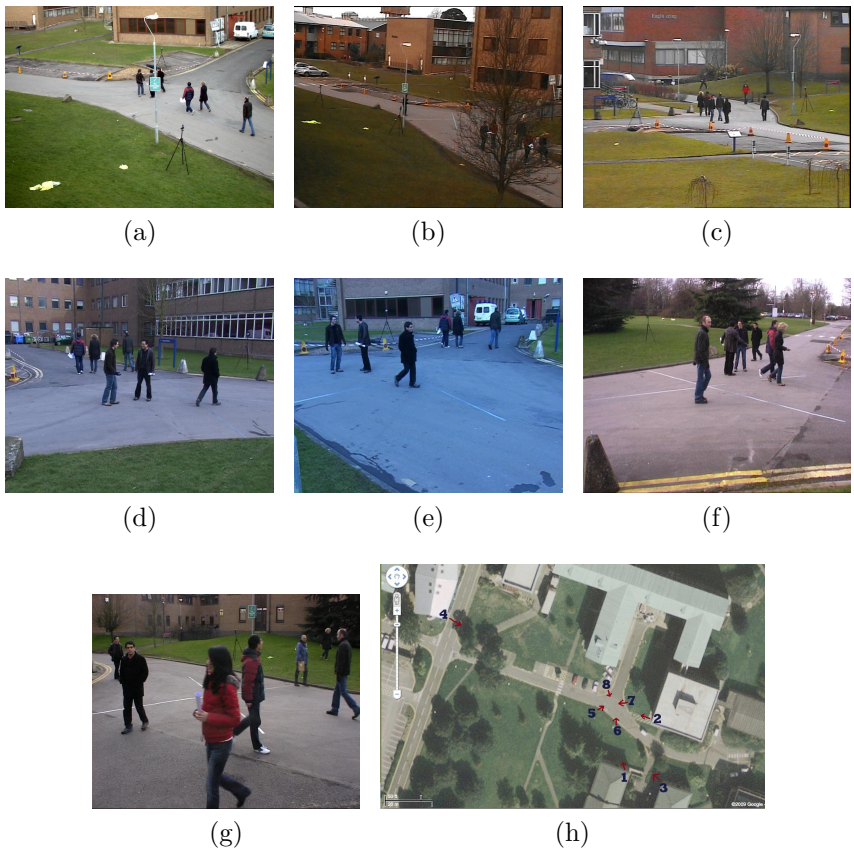



**Figure 5.2** Camera views used in the PETS2010 database. We can note that each view emphasizes one or more problem. For example, the first one (a) causes occlusions between persons and the pole while the second one (b) is characterized by several occlusions between persons and the tree. In (h) the position of the cameras on the plane is shown.

event recognition. In this thesis we focus on the dataset S2 (hereinafter PETS 2010 Dataset), made of seven videos, containing several occlusions between a person and an object, two persons or among several persons. Figure 5.2 shows an example for each considered view of the PETS 2010 database, while more information are provided in Table 5.1.

**ISSIA Soccer Dataset**: it is composed by six synchronized

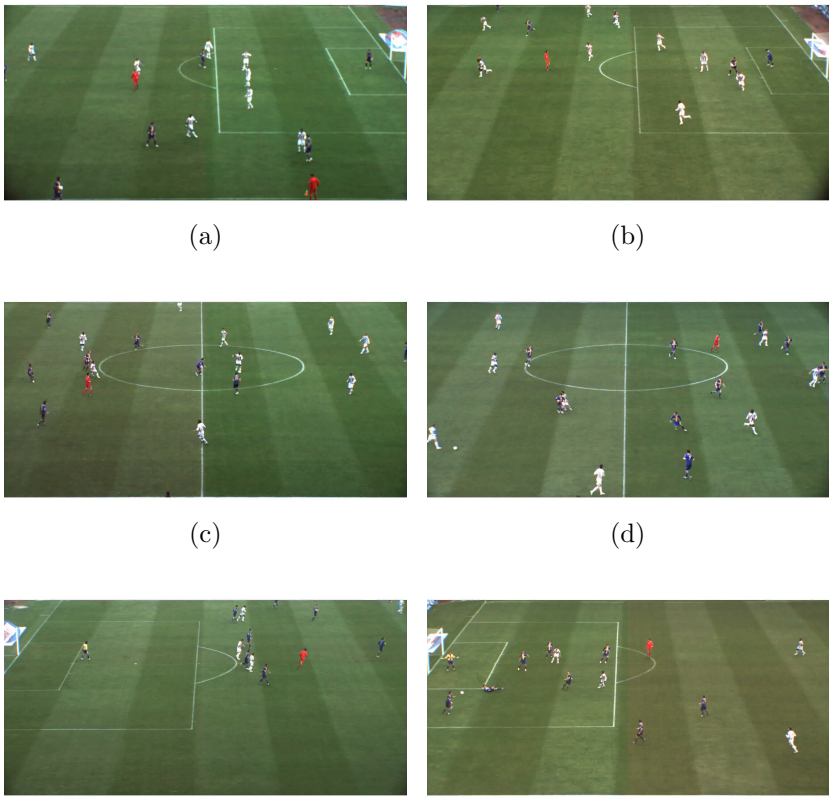


**Figure 5.3** Camera views used in the ISSIA database.

videos acquired at 25 fps during a match by six Full-HD cameras, located along the major sides of the playing-field. Examples of the different views are given in Figure 5.3; the difficulty arising during the tracking in this context is related to the rapid changes of trajectories of the players and their visual appearance: as a matter of fact, except for the referees and the goalkeepers, all the players, depending on their team, have a white or a blue uniform. Such an issue could make very difficult to recover occlusions by exploiting visual dissimilarities between involved players.

### 5.1.2 Parameters setup

In the proposed tracking method the only parameter that needs to be properly set up is the $d_{max}$ parameter of equation 3.13. In order to evaluate it, we have computed in each view the maximum speed of the objects, from which we have derived the following values: $d_{max} = 100$ for views 1, 3 and 4 and $d_{max} = 150$ for view 5, 6, 7 and 8 of the PETS dataset; on the other hand, $d_{max} = 80$ for all the views of the ISSIA Soccer Dataset.

### 5.1.3 Quantitative Evaluation

In this section a quantitative evaluation of the proposed method over both the datasets is performed.

#### 5.1.3.1 Experimentation 1

The first quantitative evaluation of the method has been carried out using the performance indexing proposed in [127]. In particular, we have used the following indices, especially suited for tracking algorithms: the *Average Tracking Accuracy* (ATA), the *Multiple Object Tracking Accuracy* (MOTA) and the *Multiple Object Tracking Precision* (MOTP). In the following we introduce some notations useful to formally define them.

Let $G_i$ and $D_i$ be the $i$th ground truth object and the $i$th detected one at the sequence level, respectively; $G_i^{(t)}$ and $D_i^{(t)}$ denote the $i$th ground truth object and the detected one in frame t; $N_G^{(t)}$ and $N_D^{(t)}$ denote the number of ground truth objects and detected ones in frame t, respectively, while $N_G$ and $N_D$ denote the number of ground truth objects and unique detected ones in the given sequences. $N_{frames}$ is the number of frames in the sequences. Finally, $N_{mapped}$ refers to the mapped system output objects over an entire reference track, taking into account splits and merges and $N_{mapped}^{(t)}$ refers to the number of mapped objects in the frame $t$.

ATA is a spatiotemporal measure that penalizes fragmentations in spatiotemporal dimensions while accounting for the number of objects detected and tracked, missed objects, and false pos-

itives. ATA is defined in terms of Sequence Track Detection Accuracy (STDA):

$$STDA = \sum_{i=1}^{N_{mapped}} \frac{\sum_{t=1}^{N_{frames}} \frac{|G_i^{(t)} \cap D_i^{(t)}|}{|G_i^{(t)} \cup D_i^{(t)}|}}{N_{G_i \cup D_i \neq 0}}. \tag{5.1}$$

The latter measures the overlap in the spatiotemporal dimensions of the detected object over the ground truth, taking a maximum value of $N_G$. The ATA is defined as the STDA per object:

$$ATA = \frac{STDA}{\left[\frac{N_G + N_D}{2}\right]}. \tag{5.2}$$

As already mentioned, the MOTA is an accuracy score that computes the number of missed detections, false positives and switches in the system output track for a given reference ground truth track. It is defined as:

$$MOTA = 1 - \frac{\sum_{t=1}^{N_{frames}} \left(c_m \cdot m_t + c_f \cdot fp_t + c_s(is_t)\right)}{\sum_{t=1}^{N_{frames}} N_G^{(t)}}, \tag{5.3}$$

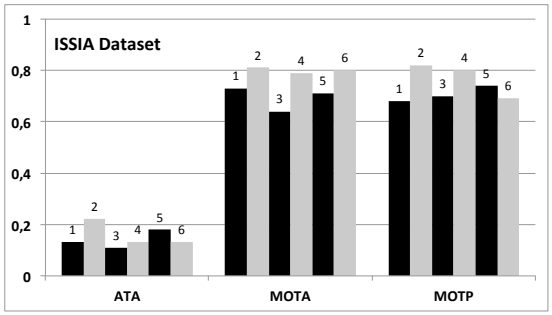
where $m_t$ is the number of misses, $fp_t$ is the number of false positives, and $is_t$ is the number of ID mismatches in frame $t$ considering the mapping in frame $(t-1)$; c values are weights chosen as follows:

$$c_m = c_f = 1; c_s = log_{10}(\cdot).$$

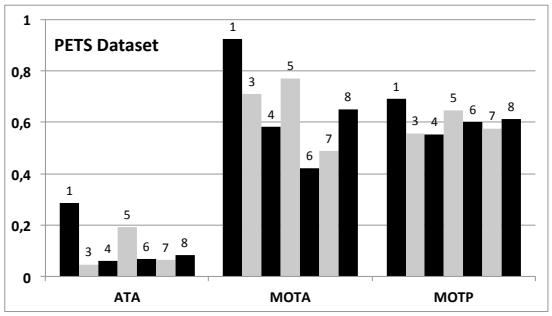
Finally, the MOTP is a precision score that calculates the spatiotemporal overlap between the reference tracks and the system output tracks:

$$MOTP = \frac{\sum_{i=1}^{N_{mapped}} \sum_{t=1}^{N_{frames}^{(t)}} \frac{|G_i^{(t)} \cap D_i^{(t)}|}{|G_i^{(t)} \cup D_i^{(t)}|}}{\sum_{t=1}^{N_{frames}} N_{mapped}^{(t)}}. \tag{5.4}$$

The obtained results are shown in Figure 5.4: in particular, Figure 5.4a summarizes the performance obtained over the ISSIA Soccer dataset, highlighting similar values over all the different




(a)




(b)

**Figure 5.4** Performance of the proposed method for the considered views of the PETS 2010 (a) and of the ISSIA Soccer (b) datasets. The numbers on the bars correspond to the views number.

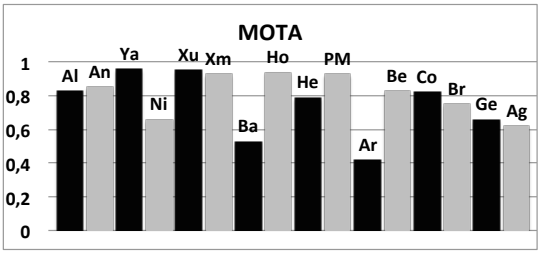
views. Note that the performance are even better at the light of the fact that this dataset contains some very complex occlusion patterns, involving a variable number of players, ranging from two to eight, that strongly influence the performance of the entire system. Although a direct comparison with the state-of-the-art methods is not possible as the results reported over this dataset are usually provided in a qualitative form, the high values obtained by our method both in terms of accuracy and precision confirm the validity of the proposed approach in the sport applicative domain.

Figure 5.4b shows the results of the proposed method over the PETS 2010 dataset, related to the individual sequences. We can note that the performance is strongly influenced by the complexity

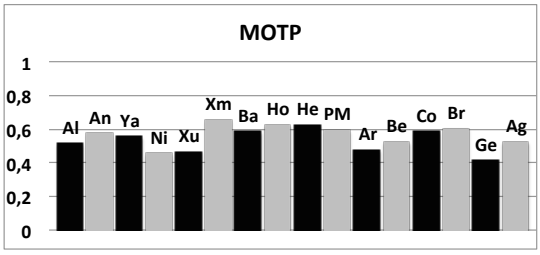
of the single sequence. This complexity is not determined only by the typology of the single view (*i.e.*, the presence of the pole in the first view or the presence of the tree in the third one, which covers one-third of the scene taken by the camera), but also by the interactions among the tracked objects.

At this point we can examine in detail the considered views, analyzing their performance in relation to the complexity of the scene. First view presents interactions among two or three objects; the only difficulty is due to the presence of the pole and of the sign hanged on it, which causes a lot of splits. Note that the proposed method proves to be particularly robust with respect to the split situations on this view. Views 3 and 4 are the most complex, as shown by the results displayed in Figure 5.4b. Indeed, as already mentioned, view 3 is characterized by the presence of a large tree (about one-third of the scene), occluding a lot of individual or group objects. The situation is further complicated by the complexity of interactions among the objects, which involves in the average $2-5$ objects for view 3 and $2-6$ for view 4. Another problem in view 4 is the presence of a white-orange ribbon, continuously moving because of the wind. Such situation causes a lot of problems also in the detection phase. The problem of the moving ribbon is also present in views 5, 6, 7 and 8, even if it is less visible. We can note that the performance obtained in views 6 and 7 is generally lower than that obtained on other sequences; this is related to more complex interactions between the tracked objects, having a very high number of occlusions associated to objects that are entering the scene (unstable objects). It is worth noting that the method, during an occlusion, does not attempt to find the exact position of an object inside a group; it continues to track the group as a whole, using the Kalman filter for obtaining a prevision of the position of each object inside the group itself; this choice obviously causes a degradation of the performance if it is measured using indices defined assuming that objects are always tracked individually.

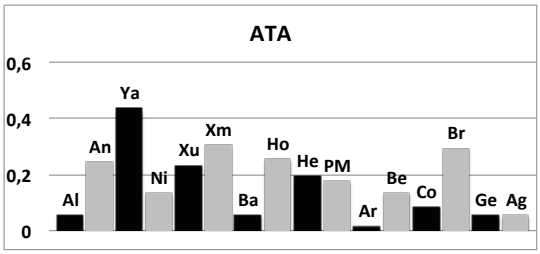
**Comparison**: PETS Contest (Performance Evaluation of Tracking and Surveillance) is a competition organized by the University




(a)




(c)




(d)

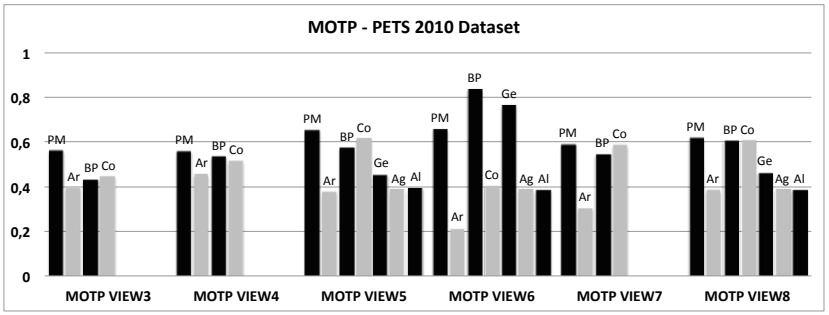
| Method | Abbrev. |
|---|---|
| Alahi Olasso | Al |
| Anon | An |
| Yang | Ya |
| Nie | Ni |
| Xu | Xu |
| Xu multiview | Xm |
| Badie | Ba |
| Hoffman | Ho |
| Heili | He |
| Proposed Method | PM |
| Arsic | Ar |
| Berclaz | Be |
| Conte | Co |
| Breitenstein | Br |
| Ge | Ge |
| Alahi Ogreedy | Ag |

(b)

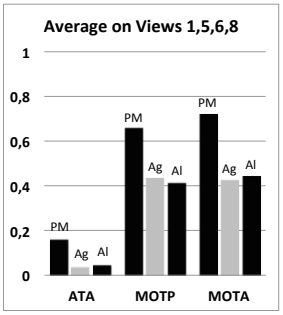
**Figure 5.5** Comparison of the proposed method with the participant to the last PETS 2014 competition, in terms of MOTA (a), MOTP (c) and ATA (d). In (b) the associations between methods and abbreviations is summarized.

of Reading, which allows to compare all the state of the art tracking algorithms: each participant has to submit the output of the proposed method over a standard dataset; the output is processed by the organizers and the results are finally disclosed during the contest session.

The proposed algorithm participated to the last PETS 2013 contest [105] and, as highlighted by the organizers, it ranked first



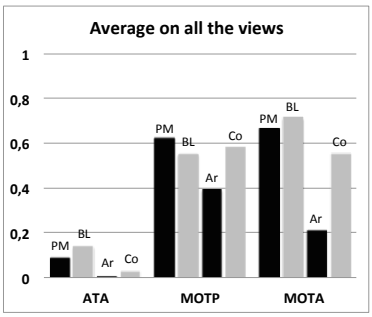

(a)




(b)

| Method | Abbrev. |
|---|---|
| Proposed Method | PM |
| Breitenstein | Br |
| Leykin | Le |
| Sharma | Sh |
| Yang | Ya |
| Berclazdp | BD |
| Berclazlp | BL |
| Arsic Winter | AW |
| Conte | Co |
| Ge | Ge |
| Alahi Ogreedy | Ag |
| Alahi Olasso | Al |
| Arsic | Ar |

(c)




(d)

**Figure 5.6** Performance of the proposed method compared with the PETS 2010 contest participants on MOTP index (a), on Views 1, 5, 6 and 8 (b) and on all the views (e). In (c) the associations between methods and abbreviations is summarized.

in terms of MOTA and in the first positions in terms of ATA and MOTP. Although the results have not been made available, in Fig-

ure 5.5 we report the ones computed over View 1, declared during the final contest session. Our method strongly outperforms all the other ones working in real time and based on a single camera view: in fact, it is worth to point out that Breitenstein et al. (Br) [47] and Xu et al. (Xu) [30] use a multi-camera approach, while Badie et al. (Ba) [21] and Hoffman et al. (Ho) [22] consider a post-processing of the trajectories in order to link all the extracted tracklets. It should be clear that our method extracts the objects' positions at each frame, without applying any kind of post-processing aimed at linking the tracklets. It is a very important and not negligible feature in the field of behavioral analysis, since we are interested in detecting abnormal behaviors in real time, when the objects are still inside the scene.

Furthermore, a deeper comparison has been performed with the methods participating to the previous PETS 2010 contest, whose results are available in [128]. Figure 5.6 summarizes the obtained results. In particular, Figure 5.6a gives an overview of the precision index (MOTP) over all the views. It is evident that the proposed method outperforms all the other methods on six out of seven considered views in terms of precision. Figure 5.6b provides the average performance, in terms of ATA, MOTA and MOTP obtained on Views 1, 5, 6 and 8, the only ones taken into account by Alahi et al. [129].

Finally, Figure 5.6d shows the average results over all the views: our method is the most precise over the entire dataset and it is outperformed, in terms of accuracy, only by the method proposed by Berclaz *et al.* [33], which takes advantage of the use of a multi-camera approach and thus it is not directly comparable with our approach.

It is worth highlighting that ATA, MOTA and MOTP do not perfectly fit, for their nature, the proposed method. It is due to the fact that our approach, during an occlusion, does not attempt to find the exact position of an object inside a group; it continues to track the group as a whole, using the Kalman filter for obtaining a prevision of the position of each object inside the group itself; this choice obviously causes a degradation of the performance if

| View | *Split* *Resolution Percentage* | | *Occlusion* *Resolution Percentage* | |
|---|---|---|---|---|
| | **[130]** | **Proposed** | **[130]** | **Proposed** |
| **1** | 45% | 73% | 75% | 95% |
| **3** | 36% | 51% | 61% | 76% |
| **4** | 35% | 74% | 45% | 74% |
| **5** | 15% | 51% | 56% | 67% |
| **6** | 12% | 56% | 51% | 62% |
| **7** | 16% | 61% | 52% | 63% |
| **8** | 20% | 51% | 42% | 59% |

**Figure 5.7** Comparison between the proposed method and [130] in terms of split and occlusion patterns resolution over the PETS dataset.

measured using indices assuming that objects are always tracked individually.

In general, our method confirms a very high accuracy and precision; this result is mainly a direct consequence of the fact that it solves many of the errors usually occurring in tracking algorithms that do not distinguish between single and multiple objects.

#### 5.1.3.2 Experimentation 2

In this section the performance of the proposed method in terms of resolution percentage of split and occlusion patterns is analyzed. The results are summarized in Table 5.7, where a comparison with our previous method [130] is performed over the PETS dataset. Note that [130] is more robust with respect to the occlusion occurrences, rather than to the split ones. It is mainly due to the association manager module, which uses a greedy strategy to solve split and occlusion patterns. The main novelty of the proposed approach lies in the introduction of a graph based approach, which proves to significantly improve the performance with respect to [130], both in terms of split and occlusion patterns.

| PETS Dataset | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **V** | **GT** | **TP** | **PTP** | **FP** | **FN** | **IDS** | **Av** | **Min** | **Max** | **AvI** |
| 1 | 21 | 21 | 4 | 2 | 0 | 43 | 232,6 | 86 | 575 | 2 |
| 3 | 21 | 15 | 0 | 7 | 6 | 82 | 306,4 | 72 | 792 | 3 |
| 4 | 23 | 18 | 0 | 19 (13) | 5 | 89 | 264,5 | 23 | 792 | 3 |
| 5 | 28 | 28 | 1 | 4 | 0 | 81 | 97,3 | 22 | 293 | 2 |
| 6 | 33 | 32 | 1 | 28(20) | 1 | 142 | 95,3 | 7 | 320 | 4 |
| 7 | 31 | 28 | 0 | 17 | 3 | 123 | 148,1 | 32 | 320 | 3 |
| 8 | 30 | 25 | 1 | 6 | 5 | 95 | 143,8 | 13 | 417 | 3 |

| ISSIA Dataset | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **V** | **GT** | **TP** | **PTP** | **FN** | **FP** | **IDS** | **Av** | **Min** | **Max** | **AvI** |
| 1 | 21 | 19 | 4 | 2 | 24 | 67 | 523,2 | 1 | 2474 | 3 |
| 2 | 28 | 24 | 7 | 3 | 5 | 33 | 470,7 | 1 | 2597 | 1 |
| 3 | 57 | 50 | 23 | 8 | 36 | 69 | 667,7 | 1 | 2177 | 1 |
| 4 | 113 | 71 | 19 | 38 | 28 | 81 | 340,9 | 1 | 2051 | 1 |
| 5 | 28 | 21 | 7 | 7 | 15 | 46 | 564,4 | 1 | 2597 | 2 |
| 6 | 22 | 20 | 8 | 2 | 23 | 55 | 626,2 | 1 | 2597 | 2 |

**Figure 5.8** Tracking results on PETS and ISSIA Soccer Dataset. (V: View; GT: Ground Truth trajectories; TP: True Positives, at least 75% of the track without id-switches; PTP: Perfect True Positives, 100% of the track without id-switches; FN: False Negatives; FP: False Positives; IDS: Id Switches; Av: Average trajectory length; Min: Minimum trajectory length; Max: Maximum trajectory length; AvI: Average number of id-switches).

#### 5.1.3.3 Experimentation 3

A further experimentation, shown in Table 5.8, presents some more general evaluation criteria, which reflect the possibility to correctly follow a trajectory, assigning it one or more id [131]. In particular:

- *TP* (True Positive) refers to the number of trajectories followed for more than the 75% of their life, also with different identifiers. An example of *TP* is shown in Figure 5.9a;
- *PTP* (Perfect True Positive) refers to the number of trajectories followed for the 100% of their life with the same identifier. An example of *PTP* is shown in Figure 5.9b;
- *FN* (False Negative) refers to the number of trajectories followed for less than the 75% of their life;
- *FP* (False Positive) refers to the number of spurious trajectories followed for more than two seconds;

- *IDS* (ID-Switch) refers to the number of times an object changes its identifier. Figure 5.10 shows the histogram of the id-switch number for both the considered datasets.
- *Av* refers to the average trajectories length;
- *Min* refers to the minimum trajectories length;
- *Max* refers to the maximum trajectories length;
- *AvI* refers to the average number of id-switches for each trajectory;

In particular, the first part of the table refers to the PETS dataset, while the second part refers to the ISSIA Soccer dataset. It is worth noting that the high number of false positive trajectories, especially in Views 4 and 6 of the PETS dataset, is caused by very frequent detection errors: in particular, as already mentioned, in View 4 it is caused by the presence of the white-orange moving wire, while in View 6 it is related both to the presence of the wire and to the white car, wrongly identified as an object by the detection phase during all the sequence. All these kinds of detection errors, identified in Table 5.8 by the numbers in brackets, could be reduced in a very simple way, by applying a filter to the detection phase, taking into account the particular shape and appearance of the spurious objects. Since we are only interested in the proposed tracking method, we do not investigate into the performing of the detection phase.

### 5.1.4 Qualitative Evaluation

A qualitative evaluation has been finally performed in order to confirm the efficiency of the proposed approach. In particular, we show how the proposed algorithm deals with splits and occlusions respectively in Figure 5.11 and 5.12: in particular, Figure 5.11 reports an example of split caused by an error during the detection phase and properly adjusted. On the other hand, Figures 5.12 shows some excerpts of the video sequences, with two complex

(a) (b)

**Figure 5.9** Examples of true positive trajectory (a) and perfect true positive trajectory(b).

occlusion patterns among three and two persons; as it can be seen, the system preserves the object identities across the occlusions.

Finally, we show in Figure 5.13 the trajectories obtained by applying the proposed algorithm over the PETS 2010 Dataset. Despite the complexity of each view, the trajectories are generally stable and reliable, so confirming the goodness of the proposed approach.

### 5.1.5 Computational cost

In order to evaluate the performance of the proposed method in terms of computational cost, we have computed the time needed to process a single frame (both detection and tracking steps) by considering different images resolutions and different number of video streams.

The middleware platform detailed in [132], installed on an Intel Xeon processor running at 3.0GHz, has been used for the experimentation.

The obtained results are summarize in Figure 5.14: we can note that the time required to process a single frame linearly increases by increasing the number of streams (1, 2, 4, 6, 8, 10, 12, 14, 16) as well as the image resolution (1/8, 1/4, 1/2, 1).

Furthermore, it is important to highlight that in the best case, arising when a single stream is processed, the entire process, from

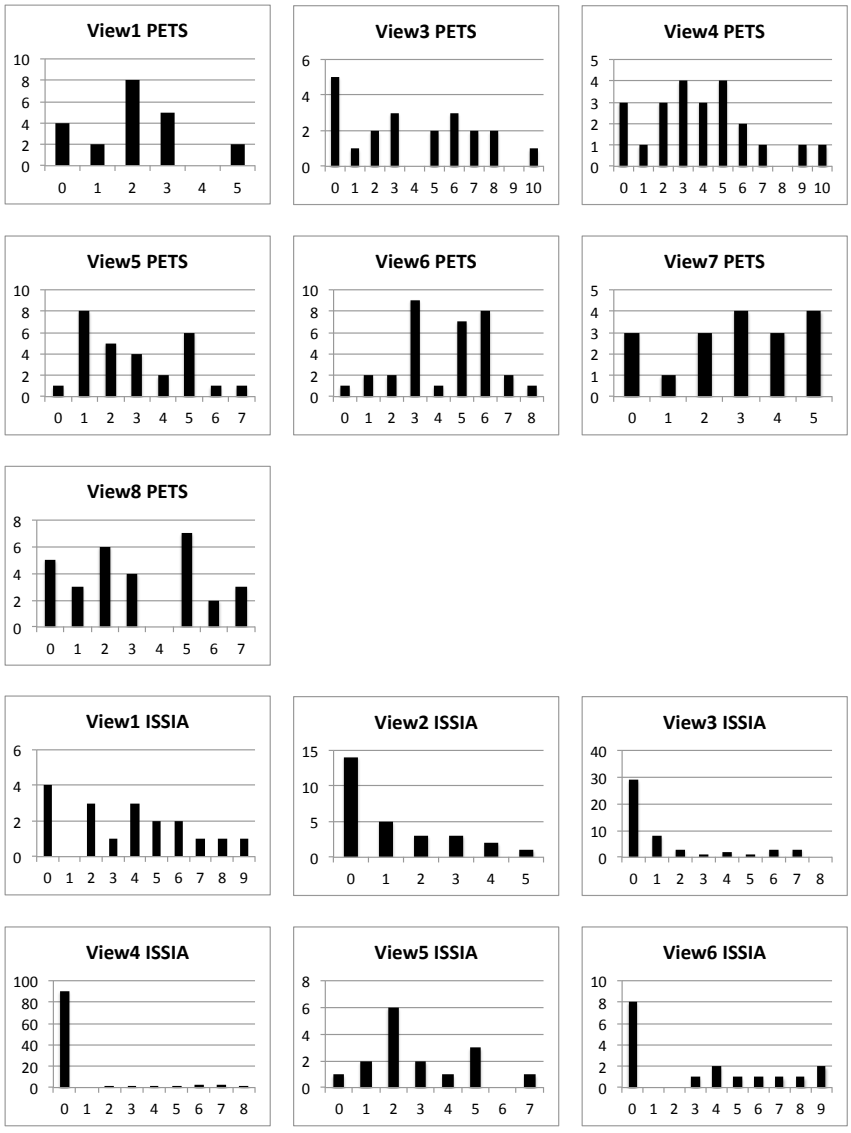



**Figure 5.10** Histogram of id-switches for all the views of the PETS and ISSIA datasets. In each graphic, the column refers to the number of trajectories having during their life zero or more id-switches; the row refers to the number of id-switches.

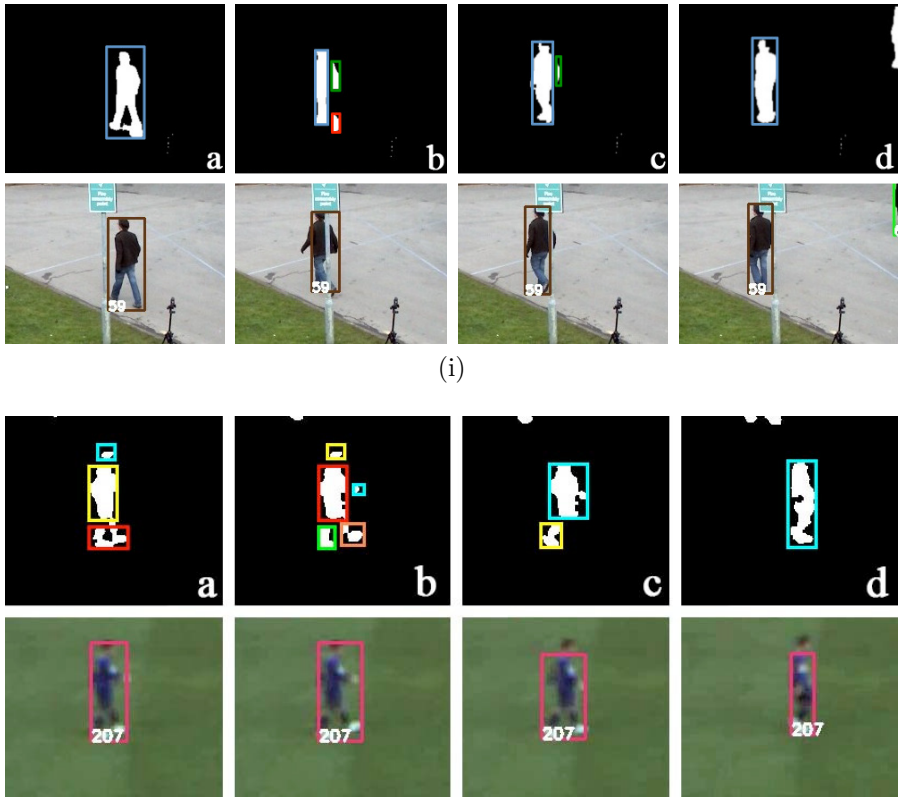
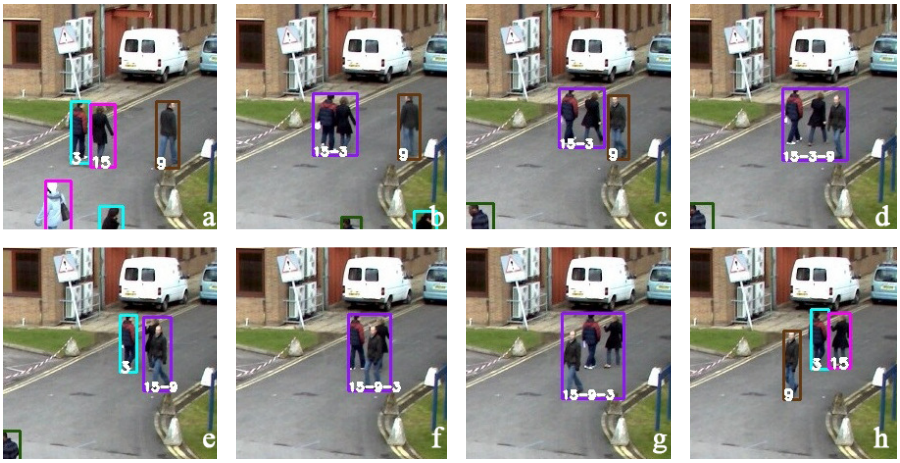



**Figure 5.11** The output of the proposed method on two sequences from the PETS (i) and the ISSIA Soccer (ii) datasets, both containing a split; the first and the second row respectively represent the input and the output of the method.

detection to tracking, runs approximatively at 35 fps on 4CIF images, confirming its usability in real time applications.

The promising results obtained by the proposed method over two very different application fields, a video-surveillance domain (PETS dataset) and a sports one (ISSIA Soccer Dataset), confirm the robustness of the approach and its applicability to any real-time context. Almost all the problems arising during the detection step, ranging from split blobs to undetected objects, are easily managed by the different states and complex occlusions correctly




(i)


(ii)

**Figure 5.12** The output of the proposed method over two sequences of the PETS 2010 dataset (i) and of the ISSIA Soccer Dataset (ii) containing an occlusion. Note how the object 9 in (i) is correctly tracked inside the different groups although it quickly changes its direction in the frame (c).

solved thanks to the introduction of the group objects.

## 5.2 Visual Behavior Analysis

In this section we will detail the results obtained by the method for visual behavior understanding detailed in Section 3.2. Furthermore, the two typologies of interactions based on the above mentioned method, namely anomaly detection (Subsection 3.4.1) and queries by sketch (Subsection 3.4.2) will be tested and the results will be analyzed. In particular, a description of the considered datasets and a discussion on the setup of the parameters

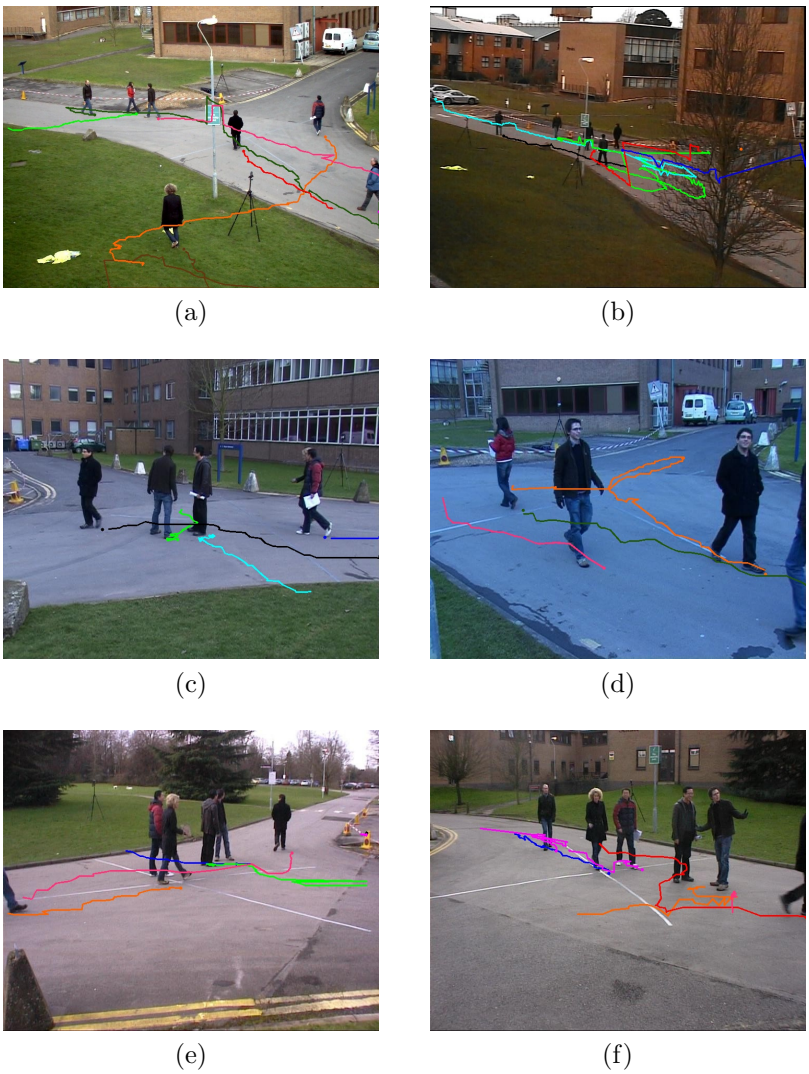


**Figure 5.13** Output of the proposed algorithm for Views 1 (a), 3 (b), 5 (c), 6 (d), 7 (e) and 8 (f) of the PETS dataset.

of our method will be provided respectively in Subsections 5.2.1 and 5.2.2; Subsection 5.2.3 will highlight the expressive power of the string based representation and in Subsection 5.2.4 we will show the performance of the clustering algorithm, while in Sub-

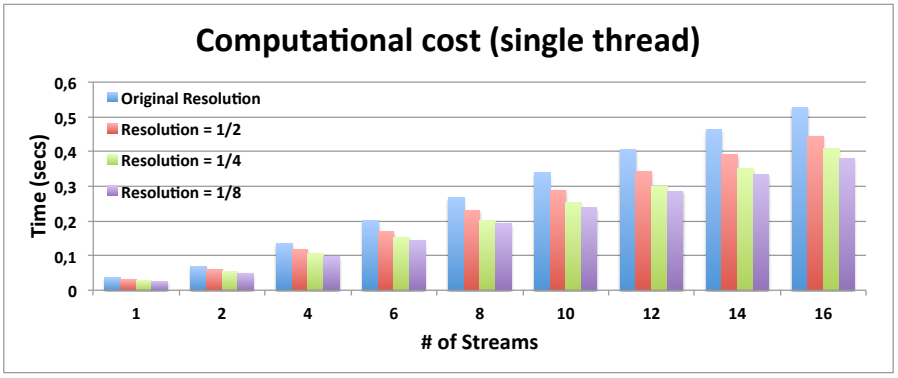
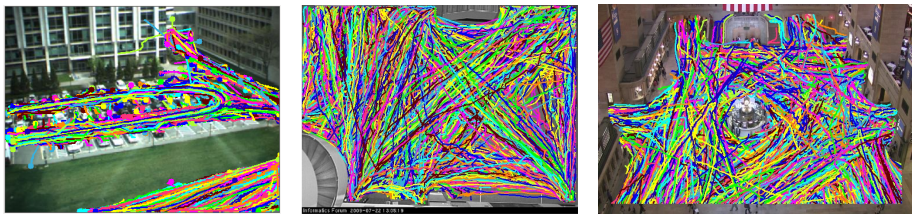



**Figure 5.14** Computational cost of the tracking algorithm by varying the resolution and the number of video streams.



(a) (b) (c)

**Figure 5.15** MIT1 (a), EDH (b) and MIT2 (c).

section 5.2.5 we will focus on the classification method. Finally, in Subsection 5.2.6 more details about the results obtained by using the proposed method for solving query by sketch will be proposed.

### 5.2.1 Datasets

Three different well-known datasets (Figure 5.15), acquired in very different environments, have been used in order to test the proposed system:

- **MIT Trajectory dataset** [106] (hereinafter MIT1) 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** [133] (hereinafter EDH) 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 MIT1, it has been manually labeled and approximatively 90% of them have been evaluated as normal.

- **MIT Train Station dataset** [134] (hereinafter MIT2) 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 to also deal with the typical errors of tracking algorithms as well as noisy trajectories.

### 5.2.2 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 5.2.5 we will show that the exact number of zones does not significantly influence the performance of the system, and the range $|Z| \in \{20, .., 40\}$ is a good compromise for all the considered datasets.

The parameters of the kernels that we need to tune are respectively $O$ (for the Triangular Kernel) and $\sigma$ (for the Speed and Shape Kernel). In particular, in [114] 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 [114] it is suggested to try $\sigma \in \{0.1, 1, 10\} \cdot K$, where $K = median||\theta(x_i) - \theta(y_i)||\sqrt{median(||x_i||)}$ and to use higher multiples (*e.g.* 2,5).

- The *Triangular* parameter $O$ 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 5.2.4 we will show how the performance of the clustering algorithm varies by using different thresholds.

### 5.2.3 String representation: experimental results

Different trajectory representations and different similarity measures between these representations have been proposed in this paper: decomposition of the trajectory into a sequence of traversed zones, encoding of the speed and shape of each trajectory within each zone and encoding of a similarity measure between zones

through a weighted graph (Section 3.2.4). In order to measure the insight provided by these different representations, we propose to compare the different Area Under Curve (AUC) obtained by a one nearest neighbor classifier based on different trajectory representations and different kernels. To that end, we consider the cluster represented in Figure 3.26(c). Given one trajectory representation, and one kernel between these representations, we consider each trajectory of this cluster and sort increasingly all the remaining trajectories of the dataset according to their distance to this trajectory. This distance is associated to each kernel and defined as:

$$d^2(s, s') = 2\left(1 - k(s, s')\right),$$

where $k(\bullet, \bullet)$ denotes the considered kernel.

Different thresholds on this distance provide a ROC curve and hence an AUC. The mean AUC value for all trajectories within the cluster is computed for each kernel and trajectory representation. These results are summarized in Table 5.16.

Note that the time series kernel represents the string kernel computed on the whole set of points of a trajectory; this kernel obtains a low AUC of 0.6. On the other hand, the Dirac kernel is based on a string representation solely based on the sequence of traversed zones. Using this kernel, we obtain an AUC of 0.86. This last result demonstrates the insight provided by the notion of zone. The introduction of a notion of similarity between zones through the Weighted Dirac Kernel allows to increase the AUC up to 0.89. The combination of this last kernel with a description of the speed and shape of trajectories within each zones allows to get the highest AUC of 0.91.

Finally the last column of Table 5.16 shows the mean execution time required to compute a kernel value for each type of kernel. We can remark that the use of zones significantly reduces the size of strings and hence execution times. Moreover, all kernels using zones have approximately equivalent execution times.

| Kernel | AUC | Time (msecs) |
|---|---|---|
| Dirac Kernel ($k_Z$, Eq. 3.28) | 0.86 | $0.48 \cdot 10^{-3}$ |
| Dirac Kernel, Speed and Shape Kernel ($k_{ZSS}$, Eq. 3.32) | 0.89 | $0.53 \cdot 10^{-3}$ |
| Weighted Dirac Kernel ($k_{WZ}$, Eq. 3.29) | 0.89 | $0.50 \cdot 10^{-3}$ |
| Weighted Dirac Kernel, Speed and Shape Kernel ($k_{WZSS}$, Eq. 3.32) | **0.91** | $0.56 \cdot 10^{-3}$ |
| Time Series | 0.60 | $0.97 \cdot 10^{-2}$ |

**Figure 5.16** Expressive power of the string based representation. Each row contains information about the particular considered kernel, the mean AUC and the average time required to compute a single normalized kernel.

### 5.2.4 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 [135] is often used in order to evaluate the performance of the clustering algorithms. It is defined as:

$$C = \frac{S - S_{min}}{S_{max} - S_{min}}, \tag{5.5}$$

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 5.17 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

| Method | MIT1 | | EDH | | MIT2 | |
|---|---|---|---|---|---|---|
| | **C-Index** | **Time** | **C-Index** | **Time** | **C-Index** | **Time** |
| Proposed Method | **0.07** | 0.53 | **0.08** | 0.19 | **0.18** | 0.72 |
| Kernel k-Means | 0.29 | 0.04 | 0.31 | 0.04 | 0.36 | 0.04 |
| Global k-Means | 0.23 | 40 | 0.27 | 63 | 0.19 | 67 |
| Fast Global k-Means | 0.32 | 0.13 | 0.34 | 0.10 | 0.27 | 0.13 |
| PCA+k-Means | 0.26 | 10 | 0.28 | 3 | **0.18** | 4 |

**Figure 5.18** Comparison with state of the art approaches in terms of C-index and time (expressed in hours).

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 5.17, 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. We can also observe in Figure 5.17 that the evolution of the C-index according to the threshold strongly depends on the particular dataset and on the homogeneity of trajectories belonging to it.

In fact, the number of clusters is a decreasing function of the threshold. By varying the number of clusters, we induce a variation of the number n of pairs of trajectories in a same cluster. This value of n has a direct influence on both the numerator and the denominator of the C-index, hence leading to results difficult to predict. The C-index is thus not specifically designed to select an optimal number of clusters but rather to compare different clustering methods using a same number of clusters. This type of protocol is applied in Table 5.18 explained bellow.

**Qualitative Evaluation**: a qualitative evaluation of the algo-

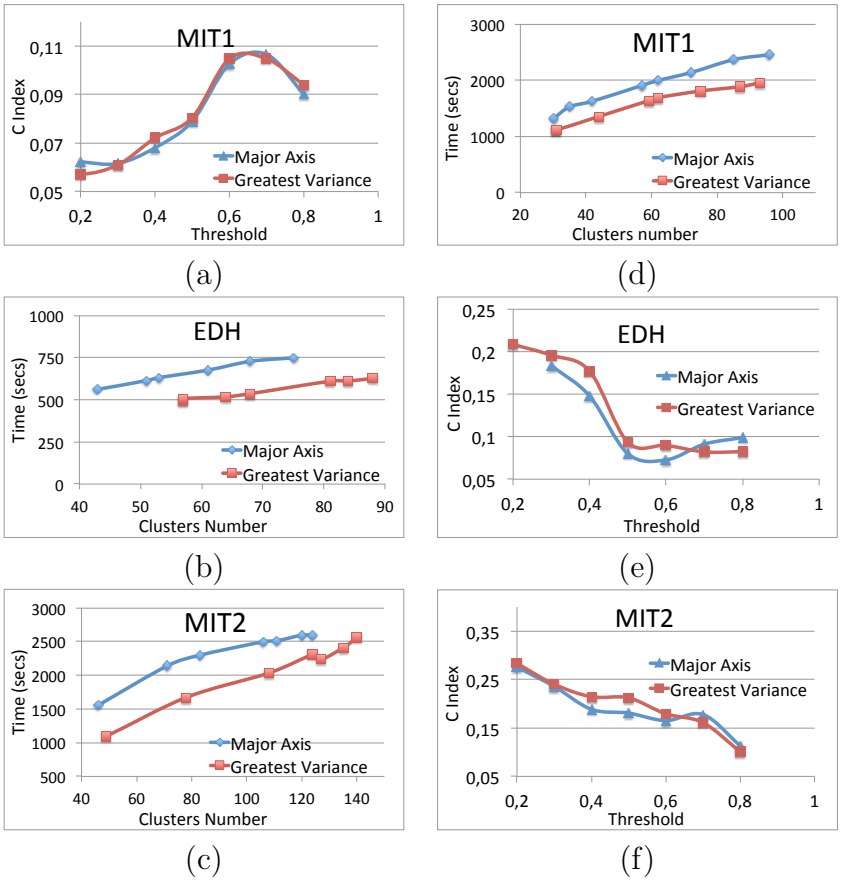



**Figure 5.17** Performance of the proposed clustering algorithm for the different datasets (MIT1 (a,d), EDH (b,e) and MIT2 (c,f)) in terms of computational costs and c-index.

rithm is provided in Figure 5.19. 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

5.19(a)). On the other hand, in the other datasets the trajectories belong to people freely moving 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 results obtained by the proposed clustering algorithm are really promising, as shown in Figures 5.19(b) and 5.19(c).

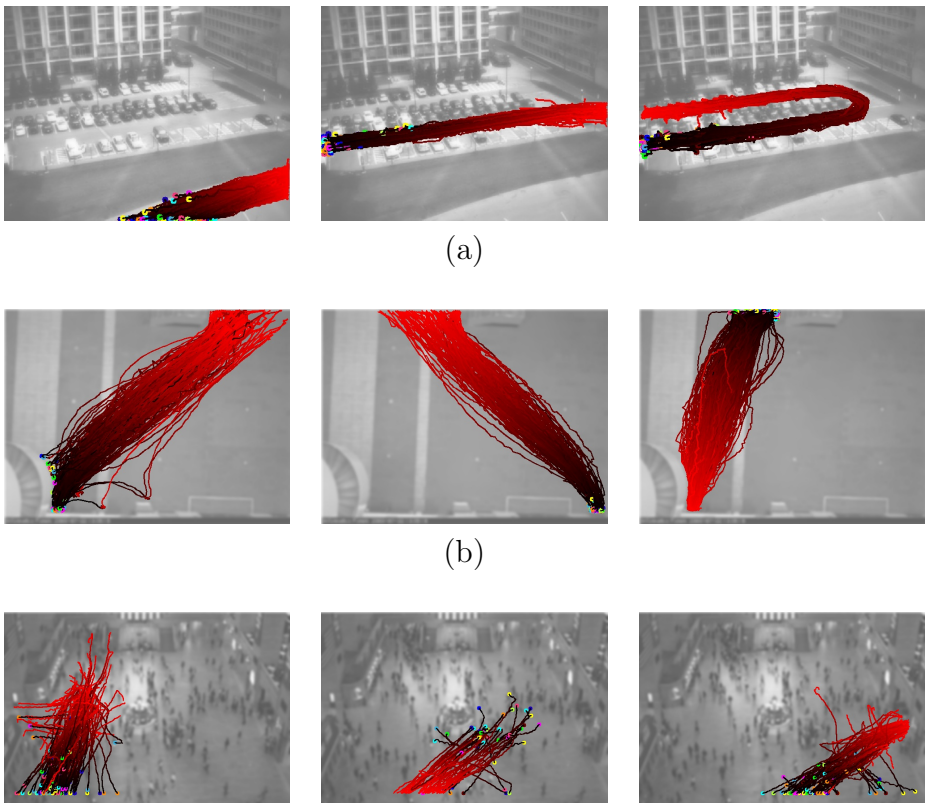


**Figure 5.19** Some of the representative clusters for the three considered datasets: MIT1 (a), EDH (b) and MIT2 (c).

**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 computational cost: in particular, the traditional Kernel $k$-means and two improved

versions, the Global Kernel $k$-means and the Fast Global Kernel $k$-means [70], are considered. A standard implementation of the Kernel K-means has been used and can be found in [136], 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 of the centroids 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. This dependency to the initial guess is one of the main difference between the proposed hierarchical clustering method based on a recursive subdivisions and the $k$-means based approaches: $k$-means methods converge to the closest local optima from the initial guess. On the other hand, our method uses heuristics based on the statistics of data in order to provide a solution which may not be locally optimum but which is usually close from a good optima.

The results for the different datasets are shown in Table 5.18. 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* [137]: 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 loose 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

furthest point in particular becomes meaningless. This issue can be overcome by reducing the dimensionality of the space [138]: the kernel 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 our method, based on successive projections, does not use a global initial projection which may induce an important loss of information, but instead analyzes at each iteration all the information pertaining a given cluster in order to find its major axis. It means that it reduces the dimensionality of the data without paying it in terms of global loss of information.

Furthermore, a qualitative comparison has been performed on the MIT2; Figure 5.20 shows some representative clusters obtained respectively by [139], [140] and [134]. The above mentioned approaches are good examples of distance-based [139] and model based [140][134] approaches, being respectively based on Hausdorff distance-based spectral clustering [139], hierarchical Dirichlet processes (HDP) [140] and Mixture model of Dynamic pedestrian-Agents (MDA) [134]. We can note that our clusters in Figures 5.20(d) and 5.20(e) seem qualitatively more compact than in Figures 5.20(a) and 5.20(b). On the other hand clusters in Figures 5.20(c) and 5.20(f) seem approximately equivalent while varying differently around a same mean trajectory.

### 5.2.5 Anomaly Detection: experimental results

The classification algorithm has been tested over two of the considered datasets, namely the MIT1 and the EDH. The absence of the MIT2 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

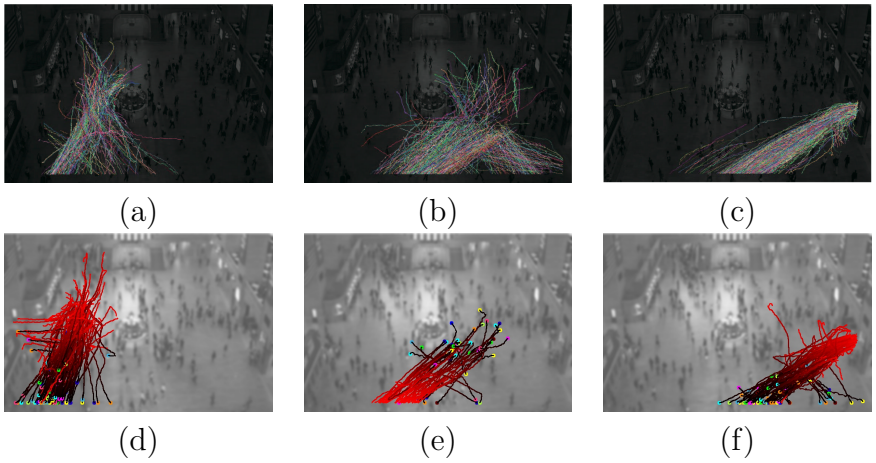
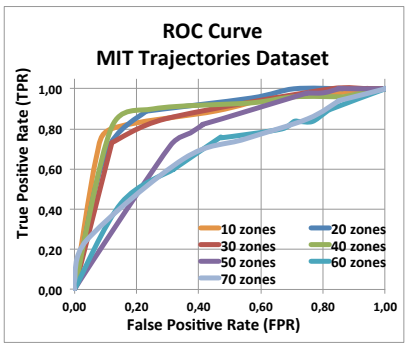



**Figure 5.20** MIT2: in first row the clusters belong to the following methods: (a) Spectral Clustering [139], (b) HDP [140] and MDA mode [134]. These figures have been provided by the authors of [134]. In second row (d,e,f) the corresponding clusters obtained by the proposed method are depicted.

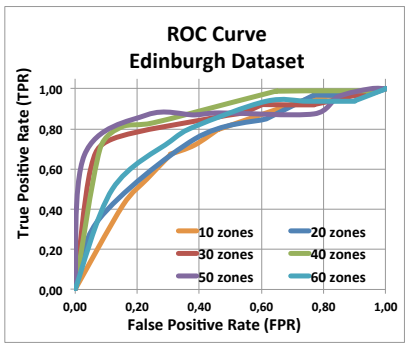
dataset $D^*$ has been randomly partitioned into three folds and one of these has been used for the learning phase. The remaining two folds have been mixed with the remaining trajectories ($D \setminus D^*$) and are used to test the system. Finally, a cross-validation strategy has been adopted in order to obtain the results presented in Figure **??**. These results present the area under curve (AUC) of the Receiver Operating Characteristic (ROC) curves computed on the MIT1 and the EDH 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 5.1e, line 1 and Figure 5.1a) we achieve comparable performance with $|Z|$ ranging from 10 to 40; the performance decreases with a 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 nor-

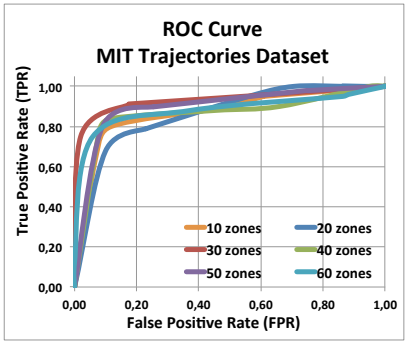
mal trajectories only slightly different from the ones included in the dataset. This consideration is also confirmed by the result obtained on the EDH (Table 5.1e, line 2 and Figure 5.1b), where the optimum number of zones ranges from 30 to 50.



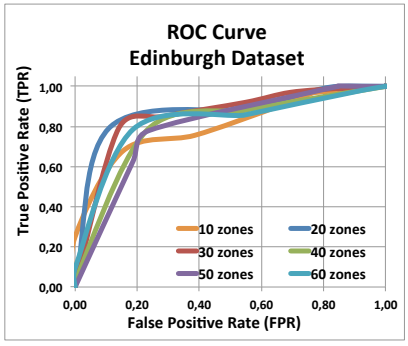

(a)




(b)




(c)




(d)

| | 10 zones | 20 zones | 30 zones | 40 zones | 50 zones | 60 zones |
|---|---|---|---|---|---|---|
| MIT1 Dirac | 0.87 | **0.88** | 0.85 | 0.87 | 0.74 | 0.67 |
| EDH Dirac | 0.72 | 0.73 | 0.84 | **0.87** | 0.86 | 0.77 |
| MIT1 Weighted Dirac | 0.87 | 0.85 | **0.92** | 0.86 | 0.89 | 0.86 |
| EDH Weighted Dirac | 0.79 | **0.85** | **0.85** | 0.82 | 0.79 | 0.80 |

(e)

**Figure 5.21** AUC of the MIT1 and of the EDH obtained by varying the $\alpha$ parameter and by using the Dirac Kernel and its weighted version.

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 5.15 that the trajectories in the MIT1 are located in a limited area of the image and are much more compact than in the EDH, where people can freely move in 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 5.1e: the improvement between the best and the worst case by using the Dirac Kernel is higher than 20%; for instance, for the MIT1, 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 5.22a: the trajectory in yellow 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

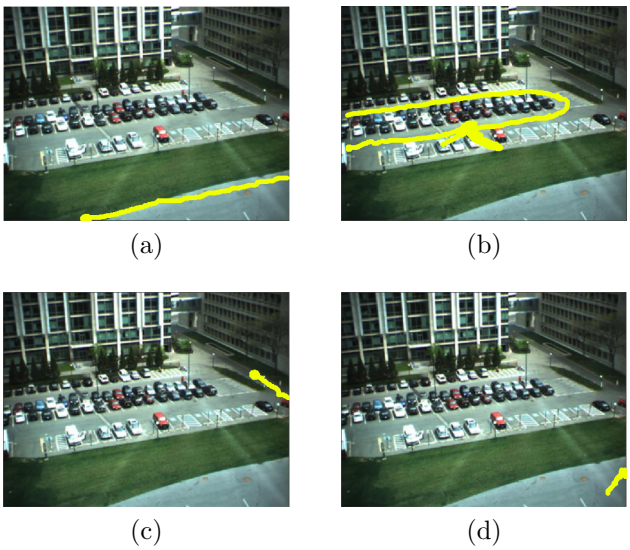


**Figure 5.22** Abnormal trajectories classified as normal (a)(b) and normal trajectories classified as abnormal (c)(d).

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 5.22b, 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

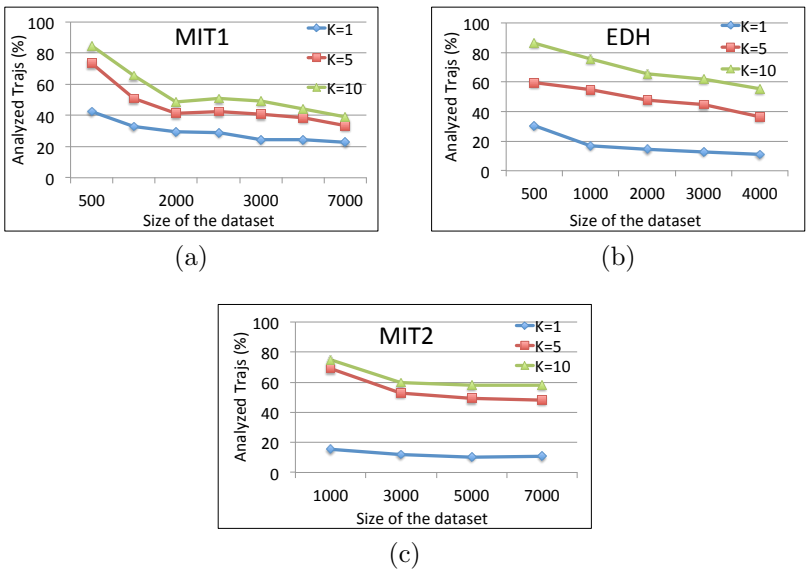



**Figure 5.23** Query by sketch: percentage of trajectories that need to be analyzed in the MIT1 (a), in the EDH (b) and in the MIT2 (c).

operator.

Opposite kinds of error occur in Figures 5.22c and 5.22d. 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.

### 5.2.6 Query by sketch: experimental results

Two different evaluations will be provided in this section, a quantitative and a qualitative one, in order to confirm the effectiveness of the method proposed for solving queries by skecth. Both these

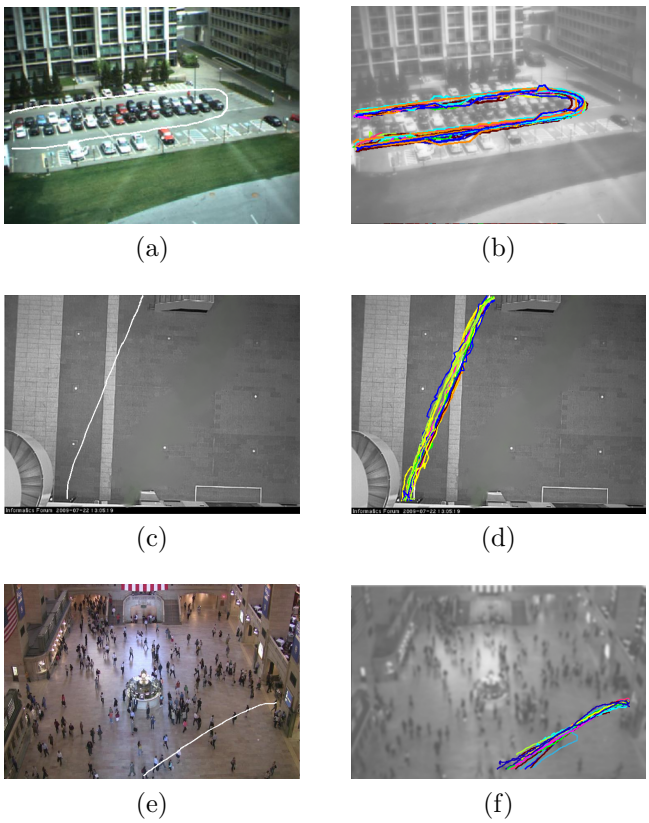


**Figure 5.24** The trajectories sketched by the user are shown in the first row, while the obtained results (with $k = 20$) are in the second row for the different datasets: MIT1 (a,d), EDH (b,e) and MIT2 (c,f).

evaluations have been performed over the three datasets described in Section 5.2.1.

The quantitative evaluation has been carried out by evaluating the improvement of the performance if compared to 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 5.23. 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, has been evaluated. The results are shown in Figure 5.23: 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 of 3000 trajectories, we only have to analyze 24% of it (approximatively 700 trajectories) in the MIT1 (Figure 5.23a) and 11% of it (approximatively 300 trajectories) in the EDH and the MIT2 (Figure 5.23b), 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 draws a trajectory and choses the number of trajectories (the $k$ value) he is interested in. Examples, one for each considered dataset, are provided in Figure 5.24. 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.

## 5.3 Indexing and Retrieval Engine

The storing and retrieving engine has been tested over both synthetic and real datasets. The former has been used to stress the proposed method with a very complex scenario, while the latter has been chosen in order to confirm the feasibility of our approach.

The database has been implemented by storing the trajectories' data in Postgres using PostGIS; data are indexed using the standard bi-dimensional R-tree over *Generalized Search Trees (GiST)* indexes since, as highlighted in the specialized literature, this choice guarantees higher performance in case of spatial queries if compared with the PostGIS implementation of R-trees.

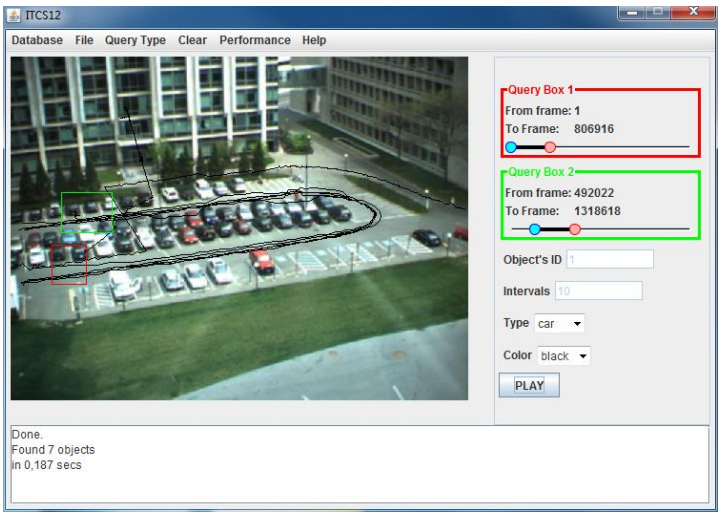



**Figure 5.25** A snapshot of the *GUI*.

In the next sections the obtained results will be presented: in particular, in Section 5.3.1 the graphical user interface designed for making easier the use of the human operators is presented; Section 5.3.2 details the experimental set up, while in Section 5.3.3 the results obtained by using the MIT Trajectory Dataset are shown. Finally, in Section 5.3.4 the synthetic dataset generator is detailed and in Section 5.3.5 a comparison with the off-the-shelf solution provided by PostGis over the synthetic dataset is performed.

### 5.3.1 Graphical User Interface

For an intuitive construction of the queries, the Graphical User Interface GUI depicted in Figure 5.25 has been realized: it allows the user to define both the geometric and temporal constraints needed for building a query box.

In particular, for a *DRSQ* query, the user specifies the temporal interval by using a sliding bar and the spatial region of interest by drawing a rectangle on the 2D plane. Furthermore, it is also possible to specify appearance-based information like the class of the object and its dominant color.

As for *Flow-DRSQ* queries, the user is also required to insert the number of intervals needed for the analysis. Finally, the definition of *M-DRSQ* queries ($M = 2$ in the example provided in Figure 5.25) requires to set parameters of two different query boxes: the user needs to draw two different rectangles, both representing the spatial area of the query boxes, and to specify two different time intervals, one for each query box.

In the example proposed in Figure 5.25, the user has built a *M-DRSQ* query box; in particular, he is interesting to see all the *black cars* which cross the *red* area, interactively drawn by the user, during the frames *1-806916* and after the *green* area during the frames *492022-1318618*. The seven trajectories satisfying the query are depicted on the GUI; the time needed to answer the query and to visualize the final result on the screen is 187 milliseconds. This result confirms that the proposed framework can surely satisfy in real time all the queries submitted by the human operator. More details about the computational cost of the proposed method are shown in the next Sections.

### 5.3.2 Experimental Set Up

As stated previously in this thesis, each query can be represented as a 3D cube; it is straightforward to observe that the time needed to process a generic *DRSQ* query (QT) is a function of the following parameters: the number of trajectories $T$, the trajectories' length $L$, the query cube dimension $D_c$ (expressed as percentage of the volume[1] $V = ||x|| * ||y|| * ||t||$), and on the position of the query box $P_c$. In particular, $P_c$ strongly influences the time needed to extract the trajectories as, in real world scenarios, the trajectories are not uniformly distributed. To avoid the dependence on the query cube position, we decided to repeat the query a number of times $N$ inversely proportional to the query cube dimension, as shown in Table 5.26; finally, results have been averaged to obtain:

$$\overline{QT}_{DRSQ} = f(T; L; D_c). \tag{5.6}$$

[1] $||x||$ and $||y||$ are the width and the height of our scene while $||t||$ be the whole time interval we are interested in.

| $D_c$ | 1% | 5% | 10% | 20% | 30% | 50% |
|---|---|---|---|---|---|---|
| $N$ | 200 | 40 | 20 | 10 | 7 | 4 |

**Figure 5.26** Query cube dimension $D_c$ with corresponding number of repetitions $N$.

It is evident that in real scenarios the dependency by the number of trajectories $T$ is imposed by the scenario itself. This is one of the main reasons why in the following two different experimentations will be carried out: the former, conducted over a real dataset, mainly aims at evaluating $QT$ by varying the query dimension $D_c$. On the other hand, the latter is devoted to stress the system by properly generating the trajectories to be stored.

### 5.3.3 Experiments over Real Dataset

The real dataset that we chose for our experimentation is the MIT Trajectory dataset (MIT1), already introduced in Section 5.2.1. The whole dataset, composed of approximately $4 * 10^4$ trajectories with 109 points in each trajectory (on average), has been used. At loading time, each trajectory has been segmented using $AreaMin = 1$, so obtaining approximately $1.92 * 10^6$ segments with 24 points in each segment (on average). We conducted our experiments on a PC equipped with an Intel quad core CPU running at 2.66 GHz, using the 64 bit version of the PostrgreSQL 9.2 server and the 2.1 release of PostGIS.

Table 5.27 shows $\overline{QT}_{DRSQ}$ (in seconds) as $D_c$ varies. It is worth pointing out that $\overline{QT}_{DRSQ}$ results from the sum of four terms: $T^1$ is the time needed to select the segments whose bounding box is completely inside the query box on each bi-dimensional plane (min selection), $T^2$ is the time needed to select the segments whose bounding box intersects the query box on each bi-dimensional plane (max selection), $T^3$ is the time to clip the segments (clipping) while $T^4$ is the time needed to extract the whole trajectory, so obtaining:

$$T^Q_{DRSQ} = T_1 + T_2 + T_3 + T_4. \tag{5.7}$$

| $D_c$ | $T^1 + T^2$ | $T^3$ | $T^4$ | $\overline{QT}_{DRSQ}$ |
|---|---|---|---|---|
| 1% | 0.003 | 0.010 | 0.009 | 0.022 |
| 5% | 0.007 | 0.064 | 0.115 | 0.186 |
| 10% | 0.013 | 0.154 | 0.320 | 0.487 |
| 20% | 0.038 | 0.533 | 1.383 | 1.954 |
| 30% | 0.097 | 1.566 | 4.014 | 5.673 |
| 50% | 0.173 | 5.878 | 14.924 | 20.975 |

**Figure 5.27** Averaged time (in seconds) to solve a *DRSQ* query.

As shown in Table 5.27, the system spends most of the time in clipping and extracting candidate trajectories ($T^3$ and $T^4$); this consideration further confirms the importance and the effectiveness of the proposed approach, which is able, as confirmed in the next section, to significantly reduce the number of trajectories to be analyzed and then to significantly improve the overall performance of the system during the retrieval phase.

Starting from the above considerations, it is possible in a simple way to obtain the expected performance of both *Flow-DRSQ* and *M-DRSQ* queries.

*F-DRSQ* is the application of several *DRSQ* queries in sequence. Suppose, for instance, that we ask to our system to retrieve the number of vehicles passing through Interstate 55 from 5 pm and 6 pm each ten minutes; what our system would do is to perform six *DRSQ* queries, one for each 10 minutes interval between 5 pm and 6 pm, only counting the number of instances satisfying the query in each interval and giving, as the final result, the total sum of the count. This means that, for each *DRSQ* query, the system performs only the intersection and the clipping stages, giving the total count as the final result of the *Flow-DRSQ* query.

According to the above considerations, the expected $\overline{QT}_{F-DRSQ}$ of a *Flow-DRSQ* query asking for the number of objects intersecting query box $B$ in the time interval $(t_1, t_n)$ each of the $N$ time

intervals is defined as:

$$\overline{QT}_{Flow-DRSQ} = N * (T_1 + T_2 + T_3), \tag{5.8}$$

in which $T_1$ is the time to verify if the trajectories 2D bounding boxes are completely inside the query boxes and $T_2$ and $T_3$ are the intersection and clipping times respectively for each of the $N$ *DRSQ* queries; finally, the sum of the $N$ count values is given as a result.

*M-DRSQ*s are slightly more complex as the multiple query boxes are virtually independent. In this case, the system processes each query box as a single *DRSQ* query, applying the intersection and clipping operations for each query box; finally, the extraction phase provides the trajectories satisfying the *M-DRSQ* query.

The expected $\overline{QT}_{M-DRSQ}$ of a *M-DRSQ* query with $M$ bounding boxes $(B_1, B_2, ..., B_M)$, each having its time interval, is:

$$\overline{QT}_{M-DRSQ} = M * (T_1 + T_2 + T_3) + T_4, \tag{5.9}$$

where we just need to extract once the trajectories' result set.

### 5.3.4 Synthetic Data Generator

In order to further stress the proposed approach, we also considered a synthetic data set, generated as follows. Let $||x||$ and $||y||$ be the width and the height of our scene and $||t||$ be the time interval we are interested in. Being in a video-surveillance context, we assume that the data are acquired for a week at a frame rate of 10 seconds per frames and the video has a resolution of 4-CIF. Each trajectory starting point is randomly chosen in our scene at a random time instant $t_1$; the initial directions along the $x$ axis and the $y$ axis, respectively $d_x$ and $d_y$, are randomly chosen. At each time step $t$, we first generate the new direction, assuming that $d_x$ and $d_y$ are updated once for second of $d$ degrees, being $d$ generated according to a uniform distribution in the interval $(-D, D)$; subsequently, we randomly chose the velocity along $x$ and $y$. The velocity is expressed in pixels/seconds and it is assumed to be generated in both the directions, $x$ and $y$, as a gaussian distribution

| | |
|---|---|
| $\|\|x\|\|$ (pixels) | 704 |
| $\|\|y\|\|$ (pixels) | 576 |
| $\|\|t\|\|$ (frames) | $10 * 3600 * 24 * 7$ |
| $T$ | $\in \{1, 2, 3, 5, 10\} * 10^5$ |
| $D$ (degrees) | 45 |
| $\mu$ (pixels/secs) | 10 |
| $\sigma$ (pixels/secs) | 1 |

**Figure 5.28** The parameters used to generate synthetic data.

$\mathcal{N}(\mu, \sigma)$. Therefore the new position of the object can be easily derived; if it does not belong to our scene, other points will not be generated for that trajectory. In this way the length $L$ of the trajectories is not fixed a-priori.

### 5.3.5 Comparison

At this point it should be evident that the system spends most of the time in the extraction of those trajectories candidate to be clipped as well as in the clipping itself. For this reason, in order to confirm the effectiveness of the proposed approach, we compare the proposed method with the solution recently provided by PostGis, which introduced a 3D indexing strategy, based on R-Trees, for verifying the intersection between bounding boxes. However, its main limitation lies in the fact that the function for verifying the containing relationship is not available in 3D dimensions but only in 2D dimensions, and then a decomposition in a 2D space is still needed in order to achieve very competitive results.

In particular, the comparison between the proposed method and the off-the-shelf solution provided by Postgis is carried out in terms of improvement of the number of trajectories to be clipped:

$$improvement = \frac{clipped_{Postgis} - clipped_{proposed}}{clipped_{Postgis}}, \qquad (5.10)$$

being $clipped_{Postgis}$ and $clipped_{proposed}$ the number of trajectory that the system need to extract and after to clip, respectively for the solution provided by Postgis and for the proposed method.

The obtained results, for different segmentation conditions, are shown in Figure 5.29. In particular, in the generic figure the *improvement* (on the y-axis) is depicted by increasing both the number of trajectories $T$ (on the x-axis) and the query box dimension $D_c$ (by overlapping on the same figure different lines).

First, we can note that the segmentation step strongly influences the performance of the proposed algorithm. As a matter of fact, the improvement shown in Figure 5.29(g), obtained by considering $AreaMin = 100\%$ (without segmentation) are only slightly better for very big query boxes. For most of the $D_c$ values, in fact, the improvement is null. This is mainly due to the fact that a very long trajectory could potentially cover the entire spatial extension, so implying that the introduction of the min-selection step does not make sense, since the containing relationship is verified only for a few trajectories.

The improvement of the performance of the proposed method is much more evident by decreasing the Area Min value. As shown in Figure 5.29(a), the best results are obtained by considering a very small Area Min value ($AreaMin = 1\%$) as well as very high query boxes dimensions: in this case, in fact, approximatively 80% of trajectories does not need to be clipped. It is mainly due to the following considerations: if the length of the segments is small enough, the probability that it is completely contained inside the query box is very high; of course, this consideration is much more true for bigger query boxes.

## 5.4 Audio Recognition

In this section we will analyze the performance obtained by the method introduced in this thesis for detecting audio events of interest. In particular, Section 5.4.1 details the considered dataset, in Section 5.4.2 the obtained results will be reported while in Section 5.4.3 a comparison with a state of the art method is performed.

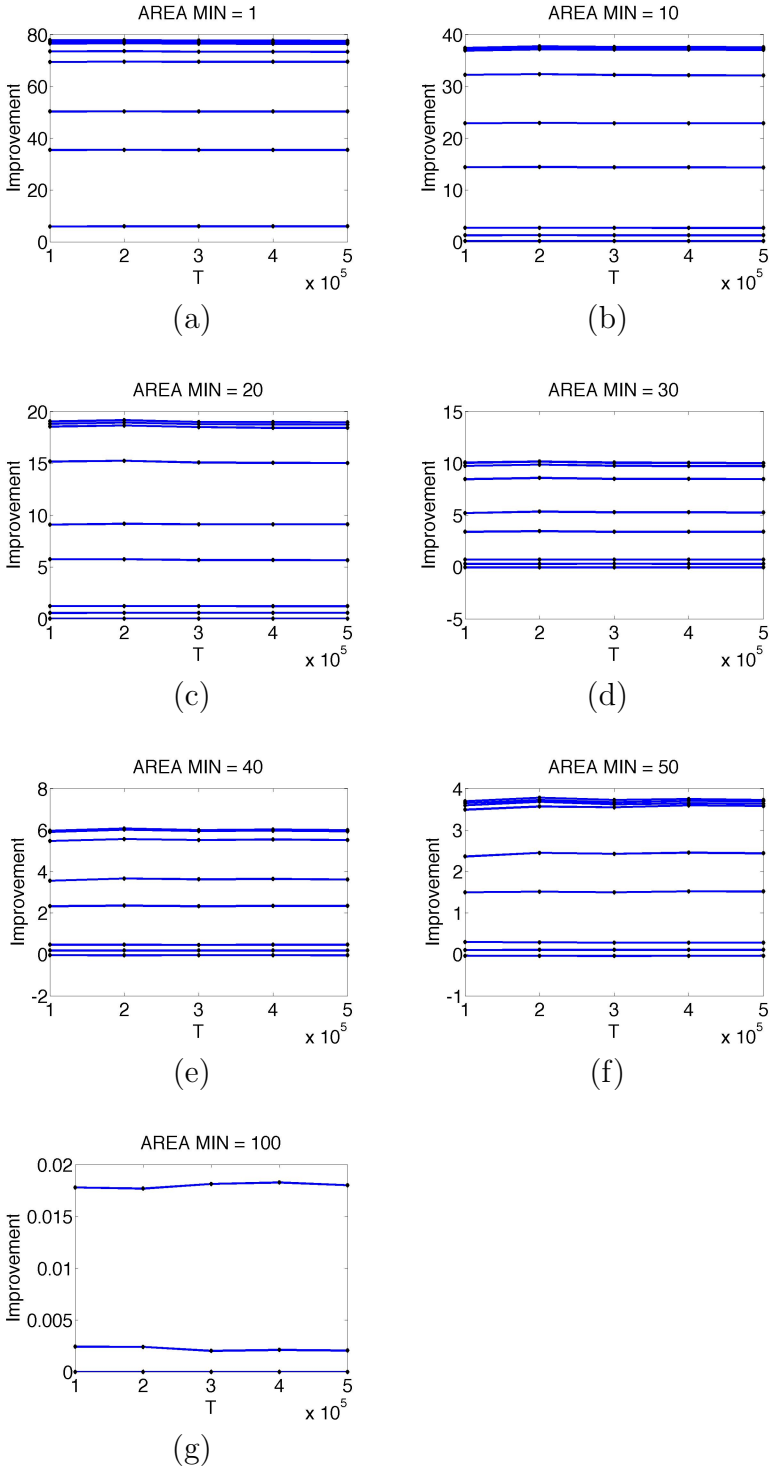



**Figure 5.29** Improvement of the proposed method if compared with the available Postgis solution by varying the segmentation conditions (Area Min) and the query cube dimension $D_c$.

### 5.4.1 The dataset

The experimental validation of the system has been carried out considering a typical audio surveillance application that requires the recognition of the following three classes of abnormal audio events: *scream*, *broken glass* and *gunshot*. From the user perspective, the system must raise an alarm in presence of an abnormal audio event, while the sounds produced by any other source should be recognized as belonging to the *background noise* class.

To the best of our knowledge, there are no publicly available datasets for the benchmarking of audio surveillance applications. Thus, we constructed our own dataset of PCM audio clips sampled at 32 KHz frequency and with a resolution of 16 bits per sample. We collected a total of 650 audio clips, some of them recorded by us in different environment conditions and others selected from compilations of sounds usually used for movie effects. In particular, we collected 271 different sounds that belong to the three classes of abnormal sounds defined above.

Furthermore, in order to test the robustness of the proposed approach with respect to the presence of different kinds of background sounds both from indoor and outdoor environments, we collected and included into the dataset 379 audio clips representative of the following types of sound: silence and white gaussian noise, rain, whistles, crowded ambiance, vehicles, household appliances, bells, applauses and claps. All the audio clips have been recorded with an Axis P8221 Audio Module and an Axis T83 omnidirectional microphone for audio surveillance applications.

A key requisite for an audio surveillance system is the ability to detect that an abnormal event (hereinafter foreground sound) occurs even when it is mixed with one or many kinds of background sounds and when its energy is only slightly higher or even comparable to the one of the background noise. That means that a surveillance system must be able to detect abnormal events also for low values of the signal to noise ratio (SNR). Thus, in order to account for these necessities, we defined and adopted a procedure to create a new dataset of audio clips suitable for abnormal events

detection and classification purposes.

The audio clips from the original dataset have been normalized so that they have all the same overall energy, as follows:

$$\overline{x}(n) = \frac{x(n)}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} x(i)^2}}, \tag{5.11}$$

Before to start generating the dataset, the normalized audio clips were split in two groups comprising 70% and 30% of the total amount of sounds from the original dataset, respectively. We used the clips from the first group to build the training set and the ones from the second group to build the test set. Thus, we generated two sets of audio clips of about 3 minutes duration, by mixing foreground sounds with different combinations of a background sounds. The procedure adopted to create a new audio clip is the same for both the training set and the test set and is explained in detail in the following.

First, an environmental sound is created by mixing a number $d \in \{1, 2, 3\}$ of sounds from the background class. The number $d$ is randomly defined before the creation of every clip. Since the original background sound files have different durations, we replicated them in order to fit the duration of 3 minutes of the new audio clip to be created. Let us define as $B_j(n) = \sum_{k=1}^{d} b_k(n)$ the background noise, where $b_k(n), k = 1, \ldots, d$ are the replicated background sounds.

Once the environmental sound is created, a number $N_e$ of foreground events is randomly chosen from the original dataset and mixed with the environmental sound, in order to simulate the occurrence of an abnormal event in a real and complex environment. The foreground events are distanced each other by $\Delta t_n$ seconds, whose value is uniformly distributed in the interval $[3.5, 5]$. A certain event from the original dataset occurs a number of times in the final dataset, but with different background noise in order to simulate the presence of a specific sound in different environment configurations. It is worth noting that in a real environment the source of a target event can be at different distances from the acquisition

equipment resulting in signals with various values of the SNR. Thus, when a foreground sound is mixed with the environmental sound, the energy of the foreground sound is amplified or attenuated in order to guarantee a specific value $SNR^p, p = 1, \ldots, 6$ of the SNR from the set $\zeta = \{5dB, 10dB, 15dB, 20dB, 25dB, 30dB\}$ for the target sound. The rule for the construction of the audio clip considering foreground events at a certain SNR value is defined as follow:

$$y_j^p(n) = \sum_{i=1}^{N_e} \left\{ B_j(n) \oplus_{[s_i,e_i]} A_p \overline{x}_i(n) \right\}, \tag{5.12}$$

where

$$A_p = 10^{SNR^p/20} \frac{rms(B_j(n))}{rms(\overline{x}_i(n))}. \tag{5.13}$$

The amplification (or attenuation) coefficient $A_p$ depends on the specific value $SNR^p$ and on the root mean square values (rms) of the environmental sound and of the foreground sound. With $\oplus_{[s_i,e_i]}$ we define an operator that allows to mix the signal $A_p\overline{x}_i(n)$ with the signal $B_j(n)$ in the interval delimited by $[s_i, e_i]$ ($s_i$ and $e_i$ are the starting and ending points of the target sounds within the clip $y_j^p(n)$, respectively).

The final dataset consists of a training set and a test set that contain, respectively, 396 and 184 audio clips of about 3 minutes. The total duration of the sounds in the dataset is about 20 hours for the training set and about 9 hours for the test set. Given a specific sequence $S = e_1, e_2, \ldots, e_m$ of foreground events, six versions of the audio clip are created. Such versions differ each other for the value $SNR^p$ of the desired SNR at which the foreground events occur.

In the following we will refer to the different classes with the abbreviations BG for broken glass, GS for gunshot, S for scream and BN for background noise. The training set of the experimental dataset is composed by 700 events for each target class. Each event is present at six different SNR values, for a total of 4200 events for each class. In the same way, in the training set 300 events per class are provided in six versions, for a total of 1800 events for

| | Training set | | Test set | |
|---|---|---|---|---|
| | #Events | Duration (s) | #Events | Duration (s) |
| **Background** | - | 58371.6 | - | 25036.8 |
| **Broken glass** | 4200 | 6024.8 | 1800 | 2561.7 |
| **Gunshot** | 4200 | 1883.6 | 1800 | 743.5 |
| **Scream** | 4200 | 5488.8 | 1800 | 2445.4 |

**Figure 5.30** Summary of the composition of the final dataset. The training set and the test set contain respectively the 70% and the 30% of the total number of events from the classes of interest. For the background noise class the total duration of the background sounds is reported.

each foreground class. In Table 5.30 we report a summary of the composition of the dataset.

### 5.4.2 Performance evaluation

For a surveillance application, it is important to correctly recognize an abnormal event and consequently fire an alarm, but it is also relevant to not detect abnormal events when only background sounds are present in the environment. Thus, we evaluated the performance of our system from two different perspectives. On one hand, we considered the recognition rate of the events of interest and, on the other hand, the evaluation of false positives (FP), i.e the detection of abnormal events when only ambiance noise is present.

The average rate of correct classification achieved by the system for the foreground events on the whole test set is 84.8%. An event is correctly detected if it is detected in at least one of the time windows of analysis that overlap with the considered event. The classification matrix reported in Table 5.31 shows that the misclassification errors between the three classes of foreground sounds (BG, GS, S) are mainly directed to the background noise class, consequently being considered missed detections. In the fourth

**BoAW classifier - Classification matrix**

| | | **Guessed class** | | | |
|---|---|---|---|---|---|
| | | **BG** | **GS** | **S** | **Miss** |
| **True class** | **BG** | 93.6% | 0.2% | 0.2% | 6% |
| | **GS** | 3.3% | 81.6% | 0.5% | 14.6% |
| | **S** | 2.8% | 0.9% | 79.3% | 17% |

**Figure 5.31** Results achieved using the proposed system on the whole test set. The entry at the row $i$ and column $j$ represents the fraction of samples belonging to the $i$-th class and attributed by the system to the $j$-th class. The Miss column reports the ratio of the foreground sounds of each class that are classified as background noise.

column of the classification matrix the rate of missed detection is reported for the events of each class. The misclassification error between the foreground classes is very low. It means that the abnormal events are well detected and separated from each other by the proposed classification system.

As said above, for a surveillance application it is important to not detect abnormal events when normal ambiance sounds are present in the environment, i.e. to reduce the false alarm rate. For our analysis, we consider that a false positive hit is counted when an abnormal event is detected in a time window where only background noise is present. If for two consecutive time windows a foreground event is detected, we consider only one false positive occurrence. Thus, the false positive rate is computed as the ratio of the detected false positive events on the total number of intervals between two foreground sounds, since those intervals contain only background noise. For the whole test set we achieved a false positive rate equal to 2.1%, divided into 0.83% false detected broken glasses, 0.74% gunshots and 0.53% screams.

In Table 5.33 we report, separately, the performance results that we achieved by testing the proposed method on audio clips with different values of the SNR for the foreground events. As expected, when the sounds have higher values of SNR there is a

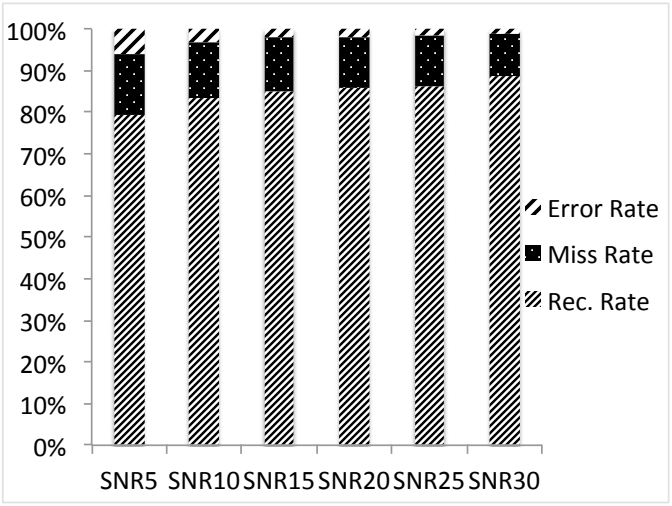



**Figure 5.32** Classification of the foreground events for different values of the SNR. Each bar represents how the performance results are distributed between correct detections, misses and errors for a certain value of the SNR.

| | **Rec. Rate** | **Miss Rate** | **Error Rate** | **FP Rate** |
|---|---|---|---|---|
| SNR $5dB$ | 79.6% | 14.2% | 6.2% | 6.2% |
| SNR $10dB$ | 83.4% | 13.4% | 3.1% | 1.6% |
| SNR $15dB$ | 85.2% | 12.8% | 2% | 1.3% |
| SNR $20dB$ | 85.9% | 12.2% | 1.9% | 1.2% |
| SNR $25dB$ | 86.1% | 12.3% | 1.6% | 1.2% |
| SNR $30dB$ | 88.6% | 10.2% | 1.2% | 1% |
| **Average** | 84.8% | 12.5% | 2.7% | 2.1% |

**Figure 5.33** Detailed results achieved by using the proposed bag of words classifier for different values of the SNR of the foreground sounds.

reduction in both the miss and error rates, resulting in a better recognition rate. In Figure 5.32 a cumulative graph of the results achieved at different SNR values is depicted. For every value of

the SNR, it is shown how the classification results are distributed between correct detections, misses and errors.

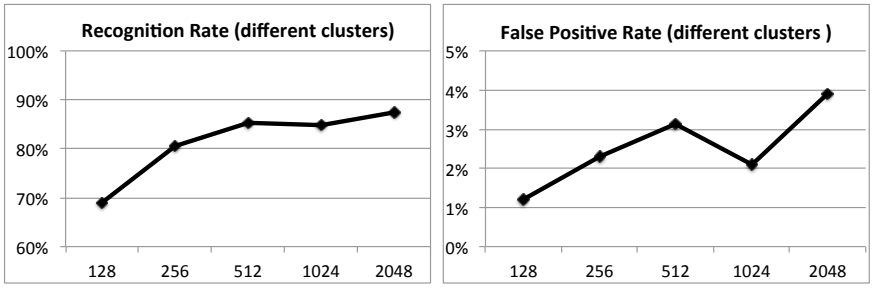
We also observed an improvement in the detection of false positives, which decrease for higher values of the SNR due to the reduced influence of the background noise on the target sound. It is worth noting that the recognition rate for events at $5dB$ SNR is only 5% lower than the average recognition rate on the whole dataset and about 9% lower than the best value achieved for the events at $30dB$ SNR. The correct classification rate and the false positive rate that we achieved in different SNR conditions prove that the proposed system is robust to background noise variation and can be used for surveillance applications also in highly noisy environments.

As said in Section 4.2, the value $K = 1024$ has been chosen by evaluating the performance results achieved for different numbers of cluster centroids. In Figure 5.34 we show the curves of the recognition rate and of the false positive rate for increasing values of $K$. We observed that for values greater than 1024, although the recognition rate is slightly higher, the false positive rate is almost double than the one reported for $K = 1024$. Since the false positive rate is an important metric for the evaluation of an intelligent surveillance system, we considered for our experiments the value $K = 1024$.

### 5.4.3 Performance comparison

In our experiments, we compared the performance results of the proposed system with the ones achieved by the method that is described in [96] and that combines a reject option with a LVQ classifier to classify 32 milliseconds audio frames.

The LVQ classifier has been trained using the same set of first-level features. During the test phase, the audio frames are classified into one of the four considered classes, i.e. the three foreground sound classes (BG, GS, S) and the background noise class (BN). The decisions at frame level are then aggregated over intervals of 3 seconds. An interval is constituted of 375 frames like




(a) (b)

**Figure 5.34** Recognition rate (a) and false positive rate (b) achieved by training the system for different values of $K$. The value $K = 1024$ has been observed to be an optimal number of clusters evaluating both the recognition and false positive rates.

the intervals used for the bag of aural words classifier. The system attributes an interval to the class $C_i$ that obtains the highest score $z_i = (n_i - \hat{n}_i)/\hat{n}_i$, where $n_i$ is the number of frames in the interval assigned to the class $C_i$; $\hat{n}_i$ is a threshold that indicates a limit under which the $i$-th class is not considered as a candidate for the final decision. In case of $z_i < 0$ for $\forall i = 1, ..., M$, that is a negative score for all the classes, the interval is classified as belonging to the background class $C_0$.

The classification performance of the two systems has been compared by using the receiver operating characteristic (ROC) curves. The closer a ROC curve approaches the top-left corner the better the performance of the algorithm is. In Figure 5.35 we plot the ROC curves for each of the three classes of foreground events and in Table 5.37 we report the values of the area under the ROC curves (AUC). We consider the AUC, which is equal to 1 for a perfect classification, as a measure of the overall performance of the two methods. We observe that the proposed method (solid line) generally outperforms [96] (dashed line). It is evident that the proposed system is able to reduce the false positive rate, mostly for the broken glass and gunshot classes, with respect to the performance of [96].

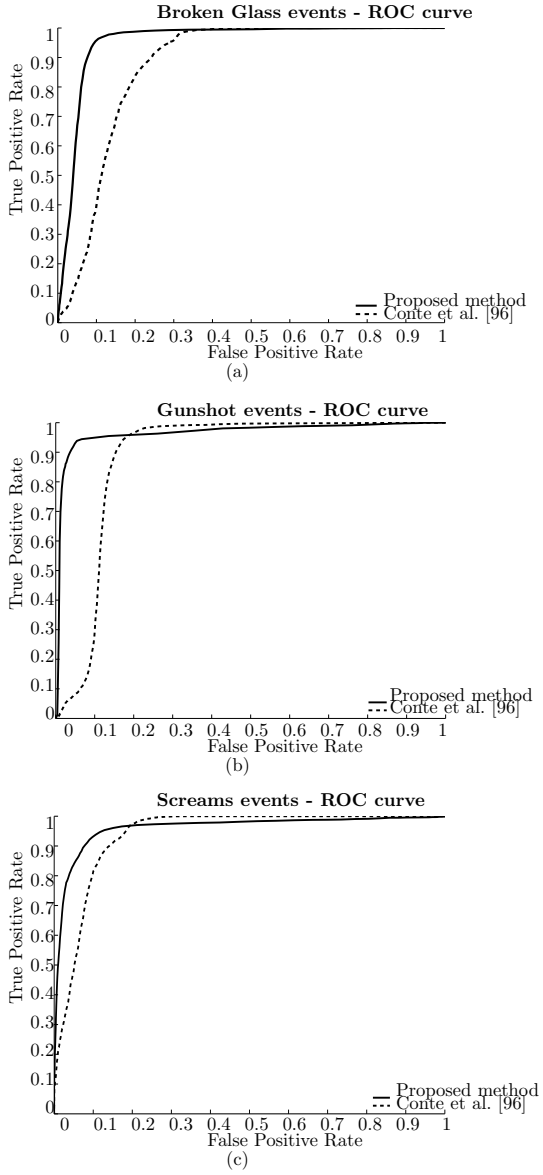



**Figure 5.35** Comparison of the performance of the proposed method with respect to the method of Conte et al. [96] in terms of ROC curves for the broken glass (a), gunshot (b) and scream (c) events. Note that the proposed method (solid line) clearly outperforms [96] (dashed line).

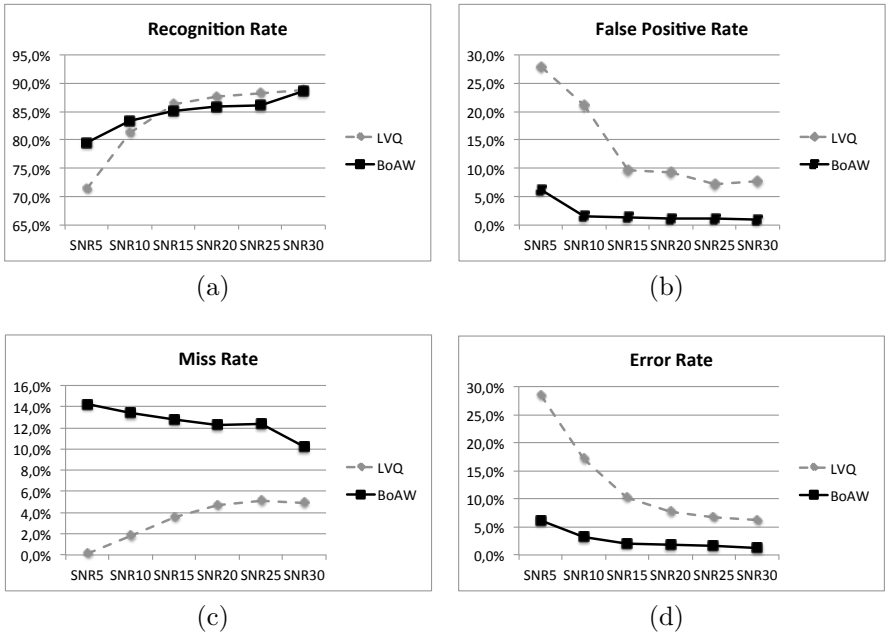



**Figure 5.36** Comparison of the performance of the proposed bag of aural words classifier (solid line) with respect to the classifier proposed in [96] (dashed line). The recognition rate (a), false positive rate (b), miss rate (c) and error rate (d) are reported for different values of the SNR of the foreground sounds.

In order to compare the performance of the two systems in operating conditions, we determined the value of the threshold $\tau = 0$ for the proposed *bag of aural words* classifier and the values of $\hat{n}_{BN} = 266$, $\hat{n}_{BG} = 95$, $\hat{n}_{GS} = 26$, $\hat{n}_S = 62$ for [96], by performing a grid search on the training set. In Table 5.38, we report the performance results by the method in [96] for the classification of the foreground events class. The average recognition rate on the whole test set is 83.9%, that is slightly lower than the one (84.8%) that we achieved by using the proposed bag of aural words classifier. As shown in Figure 5.36a, the proposed system is more accurate than the one in [96] for the analysis of highly noisy environments, i.e. when the events of interest occurr with a lower value of the SNR. For events with higher SNR, instead, the recog-

nition rate achieved by [96] on our dataset is slightly better than the one of the proposed system. The classification error between the three foreground sounds classes achieved by the bag of aural words system is remarkably lower than the one achieved by [96]. The proposed system, thus, provides a generally better recognition of the events of the classes of interest, reaching good performance results also in environments in which the target sounds occur with a low value of the SNR.

**AUC Comparison**

| | **Proposed Method** | **Conte et al. [96]** |
|---|---|---|
| **BG** | **0.954** | 0.872 |
| **G** | **0.966** | 0.886 |
| **S** | **0.961** | 0.938 |

**Figure 5.37** Comparison of the performance of the proposed method with respect to Conte et al. [96] in terms of the area under the ROC curves for the foreground classes.

**Conte et al. [96] - Classification matrix**

| | | **Guessed class** | | | |
|---|---|---|---|---|---|
| | | **BG** | **GS** | **S** | **Miss** |
| **True class** | **BG** | 91.3% | 5.3% | 1.4% | 1.9% |
| | **GS** | 12.1% | 80.6% | 3.9% | 3.4% |
| | **S** | 7.6% | 7.9% | 79.8% | 4.7% |

**Figure 5.38** Classification matrix achieved by Conte et al. [96]. The entry at the row $i$ and column $j$ represents the fraction of samples belonging to the $i$-th class and attributed by the system to the $j$-th class. The Miss columns indicates the foreground sounds that are classified as background noise.

In Table 5.39 we report the false positive rates achieved by the proposed system and by [96] on the whole test set. It is worth

**FPR Comparison**

| | **Proposed Method** | **Conte et al. [96]** |
|---|---|---|
| SNR5 | 6.2% | 27.9% |
| SNR10 | 1.6% | 21.1% |
| SNR15 | 1.3% | 9.7% |
| SNR20 | 1.2% | 9.3% |
| SNR25 | 1.2% | 7.2% |
| SNR30 | 1% | 7.6% |
| **All SNR** | 2.1% | 13.8% |

**Figure 5.39** Comparison of the false positive rates for the BoAW system and the LVQ system achieved at different levels of the SNR value. The performance increases with higher level of SNR for both the classifiers.

pointing out that, using the LVQ classifier proposed in [96], the number of false positives hugely increases with respect to the number of false positives detected by the proposed bag of aural words classifier. The overall false positive rate achieved using the method proposed in [96] is more that 10% higher than the one achieved by the proposed system. A high number of false positive occurrences is an unacceptable weakness for a real application, because it means that the system has a poor robustness to the environmental noise.

From the comparison of the recognition rates (Figure 5.36a) and false positive rates (Figure 5.36b) at different SNR values of the foreground sounds we can affirm that the proposed bag of aural words approach for audio surveillance applications proves to have a higher robustness to the environmental noise and generally a better performance than the method proposed in [96]. Although the miss rate is higher than the one achieved by [96] (Figure 5.36c), the error rate reported for the proposed system (Figure 5.36d) shows that the recognition of the events of interest is better performed by the bag of aural words classifier described in this work.

**Results Comparison**

| | **Proposed method** | **Conte et al. [96]** |
|---|---|---|
| **Recognition rate** | 84.8% | 83.9% |
| **False Positive Rate** | 2.1% | 13.8% |
| **Miss Rate** | 12.5% | 3.4% |
| **Error Rate** | 2.7% | 12.7% |

**Figure 5.40** Summary of the performance results achieved by the bag of aural words classifier on the proposed dataset in comparison to the results achieved by the method proposed in [96].

In Table 5.40 we report the summary of the comparison of the performance results of the two methods.

The representation of time intervals using the histogram of the occurrences of the aural words leads to a more fruitful and robust analysis of the audio stream. The effects of the background noise on the detection of abnormal events is limited and the performance are generally better than a method of analysis based on the classification of shorter audio frames (in [96] the classification is performed for short audio frames of 32 milliseconds duration). The proposed bag of aural words classification scheme is particularly useful in audio surveillance applications due to its robustness to the environmental noise and the consequently lower false alarm rate. With the adoption of the aural words and the second level feature vectors, the representation of the audio stream to be analyzed takes into account contextual information about the environmental sounds and leads to a more accurate and reliable classification results. Instead, when a decision is taken at a lower level, like in the case of [96], the contextual information is not considered and the effect of the noise causes a considerable decrease of the performance.

## 5.5 Achievements

In this section the results obtained by the proposed system will be briefly summarized.

- The tracking algorithm has been tested over two standard datasets, namely PETS and ISSIA Soccer datasets, and the obtained results (in terms of MOTA, MOTP and ATA) confirm the effectiveness of the proposed approach. Furthermore, the proposed approach participated to a competition (PETS 2013 contest), where it has been selected by the organizers as the most robust one in terms of accuracy (MOTA). As for the computational cost, the algorithm is able to work in real time, with a frame rate of 35 fps by considering 4 CIF images, so confirming the real applicability of the proposed approach.

- The module for visual behavior analysis has been tested over three standard datasets, namely MIT1, EDH and MIT2, and the results have been compared with state of the art approaches: the proposed clustering algorithm reveals to be very promising both in terms of c-index (0.11 vs 0.23, on average) and time required for the computation (0.48 vs 57 hours, on average). Furthermore, the high accuracy of the proposed approach is confirmed by the module for anomaly detection, which obtains an average AUC of 0.88 by considering 30 zones. Finally, a significant improvement has been also obtained by the k-NN search based on KD-Tree for solving queries by sketch: less than 20% (in the worst case) over the whole dataset needs to be analyzed for $k = 1$, while less than 50% (in the worst case) for $k = 5$.

- The storing and retrieving engine has been tested over both synthetic and real dataset (MIT1 dataset). The obtained results have been compared with the standard PostGis solutions, showing a significant improvement, up to 70 times by considering a small query box size and a small area min value.

- In order to test the algorithm for recognizing audio events of interest, a new dataset, composed approximatively by 6000 events of interest, has been introduced and made available for benchmarking purposes. The performance results that we achieved (Rec. Rate = 84.8%, False Positive Rate = 2.1%) have been compared with the results of another method from the literature (Rec. Rate = 83.9%, False Positive Rate = 13.8%), and show the robustness of the proposed approach with respect to noise and its applicability to real environments.

# Chapter 6

# Conclusions

In this thesis we proposed a system for supporting the human operator in his boring task of monitoring several areas of interest in crowded environments. The system is able to interpret the behavior of different typologies of objects by analyzing both video and audio data.

The analysis of audio information is performed by means of a novel approach based on bag of aural words, whose main advantages lie in its ability to automatically adapt to both short, impulsive sounds (like gun shots) and long, sustained ones (like screams), as well as to work in noisy environments where the sounds of interest can occur at different signal to noise ratios. The events recognized by the proposed system through audio inspection are screams, gun shots and broken glasses.

On the other hand, behavior analysis using visual information is based on the assumption that the movement of the objects inside a scene is not random, but instead is determined by their behaviors. It implies that analyzing moving objects trajectories would allow to analyze moving objects behaviors.

In order to achieve this aim, a novel tracking algorithm has been defined for extracting moving objects trajectories. The proposed approach is able to exploit the history of each object by means of a Finite State Automaton, so significantly increasing the performance of the proposed method; the update of information

related to each object is performed by a graph-based approach, while occlusions are properly managed by tracking into a different way single objects and groups of objects.

The events of interest are then identified according to a set of prototypes previously acquired into an unsupervised way during a learning step; it implies that no knowledge about the particular environment is required during the system setup. As soon as an abnormal behavior occurs, an alert is sent to the human operator in order to allow a prompt intervention for managing the particular situations. This step is achieved by defining a proper representation of the trajectories, based on strings, which takes into account position, speed and shape. The similarity between trajectories is evaluated by a novel string kernel, while the extraction of prototypes is made possible by a novel kernel-based clustering algorithm.

Furthermore, in order to allow the human operator to easily and efficiently retrieve events of interest if necessary, all the obtained data are properly stored. In particular, the most complex information to be managed pertains the trajectories, for their spatio-temporal nature. For this reason, a novel indexing schema has been defined in this thesis, by taking advantage on bi - dimensional off-the-shelf solutions.

All the above mentioned modules, respectively devoted to tracking, behavior analysis by visual inspection, audio event analysis and storing, have been tested over standard datasets and a proper comparison with state of the art approaches has been carried out.

In particular, the tracking algorithm participated to an international contest, namely the PETS competition: it ranked at the the first places in all the considered metrics and has been selected by the organizers as the best one in terms of accuracy. As for the module for visual behavior analysis, it has been tested over three standard datasets and the obtained results (c-index = 0.11 and execution time = 0.48 hours in average) have been compared with several state of the art approaches (c-index = 0.23 and execution time = 57 hours in average for the best one), confirming its effectiveness. Promising performance have been also obtained by

the module for audio event recognition: a new dataset has been made available for benchmarking purposes and the obtained results (Rec. Rate = 84.8%, False Positive Rate = 2.1%), compared with a state of the art approach (Rec. Rate = 83.9%, False Positive Rate = 13.8%), confirm its strong robustness. Finally, the module for storing and retrieving trajectories has been evaluated over both standard and synthetic datasets and the obtained results have been compared with the standard solution provided by PostGis. Even in this case, very encouraging results have been obtained, strongly outperforming (up to 70 times) PostGis solution.

In conclusion, the results obtained over all the proposed algorithms confirm the efficiency and the effectiveness of the proposed approaches, as well as their applicability in real environments. Furthermore, the high level of interaction with the human operator made possible by the proposed system strongly encourages its use in real application for supporting surveillance tasks.

However, it is evident that an advanced combination of audio and video information, obtained through the introduction of a fusion engine, may improve the overall reliability of the proposed framework by properly combining events respectively obtained by video and audio inspection. Furthermore, the introduction of a sound-based localization algorithm would allow to detect the exact position of the events of interest: in this way, the camera may zoom on that position in order to better investigate by visual inspection on the type of the event occurring. Of course, the introduction of similar algorithms risks to make the computational cost too heavy and then not yet feasible in real applications.

In this thesis I dealt with very challenging problems, and now I'm not afraid to face the above mentioned issues during my future works!

# Appendix A

# Proofs

This appendix is devoted to prove Equations 3.34 and 3.37, respectively devoted to the squared error and to the cutting position computations.

**Proof A: Squared Error Computation (Equation 3.34)**

Let be:

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

$$\delta_s^i = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

$$b_s = -\frac{1}{|C|} 1_C + \delta_s. \tag{A.3}$$

$$\mu = \frac{1}{|C|} \sum_{i \in C} \Psi_i \tag{A.4}$$

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

**Proof B: Cutting Position Computation (Equation 3.37)**
Let us additionally consider $C_t \subset C$ and:

$$\mu_t = \frac{1}{|C_t|}\sum_{i\in C_t}\Psi_i = \frac{1}{|C_t|}\sum_{i\in C} 1_{C_t}^i \Psi_i. \tag{A.5}$$

where the upper script $i$ in $1_{C_t}^i$ denotes the $i$th coordinate of $1_{C_t}$.

Let us additionally consider:

$$a^i = \frac{1^i_C}{|C|} - \frac{1^i_{C_t}}{|C_t|}; \tag{A.6}$$

$$a = \frac{1_C}{|C|} - \frac{1_{C_t}}{|C_t|}. \tag{A.7}$$

The term $||\mu - \mu_t||^2$ can be computed as follows:

$$\begin{aligned}
||\mu - \mu_t||^2 &= ||\sum_{i \in C} \frac{1^i_C \Psi_i}{|C|} - \frac{1^i_{C_t} \Psi_i}{|C_t|}||^2 \\
&= ||\sum_{i \in C} \left( \frac{1^i_C}{|C|} - \frac{1^i_{C_t}}{|C_t|} \right) \Psi_i||^2 \\
&= ||\sum_{i \in C} a^i \Psi_i||^2 \\
&= \sum_{i,j \in T} a^i a^j < \Psi_i, \Psi_j > \\
&= \sum_{i,j \in T} a^i a^j \cdot k(s_i, s_j) \\
&= a^t K a \\
&= \left( \frac{1_C}{|C|} - \frac{1_{C_t}}{|C_t|} \right)^t K \left( \frac{1_C}{|C|} - \frac{1_{C_t}}{|C_t|} \right) \\
&= \frac{1}{|C|^2} 1^t_C K 1_C - \frac{2}{|C_t||C|} 1^t_C K 1_{C_t} + \frac{1}{|C_t|^2} 1^t_{C_t} K 1_{C_t}.
\end{aligned}$$

Furthermore, starting from $||\mu - \mu_t||^2$, $||\mu - \mu_{t+1}||^2$ can be quickly computed. In particular, using the relationship $1_{C_{t+1}} = 1_{C_t} + \delta_p$ where $C_{t+1} = C_t \cup \{p\}$ and $\delta^i_p = 1$ for $i = p$ and 0 otherwize, the second term $1^t_C K 1_{C_{t+1}}$ is equal to:

$$\begin{aligned}
1^t_C K 1_{C_{t+1}} &= 1^t_C K (1_{C_t} + \delta_p) \\
&= 1^t_C K 1_{C_t} + 1^t_C K \delta_p \\
&= 1^t_C K 1_{C_t} + \sum_{i \in C} k(i, p).
\end{aligned} \tag{A.8}$$

Finally, the last term $1_{C_{t+1}}^t K 1_{C_{t+1}}$ may be decomposed as follows:

$$\begin{aligned} 1_{C_{t+1}}^t K 1_{C_{t+1}} &= \left(1_{C_t} + \delta_p\right)^t K \left(1_{C_t} + \delta_p\right) \\ &= 1_{C_t}^t K 1_{C_t} + 2 1_{C_t}^t K \delta_p + \delta_p^t K \delta_p \\ &= 1_{C_t}^t K 1_{C_t} + 2 \sum_{i \in C_t} k(i,p) + k(p,p). \end{aligned} \tag{A.9}$$

# Bibliography


[1] T. Ainsworth, "Buyer beware," *Security Oz*, vol. 19, pp. 18–26, 2002.

[2] S. Shivappa, M. Trivedi, and B. Rao, "Audiovisual information fusion in human computer interfaces and intelligent environments: A survey," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1692–1715, 2010.

[3] P. Borges, N. Conci, and A. Cavallaro, "Video-based human behavior understanding: A survey," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, no. 11, pp. 1993–2008, 2013.

[4] D. Simonnet, S. Velastin, E. Turkbeyler, and J. Orwell, "Backgroundless detection of pedestrians in cluttered conditions based on monocular images: a review," *Computer Vision, IET*, vol. 6, no. 6, pp. 540–550, 2012.

[5] N. Buch, S. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 3, pp. 920–939, 2011.

[6] S. Sivaraman and M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *Intelligent Transportation Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–23, 2013.

[7] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: a review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 5, pp. 694–711, 2006.

[8] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, pp. –, Dec. 2006. [Online]. Available: http://doi.acm.org/10.1145/1177352.1177355

[9] J. Candamo, M. Shreve, D. Goldgof, D. Sapper, and R. Kasturi, "Understanding transit scenes: A survey on human behavior-recognition algorithms," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, no. 1, pp. 206–224, 2010.

[10] H. M. Dee and S. A. Velastin, "How close are we to solving the problem of automated visual surveillance?: A review of real-world surveillance, scientific progress and evaluative mechanisms," *Mach. Vision Appl.*, vol. 19, no. 5-6, pp. 329–343, Sep. 2008. [Online]. Available: http://dx.doi.org/10.1007/s00138-007-0077-z

[11] D. Gowsikhaa, S. Abirami, and R. Baskaran, "Automated human behavior analysis from surveillance videos: a survey," *Artificial Intelligence Review*, pp. 1–19, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10462-012-9341-3

[12] G. Lavee, E. Rivlin, and M. Rudzsky, "Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, no. 5, pp. 489–504, 2009.

[13] H. Liu, S. Chen, and N. Kubota, "Intelligent video systems and analytics: A survey," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 3, pp. 1222–1233, 2013.

[14] B. Morris and M. Trivedi, "A survey of vision-based trajectory learning and analysis for surveillance," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 8, pp. 1114–1127, 2008.

[15] A. Sodemann, M. Ross, and B. Borghetti, "A review of anomaly detection in automated surveillance," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 6, pp. 1257–1272, 2012.

[16] B. T. Morris and M. M. Trivedi, "Understanding vehicular traffic behavior from video: a survey of unsupervised approaches," *Journal of Electronic Imaging*, vol. 22, no. 4, pp. 041 113–041 113, 2013. [Online]. Available: http://dx.doi.org/10.1117/1.JEI.22.4.041113

[17] O. Popoola and K. Wang, "Video-based abnormal human behavior recognition: A review," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 6, pp. 865–878, 2012.

[18] M. F. Mokbel, T. M. Ghanem, and W. G. Aref, "Spatio-temporal access methods," *IEEE Data Eng. Bull.*, vol. 26, no. 2, pp. 40–49, 2003.

[19] L.-V. Nguyen-Dinh, W. G. Aref, and M. F. Mokbel, "Spatio-temporal access methods: Part 2 (2003 - 2010)," *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 46–55, 2010.

[20] W. Nie, A. Liu, and Y. Su, "Multiple person tracking by spatiotemporal tracklet association," in *Proceedings of the 9th AVSS Conference*. Beijing, China: IEEE, Septemer 18-21 2012.

[21] J. Badie, S. Bak, S. Serban, , and F. Bremond, "Recovering people tracking errors using enhanced covariance-based signatures," in *Proceedings of the 9th AVSS Conference*. Beijing, China: IEEE, Septemer 18-21 2012.

[22] M. Hofmann, M. Haag, and G. Rigoll, "Unified hierarchical multi-object tracking using global data association," in *Performance Evaluation of Tracking and Surveillance (PETS), 2013 IEEE International Workshop on*, 2013, pp. 22–28.

[23] I. Haritaoglu, D. Harwood, and L. S. David, "W4: Real-time surveillance of people and their activities," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, August 2000.

[24] D. Conte, P. Foggia, G. Percannella, and M. Vento, "Performance evaluation of a people tracking system on pets2009 database," in

*Proceedings of the 7th IEEE International Conference on AVSS*, 2010, pp. 119–126.

[25] Z. Chen, T. Ellis, and S. A. Velastin, "Vehicle detection, tracking and classification in urban traffic," in *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2012.

[26] Z. Jiang, D. Q. Huynh, W. Moran, and S. Challa, "Tracking pedestrians using smoothed colour histograms in an interacting multiple model framework." in *ICIP*, B. Macq and P. Schelkens, Eds. IEEE, 2011, pp. 2313–2316.

[27] C. Dai, Y. Zheng, and X. Li, "Pedestrian detection and tracking in infrared imagery using shape and appearance," *Computer Vision and Image Understanding*, vol. 106, no. 23, pp. 288 – 299, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314206001925

[28] T.-L. L. Hwann-Tzong Chen, Horng-Horng Lin, "Multi-object tracking using dynamical graph matching," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2001, pp. 210–217.

[29] J. Zhang, L. L. Presti, and S. Sclaroff, "Online multi-person tracking by tracker hierarchy," in *Proceedings of the 9th AVSS Conference*. Beijing, China: IEEE, Septemer 18-21 2012.

[30] T. Xu, P. Peng, X. Fang, C. Su, Y. Wang, Y. Tian, W. Zeng, and T. Huang, "Single and multiple view detection, tracking and video analysis in crowded environments," in *Proceedings of the 9th AVSS Conference*. Beijing, China: IEEE, Septemer 18-21 2012.

[31] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Computer Vision, 2009 IEEE 12th International Conference on*, 29 2009-oct. 2 2009, pp. 261 –268.

[32] Q. Delamarre and O. Faugeras, "3d articulated models and multiview tracking with physical forces," *Computer Vision and*

*Image Understanding*, vol. 81, pp. 328–357, March 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=376890.376922

[33] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 9, pp. 1806 –1819, sept. 2011.

[34] D.-T. Lin and K.-Y. Huang, "Collaborative pedestrian tracking and data fusion with multiple cameras," *Information Forensics and Security, IEEE Transactions on*, vol. 6, no. 4, pp. 1432 – 1444, dec. 2011.

[35] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 142–149.

[36] H. Hai Tao, Sawhney and R. Kumar, "Object tracking with bayesian estimation of dynamic layer representations," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 75–89, 2002.

[37] K. Bhuvaneswari and H. Abdul Rauf, "Edgelet based human detection and tracking by combined segmentation and soft decision," in *Control, Automation, Communication and Energy Conservation, 2009. INCACEC 2009. 2009 International Conference on*, june 2009, pp. 1 –6.

[38] Z. Han, Q. Ye, and J. Jiao, "Combined feature evaluation for adaptive visual object tracking," *Computer Vision and Image Understanding*, vol. 115, no. 1, pp. 69 – 80, 2011.

[39] B. Yogameena, S. Roomi, and S. Abhaikumar, "Detecting and tracking people in a homogeneous environment using skin color model," in *Advances in Pattern Recognition, 2009. ICAPR '09. Seventh International Conference on*, feb. 2009, pp. 282 –285.

[40] Y. Cai, N. de Freitas, and J. Little, "Robust visual tracking for multiple targets," in *Computer Visionâ ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and

A. Pinz, Eds., vol. 3954. Springer Berlin / Heidelberg, 2006, pp. 107–118.

[41] H. Wang, D. Suter, K. Schindler, and C. Shen, "Adaptive object tracking based on an effective appearance filter," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 9, pp. 1661 –1667, sept. 2007.

[42] W. Hu, X. Zhou, M. Hu, and S. Maybank, "Occlusion reasoning for tracking multiple people," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 1, pp. 114 –121, jan. 2009.

[43] J. Saboune and R. Laganiere, "People detection and tracking using the explorative particle filtering," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, 27 2009-oct. 4 2009, pp. 1298 –1305.

[44] L. Bazzani, M. Cristani, and V. Murino, "Collaborative particle filters for group tracking," in *IEEE Int. Conf. on Image Processing*, 2010, pp. 837–840.

[45] S. Yin, J. H. Na, J. Y. Choi, and S. Oh, "Hierarchical kalman-particle filter with adaptation to motion changes for object tracking," *Computer Vision and Image Understanding*, vol. 115, no. 6, pp. 885 – 900, 2011.

[46] H. Medeiros, G. Holguin, P. J. Shin, and J. Park, "A parallel histogram-based particle filter for object tracking on simd-based smart cameras," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1264 – 1272, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314210000974

[47] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 9, pp. 1820 –1833, sept. 2011.

[48] J. Lee, S. Lankton, and A. Tannenbaum, "Object tracking and target reacquisition based on 3-d range data for moving vehicles,"

*Image Processing, IEEE Transactions on*, vol. 20, no. 10, pp. 2912 –2924, oct. 2011.

[49] X. Song, J. Cui, H. Zha, and H. Zhao, "Vision-based multiple interacting targets tracking via on-line supervised learning," in *Proceedings of the 10th European Conference on Computer Vision: Part III*, ser. ECCV '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 642–655.

[50] M. Wang, H. Qiao, and B. Zhang, "A new algorithm for robust pedestrian tracking based on manifold learning and feature selection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 4, pp. 1195–1208, 2011.

[51] S. Mitra and T. Acharya, "Gesture recognition: A survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 3, pp. 311–324, 2007.

[52] R. Poppe, "A survey on vision-based human action recognition," *Image Vision Comput.*, vol. 28, no. 6, pp. 976–990, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.imavis.2009.11.014

[53] 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.

[54] N. Piotto, F. G. B. De Natale, and N. Conci, "Hierarchical matching of 3d pedestrian trajectories for surveillance applications," in *Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on*, 2009, pp. 146–151.

[55] 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.

[56] F. Bashir, A. Khokhar, and D. Schonfeld, "Object trajectory-based activity classification and recognition using hidden markov models," *IEEE Trans. Image Processing*, vol. 16, no. 7, pp. 1912–1919, 2007.

[57] S. Atev, G. Miller, and N. Papanikolopoulos, "Clustering of vehicle trajectories," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, no. 3, pp. 647 –657, sept. 2010.

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

[59] Z. Fu, W. Hu, and T. Tan, "Similarity based vehicle trajectory clustering and anomaly detection," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 2, 2005, pp. II–602–5.

[60] 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, pp. 1035–1044.

[61] A. Prati, S. Calderara, and R. Cucchiara, "Using circular statistics for trajectory shape analysis," in *CVPR*, june 2008, pp. 1 –8.

[62] L. Chen, M. T. Özsu, and V. Oria, "Symbolic representation and retrieval of moving object trajectories," in *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, ser. MIR '04. New York, NY, USA: ACM, 2004, pp. 227–234.

[63] N. Piotto, N. Conci, and F. G. B. De Natale, "Syntactic matching of trajectories for ambient intelligence applications," *Trans. Multi.*, vol. 11, no. 7, pp. 1266–1275, Nov. 2009.

[64] 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.

[65] H.-Y. Cheng and J.-N. Hwang, "Integrated video object tracking with applications in trajectory-based event detection," *J. Vis.*

*Comun. Image Represent.*, vol. 22, no. 7, pp. 673–685, Oct. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jvcir.2011.07.001

[66] N. Saunier and T. Sayed, "Clustering vehicle trajectories with hidden markov models application to automated traffic safety analysis," in *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, 2006, pp. 4132–4138.

[67] T. Xiang and S. Gong, "Incremental and adaptive abnormal behaviour detection," *Comput. Vis. Image Underst.*, vol. 111, no. 1, pp. 59–73, Jul. 2008.

[68] B. Morris and M. Trivedi, "Learning trajectory patterns by clustering: Experimental studies and comparative evaluation," in *CVPR*, 2009, pp. 312–319.

[69] 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

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

[71] S.-C. Chen, M.-L. Shyu, S. Peeta, and C. Zhang, "Learning-based spatio-temporal vehicle tracking and indexing for transportation multimedia database systems," *Trans. Intell. Transport. Sys.*, vol. 4, no. 3, pp. 154–167, Sep. 2003. [Online]. Available: http://dx.doi.org/10.1109/TITS.2003.821290

[72] S. Bhonsle, M. Trivedi, and A. Gupta, "Database-centered architecture for traffic incident detection, management, and analysis," in *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, 2000, pp. 149 –154.

[73] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proc. of ACM SIGMOD Conference*. New York, NY, USA: ACM, 1984, pp. 47–57.

[74] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis, *R-Trees: Theory and Applications (Advanced Information and Knowledge Processing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[75] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel approaches in query processing for moving object trajectories," in *Proc. of VLDB Conf.* San Francisco, CA, USA: Morgan Kaufmann Publ. Inc., 2000, pp. 395–406.

[76] E. Frentzos, "Indexing objects moving on fixed networks," in *8th International Symposium on Advances in Spatial and Temporal Databases (SSTD 2003)*. Springer, 2003, pp. 289–305.

[77] V. T. De Almeida and R. H. Güting, "Indexing the trajectories of moving objects in networks*," *Geoinformatica*, vol. 9, pp. 33–60, March 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1046957.1046970

[78] I. S. Popa, K. Zeitouni, V. Oria, D. Barth, and S. Vial, "Parinet: A tunable access method for in-network trajectories," in *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, F. Li, M. M. Moro, S. Ghandeharizadeh, J. R. Haritsa, G. Weikum, M. J. Carey, F. Casati, E. Y. Chang, I. Manolescu, S. Mehrotra, U. Dayal, and V. J. Tsotras, Eds. IEEE, 2010, pp. 177–188.

[79] I. Sandu Popa, K. Zeitouni, V. Oria, D. Barth, and S. Vial, "Indexing in-network trajectory flows," *The VLDB Journal*, vol. 20, pp. 643–669, Oct. 2011.

[80] S. Rasetic, J. Sander, J. Elding, and M. A. Nascimento, "A trajectory splitting model for efficient spatio-temporal indexing," in *Proceedings of VLDB*, ser. VLDB '05. VLDB Endowment, 2005, pp. 934–945. [Online]. Available: http://dl.acm.org/citation.cfm?id=1083592.1083700

[81] V. P. Chakka, A. Everspaugh, and J. M. Patel, "Indexing large trajectory data sets with seti," in *First Biennial Conference on*

*Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, USA, 2003.

[82] P. Cudre-Mauroux, E. Wu, and S. Madden, "Trajstore: An adaptive storage system for very large trajectory data sets," in *Int. Conf. on Data Engineering*. Los Alamitos, CA, USA: IEEE CS, 2010, pp. 109–120.

[83] R. Obe and L. Hsu, *PostGIS in Action*. Greenwich, CT, USA: Manning Publications Co., 2011.

[84] M. A. Anusuya and S. K. Katti, "Speech recognition by machine, a review," *CoRR*, vol. abs/1001.2267, 2010.

[85] L. Besacier, E. Barnard, A. Karpov, and T. Schultz, "Automatic speech recognition for under-resourced languages: A survey," *Speech Communication*, vol. 56, no. 0, pp. 85–100, 2014.

[86] Z. Saquib, N. Salam, R. Nair, N. Pandey, and A. Joshi, "A survey on automatic speaker recognition systems," in *Signal Processing and Multimedia*, ser. Communications in Computer and Information Science, T.-h. Kim, S. Pal, W. Grosky, N. Pissinou, T. Shih, and D. Olfozak, Eds. Springer Berlin Heidelberg, 2010, vol. 123, pp. 134–145.

[87] A. Roy, M. Magimai-Doss, and S. Marcel, "A fast parts-based approach to speaker verification using boosted slice classifiers," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 241–254, 2012.

[88] C. Clavel, T. Ehrette, and G. Richard, "Events detection for an audio-based surveillance system," in *ICME*, 2005, pp. 1306 –1309.

[89] M. Vacher, D. Istrate, L. Besacier, J. F. Serignat, and E. Castelli, "Sound Detection and Classification for Medical Telesurvey," in *Proc. 2nd Conference on Biomedical Engineering*, C. ACTA Press, Ed., Innsbruck, Austria, Feb. 2004, pp. 395–398.

[90] J.-L. Rouas, J. Louradour, and S. Ambellouis, "Audio events detection in public transport vehicle," in *IEEE ITSC*, 2006, pp. 733–738.

[91] L. Gerosa, G. Valenzise, M. Tagliasacchi, F. Antonacci, and A. Sarti, "Scream and gunshot detection in noisy environments," in *Proc. EURASIP European Signal Processing Conference*, Poznan, Poland, 2007.

[92] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti, "Scream and gunshot detection and localization for audio-surveillance systems," in *IEEE AVSS*, 2007, pp. 21–26.

[93] S. Ntalampiras, I. Potamitis, and N. Fakotakis, "An adaptive framework for acoustic monitoring of potential hazards," *EURASIP J. Audio Speech Music Process.*, vol. 2009, pp. 13:1–13:15, Jan. 2009.

[94] ——, "Probabilistic novelty detection for acoustic surveillance under real-world conditions," *IEEE Trans. Multimedia*, vol. 13, no. 4, pp. 713 –719, 2011.

[95] A. Rabaoui, M. Davy, S. Rossignol, and N. Ellouze, "Using one-class svms and wavelets for audio surveillance," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 4, pp. 763–775, 2008.

[96] D. Conte, P. Foggia, G. Percannella, A. Saggese, and M. Vento, "An ensemble of rejecting classifiers for anomaly detection of audio events," in *IEEE AVSS*, 2012, pp. 76–81.

[97] M. Chin and J. Burred, "Audio event detection based on layered symbolic sequence representations," in *IEEE ICASSP*, 2012, pp. 1953–1956.

[98] H. Malik, "Acoustic environment identification and its applications to audio forensics," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1827–1837, 2013.

[99] A. Ribbrock and F. Kurth, "A full-text retrieval approach to content-based audio identification," in *IEEE Workshop on Multimedia Signal Processing*, 2002, pp. 194–197.

[100] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "Music classification via the bag-of-features approach," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1768 – 1777, 2011.

[101] D. Conte, P. Foggia, M. Petretta, F. Tufano, and M. Vento, "Meeting the application requirements of intelligent video surveillance systems in moving object detection," in *Proceedings of the Third international conference on Pattern Recognition and Image Analysis - Volume Part II*, ser. ICAPR'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 653–662. [Online]. Available: http://dx.doi.org/10.1007/11552499_72

[102] D. Conte, P. Foggia, G. Percannella, F. Tufano, and M. Vento, "An experimental evaluation of foreground detection algorithms in real scenes," in *EURASIP Journal on Advances in Signal Processing*, no. 11, 2010.

[103] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, june 2005, pp. 886 –893 vol. 1.

[104] M. Fredman and R. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Foundations of Computer Science, IEEE Annual Symposium on*, vol. 0, pp. 338–346, 1984.

[105] P. Foggia, G. Percannella, A. Saggese, and M. Vento, "Real-time tracking of single people and groups simultaneously by contextual graph-based reasoning dealing complex occlusions," in *Proceedings of the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*. IEEE, 2013.

[106] 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

[107] B. Morris and M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 33, no. 11, pp. 2287 –2301, nov. 2011.

[108] 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.

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

[110] 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.

[111] 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.

[112] 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.

[113] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui, "A kernel for time series based on global alignments," *CoRR*, vol. abs/cs/0610033, pp. 413–416, 2006.

[114] M. Cuturi, "Fast global alignment kernels," in *ICML)*, L. Getoor and T. Scheffer, Eds. New York, NY, USA: ACM, June 2011, pp. 929–936.

[115] 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.

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

[117] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice in C (2nd Edition)*. Addison-Wesley, 2004.

[118] OGC, "OGC Standards," `http://www.opengeospatial.org/standards/sfs`, 2012.

[119] B. Haasdonk and E. Pekalska, "Classification with kernel mahalanobis distance classifiers," in *GfKl*, 2008, pp. 351–361.

[120] 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

[121] IEEE and ISO/IEC, "Multimedia Content Description Interface - Part 4: Audio," *ISO/IEC 42010 IEEE Std 1471-2000 First edition 2007-07-15*, 2001.

[122] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," IRCAM, Tech. Rep., 2004.

[123] Z. Liu, Y. Wang, and T. Chen, "Audio Feature Extraction and Analysis for Scene Segmentation and Classification," *The Journal of VLSI Signal Processing*, vol. 20, no. 1, pp. 61–79, Oct. 1998.

[124] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[125] J. Ferryman and A. Ellis, "Pets2010: Dataset and challenge," in *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, 2010, pp. 143–150.

[126] T. D"Orazio, M. Leo, N. Mosca, P. Spagnolo, and P. Mazzeo, "A semi-automatic system for ground truth generation of soccer video sequences," in *Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on*, Sept., pp. 559–564.

[127] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 2, pp. 319–336, 2009.

[128] A. Ellis and J. Ferryman, "PETS2010 and PETS2009 evaluation of results using individual ground truthed single views," in *IEEE Int. Conf. on Advanced Video and Signal Based Surveillance*, 2010, pp. 135–142.

[129] A. Alahi, L. Jacques, Y. Boursier, and P. Vandergheynst, "Sparsity-driven people localization algorithm: Evaluation in crowded scenes environments," in *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*, dec. 2009, pp. 1 –8.

[130] R. Di Lascio, P. Foggia, G. Percannella, A. Saggese, and M. Vento, "A real time algorithm for people tracking using contextual reasoning," *Computer Vision and Image Understanding*, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2013.04.004

[131] R. Eshel and Y. Moses, "Homography based multiple camera detection and tracking of people in a dense crowd," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, june 2008, pp. 1 –8.

[132] P. Foggia and M. Vento, "A middleware platform for real-time processing of multiple videostreams based on the data-flow paradigm," in *2011 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE Computer Society, 2011, pp. 1–6.

[133] B. Majecka, "Statistical models of pedestrian behaviour in the forum," Ph.D. dissertation, School of Informatics, University of Edinburgh, 2009.

[134] 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.

[135] 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.

[136] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[137] 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

[138] C. Ding and X. He, "K-means clustering via principal component analysis," in *ICML*. ACM, 2004, pp. 29–35.

[139] X. Wang, K. Tieu, and E. Grimson, "Learning semantic scene models by trajectory analysis," in *Proceedings of the 9th European conference on Computer Vision - Volume Part III*, ser. ECCV'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 110–123.

[140] X. Wang, K. T. Ma, G.-W. Ng, and W. Grimson, "Trajectory analysis and semantic region modeling using a nonparametric bayesian model," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1–8.