Dipartimento di Informatica e
Scienze dell'Informazione

Learning to classify visual dynamic cues

by

Nicoletta Noceti

Theses Series **DISI-TH-2010-09**

DISI, Università di Genova
v. Dodecaneso 35, 16146 Genova, Italy http://www.disi.unige.it/

Università degli Studi di Genova

Dipartimento di Informatica e
Scienze dell'Informazione

Dottorato di Ricerca in Informatica

Ph.D. Thesis in Computer Science

Learning to classify visual dynamic cues

by

Nicoletta Noceti

July, 2010

Dottorato di Ricerca in Informatica
Dipartimento di Informatica e Scienze dell'Informazione
Università degli Studi di Genova

DISI, Univ. di Genova
via Dodecaneso 35
I-16146 Genova, Italy
http://www.disi.unige.it/

Ph.D. Thesis in Computer Science (S.S.D. INF/01)

Submitted by Nicoletta Noceti
DISI, Univ. di Genova
noceti@disi.unige.it

Date of submission: February 2010

Title: Learning to classify visual dynamic cues

Advisor: Francesca Odone
Dipartimento di Informatica e Scienze dell'Informazione
Università di Genova
odone@disi.unige.it

Ext. Reviewers:
Andrea Cavallaro
School of Electronic Engineering and Computer Science
Queen Mary University of London
andrea.cavallaro@elec.qmul.ac.uk

Andrea Prati
Dipartimento di Scienze e Metodi dell'Ingegneria
Università di Modena e Reggio Emilia
andrea.prati@unimore.it

Abstract

Classification based on dynamic information is a challenging research domain that finds application in a number of fields, including video-surveillance and video retrieval.
Traditional approaches based on motion analysis address many interesting applications, such as access control, anomaly detection, congestion analysis and multi-camera event description: in all these cases it is common practice to devise a measurement phase that extracts low level information from videos. To this purpose a wide variety of methods have been presented in the computer vision literature, leading to solutions that effectively describe the video content in moderately difficult conditions. A well known limit of these methods is that while they provide effective tools to model the dynamics of a single video, they do not suffice when the problem of interest requires a higher generalization level.
In the case of behaviors modeling or dynamic events classification, it is advisable to increase the abstraction of the data, designing higher-level descriptions able to model more general structures. In recent years a few interesting works employing machine learning methods showed how these techniques may improve performance in terms of accuracy and efficiency: data-driven approaches are effective to understand possible correlations between measurements and provide systems with the ability of being adaptive. In the field of video surveillance we may exploit the availability of possibly huge sets of examples, acquired by long time observations, and endowed with an internal structure provided by temporal coherence.
Within this framework this thesis focuses on:

- Studying and developing of robust methods to retrieve space-time information from a video;
- Studying and developing of higher-level descriptions, in order to include space-time information within a machine learning framework;
- Devising machine learning strategies to model common events and anomalies from huge sets of (possibly) unlabeled examples.

These objectives will be addressed both from the algorithmic and the application standpoint and will be integrated in a prototype architecture that combines vision methods for scene perception and analysis and learning techniques for high level description and decision making.

To my family

As far as the laws of mathematics refer to reality, they are not certain; as far as they are certain, they do not refer to reality. (Albert Einstein)

List of symbols

I_t	video frame at time t
B_t	background model at time t
M_t	motion segmentation at time t
$\mathcal{M}_i^t$	feature vector of target i at time t (tracking)
TH_i^t	time stamp of frame at time t (when part of $\mathcal{M}_i^t$)
$\mathbf{P}_i^t = [x_i^t, y_i^t]$	position of target i at time t
S_i^t	size of target i at time t
$\mathbf{V}_i^t = [Vx_i^t, Vy_i^t]$	velocity of target i at time t
M_i^t	velocity magnitude of target i at time t
D_i^t	velocity direction of target i at time t
$\mathbf{CH}_i^t$	color descriptor of target i at time t
W_i^t	width of bounding box enclosing target i at time t
H_i^t	height of bounding box enclosing target i at time t
x_i^t	t-th element of i-th trajectory
$\mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^{k_i})$	i-th trajectory of length k_i
$\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$	set of N trajectories
$\tilde{\mathbf{X}} = \{\{x_i^t\}_{t=1\ldots k_i}\}_{i=1\ldots N}$	subset of the feature space
$K(x, y)$	generic kernel function between x and y
$K_P(s, t)$	P-Spectrum kernel between strings s and t
$\mathcal{A}$	alphabet (partition) on the feature space
$\mathbf{W} = [W_P, W_S, W_M, W_D]$	weights in the multi-cue kernel
TH_{points}	spectral clustering cut threshold for alphabet computation
$TH_{strings}$	spectral clustering cut threshold for behavioral patterns detection
λ	regularization parameter of RLS classifier
$\{M_i\}_{i_1\ldots B}$	candidates of B behavioral patterns
τ_1,τ_2	thresholds of test data associations

Chapter 1

Introduction

1.1 Scope and motivations

Classification based on dynamic information is a challenging problem in computer vision and plays a central role in a number of applications, including video surveillance, human-machine interface, and semantic retrieval from videos [HTWM04, ZJ05, EBMM03, Rob05]. The large scale diffusion of video cameras has been favored by advances on hardware components that allow for easy storage and fast processing. As a side effect, in the last decades we have assisted to the growing interest for applications where central topics are the analysis and the interpretation of video content .
Although the urging need for effective systems, is fueling an intense research activity, several crucial issues are still either not solved or even not addressed, especially for what concerns the ability of mining information from big video data sets.

In this work we refer specifically to the field of video surveillance, where an important issue is the efficient analysis of behavioral patterns of people activities in the scene.
The study and the understanding of human activities from videos has been widely addressed in the last decades (see for instance [SG00, Gea98, IB00, PCV00, PMF08, AC08]), particularly with the availability of an enormous amount of installed video surveillance cameras. As a consequence, a huge amount of video data are daily acquired, becoming more and more difficult to be handled by human operators. This justifies the growing need for computational methods able to assist the user, suggesting where to focus attention.

In video analysis, the mutual distance between camera and observed scenario, or more specifically the camera field of view, influences the specific tasks the system might be able to address. In the video surveillance framework, in particular, it suggests a number of different applications, ranging from crowd analysis, when the distance is relevant, to the problem of considering the motion or even the specific action of single entities, and of

humans in particular.
In this work, we assume we are monitoring possibly complex scenarios from a distance where the "action of interest" is the trajectory of a moving object as a whole, and no information is available or needed on the motion of objects parts. We thus explicitly refer to data that may be modeled as time-series of instantaneous observations.

It is in this framework that our work has been developed, with the final, ambitious goal of modeling common behaviors by learning frequent patterns of activities from huge sets of unlabeled data, with a very limited a-priori information included into the pipeline. The understanding of what is usual in a given scenario, also, sets the basis for anomalies detection, where an anomaly is thought of as an event differing from what usually observed, thus potentially (but not surely) dangerous.

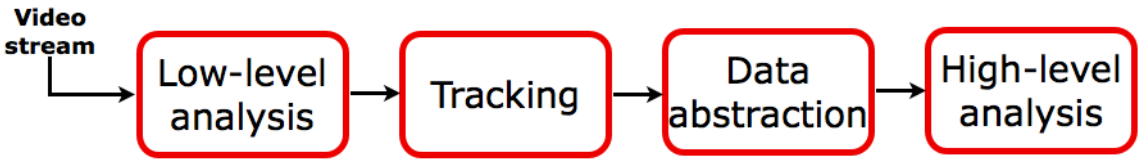


Figure 1.1: A visual representation of the general processing pipeline of a video surveillance system.

The availability in our reference application of *long-time observations* calls for solutions be *adaptive* and able to exploit knowledge coming from *previously seen scenarios*.
To achieve these results we study, implement and validate computer vision and machine learning methods that can be cast in a rather standard processing pipeline (see Fig. 1.1):

- An initial *low-level processing* allows us to detect moving objects in the observed environment by means of motion-based image segmentation. The moving structures are then described by an appropriate feature vector;
- The instantaneous observations of such objects are tracked along the video sequence in order to build a model of their dynamic evolution as long as their *identities* can be preserved. The output of the tracking procedure is a set of trajectories representative of *dynamic events* occurring into the scene;
- In order to understand and describe common patterns of activities (*behaviors*) a higher-level analysis is applied to detect internal structures among the set of trajectories.

The trajectories, estimated by the tracking process can be conveniently exploited to perform a higher level analysis aiming at understanding the dynamic evolution of the observed

scene. To this purpose, traditional computer vision approaches represent a viable solution when the focus is on a single event (or even a single class of events), while they are not appropriate when a higher generalization is required, as in the task of behavior understanding.
A priori knowledge on the analysed environment and its dynamics may be exploited in different ways. A first example, widely explored in the literature, refers to stochastic grammars (see, for instance, [IB00]). In certain application domains (e.g., traffic control) the amount of structures contained in the expected events may be profitably used to model the dynamics of the scene. For instance, in [MCB+01] the results obtained by a robust low-level processing module are associated to context information to identify possible abnormal behaviors in aerial images.
Otherwise a very effective way to exploit prior knowledge is to rely on learning from examples. In the last decades, it has been assessed how computer vision approaches can be profitably coupled with the paradigm of learning from examples [Vap98, HTF03]. The paradigm refers to a widely used set of statistical tools to extract knowledge on the basis of the properties of a given data set, called the *training set*, estimating the underlying model(s) on top of which they are generated. The learned models should be able to represent in the best way the training set, and at the same time a *generalization* to other data must be ensured. One of the main characteristics of learning is, actually, the ability to perform a prediction on the model of a new data on the basis of the rules previously learned.
It is possible to distinguish among different kinds of *learning from examples*. In *supervised learning* the available data are a collection of pairs $\mathbf{z} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where $\mathbf{x}$ is a vector whereas y takes on discrete or continuous values. The key idea of the supervised learning paradigm is to infer the unknown input-output relation on the basis on the training set. The nature of the output leads to the definition of *patterns classification* problems, if output values are taken from a finite unordered set $C = \{C_1, \ldots, C_k\}$, or *regression* problems, when output values are real numbers.
In *unsupervised learning* no output correspondent the the input data is provided: under this perspective, the training set can be interpreted as observations of a random vector having some probability density. The goal of unsupervised techniques is to directly infer properties of this probability density without the help of a supervisor. In a sense, unsupervised learning can be thought of as finding patterns in the data above and beyond what would be considered pure unstructured noise.

In the case of behavior analysis, if the available data are labeled, i.e., to each one of them we may associate a label of a known behavior, the use of state-of-the-art machine learning algorithms lead to effective behavior categorization methods. On this respect we mention the work by Pittore et al. [PCV00], where the available data are mapped in the parameter space induced by the fit of each trajectory with B splines, then Support Vector Machines (SVMs) are applied. Otherwise, the sequential structure of trajectories may be captured by

training a well designed Hidden Markov Models (HMM) [Rab89] or other similar dynamical systems (see, for instance, [BKS07] and references therein).

Focusing on the task of modeling behaviors, we point out that given the huge variety of possible behavioral patterns, even in moderately complex scenarios, and the high variability of such phenomena, a fully supervised approach in unfeasible, requiring the availability of an accurate annotation that, considering the great quantity of data, is not an option. For the same reasons, it is likely to fail to produce systems that are both robust, i.e. able to detect and recognize effectively many classes of events, and flexible, i.e. easy to be modified to recognize new (classes of) events.
The same characteristics provide an ideal test bed for the application of an unsupervised approach. The peculiarity of video-surveillance is a great help in this direction. Usually a system observes the same scene for very long periods. Thus, such system could learn common behaviors in that scene from long-time observations, carried out in an initial or intermediate calibration stage. This aim naturally calls for some form of unsupervised data analysis: an analysis of available literature makes it clear that unsupervised methods for behavior understanding or, more in general, to time series analysis are quite limited - see [JST07, JCN06, ME02]. The most popular technique among unsupervised methods is *data clustering*. It refers to a well-known and widely applied tool to automatically partition a set of data into coherent sub-sets (or *clusters*) with respect to an appropriate similarity measure. The literature on unsupervised learning, and clustering in particular, is very large, a survey for the specific case of temporal data can be found in [Lia05] to which we refer the interested reader. We consider in our work *spectral clustering* [SM00] based on the spectral analysis of a special matrix, the *Laplacian*, to compute the data partition on a graph structure adopted to summarize the pairwise similarities between points.
Referring to our setting, the presence of the temporal component requires a more accurate handling of the data. Previous works on clustering temporal series approached the problem from two distinct viewpoints (see [6] and references therein). The first approach relies on modifying suitable existing algorithms for clustering static data in such a way that time series data can be handled. An alternative is to convert temporal series data into the form of static data so that the existing algorithms for clustering static data can be directly used, trying to keep information on the structure of data. Our work is more related to this latter approach.

A challenging question when dealing with unsupervised learning methods is how to evaluate the goodness of results, and also, related to it, how to appropriately select the best set of parameters. Popular approaches are based on the use on quality measures, often referred to as *quality indices* [HBV01, GB03] that should quantify the goodness of a partition by summarizing important properties that must hold. However, there are no attempts to apply this technique to complex feature spaces and to time-series analysis, and it is not clear whether the quality criteria that are meaningful for static data still hold for temporal data, and for the specific task. We explore this direction observing how a limited benefit

may be obtained from such measures. Instead we define an evaluation procedure based on a set of rough feedbacks given by the user on the specific scenario.

Even if the research on video analysis and scene understanding is greatly contributing to these problems, in this general setting a number of questions are still open "how to represent data limiting the amount of ambiguities, when there is no or little knowledge on the underlying data acquisition process?" And, more specifically, "how to exploit the data internal structure induced by time-coherence of observations?" Also, "how to learn from large amounts of noisy data, particularly when they belong to high dimensional spaces?" Moreover, the specific domain we refer to opens various practical issues, related to the scene complexity, the specications of the video-surveillance system (e.g., the number and the position of the cameras), and the amount of prior information available on the observed phenomena. Referring to the pipeline stated above, our contributions are several. With respect to the video processing, we propose a tracking procedure based on a combined motion and appearance model of objects [NDLO09]. The method is specifically designed for video surveillance purposes, reaching high performances from the point of view of accuracy in data association while keeping the computational cost low. In the worst case we achieve an almost real-time performance (about 20fps).

For what concerns high-level analysis, we propose a pipeline for a system of behavior understanding based on the use of spectral clustering to detect frequent patterns of activity by analyzing (possibly) big sets of time-series data [NSO08b, NSO08a, NO09, NSO10, NO10]. The choice of a proper input space strongly affect the capability of the system to detect what is "common" or "anomalous" in a given scenario. Although lots of work in the literature is based on trajectories of $2D$ observations (the instantaneous position in the image plane), we experienced how the learning model can benefit from the adoption of a more heterogeneous input space, especially when the mutual positions of camera and observed scene gives rise to data ambiguities when considering only the positions.

In particular we address two main issues: (1) *How data should be represented?*, and (2) *How to compare them?*. We thus carry out an extensive comparison of representation schemes including fitting with curves (polynomials and B-Splines in particular), the well-known Hidden Markov Models and a string-based approach [NSO08b, NSO10]. The choice of the specific feature map by means of which to *represent* a specific moving object at each time instant is not crucial to our analysis: we only require that the sequences are described by means of $d \times T$ matrices, where d is the number of components of the feature map and T is the number of elements in the sequence. In general, T may change from sequence to sequence.

The last contribution of the thesis builds on top of the previous achievements and is a string-based behavior understanding pipeline relying on a loose annotation [NSO10, NO10].
An extensive experimental analysis is performed analyzing different kind of data: from synthetic trajectories, to real controlled data for which a ground truth is available, to

conclude on large-scale data sets, spanning weeks of acquisitions from a video surveillance system. Making use of learning techniques, we evaluate (i) the robustness of the representation schemes (in this case supervised and unsupervised approaches are both adopted and compared), and (ii) the reliability of clustering as a tool to detect common behaviors. The obtained results lead us to conclude that the string-based approach is the most promising for our setting, coupling the capability of strongly characterizing the input data as well as the final models, with a high adaptability to different scenarios, being the computation of the trajectories representations and the behavioral models almost completely independent from the input space.

In the last set of experiments, we perform a test analysis on data of a week, with the aim of judging our system with respect to the capability of generalizing the dynamic properties learned during the modeling phase to new observed and (possibly) different events. Notice that the ability to predict the "class" of a new event is a fundamental add-on for monitoring systems that can aspire to address the challenging problem on detecting anomalies.

1.2 Structure of the thesis

This thesis is organized as follows:

- **Chapter 2** presents an overview of state-of-art methods related to the various topics considered in this thesis. The organization of the chapter reflects the structure of the thesis. We start from the analysis of methods for low-level video processing, considering the problem of segmenting images with respect to motion information and the problem of tracking objects in video sequences. The second part of the chapter describes the literature of high-level analysis of video contents: we review possible intermediate descriptions for trajectories and show how they have been applied to behavior analysis.

- In **Chapter 3** we address the video processing stage of the pipeline. The focus of the discussion is on the details of our modules for low-level image processing and object tracking. For the latter, we use a joint model of motion and appearance information: the problem of preserving the identities of interesting targets is approached using graph structures, naturally embedding different levels of complexity depending on the properties of the scene.
  The last part presents the experimental analysis, in which we compare our system against a selection of popular approaches using an annotation on the video content based on the use of a set of source-sink regions.

- **Chapter 4** is devoted to the discussion on the different higher-level representations, explaining advantages and drawbacks of each one on a synthetic data set of $2D$ obser-

vations. Each representation is coupled with an appropriate kernel-based similarity measure and the performance of each scheme is qualitatively evaluated analyzing similarity matrices. The quantitative analysis is based instead on supervised learning. In the second part of the chapter the benefit of using a heterogeneous input space, rather than the usual $2D$ information, is discussed, showing how it is a suitable way to cope with data ambiguities. The experimental analysis reflects the one performed on synthetic data and it is done on a set of real yet controlled trajectories, acquired from a surveillance camera, for which a ground truth has been specified.

- The clustering-based approach to behavior modeling is discussed in **Chapter 5**: each representation is used to feed the Spectral Clustering, which is equipped with the corresponding kernel. The experimental analysis, based on the same data of Chapter 3, aims at evaluating the clustering results with respect to the ground truth. In essence we evaluate the correspondence among real behaviors (the ones arising from the ground truth) and estimated behaviors (the clusters). We design two different procedures to cluster association, to cope with the requirements of a typical system for behavior analysis. The comparison of adopting supervised and unsupervised learning is discussed. At the end, a conclusion in favor of the string-based scheme is reached.

- In **Chapter 6** we propose a behavior modeling system based on the use of strings as meta-descriptors for the trajectories. The behavioral patterns are estimated by means of Spectral Clustering, according to the general pipeline discussed on the thesis. We evaluate the system with an extensive experimental analysis on data acquired by a video surveillance system over a temporal range of a few weeks. During the model selection phase since manual labeling of big training sets is not advisable, we adopt an automatic loose annotation based on spatial properties of the data.
  We then select as a model, the best performing clustering result with respect to the loose annotation. For completeness, we compare the system against the other representation schemes, including supervised classifiers trained on the loosely annotated data. The second part of the chapter is the place where we present the test analysis, performed again on data of weeks: two different out-of-sample strategies allows to associate a new test data to one of the known models or detecting it as an event different from the "normality" of the scene.

- The last chapter (**Chapter 7**) is left to conclusions and future developments. After a summary on the problems tackled in this thesis and the corresponding main contributions, we briefly review on-going work and highlight future directions for the work.

Chapter 2

State-of-art

In this chapter we present an overview of state-of-art methods related to the various topics considered in this thesis. The organization of the chapter reflects the structure of the thesis. We start from the analysis of methods for low-level video processing, considering the problem of segmenting images with respect to motion information (Sec. 2.1) and the problem of tracking objects in video sequences (Sec. 2.2). The second part of the chapter describes the literature of high-level analysis of video contents: we review possible intermediate descriptions for trajectories (Sec. 2.3) and show how they have been applied to behavior analysis (Sec. 2.4).

2.1 Motion-based image segmentation

The apparent motion of objects in the image plane is a strong visual cue to understand the semantics of 3D motion. A basic operation when dealing with videos is to analyze the information regarding small temporal windows. Within this setting, it is common practice to assume that the illumination does not change, so that image variations can be

SCENE / CAMERA	Static	Dynamic
Static	absence of motion	only a subset of moving pixels
Moving	Ego motion	2 types of motions

Table 2.1: Possible settings for video acquisitions lead to different characteristics of the dynamic content of a video.

interpreted as consequences of the relative motion between camera and scene.

On this respect, Table 2.1 summarizes possible settings. In the very general setting, a moving camera acquires images of a dynamic scene: this case would lead to very general solutions that could be applied to all scenarios to the price of a high computational cost. However, in most cases, a-priori information on the setting is available and would lead to ad hoc more efficient solutions. Low-level methods for studying the motion of a scene include pixel-based analysis to determine the displacements of each pixel between two consecutive frames: this operation can be performed globally (e. g. by computing the optical flow [SH81]) or locally using a sparse approach (e. g. by tracking local features, see [WB95] as an example). In both cases, the problem of estimating motion vectors is addressed.
A higher level of analysis is centered on the *motion segmentation* problem [WB95, ea05, HHD99, KCHD05], where the focus is on moving regions, instead of single pixels. The key idea is that regions from an image can be classified as static or dynamic. *Motion-based image segmentation*, in fact, aims at detecting regions corresponding to moving objects, providing hints on where to focus the attention for subsequent analysis. Structures moving into the scene are typically referred to as *foreground*, while the *background* includes all the elements permanently belonging to the scene, not always static (e.g. a waving tree is a permanent structure of the environment but an apparent motion will be estimated due to the wind).

From the algorithmic standpoint, the problem of motion segmentation can be stated as classifying a pixel as belonging to the *background* of the scene or to a moving structure of *foreground*. A number of approaches have been proposed in the literature, in the following we briefly review the most common: the selection includes approaches with the potential of performing in real-time, a central requirement in video surveillance.

When there is no information on the conditions under which the video was acquired, or again in the case of moving camera, segmentation is interwined with motion estimation. Optical flow provides a viable way to address the problem. In Fig. 2.1 an example is provided, where two consecutive images from a video are compared using optical flow: below, the estimated motion vectors clearly identify the moving person and suggest the direction and intensity of motion. Note that the procedure can be strongly affected by noise: on the bottom-right corner, a high-saturated area provides noisy, thus unreliable, information.
Instead, when the camera is still this can be solved as a pixel based classification (moving or still?). A connected component of pixels moving coherently can be seen as a visual representation of a structure performing some dynamic action in the scene: it is usually called *target* or *blob* (in what follows the two terms will be adopted as synonyms).
In the following, we discuss more in deeply this latter setting, the one we consider in the remainder of the thesis.

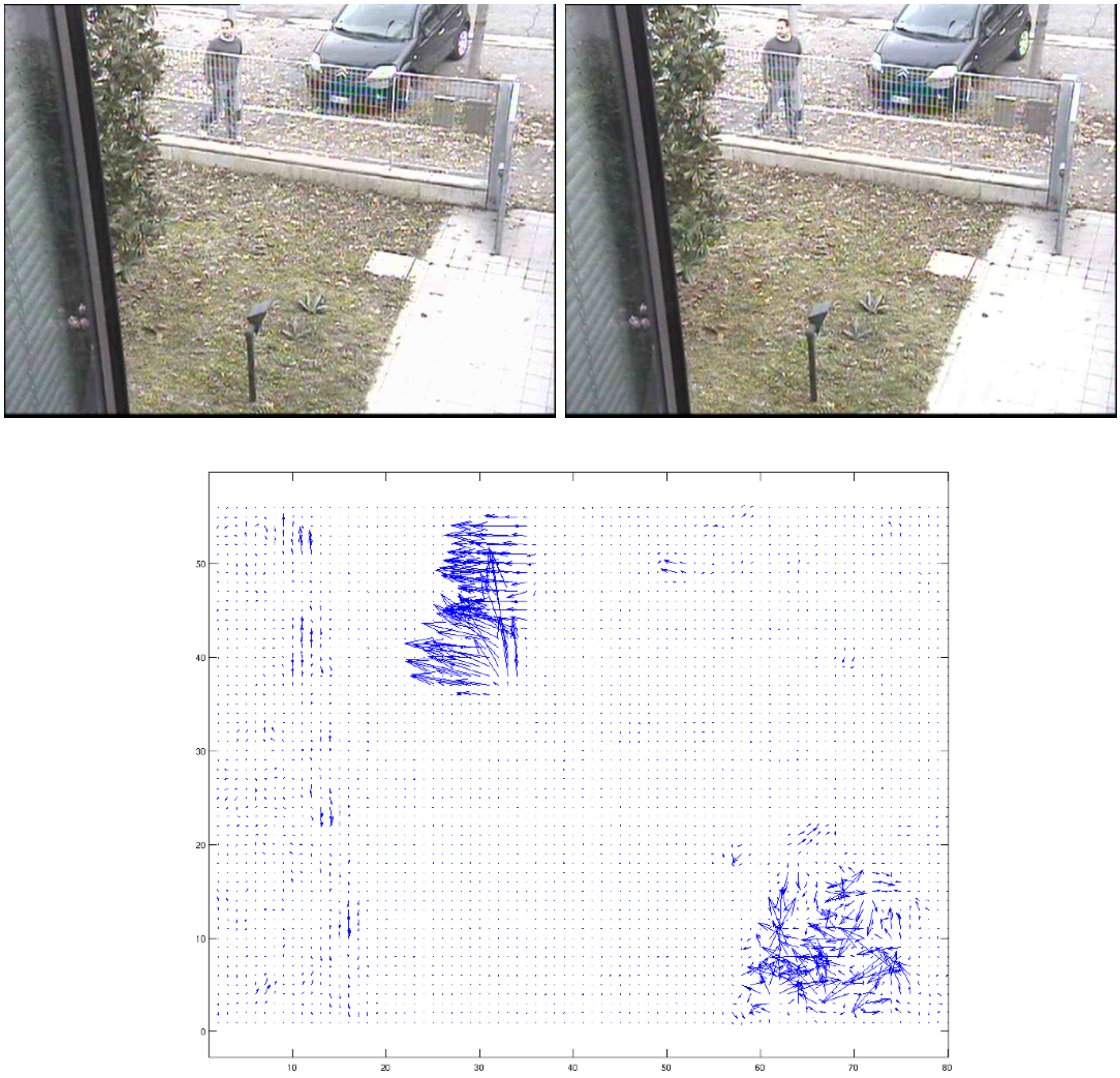


Figure 2.1: An example of motion segmentation induced by the computation of the optical flow. Top row: two consecutive images are extracted from a sequence. Below, the estimated motion vectors visually identify the moving person, also capturing the direction of motion.

2.1.1 Background subtraction

When the camera is still (and thus the background is static), the typical approach for discriminating moving objects of a frame I_t is called *change detection* [ea05]. In the most simple formulation, it consists in comparing the current frame against a background model [ea98a, Gea98, HHD98, ea98b] with the so-called *background subtraction* method[1]:

$$\Delta_t(i,j) =\mid I_t(i,j) - B(i,j) \mid \tag{2.1}$$

The pixel classification takes place by thresholding the map of changes (see Figure 2.2):

$$M_t(i,j) = \begin{cases} 1 & \text{if } \mid \Delta_t(i,j) \mid > \tau \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

The role of the background is fundamental within motion-based analysis methods, con-

(a) (b) (c)

Figure 2.2: A reference frame (a) represents the background in the change detection method: the current scene (b) is subtracted from it to detect the moving foreground (c).

cerning in particular the robustness to cope with

- local and global illumination changes (shadows and highlights);
- static background variations occurring after the modeling phase (object added or removed) or multiple backgrounds (due, for example, to waving trees).

[1]Note that this formulation applies to the case of unimodal backgrounds (see Sec. 2.1.2), where the background is an image. When the background model increases in complexity, the definition of background subtraction is too relate to the specific properties of the model and the modality can not be defined in a unique way.

Change detection methods for motion detection differ from each other mostly in the way the background model is built and updated: a review of proposed solutions can be found in [Pic04]. They can be roughly divided into statistical methods, mainly based on gaussian models [SG02, WADP97, PMTH01] , kernel-based density estimations [EHD00, BCD04], also the so called eigenbackgrounds [RRG+04].
It is finally worth mentioning that a good amount of work in literature has been devoted to address the problem of shadows removal. In [PMTC03] the authors discuss a taxonomy of shadows detection algorithms, highlighting a selection of the most interesting approaches [PMC+00, HHD99, SMO99, CGN+01]. The reference is more specifically to the field of video-surveillance applications, thus there is a particular attention to the detection of moving shadows. In the following we discuss the most popular approaches, while briefly analyzing the state of the art.

2.1.2 Unimodal backgrounds

When the background model consists of an image, then it is known as *unimodal background*. The simplest way to obtain such a model is by temporally averaging a sequence of static images, i.e. frames without foreground moving objects

$$B = \frac{1}{N} \sum_{t=1}^{N} I_t. \tag{2.3}$$

Closely related to this approach, in [YL92] the authors proposed a method based on the observation that the median value of a pixels sequence is far more robust than the mean value to the presence of noise and small (short) dynamic events. However, in real systems one cannot rely on the availability of static sequences when needed. Also, such naive approaches are extremely sensitive to changes of dynamic scenes due to lighting or extraneous events. This naturally calls for methods able to dismiss moving structures from the modeling operations.
A compromise between accuracy and computational efficiency is the so-called *running average*, based on a simple incremental strategy combined with the output of the change detection process, so that pixels laying in moving areas are discarded: starting from an average of the first N frames or, alternatively, an initial empty background (in both cases we call it B_0), the estimate at time $t \geq 1$ is:

$$B_t(i,j) = \begin{cases} \alpha B_{t-1}(i,j) + (1-\alpha) I_t & \text{if (i,j) is classified as static pixel} \\ B_{t-1}(i,j) & \text{otherwise} \end{cases} \tag{2.4}$$

where $0 \leq \alpha \leq 1$ controls how fast new structures are included in the background and the pixel classification is performed using the change detection discussed in the previous section. The procedure ends when every pixel is assigned with a value but it is common

practice to periodically update the reference frame following the same procedure. An alternative is to fit the distribution of each pixel with a gaussian (μ, σ), obtaining the so-called *running gaussian average* adopted, for instance, in [WADP97]. The updating mechanisms become:

$$\mu_t = \alpha I_{t-1} + (1 - \alpha)\mu_{t-1} \tag{2.5}$$

$$\sigma_t^2 = \alpha(I_{t-1} - \mu_{t-1})^2 + (1 - \alpha)\sigma_{t-1}^2. \tag{2.6}$$

Even if very effective in many cases thanks to the computational efficiency which makes it suitable for real-time applications, all these methods do not suffice in presence of more difficult conditions, due to the assumption on unimodal distribution for modeling the trend of pixel values. An alternative is thus the use of multimodal background models, discussed in the next section.

Figure 2.3: Pixels temporal evolutions with different characteristics: the blue arrow indicates a road pixel showing a rather stable value only temporary affected by a significant variation due to car passage. An example of multiple background, instead, is provided by the red pixel, being an element of tree or road alternatively, due to the wind influence.

2.1.3 Multimodal backgrounds

The problem of background modeling becomes more complicated in outdoor environments where multimodal (or multiple) backgrounds are frequent. The concept of multiple background is visually illustrated in Figure 2.3: a pixel belonging to the *static background* model presents a rather stable appearance, even if it might be temporary affected by strong variations due to the dynamic in the scene. It is the case of the blue pixel that, being a point of the road, is temporary occluded by the car passing by. Moreover, the pixel could change appearance after that a stable change in the environment occurred, event that the background model should be able to recognize. Instead, let us consider the red pixel which falls on the tree. Due to the wind influence, such pixel assumes alternatively values related to the leaves or the road, so that its temporal evolution appears as in Figure 2.3.

Coping such difficulties requires more refined, and thus computationally costly, solutions. Considering complex approaches, many works in literature are based on parametrizing grey level changes over a time space[Gea98, HHD98, ea98b]. These methods have been widely incorporated in algorithm with Bayesian framework [LHE03], mean-shift analysis [PT03] and region-based information [CBM02]. These kind of techniques, together with the ones based on probability estimation (see [EHD00, HNR84] as examples), in spite of their accuracy in terms of resulting segmentation, are not suitable for an applied video surveillance framework, where the computational efficiency plays a fundamental role.

An interesting trade-off between the discriminative power of the complex approaches discussed above and the computational efficiency of unimodal models can be found in [KCHD05] where a real-time algorithm for foreground-background segmentation based on the use of codebooks [Cea04] is proposed. The codebooks provide the system with the capability of representing each pixel with (possibly) multiple descriptors and capture structural background variations due to periodic-like motion over a long period of time under limited memory. A comparison between unimodal background modeling and codebook-based approach is shown is Figure 2.4: on the left, the segmentation obtained by the incremental method cannot deal with multiple backgrounds, while on the right the foreground object is correctly detected.

In this specific case, the information collected to describe the dynamics of each pixel allows us also to distinguish between different levels of background: traditional approaches consider static variations of the scene (objects added or removed) as variations with respect to the reference image, in other words as motion. But if we consider a structure added to the scene and stable over a proper interval of time, it should be interpreted as a further layer of the static background. A visual representation of this situation is visible in Figure 2.5.

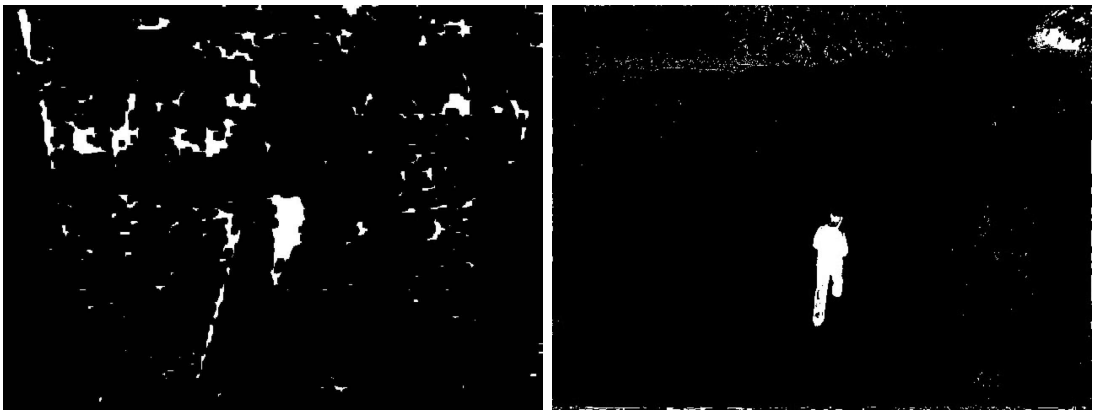


Figure 2.4: The comparison between foreground segmentation obtained with incremental background and codebook methods shows how the second better deal with multiple backgrounds.

Figure 2.5: Different layers of static background: an object (magenta) is added to the scene, becoming a stable variation.

2.2 Object tracking

A fundamental tool to recover basic space-time description of a motion event is object or feature *tracking*, whose main goal is to aggregate temporally correlated information. It consists of two subproblems:

- **Trajectory initialization** typically rely on motion-based image segmentation but several recent approaches have started to explore the possibilities of combining tracking with detection [LSG07];
- **Target following** is usually addressed by classical tracking approaches strengthened by the employment of prediction filters.

The literature on object tracking is very rich. Here we focus on methods that have been proposed in the context of real-world video-surveillance systems. Dynamic filters have been thoroughly studied in this application domain. The Kalman filter [WB] is an efficient recursive filter which estimates the state of a dynamic system from a series of incomplete and noisy measurements but it is limited to a linear assumption. However, most non-trivial systems are non linear: in such cases the Extended Kalman filter [Gel96], the Unscented Kalman filter and the Particle filter [IB98] may help to address the non-linearity issue.
In particular, particle filters, also known as Sequential Monte Carlo methods (SMC), are sophisticated model estimation techniques based on simulation and represent instances of more powerful dynamical filters [AMGC02, WVDM00]. They are usually used to estimate Bayesian models and the advantage with respect to the Kalman tools is that, with sufficient samples, they approach the Bayesian optimal estimate, achieving a higher level of accuracy. The approaches can also be combined by using a version of the Kalman filter as a proposal distribution for the particle filter.
For computational efficiency reasons, Kalman-based tracking is often preferred to more sophisticated yet costly approaches. These motion-based methods often suffer from the absence of appearance models, thus they are not robust to static events (e.g., a person standing for some time).
A complementary approach is appearance-based tracking [FT97, KS00, STB$^+$06]. Mean-shift [CRM00] is a very popular strategy, belonging to the family of kernel-based approaches to the problem: it is a nonparametric estimator of density gradient employed in the joint, spatial-range domain of gray level and color images for discontinuity preserving filtering and image segmentation. However, its original formulation leaves open problems as target initialization and occlusions handling. Another major drawback is its computational cost. Many variants have been proposed to cope with these difficulties: in [YDD05] the authors introduce a new discriminative similarity measure in spatial-feature spaces, achieving good results; a very recent attempt to exploit mean-shift properties is presented in [ZYS09] where the technique is adapted to track sift features. In Camshift (Continuously Adaptive Mean

Shift) [Bra98] the mean-shift algorithm is modified to deal with dynamically changing color probability distributions and applied to the problem of tracking faces.

Since motion and appearance approaches can be regarded as complementary, the literature proposes various methods combining appearance-based methods (and mean-shift in particular) with dynamic filters. Among different techniques available, particle filters are often adopted for their ability to maintain multiple hypothesis about target properties. [SWTO04] integrates the advantages of the two methods, showing that filter-based tracking (in the specific case of hand) benefits from introducing appearance properties; a color histogram is integrated in a particle filter framework in [NKMVG03]. In [PYL05] the authors propose to plug a color histogram, describing the visual appearance of a target, into a Kalman-based framework so that the appearance model is filtered and contributes to estimate the new object model. A possible drawback of this choice is that appearance information may be very unstable when illumination changes, intersections occur, or temporary occlusions (total or partial) take place. This would introduce uncertainty in the whole state system.

2.3 Representation and comparison of temporal series

When dealing with trajectories representation, several authors adopted approaches in which the temporal component of the observed behaviors is kept explicit [Lia05], therefore relying on methods for time-series analysis. In what follows, instead of considering representations that explicitly refer to 2D or 3D trajectories, we focus on methods that do not make any assumption on the size of each instantaneous measurement.
[MT08] reports a review of popular methods to pre-process trajectories to plug them into learning frameworks. In particular, methods for normalizing trajectories or reducing their dimensionality are discussed. Trajectories normalization ensures that they all have the same length. Zero-padding [HXF$^+$06b] and track extensions [Cuc05] are very simple techniques where the common length is automatically chosen on a training set, with major drawback due to computational issues. Alternatively, the equal length can be fixed *a-priori*: each sequence is then adapted to it by resampling [ME02, HXF$^+$07, LCST06] steps sometimes coupled with smoothing [LHH06] to attenuate the noise effects.
Even if the methods mentioned so far represent a simple and intuitive class of techniques, they are not tailored for on-line analysis, being typically applied on entire tracks. Notice, moreover, that there is not guarantee that the intrinsic properties of data are maintained: as a simple example, when subsampling two sequences corresponding to people walking and running the information related to the velocity could be lost, if not explicitly included into the description.
More related to our work, another main class of approaches relies on reducing the dimensionality of the trajectories so that they can be more easily handled, both from the

standpoint of computational issues and the capability of learning methods to extract knowledge from the data. Each trajectory (or group of coherent trajectories) is assumed to be generated by a model, finding the set of parameters best describing it. Vector quantization is a very popular and simple technique to select a sub-set of prototype able to symbolize the whole data-set [BS98, SG02, ZSV04].
The curve-based approach is based on fitting a function on the trajectories and mapping it into the parameters space [PCV00, YF05]: the complexity of the functional should be dictated by the intrinsic complexity of the data-set. When plugging scale-based features (as wavelets) into this approach [KN05, NK06, LHH06] a multi-resolution representation of the trajectory is achieved: a smoothing effect on each datum is inherently obtained, whose amount can be controlled by choosing the appropriate wavelet level.
Other two main popular approaches are based on the eigenvectors of a training set, the well-known principal component analysis (PCA) [BAT05, BQKS05] and the so-called spectral methods [XG05, HXF$^+$07, Por04, AMP06]. In the first one, a new space is built which is spanned by the largest eigenvectors of the training set; the trajectories are then projected onto the subspace accounting for most of the signal and discard low-variance direction. The spectral-based approach, instead, relies on the computation of a particular matrix, the *laplacian* [Chu97], which is decomposed to find its K biggest eigenvectors. Such vectors compose the new trajectories descriptors. Our approach shares similarities with this techniques.
The representation based on natural languages has been inherited from text retrieval and widely applied to application based on motion analysis [HXF$^+$06a, SG00]. An alphabet of symbols is defined on the input space, namely the space of the trajectory elements, and with respect to it a trajectory is represented, assuming the string-form. The alphabet is built by partitioning the input space in states and associating a symbol to each one of them. Finally, a trajectory is translated into string by associating their elements to the states they belong to and replacing each element with the corresponding state symbol.
Also, it is worth mentioning some examples of how traditional approaches can be applied from slightly different view-point. Hidden Markov Models (HMM) [Rab89] are highly popular methods to model a group of trajectories that are assumed to be generated by a common underlying hidden stochastic process. In [JKH04, JST07] the authors proposed a promising approach based on modeling each single trajectory with an HMM (as opposite to the traditional use that modeled group of trajectories). The space of the final fixed-length representation is that of parameters of the HMMs. Analogously, the Kalman filter [WB] and related ARMA models have been adopted for the same purpose [DCWS03, XY04].

For what concerns more specifically the issue of learning from examples, an important issue is what similarity measure using to extract meaningful information: [RL08] offers a complete survey to such a topic applied to sequential data. If sequences of different lengths must be compared in the native input space, Euclidean distance and derivative measures (see [BI05] as an example) performs poorly. They can be alternatively coupled

with methods for *temporal alignment*: the very popular Dynamic Time Warping [RJ93] aims at minimizing the distance between matched points of two trajectories, goal for which many other variants have been proposed. We cite here the method based on Longest Common Subsequence (LCSS) [VGK02, BSK04], more robust to noise, and the work in [PF06] which does not require the entire sequence and it is thus more appropriate for on-line and predictive analysis. The Hausdorff distance is designed to cope with data of different length and has thus ben applied to the problem of comparing trajectories [JJS04]. However, a major drawback is that it disregards the temporal component, so an adaptation was required to be tailored to the problem [AOP06]. In general, when comparing temporal series described with a non-parametric approach (in other words, using raw data) a number of mathematical distance and similarity measures have been proposed in the literature (see [Lia05] for an almost complete survey), although can be easily applied only when the series have the same lengths. In the opposite case, some adaptation to the comparison of sequences of different lengths must be implemented.
When the trajectories are mapped in some different space, instead, the distance or similarity measure strictly depends on such transformation. In some cases, one of the traditional distances cited above can be directly applied to the new representation (e.g. when the representation is composed of the parameters of a fitting function). Otherwise, in more complex spaces a specific measure must be adopted. It is the case of statistical-based representations [JKH04, JST07, XY04] or string-based descriptions. In the latter case, a complete survey can be found in [TC04] where a number of kernels to compare sequences of symbols are discussed.

2.4 Learning from temporal series for behavior modeling

The study and the understanding of human activities from videos has been widely addressed in the last decades (see for instance [SG00, Gea98, IB00, PCV00, PMF08, AC08] and references therein). With the availability of an enormous amount of installed video surveillance cameras a huge amount of video data are daily acquired, becoming more and more difficult to be handled by human operators. This justifies the growing need for computational methods able to assist the user, suggesting where to focus attention.

A priori knowledge on the analysed environment and its dynamics may be exploited in different ways. A first example, widely explored in the literature, refers to stochastic grammars (see, for instance, [IB00]). In certain application domains (e.g., traffic control) the amount of structures contained in the expected events may be profitably used to model the dynamics of the scene. For instance, in [MCB+01] the results obtained by a robust low-level processing module are associated to context information to identify possible abnormal

behaviors in aerial images.

In the case of behavior analysis, if the available data are labeled, i.e., to each one of them we may associate a label of a known behavior, the use of state-of-the-art machine learning algorithms lead to effective behavior categorization methods. On this respect we mention the work by Pittore et al. [PCV00], where the available data are mapped in the parameter space induced by the fit of each trajectory with B splines, then Support Vector Machines (SVMs) are applied. Otherwise, the sequential structure of trajectories may be captured by training a well designed Hidden Markov Models (HMM) [Rab89] or other similar dynamical systems (see, for instance, [BKS07] and references therein).

However, the huge amount of available data, for which a manual labeling is an unfeasible solution, calls for some form of unsupervised data analysis, since: an analysis of the related literature makes it clear that unsupervised methods for behavior understanding or, more in general, to time series analysis are quite limited - see [JST07, JCN06, ME02]. In [Lia05] the interested reader can find a rather complete review.

The general goal of unsupervised approaches applied on temporal data in the field of video-surveillance is to model normal behaviors from (possibly big) sets of unlabeled observations. Many successful methods have been proposed to discover classes of similar pattern in a data set. Unfortunately most of them are not tailored for clustering sequential data, such as temporal series. In fact, while analyzing temporal data, the *temporal dimension* usually induces specific, inherent structure to the sequence of feature vectors that is likely to be disregarded by traditional clustering methods. Furthermore, sequences of different lengths cannot be compared in a unique way.

Among the first contributions to this topic is the influential work by Stauffer and his co-workers (see, for instance, [SG00]), based on learning from data a code-book derived from data quantization. Co-occurrence of motion patterns in the analysed trajectories is evaluated, through the use of a hierarchical classification module. More recently [HXF$^+$06a] propose a pipeline based on k-means to track objects, represent trajectories with respect to an estimated code-book, and cluster them. Finally, a Bayes method is applied to detect anomalies. Experimental analysis is based on both synthetic and real traffic sequences. In [PMF08] normal behaviors are associated to one class only. Then a one-class SVM is used to model the class of normal trajectories, against which identify abnormal behaviors. The representation is a sequence of 2D coordinates and the fixed length property is restored by means of trajectory sub-sampling. Again, the reference application is outdoor traffic control analysis. [AC08] adopt a multi-feature representation of each trajectory based on a cumulative approach: each measurement at time t refers to all previous observations. Clustering is performed with mean-shift. A trajectory is labeled as normal or anomalous, according to the trajectories density among the clusters.

More specific to our problem is previous work on temporal series clustering. In this context

the literature addressed the problem from two distinct viewpoints (see, for instance, [JST07, Lia05, JCN06] and references therein). The first approach relies on modifying suitable existing algorithms for clustering static data in such a way that time series data can be handled. The rationale of the second approach is to convert temporal series data into the form of static data so that the existing algorithms for clustering static data can be directly used. An interesting alternative to these two mainstream approaches, is to tackle directly the problem of building suitable similarity/kernel functions that encompass dynamical properties of the events. In [VSV07], the authors provide a unifying theoretical framework to study and design different kernels based on dynamical systems, which can be used for behavioral analysis (through the so called ARMA models [DCWS03]), diffusion processes, graphs-based systems. A number of efficient methods for computing such kernels are also discussed, making the paper a valuable starting point for a more detailed study of this approach.

A rather complete account of the open issues related to events classification in an unsupervised setting is reported in [MT08].

To conclude, a challenging aspect of applying unsupervised learning to real world complex scenarios consists of evaluating the quality of the obtained clusters both with respect of the initial set of data and with respect to possible future observations. As reported in [GB03, BA03, HBV01] most of the work is devoted to estimate an appropriate model (usually related to the choice of the number of clusters to consider) by comparing the estimated membership against a ground truth, when available. Otherwise, when several instances have been computed (e.g. for different values of the parameters) an analysis based on *quality measures*, or *quality indices*, is used to select the most promising computation. A rather complete account of the open issues related to events classification in an unsupervised setting is reported in [MT09].

Chapter 3

Low-level video processing

In this chapter we address the video processing stage of the pipeline. The focus of the discussion is on the details of our modules for low-level analysis (Sec. 3.2) and object tracking (Sec. 3.3). In the latter, a joint model of motion and appearance information is adopted: the problem of preserving the identities of interesting targets is approached using graph structures, naturally embedding different levels of complexity depending on the properties of the scene.
The last part (Sec. 3.4) presents the experimental analysis, in which we compare our system against a selection of popular approaches, using an annotation on the video content based on the use of a set of source-sink regions.

3.1 Introduction

The apparent motion of objects in the image plane is a strong visual cue to understand the semantics of 3D motion and usually represents the first goal of behavior analysis models. This thesis considers scenarios acquired by a single static camera, and deals with the information loss caused by the image formation process by looking for rich data representations (Chapter 4).
At the lowest processing level the apparent motion patterns are captured by means of change detection and object tracking. We start-off from the literature discussed in Chapter 2, adopting a rather standard motion segmentation while proposing an effective object tracking method, able to perform in real-time.
In the following, the two modules are discussed in details.

3.2 Low-level analysis

We now describe or approach to extract low level information from videos (see Fig. 3.1). Let us first observe that in our reference application domain real time processing is a strong

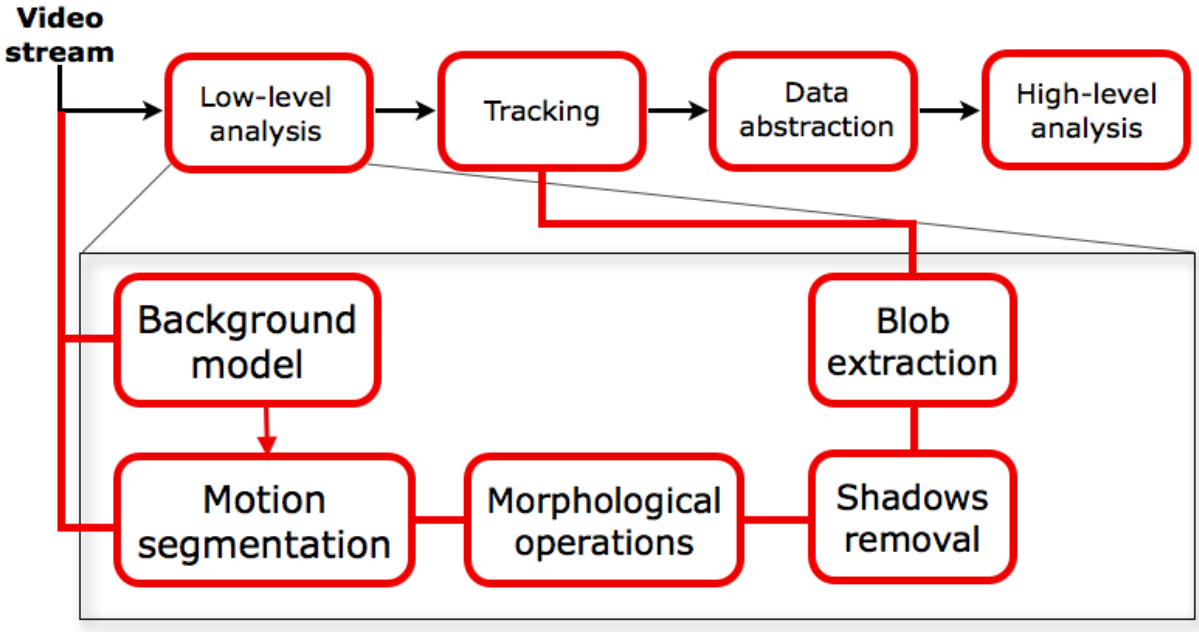


Figure 3.1: The main steps followed during the low-level analysis of each frame in a video.

requirement: this significantly reduces the set of suitable methods for motion segmentation. Second, we consider the fact we will mainly focus on indoor scenes, thus our choice falls on unimodal descriptions, and the running average method in particular.
Once a first segmentation is obtained we further process it by applying morphological operations to better approximate the shape of blobs, more specifically the *erosion* and the *dilation*: given a local patch Q around each pixel (i, j) of the binary map I_{in}, the corresponding pixel in the output map I_{out} is defined as

$$I_{out}(i,j) = \min_{(i',j')\in Q} I_{in}(i+i', j+j') \tag{3.1}$$

for the erosion, and

$$I_{out}(i,j) = \max_{(i',j')\in Q} I_{in}(i+i', j+j') \tag{3.2}$$

for the dilation.
Finally we apply a procedure to remove shadows [SFGK02], performed on the YUV color

space. YUV is the native space of videos, thus its adoption avoids the computational costs of conversions into different color spaces (as RGB or HSV).
When the final motion-based segmentation is achieved, the connected components of the binary map, called *blobs* or *targets*, can be identified and described by an appropriate set of low-level features. The next stage of the pipeline aims at correlating this information along the sequence to provide a temporal description for the dynamic evolution of the target.

3.3 A combined motion and appearance tracking

Motion-based trackers often rely on the use of dynamic filters that help to increase precision and performance. A very common choice is to adopt a Kalman filter with a hidden state

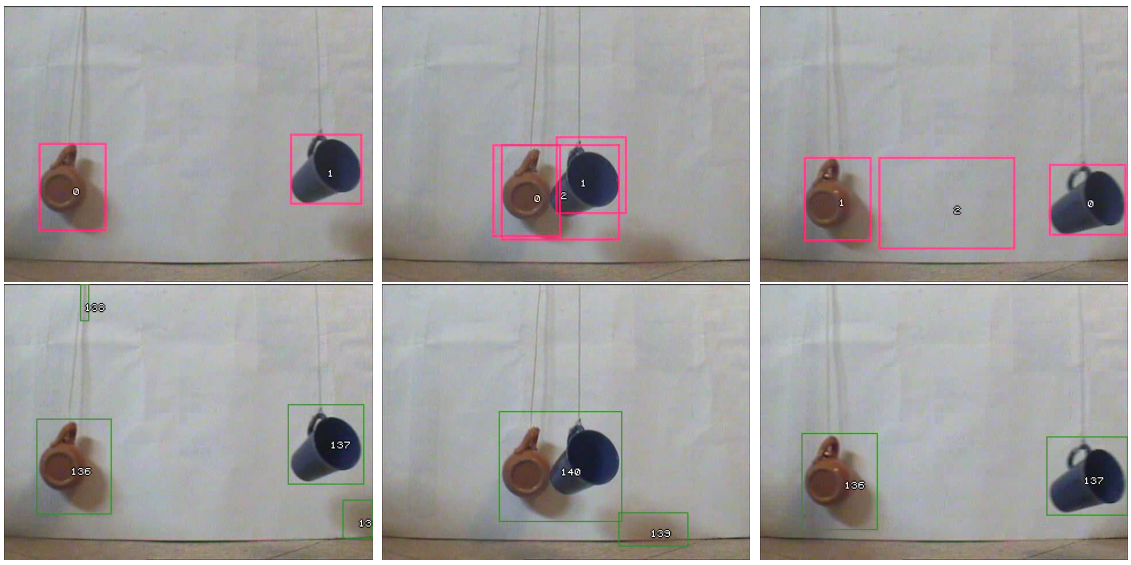


Figure 3.2: When based on motion information only, tracking might be unreliable. The Kalman-based approach let the system learn over time the motion model so that it is able to predict the future position. It is in some cases not sufficient to successfully address the tracking problem (first row), as the motion model is deceived by the change of directions. Adding appearance information, the system is able cope with the problem (second row).

$\mathbf{x}_t$ made of target position and velocity, while the noisy observation includes position only. This model was proved quite effective in moderately simple environments. As the observed scenario becomes more complex, the ability of predicting the next state of the system exploiting the model built over time might be inappropriate, leading to incorrect data association. Figure 3.2 shows an example of wrong association: in the video sequence

(belonging to a toy data set acquired in our labs), two cups get close to each other and then go back to the original positions. When the two objects start getting apart, the motion-based tracker will predict the new associations according to its state model, therefore it will wrongly assume that the objects trajectory did not change (Fig. 3.2, top row). In this simple case objects appearance would help and leads to correct matching.

A possible way to implement this simple intuition is to model an appearance description

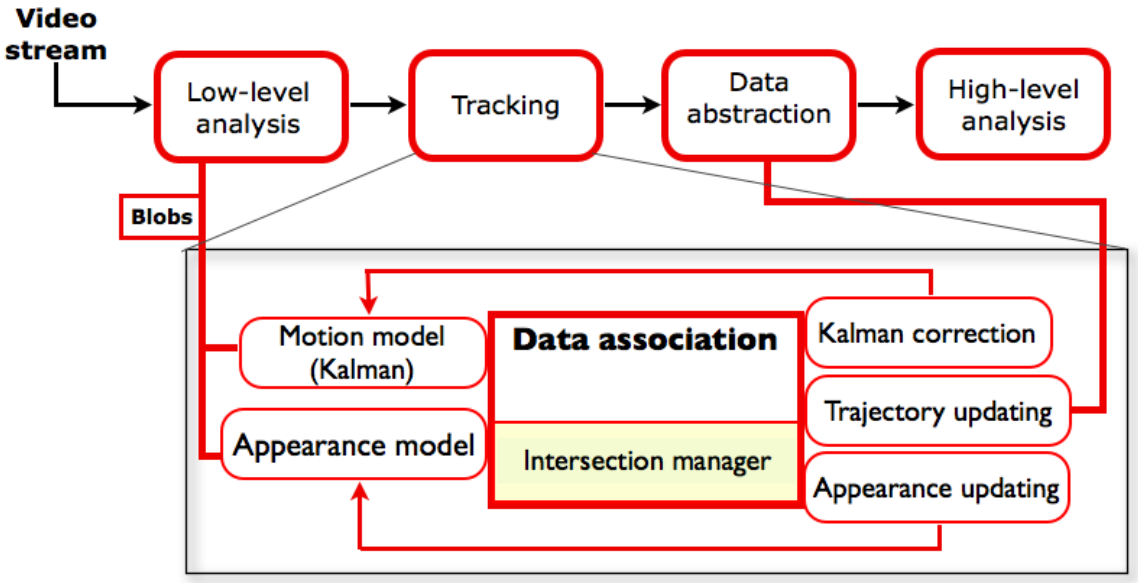


Figure 3.3: An overview of our system: motion and appearance information are processed in parallel and jointly exploited to reach a robust behaviour. Data association is based on a matching procedure, followed by a stage for managing complex events.

within the filter state model, similarly to [SWTO04] in a particle filter setting. However, all the methods briefly discussed in the previous section fully integrate appearance into a motion tracking model.

The peculiarity of the approach we propose is different, since motion and appearance information is measured, stored and updated independently, but together participate to address the tracking task as we deal with more complex events occlusions and intersections. Therefore our main contribution is a simple solution able to perform real-time as many motion-based approaches do, and to resort to appearance only when necessary, with no loss in computational efficiency.

Figure 3.3 shows a visual representation of our tracking system [NDLO09]. The pipeline is designed as a cascade of simple modules that are combined to reach a robust final result, rather than based on more complex mechanisms that achieve good results at the price of

high computational costs.
At time t, motion and appearance information describing a certain moving object or *target* O_i follow a similar process, being (1) measured and modeled with a feature vector $\mathcal{M}_i^t$, that we refer to as the*target description*, (2) associated to previously seen targets by comparing them with predicted or estimated models (*identity association*), and finally (3) updated for the next evaluation time $t+1$ (*Kalman correction and appearance updating*). Identity association includes a procedure to solve intersections and occlusions, which allows us to cope with the tracking task in presence of more complicated scenarios.
When target O_i exits from the scene its trajectory is stored as a trajectory $\{\mathcal{M}_i^t\}_{t=1...k}$ of instantaneous observations.

3.3.1 Target description

At time t, the instantaneous observation of a moving object or *target* O_i, is a vector $\mathcal{M}_i^t = [TH_i^t, \mathbf{P}_i^t, S_i^t, \mathbf{V}_i^t, \mathbf{CH}_i^t, W_i^t, H_i^t]$ where

- $TH_i^t \in \mathbb{R}$ is the frame timestamp at time t;
- $\mathbf{P}_i^t = [x_i^t, y_i^t] \in \mathbb{R}^2$ is the position of the target centroid on the image plane;
- $S_i^t \in \mathbb{R}$ is the spatial occupancy of the target, $S_i^t = \sum_{(i,j)\in CC_i^t} B^t(i,j)$, where CC_i^t is the connected component of the binary map at time t corresponding to O_i;
- $\mathbf{V}_i^t = [Vx_i^t, Vy_i^t] \in \mathbb{R}^2$ is the apparent velocity vector estimated by Kalman;
- $\mathbf{CH}_i^t = \sum_u C \sum_{j=1}^n K(||\frac{\mathbf{P}_i^t - \mathbf{P}_j^t}{h}||^2)\delta(b(\mathbf{P}_j^t) - u) \in \mathbb{R}^M$ is a weighted color histogram [CRM00] where
  - u is a candidate color from a quantization in M states of the $\{R, G, B\}$-space
  - C is a normalization constant
  - $K(x) = \frac{1}{2\pi} e^{-\frac{1}{2}||x||^2}$is the kernel profile with radius h (which depends on blob size)
  - δ is the Kronecker delta function
  - $b : R \rightarrow 1...M$ maps pixels locations into histogram bins, depending on their color
  - $\mathbf{P}_j^t$ is a pixel location lying in the spatial extent of target O_i.
- $W_i^t \in \mathbb{R}$ and $H_i^t \in \mathbb{R}$ are, respectively, width and height of the bounding box of CC_i^t.

At each time instant $t > 1$, the motion information is predicted and updated accordingly to the well-known Kalman filter pipeline.
Concerning the the appearance model, it is updated if not significant changes occurred - thus avoiding the corruption with incorrect color information due to tracking temporary failures - according to the following update rule: given the model at time $t-1$, $\mathbf{CH}_i^{t-1}$, the new one is computed as

$$\mathbf{CH}_i^t = \alpha \mathbf{CH}_i^{t-1} + (1-\alpha)\mathbf{CH}_{OBS}^t \tag{3.3}$$

if $dist(\mathbf{CH}_i^{t-1}, \mathbf{CH}_{OBS}^t) \leq \tau$, where $\mathbf{CH}_{OBS}^t$ is the color histogram measured at the current target location, α is a learning factor, *dist* is a suitable distance between histograms (e.g., Bhattacharyya distance) and τ is an appropriate threshold (the choice of α and τ is dictated by the environment and can be set in a system calibration phase). The above updating rule allows to control how quickly the system learns the color model.

3.3.2 Graph-based data association

We now briefly sketch the data association procedure, core of the tracking system. Data association (Fig.3.4 reports a simple example) is based on graphs using two distinct sets of nodes:

previously observed targets $\{\mathbf{T}_k\}_{k=1}^{N_T}$

current observations $\{\mathbf{OBS}_j\}_{j=1}^{N_{OBS}}$.

Each edge in the graph is directed from a target to an observation and models a match between them.
Following an all-vs-all matching configuration, based on computing a *distance* vector for each pair $(\mathbf{T}_k, \mathbf{OBS}_j)$, we obtain a graph which is

- *Bipartite*, because each edge goes from a target to an observation, which are elements of the distinct sets named above;
- *Complete*, as in first instance we try the match between each possible pair (target, observation);
- *Weighted*, since edges are labelled with the distance vector.

Distances, used as edge weights, measure the goodness of a match. To compare a target $\mathbf{T}_k$ against an observation $\mathbf{OBS}_j$, we build a distance vector $[D_{kj}, R_{jk}, I_{kj}, B_{kj}]$, where (1) D is the Euclidean distance of the mass centers, (2) R is the ratio between sizes, (3) I

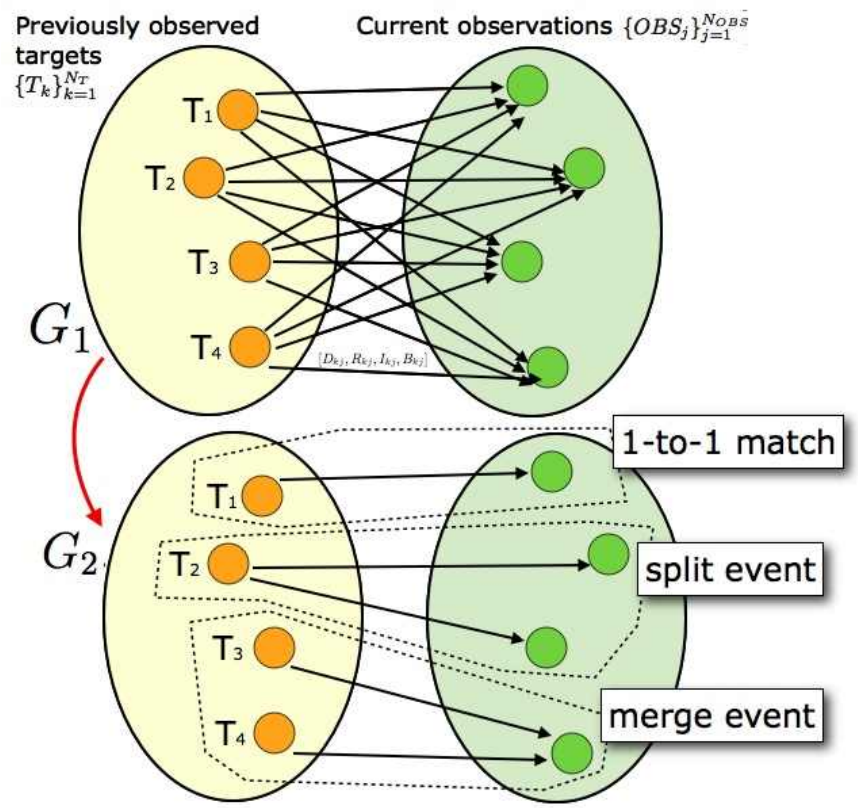


Figure 3.4: An example of data association. Above, a graph G_1 is built on targets and observations nodes with an all-vs-all matching strategy, each edge is weighted with a vector of distances between features. Below, a simplification of G_1 is obtained by rejecting edges for which a matching condition does not hold. The new graph, G_2, includes different types of matches, representing simple associations (1-to-1 match) or more complex events (split or merge).

represents the intersection of the spatial extents, and (4) B is the Bhattacharyya distance between the color descriptors. By computing the distance vector between all the possible pairs of elements in the two sets of node, a first matching graph G_1 is built.
Since edges are oriented from targets to observations, we can define

- The *outgoingflow* $OUT_F(\mathbf{T}_k)$ of a target $\mathbf{T}_k$ as the number of edges exiting from the correspondent node
- The *incoming flow* $IN_F(\mathbf{OBS}_j)$ of an observation $\mathbf{OBS}_j$ as the number of edges entering the node.

A first matching evaluation is based on a condition that considers D, R and I,

$$(D < \tau_D) \;\&\&\; (R > \tau_R) \;\&\&\; (I > \tau_I) \tag{3.4}$$

where τ_D, τ_R and τ_I are appropriately chosen thresholds (the values of the thresholds are fixed during a calibration phase, by observing the scene dynamics and estimating the average expected displacements of targets over subsequent frames). This procedure provides a set of potential matches: note that in our framework it corresponds to delete edges of graph G_1, for which the condition does not hold, obtaining a second graph G_2. We distinguish four possible match configurations that differ with respect to the flows:

1-to-1 match ($OUT_F(\mathbf{T}_k = 1$ and $IN_F(\mathbf{OBS}_j = 1$) - In this case if the color descriptors are coherent we conclude that $\mathbf{OBS}_j$ is a new realization of the previously observed target $\mathbf{T}_k$

N-to-1 match (a set of targets $\{\mathbf{T}_i\}_{i=1}^{k}$ match with the same observation $\mathbf{OBS}_j$, so that $OUT_F(\mathbf{T}_i = 1$ for each $i = 1...k$ and $IN_F(\mathbf{OBS}_j = k$) - in this case that a *merge event* might have happened, that is different targets moved in a limited spatial range so that during segmentation only one connected component has been detected. The observation gives raise to a new target but the history stores the merge event together with the identifier of the targets involved. A new identifier is univocally associated to a new target, so it is sufficient to completely determine it. However, before concluding that a merge event occurred, we compare the color descriptors to test whether one of the targets is much more similar to the observation than others. In this case, we force the association with the correct target: this happens when an **occlusion** occurs.

1-to-N match (a target $\mathbf{T}_k$ matches with a set of observations $\{\mathbf{OBS}_i\}_{i=1}^{j}$, so that $OUT_F(\mathbf{T}_k = j$ and $IN_F(\mathbf{OBS}_i = 1$ for each $i = 1...j)$ - Similarly as above, we look for possible match much more robust that the others by comparing the appearance properties, and, in case, correct the association. If such match is not found, we conclude that a *split event* occurred, that is a group of targets moving coherently differentiate their dynamics. Again, the event is stored in the history.

N-to-N match (a set of targets $\{\mathbf{T}_i\}_{i=1}^{k}$ match with a set of observations $\{\mathbf{OBS}_h\}_{h=1}^{j}$, so that $OUT_F(\mathbf{T}_k > 1$ and $IN_F(\mathbf{OBS}_h > 1$ for $i = 1...k$ and $h = 1...j)$ - We further analyse this complex event and reject the weakest matches until we reach one of the previous cases.

When a target does not take part in any match, we consider it as missing from the current scene, as well as an observation with the same behaviour is treated as a new target just entered in the camera view. The graph-based approach allows us to better deal with more

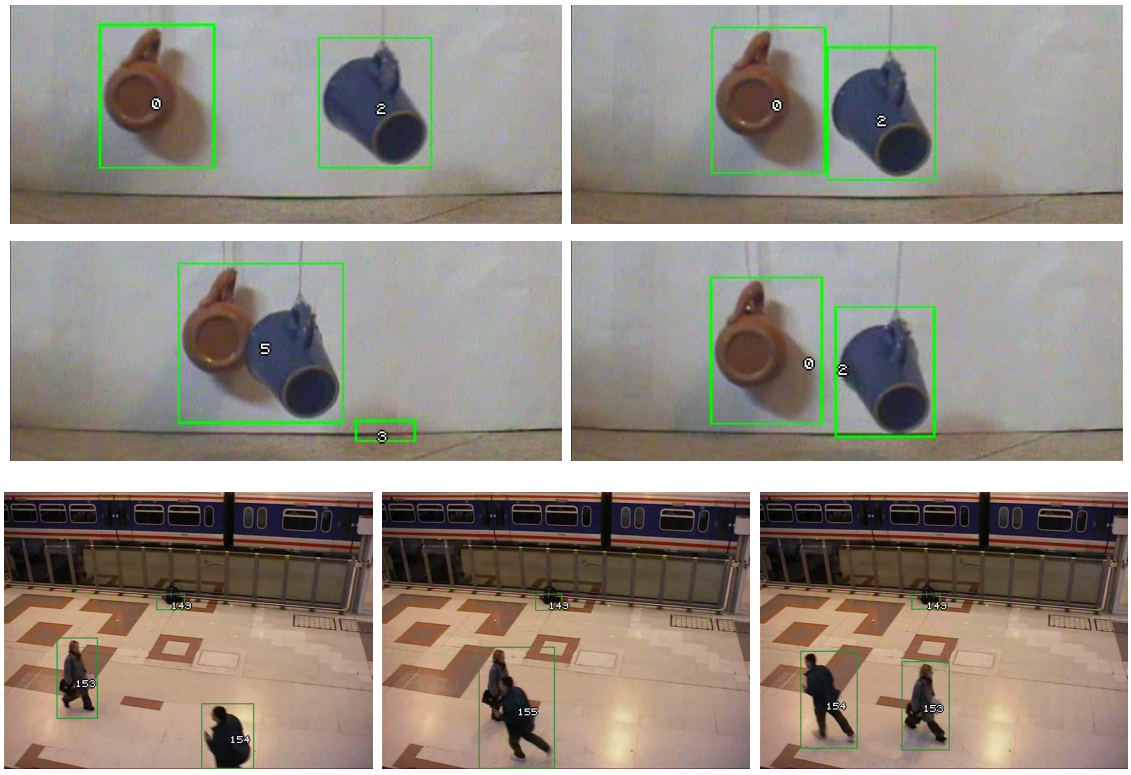


Figure 3.5: Details on tracking results for a controlled sequence (first and second row) and a real scenario (last row): by comparing the appearance model of targets using the Bhattacharyya distance, the system is able to recover after occlusion events, which are typically characterized by sequences of *merge* and *split* events.

complex data associations. *Intersections* and *occlusions* are typically characterized by a merge event followed by a split event: the sequence in Figure 3.5 (first and second) clearly explain the mechanism. The two cups go towards each other and are identified as distinct entities (first row, targets 0 and 2) up to the moment they intersect, so that the binary map includes only one connected component. The latter is interpreted as a new entry in

the targets set (second row on the left, target 5), but the matching procedure is able to recognize the *merge event*: more precisely, it encodes the event by storing the information that targets 0 and 2 become sub-targets of 5 on a dedicated data structure, that we call the *target history*. Later on, when they split, the data association is able to re-associate the correct identity to each target by comparing the color histograms of new observations and sub-targets of target 5 (second row, right). The last row of Figure 3.5 reports an analogous case for a sequence of a real environment.

3.4 Experiments

We test the performance of our algorithmic pipeline on scenarios of variable complexity:

1. The whole CAVIAR[1] data set
2. A selection of videos from the VISOR collection[2] appropriate for the problem under consideration, including highly challenging scenarios
3. The data set of PETS 2006 workshop[3], of which we processed one of the 4 views.

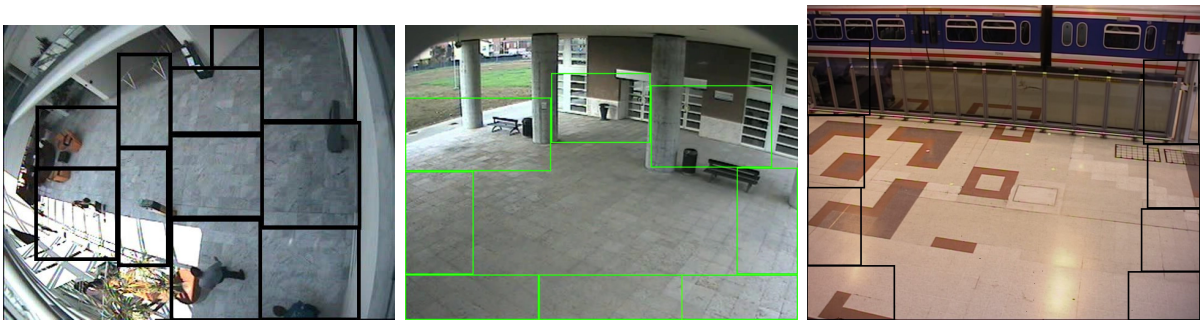


Figure 3.6: Examples of the regions adopted to build the ground truth (see text).

In order to present a coherent quantitative analysis of the results obtained on different data sets we devised a simple ground truth labeling adequate to show the appropriateness of our approach towards our task: we defined for each scenario a set of source and sink regions (see Figure 3.6 for some examples) and built a ground truth that lists, for every dynamic event entry region, entry time instant, exit region and exit time instant.

[1]http://www- prima.inrialpes.fr/PETS04/caviar_data.html
[2]http:www.openvisor.org
[3]http:www.cvg.rdg.ac.uk/PETS2006/index.html

When entry and exit points are few, as for the CAVIAR data set, we enrich the ground-truth with labels of intermediate regions that the target goes through.
The data sets we used are interesting and complex in different aspects: PETS looks at a medium-crowded scenario, where people often intersect and occlude each other but enter and leave the scene in a limited number of regions. VISOR data, instead, provide quite localized trajectories but several starting and finishing points. Moreover, it refers to an outdoor environment and thus presents some difficulties related to the illumination. Although CAVIAR might seem to include simpler videos, the variety of the represented events constitute an useful test bed for our application. Also, the presence of windows causes a disturbing illumination.

	Kalman (real time)	**Particle** ($\sim$ 12fps)	**CamShift** ($\sim$ 12fps)	[NKMVG03] ($\sim$ 9fps)	**Our Proposal** ($\sim$ 20fps)
CAVIAR	89.3%	89.1%	48.1%	48.1%	92.3%
VISOR	61.1%	65.1%	78.1%	81.4%	83.2%
PETS	70.2%	70.2%	77.3%	70.6%	80.5%

Table 3.1: Comparison between our proposal and a selection of popular tracking methods. The percentages refer to the number of estimated trajectories having a full correspondence with some entry in the ground truth.

We compare our proposed method with standard Kalman and Particle filters, whose state models consider target position and velocity, an appearance-based tracker derived from mean-shift, CamShift, and the method proposed in [NKMVG03], where particle filter is coupled with mean-shift. For what concerns Kalman and particle filter, we adopted our own implementations, which are bases on BFL libraries[4]; instead, we exploit OpenCV[5] for mean-shift and CamShift modules.

The table in Figure 3.1 summarizes the results obtained on a total of 42 videos and about 70.000 frames: the reported percentages refer to the number of measured trajectories having a full correspondence with the ground truth described above. We consider a range (of 30 frames or 50 frames, depending on the video frame rate) around the source and sink times reported in the ground truth to account for slight delays in the measurements.
As expected, CAVIAR data set suffers from illumination problems, thus a pure appearance-based tracker performs poorly. For what concerns VISOR, instead, it is apparent how the results benefit from the introduction of appearance properties of targets.
In the case of PETS, the high number of occlusions and intersections between targets causes failures for both motion and appearance-based approaches.
Globally, the method we propose outperforms the others. When the gap between results is

[4]http://www.orocos.org/bfl
[5]http://sourceforge.net/projects/opencvlibrary/

not significantly high, the computational efficiency clearly speaks in favor of our solution.

Figure 3.7: VISOR scenario: a target is correctly maintained for several frames (from left to right, time instants 1:45, 1:59, 2:02, 2:10), even in presence of temporary occlusions.

Our method has proved to be robust in time, as one can see in Figure 3.7, where a subject is nicely tracked in a long sequence, even after partial occlusion (after the third frame and before the last one). As expected, the appearance descriptor coupled with motion copes with difficult scenes, as in Figure 3.8, first row, where the target is maintained even when entering in a very difficult area, characterized by a high light intensity; mean-shift does not provide the same robustness (same figure, second row).
The system performs well even in the presence of complex configurations: Figure 3.9 shows an example from one of the VISOR scenarios, where in a scene with medium crowd most of the targets are correctly tracked along the sequence. However, as one can observe, some errors occurred.
A crucial feature of our approach is that it works at more than 20fps on 640×480 images and therefore guarantees portability on real video-surveillance systems. In Fig. 3.10 some

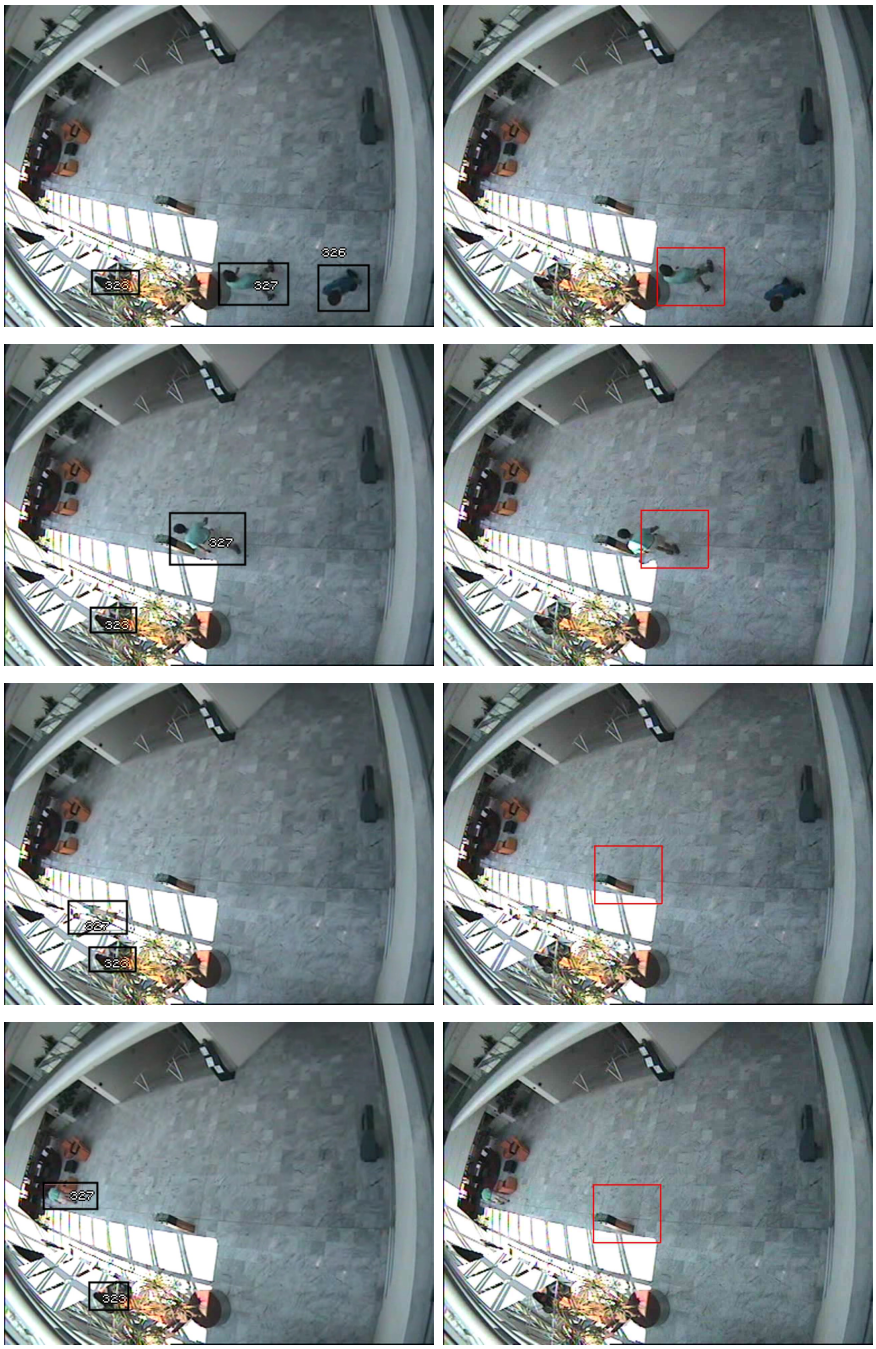


Figure 3.8: Results on a CAVIAR scenario. First column: thanks to the robust color descriptor and the use of motion information, our system can cope with difficult illumination. Right: the results obtained with mean-shift, showing wrong association caused by the failure of the appearance matching.

examples extracted from a real surveilled environment are shown, with the background smoothed for privacy issues.

Figure 3.9: A complex scene from the VISOR scenario: the system behaves nicely even in presence of many people temporary occluding each other.

3.5 Discussion

In this chapter we described the low-level analysis of our pipeline. We started from the very low-level processing, which is based on segmenting each frame of the sequence to detect moving regions. Then we moved to the tracking procedure, based on a combined motion and appearance model of objects. The method represents the data association problem as a graph simplification to adapt to different levels of the complexity of the monitored environment. A quantitative experimental analysis on benchmark data sets showed the

Figure 3.10: Sample frames of a real video-surveillance scenario (to keep the location secret, we only show the targets, smoothing the background). First and second images show multiple targets correctly maintained during time; in the others two people (identifiers 234 and 237) go towards each other, meeting and then splitting. The identities are re-associated.

appropriateness of our approach both in term of accuracy of the results and low computational cost. The evidence of the very good performance on standard data sets adopted in the field, encouraged us to adopt our tracking module as a pre-processing of a more complex behavior analysis pipeline, obtaining, as we will see in the remainder of the thesis, very good results.
As a further improvement, we plan to investigate the use of different appearance models, with the aim of reaching an higher accuracy against complex tracking configurations. Different appearance descriptors might adapt to different applications: when dealing with traffic scenarios, where the objects of interest, cars, have on average similar shape, color is strong cue. In presence of a more heterogeneous targets set, joint information on shape and color could be profitably exploited.

Chapter 4

Intermediate representations

This chapter is dedicated to the discussion on different higher-level representations (Sec. 4.2), explaining advantages and drawbacks of each one on a synthetic data set of $2D$ observations. Each representation is coupled with an appropriate kernel-based similarity measure (Sec. 4.3) and the performance of each scheme is qualitatively evaluated by means of similarity matrix (Sec. 4.4). The quantitative analysis is based instead on supervised learning (Sec. 4.4). In the second part of the chapter the benefit of using an heterogeneous input space, rather than the usual $2D$ information, is discussed (Sec. 4.5.1), showing how it is a suitable way to cope with data ambiguities. The correspondent experimental analysis is performed on a set of real yet controlled trajectories, acquired from a surveillance camera, for which a ground truth has been specified.

4.1 Introduction

Data abstraction is a mapping from input data gathered by the video processing phase (Chapter 3) to an appropriate feature space where data may be analyzed by means of machine learning algorithms.
Many successful methods have been proposed to discover classes of similar pattern in a data set. Unfortunately most of them are not tailored for learning from sequential data, such as temporal series. In fact, while analyzing temporal data, the *temporal dimension* usually induces a specific, inherent structure to the sequence of feature vectors that is likely to be disregarded by traditional learning methods.
Previous works on learning from temporal descriptions approached the problem from two distinct viewpoints (see [Lia05] and references therein):

- The first approach relies on modifying suitable existing algorithms for learning from

static data in such a way that time series data can be handled

- An alternative is to convert temporal series data into the form of static data so that the existing learning algorithms for static data can be directly used, trying to keep information on the structure of data.

The way we approach the problem of discovering similarities among temporal data is more related to the second, relying on solving two basic issues:

(1) How to map temporal series into intermediate representations suitable for learning purposes and able to keep the internal structures of the data?

(2) How to find a proper similarity measure to compare the representations?

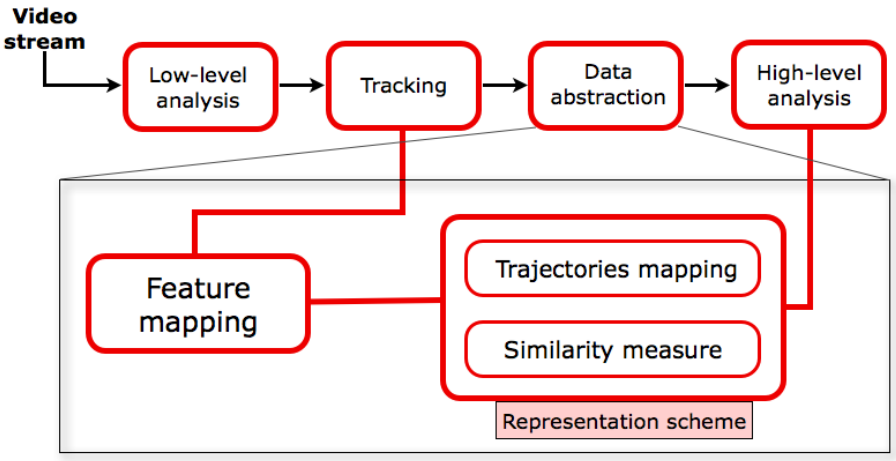


Figure 4.1: A visual representation of the module that in our system is dedicated to build higher level representations of trajectories.

In this chapter we discuss a procedure to deal with sequential data representations, shown in Fig. 4.1. The translation of the trajectories (feature mapping) gathered by the tracking is performed in order to better enhance the interesting features for the problem at hand (this phase is discussed in Sec. 4.5.1): some features extracted and stored during the tracking procedure might be unnecessary for the subsequent analysis (e.g. for our case color information) or better manageable if expressed in a different form.
Then, to increase the abstraction level of the trajectory representation, we consider different *representation scheme*, coupling specific parametric mappings of the trajectories in a different feature space with appropriate similarity measures (Sec. 4.2, Sec. 4.3). We more

Representation	Kernel
strings	P-spectrum
polynomial fitting	Gaussian
B-Splines fitting	Gaussian
HMMs	PPK

Table 4.1: The representation schemes we discuss and compare: among them, we propose the string-based solution.

specifically refer to kernel-based measure, being the kernel a common main ingredient of learning methods (for a brief introduction to kernels see the Appendix). Table 4.1 reports the schemes we compare that will be discuss in detail in the remainder of the chapter. Our main contribution is a string-based representation built on top of automatic partitions of the domain of the instantaneous observations which are elements of some trajectories.
The nice feature of our approach relies to the almost complete independent of its computation from the characteristics of the specific domain (e.g. the size of the feature vector) this it naturally applies to different scenarios. The comparison among strings is based on the P-Spectrum kernel [TC04].
We compare strings against a selection of promising representations proposed in the literature of temporal series modeling. Gaussian and B-Splines based approaches, largely used in this context, fit the trajectories with respect to one of the two and map them into the space of parameters; the well-known Gaussian kernel is appropriate for computing the similarities.
The last scheme relies on Hidden Markov Model and is suggested by the works in [JKH04, JST07], where the authors adopt the popular tool to model single temporal series with the HMM parameters, that are finally compared with the Probability Product Kernel (PPK).

4.2 Temporal series representations

In this section we provide a more detailed discussion on some popular representation schemes for temporal series, referring to techniques which showed promising performances during our experimental analysis and there adopted. The common underlying idea of the following approaches is that raw temporal data (trajectories) may be conveniently *translated* into a suitable parametric model. The subsequent behavioral analysis can then be performed using effective methods for dealing with static data.
This approach allows us to overcome two crucial issues related to temporal series: (1) the fact that the events to be represented do not last the same amount of time, and (2) they do not have the same spatio-temporal evolution (sometimes even within events of the same

class).

In the following, a data set of N temporal series is denoted by means of the set $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, where each datum $\mathbf{x}_i$ is a sequence of k_i vectors in some Euclidean space $\mathbb{R}^d$, i.e. $\mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^{k_i})^T$ and $x_i^t \in \mathbb{R}^d$, $t = 1, \ldots, k_i$. A natural way to interpret the index t is as the temporal index.
We denote with

$$\tilde{\mathbf{X}} = \{\{x_i^t\}_{t=1\ldots k_i}\}_{i=1\ldots N}$$

the set of instantaneous observations in the trajectories: it thus derives from the set $\mathbf{X}$ by discarding the temporal component (in other words, each observation is treated independently from the others).
After a brief discussion on non-parametric approaches, the reminder of the section will focus on the parametric counterpart.

4.2.1 Non-Parametric Approach

If a non-parametric approach is adopted, then the most simple and direct representation of a set of temporal series is simply $\mathbf{X}$ itself. For instance, in the special case where all the series have the same length, that is how saying

$$k_i = k \quad \forall i = 1...N.$$

It is sometimes convenient to think of a $k \times (d \cdot N)$ matrix $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ which provides a compact formulation of some structural properties of the data in terms of vector matrix calculations. When the lengths k_i are not all the same, zero-padding or resampling techniques may be applied.
However, this approach only applies to an extremely restricted class of applications, since it does not provide an appropriate solution to problems where some kind of higher abstraction is needed. A more common practice is to resort to parametric approaches: in the following we thus review, as a more general and useful alternative, a set of possible parameterization techniques for temporal data.

4.2.2 Curve Fitting

A popular parametric approach is based on *curve fitting*, in which a regression function is built for each sequence (see [PCV00] and references therein). Here an important step is to choose a specific family of fitting functions. Such choice depends on the observations distribution of set $\tilde{\mathbf{X}}$, which is strongly connected to the complexity of observed behaviors. When the feature vector includes measurements of different type, it is suggested to first

normalize the components (e.g. in the interval $[0, 1]$) to cope with the different numerical ranges where they might live.
Under reasonably simple conditions, a polynomial fitting might be a viable solution: a simple *least squares-based method* can be adopted to fit the regression function to a trajectory [LOW00].
In a more articulated scenario one relies on a family of *B-spline* models, as proposed in [PCV00], where a maximum likelihood regression is performed separately for each component to estimate the coefficients of a piecewise linear fixed-knot spline. The B-spline models have all the same fixed size according to the number m of knots. The normalization is required in order to make the coefficients of the B-spline models consistent among each other.
According to a common thread of previous works on the topic, the piecewise linear fixed knot splines are obtained using an intermediate feature space in which a point x is represented by a vector $\phi(x)$ defined as

$$\phi(x) = (1, x, |x - t_1|_+, \ldots, |x - t_m|_+) \tag{4.1}$$

where $|x - t_k|_+$ is equal to zero if $x < t_k$ and $(x - t_k)$ otherwise. In such space it is possible to define a Mercer kernel

$$K(x_i, x_j) = 1 + x_i x_j + \sum_{k=1}^{m} |x_i - t_k||x_j - t_k|. \tag{4.2}$$

At the end of the above process, each temporal series $\mathbf{x}_i$ can be described by means of a single $d \times (m + 2)$ dimensional vector obtained by concatenating the regression coefficients for all the dimensions of the feature vector x_i^t.

4.2.3 Probabilistic Models

A popular probabilistic approach to temporal series analysis is based on Hidden Markov Models (HMMs): a brief introduction to this class of methods is given in the Appendix.
The traditional use of HMMs relates to the estimation of sets of models underlying groups of temporally correlated data and whose number is known a-priori. Each model consists in the main parameters of an HMM $\theta_i = (\pi_i, \alpha_i, \mu_i, \Sigma_i)$ that univocally identifies it.
However, in the general case we consider, such labels are not available.
An alternative is thus to adopt HMMs to model single trajectories [JKH04, JST07]. From our point of view, the advantage of this second approach is that is it more related to the general representations schemes that we proposed, thus it is our choice for the experiments.

4.2.4 String-based approach

This approach is based on the observation that a temporal sequence of discrete elements can be seen as a concatenation of symbols from a finite *alphabet* $\mathcal{A}$.
Intuitively the alphabet could be associated to an appropriate partitioning of the input space, which could be performed *manually* or *automatically*: since the choice of an appropriate alphabet is crucial and as the input size d may grow manual partitioning is not always conceivable, the alternative is to address the problem of partitioning the space guided by the available data. In the literature methods for automatic space partitioning (usually based on vector quantization or clustering) have been proposed [SG00, HXF$^+$06a].

Focusing on our work, we adopt an approach originally proposed in [NSO08a] and previously discussed in [NSO08b] which relies on clustering data in the $d-$dimensional input space represented by the set $\check{\mathbf{X}}$ defined above. As for the specific clustering algorithm, we adopt *spectral clustering* (see the Appendix for more details). A peculiarity of the implementation we adopt is the possibility to control the granularity of the solution with a specific parameter, the cut threshold (TH). Such method requires an appropriate kernel function that allows us to build a similarity matrix able to capture the internal structure of the data set. To combine different measurements, which could take values in different ranges, various choices are possible. A first way is to simply concatenate input vector, after an appropriate data normalization in the $[0, 1]$ range, similar to the one adopted in the curve fitting approach. An alternative way (sometimes referred to as Multi-Cue Integration in the supervised learning literature [TOC08]), is based on a convex combination of similarity or kernel functions on sub-sets of coherent features. Given two observations in $\mathbb{R}^d$, x and y:

$$K(x, y) = \sum_{i=1}^{N_f} w_i K_i(x_i, y_i, \theta_i) \tag{4.3}$$

where

- $N_f \leq d$ is the number of features, component of the measurement vector
- $\{w_i\}_{i=1\ldots N}$ are the feature weights allowing to control the importance of each feature in the process. They are subjected to the constraint $\sum_{i=1}^{N_f} = 1$
- θ_i are the parameters of kernel K_i, which depends from its type.

Once the alphabet is built, a temporal series $\mathbf{x}_i$ may be translated into a string s with an association of each element $x_i^t \in \mathbb{R}^d$ to the partition it belongs to. The association can be naturally made if the trajectories points took part to the partition estimation; as opposite, *out-of-sample* methods can be exploited to associate new points to the partition.

To obtain compressed descriptions that capture the peculiarities of each behavior we consider only transitions between states, skipping the replicates of the same symbol, i.e., if a string contains replicates of the symbol $u \in \mathcal{A}$ it can be compressed as:

$$\alpha u^{\lambda} \beta \rightarrow \alpha u \beta$$

for every arbitrary substrings α and β.

4.3 Kernels for time-series

It is well assessed that, in the learning from examples approaches, the obtained results strongly depend on the ability to capture the underlying notion of metric - or the similarity structure - over the input space. In turn, the similarity structure is related to the choice of a proper kernel function on the data. Since in our experiments we compare different representation schemes we have to choose appropriate kernels.

In the case of curve fitting, we refer to a well established kernels, the Gaussian kernels. Given two temporal series $\mathbf{x}$ and $\mathbf{y}$ and their parameterization in the B-splines coefficients space $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$, the kernel is defined as

$$K(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = e^{-\frac{||\hat{\mathbf{x}} - \hat{\mathbf{y}}||^2}{2\sigma^2}}$$

where σ denotes the standard deviation of the gaussian and must be conveniently estimated on the input data.

The other two approaches require a more careful choice of the kernel. On one case, the HMMs, the capability of dealing with data that are distributions is needed. In the case of strings, instead, the main issue regards the variable lengths of the descriptions. In the following we briefly discuss the main characteristics of the kernels we chose.

4.3.1 Probability Product Kernel (PPK)

PPK is an efficient measure of similarity among temporal series represented by means of HMMs, introduced in [JKH04], and used in [JST07] in tasks similar to ours.
Given two probability distributions representing two pairs of data sequences over the space of all potential observable sequences X, the generalized inner product can be computed by integrating the product of the distributions:

$$K(p(x|\theta), p(x|\theta)) = \int_X p^{\beta}(x|\theta) p^{\beta}(x|\theta) dx, \tag{4.4}$$

where β is a free parameter which allows for the specification of some properties of the kernel. For instance, if $\beta = 1/2$, the PPK becomes the classic Bhattacharyya affinity metric between two probability distributions, which is a Mercer kernel and can be conveniently used in our context. In addition, it is computable in closed form for a variety of distribution families.

4.3.2 Kernels for string-based representations

In this case the choice of an appropriate kernel is quite critical, considering the peculiarities of the available data. Indeed, even after the translation of each temporal series in a string, the event description is still variable length – depending on the number of transitions of a behavior instance from one state to another.
When translating a temporal series with respect to a finite alphabet, it is possible to borrow from studies on text and biological data manipulation [LEN03, LSST$^+$02]. The most natural way to compare two series is to count how many transition they have in common from one symbol of the alphabet to a different. This is tightly related to the concept of *spectrum of order P* (also known as *P-spectrum* kernel) of a sequence s to be the histogram of frequencies of all its (contiguous) substrings of length P. Formally, the kernel may be defined as a feature map of strings followed by an appropriate dot product [TC04]. The map makes explicit all sub-strings of length P of string s:

$$\phi_u^P(s) = |\{(v_1, v_2) : s = v_1 u v_2\}|, u \in \mathcal{A}^P. \tag{4.5}$$

The associated kernel between two strings s and t is defined as:

$$K_P(s,t) = \langle \phi^P(s), \phi^P(t) \rangle = \sum_{u \in \mathcal{A}^P} \phi_u^P(s) \phi_u^P(t). \tag{4.6}$$

String length independence is achieved with by the following normalization [TC04]

$$\hat{K}_P(s,t) = \frac{K_P(s,t)}{\sqrt{K_P(s,s)}\sqrt{K_P(t,t)}}. \tag{4.7}$$

P-spectrum kernel is a Mercer's kernel, therefore it is also symmetric and positive.

4.4 Experimental assessment on synthetic data

In this section we show the results obtained during an experimental analysis performed on a set of synthetic 2-dimensional data. The discussion will focus on different issues, related to

- Clarify the specific properties of each scheme
- Compare the different representations applied to the problem of discriminate among trajectories
- Validate the whole representation pipeline.

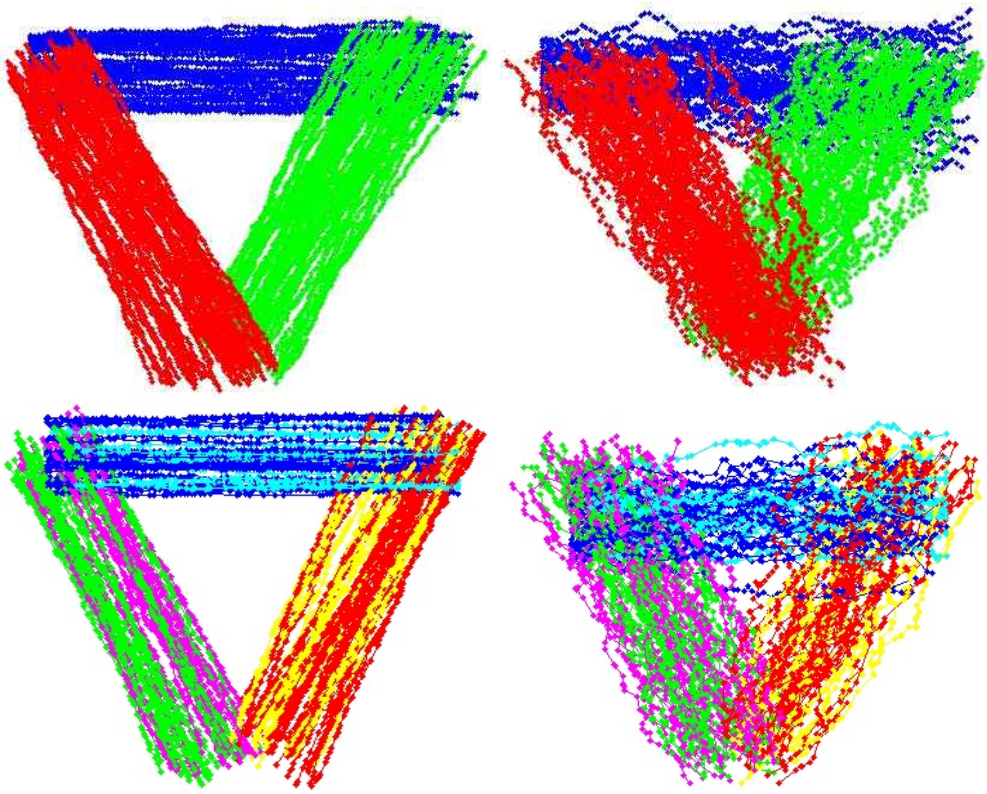


Figure 4.2: Examples of series from the synthetic data sets generated to assess the representation schemes: the top row refers to $DS3B$ (unidirectional series), while below examples from $DS6B$ (bidirectional series) are reported. On both rows, on the left the noise level is low and the trajectories are smooth and almost linear. On the right the noise level increases and the classification of the series is far more difficult.

We generated a number of synthetical temporal series of $2D$ points on a plane - see Figure 4.2 - building the data set we called $DS3B$. Such data are evocative of a simple yet realistic video surveillance scenario, where positions of objects moving at constant velocity along the image plane are used to describe dynamic events. In this case data are fully represented by their position, thus the input space is so that $d = 2$. The sequences are divided into 5 groups with an increasing level of noise on the points (0.1, 0.3, 0.5, 0.7, 0.9 respectively),

so to model situations where a smooth object tracking is increasingly more difficult (e.g., because of scene clutter or bad illumination).
Each group of series comprises 3 distinct behavioral models (represented with different colors in Figure 4.2, top row). Each behavior is composed by $N = 150$ series, each of length $L = 50$. In order to make some statistical analysis of the results, we built 20 different re-samplings of each group of data.
Since in principle some of the proposed kernels should be able to capture more complex properties of temporal series - such as to distinguish between opposite directions - we generated a second data set (named $DS6B$) where each of the 3 behaviors is split in two subgroups by choosing randomly the direction of the points. The opposite directions are highlighted in Fig. 4.2, below, with different colors. By exploiting the availability of labels we can test whether a particular parameterization coupled with the corresponding kernel have the capability to characterize groups of coherent trajectories (the 3 or 6 behaviors for $DS3B$ and $DS6B$, respectively).

4.4.1 Representation schemes: some discussions

Each scheme of representation poses a number of practical issues that have to be solved adequately: in the following we review and discuss the main issues related to each type of parameterization.

Gaussian Kernels for curve fitting

When adopting polynomials as fitting functions, the main issue concerns the choice of the degree. The lesser the degree, the smaller is the dimensionality of the parameter space and the faster is the fitting algorithm. However, if the degree is too low, the approximation of the actual points of the series may be too loose and the method may fail to capture important properties of the data. Therefore one always needs to balance between these two important aspects.
In order to avoid arbitrariness at this stage, we adopted the following heuristic strategy. For each data set, we first selected randomly a small fraction of series and fit polynomials with increasing degree. The ratio between the number of non zero coefficients and the total number of coefficients represents how complex the data are, and can be used to choose the proper degree for that specific data set. We observed experimentally that straight lines are sufficient if the level of noise is low, while noisy data can be represented by means of second (at most third) degree curves.
A normalization step is suggested by a visual inspection of the resulting coefficients that are numerically rather separated. In this experimental phase we will compare the use of the original coefficients as well as the normalized version.

For what concerns B-Splines, the derived parametric model requires, as parameters, just the definition of the fixed knots m. Since the described approach relies on several mono-dimensional curve fitting problems – with the independent variable being the time – one has to set only the temporal spacing between two consecutive knots. In our experiments[1], we set such spacing as one third of the average length of the trajectories in our data set. In such way, on average, all the trajectories are represented by a fair number of non trivial regression coefficients.

Probability product kernels for HMM-based representation

The initialization stage is the very critical for the HMM-based method.
However, according to [JKH04], since the HMM is just an intermediate step in forming the kernel and to capture the similarity among series, it is not necessary to build a very accurate model. In principle, we agree with this viewpoint, but in our application scenario temporal series corresponding to the same behavioral pattern may have very different lengths and structure. These may affect negatively the the algorithm that "learns" the HMM model [2]. In all our experiments, the number of possible states is 3 and the probability density function corresponding to each state is Gaussian.

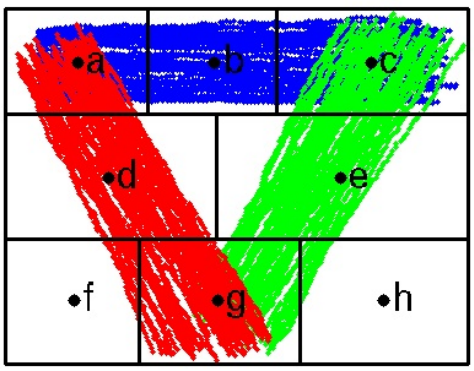
The kernel used to compare two HMMs are the probability product kernels, described briefly in section 4.3.1. According to the authors that first introduced such kernels [JKH04], they are very robust with respect to the choice of the kernel parameters. In the specific case of HMM, the kernel becomes the popular Bhattacharyya affinity matrix, and the only significative parameter T is called *mixing proportion*, and corresponds to the time interval considered for the evolution of the underlying dynamical systems. In our experiments we used the same value ($T = 10$) proposed in [JST07], where the authors showed convincing evidence about the stability of such value. Our experiments confirmed such claim.

P-spectrum kernels for string-based representations

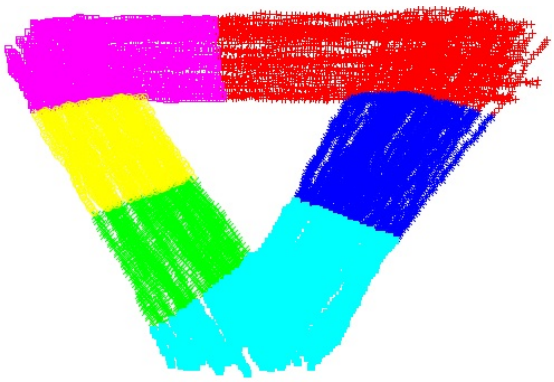
A comparison between manual and automatic partition is shown, respectively, in Fig. 4.3(a) and Fig. 4.3(b). The association of a symbol/state to each point in the sequence is shown in Fig. 4.3(b), where the colours represent different symbols and each string is a concatenation of symbols. As for the choice of the partitioning strategy, we evaluate three different approaches against increasing noise levels:

[1] Our software for experiments on B-Splines is based on the *Spline Matlab Toolbox*, further details at `http://www.mathworks.com/products/splines/`.

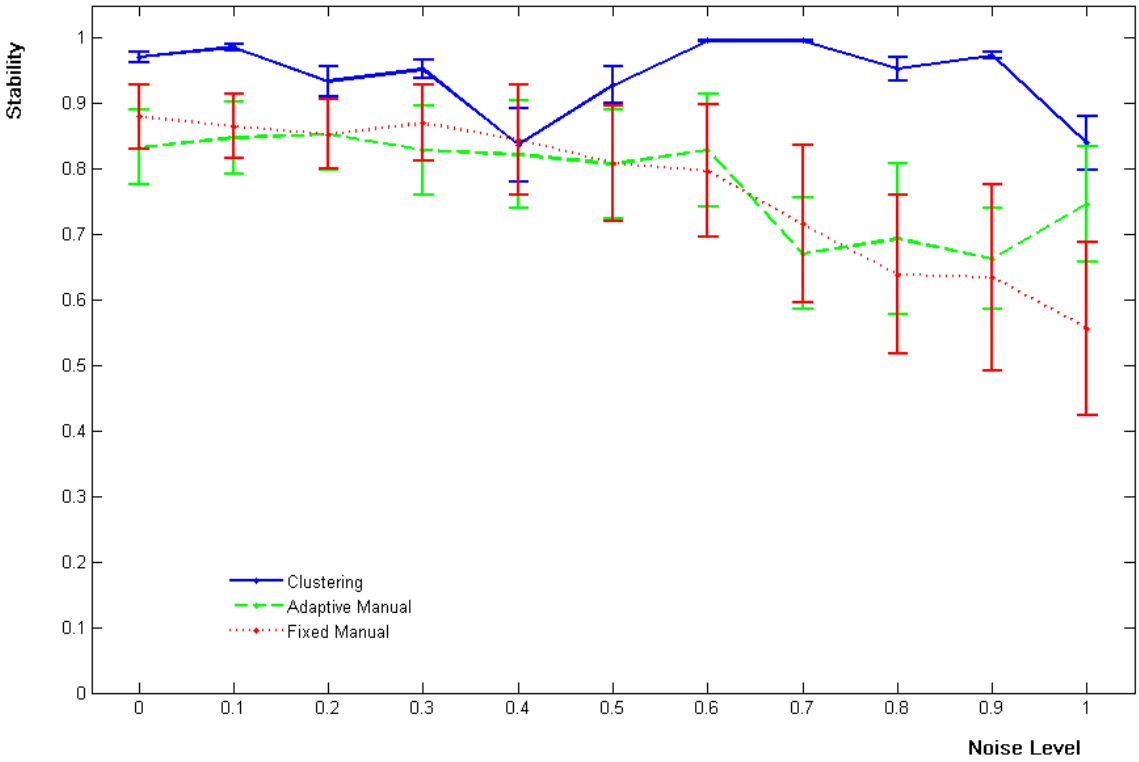
[2] We learned the parameters of HMMs from our data by means of well established *Bayes Net Toolbox* for Matlab, available for downloading at `http://people.cs.ubc.ca/ murphyk/Software/BNT/bnt.html`. Developed by Kevin Murphy, it is widespread used by machine learning practitioners.



(a) Manual Grid



(b) Automatic quantization



(c) Stability

Figure 4.3: The choice of a proper alphabet. The top-left figure shows the states assigned by manual choice, compared with the result of our automatic procedures (top-right). The bottom plots show the stability of the representations while the noise level increases.

1. Data-driven partition

2. Fixed manual partition (performed once, when the noise is low)

3. Adaptive manual partition.

From the perspective of the representation, the *goodness* of a partition, and consequently of an alphabet, reflects in its capability to similarly represent series (trajectories) which are instances of a common behavioral model. For a quantitative assessment of the *quality* of the alphabet definition on data set $DS3B$, we evaluated the *stability* of the representation in terms of how similar the corresponding strings are when two similar sequences are considered. Since the series in data set $DS3B$ and $DS6B$ are known to belong to a fixed set of behavioral classes, we evaluated the intra-class stability of the different methods via the computation of averages and variances of intra-class similarities. The values are computed separately for each of the 3 behaviors and then averaged. Similarity between strings is computed with the P-spectrum kernel with $P = 2$. This choice is dictated by the the fact that, as reported in Sec. 4.2.4, we keep only the transitions between atomic symbols to build the final strings. Fig. 4.3, below, shows the trends of mean and variance over 20 replicates of different data sets with increasing noise. Blue plot refers to the automatic partition, red plot to the definition of a fixed manual partition specified - and kept fixed - at the beginning of all the experiments. Finally, in the green plot an ad-hoc partition is defined for each experiment: even if the results are slightly better that those obtained with the fixed manual partition, this approach is not suitable for real applications. It is apparent that the most robust approach is based on automatic partitioning the data, since both its average similarities are almost independent on the quantity of noise and the variances are almost negligible.

The effect of parameter selection in spectral clustering is not crucial to the quality of the obtained alphabet. We tested

(*i*) The impact of varying the high cut threshold of the spectral clustering of the alphabet (top row of Fig. 4.4), and

(*ii*) To what extent the clustering is robust with respect to the resampling of the points along each trajectory (Fig. 4.4, below).

The results we obtained clearly confirm the robustness of the underlying method, showing that.

(*i*) Varying the cut threshold influences the level of granularity of the alphabet. In other words it allows for an adaptation of the partition which may depend on the input data and the specific task;

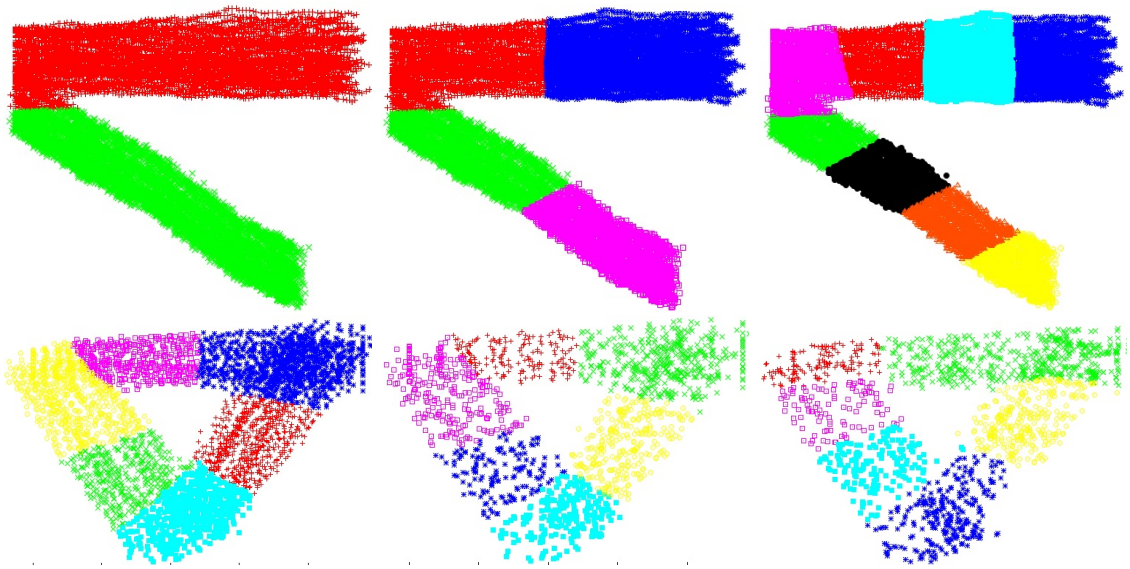


Figure 4.4: Parameter tuning for spectral clustering. Top row shows the dependence on the cut threshold TH parameter (from left to right. $TH = 0.1, 0.4, 0.6$). The results reflect the expected increase in the alphabet refinement. Bottom row shows the robustness to the points density (from left to right, the points are 25%, 10% (regular subsampling) and 10% (random subsampling) of the entire data set). A Gaussian kernel with $\sigma = 10$ was used in all experiments.

(*ii*) The capability to identify the internal structures of the input data is guaranteed, although the density of such data may vary.

4.4.2 The choice of p for the P-Spectrum

In this section we explore the P-Spectrum kernel from the perspective of a correct choice for the p parameter, to enhance possible relationships between such values and the structure of the strings to be compared. In particular we investigate the influence of p when comparing strings composed by the same symbols in reversed order. The following analysis looks towards a justification for our choice of keeping only the transitions between states in the strings representation and the consequent adoption of a 2-Spectrum kernel.
We consider in the experiments reported here a set of n_s strings $\{s_i\}_{i=1...n_s}$ built on top on an alphabet $\mathcal{A} = \{a_1, ..., a_n\}$ of n symbols. The strings have lengths l_i and are composed by concatenating $r_i = \frac{l_i}{n}$ replicates of each symbol. We fix the length of a pair of strings

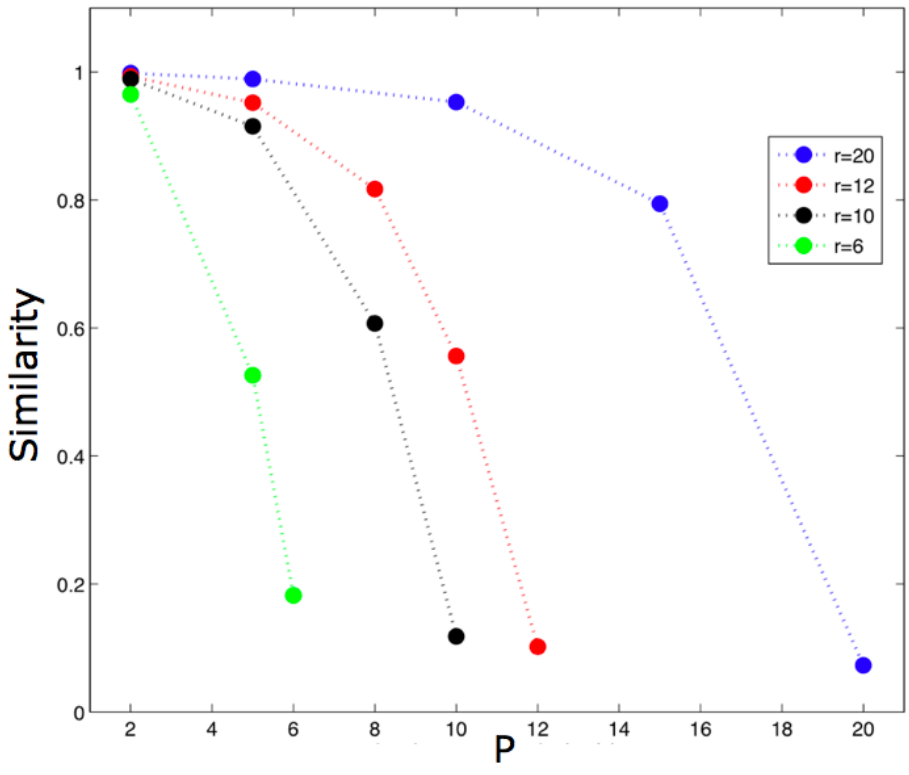


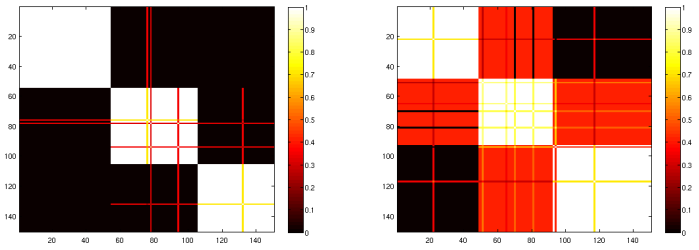
Figure 4.5: The effect of choosing p when comparing strings with the same replicated symbols in reversed order.

s and t built on the alphabet $\mathcal{A}$ and evaluate the pairwise similarity when varying the

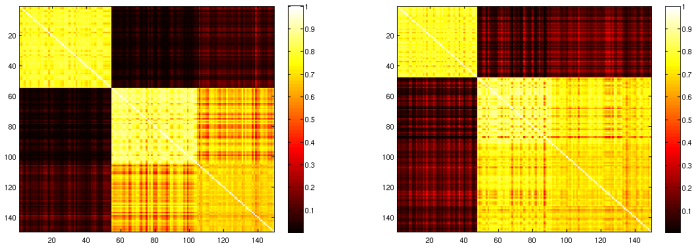
symbols order: more specifically we consider strings of reversed order so, for instance, if $s = a_1^k\, a_2^l\, a_3^m$ then $t = a_3^m\, a_2^l\, a_1^k$ where $a_1, a_2, a_3 \in \mathcal{A}$.
The goal of this experiment is to evaluate the influence on the similarity between s and t of both the symbols configuration and the choice of p: it is worth to point out that in our interpretation reversed strings correspond to different dynamic events.
Intuitively, when dealing with a high number of consecutive replicates of the same symbols, a small p should cause a low influence of the transitions between different symbols; as opposite, the transitions should become more and more important as p increases.
Fig. 4.5 reports the results of this analysis performed for different numbers r of replicates of symbols averaging the similarity of 10 pairs of reversed strings: as one can observe, for a given r, the choice of p strongly influences the similarities, that tends to decrease as the value of p approaches r. The results are helpful in different aspects. First, they suggest that when the problem at hand requires that two reversed strings are different, then transitions between symbols must be preferred. Second, such results is achieved when the value of p is similar to the number of replicates.
In our setting it is not possible to know a priori the latter, so the natural choice is to achieve the same robustness by discarding the replicates of symbols from strings, so to keep only the transitions. From the standpoint of the computational cost, moreover, it is advisable to reduce the amount of information (i.e. the length of strings) if the same performance can be achieved by means of an appropriate tuning of the parameters.

4.4.3 Temporal series classification

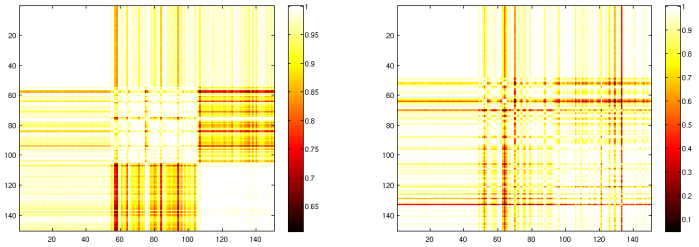
As a first analysis, we inspected the similarity matrices obtained with each kernel function and noise level on data set $BS3B$, see Fig. 4.6. In order to highlight possible block structures of the matrices and make the qualitative analysis more effective, we ordered the series beforehand so that the "optimal" classifier should produce exactly three distinct blocks along the diagonal of the similarity matrix.
It is apparent from the visual analysis of Fig. 4.6 that a P-spectrum kernel ($P = 2$) combined with string based-representation is very close to get this optimal result when the noise level is low and it is also the closest one on more noisy data (Fig. 4.6 (a), from left to right). The similarity matrices corresponding to the manual annotation look very similar to the ones we reported.
As opposite, polynomials (Fig. 4.6 (b)) provide the poorest performances even in presence of low noise. The matrices we report are estimated by normalizing the coefficients of polynomials to be in $[0, 1]$, step that slightly improves the results.
HMMs and B-splines present rather equivalent performances, even in different aspects: while the former seem to prefer discriminating between different classes, the latter better highlights intra-class similarities. This simple visual analysis makes thus difficult to speak in favor of one of the two approaches.



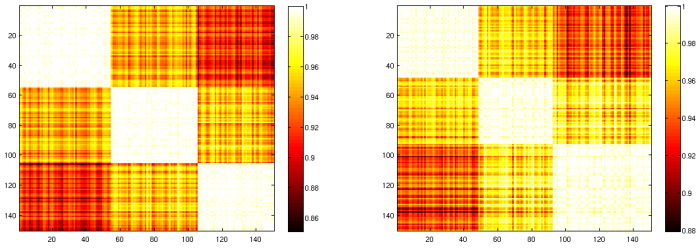
(a) Automatic partition on strings



(b) HMMs



(c) Normalized polynomials



(d) B-Splines

Figure 4.6: Similarity matrices on data set $DS3B$: the left examples refer to moderately noisy data (0.3), on the right the noise level is increased (0.9).

	Data set DS3B					Data set DS6B				
Noise Level	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
Strings (automatic)	96.8	96.5	98.4	96.9	95.5	81.7	81.8	82.3	81.6	78.9
Strings (manual)	96.8	96.7	98.2	96.6	95.3	81.9	81.3	82.4	81.1	78.2
HMMs	96.5	96.5	97.9	95.4	93.8	81.1	79.8	79.7	76.9	74.4
Polynomials	95.1	91	87.4	81.9	76.9	43.3	41.4	42.7	39.6	36.7
Norm. Polynomials	88.7	81.2	84.9	75.1	74.3	39.4	38.7	41.6	39.3	37.7
Splines	96.8	96.7	98.6	97.2	96.5	81.5	81.8	82.8	81.7	79.4

Table 4.2: Supervised analysis based on RLS on data sets $DS3B$ and $DS6B$.

When the data set is synthetically generated with respect to a set of underlying known models - in other words, the data set is annotated with labels that associate each temporal series to the corresponding behavioral model - then supervised learning from examples is a viable and efficient solution to validate the different data mapping.
The problem is a *multi-class classification* task, that we solved by means of a one-vs-all Regularized Least Squares (RLS) approach (see the Appendix).
Among the method a kernel is needed to be a Mercer's kernel, that is *symmetric* and *positive definite*. The kernels we associate so each representation scheme present this properties, then are viable functions.
The problem of selecting the parameters is tackled by k-Fold Cross Validation (k-CV), with $k = 5$ sub-samplings; the numerical range of the regularization parameter λ has been chosen as $[10^{-3}, 1]$.
Table 4.2 reports the results we obtained by following this protocol. When dealing with data set $DS3B$, almost all the mappings behave quite nicely and show robustness against the noise increasing. Polynomials decrease their performances as noise increases, even if not significantly. Moving to the slightly more difficult bidirectional series - data set $DS6B$ - their inappropriateness for the problem at hand arises. The other schemes, instead, properly handled the representation tasks.
Note that when applying the HMMs following the "traditional" formulation (which only applies to cases where the ground truth is available), the performance are in almost all the cases above the 98%, testifying how the adoption on single trajectories does not allow us to exploit in the best way the methodology (we run $k = 5$ different times the learning algorithm for each class keeping separate a fraction of the data set for the test, trying to be as close as possible to the experimental setting for RLS).

The results obtained in this assessment phase suggest that polynomials are not suitable to deal with our data. For what concerns string-based approaches, we substantially observe the equivalence between manual and automatic partition and the advantage, for the latter,

to be data-driven and thus easily adapt to changes in the observations.
For these reasons, in the remainder experimental analysis we will concentrate on *strings with automatic partition*, *HMMs* and *B-Splines* representations.

Figure 4.7: Data set $RC5B$. On the first row, the annotation of 5 behavioral patterns corresponding to people walking (left). The other images refer to samples trajectories referring to behavior 1 to 5.

4.5 Experiments on real data

We now consider a real, yet controlled, scenario: an annotated data set has been acquired by means of a video surveillance setup[3], discussed more in details in Chapter 6.
Dynamic events correspond to people walking or running according to a fixed number of behavioral patterns. More specifically, firstly we built a data set $RC5B$ of real controlled sequences, instances of 5 different behaviors, of about 50 series. Fig. 4.7 reports a visual schema of the annotation (first row on the left) and samples from each behavior. The trajectories have been covered by people that were walking following one of the allowed behavioral patterns. Secondly, we enrich $RC5B$ with some new classes, including patterns that differ from one of the known models in the velocity modules: we created such data

[3]The Imanalysis suite, we obtained within a technology transfer program with the company Imavis srl, `http://www.imavis.com/`.

set acquiring dynamic events of people running. The new set of data, whose annotation is reported in Fig. 4.8 (left) constitutes the new data set $RC11B$. The input space will be thus characterized by a more significant richness from the semantics standpoint, since behavioral patterns that share the same spatial regions - in other words the positions are overlapped - are actually different (people walking against people running). It is the case, as an example, of behavior 2 and 10, visually represented in Fig. 4.8, right: notice how the gap between consecutive points gives a visual evidence of the different dynamics.

Figure 4.8: On the left, behavioral patterns added to $RC5B$ to compose the new data set $RC11B$. The main difference is in the richness of the input data, since some groups of trajectories correspond to people running (dotted arrows on the left image). With these new inclusions, some behaviors will share the same spatial regions but differ in the dynamic properties: on the right, this is clarified for behavior 2 (blue series) and 10 (orange).

4.5.1 The choice of heterogeneous input spaces

In the literature of video analysis systems based on the use of trajectories of instantaneous observations, a common choice is to describe dynamic events by means of targets position into the image plane [NSO08b, PMF08]. In general, modeling the observations as plain positions may be ambiguous in many situations, where the camera is not optimally placed or it is not possible to intervene on it. Consider the case of a camera placed on front of the observed scenario, as illustrated in Fig. 4.9, where two different targets are moving covering spatially overlapping regions. If considering only their positions, the two different dynamic events occurring could not discriminated (Fig. 4.9, right). However, we can observe how the apparent size of the targets is clearly different among the two: the analysis of the trend of such feature, reported in the plot of Fig. 4.10 testifies its pertinence, showing that

Figure 4.9: A video surveillance scenario where the camera is placed in front of the scenario of interest. The plain positions are ambiguous and do not allow to discriminate between the events.

the numerical ranges of sizes are rather distinct. Moreover, we can notice that even the apparent targets velocity is perceived as different.

A way to cope with ambiguities is to consider heterogeneous input spaces where to measure, not only position, but also dynamic features (e.g., velocity or acceleration) or geometric features (as area, perimeter or shape features — the latter useful if a variety of different objects may be moving in the scene). Notice that such features are often available, or easy to compute, from the object tracking phase. For this reason, it is advisable to map the trajectory description (as gathered by the tracking process) into an appropriate form, with the capability of capturing information semantically meaningful for the problem at hand.

However, in general, the meaning of such measurements may change according to the application domain under consideration. Let us consider an environment where, perhaps in some temporal ranges, the access is allowed only to security guards wearing uniforms. In that case, the color of clothes should be taken into account by a monitoring application, since uniforms are expected to be comprised in a certain hue range. Supposing instead that the purpose of the system is instead monitoring a wider area where many people could be present - e.g. the level of crowd might be of interest for security - the way people are dressed is completely irrelevant.

In the next section we discuss the properties of the features we adopt.

For clarity, we recall and report here the descriptions adopted by our tracking system, as discussed in Chap. 3, to which we refer the reader for more details. At time t, the instantaneous observation of a moving object or *target* O_i, is a vector $\mathcal{M}_i^t = [TH_i^t, \mathbf{P}_i^t, S_i^t, \mathbf{V}_i^t, \mathbf{CH}_i^t, W_i^t, H_i^t]$ where TH_i^t is the frame time-stamp, $\mathbf{P}_i^t = [x_i^t, y_i^t]$ the location, S_i^t the size, $\mathbf{V}_i^t = [vx_i^t, vy_i^t]$

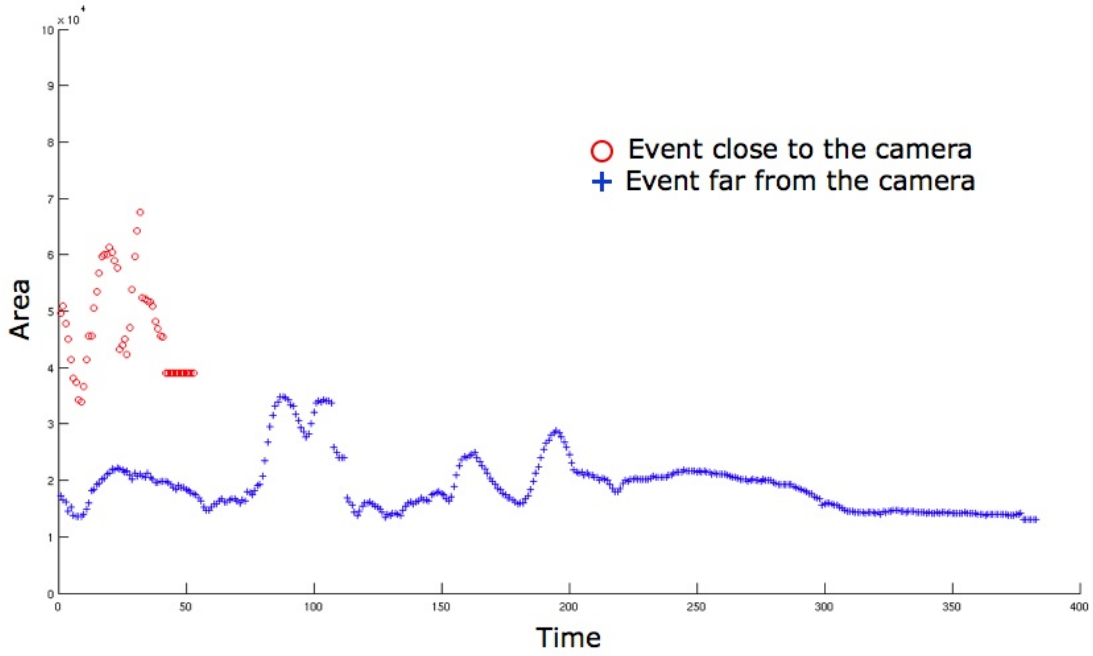


Figure 4.10: The size of targets is differently perceived depending on their distances from the camera. This suggests that they could be profitably inserted in their description to cope with ambiguities.

the apparent velocity and, finally, $\mathbf{CH}_i^t$ is the appearance description.
We derived from this vector our current description, that seems appropriate for the degree of complexity of the available test bed, that will be discussed in Sec. 4.5, and takes inspiration from the seminal work by Stauffer and his co-workers [SG00]. It may be summarized as a 5-dim vector

$$x_i^t = [\mathbf{P}_i^t, S_i^t, M_i^t, D_i^t]$$

where,

- $\mathbf{P}_i^t \in \mathbb{R}^2$ still represents the 2-dim object position on the image plane
- $S_i \in \mathbb{R}$ is the apparent object size at time t
- $M_i^t = \sqrt{(vx_i^t)^2 + (vy_i^t)^2} \in \mathbb{R}$ is the velocity module
- $D_i^t = atan(\frac{vy_i^t}{vx_i^t}) \in \mathbb{R}$ represents the direction of motion.

Appearance information is discarded, being not meaningful for the modeling experimental setting we consider in the following. The time-stamp is disregarded since temporal

alignment among events is not relevant to our purpose. Finally, width and height of the bounding box, related to of the target dimension, are synthesized with the only size feature. The velocity is represented in the module-direction space as it is semantically more meaningful (e.g. the module of the velocity helps to discriminate among walking or running people, while the direction is an apparent and powerful feature that groups of coherent events probably have in common). As we will see, these features appear to summarize a set of interesting properties among which building the behavioral models.

4.5.2 Data abstraction from heterogeneous inputs

Before moving to the analysis of the different representation, some more details on the extension of the representation schemes to the higher dimensional input vector must be given.
For all the representations involving a Gaussian function, the standard deviation σ must be selected. We empirically estimated an appropriate value as $\sigma_i = \frac{diameter_i(data)}{10}$ for each $i = 1...N_f \leq d$ - that is the values are computed separately for each feature (to deal with the fact that the numerical ranges of different features could be different as well) and depend on the diameter [4] of the data set, when projecting the data only on that feature.

String-based with Multi-Cue integration. For the automatic partition a clustering step must be performed, and to do so a proper kernel must be selected. As anticipated in Sec. 4.2.4, a viable solution to deal with an heterogeneous feature vector is adopting a strategy based on *Multi-Cue Integration* which allows us to easily extend the setting to changes of the input space. More specifically, in our case, $d = 5$ while $N_f = 4$ (being the position a 2-dim feature), so, given two observations $O_1, O_2 \in \mathbb{R}^5$ with $O_1 = (p_1, s_1, m_1, d_1)$ and $O_2 = (p_2, s_2, m_2, d_2)$, the final kernel can be written as

$$K(O_1, O_2) = W_P G_P(p_1, p_2, \theta_P) + W_S G_S(s_1, s_2, \theta_S) + W_M G_M(m_1, m_2, \theta_M) + W_D G_D(d_1, d_2, \theta_D)$$

where G_P, G_S, G_M, G_D are Gaussian functions and θs refer to their parameters, mean and standard deviation.
As discussed on Sec. 4.2.4, the vector $[W_P, W_S, W_M, W_D]$ includes the weights associated to each feature, which we might be interested in adapting with respect to different scenarios or tasks.
Instead of fixing a set of parameters, we build a family of alphabets

$$\mathcal{F} = \{\mathcal{A}_{par}\}_{par \in \mathbf{PAR}}$$

[4] The diameter d of a data set S is defined as $d = max_{x,y \in S} dist(x, y)$, where $dist$ is a proper distance (the Euclidean distance in our experiments).

where **PAR** is the set of allowed combination of parameters. As for the specific parameters choice, we consider each selection of weights in the set

$$\mathbf{W} = \cup_k \{[W_P, W_S, W_M, W_D] | W_i \in \{0, k\}, \forall i \in \{P, S, M, D\}\}$$

where $k \in \{\frac{1}{4}, \frac{1}{2}, 1\}$ and $\sum_i W_i = 1$. Moreover, since we adopt Spectral Clustering as specific method to build the partitions, a cut threshold TH_{points} must be defined (see the Appendix): even in this case we vary the value, so $TH_{points} \in [0.6, 1]$, sampling the interval with a step of 0.05. We thus end up with 99 possible models, among which we aim at selecting the one best fitting our data set with a further analysis.
To conclude this section, we briefly comment the influence of weighting different features on the automatic partition. Fig. 4.11 reports two partitions built with a low threshold $TH_{points} = 0.6$ (to better distinguish the clusters). On the left, $W_P = 1$, thus only the position is weighted when partitioning the data: the clusters reflect in turn spatial properties. On the right, instead, $W_P = \frac{1}{2}$ and $W_S = \frac{1}{2}$, that is both the position and the size features are considered. The projection of the estimated clusters on the image plane shows that different structures of the data are enhanced. From this considerations, the ductility of the method arises: changing the weights means to look at the data from slightly different view-points.

Figure 4.11: Examples of automatic partitions of the input data: on the left, only positions are considered, while on the right the size is also taken into account

Probabilistic models. In this case we considered two different configurations. According to the first one, the parameters of the HMMs are learnt from the data using, at each time, input vectors comprising all the features (i.e. X and Y position on the image plane, area of the tracked object, and amplitude and modulus of the vector representing the

approximate velocity). According to the second configuration, training data of each trajectory are based on only the two features corresponding to the position on the image plane. As already pointed out in [NSO08b], HMMs depend strongly on the initialization of their joint probability density function. Since in the unsupervised setting the HMM is used only as an intermediate step towards the computation of the similarity function between two trajectories, one is forced to train the HMMs using one trajectory only. Note that comments about this possibly troublesome choice are present in the original work [JKH04] to which we refer the interested reader for further discussions. In principle, such approach is somehow robust with respect to the specific dimensionality of the instantaneous observations. That is, the algorithm used for generating the representative vectors θ_i does not depend on the number of components of the single vectors x_i^t. Although this is certainly true from an algorithmic viewpoint, in practice we noted that the higher is the number of dimensions the coarsest is estimation of the HMM parameters, and in many cases it may be troublesome to consider them as probabilistic generators of the observed trajectories. For this reason in the experiments we report the results obtained with the second configuration only.

4.5.3 Real data classification

Let us start with data set $RC5B$ and string-based representation. Table 4.3 reports the results in order of quality: for each selection of weights, we select the cut threshold TH_{points} that produces the partition corresponding to the descriptions with the lowest classification error. The last column reports the percentages of trajectories that have been correctly classified. It is apparent that the position is a highly discriminative feature for the specific scenario and the specific data set. This is not surprising, since we train the learning systems with a set of data highly discriminated by spatial properties.

Also, the direction seems to be rather significant, being the alphabets where position and/or direction are weighted the best performing.
Fig. 4.12 reports the trends of the error, computed on average on the 5 splittings of the 5-fold cross validation, against the value of the regularization parameter λ. The figure shows such analysis for the best performing alphabet, on top, and the worst one, on bottom; the different plots refer to different values of the cut threshold TH_{points}. As one can observe, the errors reported on the left are in average consistently lower than on the right. The λ parameter does not strongly affect the performance if considering an appropriate cut thresholds. When analyzing an alphabet which is not appropriate in modeling the data set, it is apparent that the choice of the parameters strongly influences the performance, since the errors are higher.

From this analysis one could conclude that the alphabet 10 is appropriate and sufficient to well define the events occurring in the given scenario. However, when the ambition of a

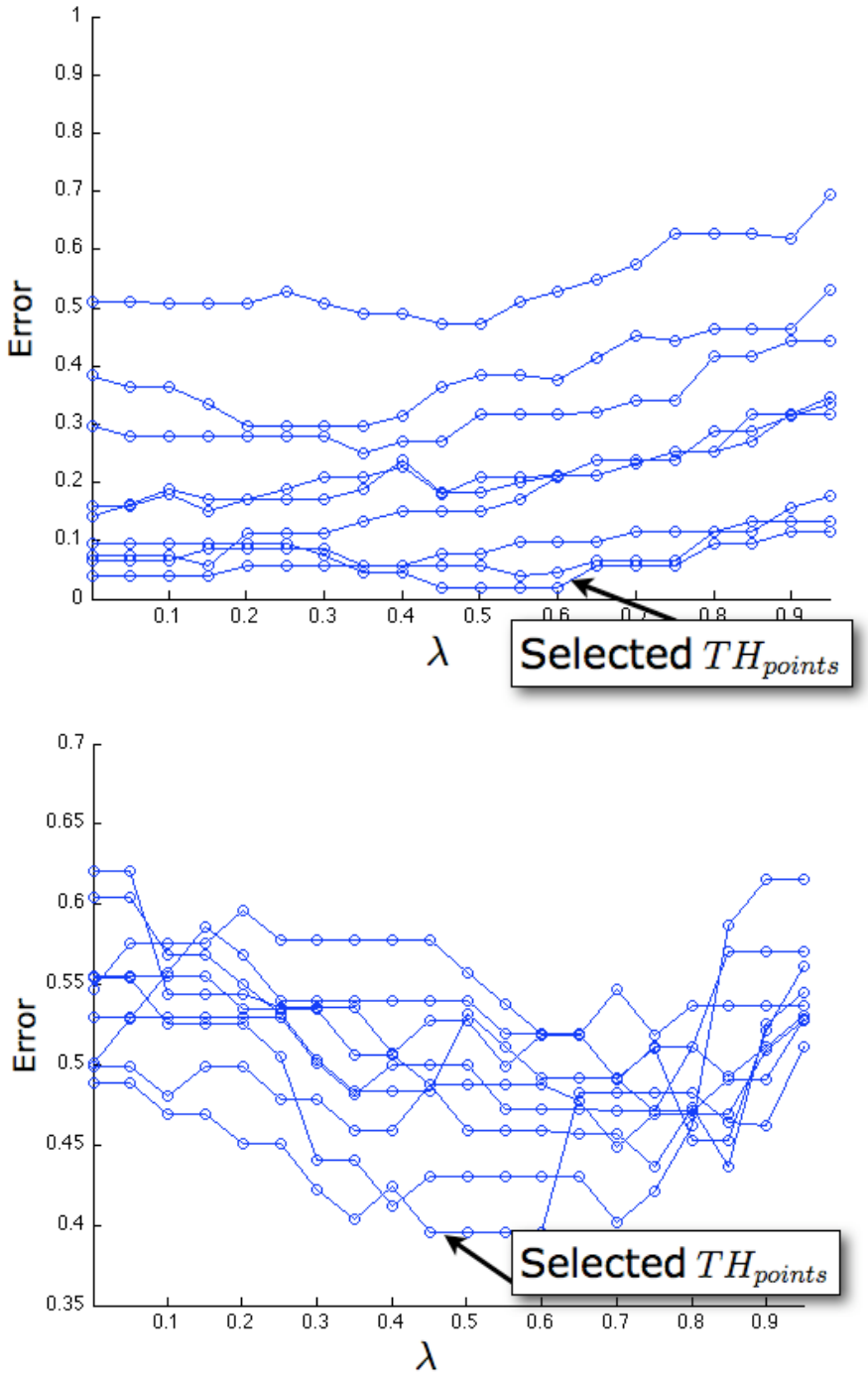


Figure 4.12: Error estimated during the 5-Fold cross validation for data set $RC5B$ considering the best performing alphabet, above, and the worst performing one, bottom. The different plots refer to different values of the cut threshold.

	W_P	W_S	W_M	W_D	TH_{points}	$\lambda(\times 10^{-3})$	Hit Rate
10	1	0	0	0	0.60	1	100
8	0.5	0	0	0.5	0.90	22.4	98
9	0.5	0.5	0	0	0.95	2.8	96.2
7	0.5	0	0.5	0	0.95	1	96.0
2	0	0	0.5	0.5	0.95	1	96.0
4	0	0.5	0	0.5	0.90	1	94.2
11	0.25	0.25	0.25	0.25	0.95	251.1	89.7
1	0	0	0	1	0.90	1	86.7
5	0	0.5	0.5	0	0.85	1	83.6
3	0	0	1	0	0.95	1	69.9
6	0	1	0	0	1	22.4	59.9

Table 4.3: RLS analysis on $RC5B$ exploiting the string-based representation with multi-cue integration.

system of behavior analysis is not only to describe a previously seen scenario, but also to classify as known new events or recognize then as new patterns, some information may be missed.
Let us consider a set of trajectories of behavior 10 and 11, from data set $RC11B$: they might be interpreted as realizations of behaviors 2 and 4 respectively, if the only positions were considered. The first row of Fig. 4.13, in fact, reports the similarity matrices obtained by comparing such trajectories against the representations of the 5 behaviors in $RC5B$ with respect to alphabet 10: the first column refers to trajectories from behavior 10, the second column from behavior 11, each row in the matrices corresponds to a test data, while each column refers to one of the known models. Each behavior has been represented by a string candidate selected via a voting strategy (that we will discuss more in details on Chapter 6); the similarity measure is based on the $2-$spectrum.
Notice that the new trajectories are recognized as instances of the known behaviors, because of the spatial overlap. However, the particular analysis application might require to discriminate among people walking or running - when monitoring wide areas, the fact that many people are running may indicate the presence of some danger. If we perform the same analysis by considering an alphabet where the position is weighted, as alphabet 8, the trajectories can not be associated to the known models, as shown on the second row of Fig. 4.13. This suggest that features that seem to be the most meaningful for a specific scenario on a particular data set, may be unreliable if data are changed. This is also testified by the results obtained by performing the same multi-class classification on data set $RC11B$, where the data consider different dynamics. In Table 4.4 the best performing alphabet is the one where *both* position and velocity module are considered. The *take home message*

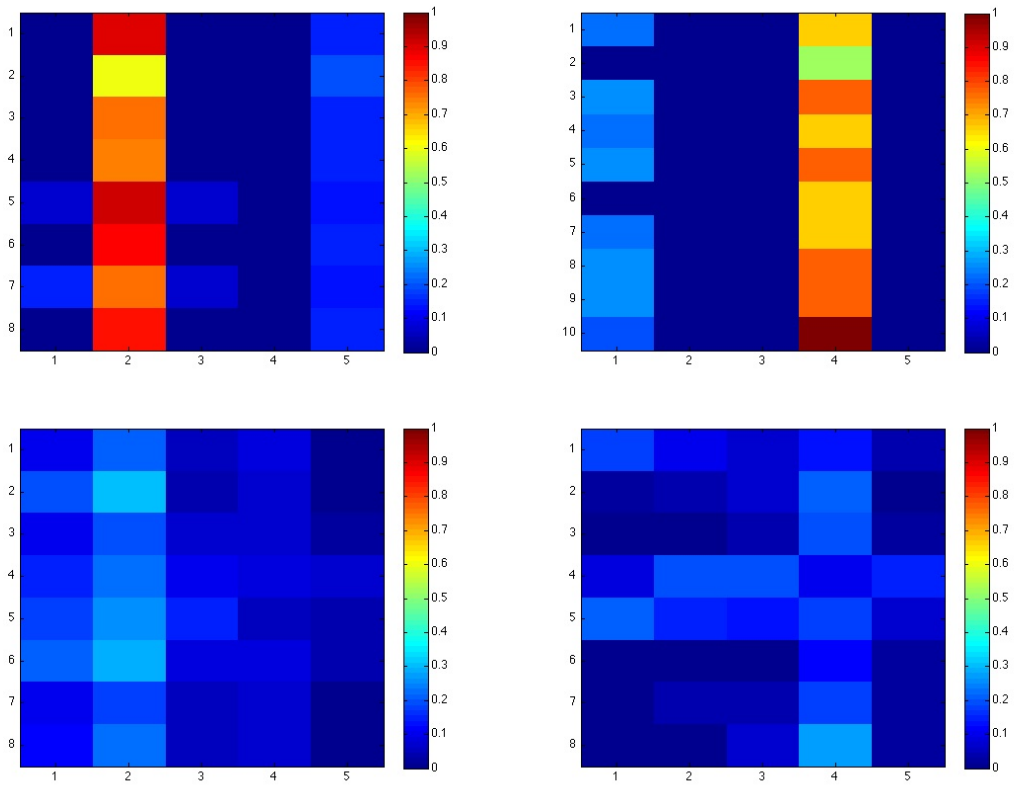


Figure 4.13: Similarity measures computed among the known models and a set of test trajectories that differ from the models for the dynamic properties (people walking vs people running). The first column refers to trajectories of behavior 10, spatially overlapped to behavior 2; the second column instead considers behavior 11, related from the standpoint of the positions to behavior 4. Thus, when only the positions are taken into account, the new trajectories are associated to known behaviors because of spatial overlapping (first row). When considering also the velocities, the differences among events arise (second row).

	W_P	W_S	W_M	W_D	TH_{points}	$\lambda(\times 10^{-3})$	Hit Rate
8	0.5	0	0.5	0	0.90	1	93.2
11	0.25	0.25	0.25	0.25	0.90	1	86.2
2	0	0	0.5	0.5	0.90	1	84.9
10	1	0	0	0	0.95	2.8	83.9
7	0.5	0	0	0.5	0.95	1	82.6
4	0	0.5	0	0.5	0.95	1	79.5
9	0.5	0.5	0	0	0.95	1	75.6
5	0	0.5	0.5	0	1	1	72.7
1	0	0	0	1	0.90	22.4	70
3	0	0	1	0	1	1	60.2
6	0	1	0	0	1	22.4	49.5

Table 4.4: RLS analysis on $RC11B$ exploiting the string-based representation with multi-cue integration.

is that when dealing with completely data-driven models it is very important that input data (or *training set*) is a reliable representative of the analyzed scenario. Alternatively, our system should cope with the problem by considering a set of potential models: in this case, the adoption of string-based representations which can constitute different views of the same data set - enhancing different properties - is a viable solution.

We conclude the section with a comparison of string-based representation and the representation schemes that resulted as promising in the analysis on synthetic data, HMMs and B-Splines fitting (Table 4.5). For what concerns B-Splines and strings, we consider position only and the whole feature vector. In the case of HMMs we do not include results on the whole feature vector since it shows convergency problems.
A first inspection shows that the performances decrease when moving from the simple $RC5B$ to the more complex $RC11B$. In the case of B-Splines, the advantage of using the whole information - rather than position only - on data set $RC11B$ is clearly visible. We can conclude from the global performances that there is a slight advantage in using strings.

4.6 Discussion

In this chapter, we explored the use of different representations schemes applied to the raw trajectories. More specifically, we compared three different schemes, including a feature mapping and an appropriate similarity measure: we proposed a string-based representation coupled with the P-Spectrum kernel, and compared it against HMMs with Probabilistic

Table 4.5: Comparisons among different promising representation schemes.

	Data set CR5B		**Data set CR11B**	
	2F	**4F**	**2F**	**4F**
Strings	100	98	83.9	93.2
HMMs	97.8	-	74.1	-
B-Splines	98	98.2	70.4	92.4

Product kernels, and curve fitting with the well-know Gaussian kernel. The experimental analysis relied on the use of supervised learning methods, and Regularized Least Squares in particular, and was based on both synthetic and real data for which the ground truth was available. The results showed the advantages of using strings with respect to the other schemes of representations.
For what concerns the criticality of choosing a particular input representation, we observed how the use of a more complex feature vector, other than the sequence of plain positions, allows us to achieve higher percentages of correct classification of temporal data characterized by a more complex internal structure (in our examples position and velocity module).

For these reasons part of the future work will be devoted to investigate how the choice of an appropriate feature vector can be derived from information on scenarios and applications. In the case of strings, we will also consider different methods of data partitioning with attention to the possibility of automatically updating over time the representations, thus the alphabet. In this case, some kind of technique for incremental clustering can be appropriate.

Chapter 5

Unsupervised behavior analysis

In this chapter we discuss a clustering-based approach to behavior modeling: each representation proposed in Chapter 4 is used to feed the Spectral Clustering, which is equipped with the corresponding kernel. The experimental analysis, based on the same data of Chapter 4, aims at evaluating the clustering results with respect to the ground truth. In essence we evaluate the correspondence among real behaviors (the ones arising from the ground truth) and estimated behaviors (the clusters). We design two different procedure to cluster association, to cope with the requirements of a typical system for behavior analysis. The comparison of adopting supervised and unsupervised learning is discussed. At the end, a conclusion in favor of the string-based scheme is reached.

5.1 Introduction

The supervised approach implicitly adopted in the previous chapter may not be appropriate to analyze common behaviors. In fact, the significant number of interesting behaviors, even when dealing with relative simple environments, requires *adaptability* and *robustness* and, also, sets conditions under which a, accurate ground truth of the data is difficult, if not impossible, to achieve.
Thus in this chapter we address the problem of modeling behaviors from an unsupervised perspective by setting out an analogy between behavior analysis and *clustering of temporal series*.
The key idea underlying our approach relies on looking for coherent groups of trajectories, among the training set, representing recurring patterns of activities, or, in other words, events commonly occurring into the scene. Under this view-point, each cluster groups trajectories which are instances of some *common behavior*.
We now summarize our approach on unsupervised behavior analysis. Fig. 5.1 shows a

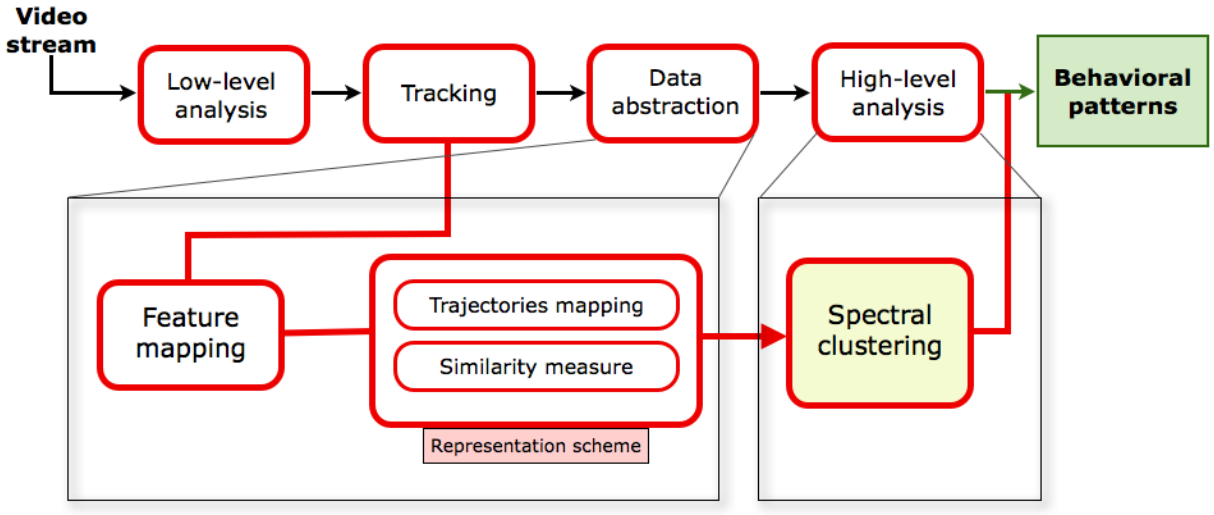


Figure 5.1: The visual representation of our pipeline to model behaviors.

visual representation of this pipeline. The representation schemes adopted to build the higher-level descriptions follow the discussion in Chapter 4.
The high level analysis we perform relies on *Spectral Clustering* [SM00], an efficient clustering method that address the problem of data partition by means of the spectral analysis of graphs: for an introduction see the Appendix.
Spectral Clustering starts by a similarity matrix computed on the available data. All the kernels discussed in Chap. 4 may be used as similarity measures, being *symmetric* and *positive*.
The choice of different parameters in the various representation schemes leads on this case to different data partitioning. For instance, in the case of strings, when the partition is built with a Multi-Cue Feature integration, this translates into the estimation of several alphabets and, consequently, different trajectories partitioning: in Fig.5.2 an example is reported where the clustering is made on representations derived from the positions.

High variety of the input space and future variations of the scenario might be better handled if more that one models are selected (we recall the discussion on Sec. 4.5.3 for an intuitive example). This should allows the system to perceive significant changes into the observed data, enabling a procedure to update the model, if possible. This would also call for a strategy to merge the results of all models.
The experimental analysis reflects the pipeline followed in the previous chapter, being based on the same data sets. To perform model selection and implicitly compare the clustering performances with respect to the goal of discriminating among behaviors, we devise a strategy to associate estimated behaviors (clusters) to the classes on the ground

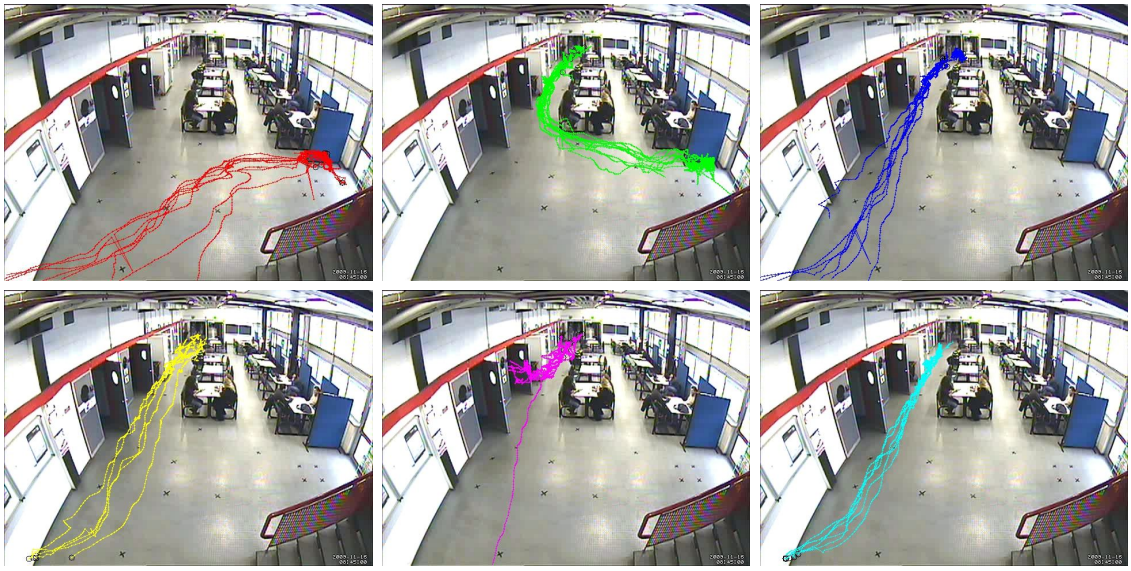


Figure 5.2: An example of trajectories partitioning: the alphabet on top of which strings are built considers only the position. As an effect, the trajectories clusters reflect this property.

truth. Some attempts to use quality indices within this framework are considered.

When dealing with unsupervised learning, model selection turns into a very complex problem, for which no solution valid in general has been proposed. Some discussion on this chapter will focus on this topic.
To evaluate a specific clustering, we thus experimented the adoption of different *quality indices* [MT09], measures proposed in the literature aiming at judging the goodness of a partition with respect to some criteria. They often refer to conditions related to *intra* and *inter* class similarities.

5.2 Temporal series clustering

In this section we carry out an analysis based on the data sets (both synthetic and real) described in Chapter 4. Since all the data are labelled in this case labels are used ad ground truth to perform model selection and report final results [HBV01, BA03, GB03]. More in general the issue of evaluating results from unlabeled data is not trivial. At the end of the section we will discuss the adoption of quality indices.
For the time being, we rely on the available ground truth. More precisely, we evaluate the

goodness of clustering results using the annotated labels to establish the correspondence between the true (annotated) behaviors and the estimated ones (i.e. the clusters). Inspired by the work in [MT09], where the *Correct Clustering Rate* is adopted to evaluate clustering results, we say that an estimated cluster C corresponds to a true behavior B if the most of elements laying in C are labeled (in the ground truth) as instances of B. By following this procedure, we can evaluate the results with respect to different granularity:

- With the *strict association to cluster* we assume that there exist a unique correspondence between true and estimated behaviors (a real behavior *must* correspond to exactly one estimated cluster)
- Using *loose association to cluster* we admit that a true behavior could be split to more that one estimated clusters (a real behavior *can* correspond to multiple estimated clusters).

Indeed, it is worth pointing our that in many applications of unsupervised learning one is not necessarily interested in assigning a class to "each" datum but to cluster coherent data, discarding noisy or uncommon patterns, see for instance [CV05]. This reason speaks in favor of the appropriateness of the second approach in our analysis. However, it might tend to favor over-estimated clustering instances in terms of number of classes. It is thus suggested, in some cases, to restrict the evaluations on the clustering results whose number of clusters lies inside an interval centered on the real number of clusters.

In the string-based approach this procedure is also implicitly adopted to select the best alphabet from the family computed when adopting the multi-cue features integration: in particular, the choice of an appropriate selection of weights allows us to obtain a sub-set of alphabets which can be appropriate for a given environment and adaptable to its stable changes. This choice is guided by the set of available data, assuming to carry all meaningful information on common behaviors. Being a completely data-driven approach, from the experimental analysis we carried out on real yet controlled data set, the hypothesis that the given training set is representative of the common behaviors we might be interested in recognizing is of fundamental importance and strongly affects the final modeling results.

5.2.1 Assessment on synthetic data

We first consider the synthetic data, $DS3B$ and $DS6B$.

We performed the analysis as follows: given one of the 20 resampling of the data set for a fixed amount of noise, we first run the spectral clustering varying the parameter related to the cut value, that we call TH_{trjs} to indicate that it refers to the clustering of trajectories. Notice that when adopting the string-based representation we have here 2 different cut thresholds:

	Data set DS3B				
Noise Level	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
Strings (automatic partition)	**99.9(3)**	**99.8(3)**	**99.5(3)**	**97.9(3)**	**95.4(3)**
Strings (manual partition)	**100(3)**	**99.8(3)**	98.5(3)	93.4(3)	89.8(3)
HMMs	89.0(3)	91.4(3)	80.7(3)	66.2(3)	55.2(3)
Polynomials	96.6(3)	88.4(3.4)	77.9(4.4)	68.2(5.6)	55.3(6.4)
Normalized polynomials	61.2(8.4)	57.9(8.1)	51.1(7.5)	50.4(6.5)	46.9(4.6)
B-Splines	93.5(3.4)	95.0(3.3)	94.5(3.3)	92.8(3)	82.7(2.7)

Table 5.1: Percentages of correct behavior strict associations on data sets $DS3B$, in round brackets the average (computed on the 20 resampling) numbers of estimated clusters are reported.

1. The first, TH_{points} is the one controlling the granularity of the partition when using Spectral Clustering to build the alphabet(s). As in the previous experiments, its values is selected from $[0.6, 1]$
2. The other, TH_{trjs} impacts of the solution of the trajectories clustering, which under this specific scheme consists in clustering strings. The value of this threshold is varied in the interval $[0.5, 1]$

We select the *strict association* result corresponding to the best performing experiment. We then average the 20 values, and the corresponding obtained average number of estimated clusters. We finally summarize the results in Tab. 5.1, for data set $DS3B$, and Tab. 5.2, for $DS6B$, where we highlighted in bold the best results for each noise level.

The above results provide experimental evidence that, in general, the approach based on the use of string kernel gives better results than the others. Manual partition seems to be slightly more accurate than the automatic method for moderate noise levels but, as the clutter increases, data-driven states estimate provides better and better performances, leading naturally to a system providing the ability to adapt to changes into the observed scenario. B-Splines provide comparable, even if significantly less accurate, performances. HMMs and polynomials perform very poorly as the noise increase, while normalized polynomials, even when dealing with synthetical and thus highly controlled data, are not able to cope with the behavior estimation task. As for the number of estimated clusters accuracy, there is a verification of the appropriateness of the string-based approach, which consistently estimates 3 classes. This result is achieved also by the HMMs, that, however, do not guarantee the same accuracy in the strict association.

Moving to bidirectional series (data set $DS6B$) the performances reflect the results just

	Data set DS6B				
Noise Level	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
Strings (automatic partition)	98.7(6.2)	98.7(6.2)	**99.2(6.1)**	**95.7(6.1)**	**94.9(6.1)**
Strings (manual partition)	**100(6)**	**99.8(6)**	**99.1(6)**	94.2(6.1)	89.0(5.9)
HMMs	51.7(5.7)	49.8(5.8)	45.5(5.95)	38.3(6)	38.7(6)
Polynomials	58.2(3.1)	52.7(4)	48.3(4.6)	42.8(5.3)	40.9(5.1)
Normalized polynomials	41.6(7.6)	39.9(8.5)	38.4(8.4)	38.2(6.7)	35.9(4.7)
B-Splines	94.55(6.8)	95.6(6.8)	95.5(6.5)	**96.0(5.9)**	73.1(4.4)

Table 5.2: Percentages of correct behavior strict associations on data sets $DS6B$. Inside the brackets we report the average number of estimated clusters.

discussed. The goodness of the string representation coupled with P-spectrum kernel is clearly confirmed, included the number of estimated behaviors. Also, B-Splines perform well, although the correspondence between real and estimated number of clusters is less precise and the strict association is strongly affected by the amount of noise is significant. HMMs and polynomials provide comparable estimations, decreasing rather quickly as the noise increase. Normalized polynomials is the worst-performing representation scheme.

5.2.2 Evaluations on a real data set

The experimental assessment on synthetic data lead us to conclude in favor of the string-based approach with automatic partition: we verify this impression on real data. The data sets $RC5B$ and $RC11B$, we introduced in Sec. 4.5, provides a natural test bed for the problem at hand: the annotation available allows us to select for each experiment the model best fitting it. Similarly to the analysis we carried out in the supervised case, we consider (1) the string-based representation with automatic partition and P-spectrum kernel, (2) the HMMs modeling on single trajectories compared with probability product kernel, and (3) the B-Splines fitting coupled with a gaussian kernel. In the case of HMMs, however, some problems on model convergence impeded the test with richer input vector. Moving to real but controlled data, we will try to conclude our analysis clarifying the following main issues:

- *Is there any advantage in using an heterogeneous input space, rather than the plain positions? If yes, under what conditions?*
- *Is clustering able to provide a robust tool to partition trajectories with the final aim of detecting frequent patterns of activity?*

- *Is there a representation scheme globally performing better than the others?*

To this purposes, we evaluate the performances of the schemes of representations plugged into the spectral clustering framework and associating each estimated behavior (clusters) to the real (annotated) sets. The percentages of correct association can be estimated by considering a one-to-one mapping between estimated and real behavior (the *strict association*). However, it is not always advisable to force such strict constraint, since in the unsupervised case there is no guarantee that the obtained data partitioning reflects the annotation given by the user. Instead, an evaluation where multiple association (that is, admitting that a real behavior can be split into multiple clusters) are allowed should be taken into account (*loose association*). A possible drawback of this procedure is that solutions with higher granularity (i.e. over-segmentation effects) might benefit from the multiple association. In this case we may consider a *loose association with a constraint*. This procedure can be applied if a ground truth on the real number of clusters, N_R, is available, to consider solutions clusterings where the number of estimated classes N_C is in a neighborhood of the correct number, that is $N_C = N_R \pm \Delta$. In our experiments, $\Delta = 1$.

	Strict		*Loose with constraint*		*Loose*	
	2F	4F	2F	4F	2F	4F
Strings	88.9 (6)	82.2 (6)	97.8 (6)	93.3 (6)	97.8 (6)	93.3 (6)
HMMs	62.2 (5)	-	77.8 (5)	-	82.2 (5)	-
B-Splines	84.4 (5)	75.6 (7)	84.4 (5)	71.1 (6)	84.4 (5)	82.2 (7)

Table 5.3: Evaluations of clustering-based classification of trajectories fro data set $RC5B$.

Table 5.3 reports the designed analysis on data set $RC5B$. It is apparent that the string-based representations allow for performances constantly better than the others. Instead it is not obvious whether an heterogeneous input space is better than plain positions in the image plane. However, such conclusion was rather predictable given the higher characterization of behaviors with respect to the position. The reported results of string-based have been selected by a further analysis considering the set of allowed alphabets: it is worth pointing out that in all 3 cases, the best performance when considering more than one feature corresponds to the alphabet where position and direction are weighted, testifying the importance of the latter in the given scenario and with respect to the given data set.
The reliability of clustering can be evaluated ignoring the constraint on the number of clusters, and simply selecting the partition associated to the highest correct association rate: the selection with strings does not vary, proving the robustness of this solution.

The analysis on data set $RC11B$ confirms the superiority of a string-based scheme, which produced the best correct association percentages coupled with a proper estimation of

	Strict		*Loose with constraint*		*Loose*	
	2F	4F	2F	4F	2F	4F
Strings	75.2 (12)	73.0 (10)	85.2 (12)	80.0 (12)	85.8 (12)	82.2 (12)
HMMs	57.6 (10)	-	70.6 (10)	-	78.8 (18)	-
B-Splines	58.8 (10)	55.3 (11)	68.2 (10)	67.1 (11)	68.2 (10)	81.2 (20)

Table 5.4: Evaluations of clustering-based classification of trajectories fro data set $RC11B$.

the number of clusters (and reliability when the evaluation is not restricted to solutions with proper number of clusters). B-Splines and HMMs highly over-estimate the number of clusters in absence of constraint. For what concerns the advantage of a richer input space, when adopting a completely data-driven solution, the good representation of the input space by means of a proper training set is a main requirements to successfully address the modeling problem. When dealing with slightly more complex data, as the ones in data set $RC11B$, it is important the characteristics we may be interested in highlighting are properly represented. In particular, in this second data set the velocity modules were sampled from a richer input space, since trajectories were produced by people walking or running, The latter constitutes a quite significant smaller set than the other. When we dealt with the supervised analysis, the availability of the labels which were plugged into the learning system allowed to compensate for such inequality. In the unsupervised framework, instead, the difference is clearer, as shown in Table 5.4. The advantage of using an heterogeneous input space, that was clear in the supervised analysis, does not hold. This suggests that a substantially equal presence of all the interesting properties of the data must be available into the training set, used to compute the model. This hypothesis is enforced by the results obtained when restricting the analysis pipeline to a subset of trajectories where running and walking people are equally present. In fact, in this case the performance of string-base descriptions increases from the 89.5% of correct association when only the position is taken into account, to 97.3% when also the velocity module is weighted.

5.2.3 The use of quality indices

In this section, we address the problem of evaluating the quality of a given data partition in the case a ground truth is not available. To this purpose we explore the use of quality indices to perform model selection on different clustering instances. We focus our analysis on string-based representations and specifically address the problem of selecting the best performing alphabet (i.e. the partition which leads to the best clustering results).
Let us first briefly analyze common approaches to the problem. The procedure of evaluating the results of a clustering algorithm is known as *cluster validity*. Three main approaches can

be adopted to investigate towards this direction [HBV01] by using some quality measures to evaluate the clustering results:

- The first one, based on *external criteria* requires the availability of a ground truth
- When using *internal criteria*, instead, the evaluation is based on quantities measured on the data themselves (e.g. quality indices, proximity matrix,...)
- Finally, the key idea of the so called *relative criteria* is evaluating the clustering structures against other clustering schemes, obtained e.g. with different values for the parameters, with respect to one of the measures used in the previous method.

The latter approach is the one which better adapts to our requirements.
When the number of clusters n is a parameter of the clustering method, then a common strategy to perform model selection can depend on a given quality index q and can be states as follows:

- Different values of n from a minimum to a maximum value allowed are a-priori decided
- For each one of such values, the algorithm is run r times using different values for the other parameters
- The best value of index q is selected for each value of n, and the analysis of the trend of such value can be an indication of the correct number of clusters.

To adapt this procedure to our clustering algorithm (that does not control explicitly the number of clusters), we fix the features weights and run the spectral clustering by varying the values of the cut thresholds TH_{points} and TH_{trjs}, selecting for each weights vector the best performing parameter. Then the analysis should lead to select the most appropriate alphabet for the behavior analysis problem.

As for the quality measures, we select a set of promising indices [HBV01], whose definitions have been adapted to handle string data. The distance adopted to compare two strings s and t depends on the P-Spectrum Kernel, and more precisely is defined as

$$D_K(s,t) = 1 - \hat{K}_P(s,t) \tag{5.1}$$

where $\hat{K}_P(s,t)$ is the normalized P-Spetrum kernels introduced in Sec. 4.3.2.
In the remainder of the section we adopt the following notation. With $\mathbf{S}$ we refer to the set of input trajectories converted into strings with respect to some alphabet. Each cluster C_i is represented with a candidate v_i by selecting one of its strings via a voting strategy

(see Sec. 6.5).
We define the variance σ_i of cluster C_i as

$$\sigma_i = \frac{1}{n_i} \sum_{k=1}^{n_i} D(s_k, v_i) \tag{5.2}$$

where n_i is the number of elements of c_i and $s_k \in C_i$ Similarly, we represent the whole input set $\mathbf{S}$ with a candidate string s and define its variance as

$$\sigma_{\mathbf{S}} = \frac{1}{n_S} \sum_{k=1}^{n_S} D(s_k, s) \tag{5.3}$$

where n_S is the number of input elements and $s_k \in \mathbf{S}$

Considering a partition P with n_p clusters, we thus adapt a set of known indices to the string case, according to the following:

Scattering

$$Scat(P) = \frac{1}{n_p} \sum_{i=1}^{n_p} \frac{\sigma_i}{\sigma_{\mathbf{S}}} \tag{5.4}$$

indicates the *Scattering*, that is the average compactness of clusters (i. e. intra-class distance). A small value for this index reflects compact clusters, as the scattering within clusters increases (i.e. clusters become less compact) it increases as well;

Total Separation

$$Dis(P) = \frac{D_{max}}{D_{min}} \sum_{k=1}^{n_p} \left(\sum_{z=1}^{n_p} D_K(v_k, v_z) \right)^{-1}, \tag{5.5}$$

where D_{max} is the higher pairwise distance between clusters candidates while D_{min} is the lower one, indicates the *Total Separation* between clusters (i.e. an indication of the inter-class distance). Such values is influenced on the geometry of the data and increases with the number of clusters.

SD-Index

$$SD(P) = \alpha Scat(P) + Dis(P) \tag{5.6}$$

defined the SD-Index which finds a trade-off between intra and inter class distances.
The two terms are in different ranges, thus the weighting factor α is needed to tackle the problem. When comparing different clustering results, in [HBV01] a value suggested for α is Dis_{max} corresponding to the Dis measure of the clustering with the maximum number of clusters.
The lower the value of this index, the better the partition P fits the data set.

Davies-Boulding Index

$$DB(P) = \frac{1}{n_p} \sum_{i=1}^{n_p} \max_{j=1...n_p, j \neq i} \frac{\bar{D}_K^i + \bar{D}_K^j}{D_K(v_i, v_j)} \tag{5.7}$$

where $\bar{D}_K^i$ and $\bar{D}_K^j$ are the average distances of strings from, respectively clusters C_i and C_j from the corresponding candidate. A low value for this index indicates the appropriateness of the clustering result.

Table 5.5 reports the values of the adopted indices for each alphabet. After a first inspection, it is apparent that they do not provide a viable solution towards model selection, as the alphabets they indicate as the best performing do not reflect the properties that, intuitively, should be enhanced. Also, the results of each one of them are unstable and clashing. A visual inspection of the results obtained when evaluating the partitions with respect to the ground truth, the last column of the table , confirms these considerations. Notice that, among the results of indices, alphabet number 10 has never been selected.
This evaluation leads us to conclude that the design of a strategy to perform model selection, as well as evaluate the clustering results, is needed when the availability of a ground truth is not guaranteed. In the next chapter, where we will be in the circumstance of dealing with high quantity of data, a strategy based on a *loose annotation* is preferred to the use of quality measures. The devised solution is inspired to the specific application domain.

Alphabet	**Wp**	**Wa**	**Wm**	**Wd**	**Scat**	**Dis**	**SD**	**DB**	**Hit rate %**
1	0	0	0	1	0.43	3.67	27.42	**0.19**	73.3
2	0	0	0.5	0.5	0.70	2.42	**12.90**	0.48	80.0
3	0	0	1	0	**0.33**	2.48	260.81	**0.19**	64.4
4	0	0.5	0	0.5	17.51	3.73	522.49	0.63	80.0
5	0	0.5	0.5	0	1.036	2.00	4955.44	0.44	71.1
6	0	1	0	0	19.08	**1.35**	155.31	0.25	66.7
7	0.5	0	0	0.5	0.74	2.51	16.04	1.09	93.3
8	0.5	0	0.5	0	0.85	2.63	73.27	0.72	82.2
9	0.5	0.5	0	0	8.21	2.63	769.38	0.67	77.8
10	1	0	0	0	1.38	2.04	86.11	0.45	**97.8**
11	0.25	0.25	0.25	0.25	3.77	4.72	52.87	1.18	77.8

Table 5.5: Quality measures on the family of alphabets computed for different values of the parameters. The last column reports the hit percentages obtained when evaluating the partitioning results using the ground truth.

5.3 Discussion

We addressed in this chapter the issue of extracting information from sets of unlabeled temporal data by means of unsupervised learning. We completed the pipeline for behavior analysis, started in the previous chapters, proposing to plug the intermediate representations into a clustering step, based on Spectral Clustering. The experiments were designed similarly to Chapter 4 so to explore the unsupervised counterpart of the learning problem. The key idea during the evaluation was to interpret the clustering results as an association tool between estimated and real clusters. Even in this case the predominance of strings is apparent. By looking at real behavior analysis applications, where in the most cases the annotation is not available, we experimented the use of quality indices to evaluate the clustering results. We conclude that such techniques are not tailored for our setting.

The main open problem relates to how to better exploit the complexity of the input data in the unsupervised setting. We have seen from the experiments in this chapter that when the data are heterogenous but the classes are not equally represented (in the data set $RC11B$ trajectories of people running were in a severely lower number that the ones of people walking), a feature vector which might have the potential of well describe the data (in our case, a description considering spatial and velocity properties) can not provide the expected result. We thus aim at better clarifying the mechanisms that govern such relation, trying to define conditions under which specific properties of the input data can be modeled with an unsupervised approach. The analysis of the input feature space could be of help.

Chapter 6

Application to a large scale scenario

In this chapter we propose a behavior modeling system based on the use of strings as meta-descriptors for the trajectories. The behavioral patterns are estimated by means of Spectral Clustering, according to the general pipeline discussed on the thesis. We evaluate the system with an extensive experimental analysis on data acquired by a video surveillance system over a temporal range of a few weeks (Sec. 6.1). During the model selection phase,since manual labeling of big training sets is not advisable, we adopt an automatic loose annotation based on the environment physical properties. We then select as model, the best performing clustering result with respect to the loose annotation. For completeness, we compare the system against the other representation schemes, including the supervised analysis. The second part of the chapter (Sec. 6.5) is the place where we present the test analysis, performed again on data of weeks: two different out-of-sample strategies allows to associate a new test data to one of the known models or detecting it as an event different from the "normality" of the scene.

6.1 Introduction

In this chapter we go through the whole pipeline of our system for learning behavioral patterns. The analysis on synthetic and real but controlled data, carried out in Chap. 4 and 5, provides evidence of the superiority of the string-based approach coupled with the P-spectrum kernel to compare data. We thus exploit such representation on the system, which is visually represented in Fig. 6.1. Notice how such representation contains the main building blocks discussed on the thesis and now better specified with respect to our final choices.

We briefly sketch here the main steps of the procedure, which includes two main phases. During the *batch training phase*, the behavioral models are estimated by first extract-

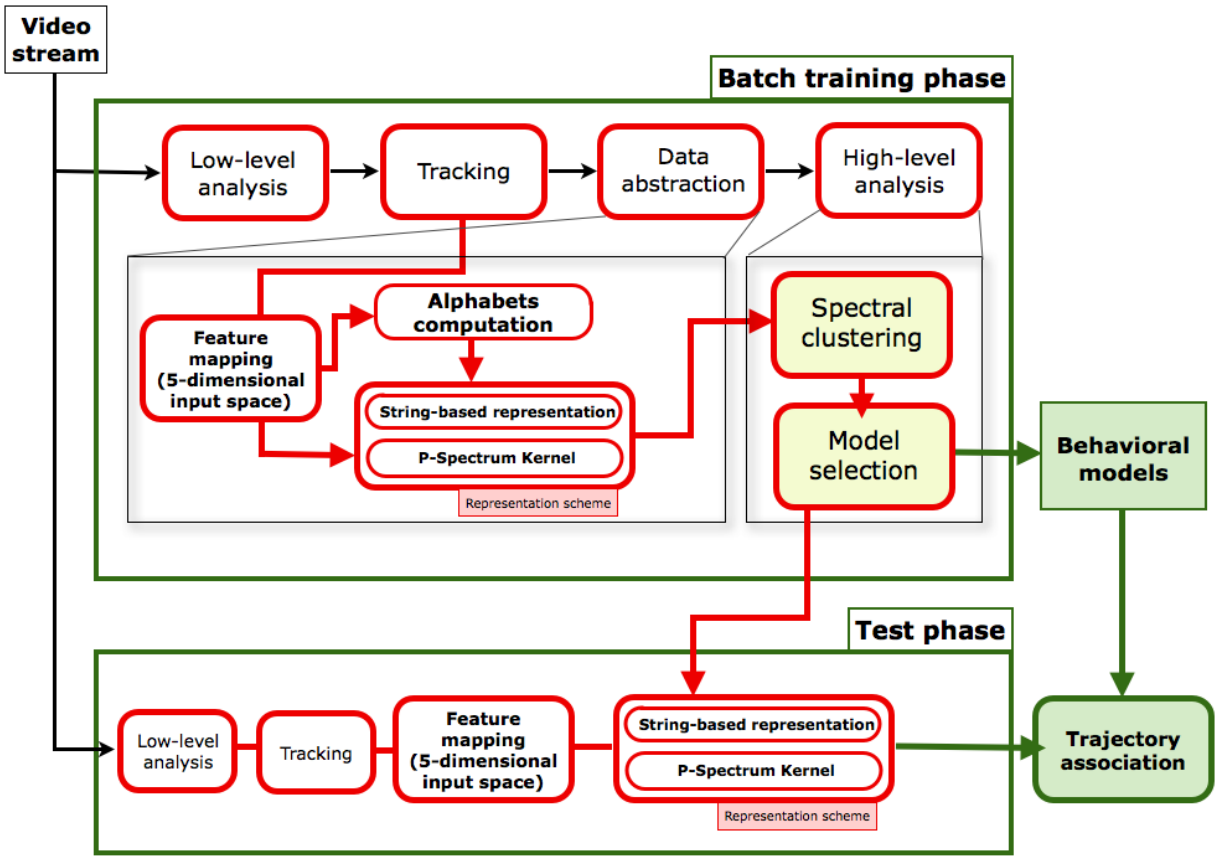


Figure 6.1: A visual representation of our system for behaviors modeling.

ing temporal information on dynamic events that are represented with respect to the 5-dimensional input space which considers position, size and velocity of each target. Then, the alphabets to compute the string-based representations are automatically estimated by means of clustering on the input points.
The model selection phase allows us to keep only the partitions better fitting the input data set with respect to a very loose manual annotation based on prior knowledge properties. When it is the case, e.g. when the confidence associated to the solution is low, it might be an option to consider different alphabets, which in our settings translates in building more than one model on the same data set with slightly different view points.
Given one of the selected partitions, the trajectories are represented on top of it as strings, that are finally clustered to detect the behavioral models.
During the *test phase*, once a new events is observed, it is translated into string accordingly to the built model (i.e. the selected alphabet) and the similarities with the known models are estimated. A procedure to associate the new event to one of the known behavior is

designed so to consider also the possibility that an event remains undefined or classified as anomalous.
This chapter is organized as follows. In the first part, an extensive evaluation of the string-based representation is performed on data spanning a week of acquisition from a real surveillance system. For completeness, we compare the string-based representation against the other schemes we considered in the experimental analysis of the previous chapters, Hidden Markov Models and B-Splines. Also, for the only strings, we compare the use of multi-cue integration against the adoption of the plain full vector of features. The conclusions will lead to an effective model selection.
With the second part, we evaluate the behavioral models on test data, spanning a week as well. In the last part, the anomaly detection is discussed.

Figure 6.2: Sample frames acquired by the surveillance camera, showing the potentially complex conditions of the observed environment: the crowd level differs depending on the temporal interval considered, while the illumination is strongly affected by physical properties of the environment, in particular the presence of windows along the right wall.

6.2 The video surveillance setting

The data we consider have been collected by means of a real video surveillance system[1], we briefly introduced on Sec. 4.5. We summarize its properties here, with a more detailed discussion on the main properties of the setup.

The system monitors an indoor open space (one of the main halls of our Department), shown in Fig. 6.2, where a good amount of dynamic events occur during daytime. Only

[1]The Imanalysis suite, we obtained within a technology transfer program with the company Imavis srl, `http://www.imavis.com/`.

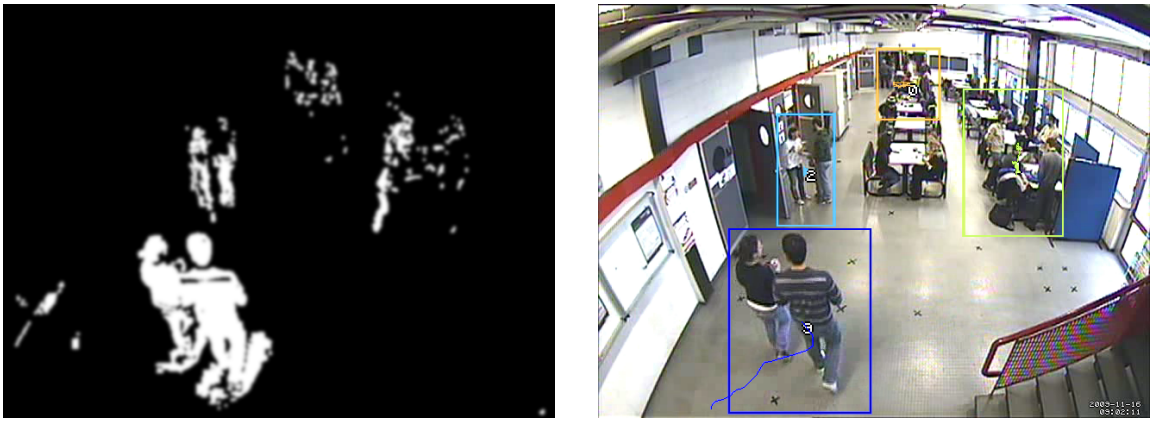


Figure 6.3: An example of the resulting binary map after the low-level video analysis: the connected components (left) are first extracted and described with a features vector, then correlated over time (right) by means of tracking to model their dynamic evolution in the scene.

people are supposed to be moving in the scene: the monitored environment provides different complexity with respect to the crowd level, which in turn depends on several factors, such as day, temporal interval, period of the academic year (presence of lessons, examinations). The weather conditions strongly affect the scene appearance (see Fig. 6.2), being the hall illuminated by windows (all along the right wall) as well as artificial lights. Accordingly to the video processing step, we discussed on Chapter 3, at each time instant, first the current frame has been segmented with respect to motion information (Fig. 6.3, left), then the tracking procedure is applied (Fig. 6.3, right) to build the trajectories and map them into the dynamic events descriptors, discussed in Sec. 4.5.1. This scenario summarizes in a sense the peculiarities of the benchmark data we adopted in Chapter 3 to evaluate our tracking procedure. Thus, the main causes of failure in that case are present

Starting time	Ending time
08:45	10:00
12:30	14:30
16:00	17:00

Table 6.1: Temporal ranges where video data have been considered for the experimental analysis.

also here. The robustness of our solutions allows us to appropriately cope with the problem of gathering trajectories, exploiting the fact the camera continuously acquires the video signal.

In order to collect the training set we extracted trajectories from observations of a week and to obtain a meaningful data set, we select the temporal ranges listed in Tab. 6.1. The dynamic content of the included peak time intervals is highly representative of the totality of events typically occurring in the scene (e.g., people arriving and leaving, entering and exiting doors, studying or chatting by the desks).

The low-level video processing phase is applied to all videos within the selected time spans, then an automatic pruning procedure is applied to build the training set:

- First, we discard trajectories shorter than a minimum length L_{min}, as we assume being generated by noise. The threshold must be accurately chosen depending on the specific scenario, to avoid the exclusion of short trajectories corresponding to interesting events (e. g. people running or spanning a limited spatial extent). Since the camera view of our setup is rather wide, the trajectories tend to be of significant length, so in our experiments we set $L_{min} = 100$;
- Then, an analysis on the trend of trajectories is performed discarding those containing many high jumps – a jump is a gap between two consecutive observations usually caused by tracking failures. We estimated the distance (gap) G between two consecutive observations, O_t and O_{t+1} as

$$G(O_t, O_{t+1}) = 1 - K(O_t, O_{t+1}) \tag{6.1}$$

  where K is the convex kernel discussed in Sec. 4.5.2.

This coarse procedure for data pruning aims at discarding extremely noisy trajectories, so to count during the training phase on a reasonably accurate data set. However, the presence of noisy trajectories is not a major problem for our setting: since clustering captures the global structure in the data, the noisy trajectories, if not prevail on the set, will be "absorbed" by the clusters.
Such intrinsic complexity of the data naturally leads to a highly challenging test bed for our pipeline. We collect the training set shown in Fig. 6.4, left.

The same considerations hold when gathering the test set, built on acquisitions of a week as well. However, as opposite to the modeling phase, where some kind of control on the data, even if rather limited, is applied by means of the coarse cleaning procedure, during the acquisition of the test data we mimic the work of a real surveillance system: in this case, in fact, the goal is to monitor each event occurring into the scene to associate it to some known model or detect it as abnormal event. We thus decide to only apply the first step of the cleaning procedure to reject very short trajectories, keeping all the other, highly heterogeneous, events. Fig. 6.4, right, reports the 5727 trajectories included in the test set, giving an impression of the richness of the observed events.

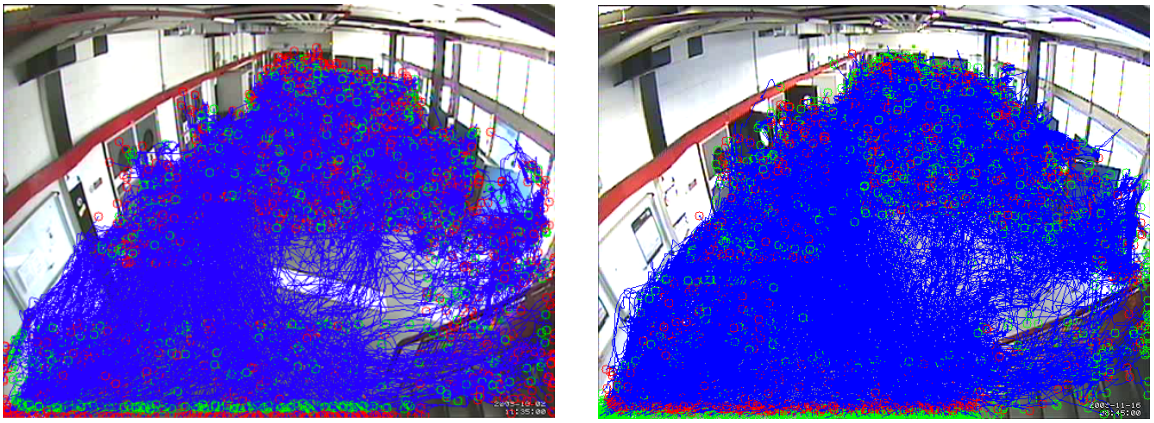


Figure 6.4: Data used in the experimental phase: from the left, training and test set. The trajectories have been acquired during two weeks considering a set of interesting temporal ranges and pruned to reject highly noisy trajectories. Red and green dots indicate, respectively, first and last points.

6.3 The loose annotation

As anticipated, the evaluation of the results of unsupervised learning is a highly challenging problem, for which no general solution has been found. When dealing with clustering, in absence of a ground truth, some quality measures might be adopted. However, as discussed on Sec. 5.2.3, this approach is not appropriate for our setting.
At the same time, the manual annotation of thousands dynamic data is difficult and very subjective. We propose a viable solution to *annotate the observed environment* on the basis of its physical properties, by detecting interesting regions corresponding to doors, tables, coffee urns and drinks dispensers and so on. This intuitive operation can be easily performed by a user which is asked to select such regions on a snapshot of the scene. The environment annotation we adopted for the experiments in this chapter is reported in Fig. 6.5, left. Such regions can be interpreted in fact as possible source and/or sink regions for trajectories observed in the scene. Following this idea, it can be argued that the manual annotation of the environment induces a loose annotation of the data, which are grouped with respect to source and sink points. The term *loose* refers to two main aspects:

- Since the coherence criteria depends only on first and last points of trajectories, very different patterns can belong to the same group. Let us consider the regions in Fig. 6.5, left. The group of coherence induced by the path $R_3 \rightarrow R_9$ may include trajectories of people going directly from region 3 to region 9, as well as motion of subjects moving from region 3 to region 2 and, finally, region 9;

- The manual annotation reflects the spatial properties of the trajectories, thus from

the point of view of the other features (target size, velocity expressed in terms of module and direction of motion) the obtained groups are heterogeneous. As an example, we do not classify moving entities in different classes, so that a trajectory can correspond to a single person as well as a group of people moving close each other. Even the direction, that in principle might appear strictly related to the position, can have strong variability among a single annotated behavior.

Starting from the trajectories shown in Fig. 6.4, left, we considered the groups of coherence induces by the regions annotation and discarded the less populated. We thus ended up with 8 general behaviors, for a total of 1205 trajectories (Fig. 6.5, right). The behaviors are detailed in the table of Fig. 6.5 and visualized in Fig. 6.6.

Behavior	Source regions	Sink regions	# trajectories
1	1	9, 10	281
2	1, 2	8	165
3	5, 6	7, 8	96
4	6	5	51
5	8	1, 2	210
6	8	7	104
7	9, 10	1, 2	215
8	9, 10	8	83

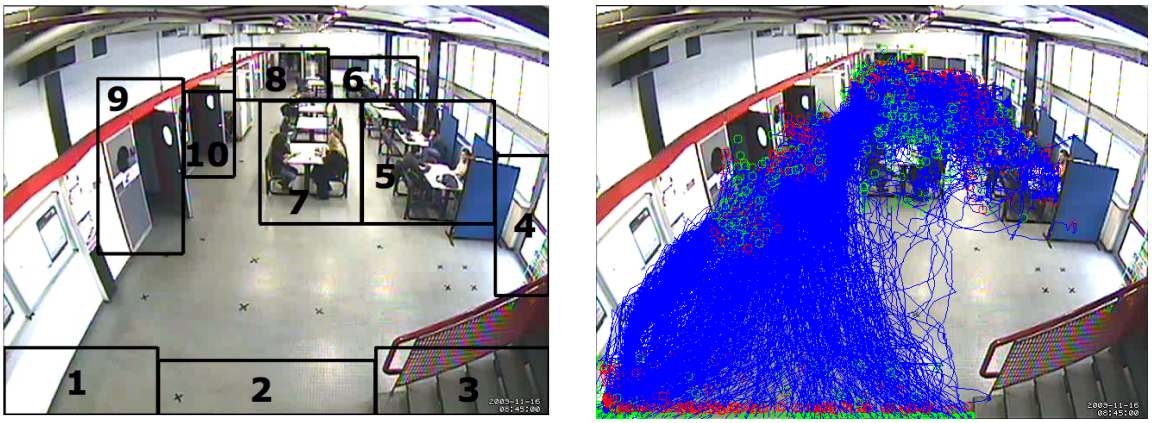


Figure 6.5: Above, the annotation of dynamic events with respect to the set of source and sink regions shown below, on the left. On the right, the resulting 1205 trajectories. Red and green circles denote, respectively, first and last points.

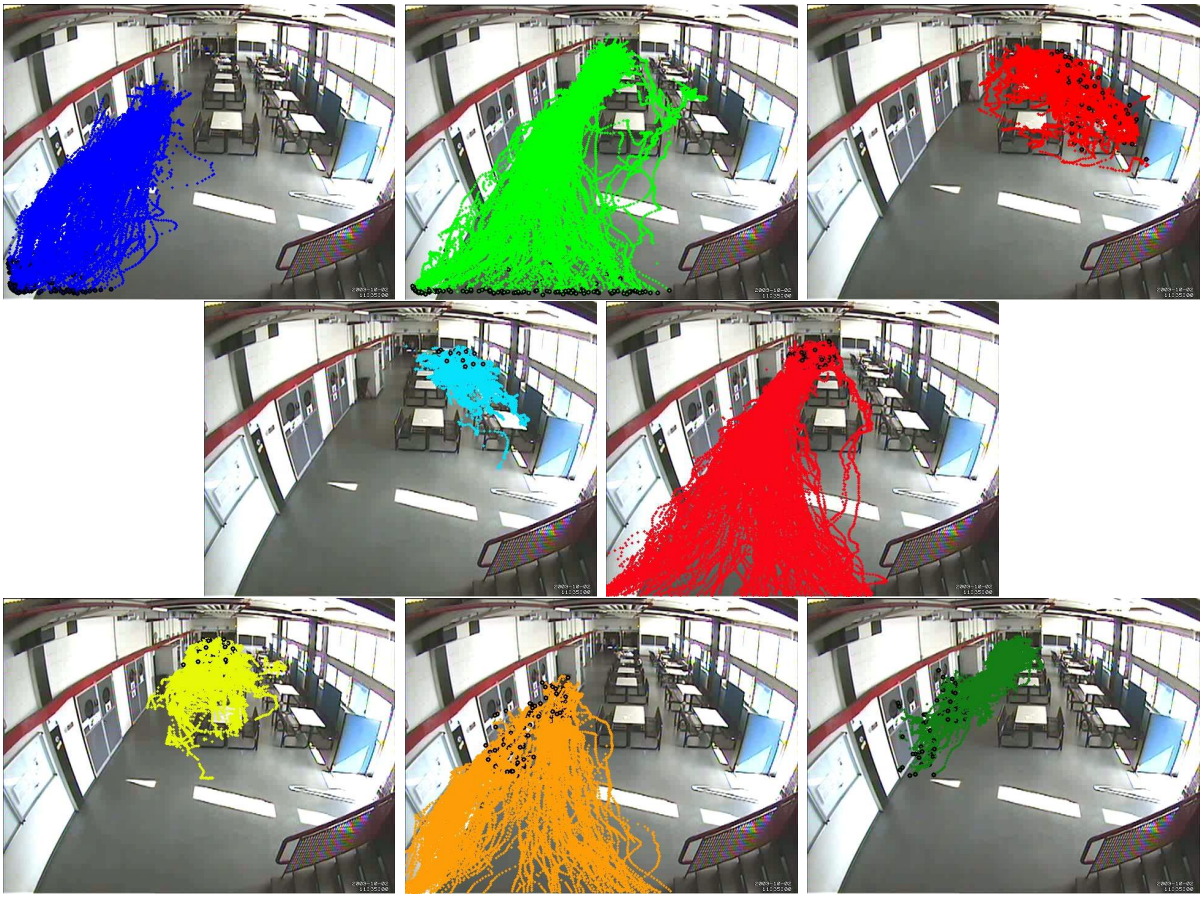


Figure 6.6: The 8 behaviors resulting after the annotation process.

6.4 Model selection

In the context of the experimental analysis of this chapter, if not different stated, all the assumptions on settings and choice of the parameters still hold. In the case of the string-based representation, we test 2 different settings during the computation of the alphabets. To combine different features we consider

1. A simple concatenation of the features in a vector, after that a normalization made the features comparable, and
2. The previously multi-cue integration,where each feature is processed singularly.

We also extend the family of alphabets we consider according to the following: the allowed combinations of features weights verify

$$\mathbf{W} = \cup_k \{[W_P, W_S, W_M, W_D] | W_i \in \{0, k\}, \forall i \in \{P, S, M, D\}\}$$

where $k \in \{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\}$ and $\sum_i W_i = 1$.

6.4.1 Analysis in the supervised case

We now evaluate the results obtained with RLS analysis. A supervised approach has the capability of learning some properties of the data on the basis of the a-priori knowledge introduced in the framework by means of the labels. This naturally leads to model highly dependent of the input data. In our case, this could be somehow troublesome since input labels correspond to a loose annotation that is not accurate. It is highly subjective and only partially describe the data, i.e. only some interesting properties of the data might be captured since it mostly relies on the positions. We thus just consider the supervised results as a crosscheck of our expectations.

In the string-based approach we first need to estimate the most appropriate alphabet with respect to the different possible weight vectors $\mathbf{W}$ for the multiple cue integration of features. Table 6.2 reports the average hit rates obtained by 5-CV for a selection of weights. It is apparent how the presence of position leads, in this case, to the best results, with the highest performance obtained by for $\mathbf{W} = (1, 0, 0, 0)$. This is mainly due to the dependence of the labels from the positions, as before mentioned, so that the learnt model reflects the properties used to feed the learning algorithm.

Table 6.3 compares the best result obtained with a string-based approach against the performances given by B-splines. By using the representation schema based on B-spline, the performances are slightly higher.

It also reports a comparison between feature concatenation and multi-cue integration for the string-based case, highlighting a clear superiority of the latter.

In order to deal with HMM-based representation, we start from a 2-dim representation based on the positions only and use a standard approach of estimating one probabilistic model for each class, and then using the models as a composite classifier. Given a new trajectory, the output label can be assigned by computing the likelihood to belong to all the models and then choosing the one that maximizes such likelihood.

Similarly with the previous 5-fold learning method, for the training of the HMM models we divided the data set into 5 distinct subsets and run the inference algorithm for each of them, using the others as test set. The average performance is 64.98%, which is below the performances of the methods reported in Table 6.3. However, by considering the confusion matrix (Fig. 6.7) of this classifier, one can easily see that the some of classifiers – those corresponding to behavior 1, 4, 5 and 8 – achieve extremely good performances while most of the errors are done by the others; more specifically those corresponding to behaviors 2, 3 and 7 (a visual inspection to Fig 6.6 suggests that these behaviors are not well discriminated

Alphabet	W_P	W_S	W_M	W_D	TH_{points}	%
10	1	0	0	0	0.95	67.48
13	0.33	0	0.33	0.33	1.00	64.33
8	0.5	0	0.5	0	1.00	63.91
7	0.5	0	0	0.5	0.95	63.9
9	0.5	0.5	0	0	0.95	63.64
11	0.25	0.25	0.25	0.25	1.00	62.58
14	0.33	0.33	0	0.33	1.00	62.5
15	0.33	0.33	0.33	0	1.00	62.16
2	0	0	0.5	0.5	0.95	52.70
12	0	0.33	0.33	0.33	0.95	52.54
4	0	0.5	0	0.5	0.95	50.29
1	0	0	0	1	0.90	48.38
5	0	0.5	0.5	0	0.95	41.18
3	0	0	1	0	0.95	38.68
6	0	1	0	0	0.90	24.57

Table 6.2: Hit rates in the string-based approach with a multi-label classification based on RLS.

Representation schema	**Success rate (%)**
strings (concatenation)	27.97
strings (multi-cue integration)	67.48
B-Spline (5 dim)	71.93
B-Spline (2 dim)	**72.26**

Table 6.3: Supervised analysis based on RLS multi-label classification: comparison between methods.

to others).

From the obtained results we may conclude that in the supervised case a B-spline abstraction module fitting the sole positions is the most appropriate data abstraction for the considered supervised scenario. However, as we will see below for the unsupervised scenario, such representation approach tends to be influenced by local properties of the data set, thus producing a large number of clusters consistent with the labels. More extensive experiments are needed in order to verify the robustness of the approach in the supervised case with respect to the increase of observed behaviors.

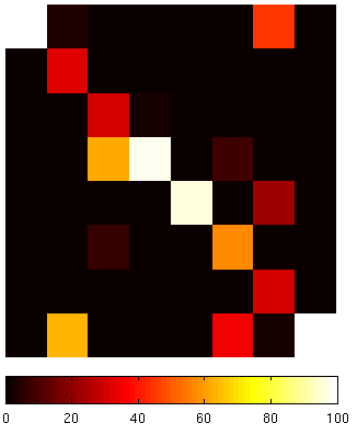


Figure 6.7: Confusion matrix relative to the classifiers obtained by training on HMM model for each behavioral pattern. The classes are ordered according to table in Fig. 6.5.

6.4.2 Analysis in the unsupervised case

In the unsupervised case we evaluate the goodness of the results using the labels as a ground truth to establish the correspondence between the true (annotated) behaviors and the estimated ones (i.e. the clusters), accordingly to the *strict* and *loose* association described in Sec. 5.2. In the string-based approach this procedure is also adopted to select the best alphabet from the family computed when adopting the multi-cue features integration: in particular, the choice of an appropriate selection of weights allows us to obtain a sub-set of alphabets which can be appropriate for a given environment and adaptable to its stable changes. This choice is guided by the set of available data, assuming to carry all meaningful information on common behaviors.

Our first experiment for the unsupervised case relies on performing such association assuming that each real behavior can correspond to just one estimated cluster (above, we called such approach *strict clusters association*). Since this could discourage solutions with higher numbers of clusters, we fix a constraint on the number of clusters by using the prior on the true number N_T ($N_T = 8$ according to our loose annotation): we admit in the final evaluation only solutions whose estimated number of clusters N_C is in $[N_T - \delta, N_T + \delta]$ where $\delta \geq 0$ is an integer number.

In Tab. 6.4 we report the performances of the family of alphabets computed when adopting the multi-cue integration: the alphabet which results to perform the best is the number 13, based on partitioning the input space by considering position and dynamic information of targets. The corresponding performance is then selected and compared against the other approaches in Tab. 6.5. Although the highest recognition rate corresponds to the

Alphabet	W_P	W_S	W_M	W_D	TH_{points}	TH_{trj}	# Clusters	%
13	0.33	0	0.33	0.33	0.95	0.6	7	65.80
10	1	0	0	0	1	0.5	7	59.00
14	0.33	0.33	0	0.33	0.9	0.7	7	55.60
7	0.5	0	0	0.5	0.95	0.5	7	55.51
2	0	0	0.5	0.5	1	0.75	7	52.61
1	0	0	0	1	1	0.85	6	50.53
11	0.25	0.25	0.25	0.25	0.95	0.85	8	49.37
12	0	0.33	0.33	0.33	0.9	0.75	7	49.29
4	0	0.5	0	0.5	0.95	0.9	8	48.96
3	0	0	1	0	1	0.9	7	34.85
5	0	0.5	0.5	0	1	0.7	6	21.90
6	0	1	0	0	-	-	-	-
8	0.5	0	0.5	0	-	-	-	-
9	0.5	0.5	0	0	-	-	-	-
15	0.33	0.33	0.33	0	-	-	-	-

Table 6.4: Strict association rate for the alphabet computing with multi-cue integration, $\sigma = \frac{DIAM}{10}$ is computed independently for each feature.

	# Cluster	%
String (concatenation)	2	34
String (multi-cue integration, $\sigma = \frac{diam}{10}$)	7	65.80
B-Spline (5 dim)	6	62.20
B-Spline (2 dim)	6	**67.94**
HMM (2 dim)	**8**	62.29

Table 6.5: Summary and comparison among representation schema for the strict association of dynamic events.

B-splines representation computed in the 2-dimensional input space, one can notice that string-based and HMM-based representations better estimate the true number of clusters. The result obtained when the string-based representation is made upon an alphabet built on observations which are the plain concatenation of features is very poor: the spectral clustering fails in splitting the data in more than 2 sub-groups, testifying the poor capability of the representation in characterizing the data-set. Again, this result speaks in favor of the advantages of Multi-Cue Integration.

We now evaluate the results with respect to the loose annotation, more appropriate to our

Alphabet	W_P	W_S	W_M	W_D	TH_{points}	TH_{trj}	# Clusters	%
13	0.33	0	0.33	0.33	0.95	0.6	7	76.18
14	0.33	0.33	0	0.33	0.95	0.6	8	73.36
7	0.5	0	0	0.5	0.95	0.5	7	72.36
11	0.25	0.25	0.25	0.25	0.95	0.85	8	69.95
2	0	0	0.5	0.5	0.9	0.75	8	69.46
10	1	0	0	0	1	0.5	7	66.39
12	0	0.33	0.33	0.33	0.95	0.85	8	65.47
4	0	0.5	0	0.5	0.95	0.9	8	62.57
1	0	0	0	1	1	0.95	8	56.26
3	0	0	1	0	1	0.9	7	45.89
5	0	0.5	0.5	0	1	0.7	6	44.56
6	0	1	0	0	0.6	0.5	2	23.31
8	0.5	0	0.5	0	0.6	0.5	2	23.31
9	0.5	0.5	0	0	0.6	0.5	2	23.31
15	0.33	0.33	0.33	0	0.6	0.5	2	23.31

Table 6.6: Loose association rate for the alphabet computing with multi-cue integration and $\sigma = \frac{DIAM}{10}$ for each feature.

setting, and without considering the constraint on the number of clusters, since our loose annotation cannot be compared to a real and accurate ground truth. Table 6.6 reports the obtained results.

Consistently with what previously observed, alphabet 13 performs better that the other. Notice that the very good performance provided by alphabet 7 (position and direction) is furthermore improved by adding the size (alphabet 14) or, more convincingly, the velocity module (alphabet 13). These considerations suggest that, on the specific input data-set that we considered, position and direction are the most semantically meaningful feature, followed by velocity module and size.

We conclude our analysis with the comparisons among different methods against the loose association, reported in Tab. 6.7. When relying only on the percentage of correct associations, B-spline fitting results in an unreliable schema, highly overestimating the correct number of clusters. The performance of HMM and string-based representations (the latter with Multi-Cue Integration) are comparable.

Fig. 6.8 (similarity matrices computed on the ordered data-set with respect to the 3 data representations) confirms this analysis: B-spline tend to over-estimate similarities and then over-segment data; string-based approaches slightly underestimate similarities, but capture the expected diagonal block structure; HMM appear to under-segment data and miss the

	# Cluster	%
String (concatenation)	2	34
String (multi-cue integration, $\sigma = \frac{diam}{10}$)	7	76.18
B-Spline (5 dim)	30	83.38
B-Spline (2 dim)	31	91.94
HMM (2 dim)	8	74.30

Table 6.7: Summary and comparison among representation schema for the loose association of dynamic events.

expected block structure.

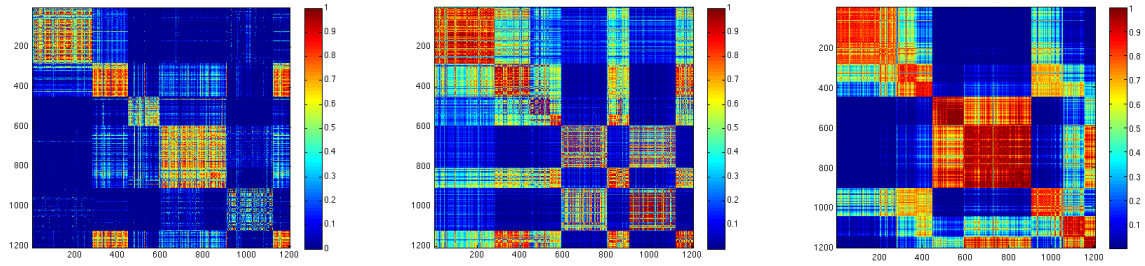


Figure 6.8: Similarity matrices. Form left to right: best alphabet for string-based representation with multi-cue features integration ($W_P = 0.33, W_S = 0, W_M = 0.33, W_D = 0.33, TH_p = 0.95, TH_t = 0.6$; B-splines based on 5 features; HMM-based representations on a 2-dimensional input space.

Finally, Fig. 6.9 and Fig. 6.10 report the estimated behaviors for string-based and HMM. It is clear in both cases they do not completely match the given annotations, but seem to reflect the fact the discriminative power of close range observations is higher than the one of far observations. This is not surprising, and simply suggests how, given the complexity of the scenario, a multi-camera video-surveillance system would be appropriate.
After a quick view of the results it might be apparent that some models appear to be rather similar, e.g., the first, the third, and the fourth from the left in Fig. 6.9, showing the results of string-based representation with multi-cue integration. Notice, however, that the visualization is unfair since it favors visual similarities among positions. An analysis on the other features better highlights the difference between clusters. Considering for examples the velocity modules for the same results when using string, shown as a box plot in Fig. 6.11: they show how apparently similar behavior are actually associated to rather different velocity features.

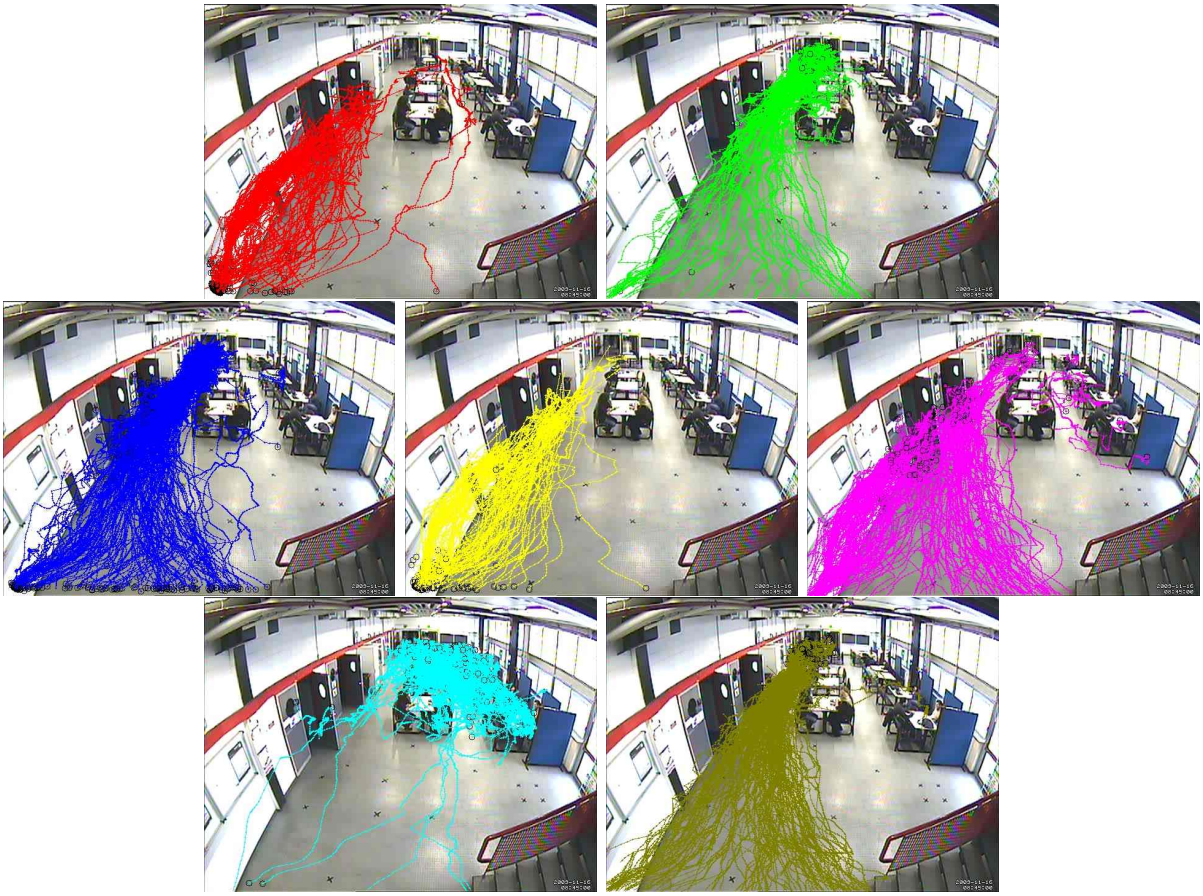


Figure 6.9: Best clustering results obtained with the string-based representation schema with the loose evaluation. The resulting weights combination is $W_P = 0.33, W_S = 0.00, W_M = 0.33, W_D = 0.33$.

6.4.3 Discussion

We conclude this section with some final comments.
As for the choice of appropriate data abstractions, in the supervised case, the B-spline fitting approach lead to the best performance, regardless the choice of a specific input space. Similar conclusions were reached in the unsupervised case, if an estimate of the number of clusters is available. In this case string-based approaches lead to comparable results. Finally, in the more realistic case the number of clusters is not available, B-splines strongly over-segment the data, while string-based and HMM report more convincing results, with a better balance between the number of estimated clusters and the percentage of correct associations.

For what concerns the criticality of choosing a particular input representation, we observed

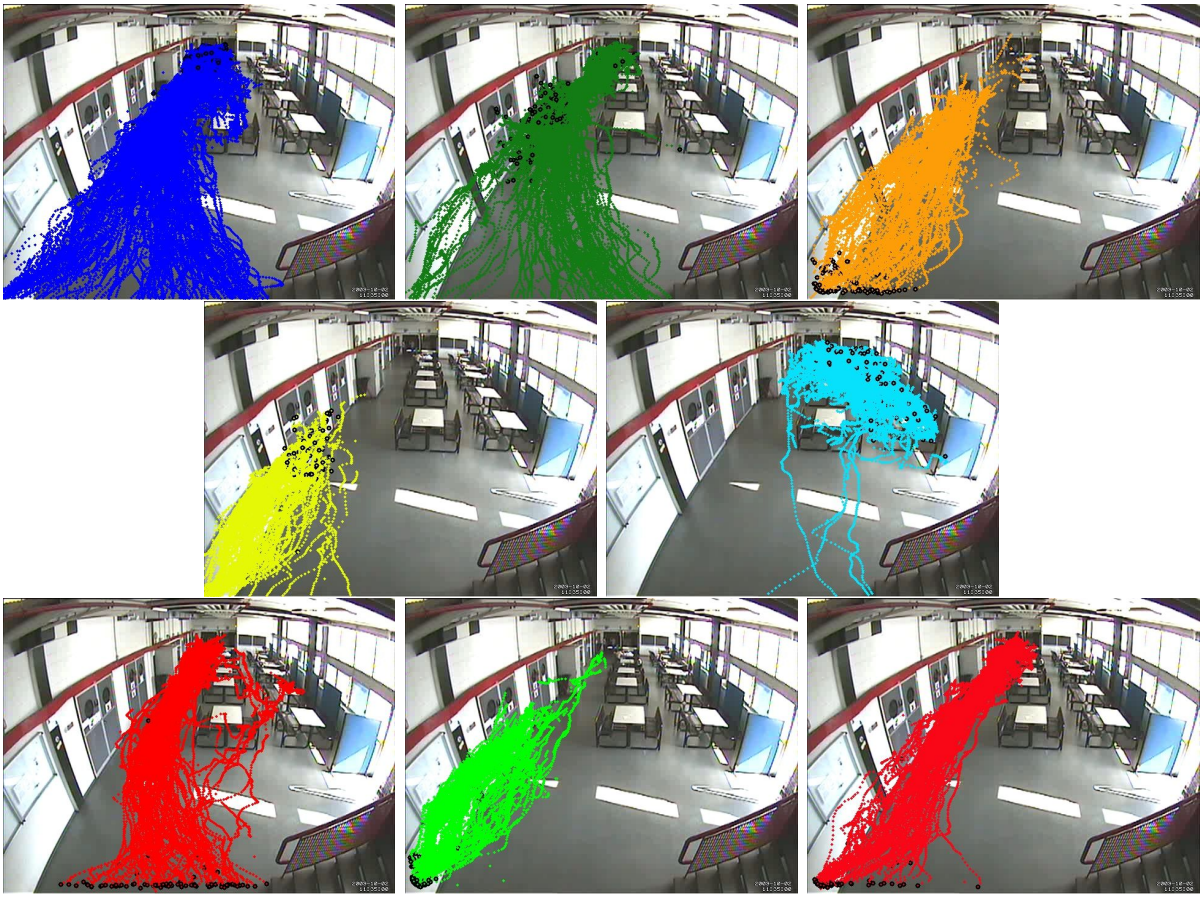


Figure 6.10: Best clustering results obtained with the HMM-based representation schema with the loose evaluation.

how in a supervised setting, where labels are available, the positions, even if visibly ambiguous, allows to obtain acceptable percentages of correct associations to known behaviors, being enforced by the labels which in turn depend on the positions. As opposite, in an unsupervised setting the position fails to disambiguate the data, while other information can be profitably inserted into the descriptions and significantly help to disambiguate the data. As a consequence, we can conclude that the unsupervised approach is more appropriate to our purpose.

Considering the ability of the data abstraction chosen to adapt to a change in the input representations (which could be useful if new input features were added to the initial representation), we observed how, in the case of curve fitting, the addition of a new measurement would require the estimation of a new fitting function; string-based approaches would require the construction of a new set of alphabets, but the process is entirely data-driven and could be performed automatically. The HMM method, in theory would scale nicely

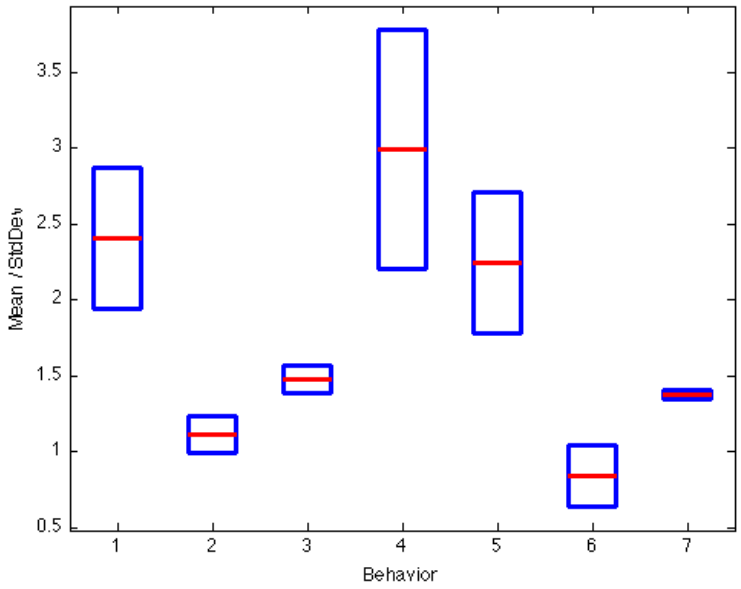
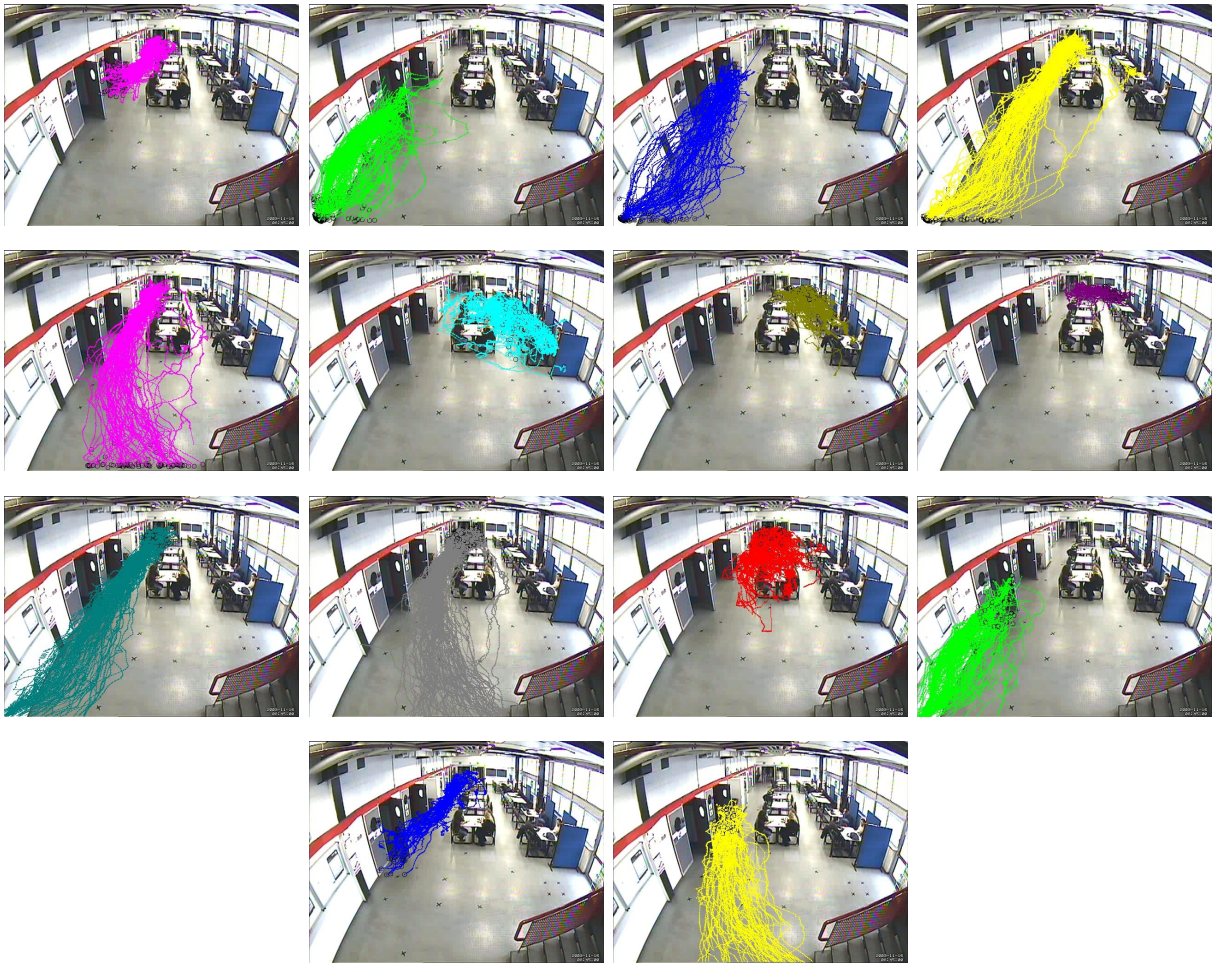


Figure 6.11: Average and standard deviations of the velocity modules of behaviors estimated when using the string-based representation with multi-cue integration.

with the input representation change, but in practice, since we model one trajectory at a time, increasing the number of parameters, immediately degrades to performance (this was observed in the experimental analysis, where a full $5 - dimensional$ input representation failed to produce convincing results).

All the reported experiments highlighted the fact that a specific labeling of dynamic events may be highly subjective and thus supervised approaches may not be in general appropriate for this application domain. Observing the manual annotation reported in Fig. 6.6 we see how behaviors 3, 4, and 6 appear to be very similar as they occur at a high distance from the viewing point. Not surprisingly, the estimated behaviors always fail to discriminate among these groups. This effect is magnified if we consider a more granular manual annotation.
To have a visual evidence of the capability of the representations to scale with the complexity, we specialize the data annotation as reported in Fig. 6.12, where, moreover, for each pattern source and sink regions are specified.

Figure 6.13 reports the similarity matrices obtained by reordering the data with respect to the new 14 behaviors (we do not consider in this analysis the B-spline fitting for its poor performance when adopted into the clustering framework, see Sec. 6.4.2). It is apparent that the string-based representation (Fig. 6.13, left) provides a way to enhance this more



	Behavior													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Source	9	2	6	1	8	8	9	6	1	1	8	10	5	10
Sink	8	8	5	8	7	1	1	8	9	10	2	8	6	2

Figure 6.12: An annotation on the training data characterized by a higher granularity.

detailed structures into the data, even if the compactness of classes might be improved. On the other hand, HMM-based approach (6.13, right), although having the capability of enforcing the similarities among component of a same class, tends to see as similar events which are instances of different behaviors. This encouraging results highlights the appropriateness of the description we propose, the one based in string, in the context of dynamic events description fir monitoring applications. We thus focus only on this specific approach for the remainder of the chapter, where we discuss the analysis performed when, at run-time, a new trajectory must be associated to one of the known behaviors or detected as an anomaly event.

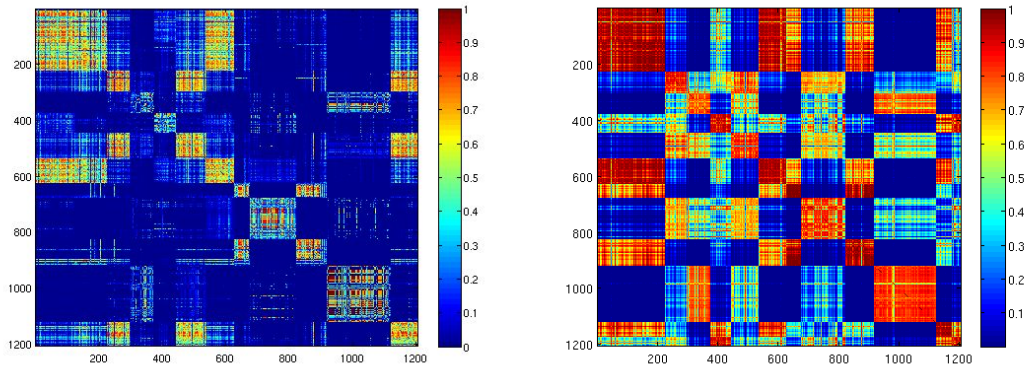


Figure 6.13: Similarity matrices for the alphabet $W_P = 0.33, W_S = 0.33, W_M = 0, W_D = 0.33$ and the HMM -based representation considering 14 annotated behaviors.

6.5 Analysis on test data

Once a new dynamic event is observed, the objective of a behavior understanding module is to associate this new observation to one of the common behaviors learned by the system. In case none of the previously observed behaviors is appropriate for the new observation we say that an *anomaly* or an *undefined event* is detected, depending on the likelihood with known models.

6.5.1 Test data association

According to the adopted pipeline, the dynamic event is first represented with respect to the appropriate input space (Sec. 4.5.1), then it is translated into strings according to one or more alphabets computed as described in (Sec. 4.2). Finally, it is compared against the

known models to associate it to some known behavior, if it is the case. For the last point, a strategy to represent a behavioral model (in our setting it corresponds to a clustering instance) must be defined.

We approach the problem following two different strategies. The first one relies on the use of a compact representation, the *string candidate*, that shares similarities with the approach based on medoid [VDLPB03].
The candidate for each cluster is computed as follows: for each cluster C, each string s of C votes for the string t in C most similar to it. The similarity is based on the P-Spectrum kernel with $P = 2$. By putting together the votes of all the strings in C, the string obtaining the higher number of votes becomes the candidate to represent the cluster. In this setting, thus, the N behavioral models $\{B_i\}_{i=1...N}$ are compactly described by means of a set of candidate strings $\{M_i\}_{i=1...N}$, one for each cluster highly representative of the content of the cluster itself.
This technique allows for a rapid visual inspection of the obtained results, since archetypical sequences can be easily visualized and compared with data waiting to be associated to clusters.
Given a test string s_t the comparison with the candidates $\{M_i\}_{i=1...N}$ is based on two thresholds, τ_1 and τ_2 . We refer to M_{first} as the most similar candidate (whose similarity with s_t is S_{first}), and M_{sec} the second most similar candidate (whose similarity is S_{sec}). Then:

- If $S_{first} \geq \tau_1$ and $(S_{first} - S_{sec}) \geq \tau_2$ associate s_t to cluster represented by M_{first};
- If $S_{first} \leq \tau_1$ an *anomaly* is detected;
- If $S_first \geq \tau_1$ and $(S_{first} - S_{sec}) \leq \tau_2$ no reliable association can be made (*undefined*).

As an alternative, since we deal with unsupervised learning, an out-of-sample method on the clustering tree [FBCM04b] may be applied. Thus, our second approach to the run-time analysis is based on the *Nystrom* method (see the Appendix for a detailed description). Although it represents a more common practice to associate a new datum to a given set of classes, it guarantees robustness at the price of a high computational cost and in a pure implementation the case of *undefined association* is not considered.

6.5.2 Experiments on test data

We complete the experimental analysis with a testing phase aiming at associating new observed trajectories to one of the known behaviors, when it is the case. Alternatively, it can be detected as an anomaly or the impossibility to decide can arise, when the information is considered not sufficient to provide a decision with an appropriately high confidence.

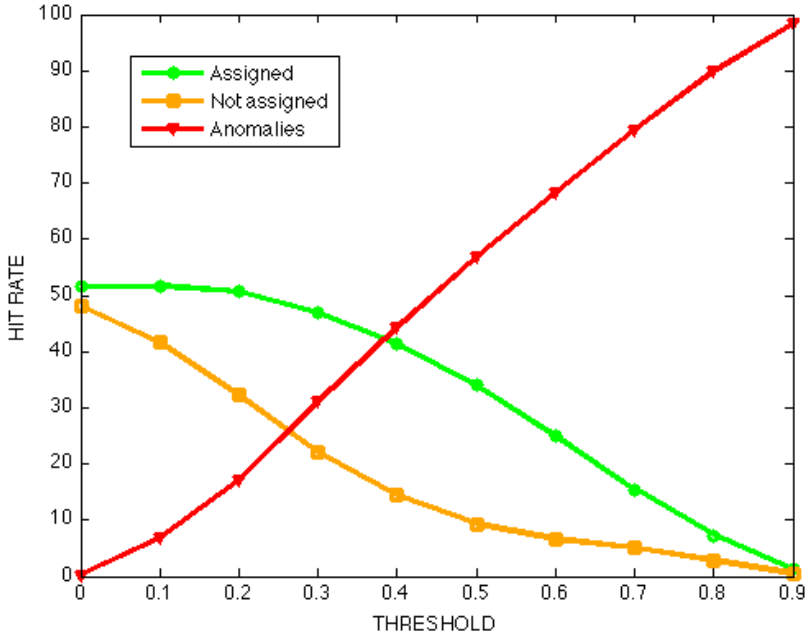


Figure 6.14: Trends of associated, undefined and anomalous events with respect to the τ_1 for a fixed model.

For what concerns the first test analysis, the one based on the use of candidates, the choice of the thresholds τ_1 and τ_2 needs to be done. We selected τ_1 so to reach a good compromise between the amount of associations and the number of anomalies: in Fig. 6.14 is reported the trend of the number of events that have been associated to some known models, detected as anomalies or undefined for uncertainty in the association. With respect to the graphic we decided to fix $\tau_1 = 0.35$, while τ_2 has been set to the value 0.1, which seems an appropriate gap between similarities to reach a sufficient confidence in the system decision.

Table 6.8 reports the results we obtained when using one of the four best performing alphabets. Some comments arise when analyzing the percentages. Test trajectories are more complying with the alphabet 14, reaching a higher percentage of association against a lower number of anomalies. The amounts of undefined events are comparable.

Both alphabets 13 and 14 consider position and direction, while differ in one of the weighted features (velocity module for the first, size for the second): the results thus suggest that changing focus of attention when considering the features strongly influence the set of trajectories considered as instances of the built models.

Model	% Assoc.	% Undef.	% Anom.
13	44.42	17.76	37.82
14	63.26	17.03	19.71
7	55.81	20.47	23.72
11	34.45	28.92	36.63

Table 6.8: Statistics on test sequences when modeling a behavior with a candidate string: the table reports the percentages of trajectories associated to some learned behavior, undefined or detected as anomaly (see text for details on the association procedure).

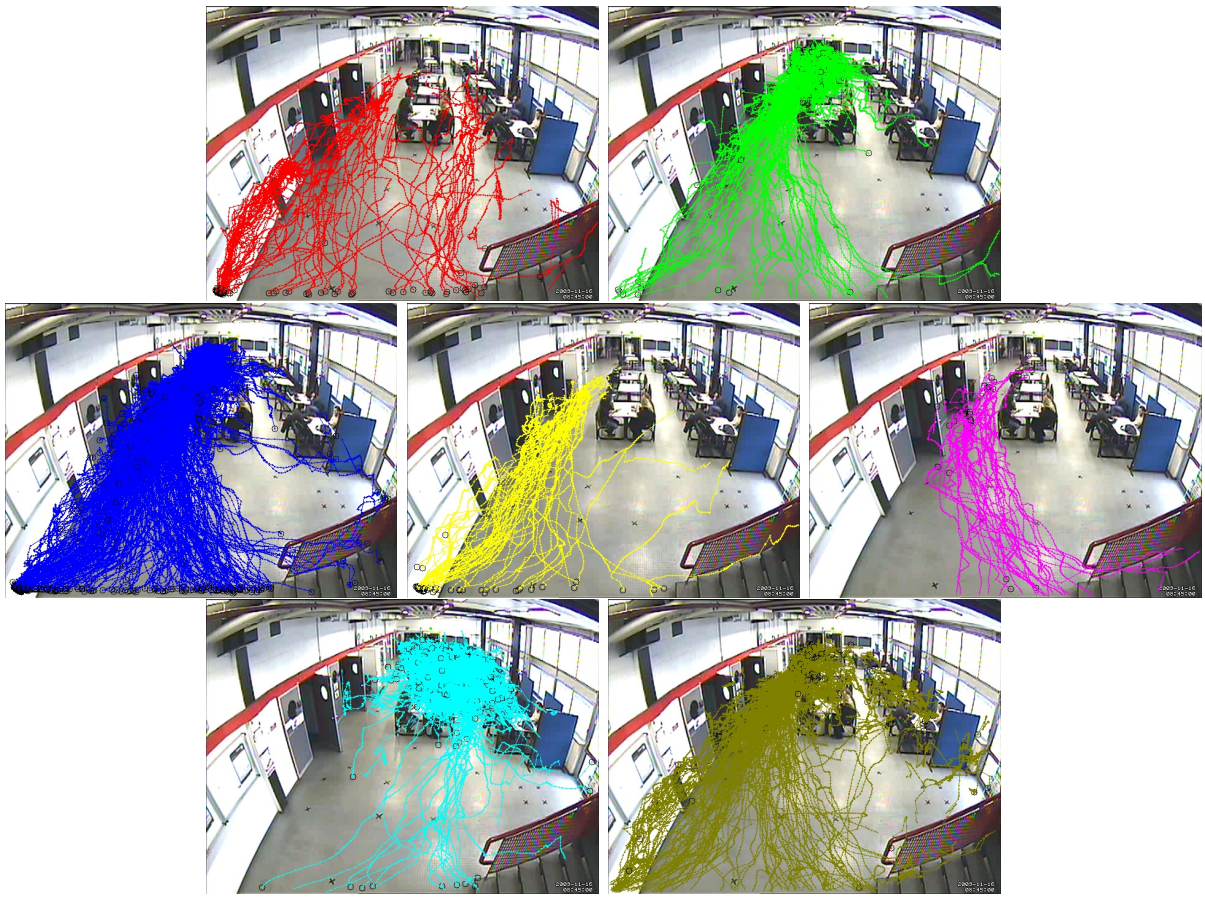


Figure 6.15: Clusters built with the candidates approach on the test set.

While discarding information (in alphabet 7 position and direction are taken into account while in the number 11 only the first is considered) the number of undefined events increases, suggesting that the the amount of information on top of which the models were

estimated is not sufficient to properly discriminate among the content into the data. Also, the variability on the percentages of association suggest that an accurate selection of the weighted features should be performed, considering the semantic of each one of them, with respect to the specific task, since they naturally lead to different views of the same scenario.

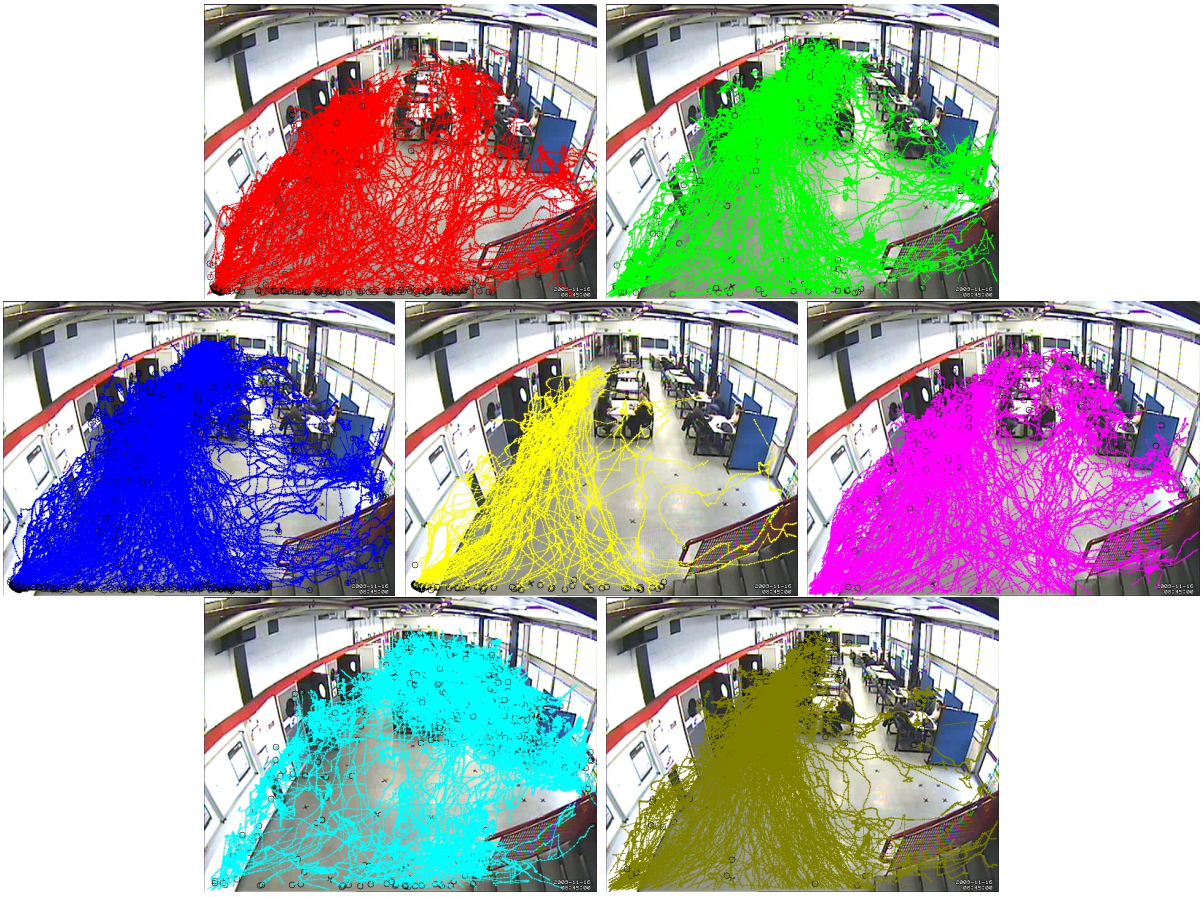


Figure 6.16: Clusters built with out-of-sample based on Nystrom method on the test set.

Fig. 6.15 depicts strings associated according to alphabet n. 13. They are compared with the test associations performed with the more standard out-of-sample based on Nystrom in Fig. 6.16. The comparison first shows how the models learned on the initial set of data nicely apply to a wider test set. The Nystrom approach produces comparable results to the price of a higher computational cost the obtained clusters are denser since a pure out -of-sample does not implement the option “undefined association”.

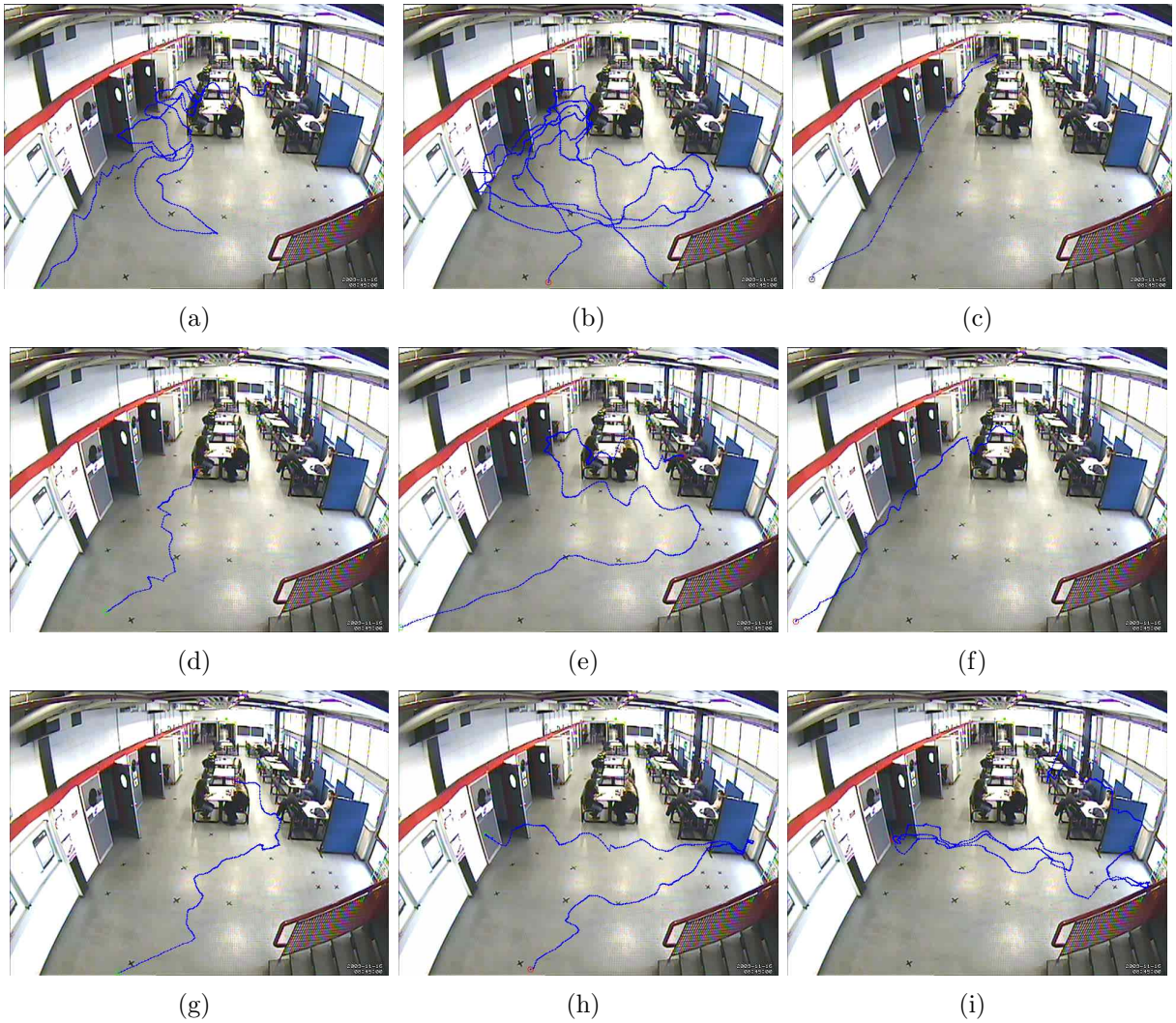


Figure 6.17: Events classified as anomalies with respect to the model built on top of alphabet 13, when adopting the candidate-based strategy.

Figure 6.17 reports examples of trajectories detected as anomalies if compare against the model corresponding to the alphabet 13 following the association strategy based on candidates.
The trajectories correspond to uncommon dynamic events (that is events weakly present, or even absent, in the training set). For instance, on the first row, the image on the left shows the trajectory of a person which starts walking from the table area and then proceeds towards the camera with a zigzagging motion, partially coming back. The central trajectory of the first row represents, instead, the motion of a loitering person, while on the right an apparently common event is the result of a person which is running away from

the camera position.
In Table 6.9, the similarities of the estimated models with each trajectory are reported: the values highlight how the events might be associated to models that for some aspects share similarities, but without reaching a high confidence. Very low similarities, as the ones for events (h) and (i), respectively 0.17 and 0.19, denote a significant discrepancy among observed events and learned models: a visual inspection confirms such intuition.
Notice that, for the particular case of the event (c), including the velocity module on the description would allows us to detect the dissimilarity with one of the models: the same test trajectory would be associated to behavior 3 by alphabet 7.

Anom.	B_1	B_2	B_3	B_4	B_5	B_6	B_7
a	0.33	0.	0.14	0.23	0.02	0.07	**0.34**
b	0.17	0.09	0.07	0.19	0.05	**0.23**	0.18
c	0.01	0.14	0.	**0.34**	0.25	0.	0.
d	0.13	0.	0.06	0.	0.	0.07	**0.18**
e	0.18	0.	0.11	0.19	0.	0.24	**0.26**
f	0.	0.08	0.02	**0.19**	0.16	0.	0.
g	0.17	0.	0.06	0.	0.	0.11	**0.32**
h	0.07	0.05	0.03	**0.17**	0.09	0.01	0.08
i	0.16	0.05	0.07	0.08	**0.19**	0.04	0.02

Table 6.9: Statistics on test sequences when modeling a behavior with a candidate string: the table reports the percentages of trajectories associated to some learned behavior, undefined or detected as anomaly (see text for details on the association precedure).

6.5.3 Discussion

This section was focused on the test analysis: given a new test datum, it is compared against all the behavioral patterns estimated in the previous modeling stage to decide whether it is a new realization of some known behavior or, instead, it can be defined an anomaly. The decision criteria we proposed, based on the use of thresholds, considers also that an event can remain undefined, if the system confidence is too low.
The analysis, performed on a big quantity of data, showed the capability of the thresholding mechanism of (i) preserving the shape of the induced clusters with respect to the original one (i.e., the one of the corresponding cluster estimated via the spectral clustering of strings) and, at the same time, (ii) highlighting anomalous events.
Given the promising results, the pipeline can be extended towards different directions, as reported in the final conclusions (Chapter. 7).

Chapter 7

Discussion and future work

This thesis investigated the problem of modeling common patterns of activity from long time observations for video surveillance applications. More specifically, an approach has been proposed that is based on exploiting string-based representations coupled with unsupervised learning theory that includes three main stages: (1) first, a low-level video processing gathers a set of descriptions (trajectories) of the dynamic events occurring in a given scenario, (2) then a higher abstraction level is reached by mapping the trajectories in an appropriate feature space, and finally (3) the new temporal descriptors are used to feed the Spectral Clustering to detect common patterns into the data.
The main contributions of our work refer to several aspects related to the behavior analysis problem:

- Regarding the video processing, we designed a tracking procedure based on a combined motion and appearance model of objects (Chapter 3). The method represents the data association problem as a graph simplification to adapt to different levels of the complexity of the monitored environment. The experimental analysis showed the appropriateness of our approach on different benchmark data sets both in term of accuracy of the results and low computational cost.

- For what concerns data abstraction, we explored the use of different representations schemes applied to the raw trajectories. In Chapter 4 we introduced three different schemes, including a feature mapping and an appropriate similarity measure: more specifically, we proposed a string-based representation coupled with the P-Spectrum kernel, and compared it against HMMs with Probabilistic Product kernels, an curve fitting with the well-know Gaussian kernel. The experimental analysis relied on the use of supervised learning methods, and Regularized Least Squares in particular, and was based on both synthetic and real data for which the ground truths were available. The results showed the advantages of using strings.

- As for the behavior analysis, we addressed the issue of extracting information from sets of unlabeled temporal data by means of unsupervised learning. In Chapter 5 we completed the pipeline for behavior analysis, proposing to plug the intermediate representations into a clustering step, based on Spectral Clustering. The experiments were designed similarly to Chapter 4 so to explore the unsupervised counterpart of the learning problem. The key idea during the evaluation was to interpret the clustering results as an association tool between estimated and real clusters. Even in this case the predominance of strings is apparent. By looking at real behavior analysis applications, where in the most cases the annotation is not available, we experimented the use of quality indices to evaluate the clustering results. We conclude that such techniques are not tailored for our setting.

- The last contribution of the thesis builds on top of the previous achievements and consists of a prototype behavior analysis tool. The tool, discussed in Chapter 6, involves two main steps, modeling phase and test phase. The first one refers to the previously described pipeline with strings-based representation and P-Spectrum kernel. The selection of the method parameters has been performed by evaluating the results with respect to a very loose annotation of the data, based on simple feedbacks from the user, that helps to capture general properties of the scenario. The behavioral models refer to the clustering results, instead of directly on the annotation, for the capability of the first to capture a higher granularity of the solution (the annotation is typically dictated by 3D properties). The second phase, instead, associates a new datum to one of the learned patterns (clusters) and has been addressed following two different out-of-sample strategies.

The final experimental analysis has been performed on data acquired along weeks, to test the robustness of the pipeline with respect to scene changes: the superiority of the string-based approach is confirmed.

At present, a number of issues are still open, deserving further investigations, and we expect the pipeline to be extended towards different directions.
A straightforward and natural improvement concerns the design of the test phase as a pure *run time analysis*. In the formulation of the problem we discussed here, a test set is gathered and processed as a whole considering one trajectory at a time. However, the same approach can be followed in an on-line setting, when trajectory modeling and association is made "on the fly" once a trajectory is observed. This calls for a requirement of low computational cost, so that the method can provide a feedback after a very little delay. In our case, the trajectory must be first represented with respect to a given alphabet (the one chosen during the model selection) by associating each instantaneous observation to one of the states in the partition: the computational cost is linear in the number of comparisons. Secondly, the string representation is compared against the candidates of model clusters by means of 2-Spectrum, which is efficiently computed by means of dynamic programming

techniques. We can thus conclude that the pipeline can be directly adopted in an online setting. As a further extension, instead of waiting for a trajectory to be complete, our test analysis can be directly applied to the trajectories under construction, to provide intermediate guess on the possible known patterns to which it might be related, until a high confidence is reached. These extensions is currently under development.

The problem of modeling behaviors on a batch training phase, we discussed in Chapter 6, has the major drawback it builds a "static" model. However, in a dynamic context as video analysis, where the temporal component can not be ignored and influences the observed data, having a first set of models that can then be extended and/or updated is the best choice. Also, it provides the robustness against the number of observed events, that continuously increases. After that the evaluation of the batch pipeline we proposed provided promising results, this is in our hypothesis a fundamental future extension.
Starting from a set of clusters, results of the batch training phase, the incremental strategy takes inspiration from the following considerations:

- Some new clusters may take shape. In our framework, it translates in saying that events that in first instance are recognized as anomalies but then are frequently observed in the scene should give rise to a new behavioral pattern.
- The clusters detected on the batch training phase should be updated with new points to capture the temporal variations into the scene.
- Finally, clusters might become irrelevant, if no instances of its data are observed for a proper amount of time. In such case, the correspondent pattern should be rejected by the model.

The technique of *incremental learning* [CCFM04, HK03, WF00] can help to address the problem. Related to the same topic, the use of some kind of *sliding window* on which focusing the analysis can be conveniently considered into the pipeline.
Finally we mention our aim of integrating our pipeline on a more complex video analysis. The low-level part we adopted is able to cope with medium crowd, and in general with scenarios where the identity of a single target or group of targets can be maintained. As the crowd increases, however, the tracking methods reach a sort of "breaking point" becoming unreliable. In such cases, our pipeline simply stops the analysis notifying the incapability to deal with the current scenario. An alternative is to design a more complex system able to trigger different video analysis procedures, with similar pipeline, on the basis of the scene conditions.

Appendix A

Machine learning ingredients

This appendix introduces the various machine learning methods exploited in the thesis. First, a brief introduction to supervised and unsupervised learning is given in Sec. A.1. Then, a more detailed discussion about supervised methods and related topics is presented in Sec. A.2. Sec A.3 is focused on the unsupervised counterpart, with particular attention to clustering problems: the Spectral Clustering is discussed in detail, including an out-of-sample strategy. Finally, the last section presents the popular Hidden Markov Models.

A.1 Supervised vs. unsupervised learning

It is possible to define and distinguish between different kinds of *learning from examples* In *supervised learning* the input data is paired with a given a sequence of *outputs* $(y_1, \ldots, y_n)$. The idea is to infer an unknown input-output relation on the basis of the given set of input-output instances. The available data, the *training set*, are a collection of pairs $\mathbf{z} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where $\mathbf{x}$ is a vector whereas y takes on discrete or continuous values. The distinction in output type has led to a naming convention for the following prediction tasks:

- *Pattern classification*, a learning problem with output values taken from a finite unordered set $C = \{C_1, \ldots, C_k\}$,
- *Regression*, a learning problem whose output values are real.

If one supposes that the input and output are observations of random variables represented by some joint probability density $\rho(\mathbf{x}, y)$, then the supervised learning can be formally defined as a density estimation problem where one is concerned with determining properties of conditional density $\rho(y|\mathbf{x})$.

In *unsupervised learning* the inputs data $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ are not provided with any target outputs. In this case one has a set of n observations of a random vector $\mathbf{x}$ having a joint density $\rho(\mathbf{x})$. The goal is to directly infer the properties of this probability density without the help of a supervisor or a teacher providing correct answer or degree-of-error for each observation. In a sense, unsupervised learning can be thought of as finding patterns in the data above and beyond what would be considered pure unstructured noise.

Within unsupervised learning, *cluster analysis* attempts to find multiple convex regions of the feature space that contains modes of $\rho(\mathbf{x})$. This can tell whether or not $\rho(\mathbf{x})$ can be represented by a mixture of simpler densities representing distinct types or classes of observations.

With supervised learning there is a clear measure of success that can be used to compare the effectiveness of different methods over various situations. Lack of success is directly measured by expected loss over the joint distribution $\rho(\mathbf{x}, y)$. This can be estimated in a variety of ways including cross-validation [Efr87].

In the context of unsupervised learning, there is no such direct measure of success. One must often resort to heuristic arguments not only for motivating the algorithms, as is often the case in supervised learning as well, but also for judgments to the quality of the results. In the next sections we present an overview of the supervised learning theory, while the focus of Sec. A.3 will be on the unsupervised approach.

A.2 Supervised learning

As we explained above, the idea of supervised learning is to infer an unknown input-output relation on the basis of a given set of input-output instances. A supervised learning problem can be described as: given a certain number of observations we want to recover an approximation of the model underlying them. The problem is not trivial since we always have a finite amount of information available and various causes of uncertainty might affect the problem.

A (simplified) graphical visualization of a 2-dimensional toy problem is useful. In Figure A.1 each input point is a 2-dimensional vector and its label is given by its color (red or blue). On the top left we see a set of data which can be thought of as a sample from a larger (possibly infinite) population on the top right. In this model the goal is then to draw a line such that points belonging to different classes falls in different sides. It is crucial to remember the goal is not only to describe the available data but rather to be predictive on new data. A solution which perfectly separates the data (above, left) can perform poorly on other points of the same population (above, right). Even in this toy model we can see some features of the problem. If we simply try to find a prediction rule which performs well

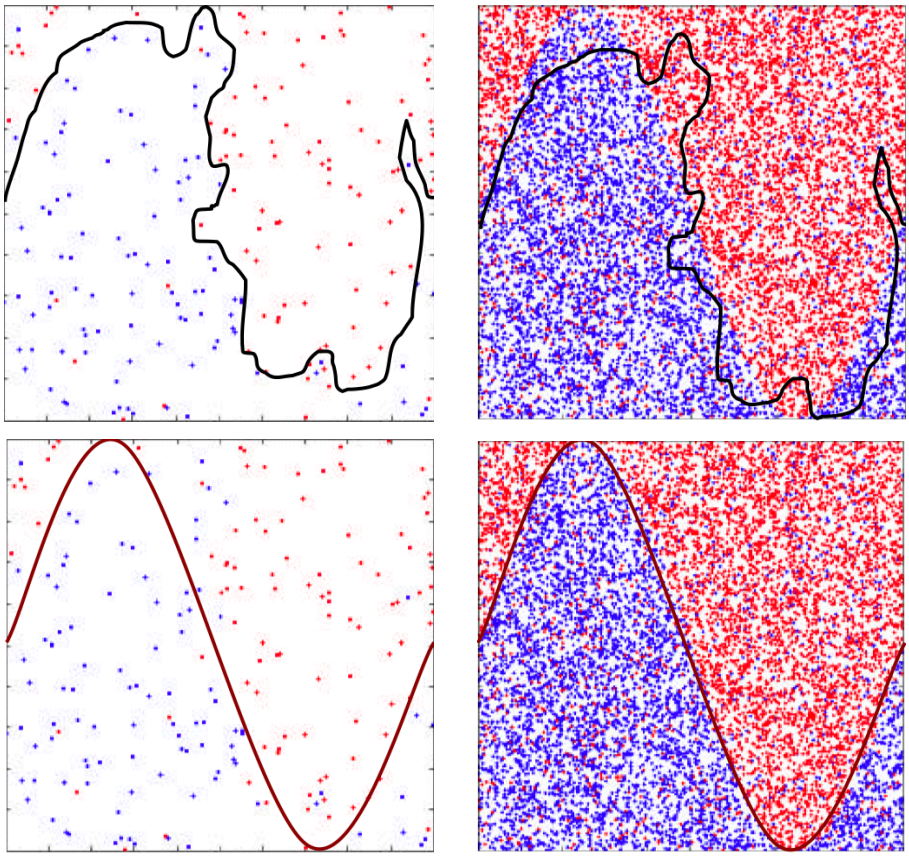


Figure A.1: A 2-dimensional toy classification problem. Top left there is a data sample from a larger population on top left. Top left, we can see a solution with very low empirical error and top right we see the poor performance of the same solution on the entire population. Below, a more regular solution better applies to new data.

on the data we tend to perform bad on new data. Clearly this is due to the fact that we have a finite number of examples. Moreover it should be clear that the more complex is the problem at hand the more examples we need: we can see an interplay between regularity of the target and number of required data.
A solution which simply describes the data is too irregular or too complex, that is it is an overfitting solution. If we postulate that the problem has some regularity properties, then we might want to impose constraints on the class of possible solutions. Regularity is described in terms of complexity (see [Bou02] and references therein) or stability of the solution. In Fig. A.1, below, a more regular solution provides both an appropriate separator of the sampled points (left) and a good predictor on new data (right).

In the next section we formally present some ingredients of supervised learning and regularization theory (*regularization networks* [EPP00] or *regularized kernel methods* [CST00], [SS02], [Vap98]).

Input and output spaces

We consider two sets of random variables $\mathbf{x} \in X \subset \mathbb{R}^d$ and $y \in Y$. The labels y_i belong into a bounded subset $Y \subset \mathbb{R}$ (for example in a binary classification problem $Y = \{-1, 1\}$). Let X and Y be related by an unknown probability distribution $\rho(\mathbf{x}, y)$ defined over the set $X \times Y$.

We are provided with *examples* of this probabilistic relationship, that is with a data set $\mathbf{z} = \{(\mathbf{x}_i, y_i) \in X \times Y\}_{i=1}^n$ called *training set*, obtained by sampling n times the set $X \times Y$ according to $\rho(\mathbf{x}, y) = \rho(y|\mathbf{x})\rho_{\mathcal{X}}(\mathbf{x})$.

We assume that the examples $\mathbf{z}$ are drawn identically and independently distributed according to $\rho(\mathbf{x}, y)$. Moreover, we assume that X is a compact subset of $\mathbb{R}^d$, and the labels y_i belong into a bounded subset $Y \subset \mathbb{R}$ (for example in a binary classification problem $Y = \{-1, 1\}$).

Given the data set $\mathbf{z}$, the "problem of learning" consists in providing an *estimator* $f_{\mathbf{z}} : X \to Y$ that can be used to predict a value $f_{\mathbf{z}}(\mathbf{x}) \sim y$ for each $\mathbf{x} \in X$. Since we know only a finite set of points $\mathbf{z}$ the estimator $f_{\mathbf{z}}$ can be seen as an approximation of the ideal estimator $f : X \to Y$, also named the *target function*.

Expected risk

The standard way to deal with the learning problem consists in defining a *risk functional*, which measures the average amount of error or risk associated with an estimator, and then looking for the estimator with the lowest risk. If $\ell(y, f(\mathbf{x}))$ is the loss function measuring

the error we make when we predict y by $f(\mathbf{x})$, then the average error, the so called *expected risk*, is:

$$I[f] \equiv \int_{X,Y} \ell(y, f(\mathbf{x}))\rho(\mathbf{x}, y)\ d\mathbf{x}dy \tag{A.1}$$

Different loss functions lead to different learning algorithms (see [EPP00]). A common choices is the square loss, $\ell(y, f(\mathbf{x})) = (f(\mathbf{x}) - y)^2$

Empirical Risk Minimization

In this section we focus on a specific learning approach, the *Empirical Risk Minimization* (ERM). The importance of ERM lies in the fact that most algorithms can be seen as refinements of it. In a few words the idea is that since we cannot minimize the expected error directly we can replace it with its empirical counterpart. Given a training set, a possible way to estimate $I[f]$ is to evaluate the *empirical risk*

$$I^{\mathbf{z}}_{emp}[f] = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)). \tag{A.2}$$

Straightforward *minimization of the empirical risk* (ERM) is an ill posed problem, since the solution is not unique. A well established approach to deal with ill posedness is regularization. In the following we first recall the concept of kernel functions and then briefly introduce regularized kernel methods.

Kernel functions

The "kernel trick" is a method for using a linear classifier algorithm to solve a non-linear problem by mapping the original non-linear observations into a higher-dimensional space, where the linear classifier is subsequently evaluated. This makes a linear classification in the new space equivalent to non-linear classification in the original space. Specifically, we can project (see Figure A.2) a data point $\mathbf{x}$ in $\mathbb{R}^d$ to a high dimensional feature space $\mathbb{R}^F$, by a nonlinear mapping function $\phi : \mathbb{R}^d \to \mathbb{R}^F$, $F > d$, and proceed to training and testing in the feature space.

The kernel trick transforms any algorithm that solely depends on the dot product between two vectors. Wherever a dot product is used, it is replaced with the kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

$$K(\mathbf{x}, \mathbf{s}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{s}) \tag{A.3}$$

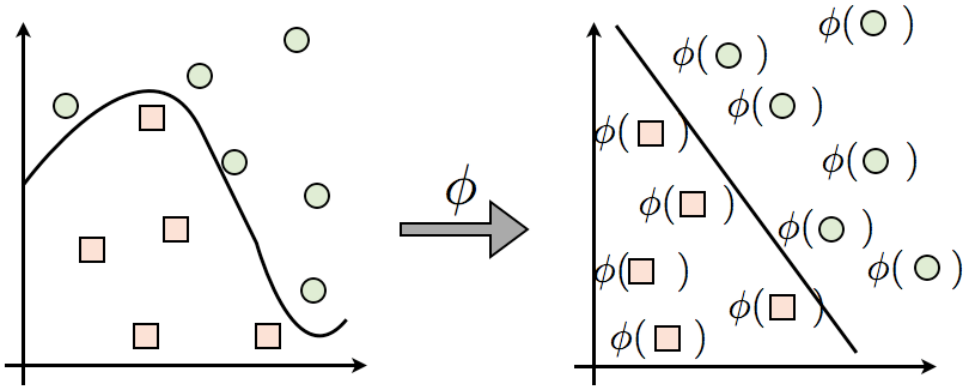


Figure A.2: A visual representation of the feature mapping induced when applying a kernel function on a given domain: the non-linear relations between data in the native space becomes linear in the feature space, so that linear algorithms can be applied to successfully capture similarities among them.

instead of defining $\phi(\mathbf{x})$ explicitly.
Given the training set $(\mathbf{x}_1, \ldots \mathbf{x}_n)$ the $n \times n$ matrix $\mathbf{K}$ formed by $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ is called the *kernel matrix*. A natural question to ask is, given a function $K(\mathbf{x}, \mathbf{s})$ how to decide whether it is a kernel function, or whether there exists a function $\phi(\mathbf{x})$ such that $K(\mathbf{x}, \mathbf{s}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{s})$. The answer is provided by the following sentence [SS02]: a function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ can be decomposed as an inner product (see Eq. A.3) for some feature map $\phi(\mathbf{x})$ if and only if the function is symmetric and the matrix formed by restriction to any subset of the space $\mathbb{R}^n$ is positive semi-definite. K is a Mercer kernel if $K : X \times X \to \mathbb{R}$ is a symmetric continuous function, which is positive definite [Aro50].
An example of kernel functions is the polynomial kernel $K(\mathbf{x}, \mathbf{s}) = (\mathbf{x} \cdot \mathbf{s})^p$ where p is the polynomial degree. Another widely-used kernel that we will make use of below is the radial basis function (RBF) kernel [TC04], [Bur98]

$$K(\mathbf{x}, \mathbf{s}) = e^{-\frac{||\mathbf{x}-\mathbf{s}||^2}{\sigma^2}} \tag{A.4}$$

where σ is a *width* controlling parameter. The RBF kernel implies a function $\phi(\mathbf{x})$ that is infinite dimensional.

Predictivity is a trade-off between the information provided by training data and the complexity of the solution we are looking for. An important class of problems in regularization theory are generated by a positive definite kernel $K(\mathbf{x}, \mathbf{s})$ and the corresponding space of functions $\mathcal{H}$ is called *Reproducing Kernel Hilbert Space*.

Regularized Least Squares

From the seminal work of Tikhonov and others [TA77] regularization has been rigorously defined in the theory of ill-posed inverse problems.
The idea of using regularization in statistics and machine learning has been explored since a long time - see for example [Wah90], [PG92] and references therein - and the connection between large margin kernel methods such as Support Vector Machines and regularization is well known – see [Vap98], [EPP00], [SS02] and reference therein. Ideas coming from inverse problems regarded mostly the use of Tikhonov regularization and were extended to several error measures other then the quadratic loss function.
Straightforward *minimization of the empirical risk* (ERM) is an ill posed problem, since the solution is not unique. The basic idea of regularization is to restore well-posedness of ERM by constraining the hypothesis space $\mathcal{H}$. A possible way to do this is considering *penalized* ERM.
We look for solutions minimizing a functional composed of two terms: the first one is a fitting term and the second is a smoothness term. This functional can be written as:

$$\underbrace{ERR(f_{\mathbf{z}})}_{empirical\ error} + \underbrace{\lambda PEN(f_{\mathbf{z}})}_{penalization\ term} \tag{A.5}$$

where λ is the regularization parameter, being a trade-off between the two terms.

We assume that the kernel K is bounded by 1 and is universal (see [MXH06] and references therein), that is the set of functions

$$\mathcal{H} = \{\sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) \mid \mathbf{x}_i \in X,\ \alpha_i \in \mathbb{R}\}$$

is dense in $L^2(X)$, the Hilbert space of functions that are square-integrable with respect to $\rho_{\mathcal{X}}$.
Tikhonov regularization or Regularized Least Squares (RLS) amounts to minimize

$$\min_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2 \right\} \tag{A.6}$$

As mentioned above, the second term is a *smoothness* or a *complexity* term measuring the norm of the function f in a suitable Hilbert space $\mathcal{H}$. The minimization takes place in the *hypothesis space* $\mathcal{H}$.

The minimizer of the regularized empirical functional, for each training set $\mathbf{z} = (\mathbf{x}, y) = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ of n-examples $(\mathbf{x}_i, y_i) \in X \times Y$, can be represented by the expression (*representer theorem*)

$$f_{\mathbf{z}}^{\lambda}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) \quad \text{with} \quad \alpha = (\mathbf{K} + n\lambda I)^{-1}\mathbf{y}. \tag{A.7}$$

where K is a Mercer kernel and $\mathbf{K}$ is the kernel matrix.
Since $\lambda > 0$, it is clear that we are numerically stabilizing a matrix inversion problem which is possibly ill-conditioned (that is numerically unstable). With the choice of the square loss, the generalization property of the estimator means that the estimator $f_{\mathbf{z}}^{\lambda}$ is a good approximation of the regression function

$$f_{\rho}(\mathbf{x}) = \int_{Y} y \; d\rho(y|\mathbf{x}), \tag{A.8}$$

with respect to the norm of $\mathcal{L}^2(X)$. In particular the algorithm is (weakly) consistent [Vap98] if, for a suitable choice of the parameter $\lambda = \lambda_n$ as a function of the examples,

$$\lim_{n\to\infty} \int_{X} (f_{\mathbf{z}}^{\lambda}(\mathbf{x}) - f_{\rho}(\mathbf{x}))^2 \; d\rho_{\mathcal{X}}(\mathbf{x}) = \lim_{n\to\infty} \|f_{\mathbf{z}}^{\lambda} - f_{\rho}\|_{\rho}^2 = 0 \tag{A.9}$$

with high probability – see for example [Vap98].

A.3 Unsupervised learning: clustering

In an unsupervised setting, a data set of input $\{x_1, x_2, ..., x_n\}$ is gathered but there is no a priori information about their outputs. The goal is to build representations of the input that can be used for decision making and predicting future inputs: in a sense, unsupervised learning can be thought of as finding patterns in the data. Almost all work in unsupervised learning can be viewed in terms of learning a probabilistic model of the data, i.e. to estimate a model that represents the probability distribution for a new input $\{x_{n+1}\}$ given previous inputs $\{x_1, x_2, ..., x_n\}$

$$P(x_{n+1}|x_1, x_2, ..., x_n)$$

In simpler cases, where the order in which the inputs arrive is irrelevant or unknown, the machine can build a model of the data which assumes that the data points are independently and identically drawn from some distribution $P(x)$. Such a model can be used for outlier detection or monitoring and for classification.
It is worth to point out that, for what concerns the evaluation of the results when dealing with unsupervised learning, there is not a counterpart for the theoretical background in the supervised case: this fact turns unsupervised results evaluation into a highly challenging problem.

A very simple and classic example of unsupervised learning is clustering, an approach which aims at extracting knowledge from an input set detecting internal structures, or, in other

words, groups of coherent data with respect to some criterion.
The literature on clustering methods is extremely wide and does not the focus of this thesis; a survey for the interested reader can be found in

Among the different techniques, the popular Spectral clustering refers to a class of techniques to partition points into disjoint clusters by analyzing the eigenstructure of the Laplacian matrix of a similarity graph associated to the data set. In the original method, the first few eigenvectors of the graph's Laplacian [Chu97] were shown to carry information about the *optimal cut* to partition the graph. However, a major drawbacks referred to the requirement of fixing a-priori the number of clusters, analogously to what happens with the majority of clustering methods traditionally adopted in literature.
Different algorithms have been proposed in the last years that exploit this result, as in [NJW02] and [SM00]. In the latter, in particular, an efficient recursive algorithm for spectral clustering is proposed where it is not required to decide the number of the clusters beforehand, which is rather a tricky parameters when no a-priori information on the problem is available. In this section we start by reviewing the original formulation of the problem, focusing afterwards on the modified recursive algorithm.

A.3.1 Clustering based on spectral analysis

We assume to have an input data set $X = \{x_1, ..., x_n\}$ and and some *pairwise similarity* $s_{ij} \geq 0$ between all the possible pairs x_i and x_j in the set. Clustering the set means that we are looking for a disjoint partition made of groups of coherent data. In Spectral Clustering, the relationships among the points into the input set are represented in a graph-based fashion, building an undirected *similarity graph* $G = (V, E)$, such that each node in $V = \{v_i, ..., v_n\}$ corresponds to a point of X. The elements in E, instead, represent the connections (edges) among all the pair of vertices (v_i, v_j) and are weighted with a value w_{ij} strictly related to the similarity between the corresponding input points, s_{ij}. Since the graph is undirected it holds that $w_{i,j} = w_{j,i}$ for all $i, j = 1, ..., n$.
From this perspective, the clustering task can be restated as the problem of finding a partition of the graph such that the weights are low for edges across the partition classes and high among each class. This translates in requiring a low *inter-class* similarity and a strong *intra-class similarity*.
Several popular construction of the similarity graph have been proposed in the literature, depending on the concept of similarity applied to the goal of modeling the local neighborhood relationships between points:

- In the **ε-neighborhood graph** two points are connected with an edge if $dist_{ij} < \varepsilon$, where *dist* is an appropriate pairwise distance;
- By applying the concept of k-nearest neighbors we obtain the **k-nearest neighbor**

graph. Since this leads naturally to a connected graph, a slight adaption must be done to obtain an undirected graph, as required by the framework;

- Finally, the **fully connected graph** is built by weighting the edges with the similarity between the corresponding points, This construction is suitable only for those similarities modeling local neighborhoods, as the well-known Gaussian similarity function

$$s(x_i, x_j) = e^{\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)} \tag{A.10}$$

where the standard deviation σ controls the width of the neighborhoods and plays an analogous role to the one of ε in the first approach.

We refer here to the latter case.
Before moving on throw the discussion, let us define some basic ingredients used later to formalize the problem. The *adjacency matrix* of the graph G is the matrix $\mathbf{W} = \{w_{ij}\}_{i,j=1,...,n}$: when $w_{ij} = 0$ then the similarity s_{ij} is null and thus it does not exist an edge between v_i (x_i) and v_j (x_j).
The *degree* d_i of a vertex $v_i \in V$ is defined as

$$d_i = \sum_{j=1}^{n} w_{ij} \tag{A.11}$$

Since $w_{ij} = 0$ when the vertices v_i and v_j are not connected, the sum runs over all the adjacent vertices to v_i.
We now define the *degree matrix* which is the diagonal matrix with the degree $d_1, ..., d_n$ on the diagonal.
Finally, we explain the concept of *volume* of a subset $A \subset V$ as the sum of degrees of elements in A:

$$Vol(A) = \sum_{i \in A} d_i \tag{A.12}$$

The normalized graph Laplacians The main ingredient of Spectral Clustering is the *Laplacian matrix* [Chu97], a matrix that has the capability of capturing the structure of the graph to which it refers, defined as

$$L = D - W \tag{A.13}$$

The Laplacian matrix L satisfies the following main properties (for a more complete overview see [Moh91, MJ97]):

1. L is *symmetric* and *positive semi-definite*

2. L has non-negative, real-values eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_m$

3. The smallest eigenvalues of L is null, while the corresponding eigenvector is equal to the constant 1 vector, $\mathbf{1}$

4. The multiplicity k of the null eigenvalue of L is equal to the number of connected components in the graph which L corresponds to.

From L, another important matrix, the normalized Laplacian can be directly derived. There is not a unique definition of such matrix, here we specifically refer to the proposal in [NJW02]:

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \tag{A.14}$$

It can be shown (see [Lux07]) that the properties 1., 2. and 4. still hold; the properties number 3. can be restated as follows: the null eigenvalues of L_{sym} corresponds to the eigenvector $D^{\frac{1}{2}}\mathbf{1}$

The algorithm: pseudo-code Following the notation introduced so far, we can now report the main steps of the spectral clustering algorithm based on the normalized Laplacian by means of pseudo-code. The main characteristic is related to a change of representation, from the input points x_i in some real space, to points $y_i \in \mathbb{R}^k$: such mapping enhances the clusters properties, so that a rather simple clustering method, as *k-means*, is able to address the problem of finding the partition.

Algorithm 1 Spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number of clusters k
1: Build the fully connected similarity graph and obtain the adjacency matrix
2: Build the normalized Laplacian L_{sym}
3: Estimate the first k eigenvectors $\mathbf{u}_1, ..., \mathbf{u}_k$ of L_{sym} (which correspond to the k smallest eigenvalues of the matrix) and build the matrix $U \in \mathbb{R}^{n \times k}$ whose columns are the eigenvectors
4: Build the matrix $T \in \mathbb{R}^{n \times k}$ by applying a normalization step to matrix U so that $t_{ij} = \frac{u_{ij}}{\left(\sum_k u_{ij}^2\right)^{\frac{1}{2}}}$
5: for $i = 1, ..., n$ do
6: set $y_i \in \mathbb{R}^k$ to the i-th row of matrix T
7: end for
8: Cluster using *k-means* the points $(y_i)_{i=1,...,n}$ obtaining the clusters $C_1, ..., C_n$
Output: partition $A_1, ..., A_k$ such that $A_i = \{j | y_j \in C_i\}$

The graph cut As mentioned before, when representing the data and their pairwise similarities with a graph structure, the problem of finding a consistent partition of the data translates in looking for a partition of the graph such that the edges across groups have very low weights, while connections among each group are very strong.
The most simple attempt in finding a graph partition relies on solving the so-called *mincut* problem. For a given number of clusters k the solution $P^* = \{A_1^*, ..., A_k^*\}$ of the mincut problem is chosen as the partition minimizing the cut function:

$$P^* = argmin_P(cut(P)) \tag{A.15}$$

where $P = \{A_1, ..., A_k\}$ is a possible partition of the input data set, and

$$cut(P) = cut\{A_1, ..., A_k\} = \frac{1}{2}\sum_{i=1}^{k} W(A_i, \bar{A}_i) \tag{A.16}$$

with $\bar{A}_i$ being the complement of A_i.
However, thus rather naive approach has the major drawback that it tends to separate isolated points from the rest of the graph, leading to unsatisfactory solutions. This suggest that a constraint on the clusters size should be included in the method. The most popular approaches in this direction are the *Ratio Cut* [HK92] and the *Normalized Cut* (or Ncut) [SM00]: here we briefly discuss the latter. In the Ncut the size of each cluster is measured by the volume of its edges:

$$Ncut(A_1, ..., A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{Vol(A_i)} = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{Vol(A_i)} \tag{A.17}$$

The objective of minimizing the function is to achieve a balance between clusters in term of their volumes, since the minimum of $\sum_{i=1}^{k}(\frac{1}{Vol(A_i)})$ is reached when the volumes of all clusters coincide.
The major disadvantage resides in the computational cost, that when introducing the constraint on the clusters size, becomes NP hard [WW93]. It is in this context that the advantage of using spectral clustering is more convincing, since its formulation represents a way to solve relaxed versions of the Ncut problem: a complete derivation of the problem from this perspective is discussed in [Lux07].

Discussions The use of Spectral Clustering poses a number of critical issues that need to be specified for an appropriate definition of the problem: the solution reached by the method is sub-optimal, so an accurate selection of the main parameters allows to achieve a higher confidence in the result, for which there is not guarantee.
The first choice relates of which type of graph adopting, among the list reported in Sec. A.3.1. This is strictly related to the choice of the similarity measure, that should be

dictated by the domain of the data and the task, so no general advice can be given. The main requirement is that the local neighborhood induced by the similarities must reflect meaningful groups. The requirement is also conditioned by the width of such neighborhood. When using the similarity based on Gaussian, which is the most popular option when dealing with fully connected graphs, it depends of the σ parameters, the standard deviation of the function.
The choice of σ is a highly challenging problem and no satisfactory solutions have been proposed so far. However, some heuristics can be employed. A rather popular approach consists in considering the $n \times n$ matrix of the pairwise euclidean distances of points and set to value of σ as the median of the K-nearest neighbors distances, where K is a reasonable percentages of the data set. A second technique relies on the use of the diameter of a data set X, which is defined in the following way:

$$diam = \max_{x_i, x_j \in X} (dist(x_i, x_j)) \tag{A.18}$$

where *dist* in an appropriate distance measure (typically the euclidean distance). The value of σ is then fixed as a percentage of the diameter.
Notice that with both these approaches, some hypothesis on the geometry of the data are implicitly done.
Another important open problem that strongly affects the final results is the a-priori choice of the number of clusters, an issue that in this original formulation the Spectral Clustering shares with the most of other popular clustering methods. The presence of the final step of k-means leads not only to the requirement for this a-priori choice, but also influences the results with the non-deterministic step typical of k-means.
These considerations lead to the proposal in literature of a different approach to Spectral Clustering able to combine the power of the hierarchical methods together with an independence from the choice on the number of clusters.

A.3.2 Two-way Spectral Clustering

The *Two-Way Spectral Clustering* is an efficient recursive clustering algorithm able to successfully combine the advantages of clustering based on spectral analysis with the approach to subdivision problems of the hierarchical methods.
The starting point is the popular work presented in [SM00], where it has been demonstrated that the optimal cut is strictly related to the first eigenvalue. The result is a clustering method which does not require the number of clusters to be fixed a-priori and does not include the final step based on k-means, present in the original formulation of Spectral Clustering: thanks to this variation, the method produces results which are stable across multiple runs.
The key idea is that, at each recursion, clusters are subdivided in two. At the first step,

the input data set is split in two groups, and to each group is then recursively applied the same operation. The subdivision is dictated by the minimization of the Normalized Cut objective function, introduced in the previous section. It can be shown that rewriting the Normalized Cut problem and relaxing the solution, the final expression can be found by solving the generalized eigenvalue system

$$D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}\mathbf{x} = \lambda\mathbf{x} \tag{A.19}$$

where D and W have been discussed when describing the original formulation of Spectral Clustering. Notice that this is a formulation adopting the normalized Laplacian matrix (L_{sym}) described above.
The values of the second eigenvector corresponding to each cluster is adopted as a tool to associate the points to the two new sub-clusters: in [SM00], in fact, the authors show that the closest approximation of the optimal graph partition is the one associated to the sign of the components of such eigenvectors. The recursion stops when a given condition is verified: in [SM00] it was related to a threshold on the Ncut, which allows for some kind of control on the granularity of the solution.
The definition of the method leads naturally to schematize the procedure as a binary tree of the input points, as shown in Fig. A.3 for a toy example. In the picture, another interesting property of the method is shown, related to the fact that at each step of recursion, the most significant separation is performed: in the first step, the circles are separated from the rest of polygonal figures; then the number of edges are considered, and finally squares are divided from rectangles.

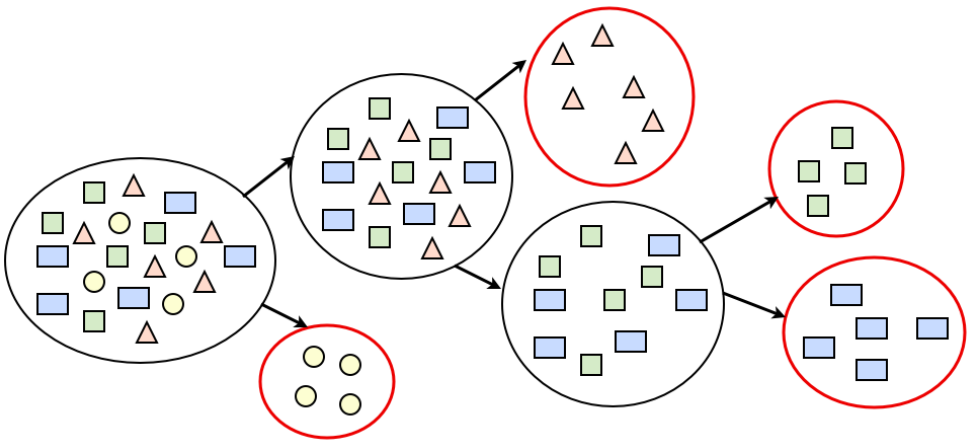


Figure A.3: A visual representation of the recursive Two-Way Spectral Clustering on a toy example. The red sets denote the final partition.

The recursive procedure can thus be algorithmically defined as folllows:

Algorithm 2 Two-Way Spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, cut threshold TH
1: repeat
2: Build the fully connected similarity graph and obtain the adjacency matrix W
3: Solve the eigenvalues system $D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}\mathbf{x} = \lambda\mathbf{x}$
4: Use the eigenvector corresponding to the second smallest eigenvalue to bipartite the graph
5: until $NCut < TH$

A.3.3 The Nystrom method

When dealing with unsupervised learning a main issue is related to the capability of assigning a new test datum to one of the clusters estimated during the training phase. A class of approaches rely on the use of some kind of "clustering structure", learned in the training phase as well. In is the case of the popular *Nystrom* method, which represents a simple and at the same time solution to the problem [Nys28, Bak77, PTVF92].
The Nystrom method is a technique to find numerical approximations to a class of problems defined as

$$\int_a^b A(x,y)\phi(y)dy \quad := \lambda\phi(x) \tag{A.20}$$

that is referred to as *autofunctions problems*.
The key idea is to extend the eigenvalues associated to a given matrix $\mathbf{A}$ to a new point using the elements of the matrix as weights to interpolate. The solution is first estimated on a set of squaring nodes and then extended to the whole domain by means of interpolation. More formally, given a set of equally spaced points $\{\xi_1, \xi_2, ..., \xi_n\} \in [a,b]$ a simple squaring rule is

$$\frac{(b-a)}{n}\sum_{j=1}^{n} A(x,\xi_j)\hat{\phi}(\xi_j) \quad = \lambda\hat{\phi}(x) \tag{A.21}$$

where $\hat{\phi}(x)$ is an approximation of the real $\phi(x)$. The system can be solved by setting $x = \xi_i$ obtaining the new system

$$\frac{(b-a)}{n}\sum_{j=1}^{n} A(\xi_i,\xi_j)\hat{\phi}(\xi_j) \quad = \lambda\hat{\phi}(\xi_i) \tag{A.22}$$

The problem can be restated in a matrix form considering, without loss in generality, $[a, b] = [0, 1]$, and fixing $A_{ij} = A(\xi_i, \xi_j)$:

$$A\Phi = n\Phi\Lambda \tag{A.23}$$

where $\Phi = [\phi_1, \phi_2, ..., \phi_n]$ are the eigenvectors of A correspondent to the eigenvectors $\lambda_1, \lambda_2, ..., \lambda_n$. When substituting the ϕ_i e λ_i values in Eq. A.21 the *Nystrom extension* for each ϕ_i is defined as

$$\hat{\phi}_i(x) \quad = \frac{1}{n\lambda_i} \sum_{j=1}^{n} A(x, \xi_j)\hat{\phi}_i(\xi_j) \tag{A.24}$$

In other words, the obtained method allows us to extend an eigenvector, computed on a fixed points set, to a new arbitrary datum, by exploiting $A(\cdot\,, \xi_j)$ as matrix of the interpolating weights. This formulation can be profitably adopted in the context of kernel-based learning methods [DM05] and Spectral Clustering in particular [FBCM04a].
We refer here to a specific application in this second context, where the Nystrom method is exploited with respect to the normalized Laplacian matrix L_{sym}.

Computing the interpolating weights

Il Normalized Laplacian L_{sym} has been defined in Eq. A.14.
It is worth first noticing that computing the smallest eigenvalues, and correspondent eigenvectors, of a generic matrix $I - A$ is equivalent to estimate the biggest eigenvalues (and eigenvectors) of A. The Nystrom extension we will consider relies on the biggest eigenvectors of $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$, that are the smallest of L_{sym}).
The matrix $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$ can be computed as follows:

$$\begin{aligned} D^{-\frac{1}{2}}WD^{-\frac{1}{2}} &= \begin{pmatrix} \frac{1}{\sqrt{d_{11}}} & \dots & 0 \\ 0 & \dots & 0 \\ 0 & \dots & \frac{1}{\sqrt{d_{nn}}} \end{pmatrix} \begin{pmatrix} w_{11} & \dots & w_{1n} \\ \dots & \dots & \dots \\ w_{n1} & \dots & w_{nn} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{d_{11}}} & \dots & 0 \\ 0 & \dots & 0 \\ 0 & \dots & \frac{1}{\sqrt{d_{nn}}} \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{d_{11}}w_{11} & \frac{1}{\sqrt{d_{11}d_{22}}}w_{12} & \dots & \frac{1}{\sqrt{d_{11}d_{nn}}}w_{1n} \\ \dots & \dots & \dots & \dots \\ \frac{1}{\sqrt{d_{ii}d_{11}}}w_{i1} & \frac{1}{\sqrt{d_{ii}d_{22}}}w_{i2} & \dots & \frac{1}{\sqrt{d_{ii}d_{nn}}}w_{in} \\ \dots & \dots & \dots & \dots \\ \frac{1}{\sqrt{d_{nn}d_{11}}}w_{n1} & \frac{1}{\sqrt{d_{nn}d_{22}}}w_{n2} & \dots & \frac{1}{d_{nn}}w_{nn} \end{pmatrix} \end{aligned}$$

At this point the computation of $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$ for a new point reduces to the following steps:

- Computing the similarities between a new point and the training set ($w_{(n+1)\,i}\ \forall i = 1, ..., n$)

- Obtaining the degree of the new node in the graph as $d_{(n+1)\,(n+1)} = \sum_{i=1}^{n} w_{(n+1)\,i}$
- Building the new row in the matrix on the basis of the previous results by computing

$$D^{-\frac{1}{2}} W D^{-\frac{1}{2}}_{(n+1)} = \left(\frac{1}{\sqrt{d_{(n+1)\,(n+1)} d_{11}}} w_{(n+1)\,1}\,, \cdots, \frac{1}{d_{(n+1)\,(n+1)}} w_{(n+1)\,(n+1)} \right) \quad \text{(A.25)}$$

Assuming that n_{eig} eigenvalues and eigenvectors have been computed, for each $i = 1, ..., n_{eig}$

$$\hat{\phi}_i(x) = \frac{1}{n\lambda_i} \sum_{j=1}^{n} D^{-\frac{1}{2}} W D^{-\frac{1}{2}}_{(n+1,j)}\, \hat{\phi}_i(\xi_j) \quad \text{(A.26)}$$

where λ_i is the i-th eigenvalues of L_{sym} which corresponds to the eigenvector $\hat{\phi}_i(\xi_j)$. Each eigenvector is thus extended to a new point a interpreting the new row of $D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ as weights for the interpolation.

Nyström extension and Two-Way Recursive Spectral Clustering

The general formulation of the Nyström extension derived so far can be easily adapted to the recursive Spectral Clustering observing that at each step only 2 eigenvalues (also eigenvectors) must be computed. More specifically, it holds that the first eigenvalues is always null [Lux07] (thus the Nyström extension can not be applied to it); to the second eigenvalues, instead, corresponds the eigenvector $\hat{\phi}_2(\xi_j)$, thus

$$\hat{\phi}_2(x) = \frac{1}{n\lambda_2} \sum_{j=1}^{n} (D^{-\frac{1}{2}} W D^{-\frac{1}{2}})_{(n+1,j)}\, \hat{\phi}_2(\xi_j) \quad \text{(A.27)}$$

Applying the Nyström Extension after a recursive clustering is equivalent to explore the clusters tree from the root and for each node, computing the Nyström Extension of the new point by only considering the points involved in the specific recursion step. Depending on the extension value with respect to the split point, the tree is visited following the left or the right path. The same procedure is repeated until a leaf node is reached and whose label is associated to the new point.

To summarize, when considering a new point x and for each step i of the Two-Way Spectral Clustering, the main steps of the procedure include the following:

1. Computing the similarities between x and all the n_i points ξ_{i_j} involved in step i as $w_{(n_i+1),j} \quad \forall\, j = 1, ..., n_i$
2. Computing the degree of the new node as $d_{(n_i+1),(n_i+1)} = \sum_{j=1}^{n_i} w_{(n_i+1),j}$
3. Updating the degrees of the old nodes to include the weights of the new one, when it is the case, $d_{jj} = d_{jj} + w_{(n_i+1),j} \quad \forall\, j = 1, .., n_i$

4. Building the new row in the matrix to obtain the interpolating weights

$$(D^{-\frac{1}{2}}WD^{-\frac{1}{2}})_{n_i+1,j} = \frac{1}{\sqrt{d_{(n_i+1)(n_i+1)}\, d_{jj}}} w_{(n_i+1),j} \quad \forall\, j = 1, ..., n_i \tag{A.28}$$

5. Computing the Nyström extension

$$\hat{\phi}_{2_i}(x) = \frac{1}{n_i \lambda_{2_i}} \sum_{j=1}^{n} (D^{-\frac{1}{2}}WD^{-\frac{1}{2}})_{n_i+1,j}\ \hat{\phi}_2(\xi_j) \tag{A.29}$$

6. Evaluating $\hat{\phi}_{2_i}(x)$ with respect to the criteria for the cut and consequently assigning the new point to one of the 2 clusters.

All the reported steps should be included during the training stage of the recursive clustering so to be able to exploit the same procedure during the test phase. More specifically information regarding (1) the points involved, (2) the degrees matrix, (3) λ_{2_i} e ϕ_{2_i} values, (4) the split value, and finally (5) the labels must be stored at each iterations.
At this point, the same information can be exploited to associate a new point to one of the clusters.

Detection of outliers

As mentioned above, during the test phase the clustering tree is visited by plugging the information of a new point into the Nyström procedure to finally reach one of the leaf and associate the correspondent label to the new point.
However, such procedure forces each point to be associated to one of the clusters even if it is an *outlier*, that is none of the previously learned patterns is appropriate to the new point.

A possible solution is based on the consideration that an outlier should have very low similarities against all the other points, that is how to say that the gap between its value for the Nystroḿ extension and 0 is remarkable. The approach to detect outliers can thus be summarized as follows (it is shown in Fig. A.4): a threshold is set which is based on the smallest, in absolute value, components of the eigenvector, including a tolerance. At this point, the Nyström extension is compared against such values and considered as outlier if in the range $[s_l, s_h]$.

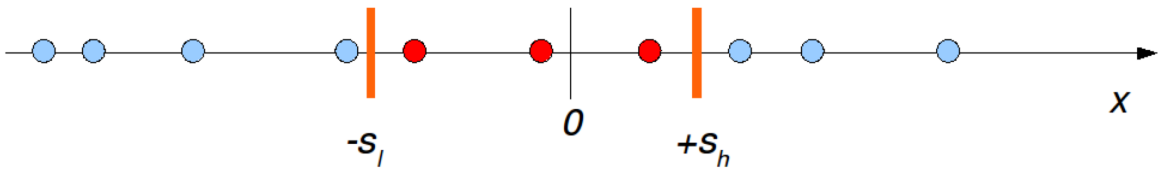


Figure A.4: A visual representation of the outlier detection. The eigenvector elements are represented as points on a straight line: light blue dots denote component of points associated to the cluster, the red ones correspond to outliers. For the latter, the value of the Nyström extension is in the ranges $[s_l, 0]$ and $[0, s_h]$ for, respectively, negative and positive components.

A.4 Hidden Markov Models

A very traditional parametric approach to temporal series modeling consists of building a probabilistic model of the present observation given all past observations:

$$p(x_{i,t}|x_{i,1}, y_{i,2}, \ldots, y_{i,t-1}).$$

Because the history of observations can grow arbitrarily large it is necessary to limit the complexity of such a model. A common approach to overcome this problem is to limit the window of past observations. The simplest model, $p(x_{i,t}|x_{i,t-1})$, is known as a first-order Markov model. One popular choice of probabilistic models that make this assumption, are the hidden Markov models (HMMs), stochastic state machines based on the use of hidden variables. In Fig. A.5 a visual representation of the architecture of an Hidden

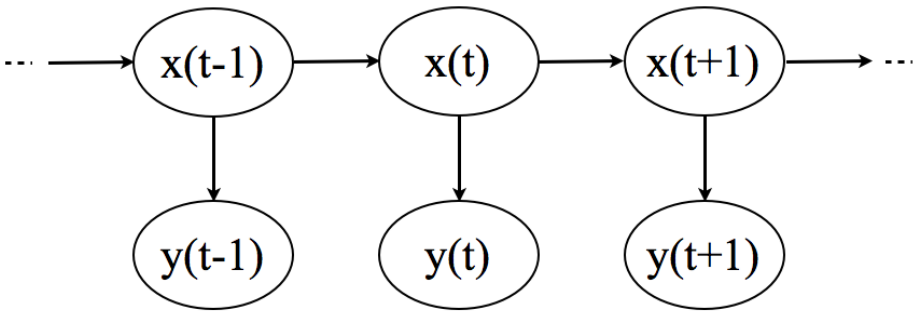


Figure A.5: A visual representation of the architecture underlying an Hidden Markov Model.

Markov Model is shown. At a certain time instant t, the hidden variable (hidden state) x_t

is statistically related to the previous state and to the current observation.
For an HMM, it is always possible to write down explicitly the likelihood function [Rab89]:

$$p(\mathbf{x}|\theta) = \sum_{q_0,\ldots,q_k} p(x_0|q_0)p(q_0)\prod_{t=1}^{k} p(x_t|q_t)p(q_t|q_{t-1}) \tag{A.30}$$

where the $\mathbf{q} = \{q_1, \ldots, q_k\}$ are the hidden states corresponding to each element of $\mathbf{x}$ and can take one of the discrete values $\{1, \ldots, M\}$. In order to define a HMM, one has to specify the parameter vector $\theta = (\pi, \alpha, \mu, \Sigma)$, where

- $\pi \in \mathbb{R}^M$ represents the initial state probability distributions $\pi_i = p(q_0 = i)$ for all $i = 1, \ldots, M$
- $\alpha \in \mathbb{R}^{M\times M}$ is the state transition probability distribution, i.e. $\alpha_{ij} = p(q_t = j|q_{t-1} = i)$
- $\mu = \{\mu_1, \ldots, \mu_M\}$ and $\Sigma = \{\Sigma_1, \ldots, \Sigma_M\}$ represents the mean and covariance of the Gaussian distributions for each state i, i.e. the emission density $p(x_t|q_t = i)$ is a normal distribution $\mathcal{N}(x_t|\mu_i, \Sigma_i)$.

In order to exploit such representation schema, given a data set of N trajectories the first step is to estimate the parameters of K distinct HMMs representing the probabilistic dynamical models that generated the observations. For example, such estimation may be achieved quite efficiently by choosing one off-the-shelf methods for maximizing the likelihood of the HMM parameters given the observations.

It is important to note that such approach is somehow robust with respect to the specific dimensionality of the instantaneous observations. That is, the algorithm used for generating the representative vectors θ_i, in principle, does not depend on the number of components of the single vectors x_i^t.

Bibliography

[AC08] N. Anjum and A. Cavallaro. Multifeature object trajectory clustering for video analysis. *IEEE Trans on Circuits and Systems for Video Technology*, 18(11), 2008.

[AMGC02] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.

[AMP06] S. Atev, O. Masoud, and N. Papanikolopoulos. Learning traffic patterns at intersections by spectral clustering of motion trajectories. In *IROS*, pages 4851–4856, 2006.

[AOP06] S. Atev, Masoud O., and N. Papanikolopoulos. Learning traffic patterns at intersections by spectral clustering of motion trajectories. In *IROS*, pages 4851–4856, 2006.

[Aro50] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68, 1950.

[BA03] N. Bolshakova and F. Azuaje. Cluster validation techniques for genome expression data. *Signal Process.*, 83(4):825–833, 2003.

[Bak77] C.T.H. Baker. The numerical treatment of integral equations. *Oxford: Clarendon Press*, 1977.

[BAT05] D. Biliotti, G. Antonini, and J.P. Thiran. Multi-layer hierarchical clustering of pedestrian trajectories for automatic counting of people in video sequences. In *Motion05*, pages II: 50–57, 2005.

[BCD04] B.Han, D. Comaniciu, and L Davis. Sequential kernel density approximation through mode propagation. In *In Proc. ACCV 2004*, 2004.

[BI05] O. Boiman and M. Irani. Detecting irregularities in images and in video. In *ICCV05*, pages 462–469, 2005.

[BKS07] F. Bashir, A. Khokhar, and D. Schonfeld. Object trajectory-based activity classification and recognition using hidden markov model. *IEEE Transactions on Image Processing*, 16, 2007.

[Bou02] O. Bousquet. *Concentration Inequalities and Empirical Processes Theory Applied to the Analysis of Learning Algorithms*. PhD thesis, 2002.

[BQKS05] F. Bashir, Wei Qu, A. Khokhar, and D. Schonfeld. Hmm-based motion recognition system using segmented pca. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–1288–91, 2005.

[Bra98] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface, 1998.

[BS98] A.J. Bulpitt and N. Sumpter. Learning spatio-temporal patterns for predicting object behaviour. In *BMVC98*, 1998.

[BSK04] D. Buzan, S. Sclaroff, and G. Kollios. Extraction and clustering of motion trajectories in video. In *ICPR*, volume 2, pages 521–524. IEEE Computer Society, 2004.

[Bur98] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[CBM02] M. Cristiani, M. Bicego, and V. Murino. Integrated region- and pixel-based approach to background modeling. In *Proceedings of IEEE Workshop on Motion and Video Computing*, 2002.

[CCFM04] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004.

[Cea04] G. Csurka and et. al. Visual categorization with bags of keypoints. In *ECCV*, 2004.

[CGN+01] R. Cucchiara, C. Grana, G. Neri, M. Piccardi, and A. Prati. The sakbot system for moving object detection and tracking. In P. Remagnino, G.A. Jones, N. Paragios, and C.S. Regazzoni, editors, *Video-based Surveillance Systems - Computer Vision and Distributed Processing*, pages 145–158. Kluwer Academic Publishers, 2001.

[Chu97] Fan R. K. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. AMS, 1997, 1997.

[CRM00] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, volume 2, pages 142–149, 2000.

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge Univ. Press, 2000.

[Cuc05] R. Cucchiara. Multimedia surveillance systems. In *VSSN '05: Proceedings of the third ACM international workshop on Video surveillance & sensor networks*, pages 3–10, New York, NY, USA, 2005. ACM.

[CV05] F. Camastra and A. Verri. A novel kernel method for clustering. *IEEE Trans. on PAMI*, 27(5):801–804, 2005.

[DCWS03] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto. Dynamic textures. *Int. J. Comput. Vision*, 51(2):91–109, 2003.

[DM05] P. Drineas and M. Mahoney. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *Technical Report*, 2005.

[ea98a] T. Boult et al. Frame-rate multibody tracking for surveillance. In *IUW*, pages 305–308, 1998.

[ea98b] T. Kanade et al. Advances in cooperative multi-sensor video surveillance. In *IUW*, pages 3–24, 1998.

[ea05] R. J. Radke et al. Image change detection algorithms: A systematic survey. In *IEEE Trans. Image Processing*, volume 14(3), pages 294–307, 2005.

[EBMM03] A.A. Efros, A.C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *IEEE Int. Conf. on Computer Vision*, pages 726–733, 2003.

[Efr87] Bradley Efron. *The Jackknife, the Bootstrap, and Other Resampling Plans (CBMS-NSF Regional Conference Series in Applied Mathematics)*. Society for Industrial & Applied Mathematics, 1987.

[EHD00] A. Elgammal, D. Harwood, and L. S. Davis. Non-parametric model for background subtraction. In *ECCV*, 2000.

[EPP00] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[FBCM04a] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral Grouping Using the Nyström Method. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2004.

[FBCM04b] C. Fowlkes, S. Belongie, F.R.K. Chung, and J. Malik. Spectral grouping using the nyström method. *IEEE Trans. on PAMI*, 26(2):214–225, 2004.

[FT97] P. Fieguth and D. Terzopoulos. Color-based tracking of heads and other mobile objects at video frame rates. In *in Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 21–27, 1997.

[GB03] S. Günter and H. Bunke. Validation indices for graph clustering. *Pattern Recogn. Lett.*, 24(8):1107–1113, 2003.

[Gea98] W. E. L. Grimson and et. al. Using adaptive tracking to classify and monitor activities in a site. In *CVPR*, pages 22–29, 1998.

[Gel96] A. Gelb. Applied optimal estimation. In *MIT Press*, 1996.

[HBV01] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:107–145, 2001.

[HHD98] I. Haritaoglu, D. Harwood, and L. S. Davis. W4: Who? when? where? what? a real-time system for detecting and tracking people. In *the third IEEE International Conference on Automatic Face and Gesture Recognition, IEEE Computer Society Press*, pages 222–227, 1998.

[HHD99] T. Horprasert, D. Harwood, and L. S. Davis. A statisctical approach for real-time robust background subtraction and shadow detection. In *IEEE Frame-Rate Applications Workshop*, 1999.

[HK92] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. In *IEEE Trans. Comput.-Aided Des*, pages 1074–1085, 1992.

[HK03] K. M. Hammouda and M. S. Kamel. Incremental document clustering using cluster similarity histograms. In *IEEE/WIC International Conference on Volume*, pages 597–601, 2003.

[HNR84] Y. Hsu, H. Nagel, and G. Rekers. New likelihood test methods for change detection in image sequences. In *Comput. Vis. Graph. Image Process.*, volume 26, pages 73–106, 1984.

[HTF03] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2003.

[HTWM04] W. Hu, T. N. Tan, L. Wang, and S. J. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Tran. on Systems, Man and Cybernetics*, 34(3):334–352, 2004.

[HXF+06a] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. *IEEE Trans on PAMI*, 28(9), 2006.

[HXF+06b] Weiming Hu, Xuejuan Xiao, Zhouyu Fu, Dan Xie, Tieniu Tan, and Steve Maybank. A system for learning statistical motion patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(9):1450–1464, 2006.

[HXF+07] Weiming Hu, Dan Xie, Zhouyu Fu, Wenrong Zeng, and Steve Maybank. Semantic-based surveillance video retrieval. *Image Processing, IEEE Transactions on*, 16(4):1168–1181, 2007.

[IB98] M. Isard and A. Blake. Condensation-conditional density propagation for visual tracking. In *IJCV*, 1998.

[IB00] Y. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans on PAMI*, 22(8), 2000.

[JCN06] L Fei-Fei J C Niebles, H Wang. Unsupervised learning of human action categories using spatial-temporal words. In *Proc. of BMVC*, 2006.

[JJS04] I.N. Junejo, O. Javed, and M. Shah. Multi feature path modeling for video surveillance. In *ICPR04*, pages II: 716–719, 2004.

[JKH04] T. Jebara, R. I. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[JST07] T. Jebara, Y. Song, and K. Thadani. Spectral clustering and embedding with hidden markov models. In Joost N. Kok, Jacek Koronacki, Ramon López de Mántaras, Stan Matwin, Dunja Mladenic, and Andrzej Skowron, editors, *ECML*, volume 4701 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2007.

[KCHD05] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3):172–185, 2005.

[KN05] S. Khalid and A. Naftel. Classifying spatiotemporal object trajectories using unsupervised learning of basis function coefficients. In *VSSN '05: Proceedings of the third ACM international workshop on Video surveillance & sensor networks*, pages 45–52. ACM, 2005.

[KS00] S. Khan and M Shah. Tracking people in presence of occlusion. In *In Asian Conference on Computer Vision*, pages 1132–1137, 2000.

[LCST06] M.H.-Y. Liao, Duan-Yu C., Chih-Wen S., and Hsiao-Rang T. Real-time event detection and its application to surveillance systems. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4 pp.–512, 0-0 2006.

[LEN03] C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: a string kernel for svm protein classification. *Bioinformatics*, 1(1), 2003.

[LHE03] D. S. Lee, J. J. Hull, and B. Erol. A bayesian framework for gaussian mixture background modeling. In *IEEE International Conference on Image Processing*, 2003.

[LHH06] X. Li, W. Hu, and W. Hu. A coarse-to-fine strategy for vehicle motion trajectory clustering. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 591–594, Washington, DC, USA, 2006. IEEE Computer Society.

[Lia05] T. W. Liao. Clustering of time series data: A survey. *Pattern Recognition*, 38(11), 2005.

[LOW00] W. Lin, M. A. Orgun, and G. J. Williams. Temporal data mining using multilevel-local polynominal models. In *Int. Conf. IDEAL*, volume 1983 of *Lecture Notes in Computer Science*, 2000.

[LSG07] B. Leibe, K. Schindler, and L. Van Gool. Coupled detection and trajectory estimation for multi-object tracking. In *ICCV*, 2007.

[LSST+02] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. text classification using string kernels. *JMLR*, 2, 2002.

[Lux07] U. Von Luxburg. A tutorial on spectral clustering. *Statistical computing*, 2007.

[MCB+01] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23, 2001.

[ME02] D. Makris and T. Ellis. Path detection in video surveillance. *IVC*, 20(12):895–903, October 2002.

[MJ97] B. Mohar and N.T.M. Juvan. Some applications of laplace eigenvalues of graphs. In *Graph Symmetry: Algebraic Methods and Applications, volume 497 of NATO ASI Series C*, pages 227–275. Kluwer, 1997.

[Moh91] B. Mohar. The Laplacian spectrum of graphs. *Graph Theory, Combinatorics, Applications*, 2:871–898, 1991.

[MT08] B.T. Morris and M.M. Trivedi. A survey of vision-based trajectory learning and analysis for surveillance. *CirSysVideo*, 18(8):1114–1127, 2008.

[MT09] B. Morris and M. M. Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, 2009.

[MXH06] C. A. Micchelli, Y. Xu, and Zhang H. Universal kernels. *JMLR*, 7:2651–2667, 2006.

[NDLO09] N. Noceti, A. Destrero, A. Lovato, and F. Odone. Combined motion and appearance models for robust object tracking in real-time. In *AVSS*, 2009.

[NJW02] A.Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS 14*, 2002.

[NK06] Andrew N. and Shehzad K. Motion trajectory learning in the dft-coefficient feature space. *Computer Vision Systems, International Conference on*, 0:47, 2006.

[NKMVG03] K. Nummiaro, E. Koller-Meier, and L. Van Gool. An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99–110, 2003.

[NO09] N. Noceti and F. Odone. Towards an unsupervised framework for behavior analysis. In *Workshop on Pattern Recognition and Artificial Intelligence for Human Behaviour Analysis (PRAI*HBA)*, 2009.

[NO10] N. Noceti and F. Odone. Anomaly detection in a loosely annotated framework. In *Submitted to the International Conference on Patterns Recognition*, 2010.

[NSO08a] N. Noceti, M. Santoro, and F. Odone. String-based spectral clustering for understanding human behaviours. In *THEMIS-BMVC*, 2008.

[NSO08b] N. Noceti, M. Santoro, and F. Odone. Unsupervised learning of behavioural patterns for video-surveillance. In *Workshop MLVMA-ECCV*, 2008.

[NSO10] N. Noceti, M. Santoro, and F. Odone. Learning behavioral patterns for video surveillance. *Submitted to the book titled Machine Learning for Vision-based Motion Analysis*, 2010.

[Nys28] E.J. Nyström. Über die praktische auflösung von linearen integralgleichungen mit anwendungen auf randwertaufgaben der potentialtheorie. *Commentationes Physico-Mathematicae, vol. 4*, 1928.

[PCV00] M. Pittore, M. Campani, and A. Verri. Learning to recognize visual dynamic events from examples. *IJCV*, 2000.

[PF06] C. Piciarelli and G. L. Foresti. On-line trajectory clustering for anomalous events detection. *Pattern Recogn. Lett.*, 27(15):1835–1842, 2006.

[PG92] T. Poggio and F. Girosi. A theory of networks for approximation and learning. In C. Lau, editor, *Foundation of Neural Networks*, pages 91–106. IEEE Press, Piscataway, N.J., 1992.

[Pic04] M. Piccardi. Background subtraction techniques: a review. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3099–3104, 2004.

[PMC+00] Ivana Miki Pamela, Ivana Miki?, Pamela C. Cosman, Greg T. Kogut, and Mohan M. Trivedi. Moving shadow and object detection in traffic scenes, 2000.

[PMF08] C. Piciarelli, C. Micheloni, and G. L. Foresti. Trajectory-based anomalous event detection. *IEEE Trans on Circuits and Systems for Video Technology*, 18(11), 2008.

[PMTC03] Andrea Prati, Ivana Mikic, Mohan M. Trivedi, and Rita Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:918–923, 2003.

[PMTH01] I. Pavlidis, V. Morellas, P. Tsiamyrtzis, and S. Harp. Urban surveillance systems: From the laboratory to the commercial world. *Proceedings of the IEEE*, 89(10):1478–1497, 2001.

[Por04] F.M. Porikli. Learning object trajectory patterns by spectral clustering. In *ICME*, pages 1171–1174, 2004.

[PT03] F. Porikli and O. Tuzel. Human body tracking by adaptive background models and mean-shift analysis. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2003.

[PTVF92] W.H. Pressm, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. Numerical recipies in c. *Cambridge Univ. Press*, 1992.

[PYL05] N.S. Peng, J. Yang, and Z. Liu. Mean shift blob tracking with kernel histogram filtering and hypothesis testing. *PRL*, 26(5):605–614, April 2005.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[RJ93] L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Signal Processing Series. Prentice Hall, 1993.

[RL08] K. Rieck and P. Laskov. Linear-time computation of similarity measures for sequential data. *JMLR*, 9:23–48, 2008.

[Rob05] I. Robertson, N. Reid. Behaviour understanding in video: a combined method. In *IEEE Proc. on ICCV*, volume 1, 2005.

[RRG+04] J. Rymel, J. Renno, D. Greenhill, J. Orwell, and G. A. Jones. Adaptive eigen-backgrounds for object detection. In *Image Processing, 2004. ICIP '04. 2004 International Conference on , vol.3, no.pp. 1847- 1850*, pages 24–27, 2004.

[SFGK02] O. Schreer, I. Feldmann, U. Golz, and P. Kauff. Fast and robust shadow detection in videoconference applications. In *Video/Image Processing and Multimedia Communications 4th EURASIP-IEEE Region 8 International Symposium on VIPromCom*, pages 371–375, 2002.

[SG00] C. Stauffer and E. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Recognition and Machine Intelligence (TPAMI)*, 22(8), 2000.

[SG02] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 2:252 Vol. 2, 2002.

[SH81] B.G. Schunck and B.K.P. Horn. Determining optical flow. In *DARPA81*, pages 144–156, 1981.

[SM00] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on PAMI*, 22(8):888–905, 2000.

[SMO99] Jrgen Stauder, Roland Mech, and Jrn Ostermann. Detection of moving cast shadows for object segmentation. *IEEE Transactions on Multimedia*, 1:65–76, 1999.

[SS02] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[STB+06] A. Senior, A.and Hampapur, Y. Tian, L. Brown, S. Pankanti, and R. Bolle. Appearance models for occlusion handling. *Image and Vision Computing*, 24(11):1233–1243, 2006.

[SWTO04] C.F. Shan, Y.C. Wei, T.N. Tan, and F. Ojardias. Real time hand tracking by combining particle filtering and mean shift. In *AFGR04*, pages 669–674, 2004.

[TA77] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill Posed Problems*. W. H. Winston, Washington, D.C., 1977.

[TC04] J. Shawe Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[TOC08] T. Tommasi, F. Orabona, and B. Caputo. Discriminative cue integration for medical image annotation. *Pattern Recogn. Lett.*, 29(15), 2008.

[Vap98] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[VDLPB03] M. Van Der Laan, K. Pollard, and J. Bryan. A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation 73*, (8):575–584, 2003.

[VGK02] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering similar multidimensional trajectories. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, page 673, Washington, DC, USA, 2002. IEEE Computer Society.

[VSV07] S. V. Vishwanathan, A.J. Smola, and R. Vidal. Binet-cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *Int. J. Comput. Vision*, 73(1):95–119, 2007.

[WADP97] C. R. Wren, A. Azarbayejiani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. In *IEEE Trans. on Pattern Analysis and Machine Intelligence, IEEE Computer Society Press*, volume 19(7), 1997.

[Wah90] G. Wahba. *Spline models for observational data*, volume 59. SIAM, Philadelphia, PA, 1990.

[WB] G. Welch and G. Bishop. An introduction to the kalman filter. *http://www.cs.unc.edu/˜welch/kalman/kalmanIntro.html*.

[WB95] H. Wang and M. Brady. Real-time corner detection algorithm for motion estimation. *IVC*, 13(9):695–703, 1995.

[WF00] W. Wong and A. Fu. Incremental document clustering for web page classification, 2000.

[WVDM00] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158, 2000.

[WW93] D. Wagner and F. Wagner. Between min cut and graph bisection. In *18th International Symposium on Mathematical Foundations of Computer Science*, pages 744–750. London Springer, 1993.

[XG05] T. Xiang and S.G. Gong. Video behaviour profiling and abnormality detection without manual labelling. In *ICCV05*, pages II: 1238–1245, 2005.

[XY04] Y. Xiong and D. Yeung. Time series clustering with arma mixtures. *Pattern Recognition*, 37(8):1675 – 1689, 2004.

[YDD05] C. Yang, Ramani D., and L. Davis. Efficient mean-shift tracking via a new similarity measure. In *CVPR*, volume 1, pages 176–183. IEEE Computer Society, 2005.

[YF05] W. Yan and D. A. Forsyth. Learning the behavior of users in a public space through video tracking. In *WACV-MOTION '05: Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05) - Volume 1*, pages 370–377. IEEE Computer Society, 2005.

[YL92] Y.H. Yang and M.D. Levine. The background primal sketch: An approach for tracking moving objects. *MVA*, 5:17–34, 1992.

[ZJ05] Y. Zhang and Q. Ji. Active and dynamic information fusion for facial expression understanding from image sequences. *IEEE Trans. on PAMI*, 27(5):699–714, 2005.

[ZSV04] H. Zhong, J.B. Shi, and M. Visontai. Detecting unusual activity in video. In *CVPR04*, pages II: 819–826, 2004.

[ZYS09] H. Zhou, Y. Yuan, and C. Shi. Object tracking using sift features and mean shift. *Comput. Vis. Image Underst.*, 113(3):345–352, 2009.