PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Trajectory Analysis for Event Detection in Ambient Intelligence Applications

Nicola Piotto

Advisor:

Prof. Francesco G.B. De Natale

Università degli Studi di Trento

February 2011

a mamma e papà...

Abstract

The automatic understanding of human activity is probably one of the most challenging problems for the scientific community. Several application domains would benefit of such an analysis, from context-aware computing, to area monitoring and surveillance, to assistive technologies for elderly or disabled, and more.
In a broad sense, we can define the activity analysis as the problem of finding an explanation coherent with a set of observations. These observations are typically influenced by several factors from different disciplines, such as sociology or psychology, but also mathematics and physics, making the problem particularly hard. In the last years, also the computer vision community focused its attention on this area, producing the latest advances in the acquisition and understanding of human motion data from image sequences. Despite the increasing effort spent in this field, there still exists a consistent gap between the numerical low-level pixel information that can be observed and measured, and the high abstraction level of the semantic that describes a given activity. In other words, there exist a conceptual ambiguity between the image sequence observations and their possible interpretations. Although several factors are involved, the activity modeling and the comparison strategy play crucial roles. In this proposal, a correlation between activity and corresponding path has been assumed.
In light of this, the work carried out tackles two strictly related issues: (i) obtaining a proper representation of human activity; (ii) define an effective tool for reliably measuring the similarity between activity instances. In particular, the object activity is modeled with a signature obtained through a symbolic abstraction of its spatio-temporal trace, allowing the application of particular high-level reasoning for computing the activity similarity. This representation is particularly effective since it provides a smart way to compensate the noise artifacts coming from low-level modules (i.e., tracking algorithms), allowing also the possibility of considering interesting properties, such as the invariance to shift, rotation, and scale factors. Since any complex task may be decomposed in a limited set of atomic units corresponding to elementary motion patterns, the key idea of this representation is to catch the object activities by suitably representing their trajectories through symbols. This syntactic activity description relies on the extraction and on the symbolic coding of meaningful samples of the path, while the similarity between trajectories is computed using the so-called approximate-matching, thus casting the trajectory comparison problem to a string matching one.
Also another representation scheme has been adopted, coding the signature according some

relevant spots in the environment: in this case, the structural pattern information is coded in ad-hoc Context-Free Grammars, and the matching problem is solved through the parsing of the incoming string according the defined rules.

Keywords:
[Activity Analysis, Trajectory Analysis, Trajectory Representation, Trajectory Matching, Ambient Intelligence, Visual Surveillance, Approximate matching, Context-Free Grammars]

Acknowledgements

There are lots of people I would like to thank for a huge variety of reasons.

Firstly, I would like to express my gratitude to my advisor, Prof. Francesco G.B. De Natale, for his excellent supervision on my research and his assistance in this past years in pointing me in the right direction. I would also like to deeply thank Dott. Nicola Conci, for the insightful discussions, the consistent support over these years and the contribution to my work.

A special thanks go to all the people that shared with me the Multimedia Signal Processing and Understanding Laboratory, from the students to all the academic members, for all those comments and suggestions helping enhancing the quality of my research. I'm particularly graceful to my friends (before than colleagues) Mattia, Valentina, and Silvia, who walked with me through this PhD trip, sharing many good and bad moments, making my PhD life easier. Thank you for the moments we spent together, your support and encouragements in critical situations, your friendship.

I'm thankful also to all the other colleagues and friends in the Department, not only for the technical help, but in particular for making these years more funny and interesting.

Finally, it is always not enough to express my deep gratitude to my family, especially my Mom and Dad, for their unconditional love: they never stopped believing and supporting in me. Without them I would not survive the long journey of my PhD study.

Last but not least, I want to thank Julie, for being entered in my life, for making me discover what is the happiness.

Contents

Chapter 1

Introduction

In this chapter an introduction to this dissertation will be given. In particular, we provide an overview of the visual technology for activity analysis, focusing on the main issues related to visual information extraction for the automatic analysis of activities. The main objectives and the novel contributions of this thesis are also presented. Finally, we describe the organization of this document.

1.1 The Context

Automatic activity understanding in dynamic scenes is a very complex and ambitious goal. In particular, the reasoning task becomes even more complicated when it has to deal with human-populated scenarios, making it particularly appealing. The emulation of such a perfect system as the Natural Vision System represents, without any doubt, a real challenge both from a scientific and technological point of view. In spite of several difficulties involved, or because of them, the automatic human activity analysis has gained significant attention from the computer vision community, becoming a very active research field. However, despite the strong interest and the substantial advances achieved in the last years, it still constitutes an ambitious open problem that is far from being solved.

The interest about this area is pushed forward by two main factors. On the one hand, the number of potential applications is continuously increasing, including not only smart video safety (Fig. 1.1-(a)) and video surveillance (Fig. 1.1-(b)), but also automatic traffic/road monitoring (Fig. 1.1-(c)), crime and dangerous situations alert (Fig. 1.1-(d), (e)), semantic-based video classification, indexing and retrieval engines (Fig. 1.1-(f)), automatic sport statistics computation (Fig. 1.1-(g), athlete training and orthopedic therapy, machine content annotation (Fig. 1.1-(h)), smart human-computer interfaces (Fig. 1.1-(i)).
On the other hand, the fast development of computer and video technologies, and the cost reduction of the capturing devices recorded in the last years has brought the systems for video analysis to be increasingly and widely applied in daily life, providing to the human operators the potential access to a growing amount of data. However, if we think

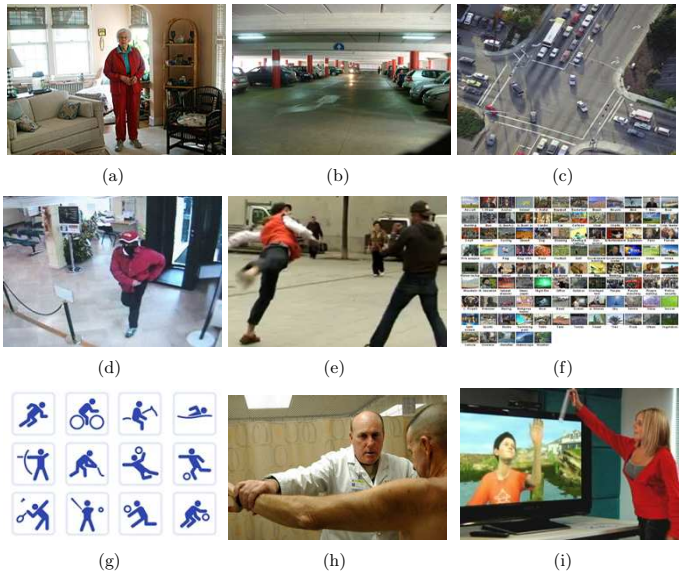


Figure 1.1: Sample of significant applications for video analysis.

about a typical surveillance scenario, the volume of video feed is so large that it quickly overwhelms the ability of security personnel to analyze and respond to the data in a time-critical fashion. In fact, most of the commercial solutions focus on the recording process, while it would be highly desirable having effective techniques for event analysis, in order to identify the occurrence of specific situations, and issue warnings automatically. In other words, the video data is used "after-the-fact" as a forensic tool, losing its primary benefit of an active, real-time medium. On the contrary, a continuous 24/7 analysis is required, to alert security officers about burglary in progress, or suspicious loitering in a parking, while there is still time to react to the on going situation.

1.2 The Problem

One of the main difficulties in designing such an automatic system lies in discriminating unusual activities against the backdrop of typical behaviors of people in each particular environment. What is informative in monitoring systems includes recognizing activities, knowing typical and threatening patterns of activities, inferring intents, and possibly seeing through attempts to mask threats with normal behaviors. Finding this information

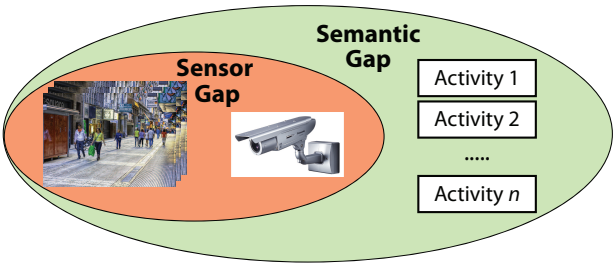


Figure 1.2: *Sensor* and *semantic gaps* configuration in an activity analysis framework.

in the wide spectrum of low-level video data is a very complex and challenging task that goes far beyond the simple provision of sensors over wider areas. The task of interpreting the activity captured in an image sequence involves subjectivity, and is strongly related to an explanation based on human-defined concepts. However, the input data available to accomplish this task come from vision-based techniques, which are basically quantitative. Therefore, an intermediate processing layer to convert quantitative data into higher-level concepts has to be put between the vision-based techniques and the activity interpretation reasoning.

When extracting/deriving qualitative information from image sequences, two main phases are usually carried out, and both of them contribute with some uncertainty to the final activity interpretation. In Fig. 1.2 the conceptual data flow is reported. Initially, a given amount of error is pulled out by the low-level computer vision device due to the so-called *sensory gap*, which refers to the lack of accuracy in the low-level information extraction from the image. In this regard, several open issues can be found in the scientific literature, for example about the object features estimation/tracking in cluttered scenes. In fact, when dealing with a poorly illuminated or crowd environment, many difficulties arise when trying to distinguish the targets from the background, bringing to errors in the estimated motion, and thus to the final activity. Moreover, multiple moving targets increase the probability of occlusions, introducing additional problems in the object shape extraction.
On the other hand, the activity modeling has to consider the *semantic gap*, which in particular refers to the conceptual ambiguity between the image sequence observed information and its possible interpretations. In fact, although a computer vision algorithm can learn a class of activity patterns, it cannot automatically extract any semantic meaning of the results. In order to bridge this gap, some constraints have to be introduced, such as the correlation between the activity concepts with some low-level feature.

1.3 Proposed Solution and Innovation

The effects of the *sensory gap* can be somehow relaxed by considering the installation of multiple cameras, or using a combination of different low-level sensors (e.g., infrared or PTZ cameras, microphones, RFIDs), and relying on the concept of *diversity*. Instead, how to bridge the *semantic gap* is still an open issue. A common solution is to select a model for the activity, and adopt a human-labeled set of samples for its training. The model selection is crucial, and usually is carried out on the basis of some heuristic assumptions. In general, a reasonable option can be to assume some kind of correlation between the conceptual activity and the effective path followed by an object for carrying out that activity. In this way it's possible to associate to some particular activity a specific set of motion patterns, bringing the problem of activity recognition to a trajectory analysis problem.

In this proposal, this last direction has been followed, supposing a correlation between the activities samples and the correspondent spatio-temporal path evolution. Since the

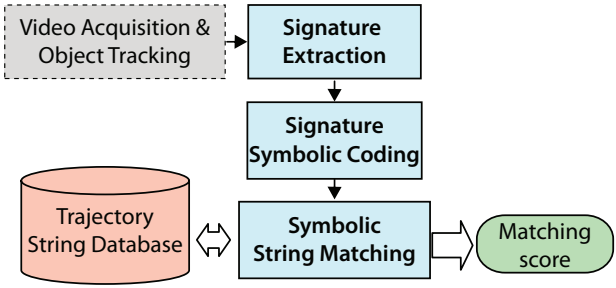


Figure 1.3: High-level flowchart of the proposed solution. The blocks considered in this work are highlighted in light blue and bold font.

sensory gap is somehow related and dependent to the particular device (or device configuration) employed, it can be not trivial to develop some general strategy to constraint the corresponding uncertainty; instead, we focused on the development of high-level representation/matching techniques in order to bridge (or at least reduce) the *semantic gap*.

In light of this, the final goal of this work is two-fold:

(i) overcome the actual constraints in trajectory representation and matching techniques and develop an alternative solution capable of fully exploit the activity informational content of the object trajectories in real-time;

(ii) build a solution capable of discerning among different high-level activities by merging the information from both the objects paths and the context surroundings (e.g., environment map, furniture and entry/exit locations, etc.).

Concerning the first goal, we propose to split complex trajectories into elementary segments and code them with an incremental symbolic-syntactic representation paradigm; in addition to interesting invariance properties, the proposed activity representation allows applying bio-inspired string-matching schemes [85][68] to evaluate the similarity among paths. In the same spirit of spell-check correctors, these algorithms rely on enhanced versions of the edit-distance, measuring the similarity among sequences as the number of elementary operations to bring on string into the other: elementary operations are the insertion, deletion, and substitution of a symbol, as well as the insertion of an empty gap.

The key idea of this proposal is to extract and encode into high-level symbols the spatio-temporal signature of the activity, and employ a syntactical matching to reach a flexible comparison between activity patterns. The rules leading the matching procedure have to be designed and tuned according to the meaning of each symbol that for a trajectory represents a specific motion pattern. In Fig. 1.3, the flowchart of the proposed solution is sketched: after the stream acquisition and the object tracking phases (i.e., the light grey dashed block), the object trajectory signature extraction step and its symbolic coding are carried out. Finally, the matching phase provides for an approximate alignment between the considered sequences.

Such an approach introduce several advantages over the solutions proposed in the actual state of art:

- it allows an on-line symbols-based activity spatio-temporal representation and comparison;
- it includes the possibility of considering the invariance to shift, rotation, and scale;
- it offer a strong tool to effectively deal with the problems from the noise affecting the sequences;
- it can be applied to sub-trajectory matching problems;
- it can be easily extended in multidimensional spaces.

In order to enhance the abstraction level in the activity representation, we also explored the use of a different representation paradigm, which encodes the activity through a topological representation relying on some interesting areas (i.e., *hot spots*). In this spirit, the *signature* corresponding to a given activity is represented with the concatenation of the *hot spots* the object has interacted with[1]. Intuitively, this second representation is in general more robust to the noise, and it is capable of extracting a 'structural' characterization of the activity. In order to fully exploit this information, the patterns of a given activity are encoded according some automatically learned Context-Free Grammars rules: in this context, the activity detection/recognition is casted to a string-parsing problem. The adoption of CFG formalism significantly enhances the flexibility in the matching phase, allowing the activity reasoning module to handle particularly complex pattern

[1]The characterization of 'interaction' will be given in detail in Chapter 5

configurations. The main advantage introduced by this representation is its capability of catching the structural patterns characterizing a specific activity.

While working either in the symbolic or in the numerical domain, a fundamental question arises about the *optimal* representation for trajectory comparison. The raw representation is in general richer from an informative point of view, than any symbolic representation. However, given its sensibility to the noise, a quantization process is often introduced. The noise power plays a crucial role: when it is *small*, the application of a quantizer would only deteriorate the representation (due to the quantization error), and thus the comparison results. Instead, when the noise power is large enough, the quantized data representation is intuitively more convenient. We formalized this idea in order to define a reasoning for obtaining the best representation for trajectory matching, under a set of assumptions for noise and trajectory data. The work is in a preliminary phase, however the results are promising. The paper has been submitted to a conference and it is under revision process.

1.4 Structure of the Thesis

The thesis is structured in 6 Chapters. The present section gave an introductory overview on the research context of the thesis, including the issues related to the automatic visual activity analysis, the contribution of this proposal, and the advances with respect to the state of art solutions.

The initial section (Chapter 2) presents the literature about the techniques for trajectory representation and comparison. In this section several approaches are presented and compared, highlighting strengths and weakness of the selected solutions.

In Chapter 3 is presented a comprehensive description of the string-based representation and matching strategy developed for activity detection in ambient intelligence applications. Here, the original two-dimensional formulation is given, while in Chapter 4 a three-dimensional extension of the algorithm is proposed together with an algorithm for the hierarchical trajectory classification.

Chapter 5 proposes an alternative solution for the trajectory-based event detection: here, a topological representation of the activity is considered, that is coded into a set of Context-Free Grammar rules. The sequence matching is carried out as a sophisticated string-parsing.

In Chapter 6 is reported a theoretical study on the effectiveness of symbolic and numerical approaches for trajectory representation in presence of varying noise. This work aims at showing the advantages of a symbolic (i.e. quantized) signal representation with respect to a numerical (i.e., raw) representation in presence of significant noise. An optimization procedure is formulated in order to obtain the quantization scheme introducing the minimal signal distortion with respect to the noise-free signal versions.

Finally, in Chapter 7 the final remarks on the thesis are drawn.

Chapter 2

State of the Art

As underlined in the introduction section, the representation paradigm and the matching strategies play a crucial role in the process of automatically detect and/or recognize and activity pattern from video streams. Several alternatives have been proposed in the last years, in particular the focus of this chapter is to present a survey on the most recent advances in representation and analysis of video object trajectories for classification and recognition purposes. The main methodologies for the description of motion trajectories, as well as the matching techniques and similarity metrics will be reviewed. Strengths and weaknesses of the different solutions will be discussed through a comparative analysis, taking into account performance and implementation issues.

2.1 Trajectory representation

The problem of trajectory representation mainly consists of achieving an approximation of the raw path through some parametric curve. The simplest model consists in the use of chain codes [45], or piecewise linear approximations [72]. More accurate representations may use curvilinear approximations such as polynomials [90] or splines [61]. The above methods consider the trajectory as a 2D projection of the spatial displacement of the point in the scene, even though 3D representations are gaining considerable ground in the research community. In the following sections we will mainly focus on 2D representations, since they are more widespread. However, some of the methods introduced for 2D have a straightforward extension to 3D, also considering that in most cases the multidimensional analysis can be carried out by combining different 2D views. Even though other features such as velocity, motion direction, temporal offset, or view invariant tensor null-space representations [17] [18] have been considered in the literature, for the sake of conciseness we will focus on spatial approximation, making it clear that the methods described hereafter could be extended to the other trajectory features.

Part of this Chapter appears in:
N.Piotto, M.Broilo, G.Boato, N.Conci, F.G.B. De Natale, "Object Trajectory Analysis in Video Indexing and Retrieval Applications" in D. Schonfeld, C. Shan, D. Tao, L. Wang, Video Search and Mining, Berlin: Springer-Verlag, 2010, pp. 1-30 - Studies in Computational Intelligence - ISBN: 978-3-642-12899-8.

2.1.1 Polygonal approximation

Polygonal approximation has been used in many pattern recognition problems, including shape and contour representation. The idea is to interpolate the raw trajectory with a piecewise linear curve with limited number of vertices. Since most of the information is connected to the points of maximum curvature, this representation turns out to be significant as well as compact. The resulting polygon should fulfill two requirements: (i) best fit of the original curve, and (ii) minimum number of segments. The two conditions are conflicting, then a trade-off has to be found by concurrently solving two optimization problems:

- *Error minimization problem:* given a set of N points representing a raw trajectory $T = \{t_i\} = \{x_i, y_i\}_{i=1}^{N}$, find the polygonal curve $P = \{p_j\} = \{x_j, y_j\}_{j=1}^{M}$ with a number of line segments M, so that the approximation error $\epsilon(T, P)$ is minimized.
- *Number of dominant points minimization problem:* given a set of N points representing a raw trajectory T, find the polygonal curve P with the minimum number of segments M so that the approximation error $\epsilon(T, P)$ does not exceed a maximum tolerance ϵ_{tol}.

The most common criteria used for optimization are the compression ratio $CR = N/M$ and the integral square error [81] between vertices of T and linear segments of P. Fig. 2.1 shows a real trajectory and its approximation. The final number of segments varies according the reconstruction error threshold imposed. Many polygonal approximation

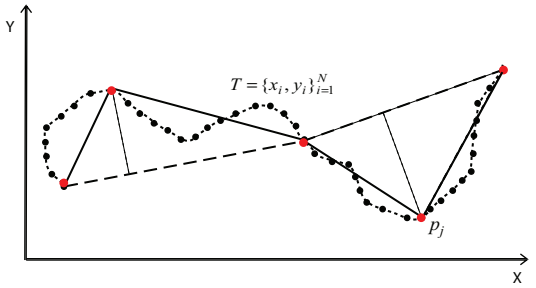


Figure 2.1: Polygonal approximation.

techniques have been proposed in the literature. Jointly optimal algorithms are quite slow, with a complexity in the order of $O(N^2)$ or even $O(N^3)$. It is possible to reduce the complexity to $O(Nlog(N))$ by focusing on one of the two minimization problems [14]. Heuristics are also widely used.

Approximation methods differ upon specific requirements, such as the application context (e.g., pedestrians *vs.* vehicles), and the error metrics used for evaluation. The

most important approximation methods can be roughly classified into four categories: (1) sequential, (2) split & merge, (3) dominant point-detection, and (4) optimization algorithms approaches. In the next paragraphs we will provide a general overview of them.

2.1.1.1 Sequential tracing approaches

In sequential algorithms, the trajectory is progressively scanned and a mismatch condition evaluated, when the error exceeds a threshold a new segment is started. Algorithms are usually fast and can be applied in real-time, thus making them attractive in trajectory representation even though the accuracy of the approximation is quite limited. Among these methods, Sklansky and Gonzalez [84] proposed a scan-along procedure for digitized curves, which starts from a point chosen randomly and tries to find the longest line segments sequentially. Kurozumi and Davis [51] proposed instead a minimax method, which determines the segments by minimizing the maximum distance between a given set of points and the corresponding segment. Wall and Danielsson [92] proposed a sequential method, which scans the points and outputs a new segment when the area deviation per length unit of the current segment exceeds a pre-defined error. Ray and Ray [50] determine the longest possible line segments with the minimum possible error. All the aforementioned algorithms are designed to solve one of the minimization problems and reach the solution in $O(logN)$ or $O(N)$ steps of binary search [14].

2.1.1.2 Split & merge methods

Split-based methods use a top-down approach where the coarsest approximation is the segment connecting the first and last point of the path. If the approximation is not satisfactory, it is refined by recursively splitting the segments until the accumulated error reaches a predefined threshold or the maximum number of segments is exceeded (see Fig. 2.2). Depending on the split procedure, the number of pieces at the end of the process may be higher than needed. In this case, a merging may occur to re-connect adjacent segments with similar direction. The depth of the split is driven by the application requirements and can be adjusted by varying the split criterion or the thresholds. The algorithm requires the availability of the whole path.

The most popular algorithm in this field is a heuristic method known as "Douglas-Peucker" [24], adopted in both [5] and [25]. The iterative procedure splits the curve into smaller elements and, at each iteration, calculates the distance of each vertex from the original curve. The stop condition is fulfilled when the cumulative distance is smaller than a given tolerance ϵ. The complexity of the method is $O(N^2)$ in the worst case, and $O(NlogN)$ on average. Leu and Chen [53] presented a hierarchical merging method, which considers the trajectory as composed by a number of consecutive arcs. They are replaced by their chord only if the arc-to-chord deviation results lower than a given threshold. Ansari and Delp [3] proposed a technique that first uses Gaussian smoothing to reduce the noise and then selects the points with maximal curvature as break points. The split

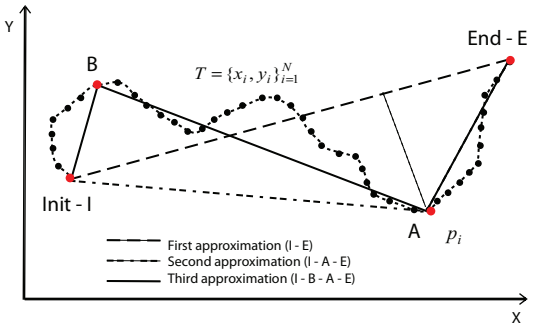


Figure 2.2: Polygonal approximation using iterative splitting.

& merge process is then applied to the obtained samples. Ray and Ray [79] proposed an orientation-invariant and scale-invariant method by introducing the use of ranks of points and normalized distances. In this case, the approximation returned by the split-and-merge may be far from the optimal one if the initial segmentation is not accurate.

2.1.1.3 Dominant point-detection methods

The core idea of this class of algorithms is that a shape is well represented by its high-curvature points [4]. Then, a contour can be described by using such points as the vertices of a piecewise linear interpolation. Several heuristics have been designed to this purpose. Teh and Chin [88] determine the curvature at each point based on a support region, and detect the dominant points through a non-maxima suppression process. Other approaches rely on the detection of salient points. Held et al. [32] first apply a coarse-to-fine smoothing to identify dominant points, and then define a hierarchical approximation based on perceptual significance. Zhu and Chirlian [102] determine the importance of each point by transforming the curve into polar coordinates and then calculating the relevant derivatives. In this class of algorithms it is also possible to identify methods that search for the most significant points using relaxation labeling [62]. The paper focuses on the contour extraction of shapes, but the extension of the work to trajectory analysis is straightforward. In this approach, the left and right slopes and the curvature are evaluated and associated with an attribute list to each point of the input curve. This information determines the initial probability of the current point to be a *side* (a linear piece in the case of a trajectory), or an *angle* (a point with strong curvature). The relaxation process iteratively updates the probabilities until convergence. The obtained *angle* points can therefore be used as a meaningful representation of the whole trajectory.

Similarly to split & merge, the above methods require the availability of the whole trajectory. Moreover, their performance is bounded by the accuracy achieved in the

evaluation of the curvature.

2.1.1.4 Optimization algorithms

The approximation problem is here considered as an optimization task where the global error is the cost to be minimized. The search of the solution that provides the minimum error can be performed by stochastic optimization methods (e.g., genetic algorithms [34], ant colony [98], particle swarm optimizations [99]), or by local optimization methods (e.g., tabu search [97] and vertex adjustment [70]). The initialization is obtained based on some heuristics, and the approximation is progressively improved towards the minimum of the global error. The final solution can be considered nearly optimal, although the global minimum is usually not guaranteed. In these algorithms, the trajectory points are typically examined in sequential order. The computational cost of these algorithms is pretty high, but the achievable results have a higher fidelity since they are specifically designed to climb local minima associated to suboptimal representations.

2.1.2 Spline approximation

A completely different approach consists in the use of splines [61]. Splines are smooth curves (typically polynomials) that interpolate a set of points in a plane. Among the many different spline types, B-Splines (a generalization of the Bèzier curves) are very commonly adopted:

$$A(u) = \sum_{i=0}^{n} N_{i,d}(u) p_i \tag{2.1}$$

where $p_i, i = 1, 2, \ldots, n$ are the control points and $N_{i,d}(u)$ are the B-spline basis functions of order d. Control points represent the points of the original trajectory. To create the spline approximation $A(u)$, a vector of *knots* $U = \{u_0, u_1, \ldots u_{m-1}\}$ is needed. Given the degree of the polynomial d and n control points, the number of *knots* m should be equal to $n + d + 1$. U is a set of non-decreasing values in $[u_0, u_{m-1}]$ and all basis functions lie in this interval. The i-th basis function $N_{i,d}$ is calculated using the Cox de Boor formula:

$$N_{i,0}(u) = \begin{cases} 1 & if \quad u_i \leq u < u_{i+1} \\ \\ 0 & otherwise \end{cases}$$

$$N_{i,d}(u) = \frac{u - u_i}{u_{i+d} - u_i} \cdot N_{i,d-1}(u) + \frac{u_{i+d+1} - u}{u_{i+d+1} - u_{i+1}} \cdot N_{i+1,d-1}(u). \tag{2.2}$$

The spline approximation consists of determining the optimal coefficients of the polynomial model to fit the trajectory. If the trajectories to be described are complex, higher degree polynomials may be used, or more often the curve is segmented and represented through piecewise polynomial fitting. In this case, trajectory segmentation problems arise similar to those discussed in polygonal approximation. Usually, the degree of the polynomial is kept low (e.g., third degree) to ensure smoothness and avoid oscillations.

Spline approximation can be considered as a subset of the polynomial approximations used in many areas of computer graphics. In the specific case of video indexing and retrieval it is mainly used to smooth the noisy path of a moving object in tracking. An important property of the approximated curve is that it is invariant to affine transformations. Another advantage of the B-splines is that a local change in the raw trajectory value does not affect the whole approximated curve, because each control vertex influences $\pm k/2$ segments of the polygon in u. A sample curve approximated using a spline is shown in Fig. 2.3. Even though splines are usually computed directly on the raw

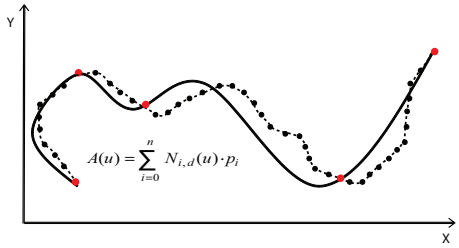


Figure 2.3: Spline approximation.

trajectory for computational reasons, it may be desirable to adopt a curvature detection algorithm to extract dominant points to be used as control points. The above problem can be formulated as a nonlinear optimization problem as follows. Given:

- a set of points of the raw trajectory $t_k,\ k = 1, 2, \ldots, K$ in the plane
- a B-spline curve $A(u) = \sum_{i=0}^{n} N_{i,d}(u) p_i$ with control points p_i
- the order and the knots of the B-spline curve (not subject to optimization)

find the control points $p_i, i = 1, 2, \ldots, n$ such that f, defined in (2.3), is minimized

$$f = \frac{1}{2} \sum_{k=1}^{K} ||A(u) - t_k||^2 + \lambda f_s. \tag{2.3}$$

where f_s is a regularization term to ensure a smooth curve, and λ is a positive constant that determines the weight of f_s.

2.1.3 PCA coefficients

Principal Components Analysis (PCA) is a method that reduces data dimensionality based on the analysis of samples covariance. As such, it is suitable for data sets in multiple dimensions, such as trajectory points [64]. The idea is to characterize a raw

trajectory only with its principal components, getting rid of all features that do not convey significant information. Given a vector trajectory T made by a set of random variables and known correlation matrix C, the $k-$th principal component (PC) is given by an orthonormal linear transformation:

$$PC_k = a_k T \tag{2.4}$$

where a_k is an eigenvector of C corresponding to its $k-$th largest eigenvalue λ_k. Algorithms for the computation of the PCs are usually based on correlation or covariance [42]. Another approach that exploits PCA, is presented Bashir et al. in [6], where the authors first segment trajectories using a curvature zero-crossing approach, followed by a clustering routine. The method applies then a two-level PCA analysis, in which the principal components are first extracted from the whole object trajectory, and successively analyzed to determine the corresponding sub-trajectories. In other approaches the PCA is applied to the null space representation [17], allowing achieving an affine invariant representation used in retrieval and classification. In [18] [58] extensions of this representation are proposed to allow the classification and retrieval of multiple interactive trajectories.

2.1.4 String-based (symbolic) representation

Syntax-based approaches convert the analytic representation of a trajectory into strings of symbols, to provide a description of the path with a higher-level of abstraction. String-based representation may be applied to raw trajectory samples, or to the approximated representation achieved by one of the methods described above. Once the key points of the trajectory are extracted, they are translated into symbols according to the associated spatio-temporal information, and aligned into strings. Once the information is in symbolic form, several approaches for matching can be considered, in particular it is possible to rely on some alignment strategies used in bioinformatics to match genomic sequences [68].

Different approaches can be used for bringing the numerical trajectory samples to a symbolic domain. For example, a very naive approach is proposed in [20]: here, a static grid is built over the map of the environment, and a spatial sampling of the original path is operated followed by a merging procedure for removing the useless regions. In [40], the authors propose to use standard independent probabilistic event detectors over the incoming trajectory data (i.e., a bank of HMMs trained on each specific primitive they want to consider). Instead, the works in [35] and [76] consider some trajectory-related information (e.g., direction, local speed, and more).

String-based algorithms have been used in computer vision especially for shape classification [29] and they have been introduced more recently in object trajectory representation. For instance, in [95] the authors assume that each trajectory segment is labeled with a semantic symbol. Using the chain of successive symbols, a support vector machine is trained to classify different events. In this case, the clustering scheme requires a lot of training samples and each fragment has to be labeled in advance. In the work of Chen [35], key points are labeled using characters to map the moving direction. In this way, strings can be compared for both matching and clustering purposes. This representation

can easily tackle the problem of partial matching, making it possible to detect sub-strings within the whole sequence of symbols. Furthermore, this representation allows to easily achieve the invariance to spatial shift and scaling. [76] provides a description of the trajectory using syntactical elements. Here, key trajectory points are extracted and represented by three characters, corresponding to angle, speed and temporal offset with respect to the previous point, to achieve a full spatio-temporal representation.

The advantage of syntactical approaches is that the matching phase can be simply implemented as a matching of words, like in text processing tools, typically using similarity metrics such as the *edit distance* [80]. As explored deeper in Section 2.2, other matching approaches propose for this representation, relying for example on graphical probabilistic models (e.g., HMM, DBN, CFG).

Table 2.1 reports a comparative analysis of the representation methods cited above.

2.2 Trajectory matching

This section deals with matching strategies that can be adopted to measure the similarity among trajectories. In developing a matching algorithm several issues should be taken into account. In fact, the extraction of object trajectories from video streams is typically imprecise due to environmental noise, illumination variations, processing errors, occlusions, and so on. The joint effect of these uncertainties typically leads to noisy trajectories that contain gaps and outliers. Moreover, even trajectories referring to similar events may present significant differences in several respects, such as initial direction and location, spatial length, temporal duration, and sampling rate, thus leading to mismatches. According to the abstraction level adopted for trajectory representation, matching techniques can be roughly divided into three categories: dynamic, statistical, and vector-based. Dynamic matching includes a set of simple yet effective comparison tools that can be applied to raw or filtered samples, and are able to deal with limited trajectory misalignments. Statistical matching is more sophisticated and requires a pre-processing to extract a set of consistent low-level features, the similarity measure is then obtained by comparing the distribution of query and target samples in the feature space. Finally, vector matching algorithms rely on a high-level representation of the trajectories, where the feature vectors are mapped into symbols. This representation strongly simplifies the matching phase, which can be achieved through simple metrics (e.g., L_p-norm, Hausdorff or city block distances, string alignment techniques) and weighted combinations of the features.

2.2.1 Dynamic matching

Methods referred to as dynamic matching are basic comparison tools, enabling the user to process sequences of different lengths. This is a key feature, since in real applications it is almost impossible to impose the same length to trajectories. The main feature of these methods is the capability of applying a local warping to the sequences, in order to achieve the best alignment. Depending on the application requirements, the stretch may

Category	Description	Strengths and Weaknesses
Polygonal approximation	Interpolate the trajectory with piecewise linear curves with the minimum number of vertices.	**Pros:** simple and fast. **Cons:** depends on thresholds.
Sequential tracing [84] [51] [50]	Scan the trajectory to evaluate error conditions and start new segments.	**Pros:** minimize the delay. **Cons:** segmentation is not optimal.
Split and merge [5] [25] [53] [3] [79]	Iterative split until the accumulated error is below a threshold.	**Pros:** simple and fast. **Cons:** entire trajectory required.
Dominant points detection [88] [32] [102] [62]	Find high-curvature points, and connect these key points via linear segments.	**Pros:** the trajectory curvature is preserved. **Cons:** entire trajectory required.
Optimization algorithms [34] [98] [99] [97] [70]	From an initial approximation, iterate to find the global approximation error.	**Pros:** near-optimal solution is found. **Cons:** high computational cost.
Spline approximation	Interpolate points with a smooth curve.	**Pros:** approximation as a smooth curve. **Cons:** entire trajectory needed.
Polynomial interpolation [90]	Numerical analysis to represent an interpolated curve with a polynomial.	**Pros:** numerically optimal solution. **Cons:** high computational cost; order of the polynomial dependent on the number of keypoints.
B-spline approximation [83] [57] [39]	Piecewise polynomial approximation.	**Pros:** keypoint choice affects only a small part of the approximated curve. **Cons:** non-linear approximation.
PCA [42] [6] [16]	Characterize a raw trajectory only with its principal components.	**Pros:** compact representation. **Cons:** part of the information is lost.
String-based representation	High-level description using symbols.	**Pros:** add semantic information. **Cons:** requires preprocessing.
Points labeling [35]	Attach to a point/set of points a text label describing motion primitives.	**Pros:** it accepts textual queries. **Cons:** machine learning needed.
Characters coding [76] [8]	Bring key points information into symbols and code the sequence as a single string.	**Pros:** easy local and global matching. **Cons:** key points detection/quantization is required.

Table 2.1: Trajectory representation: a comparative analysis.

be operated either in the temporal or spatial domain, thus providing time warping and spatial warping, respectively.

Concerning the temporal domain, dynamic time warping (DTW) is a distance measure used in 1-D time-series comparison [9]. Initially applied to speech signal analysis, it has been recently extended with success to different application domains including sign language recognition [46] and trajectory matching [36], because of its conceptual simplicity and versatility. Basically, the method relies on a classic distance operator (e.g., L_p-norm) and on a particular matching procedure that finds the optimal alignment between the query and the target series, allowing temporally shifted matches between samples. The matching score is calculated as the cumulative distance among samples.

The distance between two generic series $X = \{x_n\}_{i=0}^{N}$ and $Y = \{y_n\}_{j=0}^{M}$ can be measured by constructing a warping path [46], as in Eq. (2.5):

$$W = w_0, w_1, \ldots, w_K \qquad max\{N, M\} < K < (N + M - 1) \tag{2.5}$$

where K is the length of the warp path and $w_k = (i, j)$ where i, j index X and Y, respectively. The warping path involves all samples in the trajectory (i.e., $w_0 = (0, 0)$ and $w_K = (N, M)$); moreover, i and j have to be continuous (i.e., every index in all series has to be used) and monotonically increasing. The distance between X and Y is the optimum warp path that minimizes the overall warping distance, satisfying Eq. (2.6):

$$DTW(X, Y) = min\left\{\frac{1}{K}\left[\sum_{k=1}^{K} w_k\right]\right\} \tag{2.6}$$

where, w_k is the minimum distance between two samples indexes (one from X, one from Y) in the $k-th$ element of the warp path (see Fig. 2.4). DTW presents some major drawbacks

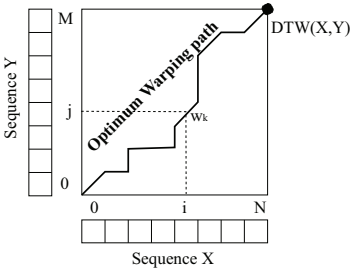


Figure 2.4: DTW: optimum warping path construction.

in the sensitivity to noise and outliers. Furthermore, since the global similarity score is evaluated on the basis of cumulative sample-to-sample distances, different sampling rates (scaling) and misalignments (shifting) may lead to high distances even in the presence of

very similar series. Finally, the computational complexity of the method is relatively high: $O((N+1)(M+1))$ with $N+1$, $M+1$ length of the sequences. Dynamic programming techniques are usually employed to effectively achieve the best alignment and to drastically reduce the complexity. As an example, given the two sequences X and Y, the distance $DTW(X,Y)$ is calculated as:

$$DTW(X_i,Y_j) = min \left\{ \begin{array}{l} DTW(X_i,Y_{j-1}) \\ DTW(X_{i-1},Y_j) \\ DTW(X_{i-1},Y_{j-1}) \end{array} \right\} + d(x_i,y_i) \tag{2.7}$$

where $d(\cdot,\cdot)$ is a distance metric that strictly depends on the employed trajectory representation, $X_i = \{x_0, x_1, \ldots, x_{i-1}\}$ and $Y_j = \{y_0, y_1, \ldots, y_{j-1}\}$. At each sample, the warping distance $DTW(X_i,Y_j)$ between X_i and Y_j indicates the cumulative sum of the local distance $d(x_i,y_j)$ and the minimum of cumulative distances among adjacent samples. In particular, $DTW(X_i,Y_j)$ is the optimum warping path between the first i samples of X and the first j of Y. Regarding this measure, several advances have been carried out allowing enhancing the robustness to noise [23] and reducing the computational complexity [47].

An improved approach is presented in [22], which nearly replicates the DTW matching scheme in the spatial domain with significant algorithmic enhancements. Given two sequences, the Longest Common SubSequence (LCSS) is used to optimize the alignment by finding the longest subsequence between two trajectories. This concept provides a higher flexibility allowing non-consecutive samples and the insertion of gaps. This feature is fundamental, since it introduces the capability of (i) effectively processing paths of different lengths, (ii) coping with different sampling rates, and (iii) partially handling problems related to noise and outliers.

Fig. 2.5 sketches the difference between the alignments of the same two sequences, obtained through temporal and spatial warping, respectively. LCSS leads to a more significant alignment, since it allows excluding some samples from the matching process; on the contrary, DTW requires to match every query sample, thus causing a one-to-many

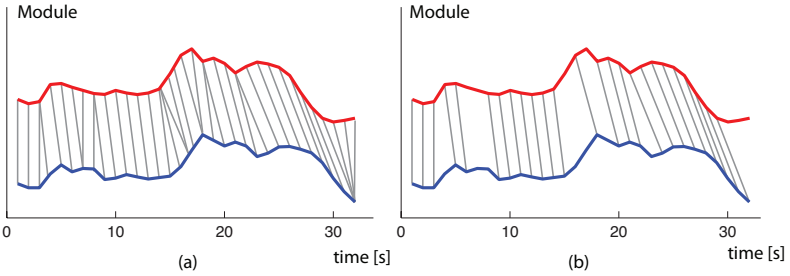


Figure 2.5: Comparison between the alignments obtained through (a) DTW and (b) LCSS.

association between query and target paths (Fig. 2.5-(a)). Similarly to DTW, LCSS relies on standard distance metrics, although employed in a different way. While in DTW the final score is evaluated by computing the sample-to-sample distance, in LCSS the distance is used to check whether two samples are correlated or not [89]. In particular, this strategy consists in checking if the samples in the target trajectory fall within a spatio-temporal region defined in the query. If the condition is verified, the match occurs. The matching region is defined by two thresholds ϵ and δ, in space and time, respectively.

Since the computational burden of the alignment process is quite high, also the LCSS scheme is usually implemented with dynamic programming techniques, with a computational complexity in the order $O((N+1)(M+1))$. Given two 1D time series X and Y, the LCSS distance is evaluated by dynamically computing the matrix coefficients according the following recursion:

$$LCSS_{\epsilon,\delta}(X_i, Y_j) = \begin{cases} 0 & \text{if } i = 0 \quad \text{or } j = 0 \\ 1 + LCSS_{\epsilon,\delta}(X_{i-1}, Y_{j-1})) & \text{if } |x_i - y_j| < \epsilon \text{ and } |i - j| < \delta \\ \max \begin{cases} LCSS_{\epsilon,\delta}(X_{i-1}, Y_j), \\ LCSS_{\epsilon,\delta}(X_i, Y_{j-1}) \end{cases} & \text{otherwise} \end{cases} \tag{2.8}$$

The LCSS algorithm outputs the length of the longest common subsequence between the series. A similarity score in the range [0-1] is then defined as follows:

$$S(X, Y, \epsilon, \delta) = 1 - \frac{LCSS_{\epsilon,\delta}(X_M, Y_N)}{max\{M, N\}} \tag{2.9}$$

Although performing generally better than DTW, LCSS still presents some limits in dealing with significant noise or outliers, since the insertion of gaps is neither penalized nor taken into account in the aligned subsequence.

2.2.2 Statistical matching

In statistical matching, trajectory similarity is evaluated by analyzing the distribution of low-level features such as spatial location, local direction, or speed. In particular, these methods aim at estimating the probability density functions (*pdf*) of relevant parameters in order to build a statistical model of the target trajectory. Once the model has been defined, the similarity between query and target is calculated on the relevant distributions. A statistical inference process associates the input sequence T to the model M_n that most likely fits the query (see input/output flowchart in Fig. 2.6-(b)). This classification requires a training phase (sketched in Fig. 2.6-(a)), where the parameters of the machine learning algorithm (e.g., Self-Organizing Network, Hidden Markov Models) are learned from a pre-classified set of trajectories or through unsupervised techniques.

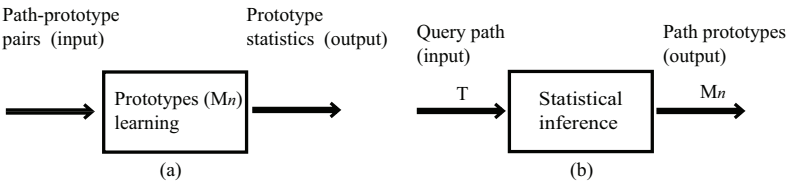


Figure 2.6: Input/Output of statistic matching methods: (a) learning phase, (b) classification.

The methods belonging to this family are proved to be particularly robust against noise and outliers. Two major aspects have to be taken into account in statistical matching: (i) the definition of target models, and (ii) the definition of the matching strategy. As to the first point, models are usually obtained by processing a significant set of paths associated to a given activity and estimating the distribution of the relevant features: spatial location, moving direction, object speed, object acceleration. The second point requires the definition of suitable metrics to evaluate the similarity between the statistics of input and model.

Among statistical methods for trajectory analysis, clustering techniques are very popular. In these techniques, similar paths are grouped together in clusters and compared against a query sample, to determine the class it belongs to, with a maximum similarity criterion. [77] proposes a strategy for trajectory distance measurement and clustering relying on Hidden Markov Models (HMM). The model of the path is build upon a mixture of HMMs and the similarity evaluation is performed by checking the statistical distribution of a given query over each model. More formally, considering two sequences X and Y, their distance is defined relying on their HMM parameterization as follows:

$$D(X,Y) = |L(X;\lambda_X) + L(Y;\lambda_Y) - L(X;\lambda_Y) - L(Y;\lambda_X)| \tag{2.10}$$

where λ_X and λ_Y are the models for X and Y, respectively, and the terms $L(\cdot;\cdot)$ indicates the likelihood of a path with respect to a model. It is worth noticing that when two sequences are similar, the cross terms are generally high. A major advantage of this approach is that the speed can be considered together with geometrical/spatial features of the trajectory. Moreover, the system can cope with the so-called uneven sampling instances (i.e., non-uniform trajectory sampling between consecutive points), which are typical of real-time tracking applications.

Top-down approaches introduce the concept of hierarchical clustering in statistical matching. [55] presents a hierarchical clustering strategy that first identifies global similarities and then refines the analysis of each coarse cluster. An initial wavelet decomposition is employed to tackle noise in the raw trajectory. After smoothing, a set of features is extracted concerned with the so-called trajectory resampling point set (TRPS) and trajectory directional histogram (TDH). TRPS is the result of the path resampling at regular spatial steps (i.e., it encodes specific positions); TDH is a directional histogram

that considers the direction between consecutive track samples (i.e., it encodes the statistical trend of the direction providing a rough path representation). Finally, a two step clustering is carried out: first, TDH information is exploited in a dominant-set clustering algorithm to identify coarse clusters; second, the similarity between two paths i and j is calculated with the Bhattacharyya distance as follows:

$$BD(i,j) = [1 - \sum_{b=0}^{N} \sqrt{TDH_{ib} TDH_{jb}}]^{1/2} \tag{2.11}$$

where TDH_{ib} and TDH_{jb} are the b^{th} elements of the directional histogram for the paths i and j, respectively. Once rough clusters have been identified, TRPS information is used to refine them, and the L_p-norm distance is used as metric.

More recently, [2] proposed an effective unsupervised clustering algorithm using mean-shift to detect coarse clusters, and a merging procedure to group adjacent blobs and eliminate outliers. Even though this method outperforms the work in [55] in the presence of noise, both methods require a resampling phase, thus not ensuring the preservation of the original temporal information. Another interesting work for trajectory clustering in video surveillance can be found in [73], where a system is proposed, able to create and update the trajectory clusters as soon as the samples are acquired by the tracker. The trajectory data are represented as concatenations of raw samples $T = \{t_i\} = \{x_i, y_i\}_{i=1}^{N}$, while each cluster C is represented by a prototype, defined as a stream of raw spatio-temporal locations in conjunction with an additional parameter (σ_i^2) that indicates the local variance of the cluster at time i:

$$C = \{c_i\} = \{x_i, y_i, \sigma_i\}_{i=1}^{M} \tag{2.12}$$

The metric to compare the incoming trajectory T against each detected cluster C is defined as the average normalized distance of every trajectory point (t_i) from the nearest point of the cluster, calculated within a variable-size temporal window w_i centered in i, as follows:

$$D(T,C) = \frac{1}{n} \sum_{i=1}^{N} d(t_i, C) \; ; \quad d(t_i, C) = \min_j \left(\frac{dist(t_i, c_j)}{\sqrt{\sigma_j}} \right) \quad j \in w_i \tag{2.13}$$

where $dist(.,.)$ is the Euclidean distance.

A particular case of matching is when one wants to detect trajectories that do not comply with any model: this case is usually referred to as anomalous trajectory detection. Johnson et al. [41] propose a method for anomalous trajectory identification employing a sequence of feature vectors to represent the spatial location and the velocity of the object at each time instant. The approximation of the statistical distribution of the vectors in the feature space is achieved through a vector quantization. In particular, two concurrent neural networks are developed: initially the sequence of vectors that best represents a target trajectory is identified and then, similar tracks are clustered. According to the

proposed network topology, each output node represents one of the models and it is said to *'win'* if the associated model is the nearest to the feature vector presented as query. Leaky neurons with short-term memory capabilities are employed, in order to model also the temporal nature of the paths. The major drawback of this technique is that it cannot handle sub-trajectories. Another critical issue lies in the vector quantization phase, which provides a *pdf* approximation relying on the point distribution of prototypes. In particular, the number of the prototype vectors and their initial positioning within the feature space has to be manually defined. To cope with these problems, [60] proposes an improved method based on a completely autonomous system to detect anomalous motion. Such method extends [41], providing a learning module that ensures higher accuracy in the clustering phase and allows for an automatic setup of trajectory prototypes, i.e., the representative of the cluster. Each prototype is supposed to have a Gaussian distribution, and the anomaly detection is carried out by checking the fitness of the incoming path over the available models, according to a *Maximum-A-Posteriori* criterion. To improve the reliability of the system in detecting routes that range over wider time intervals a feedback to the neural network is introduced in [87], while Owens et al. in [71] employ a Kohonen self-organizing map [49]. The approach in [38] further improves [71] by introducing a new hierarchical network structure that allows faster learning.

2.2.3 Vector matching

The general idea behind vector-based matching techniques is to extract a symbolic signature of the path in the form of a feature vector, in order to evaluate the similarity between trajectory pairs on the basis of the distance of the relevant signatures [44]. The path signature is calculated in two phases: first, the features are extracted from the raw data; second, quantization and symbolic coding are performed. Since the information is coded at the symbolic level, the vector distance can be effectively evaluated using simple metrics (e.g., Euclidean, Minkovsky, or Hausdorff distances). In Fig. 2.7 the generic flowchart of a vector-based matching is reported. The incoming path is pre-processed in order to bring its representation to a symbolic domain. The symbolic stream is then fed into the comparison routine to match the query with the signatures extracted from database entries.

Since the representation consists of a string of symbols, the comparison between trajectories can be casted to a string matching problem. The most popular metrics used in this context are based on the edit-distance [59], which defines the distance between two sequences as the minimum number of elementary operations required to convert one string into the other. The allowed operations are: *deletion*, *insertion* and *substitution*. Fig. 2.8-(a) reports an example of edit distance calculation between two textual strings, where *non-matching* characters are highlighted in bold. Referring to the alignment string, *pipe* stands for matching symbols, *cross* stands for symbol substitution, and - indicates a symbol insertion. In this example it is easy to see that the final edit distance is 6 since S_1 can be reverted to S_2 by substituting the symbol "A" with "P" and inserting 5 *gaps*. The most common way to calculate the edit distance is through a dynamic programming

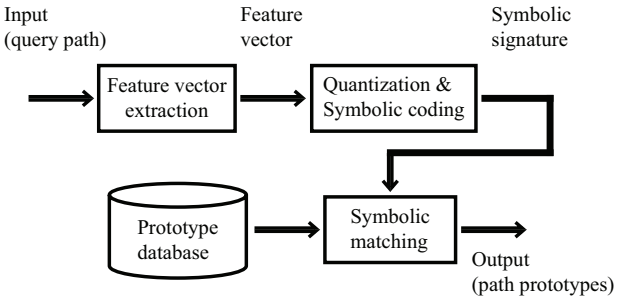


Figure 2.7: Flowchart of vector-based matching methods.

approach. Given two strings of symbols $X = x_0 \dots x_N$ and $Y = y_0 \dots y_M$ from a given alphabet, a matrix ED of $M+1$ rows and $N+1$ columns is initialized and filled starting from the upper left to the lower right corner, running the recursive algorithm reported in Table 2.2-(top). Here, d is the penalty for the gap insertion and it is set to 1 (i.e., $d = 1$), while the symbol substitution cost assumes binary values (i.e., $cost = 0, 1$), depending whether symbols match or not. Once the dynamic programming problem is solved, it returns the final edit-distance $ED(N, M)$.

Among the most interesting techniques that employ this approach, Chen et al. [15] introduce a symbolic trajectory retrieval system called movement pattern string (MPS). Raw samples are processed in order to recover information about the local direction and the distance ratio, i.e., the ratio between the current segment and the whole trajectory. These features are then quantized and each level is associated to a symbol. Accordingly, a similarity metric based on the Levenshtein distance [54] is used for trajectory alignment. Zheng et al. propose in [101] another interesting video retrieval system that compares video clips using a string-based alignment of trajectories. Although the temporal evolution of the path is a fundamental factor characterizing motion, both methods do not take into consideration temporal displacements. This problem is partially solved in [37], which proposes a complete retrieval system aiming at bridging the semantic gap between the users' query and the trajectory representation. The incoming samples are filtered and hierarchically clustered in space and time through spectral clustering. The classes are determined using the minimum cumulative square distance as a metric. Each cluster is then associated with high-level activity models automatically learned from the paths. Finally, the acquired activity models are indexed in a hierarchical tree. A more recent implementation that exploits the edit-distance is presented in [13]. Here, the object trajectories are processed and represented by a chain of symbols indicating the direction and velocity components (sampling time is assumed unitary and constant). The symbolic mapping of the path is then achieved by quantizing each component, in order to reduce the redundancy. Since no resampling or trajectory smoothing is applied, the symbolic

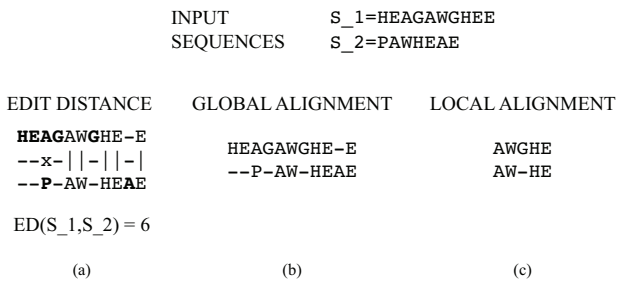


Figure 2.8: Comparison between (a) edit distance, (b) global alignment and (c) local alignment.

mapping may lead to long symbol chains where each sample is encoded as a symbol. In the same class of methods, [75] proposes a comparison strategy inspired by the alignment methods adopted in bioinformatics to match genomic sequences [68][85], also referred to as *inexact* or *approximate matching*.

The adopted metrics rely on modifications of the Levenshtein distance. As for edit-distance, they are still based on the combination of elementary operations such as deletion, insertion and substitution of symbols, but they assign arbitrary scores to each of them, in order to make more flexible the fitness function. This kind of matching algorithms provide several advantages over the implementations described in Section 2.2.1. In particular, they provide a confidence parameter as compared to hard (match/no-match) criteria. Referring to the formula in Table 2.2, the modifications concern the evaluation of substitution *costs*, expressed as real numbers in the range $[0 - 1]$. Since alphabet of symbols is fixed, substitution scores of each symbol pair can be encoded in a set of *substitution matrices*.

The examples in Fig. 2.8-(b) and 2.8-(c) show the results achieved when matching the same pair of genetic sequences using a global and a local alignment, respectively. The global alignment [68] optimizes the score corresponding to the overall matching of the whole sequence, while the local alignment [85] searches for the most similar subsequences. Dynamic problems that have to be solved in order to recover global or local alignments are fairly the same, except for the matrix initialization and some slight differences in the recursive algorithm [68][85]. In [85] the first row and column of the dynamic matrix are padded with zeros and the recursion is made using a *maximum* operator over the same entries as in edit-distance. In [68], the initialization follows the scheme of edit-distance and considers a negative score for gaps. The recursion part is instead the same as for the edit-distance. In Table 2.2 a schematization is reported to underline algorithmic differences among edit-distance, global and local alignment.

Recently, a syntactic representation was introduced in [76], where the path is decomposed in high-level syntactic elements that represent significant substrings of the original trajectory. The structure of the symbols has been arranged according to a set of rules that ensure a flexible representation. Referring to Fig. 2.7, the flow chart of the algorithm

Category	***Cost* value**	**Matrix setup**	**Recursion**
Edit-distance [54]	Binary	$for \quad i = 0 \quad to \quad N$ $\quad ED(i,0) = i \times d$ $for \quad j = 0 \quad to \quad M$ $\quad ED(0,j) = j \times d$ $with \quad d = 1$	$ED(i,j) = min \begin{cases} ED(i,j-1) + d \\ ED(i-1,j) + d \\ ED(i-1,j-1) + cost \end{cases}$
Global alignment [68]	Fuzzy	$for \quad i = 0 \quad to \quad N$ $\quad G(i,0) = i \times d$ $for \quad j = 0 \quad to \quad M$ $\quad G(0,j) = j \times d$ $with \quad d < 0$	$G(i,j) = min \begin{cases} G(i,j-1) + d \\ G(i-1,j) + d \\ G(i-1,j-1) + cost \end{cases}$
Local alignment [85]	Fuzzy	$for \quad i = 0 \quad to \quad N$ $\quad L(i,0) = 0$ $for \quad j = 0 \quad to \quad M$ $\quad L(0,j) = 0$ $with \quad d < 0$	$L(i,j) = max \begin{cases} 0 \\ L(i,j-1) + d \\ L(i-1,j) + d \\ L(i-1,j-1) + cost \end{cases}$

Table 2.2: Different metrics for comparison evaluation: (top) edit-distance, (center) Global alignment, (bottom) Local alignment.

provides a preprocessing phase to detect meaningful spatio-temporal discontinuities. Each element is quantized in direction, velocity, and differential time, and then mapped into a symbol triplet according to a predefined codebook. The matching among trajectories can be therefore expressed in terms of the highest score obtained by aligning the strings of symbols.

In Table 2.3 a comparison among the considered matching categories in terms of representation, strengths and weaknesses is reported.

Other trends can be identified in literature relying on symbolic path representations. A common and widely used approach to model the structure of human behaviors relies on purely probabilistic approaches exploiting, for example, Hidden Markov Models (HMM), related modifications [26], as well as Dynamic Bayesian Networks (DBN) [52]. The gen-

eral idea of these approaches is to extract sets of features from the low-level data and feed them into the probabilistic graphical model used to define the event structure.
As an example, the work in [26] implements a strategy to learn and recognize human activities through a special type of Hidden Markov Models (Switching Hidden Semi-HMM). A two-layer representation is proposed: in the bottom layer a sequence of concatenated Hidden Semi-Markov Model (generalization of HMM with random state duration) defines the atomic activities; the upper layer handles the temporal structure of the activities composing the event by means of a sequence of switching variables. In the same spirit, the authors of [69] proposed a Hierarchical HMM, in order to exploit both the hierarchical structure and the shared semantics contained in the movement trajectories. Moreover, they introduce a Rao-Blackwellised particle filter in the recognition engine in order to cope with real-time recognition constraints. Exploiting such a representation, the method first learns the actions of a subject from an unsegmented training data set, and successively performs an online activity classification, segmentation and anomaly detection. Among the purely probabilistic approaches, an alternative is proposed in [52] where a scalable approach for complex activity recognition is described. The system includes three major modules: a low-level action detector, for the extraction of sub-events from the low-level data, a Dynamic Bayesian Network that encodes the prior knowledge of sub actions ordering constraints, and a Viterbi-based inference algorithm, used to maintain the most likely activity given the DBN status and the output of the low-level detectors.
The main advantage of these methods is the capability of handling the uncertainties generated during the low-level processing. On the other hand, as the event complexity increases, the recognition performance dramatically drops, due to a combination of factors including insufficient training data, semantic ambiguity in the model of the process, or temporal ambiguity in competing hypotheses. Although some methods for unsupervised parameter estimation of the graphical model have been proposed [10], the major problem remains the definition of the network topology, which is usually too complex to be learned from a sparse dataset, and is commonly pre-defined by human operators.

Some other approaches perform the activity recognition in a symbolic domain. In particular, in these works an intermediate step is introduced between the low-level feature extraction and the high-level reasoning. The low-level primitive processing is carried out in different ways (e.g., HMM or similar), while for the high-level behavior modeling a common approach is to adopt the Context-Free Grammar formalism. In [40], for example, the authors propose to split the problem into two parts, using a statistical approach to detect primitives (low-level activities), and a syntactic approach to detect the high-level structures. In the first phase, HMMs are employed to propose candidates for low-level temporal features; these features serve then as input for the Stochastic Context-Free Grammar (SCFG), providing longer-range temporal constraints, disambiguating uncertain low-level detections, and allowing the inclusion of a priori knowledge about the structure of temporal events. In [63] a system is proposed to generate detailed annotations of complex behaviors of humans performing the Towers of Hanoi through a parameterized and manually-defined stochastic grammar, able to identify both single operations as well

as more complex tasks. In [65] the authors also use SCFG to extract high-level behaviors from video sequences, in which multiple subjects can perform different separable activities. An alternative approach is proposed in [43]. Here, the so-called attribute grammars [48] are employed as descriptors for features that are not easily represented by finite symbols. They provide, in other words, a formal way to define attributes for the production rules of a formal grammar. The final goal of the proposed work is to recognize activities and potential signal anomalies. In particular, the proposed framework can handle concurrent behaviors involving multiple entities, as well as uncertainties in semantic conditions on the attributes that are used to express a confidence measure over the recognized events.

A common drawback of the systems relying on formal grammars is in the definition and updating of the production rules. In fact, an exhaustive formalization and structuring of the observable activities a person can perform in everyday life, is in practice not available, since all possible actions cannot be defined a priori. For this reasons, in [31] a computational framework is proposed, able to recognize behaviors in a minimally supervised manner, relying on the assumption that everyday activities can be encoded through their local event subsequences, and assuming that this encoding is sufficient for activity discovery and classification. In this work, the authors introduce the concept of *Motif*, defined as the most frequent subsequences that appeared in the data collection phase that may be associated to relevant atoms to be recognized as behaviors. The activity recognition is then based on the discovery and matching of the *Motif* elements. However, since behaviors are modeled using rigid variable-length event subsequences, the method is sensitive to the noise introduced for example by changing the order of the sub-events. Another major limitation of SCFG based system that prevent their use in real applications is that the parsing strategy can handle only sequential relations between sub-events, with no capability in catching the parallel temporal relations often existing in complex events. Recently, to overcome this issue, the authors of [100] have proposed to extract the terminal symbols of a SCFG from motion trajectories. In particular, the motion trajectories are transformed in a set of basic motion patterns (*primitives*) that are taken as terminals for the formal grammar. Then, a rule induction algorithm based on the Minimum Description Length (MDL) is proposed to automatically derive the spatio-temporal structure of the event from the primitive stream. The complex temporal logic between atomic events is modeled through a combination of SCFG and Allen's temporal logic, while a Multi-Thread Parsing algorithm with Viterbi-like error recovering is developed in order to recognize interesting events in the stream.

Category	Path	Description	Pros & Cons
Dynamic [9] [46] [22] [89]	Raw sample stream, syntactic-symbolic representation.	Stretching in temporal/spatial domain to determine the best alignment. The similarity is the cumulative distance between samples or longest common subsequences. Dynamic programming is used to recover the warping path and the final similarity score.	**Pros:** match series with different lengths. **Cons:** high computational complexity.
Statistic [77] [55] [2] [73] [41] [60] [87] [71] [38]	Vectors of low-level features.	A statistical model is build processing low level features. The similarity is evaluated comparing the current values with the models. Clustering is a popular approach.	**Pros:** different features are considered and modeled separately. **Cons:** large data set required for training; the *pdf* modeling heavily depends on the quality of the training.
Vector-based [76] [44] [15] [101] [37] [13] [89] [48] [40] [100]	Syntactic-symbolic representation.	Extraction of low-level features, coded at symbolic level. The trajectory similarity is measured using metrics such as Euclidean, Minkovsky, Hausdorff distances, or modifications of the edit-distance. Other approaches rely on probabilistic graphical models.	**Pros:** Comparison between string of symbols. **Cons:** Pre-processing is required to map the samples into the symbolic domain. Parameters depend on the scenario.

Table 2.3: Trajectory matching: a comparative analysis.

Chapter 3

Syntactic Matching of Trajectories for Ambient Intelligence Applications

In this chapter a novel approach is proposed for syntactic description and matching of object trajectories in digital video, suitable for classification and recognition purposes. Trajectories are first segmented by detecting the meaningful discontinuities in time and space, and are successively expressed through an ad-hoc syntax. A suitable metric is then proposed, which allows determining the similarity among trajectories, based on the so-called inexact or approximate matching. The metric mimics the algorithms used in bio-informatics to match DNA sequences, and returns a score, which allows identifying the analogies among different trajectories on both global and local basis. The tool can therefore be adopted for the analysis, classification, and learning of motion patterns, in activity detection or behavioral understanding.

3.1 Introduction

The growing interest in ambient-intelligence and the significant reduction in the price of image capture devices and digital signal processing systems, has contributed to the widespread adoption of video technologies in most monitoring and surveillance applications. On the other hand, large and distributed sensing architectures provide human operators with huge amounts of data (mostly real-time video) that quickly overwhelm the ability of the security personnel to analyze and react to events, especially in safety-critical applications. As a matter of fact, most of the available consumer products mainly focus

Part of this Chapter appears in:
N. Piotto, N. Conci, F.G.B. De Natale, "Syntactic Matching of Trajectories for Ambient Intelligence Applications" in IEEE Trans on Multimedia, v. 11, n. 9 (2009), p. 1266-1275.
N. Piotto, N. Conci, F.G.B. De Natale, "Syntactic Matching of Pedestrian Trajectories for Behavioral Analysis". IEEE Int'l Conf. on Multimedia Signal Processing, pp. 877 - 882, Cairns, Queensland, Australia, 08-10 October 2008.

on the recording of video sequences for after-event analysis that are useful as forensic tool, but disregard the primary benefit of surveillance systems as active and real-time prevention instruments. More sophisticated systems attempt to process data in real-time in order to detect significant events that need to be promptly reported to the operator. This is the case of systems for decision support, where the automation of certain procedures allows real-time detection of relevant events. Such events are typically related to changes detected in the monitored area, which can be caused by human actions (e.g., entering/exiting the scene, accessing some specific areas), modifications of the environmental conditions (e.g., objects relocation, objects left unattended, changes in illumination, presence of shimmering lights or fire), or suspect behaviors (e.g., identification of specific movement patterns, interaction with objects in the scene). Most of these events are associated with the presence of moving entities like people and/or other objects in the scene. Nowadays, sophisticated and reliable object trackers can be found in the literature that make it possible to extract an accurate representation of the spatio-temporal trajectory of each object in a video sequence (see [36][96]) also in very complex scenarios. Starting from the acquired trajectory, a common way to detect activities or behaviors consists in translating the trajectories of the moving objects into sets of descriptors, and successively comparing such descriptors with predefined (or learned) models. This approach has been widely used in many application fields such as smart environments (motion is analyzed to understand people presence and behaviors [28][94][67]), content-based video indexing and retrieval [35], gesture and gait analysis [91], and biometry [93].

Starting from a preliminary study proposed in [75], we present in this work a complete representation and matching framework, and provide an in-depth description of the relevant processing techniques and a thorough experimental validation, also in comparison with state-of-art approaches of the same class.

The chapter is structured as follow: Section 3.2 give a qualitative overview of the proposed system, while Section 3.3 gives a more detailed description of the framework architecture. Section 3.4 focuses on the experimental validation on two different sets of trajectories in different indoor scenarios. Concluding remarks are drawn in Section 3.5.

3.2 Overview of the system

The implementation of an effective trajectory similarity metric requires a few preliminary considerations. The extraction of object trajectories from video data is typically imprecise due to environmental noise, segmentation errors, and occlusions: these uncertainties typically produce unreliable tracks containing gaps and misplacements. Moreover, the same spatial trajectory could be associated to different duration, speed and acceleration patterns. A good representation and matching strategy should be able to catch similarities and differences in all these respects, assigning the appropriate weight to each parameter. According to these considerations, the key idea of the matching scheme proposed in this work has been inspired by the alignment procedure adopted in bio-informatics to match genomic sequences [68][85], also referred to as inexact or approximate matching. These

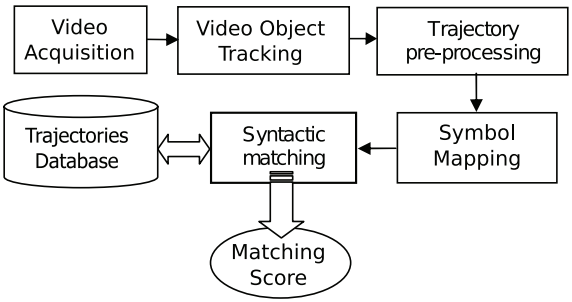


Figure 3.1: Application flowchart.

techniques do not provide a hard matching (i.e., point by point as in DTW), since they rely on modifications of the edit-distance [54].

Accordingly, we propose to segment the track in syntactic elements that represent significant substrings of the original trajectory that are used as basic symbols of a string representation. The structure of the symbols has been arranged according to a set of rules that ensure a flexible representation, as we will discuss in the following sections. The string-based representations are then aligned according the above strategies. An overview of the processing flow is shown in Fig. 3.1: raw trajectories are pre-processed to detect the spatio-temporal discontinuities, thus identifying a reduced set of meaningful trajectory segments. The concatenation of the obtained segments can thus be assumed to be an approximation of the original trajectory. The quantization of each segment in terms of direction, velocity and time, lets mapping each level into a symbol, selected from a pre-defined codebook. Then, the matching between two trajectories can be expressed as the cost of aligning the corresponding strings of symbols. The major advantages of this representation and the matching strategy we propose, can be summarized in two main points:

- reduction in the complexity of representation and matching and capability of considering the invariance to scale, rotation, temporal or spatial shifts;
- temporal and spatial features jointly contribute to the score calculation, thus leading to a more accurate alignment, able to detect similarities on both global and local level.

Additionally, we highlight the capability of building the symbol string *on-the-fly*, thus making it possible to analyze the sequence and to evaluate the matching score in real-time, even if the complete trajectory is not available yet. The nature of the edit-distance turns out to be effective also in tackling the local noise; in fact, the best match is found when coupling close symbols and discarding the outliers, which are handled at the syntactic

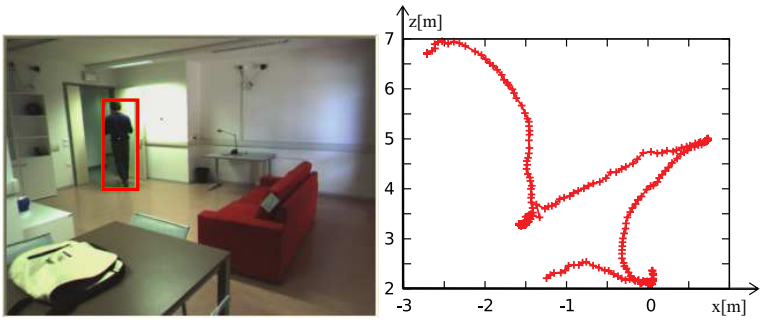


Figure 3.2: Object tracking and top-view trajectory.

level. An outlier in the trajectory may generate a very brief sequence of wrong symbols (1-2) associated to gaps in the alignment process.

3.3 The proposed approach

In this section we describe the proposed trajectory representation and matching algorithm. We would like to point out that video object tracking is beyond the scope of this work. We therefore adopted a state of art methodology. The trajectories we use consist of the projection of the objects *centroid* on the floor, which represent the top-view of the object displacement in the environment. The tracking module we used is based on [19] for the background suppression stage, while the tracking algorithm uses a proximity criterion to detect adjacent blobs across frames based on their color appearance and distance. Since this would result in an inaccurate discrimination of objects in the presence of occlusions, we adopted a stereo camera to derive the depth information, through which it is possible to project the blobs on the ground floor and merge them accordingly. Analogous results can be obtained by using multi-camera systems. Fig. 3.2 shows an example of a moving object detection and the corresponding top-view trajectory ($x - z$ plane), where the coordinate (0,0) refers to the camera position.

3.3.1 Trajectory segmentation and characterization

Starting from the raw trajectory extracted by the tracker, an on-line filtering is applied in order to identify the spatio-temporal discontinuities in the path (trajectory pre-processing in Fig. 2). The input of the pre-processor unit is:

$$T_i = \{x_j^i, z_j^i, t_j\};\ j = 0...N \tag{3.1}$$

where x_j^i and z_j^i determine the top-view position of the i-th tracked object at the time t_j as shown in Fig. 3.2, and N is the number of trajectory samples. To detect sharp velocity discontinuities in the object motion, and in particular stops/re-start events, the coordinates of the object (x_k^i, z_k^i) are evaluated in the time window $[t_k, t_{k+l}]$. If the object position does not change within the selected time interval, $P_k^i = (x_k^i, z_k^i, t_k)$ is marked as a *temporal breakpoint*. Since the *centroid* of the object is subjected to small position variations due to noise, a guard area of radius ρ proportional to the object size is used to check the stop condition [11]. Considering an indoor scene, the characteristics of human motion, and an acquisition rate of 25 fps, in our tests we set the radius ρ in the range $[0.5, 1]$ meters, and a time frame l in the range $[50, 75]$ frames (equivalent to 2-3 sec).

As far as the spatial analysis is concerned, two separate procedures are implemented (Fig. 3.3). The former detects sharp direction variations by analyzing a temporal window of three consecutive samples: the current point $P_k^i(x_k^i, z_k^i, t_k)$ and two previous observations $P_{k-1}^i(x_{k-1}^i, z_{k-1}^i, t_{k-1})$ and $P_{k-2}^i(z_{k-2}^i, y_{k-2}^i, t_{k-2})$. The interpolating lines $r_{k-1}(x)$ and $r_k(x)$ are then calculated, being the lines passing through P_{k-2}^i-P_{k-1}^i and P_{k-1}^i-P_k^i:

$$r_{k-1}(x) = \quad m_{k-1}x + q_{k-1} \tag{3.2}$$

$$r_k(x) = \quad m_k x + q_k \tag{3.3}$$

where m_{k-1} and m_k represent the slope of the lines connecting the two points, and q_{k-1} and q_k are the corresponding offsets.

Then, the angle β_k (Fig. 3.3(a)), is calculated according to Eq. (3.4): if the angle exceeds a predefined threshold β_{th}, P_k^i is marked as a *spatial breakpoint*.

$$\beta_k = tan^{-1} \left| \frac{m_{k-1} - m_k}{1 - m_{k-1} m_k} \right| \tag{3.4}$$

The above criterion (derivative) cannot detect cumulative changes in direction generated by successive small variations, (Fig. 3.3(b)). An integrative criterion has been therefore implemented to calculate the area γ subtended by the trajectory, starting from the last breakpoint up to the current sample:

$$\gamma_{(k-g,k)} = \frac{1}{2} \sum_{q=k-g}^{k} [(h_q^i + h_{q+1}^i)(\left| R_{q+1}^i - R_q^i \right|)] \tag{3.5}$$

In Eq. (3.5), h_q^i is the Euclidean distance between the current sample P_q^i and the line r that connects P_{k-g}^i (last breakpoint) with P_k^i(current sample); R_q^i is the projection of the sample P_q^i on r. Again, if the resulting area $\gamma_{(k-g,k)}$ exceeds a given threshold (γ_{th}), the sample P_k^i is marked as a spatial breakpoint (filled dots in Fig. 3.3(b)). The choice of the two thresholds β_{th} and γ_{th} will be analyzed in the Section 3.4 based on the evaluation of the accuracy of the system on a set of training sequences. As a rule of thumb, we can state that β_{th} determines the reactivity to local variations, while γ_{th} affects the sensitivity to long-term deviations. Since pedestrians tend in general to walk along smooth trajectories, the most important threshold is usually γ_{th}.

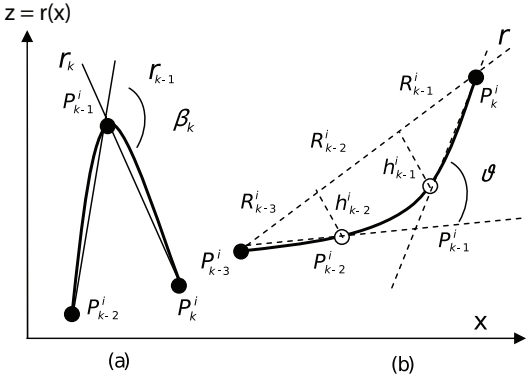


Figure 3.3: (a) Local variation angle and (b) cumulative variations leading to a significant direction change.

3.3.2 Key points symbolic mapping

The above described spatio-temporal analysis identifies a chain of breakpoints B^i_m (being $m = 1...M$ the number of detected breakpoints for the i-th object), along the original trajectory. Each pair of successive breakpoints identifies a rectilinear segment $S^i_m = B^i_m \leftrightarrow B^i_{m+1}$ that approximates a portion of the original path. Accordingly, the approximated trajectory can be represented by an appropriate description of the segment chain $\{S^i_m\} : m = 1 \dots M-1$. In this representation, each segment S^i_m is characterized by its orientation θ^i_m, its velocity v^i_m, and the relevant temporal interval Δt_m.

The above parameters are determined as follows: direction and duration are calculated with respect to the previous segment (Eq. (3.6) and Eq. (3.7)), while the speed is computed as the length of the segment divided by its duration (Eq. (3.8)).

$$\theta^i_m = \beta^i_m \tag{3.6}$$

$$\Delta t_m = t_m - t_{m-1} \tag{3.7}$$

$$v^i_m = \frac{d(B^i_m, B^i_{m-1})}{\Delta t_m} \tag{3.8}$$

β^i_m is calculated according to Eq. (3.4) in B^i_m; t_{m-1} and t_m are the absolute time references corresponding to B^i_{m-1} and B^i_m, respectively, and $d(a, b)$ is the Euclidean distance. The approximated trajectory description for the i-th object is then given by Eq. (3.9):

$$T^*_i = \{\theta^i_m, v^i_m, \Delta t_m\};\ m = 1...M \tag{3.9}$$

Table 3.1: Quantization levels.

Variable	Range	Quantization Levels
θ_m	$[-180^\circ \ +180^\circ]$	θ_0 ... θ_{11}
v_m	$[0 \ v_{max}]$	v_0 ... v_3
Δt_m	$]0 \ \infty]$	τ_0 ... τ_3

This representation is inherently invariant to rotation and translation and fulfills several requirements. In fact, only the coordinates and orientation of the first segment refer to an absolute positioning, then this information can be easily discarded to achieve invariance to translation and orientation. Similarly, if the temporal discontinuities of the trajectory are not relevant, stops can be removed by simply dropping samples with null speed. As it can be noticed, these features allow performing different types of matching such as, for instance, identifying trajectories with similar geometry but different speed, or detecting similar behaviors (e.g., *zig-zag* moving patterns) in different spatial locations.

The last step to achieve a complete syntactic representation consists in mapping each segment into symbols. This can be obtained by properly quantizing the parameters $\{\theta_m^i, v_m^i, \Delta t_m\}$, in order to make the symbols enumerable. Since the application we address in our tests is people tracking in indoor environments, owing to the limited speed and typical movements of the target, we quantized the direction θ_m in 12 non-uniform levels, while speed and time components have been quantized in 4 levels (see Table 3.1). The choice of the values associated to each level is discussed in Section 3.4. Fig. 3.4 shows a time-space diagram, in which the original and the reconstructed trajectories are plotted. Markers represent the breakpoints detected by the segmentation algorithm.

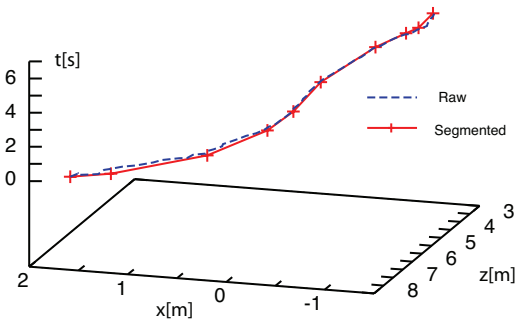


Figure 3.4: Path segmentation.

3.3.3 Trajectory alignment and matching

The goal of the syntactic matching engine is to find the best alignment among strings that represent different trajectories, and to calculate the corresponding similarity score. Depending on the application, the trajectories to be matched can be part of a pre-defined database (e.g., knowledge-based behavioral analysis), queries sketched by the user (e.g., content-based video retrieval), or actions automatically learned by the system (e.g., behavior classification and automatic detection of anomalous events). Thanks to the syntactic representation of the object paths, the proposed matching procedure is very fast and efficient, similarly to what text processors do in detecting and correcting errors.

The alignment algorithm we present in this work relies on the so-called edit-distance. Basically, the difference between two strings of symbols is measured as the minimum-cost set of elementary actions (i.e., insertion, deletion and substitution) required to transform one sequence into the other. To achieve this goal, a cost (weight) is associated to each operation according to its relevance in the transformation. This makes it possible to assign different weights to different types of actions (e.g., a substitution can have higher impact than a deletion) and to the symbols involved in the action. The association of an operation to a weight is achieved by using a substitution matrix. The total cost of the transformation is the sum of the single weights. The edit-distance has been chosen against other metrics because of its flexibility, and in particular the possibility of easily adapting the matching to different application requirements. In fact, any kind of matching rule can be implemented by properly adjusting the substitution matrix. Another significant advantage of the selected approach is the capability of operating *on-the-fly*, i.e., processing the samples as soon as they are acquired. As to the specific implementation, we introduce a modified version of the theory presented in [68] and [85]. The concept we apply is similar, in the sense that we assign a score to each symbols pair: the higher the score, the better the matching (equal symbols receive maximum score). The best global matching is the one that maximizes the global score. The identification of the most suitable substitution matrix is *application-dependent*: for instance, the entries in the matrix employed in [68] (DNA sequences alignment) are defined on the basis of the biological similarity among amino-acids (A, T, G, C). The alignment procedure adopted in this work provides the simultaneous matching in three different domains, namely space, speed, and time. Since these three parameters have different impact on the matching, the score is calculated starting from three separate substitution matrices, one for each parameter, and then adding the single scores to achieve the overall substitution cost.

As far as the substitution matrices are concerned, we provide here a simple example that can explain the rationale behind the choice of the entries. Let us assume that a trajectory is described through five symbols: move forward (f), slight turn right (st_r), slight turn left (st_l), sharp turn right (St_r), and sharp turn left (St_l). While performing the matching, it is likely that the cost of substituting f with st_r is larger than the cost of substituting f with St_r, since a slight turn is more similar to a straight path than to a sharp turn. The cost of substitution pair St_r-St_l should be clearly even larger. Time and speed parameters behave a little differently, since they have theoretically no upper

bound. In this case, we assume that the score drops down to zero, once exceeding a given distance, meaning that the two symbols are no more correlated. It has to be pointed out that the tuning of the matrices has an important semantic impact: for instance, if speed should be ignored in the matching, the corresponding matrix will have the same value for all entries, thus becoming irrelevant in the comparison phase. Section 3.4 provides a detailed explanation of our experimental setup, together with actual examples of matrix configurations. Given the matrices, the alignment procedure can be described as follows. Let us take two generic strings of symbols, A and B, of length N_A and N_B, respectively. A two-dimensional array, commonly called F matrix, is created with dimension $N_A \times N_B$. The F matrix is iteratively filled by assigning to each entry $F(i, j)$ the score of the optimum alignment between the symbols of string A and string B, according to the Algorithm 1. Here, $S(A(i), B(j))$ represents the function that calculates the substitution score between the symbols $A(i)$ and $B(j)$.

Algorithm 1 Dynamic matrix filling for Global alignment.

Require: N_a, N_b, A, B, w, $S(A(i), B(j))$ with $i = 0..N_a$ and $j = 0..N_b$
Ensure: Best global alignment $F(N_a, N_b)$
  F Matrix initialization.
  $F(0, 0) = 0$,
  for $i = 0$ to N_a **do**
    $F(i, 0) = w * i$
  end for
  for $j = 0$ to N_b **do**
    $F(0, j) = w * j$
  end for
  for $i = 0$ to N_a **do**
    for $j = 0$ to N_b **do**
      F(i,j) = max(F(i-1,j-1)+S(A(i),B(j)), F(i,j-1)+w, F(i-1,j)+w)
    end for
  end for

At the end of the process, the last entry $F(N_A, N_B)$ returns the best global alignment between the two strings. Successively, a trace-back procedure allows retrieving the sequence of elementary operations that lead to the specific alignment (i.e., the best path to go back to $F(0, 0)$).

The calculation of the F matrix is the most computationally expensive phase: dynamic programming techniques have been introduced to reach a good trade-off between space and time complexities. In our solution the running time and the used memory is $O(nm)$, even though more efficient implementations can lead to significant reductions in complexity.

The adopted cost function is expressed in Eq. (3.10) and Eq. (3.11):

$$\delta_k = F_k(i,j); \qquad k = \theta, t, v \tag{3.10}$$

$$\Psi(i,j) = \frac{\alpha_\theta \delta_\theta}{Q_\theta} + \frac{\alpha_t \delta_t}{Q_t} + \frac{\alpha_v \delta_v}{Q_v}; \qquad \alpha_\theta + \alpha_t + \alpha_v = 1 \tag{3.11}$$

where α_θ, α_t, and α_v are the feature weighting coefficients, Q_θ, Q_t, Q_v, are normalization factors corresponding to the maximum score associated to each feature, and δ_θ, δ_t and δ_v are global scores for each single feature, obtained by applying the substitution matrices. As the alignment algorithm proceeds, the temporary score is normalized over the whole number of elementary operations required to align the substrings. It is to be noticed that Eq. (3.10) and Eq. (3.11) provide the global alignment without considering the initial rotation and translation. In order to retrieve the absolute direction and position and use them as inputs to the alignment algorithms, two additional parameters Ψ_{pos} and Ψ_{dir} need to be considered:

$$\Psi_{pos} = \frac{T}{Q_{pos}} \tag{3.12}$$

$$\Psi_{dir} = \frac{R}{Q_{dir}} \tag{3.13}$$

$$T = F_{pos}(1,1) \tag{3.14}$$

$$R = F_{dir}(1,1) \tag{3.15}$$

Q_{pos} and Q_{dir} represent again normalization factors (the maximum score for initial direction and translation alignment, respectively), while T and R correspond to the score for initial position and direction variations, respectively. T and R are calculated using appropriate substitution matrices that specifically match the first point of the trajectory.

Finally, the final spatio-temporal matching is achieved by combining Eq. (3.12), (3.13), (3.14), (3.15) in a weighted sum, as in Eq. (3.16).

$$\begin{aligned} \Psi_{global}(N_a, N_b) = \psi\Psi(N_a, N_b) + \\ \psi_{pos}\Psi_{pos} + \\ \psi_{dir}\Psi_{dir} \end{aligned} \tag{3.16}$$

where again $\psi + \psi_{pos} + \psi_{dir} = 1$.

According to the application requirements, the roto-translation parameters can be appropriately weighted. In the specific case where $\psi_{pos} = \psi_{dir} = 0$, only the general shape of the trajectory is considered, no matter its absolute positioning and orientation.

As far as the score function $S(A, B)$ is concerned, we have imposed a non-linear distribution. The score evaluation is performed using the recursive function reported in Eq. (3.17) that calculates the score between two symbols $A(i)$ and $B(j)$, where i and j are the symbol quantization levels. Through this representation it is possible to fill the matrix entries for each feature. As expected, the highest values are along the main

(a) (b)

Figure 3.5: Snapshots of the environments used for validation.

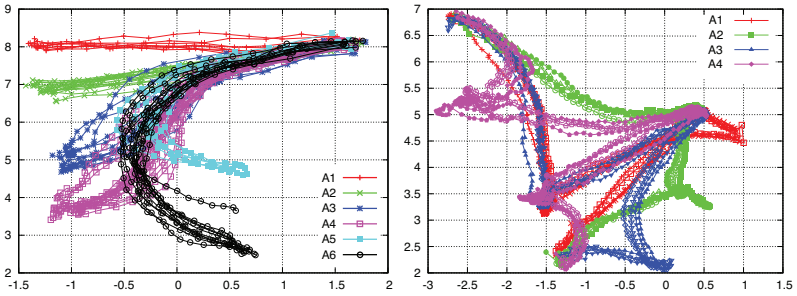
diagonal, gradually decreasing as soon as the distance between symbols increases, in a cyclic fashion.

$$S(A(i), B(j)) = S(i, j) = S(i, j-1) + |i - j| \tag{3.17}$$

The above weight assignment produces symmetric matrices, so that the cost of inverting a symbol pair is equivalent in the two directions.

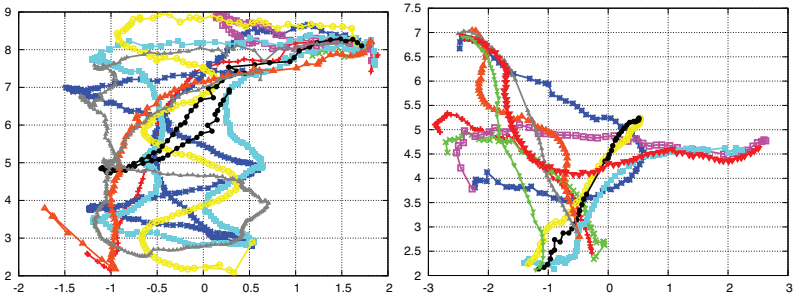
3.4 Experimental results

The proposed strategy has been tested in an indoor environment, using a standard PC connected to a stereo camera. The tests concerned human activity monitoring. The acquired trajectories refer to a 2D top-view of the person motion as detected by the tracker, i.e., the location of the person with respect to the floor of the observed room. To validate the proposed method we adopted two different data sets. The first data set (*MMlab*) refers to relatively simple trajectories where the starting point is common for all sequences. It is composed by 112 tracks divided in 6 different actions and including 35 anomalous paths. The environment used for testing is shown in Fig. 3.5(a) and the set of trajectories is reported in Fig. 3.6(a). A second data set (*Application Lab*) refers to more complex trajectories acquired in an experimental smart environment, dedicated to develop technologies for assisted living. The laboratory is fully equipped with furniture, in order to simulate real moving patterns corresponding to typical activities (e.g., move from the sofa to the kitchen, bring an object and take it back to the sofa). In this case the set includes 100 trajectories grouped into 4 different clusters (see Fig. 3.5(b) and Fig. 3.6(b)) and including 42 anomalous paths. In both figures the coordinates (0,0) refer to the camera position and the coordinates of the points always refer to the $x - z$ plane, the top-view of the room. The two sets of anomalous paths are shown in Fig. 3.7.



(a) (b)

Figure 3.6: (a) Known actions for MMLab and (b) Application Lab data sets.



(a) (b)

Figure 3.7: (a) Anomalous paths for MMLab and (b) Application Lab data sets.

Experiments required the proper setup of the parameters, and in particular the thresholds for trajectory segmentation. As anticipated in Section 3.3.1, spatial and temporal thresholds are chosen in order to maximize the accuracy of the classification of activities. To this purpose, a prototype is defined for each class as the cluster medoid (i.e., the path with average minimum distance from all the others in the same cluster) and the patterns are classified according to the minimum distance prototype.

In the application domain we address, it can be observed that humans tend to perform smooth trajectories instead of abrupt sharp turns within small time windows. Therefore, we have studied the performance of the segmentation scheme by varying γ_{th} for a fixed β_{th} = 30 degrees and selected the value, which returned the highest classification accuracy.

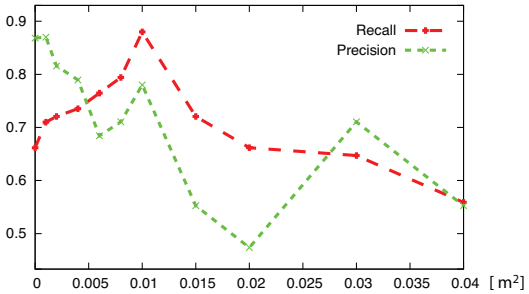
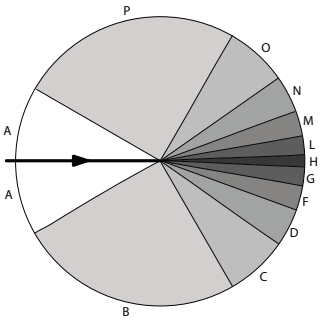


Figure 3.8: Performance *vs.* γ_{th} variation

To this purpose we employed the first data set. Results are plotted in Fig. 3.8 and report the accuracy in terms of recall and precision at different values of γ_{th}, the former corresponding to the average true positive rate, while the latter being TP/(TP+FP), where TP and FP are the numbers of true and false positives, respectively. In our scenarios, in order to discriminate among different actions, the resulting threshold is very small and it basically imposes to fragment the trajectory in segments of around half meter. Such a fine segmentation is due to the specific geometry of the environment, which does not impose any constraint in movement (unlike for example in vehicular applications, where cars move along specific directions), and the detection of specific actions must be carried out with a finer granularity.



range [degree]	Associated symbol
150: -150	A
-150: -60	B
-60 : -35	C
-35 : -20	D
-20 : -10	F
-10 : -2.5	G
-2.5 : 2.5	H
2.5 : 10	L
10 : 20	M
20 : 35	N
35 : 60	O
60 : 150	P

Figure 3.9: (a) Quantization levels for direction and (b) the corresponding symbols.

	A	B	C	D	F	G	H	L	M	N	O	P
A	12,	11,	9,	5,	0,	0,	0,	0,	0,	5,	9,	11
B	11,	12,	11,	9,	5,	0,	0,	0,	0,	0,	5,	9
C	9,	11,	12,	11,	9,	5,	0,	0,	0,	0,	0,	5
D	5,	9,	11,	12,	11,	9,	5,	0,	0,	0,	0,	0
F	0,	5,	9,	11,	12,	11,	9,	5,	0,	0,	0,	0
G	0,	0,	5,	9,	11,	12,	11,	9,	5,	0,	0,	0
H	0,	0,	0,	5,	9,	11,	12,	11,	9,	5,	0,	0
L	0,	0,	0,	0,	5,	9,	11,	12,	11,	9,	5,	0
M	0,	0,	0,	0,	0,	5,	9,	11,	12,	11,	9,	5
N	5,	0,	0,	0,	0,	0,	5,	9,	11,	12,	11,	9
O	9,	5,	0,	0,	0,	0,	0,	5,	9,	11,	12,	11
P	11,	9,	5,	0,	0,	0,	0,	0,	5,	9,	11,	12

(a)

	Q	R	S	T
Q	12,	7,	2,	0
R	7,	12,	7,	2
S	2,	7,	12,	7
T	0,	2,	7,	12

(b)

	W	X	Y	Z
W	12,	7,	2,	0
X	7,	12,	7,	2
Y	2,	7,	12,	7
Z	0,	2,	7,	12

(c)

Figure 3.10: (a) Direction, (b) speed, and (c) time substitution matrices.

The correct setup of the parameters for spatial segmentation is clearly the most significant, since they refer to the geometrical displacement of the object in the scene; the choice of the temporal thresholds is instead more arbitrary. In our experiments an object is considered stopped if its *centroid* remains within the same guard area (of radius $\rho = 0.3m$), for at least 2 seconds.

To determine the symbols, and referring to the direction, we adopted a non-uniform quantization for a better description of small direction variations. This is a reasonable assumption, considering that in normal walking, sharp direction changes are more unlikely. Fig. 3.9 sketches the adopted quantization scheme where the bold arrow refers to the incoming direction. The deviation with respect to the outgoing sector is depicted with increasing gray levels. Each level is then associated to the corresponding symbol (see table in Fig. 3.9 (b)).

Table 3.2: Matching scores in different spatio-temporal configurations.

Trajectories	Space	Space-Speed	Spatio-Temp.
Reference vs T1	0.88	0.76	0.75
Reference vs T2	0.88	0.57	0.54
Reference vs T3	0.55	0.36	0.4

Table 3.3: Weights associated to different matching schemes.

	α_θ	α_v	α_t
Space	1	0	0
Space-speed	0.5	0.5	0
Spatio-temp.	0.33	0.33	0.33

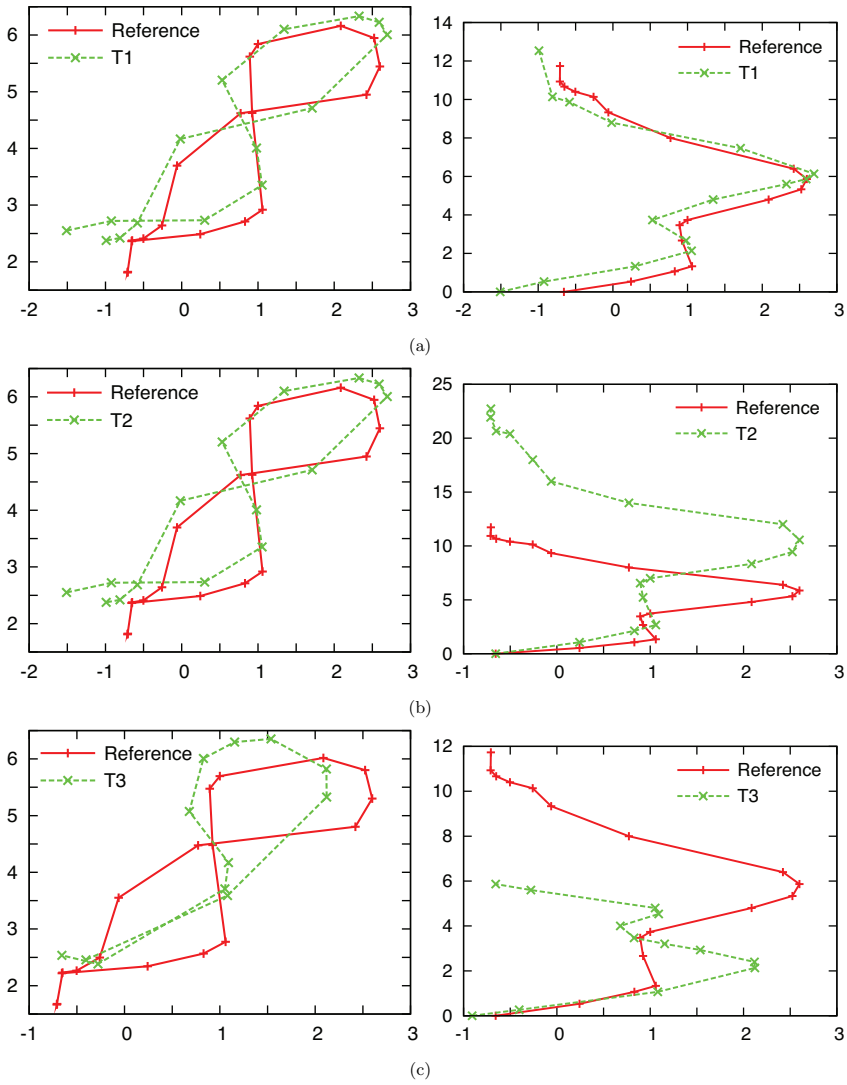


Figure 3.11: Sample trajectories with different space (left) and time (right) characteristics: (a) similar paths in both spatial and temporal domain; (b) remarkable differences in time; (c) remarkable differences in space and time.

As far as speed is concerned, and referring to Table 3.1, one level has been reserved for the null velocity (stop), while the last level covers the range from v_3 up to v_{max}, which specifies the maximum possible velocity. Since no maximum value can be foreseen, the level is used to discriminate velocities that exceed 5 km/h.

A similar approach is applied to time, where the maximum level is set for temporal intervals exceeding the stop threshold (2 seconds in our case). The selected symbols for speed and time are {Q, R, S, T} and {W, X, Y, Z}, respectively. According to the function $S(\cdot)$ of Eq. (3.17), the resulting substitution matrices are shown in Fig. 3.10. Again, different scenarios such as vehicular applications would require appropriate settings (for instance much higher velocity threshold). It is to be pointed out, however, that the selection of parameters used for human targets proved to be very robust across different tests in different environmental situations as shown in the following results.

We present hereafter a selection of results to demonstrate the capability of the system to identify similarities among trajectories, with different spatio-temporal configurations enabling/disabling the invariance to rotation and translation. In the first set of tests we compare two paths that are: (i) similar in space but denoting minor differences in time (T1, Fig. 3.11-a); (ii) similar in space but with remarkable differences in time (T2, Fig. 3.11-b); (iii) significantly different in both space and time (T3, Fig. 3.11-c). In particular, comparing T1 and T2 it is possible to notice that the execution of the same path at different speed results in a compression of the graph in the time axis.

The results obtained by applying Eq. (3.10) and (3.11) are reported in Table 3.2 in terms of normalized score in space, space-speed and full space-temporal domains, respectively. The scores are obtained by setting different weights in Eq. (3.11) as shown in Table 3.3. The final values are normalized with respect to the total number of elementary operations required to transform one symbolic string into the other.

The second set of tests aims at demonstrating the effectiveness of the described method while considering global rotation and/or translation. To this purpose, a random path (purple) has been selected as reference and compared with equally shaped paths that differ in: (i) initial direction (Fig. 3.12-a), and (ii) initial position (Fig. 3.12-b). Numerical scores are reported in Table 3.4 and Table 3.5.

In this last part of the section, we present the results achieved by processing the two data sets with the objective of detecting anomalies. The setup of the experiments reflects the same parameters configuration adopted to derive the best segmentation thresholds, as explained in the first part of this section. In this case, after finding the best match with the available classes, the matching score is evaluated. If it exceeds a given threshold (set to 70% in our tests) the trajectory is assumed to belong to the cluster; otherwise the path is tagged as anomalous.

We compared the performances of our method with two different state-of-art algorithms. The first is the one in [89] and refers to a common metric for sequence comparisons using the Longest Common Sub-Sequence. The second method is the one in [13], due to the fact that it shares some common principles with our work. It is to be noted that our technique does not require a uniform sampling of the points as required by the

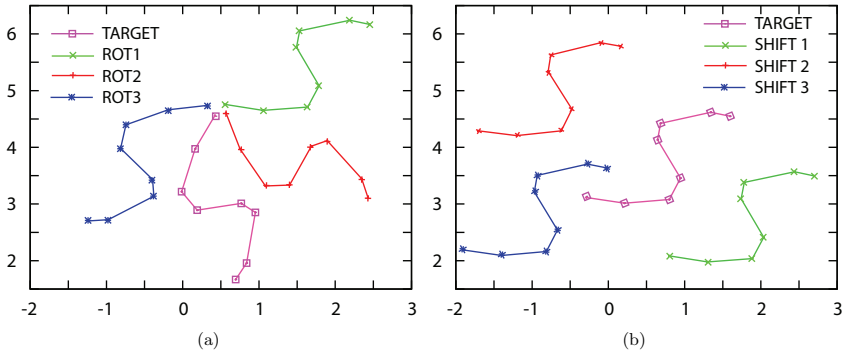


Figure 3.12: (a) Copies of same paths with different initial directions, and (b) different locations.

other two methods. Furthermore, it exploits the temporal information as a critical data for trajectory segmentation and matching.

In Table 3.6 and Table 3.7 we report the numerical results obtained from the two data sets and applying the three methods. Again, the evaluation parameters reported are Recall and Precision. Additionally, being the anomaly detection a binary classification problem, we show also the Accuracy, defined as (TP + TN)/(TP + TN + FP + FN). As it can be observed, the proposed method performs in general better than the competitors. In particular, the improvements in terms of accuracy are of 8% and 5% with respect to

Table 3.4: Matching scores for rotation invariance.

	Ψ_{dir}	Ψ_{global}
TARGET vs ROT1	0.91	0.96
TARGET vs ROT2	0	0.5
TARGET vs ROT3	0.41	0

Table 3.5: Matching scores for translation invariance.

	Ψ_{pos}	Ψ_{global}
TARGET vs SHIFT1	0.2	0.6
TARGET vs SHIFT2	0.2	0.6
TARGET vs SHIFT3	0.8	0.9

Table 3.6: Performance comparisons for paths in the MMLab data set.

	method in [89]	method in [13]	proposed method
Recall	0.97	0.92	1
Precision	0.69	0.72	0.78
Accuracy	0.79	0.82	0.87

Table 3.7: Performance comparisons for paths in the Application Lab data set.

	method in [89]	method in [13]	proposed method
Recall	0.94	0.91	0.97
Precision	0.67	0.69	0.83
Accuracy	0.75	0.76	0.88

[89] and [13] in the *MMLab* data set. For more complex trajectories (*Application Lab*), the improvements are more consistent: 13% and 12%.

3.5 Conclusions

In this chapter we presented a new approach to perform syntactic matching of trajectories, as a basis for applications such as activity detection, event analysis, or content-based video retrieval. Starting from the acquisition of the path in the $x - z$ plane, the meaningful spatio-temporal discontinuities are identified. Trajectory segments are then quantized and converted into symbols corresponding to the variations in terms of direction, speed, and time, with respect to the previous sample. The resulting syntax is used to compare different trajectories, adopting a bio-inspired approximate matching algorithm based on the so-called *edit-distance*. Experimental validation concerned the analysis of human walk patterns in indoor environments. Results confirm the good performance of the method in dealing with different data sets, and its flexibility in managing the invariance to translation and rotation. Moreover, the comparison with state of art approaches of the same class showed a significant performance enhancement.

Chapter 4

Hierarchical Matching of 3D Trajectories for Surveillance Applications

In this chapter we propose a string-based approach to effectively represent trajectories in the 3D space. The strategy is coupled with a syntactical matching algorithm that allows evaluating the similarity of the retrieved data with pre-stored templates. The hierarchical application of the algorithm on the spatial and temporal components helps detecting anomalous trajectories, and has proven to be robust in automatically learning new classes of paths. We present the results achieved by performing a number of tests in an indoor lab used as a testbed for assisted living applications. The algorithm can discriminate among different classes of trajectories, and can recognize actions and detect anomalies within the same class.

4.1 Introduction

The evolution of computer vision technologies has facilitated the process of detection and classification of moving objects in the observed scene thanks also to multi-camera and stereo vision. As a consequence, the extraction and analysis of the trajectory in the 3D space is a natural development.
The representation of the trajectory is significant because it describes the movement of an object within the observed space and it can be considered as one of the most critical features that helps discriminating between *normal* or *anomalous* activities. To do so, an accurate description of the acquired trajectory is mandatory and the extracted features need to be compared with pre-learned or predefined templates for classification purposes.

Part of this Chapter appears in:
N. Piotto, F.G.B. De Natale, N. Conci, "Hierarchical Matching of 3D Pedestrian Trajectories for Surveillance Applications". IEEE Int'l Conf. on Advanced Video and Signal Based Surveillance, pp. 146-151, Genova, 2-4 September 2009.

Together with the spatial displacement of the object, also additional features like speed and acceleration can help improving the quality of the description. Therefore, the need for methods capable of evaluating the similarity between n-dimensional curves is increasingly assuming a key-role. The problem is well known in literature and matching strategies for n-dimensional spaces have already been proposed, especially in the field of activity detection and recognition, surveillance, and monitoring. Surveillance is not the only research area that is concerned with this issue, but it involved also others disciplines such as biological structure matching or gesture recognition for HCI.

In this work we focus on the trajectory representation and matching for surveillance applications in indoor environments. The final goal is to derive anomalies in common actions through a 3D analysis enriched by the information of additional features, like speed and time. A preliminary version about trajectory analysis in the 2D domain has been presented in [75] and [76]. Here we enhance the system by introducing the third dimension to improve the trajectory classification in a hierarchical fashion, making it possible to deploy a tool for automatic learning of new movement patterns.

The chapter is structured as follows. Section 4.2 describes the proposed method for trajectory representation, and Section 4.3 details the matching procedure we have implemented for the 3D analysis. Section 4.4 presents the achieved results, and concluding remarks are given in Section 4.5.

4.2 3D trajectory representation and reconstruction

The trajectories to be compared may differ in several aspects like length, duration, and speed, to name a few. A good representation should be able to catch similarities and differences with respect to all variables, assigning an appropriate weight to each parameter. Moreover, the scheme has to be capable of dealing with the typical noise that affects the n-dimensional data, thus providing reliable similarity results even in presence of outliers.

According to these considerations, the key idea of the representation and matching scheme proposed in this work has been inspired by bio-informatics algorithms used to match genomic sequences [68] [85], also referred to as approximate matching.

In our proposal, as described in Chapter 3, the track is decomposed in high-level syntactic elements that represent significant portions of the original trajectory. The formal definition of the elements has been formulated according to a set of rules that ensure a flexible representation. The processing flow of the algorithm is shown in Fig. 4.1: after acquisition and tracking, the retrieved trajectories are pre-processed to detect the meaningful spatio-temporal discontinuities, called breakpoints, thus identifying a reduced set of trajectory samples. The concatenation of the obtained breakpoints turns out to be an approximation of the trajectory, where each breakpoint is characterized by four features:

- direction in $x - z$ plane;

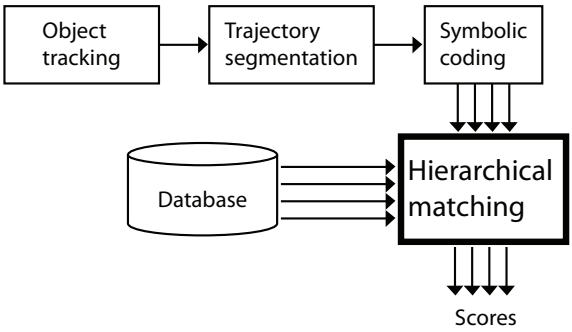


Figure 4.1: Flowchart of the proposed system.

- height y from the ground;
- velocity;
- temporal offset with respect to the last breakpoint.

Considering the representation introduced in Chapter 3 for 2D trajectory, an additional feature (i.e., the height y from the ground) for each anchor point has been introduced in order to describe the 3D position of the sample. Each parameter is quantized in a specific number of levels according to the accuracy required by the application, and it is mapped into a symbol. The major improvement over the state-of-art approaches is therefore represented by the capability of jointly processing temporal and spatial features to improve the accuracy of the matching. At the same time, it is worth noting that the symbol string can be obtained in real-time, even if the complete path is not available yet.

The segmentation strategy we have adopted in this work is an evolution of the algorithm proposed in [75][76], and the main points of innovation can be summarized in (i) the substantial improvements of the features quantization to better represent pedestrian trajectories and (ii) the introduction of the height y to achieve a full 3D representation.

Starting from the raw trajectory returned by the video object tracker, the proposed approach applies an *on-line filtering* in order to identify the spatio-temporal discontinuities in the track. The input of the segmentation strategy refers to the object centroid in the 3D space and according to the features listed above, the raw trajectory T is characterized by a vector of 4 elements, like in Eq. (4.1), where the single elements are evaluated for each i-th object at each j-th acquired sample. N is the total number of samples for T.

$$T_i = \{x_j^i, y_j^i, z_j^i, t_j\};\ j = 0...N \tag{4.1}$$

At each trajectory sample $P_j^i = (x_j^i, y_j^i, z_j^i, t_j)$ the procedures described in [75][76] are evaluated to determine stops and direction variations according to temporal and spatial

constraints. At the end of the segmentation process, the obtained sequence of breakpoints constitutes the string-based representation. The concatenation of these samples reproduces an approximated version of the trajectory. Given the chain of breakpoints B_m^i (being $m = 1...M$ the number of detected breakpoints for the i-th object), each pair of successive breakpoints identifies a rectilinear segment $S_m^i = B_m^i \leftrightarrow B_{m+1}^i$ that approximates the corresponding piece of the original path. Accordingly, the approximated trajectory can be represented (and reconstructed) by the concatenation of the segment chain $\{S_m^i\} : m = 1, , M-1$.

In this representation each segment S_m^i is characterized by its orientation θ_m^i, the height of the centroid y_m^i, its velocity v_m^i, and the temporal interval Δt_m. These parameters are determined exactly following the procedures in Chapter 3.

The approximated trajectory description for the i-th object is then given by Eq. (4.2):

$$T_i^* = \{\theta_m^i, y_m^i, v_m^i, \Delta t_m\};\ m = 1...M \tag{4.2}$$

The final step to achieve a syntactic representation consists in encoding each segment with a unique symbol. Since the nature of the parameters $(\theta_m, y_m, v_m, \Delta t_m)$ is continuous, it is necessary to quantize each feature to make the symbols enumerable. The number of quantization levels and the quantization criteria for each feature must be defined according to the expected track evolution: since the target application in our tests is tracking people in indoor environments, owing to the limited speed and typical behaviors of the target, we have quantized the orientation of the segments in the $x-z$ ground plane in 12 uniform levels, the height y of the centroid in 5 uniform levels and v and Δt in 4 non-uniform levels. A more detailed description of these settings is shown in Table 4.1. With respects to the speed variation range reported in Table 4.1, one level has been allocated to the null velocity (stop), while the upper level covers the range from v_3 up to v_{max}, which specifies the maximum possible velocity. A similar approach is applied to time, where the maximum level is set to the threshold used to detect stops (see [75][76] for details). The quantization of the 3D position of the object is uniformly discretized along y at regular intervals of 30cm, while the direction in $x-z$ has been quantized considering a new level every 30°. The choice of the thresholds has been extensively discussed in our previous work.

Table 4.1: Quantization levels.

Variable	Range	Quantization Levels
θ_m	$[-180° \ +180°]$	θ_0 ... θ_{11}
y_m	$[0\ 180cm]$	y_0 ... y_5
v_m	$[0\ v_{max}]$	v_0 ... v_3
Δt_m	$]0\ \infty]$	τ_0 ... τ_3

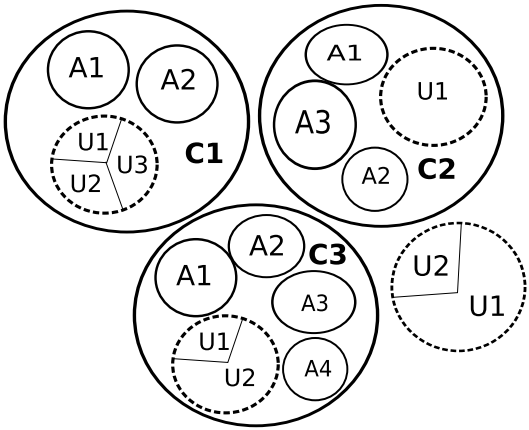


Figure 4.2: Actions tree.

4.3 Syntactic hierarchical matching in 3D space

The major improvement proposed in this work with respect to the 2D matching in [75][76], refers to the hierarchical matching that allows for the automatic learning of new moving patterns. In practice, the 2D top-view allows discriminating among different trajectory classes on the basis of their displacement in the room, while the additional information of height, speed and time is significant to determine different actions within the same class. The choice of a hierarchical approach is mandatory in our case, since the information about speed and time could be misleading, yielding to the wrong association of moving patterns that don't have any spatial similarity. This application turns out to be very effective in applications like assisted living and elderly monitoring, since the same spatial path performed in different ways, could be significant in determining abnormal health conditions. The alignment algorithm we adopt relies on a modification of the so-called edit-distance. The comprehensive description of the matching algorithm setup has been described in Chapter 3, Section 3.3.

In our implementation we first apply this algorithm to determine the pertinence of the incoming track to a known class using the $x - z$ orientation, and in a second step we run the algorithm again to discriminate the actions within the same cluster using the additional features of height, speed and time, as shown in Fig. 4.2. In this example, 4 classes are identified, each of them having one main actions and few anomalous paths. In case the occurrence of the anomalous paths is repeated several times, the trajectories can become a new action. This is the case, for example of rearrangements in the furniture that oblige the subjects to follow new routes.

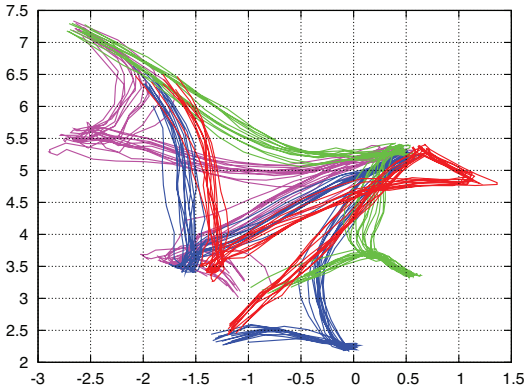


Figure 4.3: Top-view of the extracted clusters (Application Lab environment).

4.4 Experimental results

The proposed strategy has been tested in lab on a PC connected to a stereo camera, to allow a complete 3D positioning of moving objects. Since the proposed work focuses on the representation and matching of trajectories, we do not address here the problem of either video object tracking, or trajectory extraction. All tests concern the monitoring of people in indoor environments. The considered events include 4 classes of trajectories, each of them including a set of normal trajectories and a number of anomalous trajectories. The class refers to the high-level description (e.g. go to the sofa, go to the stove, etc.); the actions (A1, A2, AN) within each class describe the way in which the action is carried out (e.g. walk, run, crawl, jump). We selected the Application Lab environment (see the Section 3.4). The experiments required the proper setup of the algorithm parameters and in particular the thresholds for trajectory segmentation. Referring to [75][76], we set $\theta = 30°$, $\gamma = 0.1m^2$, the temporal delay for pause detection equal to 2 sec, and the stop area radius $\rho = 0.2m$. In addition to those parameters, here we added the vertical offset along y to 30 cm. The choice of the threshold has been performed by running the segmentation algorithm at different levels of granularity and by selecting then the values with the best accuracy for the specific application domain (people monitoring in indoor). The substitution matrices are filled according to Eq. (3.17).

We present hereafter a selection of results, with a twofold goal: first, we want to demonstrate the capability of the system in identifying similar classes of trajectories with different spatio-temporal configurations, allowing the discovery of new tracks (i.e. new clusters); second, we want to show how the proposed methodology allows disclosing specific actions within each particular class. To this aim we collected about 150 paths including 4 different classes of events, each of them performed in different configurations

Table 4.2: Inter-class statistics

FPR	0,07
Accuracy	0,9
Recall	0,82
Precision	0,758

Table 4.3: Intra-class statistics

Statistic	**Class 1**	**Class 2**	**Class 3**	**Class 4**
FPR	0	1	0	0
Accuracy	1	0,541	0,833	0,833
Recall	1	0,722	0,777	0,777
Precision	1	0,6841	1	1

(standing, walking or running, jumping, picking up objects, crawling, falling). A snapshot of the top-view trajectory classes is reported in Fig. 4.3. In particular we have considered all trajectories as normal, apart from those ones referring to falling and crawling events that should be revealed as anomalous. Concerning the capability of the system in disclosing among the 4 classes, we randomly selected a subset of 10 trajectories from each cluster (40 total paths) and performed the pairwise matching among all possible pairs in the dataset 1560 cross-comparison). The matching results are 4 independent scores for each trajectory pair, expressing the similarity in the 4 features (i.e. $x - z$ orientation, height of the centroid, speed and time) independently. In order to shrink the sequences into the available classes, the first step of the hierarchical match has been carried out setting a threshold of 70% on the $x - z$ score. The obtained results are reported in Table 4.2.

Considering a given number of known trajectory classes, the system should be also capable of detecting novel events relying on the threshold-based classification: in particular, novel trends should obtain low scores against the existing clusters. As the occurrence of these novel routes increases, the associated event is included in the database of known activities, providing the capability of learning new events. As it can be noticed from the sample scenario in Fig. 4.4, the algorithm first highlights these trajectories as anomalous, and then progressively associates them to a new Class 3, or to a new action (A3 in this case) if the occurrence exceeds a sample threshold $A_{th} = 10$. After determining the pertinence of the trajectory to one of the known classes, a new score is calculated to determine the similarity with known actions. The intra-class matching results are shown in Table 4.3.

The system demonstrated a good robustness in detecting anomalies for three out of four clusters. Only in cluster B no anomaly out of 6 has been detected: a visual inspection of the paths has revealed that the evolution of normal and anomalous trajectories was very similar, making the analysis difficult also for a human operator.

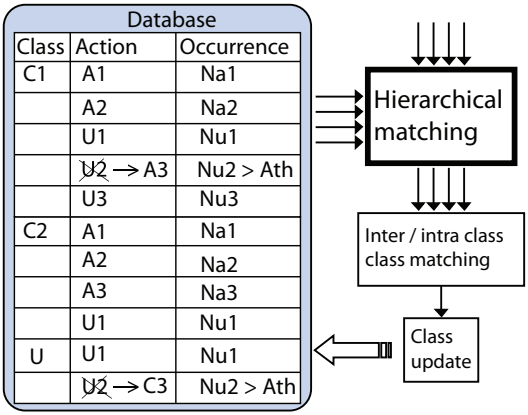


Figure 4.4: Class and Action update.

4.5 Conclusions

In this chapter we have proposed a novel scheme for hierarchical matching of 3D trajectories of pedestrians. The method relies on the definition of an ad-hoc syntax to describe the trajectory evolution, which takes into account the 3D spatial location, together with the information about speed and time. To group the trajectories and identify frequent activities, a modified version of the edit-distance has been adopted. Using the proposed approach it is possible to associate a new trajectory to the action that matches most. In case the trajectory does not belong to any of the pre-stored classes, it is placed in the set of unknown tracks that can successively be associated to a new class/action based on the number of occurrences, making the system self-configurable and adaptive.

Acknowledgment

This work has been partially developed under the A-Cube project, funded by the Provincia Autonoma di Trento (Italy).

Chapter 5

Context-Free Grammars for Activity Modeling and Matching

In this chapter we propose a novel method to analyze trajectories in surveillance scenarios by means of Context-Free Grammars. Given a training corpus of trajectories associated to a set of actions, a preliminary processing phase is carried out to characterize the paths as sequences of symbols. This representation turns the numerical representation of the coordinates into a syntactical description of the activity structure, which is successively adopted to identify different behaviors through the CFG models. Such a modeling is the basis for the classification and matching of new trajectories vs. the learned templates and it is carried out through a parsing engine that enables the online recognition of human activities. An additional module is provided to recover parsing errors (i.e., insertion, deletion, or substitution of symbols) and update the activity models previously learned. The proposed system has been validated in the context of assisted living applications, demonstrating good capabilities in recognizing activity patterns in different configurations, also in presence of noise in the acquired trajectories, or in case of concatenated and nested actions.

5.1 Introduction

In recent years there has been a relevant attention towards the implementation of automatic systems for activity detection and analysis in several application areas, including environmental monitoring and video surveillance [66]. The analysis consists of the ex-

Part of this Chapter appears in:
M. Daldoss, N. Piotto, N. Conci, F.G.B. De Natale, "Activity Detection using Regular Expressions". IEEE Int'l Workshop on Image Analysis for Multimedia Interactive Services, pp. 1-4, Desenzano del Garda - Italy, 12-14 Apr 2010.
M. Daldoss, N. Piotto, N. Conci, F.G.B. De Natale, "Learning and Matching Human Activities using Regular Expressions". IEEE Int'l Conf. on Image Processing, pp. 1-4, Hong Kong, 26-29 September 2010.
M. Daldoss, N. Piotto, N. Conci, F.G.B. De Natale, "Context-Free Grammars for Activity Modeling and Matching" in N. Adami, A. Cavallaro, R. Leonardi, P. Migliorati. Analysis, Retrieval and Delivery of Multimedia Contents, Berlin: Springer-Verlang, 2011. (Lecture Notes on Computer Science), Accepted for publication.

trapolation of meaningful information about the event occurring in the observed scene by interpreting and classifying low level features such as objects trajectories [76] [78]. There are though situations in which the trajectory analysis tools may be misleading, due to their connection to physical and geometrically referenced displacements. In fact, geometric displacements can be of great help in a lot of situations where the personalization does not play a crucial role, such as for example traffic monitoring (vehicles are driven by humans, but they have to comply with a set of rules) [74], or in case the structure and the semantic of the environment are not known a priori (generic monitoring in outdoor [86]). On the contrary, in more defined scenarios, the structure of the environment is well known, typically facilitating the interpretation of the context in which a specific activity is carried out. This is the case for example of human behavior analysis in homes or offices. In these situations the topology of the rooms is typically kept constant over a considerably long range of time. However, even though the observed space is static, subjects move freely and tend to perform also the most common actions in slightly different manners each time. This personalization factor is clearly not voluntary and relies on a number of factors that cannot easily be measured. In practice, this consists of collecting sets of trajectories, in which the point-to-point displacement of the moving subject is different each time, thus making it impossible to apply any kind of curve matching. The activity detection can though be achieved by analyzing the event at a higher-level of abstraction. Such kind of analysis aims at establishing connections between the moving subject and the environment, trying to categorize the activities on the basis of interactions. Interactions can be of different forms, but in general, and considering trajectories as the principal source of information, they can be modeled as the permanence for a specific amount of time in the neighborhood of one or more meaningful spots in the room.

This work stems from a preliminary research [21] the authors have carried out in this area, and concerns a video analysis tool that exploits regular expressions to automatically associate an observed activity pattern to a template belonging to a set of activities learned a priori. In this work actions are modeled through an abstraction of the top-view trajectory, which is summarized into a symbolic stream of *Hot Spots*, each of them corresponding to specific and relevant areas in the observed environment. The derived expressions corresponding to the activity models are automatically learned as separate CFGs using a set of training sequences. In the test phase, the resulting symbolic strings are parsed using the Earley-Stolcke algorithm to determine the similarity against all available CFGs. The main contribution with respect to the previous work consists of the progressive update of the activity database and therefore the capability of the system in adapting to the changes in the environment, as well as the users' habits.

The chapter is structured as follows. Section 5.2 provides a concise description of the CFG formalism. Section 5.3 introduces the proposed framework focusing on the representation and discovery of activities, together with the presentation of the corresponding matching strategy. The experimental validation is presented in Section 5.4 for an indoor scenario.

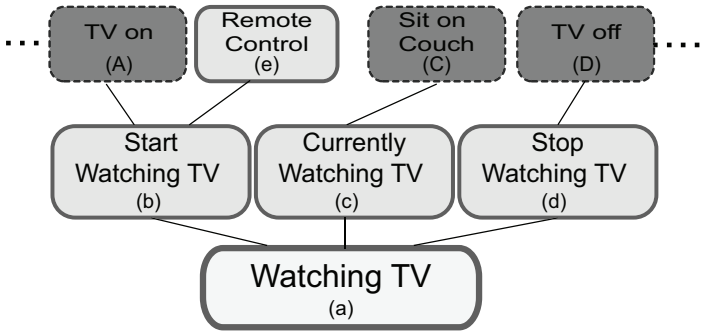


Figure 5.1: Flowchart for "Watching TV".

5.2 Overview on Context-Free Grammars

As a brief introduction to CFGs [27], we define a *language* as a set of strings over a finite set of symbols. The CFG is used as a formal mean to specify which strings belong to the language. A CFG is defined as in Eq. (5.1):

$$G = (N, \Sigma, P, S) \tag{5.1}$$

where N is a finite set of *non-terminal* symbols, Σ is a finite set of *terminal* symbols ($N \cap \Sigma = 0$), P is a finite set of rules of the form $A \rightarrow \alpha$ ($A \in N$ and $\alpha \in (N \cup \Sigma)^*$), and S is the starting symbol ($S \in N$).
In order to decide whether a given string X is compatible with a grammar G, a *parser* scans it from left to right: for each symbol X_i a set of states is constructed, representing the conditions of the recognition process. The algorithm is composed of three stages, recursively executed:

- *Prediction*: estimates the possible continuation of the input, based on the current position in the parsing process
- *Scanning*: reads the next input symbol and matches it against all pending states. The states not confirmed by the read symbol are discarded.
- *Completion*: updates the states confirmed by the scanning phase

If during *Scanning* the procedure encounters symbols that do not match any of the predicted pending states (i.e., the input does not verify any of the available rules), the procedure is aborted; the algorithm terminates successfully otherwise.

Before going into the details of the proposed solution, a toy example that describes how the grammars work, is provided in the following, in order to highlight the power and

Table 5.1: Complete CFG rules for example in Fig. 5.1.

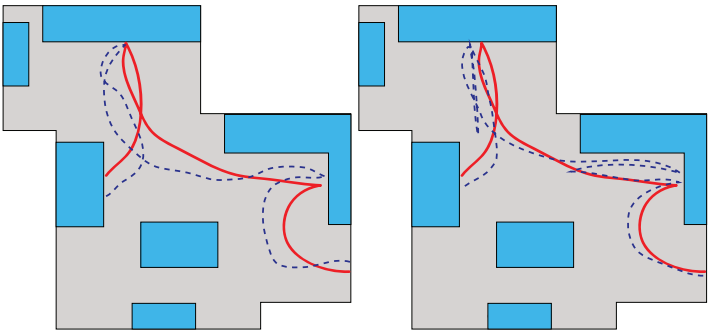
$$\begin{aligned} act &\rightarrow a \\ a &\rightarrow b\ c\ d \\ b &\rightarrow A\ e \\ e &\rightarrow \epsilon\ |B \\ c &\rightarrow C \\ d &\rightarrow D \end{aligned}$$

the flexibility of the matching through CFGs. The flowchart of Fig. 5.1 summarizes the steps required to implement the action *Watching TV*: the activity includes three main parts, i.e., the initial stage (b), the core of the action (c) and an end phase (d). Every section, represented by a *non-terminal* symbol, is composed by one or more symbols, each of them being either a *terminal* (i.e., dark grey boxes) or a *non-terminal* symbol(light grey boxes), leading to the tree structure reported in Table 5.1. Some symbols can also represent *wildcard* elements, as for example the block corresponding to the *Remote Control (e)*, making the presence of one or more specific symbols optional (ϵ). As a consequence, the sequences reported in 5.2 and 5.3 are both recognized as the action *Watching TV*:

$$act_1 \rightarrow ACD \tag{5.2}$$
$$act_2 \rightarrow ABCD \tag{5.3}$$

5.3 Proposed Framework

Observing and tracking an object in an environment allows the characterization of its motion by means of a trajectory. The trajectory consists of the frame-by-frame displacement of the moving object with respect to a reference system (the ground plane is typically adopted). In order to fulfill the requirements of the CFG framework, it is necessary to convert the numerical representation of the trajectory into a symbolic stream. This process can be carried out at different abstraction layers, such as for example the quantization of the ground floor into a coarser grid, in order to avoid sudden fluctuations of the object centroid projection. In our implementation we have chosen to synthesize the trajectory as the succession of significant spots as pointed out also in [31], therefore describing an activity as the corresponding symbol string resulting from the concatenation of these elements. At first, a number of areas of interest in the observed room is selected, and then each raw trajectory is converted into an event stream of concatenated spanned regions. This representation, although much coarser than the storage of the frame-by-frame coordinates, allows gathering an exhaustive overview of the path the moving object has performed. The motion rules identifying the training paths are automatically extracted and a specific CFG is computed for each activity, allowing a parallel and online classification of specific



(a) (b)

Figure 5.2: Two different instances (dashed line) of the same action (solid line).

behaviors.

After the initial setup, the method we propose operates as follows:

- (i) *pre-processing* of the incoming paths and representation in a symbolic domain;
- (ii) *automatic discovery* of the grammar sets encoding the motion rules for the training activities (learning);
- (iii) *verification*, through parsing, of whether an incoming trajectory fits any of the available rules (classification);
- (iv) *update* of the grammar rules corresponding to the learned activities.

The choice of a framework, purely based on syntactical elements, with respect to more traditional approaches for trajectory analysis, arises from the fact that people perform actions by introducing each time a certain level of personalization, which makes any instance of the same activity different one from each other. Therefore, the application of techniques that rely only on the spatial displacements may be in these situations less appropriate. For the same reason it can happen that the connection among relevant elements for a specific actions is at some point interrupted due external events such as for example, the phone ringing, or someone knocking at the door. In this situation, the resulting trajectory would be strongly altered and the match with the pre-stored template would be missed, even though the global action is the same. As an example in Fig. 5.2 it is possible to observe two different instances of the same action, each of them showing non-negligible spatiotemporal differences with respect to the original template.

5.3.1 Activity representation

Each trajectory T, considered as the concatenation of the moving object locations at successive time instant, is represented as a stream of 2-dimensional samples with a temporal reference as in Eq. (5.4)

$$T = \{P_i, t_i\};\ i = 0...N \tag{5.4}$$

where each sample $P_i = (x_i, z_i)$ is the projection on the ground plane of the moving object centroid at time i. As anticipated, it is worth noting that apart from the noise, object traces associated to the same activity may evolve in very different ways. This is the main motivation behind our choice of defining a number of *Hot Spots* instead of considering the whole motion trajectory, allowing for the extrapolation of a sort of *signature* of the activity. Finally, the activities are represented as the stream returned by the concatenation of *Hot Spots* the actor has interacted with, where the term interaction is reduced here to the proximity of the actor to the specific *Hot Spot* for a predefined temporal interval. This allows simplifying the representation in Eq. (5.4) to a stream of indexed regions associated to a timestamp, as in Eq. (5.5).

$$T' = \{R_j, t_j\};\ j = 0...M \tag{5.5}$$

In Eq. (5.5), R_j is an indexed *Hot Spot* and t_j is the corresponding temporal reference. Sampling the path in *Hot Spots* rather than at fixed time intervals allows preventing the potential issues arising from the acquisition phase such as noise and outliers, yet preserving the general spatial evolution of the activity.

5.3.2 CFG Rules Discovery

For any of the considered activities a set of symbolic sequences is employed in the grammar induction phase to discover its specific CFG rules. The strategy employed in this work relies on [1], a tool originally employed for NLP applications: here, each sentence (i.e., each symbolic sequence) is iteratively decomposed in *expressions* and *contexts*. For instance, in the sentence *Jack (drinks) juice*, *drinks* is the *expression* while *Jack (-) juice* is the *context*. Intuitively, given the entire set of training sentences, the algorithm searches for frequent combinations of *expressions* and *contexts* and interprets them as a grammatical type. Types are then extended, whenever possible, and the derivation rules based on the types are formulated as a CFG. Operatively, two distinct stages are carried out: initially, all the *expression/context* pairs are arranged in a matrix form, then a 2D-clustering algorithm is run to group the expressions appearing in the same context, allowing the effective grammar definition.

As a simple example taken from NLP let us consider 4 sentences:

- (s1) *Jack drinks juice*;
- (s2) *Jack likes juice*;
- (s3) *Jack drinks water*;

	(-) drinks juice	Jack (-) juice	Jack drinks (-)	(-) juice	Jack (-)	(-)	(-) likes juice	Jack likes (-)	(-) drinks water	Jack (-) water	(-) water	(-) likes water
Jack	●						●		●			●
drinks		●								●		
juice			●					●				
Jack drinks				●							●	
drinks juice					●							
Jack drinks juice						●						
likes		●								●		
Jack likes				●							●	
likes juice					●							
Jack likes juice						●						
water			●					●				
likes water					●							
Jack likes water						●						
drinks water					●							
Jack drinks water						●						

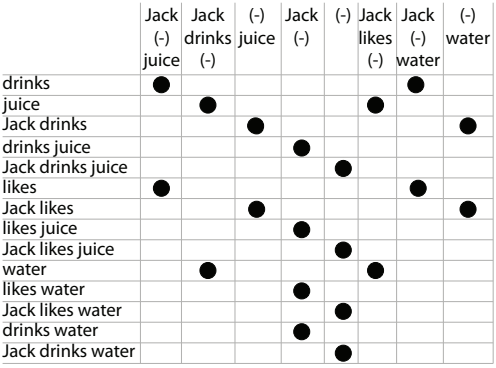
Figure 5.3: List of all the *expression/context* pairs for s1, s2, s3 and s4.

- (s4) *Jack likes water*.

All the initial *expression/context* pairs are reported in tabular format in Fig. 5.3. In an initial stage, a clustering of the context appearing in the same expressions is carried out, leading to the pairs in Fig. 5.4-(a) and to the rules in Fig. 5.4-(b). A second clustering is successfully performed leading to the final pairs shown Fig. 5.5. The final grammar rules for the considered sequences are shown in Fig. 5.5-(b).

5.3.3 CFG-based parsing for activity recognition

In order to recognize predefined activities in a stream of events, we developed an online algorithm to parse and analyze the incoming symbols. The stream of *Hot Spots* (referred to as *events*) as described in Section 5.3.1, is taken as input and the presence of symbol chunks (i.e., activities) satisfying the learned CFGs (as in Section 5.3.2) are verified. Symbols are processed as soon as they are acquired (see Fig. 5.6-(a)) and combined with the previous events to verify the pertinence of the chunk against the learned CFGs. For computational reasons, the string is parsed backwards, removing the detected chunks as soon as they are identified. In this way, an activity is detected when it ends, without the need of generating multiple hypotheses: moreover, the temporal reference of each event allows recovering both the duration and the hierarchy among possibly nested activities. In Fig. 5.6-(b) the procedure is illustrated, considering $C\ D\ G$ and $A\ G\ F\ B$ as activities: according to the parsing order, the most recent symbol at each iteration is on the left hand side of the string. In order to maintain a finite list of possible events, a time-to-live constraint is also applied to the symbols, such that the oldest events are dropped from the stream and not considered for further processing.



["Jack (.) juice",	{"drinks", "likes"}]
["Jack drinks (.)",	{"water", "juice"}]
["(.) juice",	{"Jack drinks", "Jack likes"}]
["Jack (.)",	{"drinks juice", "likes juice", "drinks water", "likes water"}}]
["(.)",	{"Jack drinks juice", "Jack likes juice", "Jack drinks water", "Jack likes water"}]
["Jack likes(.)",	{"water", "juice"}]
["Jack (.) water",	{"drinks", "likes"}]
["(.) water",	{"Jack drinks", "Jack likes"}]

(b)

Figure 5.4: (a) List of all the *expression/context* pairs for s1, s2, s3 and s4 after initial clustering; (b) explicit rules.

Besides the detection of nested actions, an additional problem consists of the activity detection in presence of noise, mainly due to sparse events not related to any of the known activities. To overcome this issue, we perform a second level of analysis. We call $G(n)$ a set of n grammars, and $G_{max}(i)$ the maximum length of any of the possible sequence belonging to the i-th grammar. From the original sequence S of length l, we compute

$$S' = S(l-1)...S(l-q) \tag{5.6}$$

with

$$q = max_i\{G_{max}(i)\} + f \tag{5.7}$$

and f the number of corrupted symbols we choose to tolerate. Starting from S', we generate m new sequences S'', removing at each step one symbol $S(x)$ from the original string, as shown in Fig. 5.6-(c). The quantity m is obtained as follows:

$$m = \sum_{i=0}^{f-1} \prod_{j=0}^{i} (q-j) \tag{5.8}$$

We apply $G(n)$ on S'', in order to verify whether the sequences belong to any of the available grammars. If so, the symbols involved in the detected action are removed from the sequence, and $S(x)$ is restored.

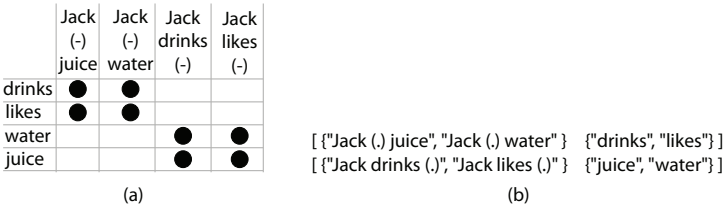


Figure 5.5: (a) List of all the *expression/context* pairs for s1, s2, s3 and s4 after the final clustering; (b) explicit final rules.

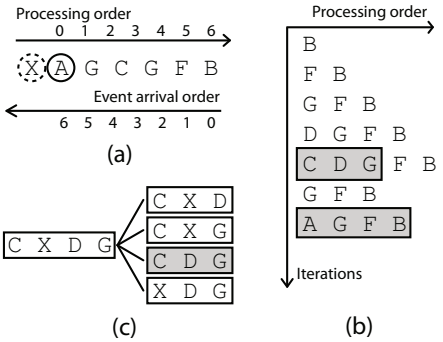


Figure 5.6: (a) Sequence layout at the parser; (b) detection of nested actions; (c) detection of 1-corrupted sequence.

5.3.4 CFG rule update

In order to maintain a coherent model for the learned activities that can take into account the temporal evolution of the activities, an update procedure is required to include potential modifications. The choice we have adopted is to concentrate on a temporal window and rebuild the grammar rules considering the most frequent observations matching a given activity within that window. To this aim, a separate database is built to keep track of all the incoming sequences fulfilling the rules of each grammar. For every sequence matching the grammar, a set of features is stored, such as the corresponding activity (i.e., the grammar) identifier and the list of symbols belonging to it, together with the temporal reference that informs about the interaction time with the specific *hotspots*. Moreover, for every grammar, the system keeps track of the most recent observation, and the list of all possible activity variations according the error tolerance strategy.

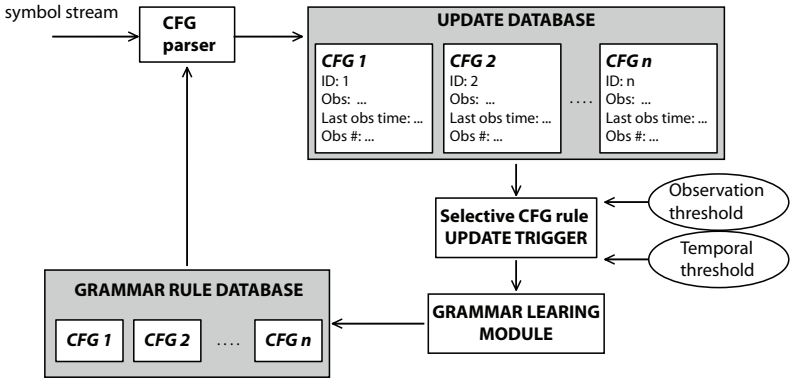


Figure 5.7: Flowchart of the CFG rule update strategy.

The system is updated on a regular basis and in accordance with the application requirements. When an update takes place, the grammar rules for the models are recalculated following the procedure illustrated in Section 5.3.2 and considering the most recent set of observations for every particular grammar. Two combined criteria are considered before updating:

- the number of occurrences of the single instance;
- the timestamp of the last recognized instance.

In other words, an update is operated only when the number of observations for a specific grammar exceeds a given occurrence threshold, and the time instant of the last observation is within given temporal window. The occurrence threshold and the temporal window extension are tunable parameters regulating the update process. As a consequence of the update, the system can successfully adapt the detection of the defined actions to the variations that may occur due to changes of the environment (e.g., new environmental objects to interact with) or in the habits of the user.

5.4 Results

In order to validate the proposed approach, we acquired a set of video sequences and extracted the top-view trajectories of each moving subject by means of calibrated camera. Being the tracking algorithm out of the scope of this work, the experimental validation will only concern the trajectory analysis module. All tests have been carried out in an assisted living lab (Fig. 5.8-(a)) and concern people monitoring while performing three classes of

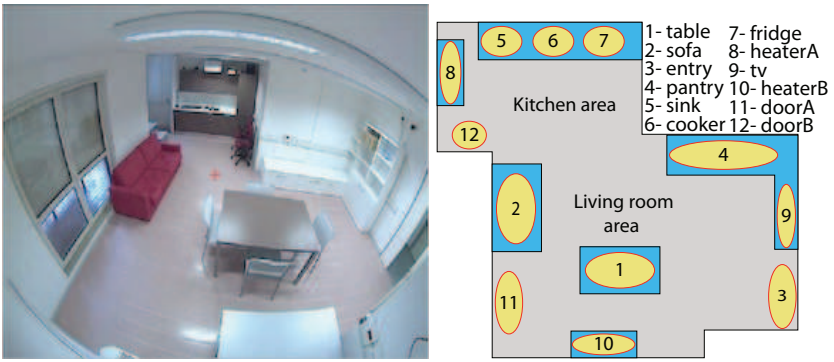


Figure 5.8: (a) Snapshot and (b) map of the environment. *Hot Spots* are provided with the corresponding legend.

activities, namely *Cooking*, *Serving food*, and *Taking a break*. In Fig. 5.9 a sample instance of each activity is reported for completeness. Given the nature of the considered activities, the pre-processing phase described in Section 5.3 is carried out considering the *Hot Spots* in Fig. 5.8-(b). For the grammar induction phase, we selected and segmented five instances of each activity obtaining three different CFGs. For conciseness reasons, only the rules for the *Taking a break* activity are reported in Fig. 5.10-(a). The choice of these actions among all possible scenarios is due to the consideration that they can occur in sequential or nested configurations, thus slightly complicating the recognition process.

The test phase is divided in two different stages: we will first show how the learned grammars can generalize, recognizing activity instances not defined in the training set; we will then show the capability of spotting activity sequences from a symbolic stream, also in a nested configurations.

For the first tests, let us take as an example the CFG learned from one simple activity (e.g., *Taking a break*), even though the reasoning can be extended to more complex grammars. As it can be seen from the rules reported in Fig. 5.10-(a), the structural skeleton of the activity is modeled through the first three rules: each rule provides for three possible variations, leading to a total number of sequences satisfying the CFG rules, equal to 9. In general, the number of sequences belonging to a given grammar depends on the grammar structure, and thus on the complexity of the sequences used in the training. In this specific case, the rules have been extracted from five training sequences (Fig. 5.10-(b)) and the grammar returns a total number of nine candidates. The generalized sequences are shown in Fig. 5.10-(c), and result in semantically legitimate examples of

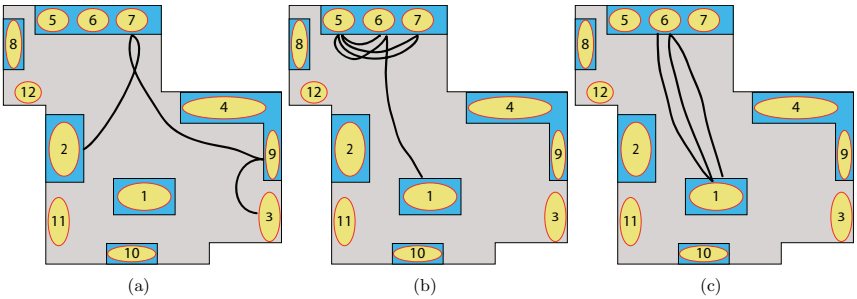


Figure 5.9: Sample activity for (a) Taking a break, (b) Cooking, and (c) Serving Food.

the activity *Taking a break*, even though they were not available in the original training set.

The second test phase aims at demonstrating the capability of the parsing strategy in spotting known activity patterns from a continuous event stream. In particular, given the backward parsing paradigm, we show how the proposed engine can recognize activities also in a nested form. To this aim, we randomly selected one activity realization for each of the three learned grammars and composed them in different nested configurations, as shown in Fig. 5.11. We first consider three consecutive activities (a), then two simply nested sequences (b), and finally a complex hierarchy among 3 activities with noisy symbols

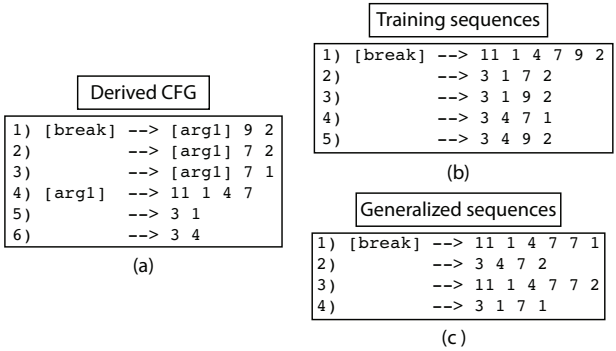


Figure 5.10: (a) CFG for *Taking a break* (regions as in Fig. 5.8); (b) training sequences; (c) generalized realizations.

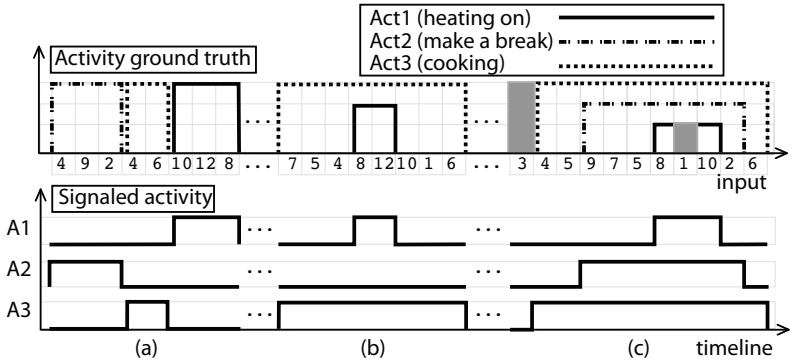


Figure 5.11: Activity spotting example: (a) 3 consecutive sequences; (b) hierarchy between 2 activities; (c) 3 nested activities with noisy events.

(i.e., grey box in the figure). From top to bottom we indicate (i) the ground truth for the activity stream, (ii) the event input stream, and (iii) the signaled activity. As it can be noticed, the system is fully capable of disclosing chunks of activities even if the incoming data stream is corrupted by noisy symbols. The algorithm we have implemented is able to carry out the analysis of the incoming data, in compliance with the real-time constraints, in order to raise alarms in case an anomaly is detected.

5.5 Conclusions

In this chapter we have proposed a reconfigurable tool for activity detection in indoor scenarios based on CFG. Starting from the trajectory acquired by the cameras installed in the environment, the algorithm takes as input a set of training trajectories. The data is processed in order to extract the meaningful information of the path, namely the interaction with a number of relevant areas in the observed environment. Through the identification of these *Hot Spots*, it is then possible to construct the CFG-based representation of the activities. During the test phase, the parsing algorithm is run to evaluate the CFG that best matches the current trajectory with respect to the acquired prototypes. The algorithm has been validated in an experimental test site targeted at monitoring daily activities of people living in the environment and it is able to recognize the activities, both as standalone, as well as in consecutive or nested hierarchies, also handling noisy events.

Chapter 6

Optimal Quantization for Robust Trajectory Analysis

The implementation of algorithms for the comparison of spatio-temporal series reveals the need of finding the most suitable data representation format to fulfill the specific application requirements. According to the properties of the signals to be analyzed and considering the noise corrupting the acquired samples, it may be convenient to consider a pre-processed and quantized representation of the data, rather than the original sequence. In this chapter we aim at defining a formal reasoning, in order to help the identification of the most suitable representation, on the basis of the quality of the collected data. The analysis is carried out considering the comparison of the sequences with respect to the L_p norm. The metric is applied on data corrupted by noise, both in the unprocessed and the quantized domains. The validation of the proposed method has been carried out in the context of trajectory analysis considering a standard dataset of real trajectories, demonstrating its effectiveness for both classification and retrieval purposes.

6.1 Introduction

Algorithms for trajectory comparison and representation techniques have received increasing attention from the scientific community due to their potential interest in various application areas including video classification and retrieval, but also environmental monitoring, video surveillance and motion-based activity recognition [12]. In both cases, a crucial role is played by the statistics of the noise affecting the acquired data: different factors are involved, such as imprecisions in the tracking algorithm, partial or complete occlusions, as well as artifacts introduced by the compression. A common procedure to limit the effects of noise is to preprocess the trajectory applying a quantizer, thus bringing the trajectory from the continuous world into a discretized/symbolic domain. Although

Part of this Chapter appears in:
N.Piotto, L.Wang, N.Conci, F.G.B.De Natale, D.Schonfeld, "Optimal Quantization for Robust Trajectory Analysis of Video Sequences", IEEE Int'l Conf. on Image Processing, 2011. Submitted

the raw representation is in general richer from the informative point of view if compared to any quantized representation, it is also particularly sensitive to noise, which can be mitigated if an ad-hoc quantizer is found.

In signal processing theory, optimum quantization has always been considered as the process of minimizing the distortion between the quantized and the original samples [56]. In this work we aim instead at implementing an optimum quantization scheme suitable for finding the best representation for sets of trajectories belonging to the same class, thus facilitating the classification and retrieval processes.

As a rule of thumb it is reasonable to think that the raw representation is more convenient in presence of low noise, while in case of higher noise intensities the quantized version becomes more appropriate, possibly improving the system performance.

To this aim, a mathematical formulation is presented, which models signals and noise as stochastic processes, allowing the definition of an optimization procedure to achieve:

- the best representation between raw/numerical and symbolic/quantized;
- the optimum quantization scheme given the statistical properties of the considered signals.

The proposed optimization algorithm can be applied in principle to any type of signal. For the sake of demonstration, in this work we have focused on trajectory matching by considering the ASL [33] database. Although computational expensive, the extension of the method to higher dimensional spaces is straightforward.

Historically, signal quantization has been implemented for modulation and analog-to-digital signal conversion. In these contexts, the traditional way to determine the quality of a quantization scheme is to measure the distortion (typically by means of the MSE) between the quantized signal in comparison with the original source [30]. As a consequence, the computation of an optimum quantization scheme is obtained by choosing the parameters that guarantee the distortion minimization [56].

A different situation occurs instead when we are required to compute the similarity between signals, acquired by a noisy process, such as for example the trajectories of moving subjects in a monitored environment. In this case, the best quantization parameters have to be chosen according to different distortion measures, considering the difference between classes of data, rather than the distance between original/symbolic signals.

Although the literature is rich of approaches relying on both raw and symbolic representations (see Section 2), it is still not clear when a representation is preferable to the other, since the choice is tightly coupled with the application context. To this aim in this work we present a formal reasoning, to drive the user in selecting the best representation with respect to the noise present in the signal samples, and to evaluate the corresponding optimal quantization parameters.

6.2 Optimal Quantization

The implementation of an optimal quantization scheme is generally aimed at achieve a better signal representation that can help the further processing steps in terms of computation complexity or storage requirements, as well as to mitigate the noise artifacts. This is particularly true in the framework of power sensitive systems, where the features have to be quantized due to space and computational constraints. In the following paragraphs we will consider two generic signals as a discrete-time random processes $X_i(t)$ with $i = 1, 2$, and t the time index. Due to the noise corruption, the observed signals can be modeled as $Y_i(t) = X_i(t) + N_i(t)$, where $N_i(t)$ is the random noise contribution. Let $Q_t(x)$ denote the quantization at each time instant t. $Q_t(x)$ is specified by a set of region boundaries (i.e., μ_l), and a set of reconstruction values (i.e., q_{l+1}), with $l = 1..k$. k is the number of quantization levels. In order to obtain a formulation for the definition of the optimum quantizer Q^*, the following terms are introduced

$$ND(t) = \|Y_1(t) - Y_2(t)\|_p - \|X_1(t) - X_2(t)\|_p \tag{6.1}$$

$$QD(t) = \|Q_t(Y_1(t)) - Q_t(Y_2(t))\|_p - \|X_1(t) - X_2(t)\|_p \tag{6.2}$$

$ND(t)$ and $QD(t)$ are the distortions w.r.t. the noise-free signal distance when adopting a numerical and a quantized representation, respectively. Given the number of quantization levels, we define the optimum quantizer as the scheme introducing the minimal distortion (in terms of L_p norm) w.r.t. the noise-free signal when comparing sequence pairs. Formally, the optimum quantizer can be obtained by maximizing the distance between $ND(t)$ and $QD(t)$ w.r.t. the parameter Q_t, as shown in Eq. (6.3).

$$Q^* = \arg\max_{Q_t} \sum_t \mathbb{E}(\|ND(t)\|_p - \|QD(t)\|_p) \tag{6.3}$$

Although the reasoning can be extended in multidimensional spaces, in the following we will apply W.L.O.G. the L_2 norm.

By considering the optimization problem in the time domain, the optimal quantization has to be computed for each time instant t. The problem can be presented as follows: let x_i, n_i, y_i with $i = 1, 2$ be realizations of the random processes $X_i(t)$, $N_i(t)$, and $Y_i(t)$, respectively.
$Q(x)$ is a quantization function of the form

$$Q(x) = \sum_{i=1}^{k} q_i \chi_{(\mu_i, \mu_{i+1})}(x) \tag{6.4}$$

where χ is the characteristic function, $q_i \in \mathbb{R}$, k is the number of quantization intervals.

The terms $y_i = x_i + n_i$ are the noisy versions of the signals at time t. We can assume a zero mean noise, W.L.O.G. The optimal quantization of level k is the solution to the following optimization problem:

$$\max \mathbb{E}(\|nd\|^2 - \|qd\|^2) \tag{6.5}$$

where $nd = \|y_1 - y_2\|^2 - \|x_1 - x_2\|^2$ and $qd = \|Q(y_1) - Q(y_2)\|^2 - \|x_1 - x_2\|^2$. We will denote the objective function in Eq. (6.5) given the quantization Q as $f(Q)$.

With the following theorem we show that, if the noise corrupting the samples is large enough, the quantization is worth to be used.

Theorem 1. *If the second or fourth order moment of noise is sufficient large, then we have* $f > 0$.

Proof. Let $X = x_1 - x_2$, $N = n_1 - n_2$. Consider the quantization Q, which has all the reconstruction levels $q_i = 0$. Under these assumptions, Eq. (6.5) can be rewritten as

$$f(Q) = \mathbb{E}((X + N)^2 - X^2)^2 - \mathbb{E}(X^4) \tag{6.6}$$

Exploiting the terms in Eq. (6.6) we obtain

$$\begin{aligned} f(Q) =& 4\mathbb{E}(X^2)\mathbb{E}(N^2) + 4\mathbb{E}(X)\mathbb{E}(N^3) + \\ & \mathbb{E}(N^4) - \mathbb{E}(X^4) \end{aligned} \tag{6.7}$$

Since $\mathbb{E}(N^2) \sim (\mathbb{E}(n_1^2) + \mathbb{E}(n_2^2))$ and $\mathbb{E}(N^4) \sim (\mathbb{E}(n_1^4) + \mathbb{E}(n_2^4))$, if the second moment or fourth moment is large enough, we have $f(Q) > 0$. □

Assuming that the noise-free signal statistics are known and invariant over time, the expression in Eq. (6.7) is a function of the noise properties only. Given a quantization scheme Q, the threshold value between the numerical and the quantized representations is obtained evaluating the noise power satisfying Eq. (6.8).

$$\mathbb{E}(\|nd\|^2) = \mathbb{E}(\|qd\|^2) \tag{6.8}$$

Optimization algorithm. In order to evaluate the optimal quantization scheme, the optimization problem of Eq. (6.5) is to be solved w.r.t. the parameters of Q_k, i.e., μ_i and q_{i+1}. Unfortunately, since the optimization process involves the minimization of a non-convex function, the solution may not converge to the global optimal quantization: in any case it guarantees the convergence to a local optimal point. Since a closed form for the final solution is not available, a gradient descent with line search is used.

If the noise-free signal and noise statistics are known, the nd term in Eq. (6.5) can be considered constant. Thus, the initial optimization problem becomes

$$\min \mathbb{E}((\|Q(y_1) - Q(y_2)\|^2 - \|x_1 - x_2\|^2)^2) \tag{6.9}$$

The optimization is started considering a uniform quantization scheme, so that the regions are defined by equally spaced μ_i and the reconstruction levels q_{i+1} are the centroids of each region. Our problem involves a multivariable optimization (i.e., $2k+1$ variables, k number of quantization levels) which is solved by computing an iterative single-variable optimization. In other words, the optimization is solved for each single variable, and considering all the others fixed: when optimizing the first region boundary μ_1, all the boundaries μ_i

with $i = 2..k$ and the reconstruction levels q_j with $j = 1..k+1$ are kept fixed. The iteration process over the considered $2k+1$ variables is stopped when a predefined decreasing rate is achieved between successive function evaluations. The pseudo-code of proposed algorithm is reported in Algorithm 2. Here, the quantities $\Phi_i(i)$, $i = x1, x2, n1, n2$ are the *pdf* functions of the noise-free signals and the corrupting noises, respectively.

Algorithm 2 Pseudo code for the optimization

Require: $\Phi_{x1}(x_1)$, $\Phi_{x2}(x_2)$, $\Phi_{n1}(n_1)$, $\Phi_{n_2}(n_2)$, k, $Stop$
Ensure: Optimal μ_i, q_j with $i = 1..k-1$, $j = 1..k$
  $OldVal = 0$, $NewVal = 0$
  Uniform quantizer variables initialization
  while $100 \times \frac{Oldval - NewVal}{Oldval} \geq STOP$ **do**
    $OldVal \leftarrow NewVal$
    for $i = 1$ to k-1 **do**
      Fix all μ_l with $l = 1..k-1 \wedge l \neq i$
      Fix all q_m with $m = 1..k$
      Find μ_i minimizing Eq. (6.9)
    end for
    for $j = 1$ to k **do**
      Fix all μ_l with $l = 1..k-1$
      Fix all q_m with $m = 1..k \wedge m \neq j$
      Find q_j minimizing Eq. (6.9)
    end for
    $NewVal \leftarrow$ Minimum value for Eq. (6.9) for this iteration
  end while

Concerning the role of the number of quantization levels k, we show with the following theorem that under the assumption of strictly increasing distribution functions, the objective function in Eq. (6.5) is strictly increasing with the number of quantization levels.

Theorem 2. *If $k_2 > k_1$, for any optimal quantization Q_1 for k_1 there exists a quantization Q_2 such that $f(Q_2) > f(Q_1)$.*

Proof. Let Φ denote the probability distribution function. It is enough to consider the qd part of Eq. (6.5) (i.e., $qd = \mathbb{E}(\|\|Q(y_1) - Q(y_2)\|^2 - \|x_1 - x_2\|^2\|^2)$).

Assume there is a quantization scheme Q_0 such that $\forall k' > k$, we have $f(Q') \leq f(Q_0)$. For Q_0, let Q' be the degenerated situation, i.e. $q_i = q_{i+1}$ for some i. We have

$$qd = \sum_{i,j} \int ((q_i - q_j)^2 - X^2)^2 \Phi_{y_1,y_2}(\mu_i, \mu_{i+1}, \mu_j, \mu_{j+1}) d\Phi_X \tag{6.10}$$

Note that $qd(q_i, q_{i+1}) : \mathbb{R}^2 \to \mathbb{R}$ is a polynomial. By the assumption that qd only depends on the differences of the quantization levels, i.e., $qd(x, x) \equiv 0$ for any $x \in \mathbb{R}$, therefore we

have that qd has to be a constant. Also, if we perturb μ_i within the interval $[\mu_{i-1}, \mu_{i+1}]$, the value of $f(Q')$ will not change, simply because we have a degenerate case. In other words, the function $qd'(q_i, q_{i+1}, \mu_i)$ is a constant function. However, since all the distributions are strictly increasing, we can always perturb μ_i, which corresponds to altering $qd(q_i, q_{i+1})$, contradicting the statement that qd' is a constant function. □

If we assume that the limit of quantization converges as the number of intervals increases, the optimal number is defined as:

$$Q^{opt} := \lim_{k \to \infty} Q_k \tag{6.11}$$

However, we may not know the Q^{opt}, since the algorithm only converges to local minima, and also in case we achieved the global optimal for each k, the limit may not converge point-wise.

The following results show that under certain circumstances, and if the estimate of the signal distribution is not accurate, the error increases with the number of quantization intervals.

Theorem 3. *There exists a form of perturbation on the signal distribution function such that if $k_2 > k_1$, for optimal quantizations Q_1 and Q_2 for k_1 and k_2, we have $f'(Q_2) < f'(Q_1)$, where f' denotes the perturbation of f.*

Proof. Since the noise is modeled as additive, any perturbation on the signals can be viewed as perturbation on the noise. Therefore, we can assume the perturbed distribution $\Phi'_{y_1,y_2} = \Phi_{y_1,y_2} - \Delta\Phi_{y_1,y_2}$. We then obtain

$$\begin{aligned} f'(Q_2) - f'(Q_1) = \sum_{i,j} \int ((q_i(Q_2) - q_j(Q_2))^2 - X^2)^2 d\Phi_X \cdot \\ \Delta\Phi_{y_1,y_2}(\mu_i(Q_2), \mu_{i+1}(Q_2), \mu_j(Q_2), \mu_{j+1}(Q_2)) d\Phi_X - \\ \sum_{i,j} \int ((q_i(Q_1) - q_j(Q_1))^2 - X^2)^2 d\Phi_X \cdot \\ \Delta\Phi_{y_1,y_2}(\mu_i(Q_1), \mu_{i+1}(Q_1), \mu_j(Q_1), \mu_{j+1}(Q_1)) d\Phi_X \end{aligned} \tag{6.12}$$

requiring the definition of a specific $\Delta\Phi_{y_1,y_2}$.

For Q_1, choose i_0 and j_0 such that

$$\int ((q_{i_0} - q_{j_0})^2 - X^2)^2 d\Phi_X = \sup_{a,b} \int ((a-b)^2 - X^2)^2 d\Phi_X \tag{6.13}$$

where $\mu'_{i_0} < a < \mu'_{i_0+1}$ and $\mu'_{j_0} < b < \mu'_{j_0+1}$ for some $\mu'_{i_0}, \mu'_{i_0+1}, \mu'_{j_0}, \mu'_{j_0+1}$.

Then, it is easy to construct $\Delta\Phi_{y_1,y_2}$, which vanishes outside the range $\{\mu'_{i_0} < a < \mu'_{i_0+1}, \mu'_{j_0} < b < \mu'_{j_0+1}\}$.

We now define

$$F(a,b) = \int ((a-b)^2 - X^2)^2 d\Phi_X \tag{6.14}$$

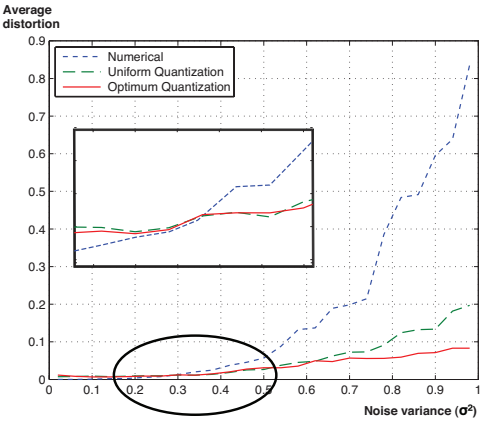


Figure 6.1: Average distortion for the original representation, and uniform and optimum quantization, as function of the noise variance (ASL dataset).

and consider the Riemann-Stieltjes integral

$$G = \int F(y_1, y_2) d\Delta\Phi_{y_1,y_2}. \tag{6.15}$$

Let $S(F, P_i)$ denote the Riemann-Stieltjes sum of G for partition P_i inherit from Q_i, $U(F, P)$ for the upper Riemann sum. Therefore, rewriting Eq. (6.12) we have

$$\begin{aligned} f'(Q_2) - f'(Q_1) = \\ S(F, P_2) - U(F, P_1) \leq \\ U(F, P_2) - U(F, P_1) \leq 0 \end{aligned}$$

The last inequality follows from the non-increasing property of upper Riemann sum [82]. □

6.3 Trajectory Analysis

The validation of the proposed framework has been carried out in the context of trajectory analysis, by adopting the ASL [33] dataset. In the experiments we show the average evolution of the terms $\|ND\|^2$ and $\|QD\|^2$ of Eq. (6.1) and (6.2) by increasing the noise variance, giving a quantitative evidence of the advantages introduced by using a symbolic representation in presence of significant noise (Section 6.2, Theorem 1).
As a final step, we will give experimental evidence to Theorem 2 of Section 6.2, by showing

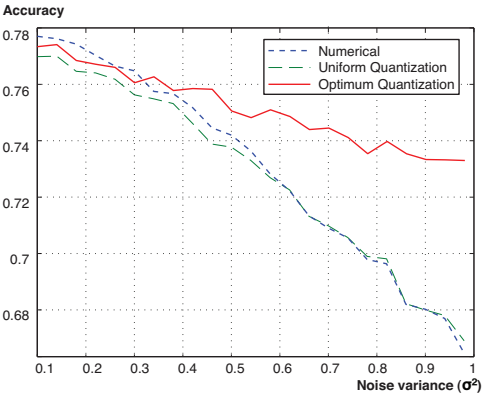


Figure 6.2: Classification performance: accuracy *vs.* noise variance (ASL).

how the objective function f is strictly increasing with the number of quantization levels. The implemented framework is then used on the selected dataset to validate the applicability in both classification and retrieval applications. In order cope with 2 dimensional time series, and define a proper quantization scheme, we applied the Centroid Distance Function (CDF) transformation as described in [7]. The number of quantization levels has been fixed to 7 for each cluster. After estimating the *pdfs* for each considered trajectory class, we applied the optimization routine described in Section 6.2 for the evaluation of the best quantization parameters for each cluster. In this phase, the optimization is initialized with a uniform quantizer centered about the mean of the cluster distribution. The noise is considered to be additive and Gaussian. 25 different levels of noise power have been considered for validation, ranging from 10% to 98% of the original signal variance. Each level results in a specific quantization scheme for each class.

In order to show the trend of the distortion introduced by the different representations with increasing noise power, the average pairwise distance between samples of the same class has been calculated for three configurations, and for each level in the noise variance. In particular, we evaluated the distortion between the noise-free signal difference and the difference of the same samples corrupted by noise in the numerical, uniformly, and optimally quantized domains. Figure 6.1 depicts the average distortion for the considered configurations (y-axis) w.r.t the noise power (x-axis). As it can be observed, the best performance is achieved, in case of low noise, by the numerical representation; on the contrary, as the noise power increases, the average distortion for the numerical representation rapidly grows, while the distortion for the optimal quantized representation increases more slowly.

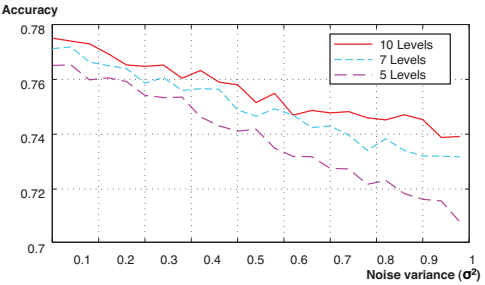


Figure 6.3: Classification accuracy for symbolic representation in different quantization level configurations.

In the second simulation phase, we applied the proposed framework to a simple classification problem involving the three different trajectory classes. The classification accuracy in the three configurations (i.e., numerical, uniform quantization, and optimum quantization) has been computed at different noise levels. The classification process works as follows. The optimum and uniform quantization schemes are obtained for each class, as described in Section 6.2, for 25 noise levels. For each trajectory class a test set of 69 samples is considered and the centroid is selected as the template for the cluster. Given an incoming path corrupted by noise n, the distance between the trajectory and each cluster template is evaluated in the three considered configurations: the label of the template with minimal distance from the incoming path is selected as the best match. Since the original trajectory is corrupted by artificially generated noise, to guarantee the statistics reliability of our results, the algorithm is run 100 times for each configuration, and considers the average value as the outcome of the analysis. Figure 6.2 reports the average classification accuracy for the three classes as function of the noise variance, confirming that for low noise power, the best classification results are through a numerical representation. On the contrary, as the variance of the additive noise increases, the numerical representation performance reduces dramatically, highlighting the advantages of the symbolic representation.

In order to give experimental evidence to Theorem 2 of Section 6.2, we solved the same classification problem considering quantization schemes with increasing number of levels. Figure 6.3 depicts the curve for the classification accuracy in case of 5, 7, and 10 quantization levels. It is possible to notice how the values of the function f are strictly increasing with the number of quantization levels considered.

Finally, the same dataset has been used for retrieval purposes. In this phase we considered all the trajectories from all the classes as queries (a total of 69*3 = 207 queries), obtaining a ranked list for all of them. Leave-one-out strategy has been chosen. The process has been iterated over the 25 noise levels, as before. In order to show the retrieval

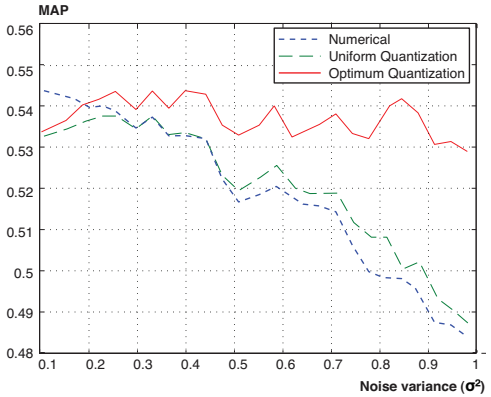


Figure 6.4: Retrieval performance in terms of Mean Average Precision *vs.* noise variance.

performances at increasing noise power, at each noise variance level the Mean Average Precision (MAP) of all queries has been evaluated. The MAP is generally referred to as the geometrical area under the Precision-Recall curve and it's evaluated according the formula in Eq. (6.16).

$$MAP = \frac{\sum_{r=1}^{N}(P(r) \cdot W(r))}{\#\quad of\quad relevant\quad documents} \tag{6.16}$$

where N is the number of retrieved documents, $W()$ is a binary function weighting the relevance at a given rank, and $P(r)$ is the precision evaluated at rank r. The retrieval results are reported in Fig. 6.4 as MAP value (y-axis) *vs.* noise variance (x-axis). Also in this case, the performances in terms of MAP confirm that for low noise power, the numerical representation outperforms the symbolic representation, while for increasing noise variance the optimal quantization offers the best performances.

6.4 Conclusions

A mathematical framework is presented to evaluate the applicability of numerical *vs.* symbolic representations to compare signals corrupted by additive noise. The formulation allows the definition of a threshold for the noise power as a function of the signal and noise statistics. Given the threshold, one can decide whether to adopt the raw or the symbolic representation. In order to find the best quantization scheme, an optimization problem is implemented and solved to maximally reduce the distance between the the noise-free and the quantized signals. Simulations have been run considering 2D trajectories with known

noise statistics. The results in classification and retrieval demonstrate that for low noise power, the best solution is to adopt a numerical or raw representation, while for noise power exceeding a given threshold, the symbolic or quantized representation is more appropriate. Although the experimental section is focused on trajectory analysis, it is fundamental to note the underlying mathematical framework is application-independent, thus the reasoning can be applied to any field, in which a signal comparison in raw or symbolic domain is involved. Future enhancements will be voted to the extension of the L_p norms-based optimization to other distance measures. Other aspects concern the series modeling: instead of converting the 2D path to 1D series, it's possible to jointly consider both the dimensions in a unique optimization procedure to obtain a multidimensional (i.e., vector) quantizer.

Acknowledgment

This work has been developed within the Multimedia Communication Lab, University of Illinois at Chicago, under the supervision of prof. Dan Schonfeld.

Chapter 7

Conclusions and Future Work

The work carried out and described in this thesis has been motivated by the need of bridging the semantic gap between the low-level activity observations and the high-level concepts describing an object activity. Assuming a correlation between a specific activity and the corresponding object motion patterns, the research focused on the development of innovative representation and comparison techniques for motion trajectories. Although several application domains are interested in automatic activity analysis, we focused on monitoring and surveillance scenarios: anyway, the proposed solutions can be customized in several application contexts with a minimal effort.

The general approach is to extract and code in symbols some spatio-temporal signature from the object traces, allowing the adoption of comparison techniques relying on symbolic information. In particular, we have proposed two approaches: the former, detailed in Chater 3, builds the activity signature on the space-time discontinuities of the path, representing the activity as the symbolic coding of these samples. Once the signature is extracted and coded in symbols, the comparison is computed adopting the so-called approximate matching. The scheme has been initially validated considering 2D trajectories, but also a 3D extension has been proposed and described in Chapter 4.

Instead, in the latter approach, described in Chapter 5, a more topological representation is chosen, being the signature coded as the concatenation of some relevant areas (i.e., *Hot Spots*) in the environment. In this case, in order to fully exploit the structural content of the patterns, the signatures of a given activity have been associated with an automatically derived Context-Free Grammar, casting the activity comparison to a string-parsing problem.

As far as the first approach is concerned, being the particular representation scheme derivative by nature, it allows considering interesting invariance properties, such as to shift, rotation, and scale factors, making the strategy suitable for detecting patterns in different spatio-temporal configurations; moreover, the trajectory comparison based on approximate-matching techniques introduces some flexibility in the comparison process, allowing a significant robustness to low-level noise artifacts. Also, both the representation and the matching schemes can be easily extended in multidimensional spaces. This method has proved its reliability in several experiments, however, in case of particu-

larly complex motion patterns where length of the signature strings tends to increase, we recorded a progressive inefficiency in the string matching.

In order to enhance the robustness, the second approach can be adopted, allowing a more compact activity representation, and a more flexible reasoning for the matching. However, due to constraints in the CFG learning algorithms, the induction of the grammar rules can be executed only considering positive samples, thus bringing the system to be particularly sensible to false alarms.

Although not offering comparable flexibility in terms of description, a possible alternative to overcome the CFG induction problems is to adopt a different graphical model for the activity coding, such as Finite State Automaton. In fact, in this context, algorithmic solutions are available for learning FSA topology and parameters from positive and negative examples, allowing a proper activity coding.

The approaches presented in this thesis cover some important aspects for the development of automatic activity analysis systems. We think that the contribution of this work, detailed in the chapter, can appreciably increase the knowledge in this area, reducing the distance towards the final goal of solve this problem. As future development, some aspects can be considered with greater detail: as far as the first approach is concerned, more flexible algorithms for signature extraction can be developed, also other scheme can be implemented. In light of this, the matching strategy should be self-reconfigurable, allowing an online evaluation of substitution matrix entries according the low-level symbol meanings.

Regarding instead the second approach, the system can be enhanced with an automatic detection of *Hot Spots*, allowing the system to build the topological map of the environment by itself. Moreover, an algorithm for the rule induction should be developed considering both positive and negative samples, allowing a better pattern representation, and a superior capability in discerning different activity classes.

Bibliography

[1] P. Adriaans and M. Vervoort. The emile 4.1 grammar induction toolbox. In *Grammatical Inference: Algorithms and Applications*, volume 2484 of *Lecture Notes in Computer Science*, pages 714–718. Springer-Verlag GmbH, 2002. 60

[2] N. Anjum and A. Cavallaro. Unsupervised fuzzy clustering for trajectory analysis. *IEEE Int'l Conf. on Image Processing*, 3:213–216, 2007. 20, 27

[3] N. Ansari and E.J. Delp. On detecting dominant points. *Pattern Recognition*, 24(5):441–451, 1991. 9, 15

[4] F. Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3):183–193, 1954. 10

[5] D.H. Ballard. Strip trees: A hierarchical representation for curves. *Commun. ACM*, 24(5):310–321, 1981. 9, 15

[6] F. Bashir, A. Khokhar, and D. Schonfeld. Segmented trajectory based indexing and retrieval of video data. *IEEE Int'l Conf. on Image Processing*, 2:623–626, 2003. 13, 15

[7] F. Bashir, A. Khokhar, and D. Schonfeld. View-invariant motion trajectory-based activity classification and recognition. *Multimedia Systems*, 12:45–54, 2006. 76

[8] F. Bashir, A. Khokhar, and D. Schonfeld. Real-time motion trajectory-based indexing and retrieval of video sequences. *IEEE Trans. on Multimedia*, 9(1):58–65, 2007. 15

[9] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. *Workshop on Knowledge Discovery and Databases*, pages 359–370, 1994. 16, 27

[10] J. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, 1998. 25

[11] N. Brandle, D. Bauer, and S. Seer. Track-based finding of stopping pedestrians - a practical approach for analyzing a public infrastructure. *IEEE Int'l Conf. on Intelligent Transportation Systems*, pages 115–120, 2006. 33

[12] M. Broilo, N. Piotto, G. Boato, N. Conci, and F.G.B. De Natale. Object trajectory analysis in video indexing and retrieval applications. In *Video Search and Mining*, volume 287, pages 3–32. Springer-Verlag GmbH, 2010. 69

[13] S. Calderara, R. Cucchiara, and A. Prati. A Dynamic programming technique for classifying trajectories. *IEEE Int'l Conf. on Image Analysis and Processing*, pages 137–142, 2007. 22, 27, 44, 46

[14] W.S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments. In *Int'l Symposium on Algorithms and Computation*, pages 378–387. Springer-Verlag GmbH, 1992. 8, 9

[15] L. Chen, M. T. Özsu, and V. Oria. Symbolic representation and retrieval of moving object trajectories. In *ACM SIGMM Int'l Workshop on Multimedia Information Retrieval*, pages 227–234, 2004. 22, 27

[16] W. Chen and S.F. Chang. Motion trajectory matching of video objects. *IS&T/SPIE*, pages 544–553, 2000. 15

[17] X. Chen, D. Schonfeld, and A. Khokhar. Robust null space representation and sampling for view-invariant motion trajectory analysis. In *IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, pages 1–6, June 2008. 7, 13

[18] X. Chen, D. Schonfeld, and A. Khokhar. View-invariant tensor null space representation for multiple motion trajectory retrieval and classification. In *IEEE Int'l Conf. on Acoustics, Speech and Signal Processing*, pages 3545–3548, 2009. 7, 13

[19] R. Cucchiara, C. Grana, G. Neri, M. Piccardi, and A. Prati. The sakbot system for moving object detection and tracking. *ACM Int'l Conf. on Multimedia*, pages 223–226, 2002. 32

[20] M. Daldoss, N. Piotto, N. Conci, and F.G.B. De Natale. Activity detection using regular expressions. In *IEEE Int'l Workshop on Image Analysis for Multimedia Interactive Services*, pages 1–4, 2010. 13

[21] M. Daldoss, N. Piotto, N. Conci, and F.G.B. De Natale. Learning and matching human activities using regular expressions. In *IEEE Int'l Conf. on Image Processing*, pages 4681–4684, 2010. 56

[22] G. Das, D. Gunopoulos, and H. Mannila. Finding similar time series. *European Symp. on Principles of Data Mining and Knowledge Discovery*, pages 88–100, 1997. 17, 27

[23] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1:1542–1552, August 2008. 17

[24] D.H. Douglas and T.K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The Int'l Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973. 9

[25] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. New York, 1973. 9, 15

[26] T.V. Duong, H. Bui, D.Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. *IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 1:838–845, 2005. 24, 25

[27] J. Earley. An efficient context-free parsing algorithm. *Communications of ACM*, 13(2):94–102, 1970. 57

[28] G. Foresti. Object recognition and tracking for remote video surveillance. *IEEE Trans. on Circuits and Systems for Video Technology*, 9:1045–1062, 1999. 30

[29] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application toautomatic hierarchical classification of silhouettes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(12):1312–1328, 1999. 13

[30] R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Trans. on Information Theory*, 44(6):2325 –2383, Oct 1998. 70

[31] R. Hamid, S. Maddi, A. Johnson, A. Bobick, I. Essa, and C. Isbell. A novel sequence representation for unsupervised analysis of human activities. *Journal of Artificial Intelligence*, 173(14):1221–1244, 2009. 26, 58

[32] A. Held, K. Abe, and C. Arcelli. Towards a hierarchical contour description via dominant point detection. *IEEE Trans. on Systems, Man and Cybernetics*, 24(6):942–949, 1994. 10, 15

[33] S. Hettich and S.D. Bay. The uci kdd archive [http://kdd.ics.uci.edu]. *University of California, Department of Information and Computer Science*, 1999. 70, 75

[34] S.Y. Ho and Y.C. Chen. An efficient evolutionary algorithm for accurate polygonal approximation. *Pattern Recognition*, 34(12):2305–2317, 2001. 11, 15

[35] J.W. Hsieh, S.L. Yu, and Y.S. Chen. Motion-based video retrieval by trajectory matching. *IEEE Trans. on Circuits and Systems for Video Tech.*, 16(3):396–409, 2006. 13, 15, 30

[36] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Trans. on System, Man and Cybernetics*, 34(3):334–352, 2004. 16, 30

[37] W. Hu, D. Xie, Z. Fu, W. Zeng, and S. Maybank. Semantic-based surveillance video retrieval. *IEEE Trans. on Image Processing*, 16(4):1168–1181, 2007. 22, 27

[38] W.M. Hu, D. Xie, and T.N. Tan. A hierarchical self-organizing approach for learning the patterns of motion trajectories. *IEEE Trans. on Neural Network*, 15:135–144, 2004. 21, 27

[39] Y. Ikebe and S. Miyamoto. Shape design, representation, and restoration with splines. *Picture Engineering*, pages 75–95, 1982. 15

[40] Y.A. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000. 13, 25, 27

[41] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14:609–615, 1996. 20, 21, 27

[42] I.T. Jolliffe. *Principal component analysis*. Springer-Verlag GmbH, 2002. 13, 15

[43] Seong-Wook Joo and R. Chellappa. Attribute grammar-based event recognition and anomaly detection. In *IEEE Int'l Conf. on Computer Vision and Pattern Recognition Workshop*, pages 107 – 107, June 2006. 26

[44] I.N. Junejo, O. Javed, and M. Shah. Multi feature path modeling for video surveillance. *IEEE Int'l Conf. on Pattern Recognition.*, 2:716–719, Aug. 2004. 21, 27

[45] T. Kaneko and M. Okudaira. Encoding of arbitrary curves based on the chain code representation. *IEEE Trans. on Communications*, 33(7):697–707, 1985. 7

[46] E. Keogh and M.J. Pazzani. Scaling up dynamic time warping for data mining application. In *Int'l Conf. on Knowledge Discovery and Data Mining*, pages 285–289, 2000. 16, 27

[47] E. Keogh and C.A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.*, 7:358–386, March 2005. 17

[48] D.E. Knuth. Semantics of context-free languages. *Theory of Computing Systems*, 2(2):127–145, June 1968. 26, 27

[49] T. Kohonen. Self-organizing maps. *Springer - Verlag GmbH*, 1995. 21

[50] B. Kumar Ray and K.S. Ray. Determination of optimal polygon from digital curve using L1 norm. *Pattern Recognition*, 26(4):505–509, 1993. 9, 15

[51] Y. Kurozumi and W.A. Davis. Polygonal approximation by the minimax method. *Computer Graphics and Image Processing*, 19(3):248–264, 1982. 9, 15

[52] B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 24, 25

[53] J.G. Leu and L. Chen. Polygonal approximation of 2-D shapes through boundary merging. *Pattern Recognition Letters*, 7(4):231–238, 1988. 9, 15

[54] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966. 22, 24, 31

[55] X. Li, W. Hu, and W. Hu. A coarse-to-fine strategy for vehicle motion trajectory clustering. *Int'l Conf. on Pattern Recognition.*, 1:591–594, 2006. 19, 20, 27

[56] S. Lloyd. Least squares quantization in pcm. *IEEE Trans. on Information Theory*, 28(2):129 – 137, march 1982. 70

[57] S. Loncaric. A survey of shape analysis techniques. *Pattern recognition*, 1998. 15

[58] X. Ma, F. Bashir, A. Khokhar, and D. Schonfeld. Event analysis based on multiple interactive motion trajectories. *IEEE Trans. Circuits System Video Techn.*, 19(3):397–406, 2009. 13

[59] A. Marzal and E. Vidal. Computation of normalized edit distance and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15:926–932, 1993. 21

[60] A. Mecocci and M. Pannozzo. A completely autonomous system that learns anomalous movements in advanced videosurveillance applications. *IEEE Int'l Conf. on Image Processing*, 2:II–586–589, 2005. 21, 27

[61] G. Medioni and Y. Yasumoto. Corner detection and curve representation using cubic B-splines. In *IEEE Int'l Conf. on Robotics and Automation*, volume 3, 1986. 7, 11

[62] A. Mikheev, L. Vincent, V. Faber, L.T. Inc, and W.A. Seattle. High-quality polygonal contour approximation based on relaxation. In *Int'l Conf. on Document Analysis and Recognition*, pages 361–365, 2001. 10, 15

[63] D. Minnen, I. Essa, and T. Starner. Expectation grammars: leveraging high-level expectations for activity recognition. *IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2:626–632, 2003. 25

[64] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Trans. on Automatic Control*, 26(1):17–32, 1981. 12

[65] D. Moore and I. Essa. Recognizing multitasked activities using stochastic context-free grammar. *Conf. on Artificial Intelligence*, pages 770–776, 2002. 26

[66] B.T. Morris and M.M. Trivedi. A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance. *IEEE Trans. on Circuits and Systems for Video Tech.*, 18(8):1114–1127, 2008. 55

[67] F.G.B. De Natale, A. Katsaggelos, O. Mayora, and Y. Wu eds. Special issue on signal processing technologies for ambient intelligence in home-care applications. *EURASIP Journal of Advances in Signal Processing*, 2007. 30

[68] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970. 5, 13, 23, 24, 30, 36, 48

[69] N.T. Nguyen, D.Q. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov models. *IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2:955–960, 2005. 25

[70] K. J. O'Connell, M. Inc, and I. L. Schaumburg. Object-adaptive vertex-based shape coding method. *IEEE Trans. on Circuits and Systems for Video Tech.*, 7(1):251–255, 1997. 11, 15

[71] J. Owens and A. Hunter. Application of the self-organizing map to trajectory classification. *IEEE Int'l Workshop Visual Surveillance*, 18:77–83, 2000. 21, 27

[72] T. Pavlidis. Polygonal approximations by newton's method. *IEEE Trans. on Computing*, 26(8):800–807, 1977. 7

[73] C. Piciarelli, G. Foresti, and L. Snidaro. Trajectory clustering and its application for video surveillance. *IEEE Int'l Conf. on Advanced Video and Signal Based Surveillance*, pages 40–45, 2005. 20, 27

[74] C. Piciarelli, C. Micheloni, and G.L. Foresti. Trajectory-based anomalous event detection. *IEEE Trans. on Circuits and Systems for Video Tech.*, 18(11):1544–1554, 2008. 56

[75] N. Piotto, N. Conci, and F.G.B. De Natale. Syntactic matching of pedestrian trajectories for behavioral analysis. In *IEEE Workshop on Multimedia Signal Processing*, pages 877–882, 2008. 23, 30, 48, 49, 50, 51, 52

[76] N. Piotto, N. Conci, and F.G.B. De Natale. Syntactic matching of trajectories for ambient intelligence applications. *IEEE Trans. on Multimedia*, 11(7):1266 –1275, nov. 2009. 13, 14, 15, 23, 27, 48, 49, 50, 51, 52, 56

[77] F. Porikly. Trajectory distance metric using hidden markov model based representation. *IEEE European Conf. on Computer Vision, PETS Workshop*, 2004. 19, 27

[78] A. Prati, S. Calderara, and R. Cucchiara. Using circular statistics for trajectory shape analysis. In *IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, pages 1–8, 2008. 56

[79] B.K. Ray and K.S. Ray. A new split-and-merge technique for polygonal approximation of chain coded curves. *Pattern Recognition Letters*, 16(2):161–169, 1995. 10, 15

[80] E.S. Ristad, P.N. Yianilos, M.T. Inc, and N.J. Princeton. Learning string-edit distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998. 14

[81] P.L. Rosin. Techniques for assessing polygonal approximations of curves. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(6):659–666, 1997. 8

[82] W. Rudin. *Principles of mathematical analysis*. McGraw-Hill Book Co., New York, third edition, 1976. International Series in Pure and Applied Mathematics. 75

[83] I.J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quarterly of Applied Mathematics*, 4:45–99, 1946. 15

[84] J. Sklansky and V. Gonzalez. Fast polygonal approximation of digitized curves. *Pattern Recognition*, 12(5):327–331, 1980. 9, 15

[85] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981. 5, 23, 24, 30, 36, 48

[86] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000. 56

[87] N. Sumpter and A. Bulpitt. Learning spatio-temporal patterns for predicting object behavior. *Image and Visio Computing*, 18:697–704, 2000. 21, 27

[88] C.H. Teh and R.T. Chin. On the detection of dominant points on digital curves. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(8):859–872, 1989. 10, 15

[89] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. *Int'l Conf. on Data Engineering*, pages 673–684, 2002. 18, 27, 44, 46

[90] O. Von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373, 1992. 7, 15

[91] J. P. Wachs, H. Stern, and Y. Edan. Cluster labeling and parameter estimation for the automated setup of a hand-gesture recognition system. In *IEEE Int'l Conf. on Systems, Man and Cybernetics*, volume 35, pages 932–944, 2005. 30

[92] K. Wall and P.E. Danielsson. A fast sequential method for polygonal approximation of digitized curves. *Computer Vision Graphics Image Processing*, 28(3):220–227, 1984. 9

[93] L. Wang, H. Ning, T. Tan, and W. Hu. Fusion of static and dynamic body biometrics for gait recognition. *IEEE Trans. on Circuits and Systems for Video Tech.*, 14:149–158, 2004. 30

[94] Q. Wang, W. Shin, X. Liu, Z. Zeng, C. Oh, B.K. AlShebli, M. Caccamo, C.A. Gunter, E. Gunter, J. Hou, K. Karahalios, and L. Sha. I-living: An open system architecture for assisted living. In *IEEE Int'l Conf. on Systems, Man and Cybernetics*, volume 5, pages 4268–4275, 2006. 30

[95] G. Wu, Y. Wu, L. Jiao, Y.F. Wang, and E.Y. Chang. Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. In *ACM Int'l Conf. on Multimedia*, pages 528–538. ACM New York, NY, USA, 2003. 13

[96] A. Yilmaz, O. Javed, and M. Shah. Object tracking: a survey. *ACM Journal of Computing Surveys*, 38(4), 2006. 30

[97] P.Y. Yin. A tabu search approach to polygonal approximation of digital curves. *Journal of pattern recognition and artificial intelligence*, 14(2):243–255, 2000. 11, 15

[98] P.Y. Yin. Ant colony search algorithms for optimal polygonal approximation of plane curves. *Pattern Recognition*, 36(8):1783–1797, 2003. 11, 15

[99] P.Y. Yin. A discrete particle swarm algorithm for optimal polygonal approximation of digital curves. *Journal of Visual Communication and Image Representation*, 15(2):241–260, 2004. 11, 15

[100] Z. Zhang, T. Tan, and K. Huang. An extended grammar system for learning and recognizing complex visual events. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(2), February 2011. 26, 27

[101] J.B. Zheng, D.D. Feng, and R.C. Zhao. Trajectory Matching and Classification of Video Moving Objects. *IEEE Workshop on Multimedia Signal Processing*, 10:1–4, 2005. 22, 27

[102] P. Zhu and P.M. Chirlian. On critical point detection of digital shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(8):737–748, 1995. 10, 15

Publications

Journals and Book Chapters

J1 - M. Daldoss, N. Piotto, N. Conci, F.G.B. De Natale, "Context-Free Grammars for Activity Modeling and Matching" in N. Adami, A. Cavallaro, R. Leonardi, P. Migliorati. Analysis, Retrieval and Delivery of Multimedia Contents, Berlin: Springer-Verlang, 2011. Lecture Notes on Computer Science. Accepted for publication.

J2 - N. Piotto, M. Broilo, G. Boato, N. Conci, F.G.B. De Natale, "Object Trajectory Analysis in Video Indexing and Retrieval Applications" in D. Schonfeld, C. Shan, D. Tao, L. Wang. Video Search and Mining, Berlin: Springer-Verlag, 2010, pp. 1-30 - Studies in Computational Intelligence - ISBN: 978-3-642-12899-8.

J2 - N. Piotto, N. Conci, F.G.B. De Natale, "Syntactic Matching of Trajectories for Ambient Intelligence Applications" in IEEE Trans. on Multimedia, v. 11, n. 9 (2009), pp. 1266-1275.

International Conferences and Workshops

C1 - N. Piotto, L. Wang, N. Conci, F.G.B. De Natale, D. Schonfeld, "Optimal Quantization for Robust Trajectory Analysis of Video Sequences". IEEE Int'l Conf. on Image Processing, 2011, submitted.

C2 - M. Daldoss, N. Piotto, N. Conci, F.G.B. De Natale, "Learning and Matching Human Activities using Regular Expressions". IEEE Int'l Conf. on Image Processing, pp. 1-4, 2010.

C3 - M. Daldoss, N. Piotto, N. Conci, F.G.B. De Natale, "Activity Detection using Regular Expressions". IEEE Int'l Workshop on Image Analysis for Multimedia Interactive Services, pp. 1-4, 2010.

C4 - G. Boato, N. Conci, M. Daldoss, F.G.B. De Natale, N. Piotto, "Hand tracking and trajectory analysis for physical rehabilitation". IEEE Int'l Workshop on Multimedia Signal Processing, pp. 1-6, 2009.

C5 - N. Piotto, F.G.B. De Natale, N. Conci, "Hierarchical Matching of 3D Pedestrian Trajectories for Surveillance Applications". IEEE Int'l Conf. on Advanced Video and Signal Based Surveillance, pp. 146-151, 2009.

C6 - N. Piotto, N. Conci, F.G.B. De Natale, "Syntactic Matching of Pedestrian Trajectories for Behavioral Analysis". IEEE Int'l Workshop on Multimedia Signal Processing, pp. 877 - 882, 2008.