Universitat Autònoma de Barcelona

Modeling Human Behavior for Image Sequence Understanding and Generation

A dissertation submitted by **Pau Baiget Arasanz** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor en Informàtica**.

Bellaterra, May, 2009

Director: **Dr. Jordi Gonzàlez i Sabaté**
Computer Vision Center, Universitat Autònoma de Barcelona.
Dept. de Ciències de la Computació, Universitat Autònoma de Barcelona.
Co-director: **Dr. Xavier Roca i Marvà**
Computer Vision Center, Universitat Autònoma de Barcelona.
Dept. de Ciències de la Computació, Universitat Autònoma de Barcelona.

Centre de Visió per Computador

This document was typeset by the author using LaTeX 2_ε.

The research described in this book was carried out at the Computer Vision Center, Universitat Autònoma de Barcelona.

ISBN 84–922529–8–7

Printed by Ediciones Gráficas Rey, S.L.

God grant me the serenity
To accept the problems that I cannot solve
The persistence to solve the problems that I can
And the wisdom to know the difference

Shahriar Manzoor

You can get it if you really want,
you can get it if you really want,
but you must try, try and try,
try and try . . .
you'll succeed at last

Jimmy Cliff

Acknowledgment

Sembla que per fí s'acaba. Qui em coneix sabrà que una bona part dels darrers dos anys he estat esperant aquest moment. Això no vol dir que ho hagi passat malament, simplement patia pensant que no hi arribaria. Per sort m'he rodejat d'un conjunt excepcional de persones, que m'han fet molt fàcil aquesta etapa i a les quals vull dedicar unes paraules tot seguit.

En primer lloc, voldria agraïr a en Poal la seva múltiple tasca com a director de tesi, psicòleg i amic. La seva persistència ha fet crèixer en mi, finalment, la certesa de que la meva feina valia alguna cosa, sobretot en els meus pitjors moments. De no ser per ell potser no estaria escrivint aquestes línies. També voldria agraïr especialment a en Juanjo Villanueva i a en Xavi Roca, per la oportunitat de treballar en aquest magnífic entorn i per les converses motivadores que hem mantingut en aquest temps.

La meva etapa com a doctorand no hagués estat la mateixa sense els meus companys de viatge, en Carles i l'Ivan. Amb ells he desenvolupat el gruix de la meva Tesi, hem escrit junts els artícles i hem conviscut per mitja Europa. D'aquí a poc els tocarà a ells escriure les paraules que estic escrivint jo ara. Ànim!

Voldria també recordar a la resta de components d'ISE, que ha anat creixent i durant la meva etapa s'ha consolidat com un important grup de recerca. A Dani i Ignasi, els veterans del grup, que ens van ensenyar de què anava això de fer el doctorat. A en Bhaskar, Marco, Ariel, Murad i Ogi, que han compartit amb mi els meus darrers anys i que ara seràn el referent del grup. Als enginyers del grup, en Miguel i en Joan Soto, que s'ho han currat per fer visible el nostre tinglado. Per últim, les noves incorporacions d'en Pep, Noha i Wenjuan, que tiren molt fort!!

És difícil trobar un entorn de treball on pots ser company i amic de tothom. En primer lloc, els que han fet part del viatge amb mi, i que hem compartit docència, cafés, recerca i bones birres!! Especialment vull recordar en Xevi, Aura, Marçal, Alicia, Sergio, Eduard, Ágata, David, Jaume, Agnés, Debora i Joan Mas. També vull tenir un record per a la resta de companys del CVC, que m'han facilitat extremadament la feina, especialment la Mari, Montse, Raquel, Gigi, Joan i Raquel. A tots els demés, moltes gràcies també.

També vull agrair la comprensió als meus amics Manel, Lucas, Albert, Lou, Creus, Griselda, Jose i d'altres que no caben aquí. M'han donat el suport incondicional que em calia per superar algunes hores baixes, i estar finalment aquí.

I would like to also thank to several special persons I have met during my PhD. First, I want to thank Prof. H.-H. Nagel and Dr. Ian Reid for accepting and driving my stages in Karlsruhe and Oxford. Also, I want to thank Dr. Dimitrios Makris for

sharing me his Dataset, which I used a lot for my experiments. Finally, I would like to thank my partners Eric Sommerlade, Hanno Harland, Daniel Roth and others, for these great discussions and beers all over Europe.

Vull tenir un agraïment especial per a la Vanessa, a la que vaig conèixer dins el CVC. Només per això ja fa que el doctorat hagi valgut la pena. Des d'aleshores ha estat al meu costat en tot moment i, malgrat veure'm de tots els colors, encara vol estar amb mi.

Finalment, vull dedicar aquest treball als meus pares i a la meva família, als quals no els importa res més que la meva felicitat, i ho he notat durant tots aquests anys.

Salut a tothom!!

Abstract

The comprehension of animal behavior, especially human behavior, is one of the most ancient and studied problems since the beginning of civilization. The big list of factors that interact to determine a person action require the collaboration of different disciplines, such as psichology, biology, or sociology. In the last years the analysis of human behavior has received great attention also from the computer vision community, given the latest advances in the acquisition of human motion data from image sequences.

Despite the increasing availability of that data, there still exists a gap towards obtaining a conceptual representation of the obtained observations. Human behavior analysis is based on a qualitative interpretation of the results, and therefore the assignment of concepts to quantitative data is linked to a certain ambiguity.

This Thesis tackles the problem of obtaining a proper representation of human behavior in the contexts of computer vision and animation. On the one hand, a good behavior model should permit the recognition and explanation the observed activity in image sequences. On the other hand, such a model must allow the generation of new synthetic instances, which model the behavior of virtual agents.

First, we propose methods to automatically learn the models from observations. Given a set of quantitative results output by a vision system, a *normal* behavior model is learnt. This results provides a tool to determine the normality or abnormality of future observations. However, machine learning methods are unable to provide a richer description of the observations. We confront this problem by means of a new method that incorporates prior knowledge about the enviornment and about the expected behaviors. This framework, formed by the reasoning engine FMTL and the modeling tool SGT allows the generation of conceptual descriptions of activity in new image sequences. Finally, we demonstrate the suitability of the proposed framework to simulate behavior of virtual agents, which are introduced into real image sequences and interact with observed real agents, thereby easing the generation of augmented reality sequences.

The set of approaches presented in this Thesis has a growing set of potential applications. The analysis and description of behavior in image sequences has its principal application in the domain of smart video–surveillance, in order to detect suspicious or dangerous behaviors. Other applications include automatic sport commentaries, elderly monitoring, road traffic analysis, and the development of semantic video search engines. Alternatively, behavioral virtual agents allow to simulate accurate real situations, such as fires or crowds. Moreover, the inclusion of virtual agents into real

image sequences has been widely deployed in the games and cinema industries.

Resum

La comprensió del comportament dels animals, i en especial dels humans, és un dels problemes més antics i estudiats al llarg de la història, quasi des de l'inici de la civilització. La quantitat de factors diferents que actuen alhora de determinar les accions d'una persona requereixen la participació de diferents disciplines, com la psicología, biología, o sociología. En els darrers anys l'anàlisi del comportament humà ha esdevingut també un tema molt interessant per a la comunitat científica de visió per computador, gràcies als darrers avenços en l'adquisició de dades sobre el moviment humà a partir de seqüències d'imatges.

Malgrat la creixent disponibilitat d'aquestes dades, existeix encara una barrera per obtenir una representació conceptual de les observacions obtingudes. L'avaluació del comportament humà en seqüències d'imatges està basat en una interpretació qualitativa dels resultats, i per tant l'assignació de conceptes a les dades quantitatives obtingudes està lligada a una certa ambigüetat.

Aquesta Tesi confronta el problema d'obtenir una representació correcta del comportament humà en els contextes de la visió i animació per computador. En primer lloc, un bon model de comportament ha de permetre reconèixer i descriure l'activitat observada en seqüències d'imatges. D'altra banda, el model ha de permetre generar sintèticament noves instàncies, que permetin modelar el comportament d'agents virtuals.

En primer lloc, proposem mètodes per aprendre els models directament de les observacions. A partir de les dades obtingudes mitjançant l'anàlisi de seqüències d'imatges, construim models de comportament *normal* dins l'escenari. Això ens proporciona una eina per determinar la normalitat o anormalitat de futures observacions. Tanmateix, els mètodes d'aprenentatge automàtic son incapaços de proveïr una descripció semàntica de les observacions. Aquesta problema és tractat mitjançant un nou mètode que incorpora un coneixement a–priori sobre l'escena i els comportaments esperats. Aquesta estructura, formada pel motor de raonament difús FMTL i l'eina de modelatge SGT, permet obtenir descripcions conceptuals del contingut de noves seqüències de video. Finalment, l'estructura FMTL + SGT ens permet simular comportament sintètic i introduïr agents virtuals dins d'escenes reals que interactuen amb els agents reals existents, construïnt d'aquesta manera seqüències de realitat augmentada.

El conjunt de mètodes presentats en aquesta Tesi tenen un conjunt potencial d'aplicacions cada cop més gran. Per un costat, el reconeixement i descripció de comportament en seqüències d'imatges té com a principal aplicació la video–vigilància

intel.ligent, permetent detectar comportaments delictius o perillosos. Altres aplicacions inclouen la transcripció d'esdeveniments esportius, monitorització de centres geriàtrics, anàlisi de tràfic en carreteres i la construcció de buscadors de video basats en conceptes semàntics. D'altra banda, l'animació d'agents virtuals amb comportaments complexes permet obtenir simulacions acurades de situacions reals, com per exemple incendis o multituds. A més, la inclusió d'agents virtuals en entorns reals té forta implantació en els mons dels videojocs i el cinema.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Almost since the beginning of the human civilization and the rational thinking, one of the most studied problems has been to understand the behavior of everything around human beings. In the first place, the attention was focused on the most trascendent physical phenomena, e.g the meaning of days and nights, the fire, the sun, etc. Those were initially associated to divine action. However, the pass of centuries has led the cilivilization to grow scientifically and to leave the initial dogmatism. Thus, some of these physical phenomena have become explained by physics law or by mathematic theorems. The contributions of Newton, Einstein, and Euler, among others, constituted a great help towards understanding accurately how non–living objects behave inside the space and the consequences of their interactions.

Regarding living creatures, the existence of consciousness and emotions has added complexity in the task of understanding their actions. Hence, besides the study of their physical properties, efforts have been devoted to understand what are the internal impulses that make them acting in a particular fashion. Non–rational animals (which at the moment are all the living creatures except humans) are mostly driven by their survival instinct. Roughly speaking, survival instinct is a kind of internal memory containing a serie of actions, which will be performed by animals in order to feed, reproduce and defense. According to the Theory of Evolution of Species, the biological properties and actions of animals have evolved over time in order to adapt themselves to their environment. Moreover, animals experiment a learning process during their lives, which enriches their behavior.

In the particular case of human beings, their capacity for the abstract thinking, language communication, and action planning differentiates them from the rest of animals. On the one hand, abstract thinking allows humans to think and reason about non–visible concepts and to create complex representations of the world. On the other hand, humans can planify long–term actions and estimate their consequences beforehand. These two features produces that the number and granularity of behaviors that can be performed by human beings increases exponentially the complexity of its recognition and, especially, its prediction.

A novel interdisciplinary domain which aims to emulate some of the previously mentioned capabilities has raised within Computer Science in the last decades [69].

It comprehends techniques of Image Processing and Analysis, Pattern Recognition, Artificial Intelligence, Computer Graphics, and Robotics, among others. This new domain analyzes and evaluates sequences of images concerning human-populated scenes. Impressive developments have also been possible thanks to a large number of technological advances in the hardware field. Emerging capabilities have led to a wide range of scientific contribution, and, subsequently, to new software implementations.

1.1 Motivation

Behavior analysis in dynamic scenes is a complex task, specially when it concerns human-populated environments. Trying to emulate the astonishing performances of such a perfect system as the Natural Vision System represents, without any doubt, a real challenge.

The reasoning task is even more complicated when it deals with human beings, thereby making it particularly appealing. In spite of the numerous difficulties involved —or perhaps, because of them— behavior analysis in human-populated scenes has become a very active research field: it has already generated a vast number of scientific contributions in recent years [69]. However, despite this interest and the substantial developments achieved, this still constitutes an ambitious open problem which is far from being solved.

Further, this interest is also prompted by the increasing number of potential applications. These include smart video safety and video surveillance, automatic sport-statistics computation, intelligent human-computer interfaces, machine content annotation, or efficient athlete training and orthopedic therapy, among others. Thus, the numerous promising applications constitute an important practical motivation which raises significant funds for this field of research.

Recent developments in Human-Sequence Evaluation have made possible to consider a huge number of promising applications. Moreover, the benefits that can be obtained from these applications are promoting research in this particular computer vision area.

These applications can be classified according to their aims. A division between analysis and synthesis applications is here considered. The former attempts to process an input video signal, whereas the goal of the latter is to generate synthetic scenes, agents, and their motion.

An important application of high-level event recognition in image sequences is smart video–surveillance, which increases the security agains vandalism, thefts or terrorism, see Fig. 1.1.(a) and (b). The recognition of events in traffic environments allows to acquire information about the state of the roads, traffic holdups and accidents, see Fig. 1.1.(c). Moreover, smart video safety could assist remote elderly care [109], see Fig. 1.1.(d). An application into controlled environments can lead to a complete recognition of observed activity, e.g. automatic description of sport events, see Fig. 1.1.(d). Finally, event recognition in video is receiving important attention towards the creation of a semantic–based video indexing engine. In such a system, the user can search for a video based on the expected activity, see Fig. 1.1.(f).

Alternatively, the generation of synthetic behavior in virtual enviornments allows

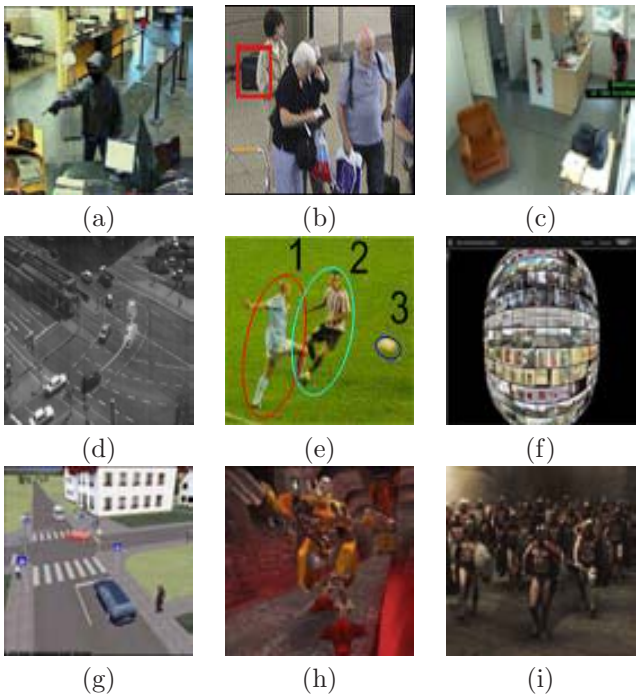


Figure 1.1: Examples of applications. (a) Detection of dangerous events (b) Left baggage detection (c) Elderly care monitoring (extracted from [109]) (d) Traffic monitoring [49] (e) Automatic sport commentaries (f) Semantic video indexing [54] (g) Complex Event Simulation (h) Generation of virtual characters in games, extracted from [91] (i) Simulation of realistic crowds for special effects in movies, extracted from [92].

the simulation of complex activity in selected scenarios, e.g. traffic simulation in Fig. 1.1.(g). The entertainment industry has also shown great interest in the creation of special effects in movies, see Fig. 1.1.(h) and (i).

This Thesis focuses on obtaining proper models to recognize, interpret, and reproduce behaviors in order to understand and generate image sequences. The approaches and methods that will be described along next chapters are intended to apply for any kind of subject: animals, objects or physical phenomena for which there is a physical or mathematical law explaining their behavior. Nevertheless, we have chosen human beings given that they represent the highest level of complexity, as explained above.

1.2 Definitions of Behavior in Computer Vision

The general concept *behavior* is the reaction of human beings to a set of external and internal impulses. These impulses may be caused by very different sources, and only the combination of them can represent an objective explanation of the observed behavior.

From a biological point of view, the elements that affect and modify human behavior are those related to the instincts carried from its animal nature, and can be generalized with the self–preservation principle. Such instinct enables the actions that must be taken in order to stay alive, for instance, self–feeding, reproduction, defense, etc.

The psichological aspect is also another source of impulses that severely affect to the performance of human behavior. Roughly speaking, these impulses would be the answer to questions like: *what is the person A worried about?* and *how is person A feeling?*. Obviously, the internal mood of a human being directly conditions on its behavior, and even in the intensity of the chosen one. Moreover, human personality changes over time and depending on the environment. A more sophisticated question would be *how is person A feeling, at this precise moment, being in this place, and with that partners?*.

Other different factors that condition on human behavior are briefly summarized next. First, physical properties of human beings determine a correct interpretation of their behavior. For instance, a human being moving may be considered to be *walking* or *running*, depending on whether it is an old or young person. Second, the role of the person in the environment has also a critical correlation with the resulting behavior. This role can be general (for instance, being a policeman) or particular (e.g. the father–son role).

The principal aim of this Thesis is to represent human behavior using the features estimated by state–of–the–art computer vision systems. This task must be accomplished in a consistent manner, so that the result can be used to explain new observations and to generate valid synthetic instances. Such representation is dependant on qualitative and quantitative knowledge about specific human features and also on the scenario conditions. These features are classified into four groups:

1. Knowledge about human motion. This group includes the features of humans as moving objects inside a scenario. Examples are *speed* and *orientation*. These values may show a different distribution in different scenario locations, e.g. roads or sidewalks.

2. Knowledge about human actions. A human agent is represented by a body model and thus features consist of sequences of movements of the components. For instance, the *walking* action is a cyclic action which involves the movement of particular body parts, which differentiates from other actions like *bending* or *jumping*.

3. Knowledge about human interactions. Human agents can hardly behave without considering their environment. Such environment consists of all possible factors that can affect human behavior. Indeed, the interaction with either other human agents or objects from the scenario determines future agent reactions.

4. Knowledge about human emotions. The internal mood of an agent conditions not only on its behavior but also on the intensity of that behavior.

Nowadays, the main challenge in the state–of–the–art is how to provide this required knowledge. Recently, the focus has been set on maximizing the amount of information that can be automatically acquired from image sequences.

Figure 1.2: Open issues in multiple–target tracking [88]. (a) Cluttered environments (b) Camouflage with the background (c) Illumination problems (d) Crowds of people (e) Unknown shape of targets (f) Mutual occlusions between targets

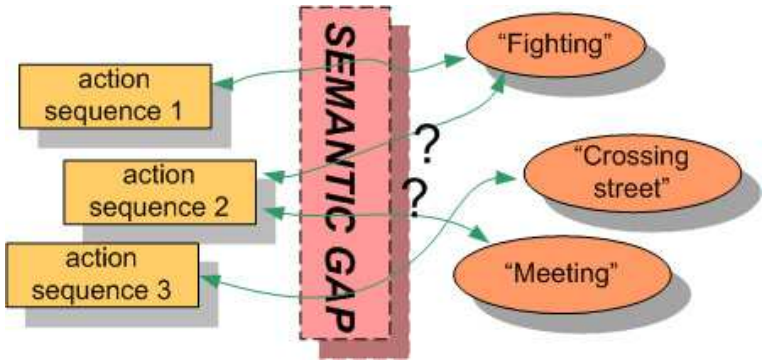
1.3 Inherent difficulties

The task of interpreting the events occurred 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 step to convert quantitative data into concepts has to be considered between the vision–based techniques and the behavior interpretation.

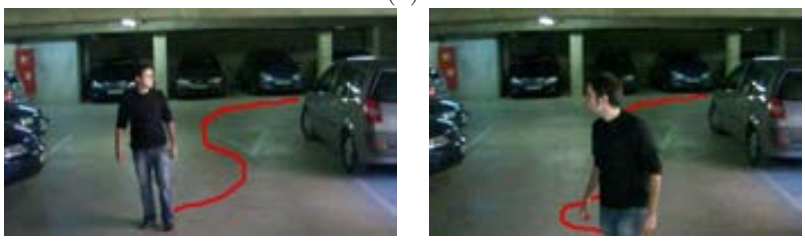
The acquisition of qualitative knowledge from image sequences involves two steps which can induce uncertainty to the final image sequence interpretation. As a result of integration, numerical information obtained from image sequences can be used to instantiate qualitative predicates and, the other way around, conceptual knowledge can be used to assist pattern analysis processes [72].

On the one hand, the error pulled from the computer vision subsystem caused by the *sensory gap*, which refers to the lack of accuracy in the quantitative data acquisition from the image sequences. There still exists several issues open in the vision–based estimation of motion in image sequences. Fig. 1.2 depicts the current problems that are still unsolved. For instance, a cluttered or poorly illuminated environment difficulties distinguishing the targets from the background, thus obtaining a wrong estimation of their position and obviously of their activity. Moreover, multiple targets produce mutual occlusions with respect to the point of view, preventing an accurate acquisition of their silhouettes.

On the other hand, human behavior modeling has to deal with the uncertainty due to the *semantic gap*, which refers to the conceptual ambiguity between the image



(a)



(b)

Figure 1.3: (a) The semantic gap between the observations (quantitative) and the concepts (qualitative). (b) Example of semantic gap. Given the red agent trajectory, two possible interpretations can be done 1) The agent is searching his parked car (considered as *normal* behavior), and 2) The agent is trying to steal a car (*suspicious* behavior). Given that situation, no possible reactions can be done until further information is obtained or provided beforehand.

sequence and its possible interpretations. In our context, the semantic gap is present in both learning and recognition of behaviors, see Fig. 1.3.(a). On the one hand, although a computer can learn a set of behavior patterns given some training data, it is unable to extract a semantic explanation of the results. Indeed, it is not possible to assert if some of the learnt patterns represent a *normal* or an *anomalous* behavior unless the training set has been previously labeled. On the other hand, behavior recognition consists of matching an observed action sequence with a set of behavior patterns. In this case, the inaccuracy pulled from the vision system and the ambiguity of possible interpretations, see Fig. 1.3.(b), can lead to a misclassification of the input action sequence. Thus, *Uncertainty* arises because of the difficulty of modeling all the possible human behaviors in a given discourse domain. Therefore, uncertainty has to be considered, so logic predicates, which represent knowledge acquired from image sequences, have to be accurately designed in order to fulfill all possible situations, otherwise ambiguity could make the system to misinterpret the input image sequence. For example, an agent walking in a parking lot moving his face back and forth can be interpretated as a person searching his parked car or a thief deciding which car to steal.

1.4 Towards a Cognitive Vision System

Due to the complexity of the proposed goal, such a kind of system must be designed in a modular structure. In this case, we obey the modular scheme of *Human Sequence Evaluation*(HSE) [44], which is shown in Figure 1.4. The modules in this scheme are linked with bidirectional arrows, which means that data can flow in both directions. This Thesis considers both data flows. On the one hand, the Bottom-up data flow goes from the lowest level, an image sequence, to a high level representation that can be sent to a user interface, e.g. a natural language text description. This tranformation embeds two separate tasks: first, to obtain a geometric description of the scene in terms of quantitative values (the Visual Subsystem), and second, to associate the geometric description with qualitative concepts within a proper knowledge representation for conceptual description (Conceptual Subsystem).

A brief description of the HSE modules is provided next:

- *Active Sensor Level*. It represents the camera system. Provides the lowest level information, which is the recorded image sequence.
- *Image Signal Level*. This level contains two low level processes: The human agent detection and segmentation.
- *Picture Domain Level*. In this level 2D representations of the human agent are calculated based on the results of the segmentation done in the ISL.
- *Scene Domain Level*. This level represents the process followed to obtain 3D agent coordinates from the 2D coordinates obtained in the Picture Domain Level. This process requires a camera calibration, which needs information about the camera model in the Active Sensor Level.
- *Conceptual Integration Level*. At this level, geometrical information obtained from the previous levels is integrated into conceptual predicates, based on fuzzy logic.
- *Behavior Interpretation Level*. The conceptual knowledge obtained in the previous level is organized in order to classify behavior patterns observed in the input image sequence. In this work the deterministic formalism Situation Graph Tree is used to model human behavior and to extract conceptual descriptions, which are high level semantic predicates.
- *User Interface Level*. This level allows users to interact with the system, providing the results obtained in the previous levels in a high level of abstraction like a natural language generated text or a synthetic image sequence which describes the image sequence recorded in the Active Sensor Level.

In this Thesis, we focus on the ultimate aim of this novel domain: the cognitive stage of HSE, which refers to the linkage between the quantitative information that can be obtained by a computer vision system, with a conceptual representation of the world. Such models are intended to provide an explanation of *what is going on* in an image sequence. The term *Human-Sequence Evaluation* (HSE) denotes the analysis

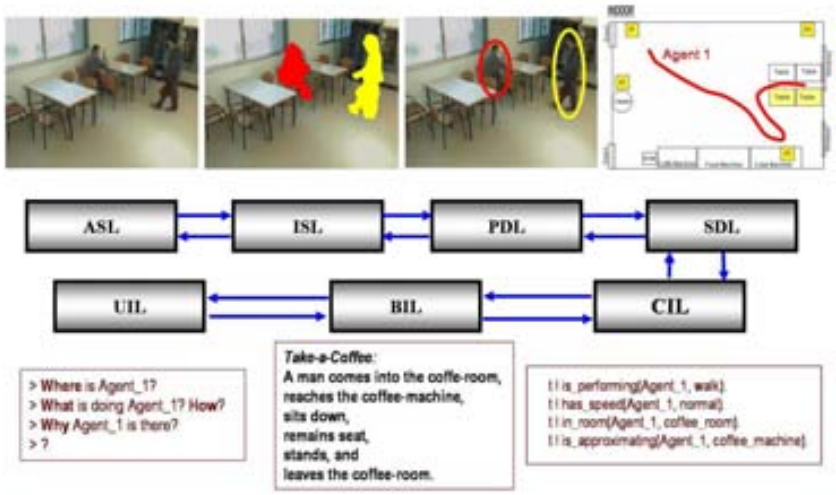


Figure 1.4: The *Human Sequence Evaluation* scheme. This work is enclosed between the modules CIL and BIL. More details in the text.

of human motion in order to achieve the understanding of human behaviour, that is, the explanation and reasoning about *why* motion is performed. Further, it would be able to provide conceptual scene descriptions, and to generate synthetic views of the environment in order to (i) visualize recognised behaviours, and (ii) simulate potential situations. Therefore, HSE defines an extensive *Cognitive Vision System* (CVS) which transforms acquired image values into semantic descriptions of human behaviour and synthetic visual representations. Hence, HSE represents a huge challenge in which the aim is to emulate the fascinating performances of a Natural Vision System, and the reasoning and communication skills of a human observer.

Towards this end, we confront two alternative and complementary paradigms, which are standard methods for modeling real observable phenomena. These are, namely, bottom–up and top–down approaches, see Fig. 1.5. On the one hand, bottom–up behavior modeling is based on automatically constructing models from observed events in large recordings. Alternatively, top–down techniques make use of prior knowledge to deterministically design a selected set of behavior models. Along this document we present approaches for this two paradigms and we discuss and justify the benefits and drawbacks that are obtained with each method. Bottom–up approaches can only establish a similarity between learnt behavior models and the observations, since they lack of a proper semantic layer. On the other hand, Top–down approaches enable the use of semantic information to model behaviors in a selected context, allowing to extract conceptual representations of the results. Then, we use the obtained behavior models to recognize and describe the quantitative information obtained from vision–based techniques.

However, a desirable feature of a modeling formalism is that model instances can

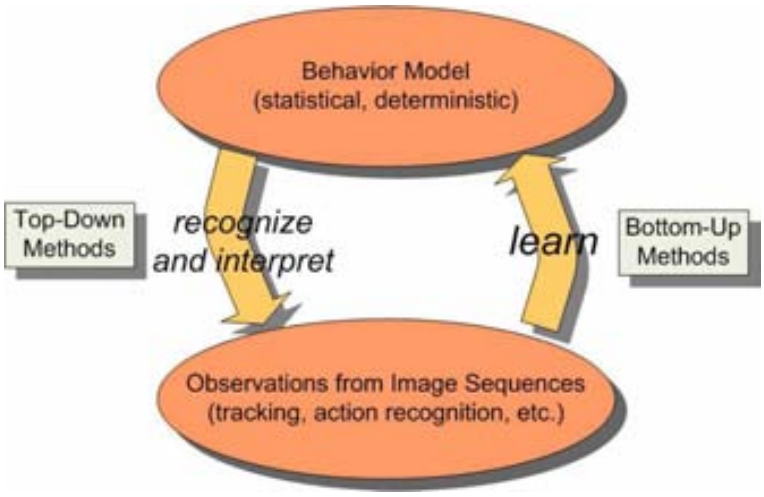


Figure 1.5: Top–down and Bottom–up paradigms in the task of behavior modeling. Top–down approaches make use of precomputed behavior models to describe the observations, while Bottom–up approaches analyze these observations to learn the models.

not only be represented and recognized, but also synthetically generated. In the case of behavior modeling, this implies the creation of synthetic behavior instances. We demonstrate the feasibility of our proposed Top–down behavior model towards the creation of virtual environments. A suitable motion model is learnt for human beings, which allows to differentiate between motion actions (e.g. walking or running) and the conceptual descriptions are used to generate quantitative information, just the other way around as in the recognition process.

Finally, we apply our behavior modeling framework to generate Augmented Reality sequences, involving real and virtual characters. Augmented Reality (AR) is a growing area in virtual reality research [78, 79, 55]. AR combines a real scene viewed by the user and a computer–generated virtual scene, which enriches the original scene with additional information. Most of the augmented reality applications are primarily designed for letting a user browsing a 3D virtual world registered with the real world. Nowadays, the generation of augmented sequences through the animation of virtual characters in real scenarios has raised as a promising challenge, with wide applications in the game industry [85] and education [53].

1.5 Basic Concepts

In this section we briefly introduce the definition of the basic concepts that are present along this Thesis.

1.5.1 Agent

Any entity present within the scene which could be subject to special interest and, consequently subject to be detected and tracked, is called a *target*. Further, any target with intentional capabilities is referred as an *agent*. Depending on the application, the term may include people, vehicles, or even animals.

In this Thesis we distinguish between real and virtual agents. Real agents are those passive entities present in image sequences whose motion is observed by some computer vision system. Alternatively, virtual agents are user–designed agents that perform synthetic instances of behavior in a controlled environment.

1.5.2 Scenario

In the following sections, the words *scenario* and *scene* will be used to reference the environment where activity is supposed to take place. In order to be interesting in our context, a valid scenario should allow a particular human agent to develop at least one of the previously enumerated behavior types.

Considering the context of video surveillance, scenario is primarily represented by a finite set of reference images, taken by a single or multiple camera system. Thus, a straightforward way to represent the existence of objects and agents within a scenario is specifying their location with respect to the coordinate system of the reference image (this will be called *image plane* coordinates in the following). Hence, the agent position is represented using a specific pair of pixels (i, j) of the image. Any further reasoning must consider the projective distortion of agent locations given the camera position. The image–plane representation does not provide a proper description of the scenario configuration, since real distances can not be a–priori measured. In essence, the real distance between two regions at locations (i_1, j_1) and (i_2, j_2) is completely dependant on the image–plane.

An alternative representation of the scenario is expressed in *real–world* coordinates, being those independent from the point of view. Such coordinate system is obtained by mapping pixel locations into a ground–plane, assumed to be within the scenario.

Surveillance scenarios represent *outdoor* or *indoor* environments, and each type entail different kinds of behaviors to be analyzed. On the one hand, outdoor scenarios represent unconstrained open areas, where agents have freedom to move back and forth. Since cameras cover a wide region, agents occupy a narrow part of the image, difficulting the analysis of body action or face expression. In that scenarios, analysis is focused on the agent silhouette and trajectory. Nevertheless, indoor scenarios typically represent small environments, e.g. rooms, where agent mobility is sometimes limited. Also, indoor environments contain a richer set of objects that the agents interact to. Cameras usually capture a close–up of agents, allowing the analysis of face and gestures.

It is important to distinguish different semantic regions in the scenario, since this conditions on final human behavior. For instance, in an urban scenario, human agents are expected to behave different when they walk within a crosswalk or a sidewalk. Following chapters address the challenge of obtaining a proper conceptual scenario model, either by learning from observations or by establishing prior knowledge.

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

Figure 1.6: Different data sources for behavior analysis. (a) Multi–object tracking (b) Body action recognition (c) Face expression recognition (d) Head pose estimation

1.5.3 Data from Vision Systems

The detection of human features in images has concerned a great part of the vision community in the last decade. The exponential enhancement of hardware performance and the implementation of new algorithms has led to perform a more efficient analysis of images. These human features are searched in order to provide a representation of the activity observed in an image, and try to answer the questions described in previous section. Nowadays, human feature detection has become a wide general topic, thereby being divided into several research lines, see Fig. 1.6:

- *Multi–object Tracking* [106]. The concept *tracking* is defined as the process of establishing coherent relations amont *targets* between frames; or as inferring the *agent status* over time using all evidence up to date [88].
- *Body Action Recognition* [17, 102, 48]. The recognition of human body actions enhances the granularity of behaviors but entails a high degree of complexity due to self–occlusions and pose variability.
- *Facial Expression Recognition* [63, 32]. Facial expression analysis provides a good estimation of the mood of the agent. Such an information can ease explanations of global agent behavior.
- *Head Pose Estimation* [15]. The estimation of head pose approximates the gaze of the agent at a given frame step.

1.5.4 Agent Status

Given that time information is discretized into frame steps in the image sequence, the status of an agent, or *agent status*, is defined as the parametrized knowledge wich characterizes the target evolution, i.e, all the information obtained from a single agent at a certain time step.

1.6 Approaches and Contributions

This chapter summarizes the approaches and subsequent contributions described in this document. A hierarchical scheme of this section can be seen in Fig. 1.7. There,

gray ellipses represent processes, gray boxes represent the data and blue boxes represent the final applications of each contribution.

- In this thesis, we focus on developing behavior models for the analysis and generation of image sequences. First chapters are devoted to analyze the observed behaviors in video–surveillance scenarios. To this end, two different approaches for behavior analysis in images sequences are described and confronted.

- First, we tackle the challenge of learning behavior models from the observation of activity in surveillance scenarios. We propose a statistical method to learn motion patterns and to build up the *normal* behavior model in particular regions of the scenario. Given that result, we are able to differentiate between normal and abnormal behavior in future observations. Subsequently, we can predict the development of incoming observations by assuming that the performed behavior is normal. Finally, we can sample the constructed models in order to generate synthetic normal instances of behavior.

- The unsupervised learning of behavior models does not take care of the inherent semantics of human behavior. In order to tackle this issue, we propose to use a deterministic top–down approach for the analysis of human behavior which takes advantage of prior semantic knowledge about the enviornment and about the behaviors that we expect to observe there. We use the framework formed by the reasoning formalism called Fuzzy Metric Temporal Logic and the behavior modeling formalism called Situation Graph Tree for the recognition and description of a predefined set of human behaviors in outdoor scenarios. The flexibility of the proposed approach is shown by applying the framework into different discourse domains: video–surveillance and sport shows.

- We exploit the reaction capabilities of the SGT formalism to generate synthetic instances of the modeled behaviors and thus simulate synthetic agents in virtual environments. We present a robust framework to simulate the development of virtual agents that interact with each other and show complex behaviors in a selected environment. By combining the SGTs for recognition with the simulation ones, we can obtain interactions between real and virtual agents. This leads to the generation of augmented reality (AR) sequences in where real and virtual agents exist in a consistent manner.

- The unified approach to produce AR sequences is tested on two application domains. First, we describe a virtual storytelling system that converts a natural language explanation into its representation in images. A natural language understanding (NLU) module is used to convert sentences into predicates that represent goals in virtual agent behavior. Through the acchievement of these goals, a virtual story is generated. Second, we face the problem of tracking evaluation by providing a mechanism to gradually increase the complexity of a given scenario. This is achieved with the addition of behavior–based virtual agents into a real scenario, thereby generating new targets to be observed by the tracker. Since being synthetically generated, these new agents automatically generate the ground–truth to be compared against the tracking results.

Summarizing, in this Thesis we explore the current open issues in the long path towards behavior analysis in image sequences and we have proposed a contribution for each of them. However, the unavoidable influence of semantics in human behavior makes statistical approaches limited to provide an objective explanation in an uncontrolled environment. On the other hand, explicit behavior models depend on the expertise of the human designer. Concluding, there is still a long ground towards having a unified solution to the overall problem.

1.7 Document Outline

The remaining of the document is structured as follows. Next chapter reviews the current state of the art in the topics related to this work, namely behavior analysis in image sequences, behavior modeling for virtual agents and augmented reality. A brief review of the applications of the proposed approaches is also added at the end of the chapter.

Next, Chapter 3 introduces two approaches for the automatic learning of motion patterns in surveillance scenarios from computer vision outputs. The main difference between the two methods is the way the incoming data is received. Section 3.2 proposes to on–line process observations as they are estimated by the vision system, and Section 3.3 presents an offline version which exploits the advantage of having complete observations to learn typical entry and exit areas, thereby creating common paths in the scenario. Section 3.4 applies the latter approach to detect anomalies in image sequences and, finally, Section 3.5 compares the two proposed methods and justifies ther advantages and drawbacks.

The use of semantics is exploited in the approach presented in Chapter 4. There, we describe the framework formed by a fuzzy temporal reasoning engine (FMTL) and a behavior modeling tool (SGTs), used to recognize, interpret and describe a selected set of behaviors in image sequences. The approach is tested over image sequences from two different discourse domains and the benefits and drawbacks are discussed at the end of the chapter.

Chapter 5 extends the previous framework towards the generation of synthetic behavioral agents in the simulation of virtual environments. Then, Section 5.2 describes the combination of real and virtual agents into real environments, thereby producing augmented reality sequences. The subsequent experiments demonstrate the feasibility to build interactions between real and virtual agents and the consistent adaptation of virtual agents into real environments. Sections 5.4 and 5.5 show the application of the presented approach into tracking performance evaluation and virtual storytelling, respectively.

Finally, Chapter 6 presents a general discussion about the approaches and results obtained in this Thesis. For each topic related to the presented contributions we point out the remaining open issues and future directions of research.

1.8 Resum

La comprensió del comportament dels animals, i en especial dels humans, és un dels problemes més antics i estudiats al llarg de la història, quasi des de l'inici de la civilització. La quantitat de factors diferents que actuen alhora de determinar les accions d'una persona requereixen la participació de diferents disciplines, com la psicología, biología, o sociología. En els darrers anys l'anàlisi del comportament humà ha esdevingut també un tema molt interessant per a la comunitat científica de visió per computador, gràcies als darrers avenços en l'adquisició de dades sobre el moviment humà a partir de seqüències d'imatges.

Malgrat la creixent disponibilitat d'aquestes dades, existeix encara una barrera per obtenir una representació conceptual de les observacions obtingudes. L'avaluació del comportament humà en seqüències d'imatges està basat en una interpretació qualitativa dels resultats, i per tant l'assignació de conceptes a les dades quantitatives obtingudes està lligada a una certa ambigüetat.

Aquesta Tesi confronta el problema d'obtenir una representació correcta del comportament humà en els contextes de la visió i animació per computador. En primer lloc, un bon model de comportament ha de permetre reconèixer i descriure l'activitat observada en seqüències d'imatges. D'altra banda, el model ha de permetre generar sintèticament noves instàncies, que permetin modelar el comportament d'agents virtuals.

En primer lloc, proposem mètodes per aprendre els models directament de les observacions. A partir de les dades obtingudes mitjançant l'anàlisi de seqüències d'imatges, construim models de comportament *normal* dins l'escenari. Això ens proporciona una eina per determinar la normalitat o anormalitat de futures observacions. Tanmateix, els mètodes d'aprenentatge automàtic son incapaços de proveïr una descripció semàntica de les observacions. Aquesta problema és tractat mitjançant un nou mètode que incorpora un coneixement a–priori sobre l'escena i els comportaments esperats. Aquesta estructura, formada pel motor de raonament difús FMTL i l'eina de modelatge SGT, permet obtenir descripcions conceptuals del contingut de noves seqüències de video. Finalment, l'estructura FMTL + SGT ens permet simular comportament sintètic i introduïr agents virtuals dins d'escenes reals que interactuen amb els agents reals existents, construïnt d'aquesta manera seqüències de realitat augmentada.

El conjunt de mètodes presentats en aquesta Tesi tenen un conjunt potencial d'aplicacions cada cop més gran. Per un costat, el reconeixement i descripció de comportament en seqüències d'imatges té com a principal aplicació la video–vigilància intel.ligent, permetent detectar comportaments delictius o perillosos. Altres aplicacions inclouen la transcripció d'esdeveniments esportius, monitorització de centres geriàtrics, anàlisi de tràfic en carreteres i la construcció de buscadors de video basats en conceptes semàntics. D'altra banda, l'animació d'agents virtuals amb comportaments complexes permet obtenir simulacions acurades de situacions reals, com per exemple incendis o multituds. A més, la inclusió d'agents virtuals en entorns reals té forta implantació en els mons dels videojocs i el cinema.

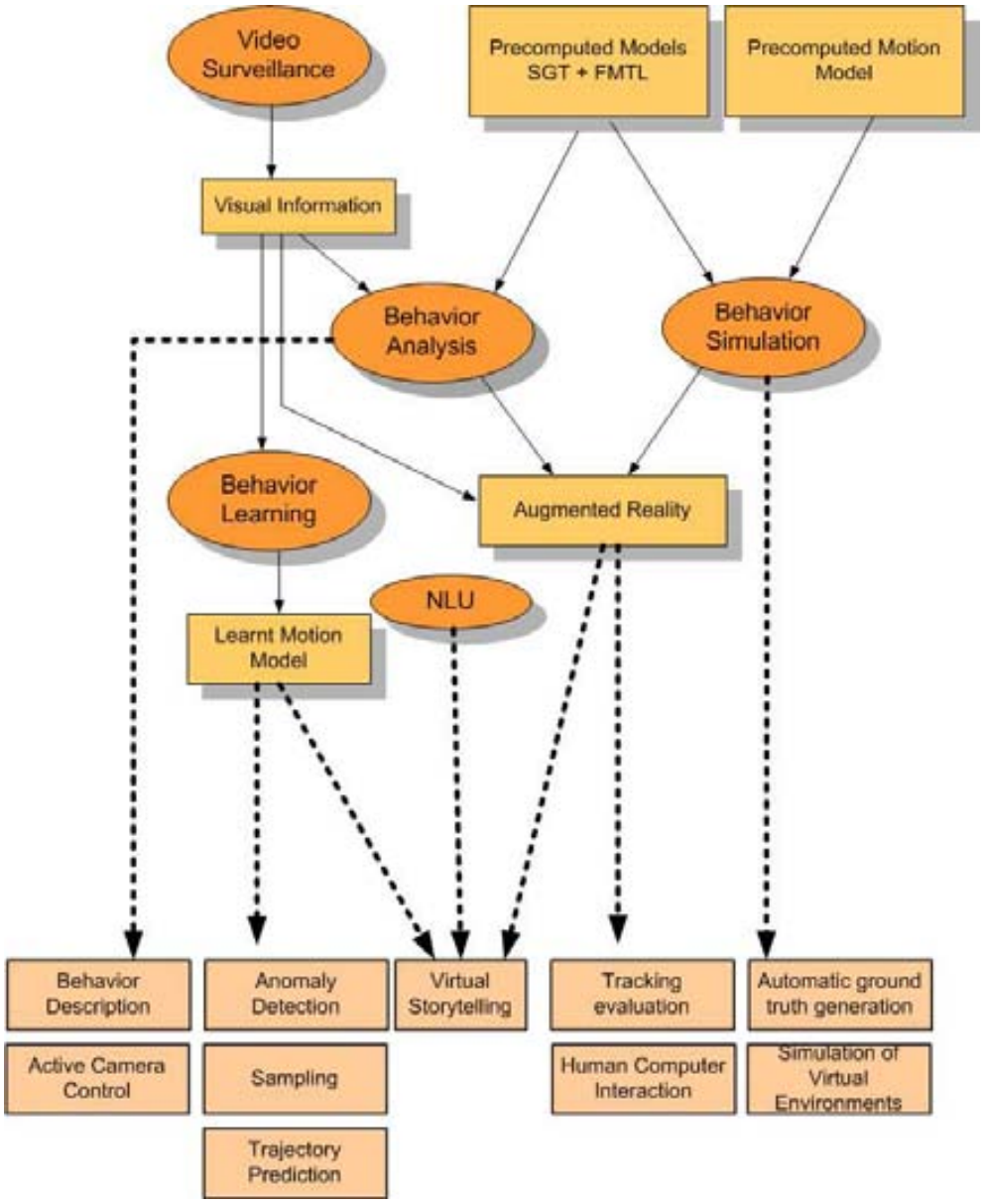


Figure 1.7: Scheme of the proposed approaches and contributions.

Chapter 2

Related Work

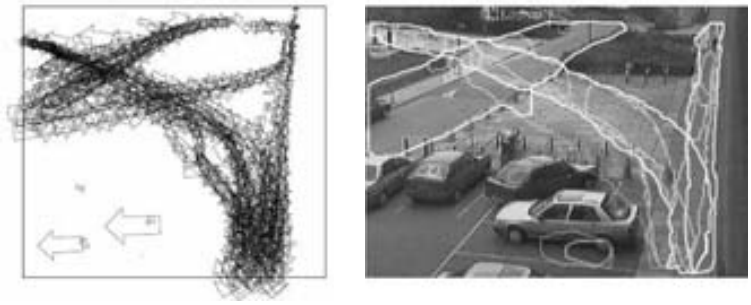
In spite of being a relatively new research area, a massive number of contributions related to behavior analysis in image sequences have been published in the last decade. This topic undoubtedly presents a big challenge and is receiving important amounts of private and public funds given the wide set of potential applications. However, most of the existing approaches present a contribution on a specific environment or a single agent of interest, sometimes making them too specific for general situations.

First, approaches are divided into two groups depending on the final purpose of the contribution. On the one hand, bottom–up approaches learn behavior patterns from observing activity inside a scenario. Obtained patterns are further used to identify new instances of the same behavior. Alternatively, top–down approaches contain prior knowledge about the behavior to be recognized, which can be specified using semantic information, thus allowing to extract semantic descriptions of new observed behavior instances.

First section reviews existing methods to learn scenario models from analyzing features obtained from tracking. Next, a complete description of the state of the art in behavior understanding is provided. Finally, last section presents related contributions to the field of augmented reality and computer animation.

2.1 Learning Scene Models by Trajectory Analysis

Current literature on analysis of surveillance videos has tried to automatically learn some conceptual data from the analysis of tracked individuals. This has lead to two main tasks: first, the clustering of trajectories allows to learn behavior patterns on specific regions of the scenario. Second, some regions of the scenario can be classified as one semantic class depending on the observed features (speed, orientation), for instance, road, sidewalk, etc. Nevertheless, these two tasks represent the same problem and there exists an association between semantic explanation of a specific region and the behaviors that can be observed there. Thus, on the one hand a region of the scenario with a specific semantic explanation denotes a set of behaviors that can be observed there. On the other hand, the learning of motion patterns from trajectories permits the classification of regions using semantic labels.



(a) (b)

Figure 2.1: Early approaches on scene learning by trajectory analysis. (a) The neural network approach by Johnson and Hogg [58] (b) Semantic regions obtained by Fernyhough et al. [39].

Several approaches faced this problem for the traffic domain. The analysis of vehicle trajectories allows to extract accurate models of behavior, since vehicles have low complexity in terms of mobility and behavior. In fact, the analysis of human trajectories instead of vehicle trajectories implies a high addition of complexity. On the one hand, while vehicles are supposed to follow a predefined set of marked lanes, human beings can perform any kind of trajectory, unless there is a physical barrier limiting their mobility. On the other hand, human motion is highly non-linear, a-priori unknown, and it is always subject to sudden and unforeseeable changes of orientation and speed.

Initial attempts were done by Johnson and Hogg in [58]. Their proposed method learns probability density functions of object trajectories generated from image sequences. the movement of an object is described in terms of a sequence of flow vectors, where each vector consists of four elements representing the position and velocity of the object in the image plane. The patterns of object trajectories are formed with two competitive learning networks which are connected by leaky neurons. Both of the neural networks are learnt using vector quantization.

Fernyhough et al. [39] proposed a method to learn and classify semantic regions from a scenario. This approach recognizes common paths by extending trajectories with the spatial extent occupied by the agents in camera coordinates. Although the method does not need any a–priori information, it requires full trajectories and cannot handle on–line learning. In addition, this method does not use orientation to compute paths and thus does not distinguish between objects following the same route but different directions. Despite showing promising intermediate results, the formulation lacks extensibility to other domains and no semantic description is reported. The method uses the size of tracked blobs to stablish the spatial extent of the agent at each frame step. It makes the method very dependent of the training set used to construct the path database.

The use of Hidden Markov Models (HMM) for the modeling of events has been widely studied. Galata et al. [40] proposed a Variable–Length HMM. In that work,

(a) (b) (c)

Figure 2.2: Example results from different approaches. (a) Route models constructed by Makris et al. [67](b) Division of the scenario into cells by Veeraraghavan et al. [100] (c) Statistical motion patterns obtained by Hu et al. [52]

the learning is done assuming a invariant background, where landmarks never change position. The method is not tested under a noisy dataset, where unobserved behaviors may appear.

The lack of conceptual labelling of the scenario is addressed by Makris and Ellis in [67], learning entry/exit zones and routes from trajectory samples. The start/end points of trajectories are used to learn entry/exit zones applying the Expectation-Maximization (EM) algorithm. For the learning of routes, a new trajectory is compared with all routes already in the database using a simple distance measure. If a match is found, the trajectory is added to the matching route and the route is updated. Otherwise, a new route is initialized.

The use of constructive models have been proposed to learn sequences of events observed in sequences. Veeraraghavan et al. [100] present a novel method for representing and classifying events in video sequences using reversible context-free grammars (CFG). The grammars are learned using a semi-supervised learning method. More concretely, by using the classification entropy as a heuristic cost function, the grammars are iteratively learned using a search method. The learning algorithm requires some knowledge of the structure of the grammar and a small set of supervised examples. Moreover, the scenario model is discretized into cells, which have to be manually drawn or randomly generated. The cells are deterministically generated, affecting the learnt event patterns.

Another statistical method to learn motion patterns is presented by Hu et al. in [52], where trajectories are clustered hierarchically using spatial and temporal information and motion pattern are represented with a chain of Gaussian distributions. Learnt motion patterns are used to detect anomalies and predict object future behaviors. The method construct statistical motion patterns from a set of object trajectories, which are used to detect anomalies and predict object future behaviors. The tracking module is implemented by clustering foreground pixels in each image frame and comparing pixel distributions between successive images. It outputs object trajectories and features (color, size, etc.), which form the sample data for learning motion patterns. After enouch sample data are acquired, object motion patterns are learned using a hierarchical trajectory clustering based on the fuzzy K–means algorithm. Each pattern is represented by a chain of Gaussian distributions, and sta-

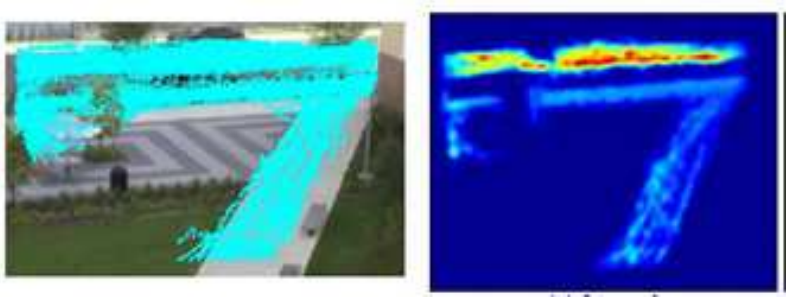


Figure 2.3: Scene model learnt by Basharat et al. [13]

tistical descriptions are then formed. For detection of anomalies, the probabilities of the matching between observed behaviors and the learned motion patterns and the probability values of abnormality of the observed behaviors are computed. Moreover, partial trajectories are matched to the learned motion patterns, and future behaviors are stochastically inferred.

Recent contributions have extended the set of features used to learn scenario models. For instance, Basharat et al. [13] presented a method to detect anomalies based on a probabilistic map of the scenario that takes into account not only trajectory positions but also object sizes, allowing to detect anomalous object sizes. They use observation vectors containing 5 features: timestamp t, position (x, y) and the dimensions of the bounding box (w, h). For each observation $O_i = (t_i, x_i, y_i, w_i, h_i)$, a set of transition vectors are created, which capture the transition from the given observation to future observations of the same trajectory. Considering a temporal window τ, the transition vectors are used to create a observation space at each pixel location, which is modeled using a Gaussian Mixture Model. The resulting probabilistic model is used to detect *local* and *global* anomalies. The former represent variations in speed and size of a new trajectory, while the latter represent completely unobserved routes. The method is also used to improve object detection, since it estimates the size of objects at each pixel location. However, the method is very dependent on a clean training dataset and classifies as anomaly very tiny variations of previously observed trajectories.

An approach to on–line learning trajectory clusters is presented by Piciarelli et al. in [83]. On–line clustering is about clustering computed as the incoming data are acquired. Clusters are represented in a tree like structure representing the shared prefixes of clusters, where a *prefix* is just the starting piece of a cluster. A new distance measure is proposed to compare clusters with incoming trajectories, based on the Euclidian distance. Clusters can be split and merged depending either on new acquired data or on the time a cluster has remained static in the database. Obtained clusters are used to perform behaviour analysis, since they allow the identification of anomalous events. However, this method does not tackle human trajectory features like bidirectional paths. Moreover, the method does not derive any semantic description about obtained clusters.

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

Figure 2.4: (a) Normal and (b) Abnormal trajectories learnt by [57]

Anjum and Cavallaro [5] employ PCA to reduce the dimensionality of trajectories. They analyze the PCA first two components of each trajectory together with their associated average velocity vector. Mean-shift, with these features, is employed to seek the local modes and generate the clusters. The modes associated with very few data points are considered as outliers. The outlier condition is set as the 5% of the maximum peak in the dataset, but again the drawback of the approach is that the analysis is adapted to highly structured scenes. Similarly, Naftel and Khalid [70] first reduce the dimensionality of the trajectory data employing discrete Fourier transform coefficients and then apply a self-organising map (SOM) clustering algorithm to find normal behaviour. Antonini and Thiran [6] transform the trajectory data employing independent component analysis, while the final clusters are found employing an agglomerative hierarchical algorithm. In these approaches it is however delicate to select the number of coefficients that will represent the data after dimensionality reduction. Calderara et al. [24] employ a k-medoids clustering algorithm on a transformed space modelling different possible trajectory directions to find groups of normal behaviour. Abnormal behaviour is detected as a trajectory that does not fit into the established groups, however the approach is validated with acted abnormal trajectory. Recently, Patino et al. [81] applied agglomerative hierarchical clustering to find the main trajectory patterns of people and relational analysis clustering to extract the relationship between people, contextual objects and events.

Given few training samples, clustering methods based on a similarity or dissimilarity measures can provoke overfitting. This is true for the first several steps of hierarchical clustering: when clusters contain only a few trajectories (starting from only one trajectory), the learnt patterns tend to be overfitted and the dissimilarity measures based on them are quite unreliable, thus resulting in clustering errors. These errors will propagate to future clustering steps. To address this problem, Jiang et al. [57] considers trajectories as dynamic processes and uses Dynamic Hierarchical Clustering (DHC) to train HMMs for each cluster, thus updating the clustering results at each merging step. Once a new HMM is trained after trajectory merging, all the trajectories in the database are reclassified. Possibly, some incorrectly clustered trajectories at previous steps are associated to the new HMM. All the HMMs are then retrained based on the updated trajectory clusters. This is a typical data reclassification and model retraining process, which is used in many iterative algorithms such as the expectation-maximization algorithm. This updating also enables error correction at later clustering steps, as clusters have gathered more samples and the

Approach	Model used	Types of anomalies
Fernyhough and Hogg [39]	Vector quantization	HA
Makris et al. [67]	GMM and vector quantization	HA
Hu et al. [52]	Fuzzy C-means and GMM	SA, HA
Piciarelli et al. [83]	Vector quantization	HA
Basharat et al. [13]	Transition vectors and GMM	SA, HA
Our approach	GMM with Splines	SA, IA, HA

Table 2.1
Comparison of previous approaches in terms of model and the types of anomalies detected.

trained HMMs are more reliable.

The principal application of most of the previously described learning approches is the detection of anomalies from new observations in image sequences. From a statistic point of view, normal behavior occurs more frequently than anomalous behavior. This is the main assumption of all recent works performed in anomaly detection research [13, 52, 67, 83].

Our proposal focuses on the definition of anomalies in terms of a hierarchy of deviations from a previously learnt model. In this work we differentiate three semantically different anomalies, namely *Soft* (SA), *Intermediate* (IA), and *Hard* (HA) anomaly. Each anomaly is related with a level of deviation between the learnt model M and new trajectory observations. Table 2.1 summarizes existing contributions in the field of statistical anomaly detection, compared to our proposed anomaly definition.

2.2 Behavior Understanding

In this section, the state of the art in behavior understanding is reviewed. Such research lines has raised several applications in different fields, as explained in the Introduction. Thus, the methods introduced next have been used to analyze behavior from different moving targets, e.g. faces, animals, vehicles, and human beings. The latter represents nowadays the most challenging goal in behavior understanding, due to special features of human behavior, such as variety and unpredictability.

Current motion understanding systems rely on numerical knowledge based on (i) the quantitative data obtained from tracking procedures and (ii) the geometrical properties of the scene[23],[62]. Therefore, this process is usually scene-dependent, and a-priori information of the spatial structure of the scene is required. The questions about the *what* and *why* can be answered by reasoning about the tracking data and transforming it to semantic predicates which relates each tracked agent with its environment. Common problems are the *semantic gap* which refers to the conceptual ambiguity between the image sequence and its possible interpretations, and *uncertainty* raised due to the impossibility of modeling all possible human behaviors.

2.2.1 Probabilistic Behavior Understanding

The major existing methods in the literature related to behavior understanding make use of standard probabilistic techniques and are outlined as follows.

Principal Component Analysis (PCA), as a statistical approach, has been applied to the recognition of object behaviors. For instance, Yacoob and Black [105] learn behavior models using PCA of a number of exemplar actions.

Dynamic Time Warping (DTW) is a template–based dynamic programming matching technique which has been used to match human movement patterns. For instance, Bobick and Wilson [18] use DTW to match an input signal to a deterministic sequence of states.

Hidden Markov Models (HMMs)generally outperform DTW in the processing of undivided successive data and are therefore extensively applied to behavior understanding. Brand and Kettnaker [21] show that, by minimizing the entropy of the joint distribution, observed behaviors can be organized into meaningful states within an HMM. Oliver et al. [76] propose and compare two different state-based learning architectures, namely, HMMs and CHMMs (Coupled Hidden Markov Models) for modeling people behaviors and interactions. Nguyen et al. [75] present an Abstract Hidden Markov Memory Mode-based approach for recognizing high-level human behaviors. Duong et al. [34] apply the Switching Hidden Semi-Markov Model to learn and recognize human behaviors and detect anomalies. More recently, Nguyen et al. describes in [74] the application of Hierarchical Hidden Markov Models (HHMM) to learn and detect complex behaviors in indoor scenarios. They define a two–level HHMM by considering complex and primitive behaviors. Similar to work from Park and Aggarwal [80] work, they recognized primitive behaviors based on HMMs. Treating the recognition of primitive behaviors as observations for complex behaviors, they constructed another layer of HMMs on top of the primitive behavior recognition system. As a result, their system was able to represent and recognize two levels of behaviors with a probabilistic model.

Unlike HMMs, Variable–Length Markov Models (VLMMs) can capture behavioral dependencies that may have a variable or long time scale. Galata et al. [40] propose a method for automatically acquiring stochastic models of a behavior. VLMMs are used for the efficient representation of behaviors.

Grammar Techniques (GT), based on low level features detected by standard independent probabilistic temporal behavior detectors, provide longer–range temporal constraints and disambiguates uncertain low-level detections, etc. Brand [20] uses a simple non–probabilistic grammar to recognize sequences of discrete behaviors. Ivanov and Bobick [56] describe a probabilistic grammar approach for the detection and recognition of temporally extended behaviors and interactions between multiple objects. Minnen et al. [68] present a system that uses human–specified grammars to recognize a person performing the Towers of Hanoi task from a video sequence by analyzing object interaction behaviors.

Bayesian Networks (BNs) offer many advantages for using prior knowledge and modeling the dynamic dependencies between parameters of object states. Town [97] uses an ontology to train the structure and parameters of Bayesian networks for behavior recognition. Park and Aggarwal [80] describe a framework for recognizing

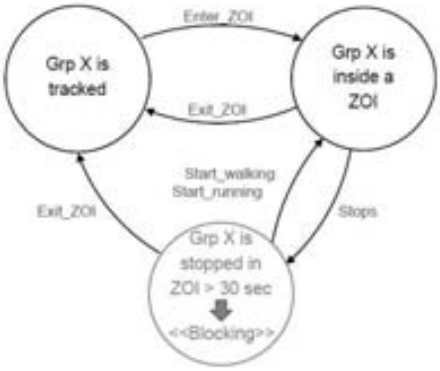


Figure 2.5: Bremond et al. [22] use Finite State Automatons for sequential scenarios.

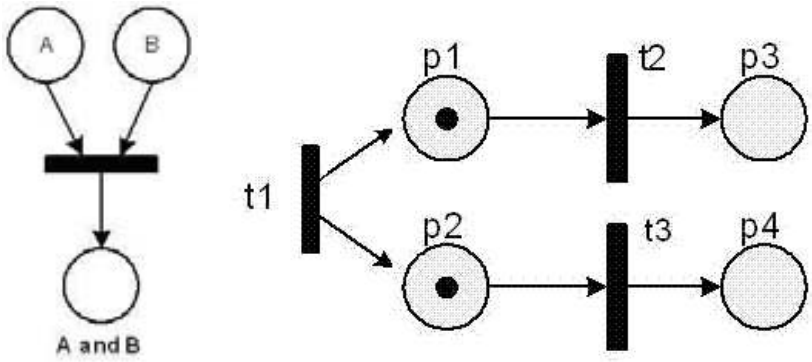
human actions and interactions. In the framework, the poses of individual body parts are recognized using individual Bayesian networks, and the actions of a single person are modeled using a dynamic Bayesian network. Statistical Shape Theory (SST) is an effective tool for analyzing object behaviors. Vaswani et al. [99] model a behavior by the polygonal *shape* of an associated configuration of point objects and its deformation over time. Both *drastic* and *slow* anomalies can be detected.

Non–deterministic Finite Automaton (NFA) are employed by Wada and Matsuyama [101] as a sequence analyzer. They present an approach for multiobject behavior recognition based on behavior-driven selective attention.

2.2.2 Semantic–based Behavior Understanding

The above introduced methods do not include a semantic explanation of the recognized behaviors. The semantic gap, explained in the Introduction, is therefore not tackled. In this section we describe approaches that make use of prior semantic knowledge to enhance the interpretation of the observed behavior patterns. These approaches, often called top–down or description–based approaches, normally represent the behaviors to be recognized in terms of logic predicates and chains of causality. The design of such descriptions has to be done by a human expert and may be scenario dependant. This characteristics make them unsuitable for unconstrained environments, however they can provide an accurate explanation of observed behaviors in selected scenarios.

A complete behavior recognition system is proposed by Bremond et al. [22]. For each tracked actor, the behaviour recognition module performs three levels of reasoning: states, events and scenarios. For simple scenarios, they verify that a state has been detected during a predefined time period using a temporal filter. For sequential scenarios, they use Finite State Automatons (FSA). For composed scenarios defining a single unit of movement composed of sub scenarios they propose AND/OR trees of sub scenarios. Behaviors are defined using an event description language. An example of the previously mentioned structures is shown in Fig. 2.5.



(a) (b)

Figure 2.6: Example of Petri Nets extracted from [42].

Alternatively, the use of Petri Nets has been studied for event recognition in indoor and outdoor environments by Ghanem et al. [42]. A Petri Net (PN) is a directed bipartite graph, in which the nodes represent transitions (i.e. discrete events that may occur), places (i.e. conditions), and directed arcs (that describe which places are pre- and/or postconditions for which transitions), see Fig. 2.6. In the context of event detection, moving objects in the scene are considered tokens that are moving along the PN whilst existing in the image sequence. Tokens are situated in one place and can change their place when an event is recognized, i.e. a neighbor transition is activated. A place represents a possible state of one of more objects. The recognition of the event is based on logic conditions that are explicitly predefined by a human expert, e.g. object speed is greater than 0, object is of type *person*, etc. The PN formalism allows to express logical and temporal relations, concurrency, and partial ordering of events, see Fig. 2.6. The transitions can be ordered by priority, allowing to solve conflicts when multiple transitions are available from the same place. Moreover, transitions are associated to a time period, which has to be ellapsed for the transition to be fired.

More recently, some extensions have been proposed by Borzin et al. in order to a provide the PN formalism with stochastic timed transitions [19]. The delay between transitions is learnt from a set of training examples, using a negative exponential probability density function. The resulting formalism, Generalized Stochastic Petri Net (GSPN), represents the coexistence of immediate transitions and stochastic timed transitions. Latest improvements for the PN formalism entail the addition of probabilities in the directed arcs going to a transition, which can also be learnt from a set of training examples. In order to tackle the uncertainty carried from the detection and tracking processes, Albanese et al. have proposed to attach a probability score to each token, i.e. each object [3].

Fuzzy Metric Temporal Logic (FMTL) [90] also copes with the temporal and uncertainty aspects of integration in a goal-oriented manner. This predicate logic language treats dynamic occurrences, uncertainties of the state estimation process,

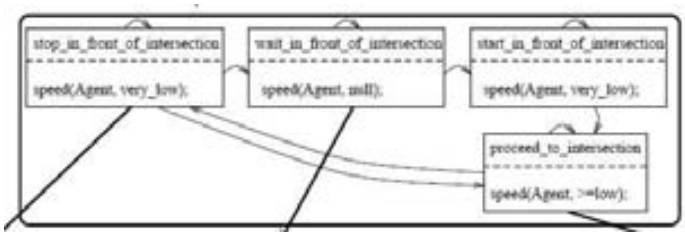


Figure 2.7: Situation Graph describing a vehicle approaching to an intersection, extracted from [49].

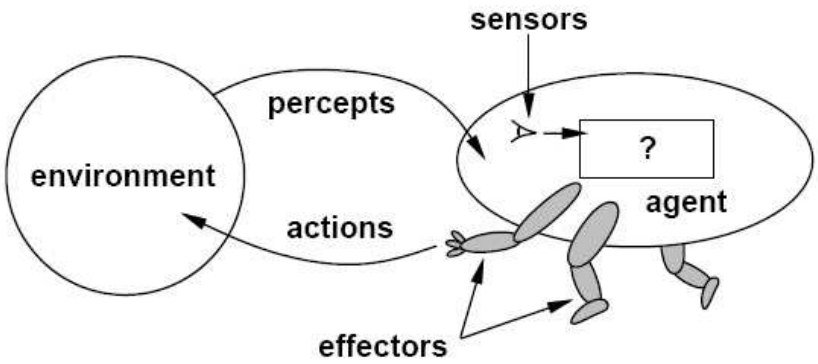


Figure 2.8: Intelligent Agent perception/reaction scheme from [89].

and intrinsic vagueness of conceptual terms in a unified manner. FMTL uses three different strategies to accomplish such an abstraction process, according to the source of knowledge which is exploited to generate the qualitative description. The main advantage of FMTL over the previously referred algorithms relies on the promise to support not only the interpretation, but in addition diagnosis during the continuous development and test of the so-called *Cognitive Vision Systems* [72]. Further, Situation Graph Trees (SGTs) constitute a suitable behavior model which explicitly represents and combines the specialization, temporal, and semantic relationships of the constituent conceptual predicates in FMTL. The SGT is introduced in the following chapters and has been initially used by Nagel et al. for the recognition of traffic environments involving vehicles [49]. Fig. 2.7 shows a Situation Graph describing the situations involved in the behavior of a vehicle approaching to an intersection.

2.3 Behavior Modeling for Augmented Reality

The design of intelligent virtual agents (IVAs) in virtual environments (VE) entails the collaboration of several areas of computer science.

First, intelligent agent modeling constitutes an entire field in artificial intelligence research, and a complete description can be found in Russell and Norvig's book [89].

There, an agent is defined as something capable of getting data from the environment using sensors, which is called *perception*, and reacting to these stimuli through effectors, see Fig. 2.8. Based on this initial interaction scheme, the *ideal rational agent* is capable of driving its actions so that it maximizes some performance measure. Intelligent agents are classified into two main types, depending on how the initial interaction scheme shown in Fig. 2.8 is particularized:

- Reflex Agents. This class represents agents that just react to perceived stimuli by means of if–then rules. A more sophisticated version of these agents can maintain a internal memory to keep track of the current state of the environment.

- Goal–based Agents. Agent behavior is determined not only by the perception from the environment, but also by some internal goals that the agent should acchieve in the future. In order to successfully accomplish that goals, agents must planify an action sequence that drives towards the desired configuration. Based on an *utility* function, the agent should also find the *best* way to acchieve the goals, thereby solving possible conflicts between goals.

In this work our aim is to create virtual characters capables of performing complex behaviors whilst interacting with the environment. Thus, the required agent structure is similar to the Goal–based Agents proposed by Russell and Norvig: Behavior–based agents have a set of goals (described as their *behavior*), and they accomplish it depending on the perceived actions from other agents in the environment. The representation of these goals is then a crucial point in the design of virtual agents [59]. So far, current contributions have developed models by considering two different approaches: bottom–up and top–down techniques.

Bottom–up approaches make use of machine learning techniques to represent behaviors from a set of supervised examples. Thus, *behavioral learning* has been exploited recently using techniques such as k–Nearest Neighbours [28] or Reinforcement Learning (RL) [94, 28]. On the one hand, the k-NN algorithm provides a local approximation of the target function and can be used automatically without the designer selecting the inputs. It is guaranteed to learn the target function based on the quality of the examples provided and to memorize the decisions made by planning through a cognitive model. On the other hand, the RL algorithms allow one or several agents to carry out a series of optimal actions according to a given environment thanks to reward/penalty techniques. Through the repetition of non-pertinent or pertinent tests, these agents learn the task asked for. Alternatively, top–down techniques predefine a set of behavior patterns which represent the set of agent behaviors to be synthesized.

Top–down approaches have a limited learning capability, but they take advantage of the richness provided by semantic knowledge to represent human behaviors in a consistent manner, and they do not depend on the specific data from the training set [16, 82, 8]. In this Thesis agent behavior is modeled also using the Situation Graph Tree formalism. Thus, the SGT is used to model the behavior of real and virtual agents, thereby integrating both types of agent in the same scenario.

Second, intelligent virtual agents may emulate the capabilities of a real characters inside a virtual environment, for instance vehicles or humans. To this end, the sensors

and effectors from the original agents have to be emulated by the virtual ones. In the case of humans, which is the topic of this Thesis, sensors are eyes, ears, and other organs, and hands, legs, mouth, and other body parts represent the effectors [89]. This process entails the collaboration of two complementary subprocesses, phisic and shape modeling:

- The physical model of the agent is the representation of the physical properties and constraints of the virtual agent, according to those from the environment. This process entails the creation of a simplified representation of a character's anatomy, analogous to a skeleton or stick figure. The position of each segment of the skeletal model is defined by animation variables, or Avars. In human and animal characters, many parts of the skeletal model correspond to actual bones, but skeletal animation is also used to animate other things, such as facial features. Traditionally, animators manipulate the Avars directly. However, rather than set Avars for every frame, they usually set Avars at strategic points, key–frames, and the rest of the animation is automatically interpolated. A different method called *motion capture* makes use of observing real motion [66, 8]. In motion capture, a real character performs the desired actions and the obtained motion is encoded using video cameras and markers. Finally, the action sequence is replicated into the virtual character.

- The shape model represents the appearence of the agent within the environment. This process resides in the area of computer graphics and deals with aspects like realism, illumination, occlusion consistency, etc. Nowadays, objects are represented by complex structures usually containing thousands of polygons. The animation is created by rendering the scenario, i.e. introducing the shape models into the *rendering pipeline* [95]. The technique that produces the most realistic rendering is *Ray–tracing*, which emulates the reflexion and refraction of light sources in their interaction with the objects in the scenario [43]. This technique involves a high level of computational complexity and a general real–time solution is currently a hot research topic.

  Magnenat and Thalmann list the three steps described above as the exhaustive way to model virtual humans [65]. Also, they state the current challenges in the field of virtual human modeling. Among them, two important remaining issues are ensuring a realistic on–the–fly generation of motion of virtual humans and creating believable relationships between real and virtual humans in mixed environments.

2.3.1 Augmented Reality

A complete description of the stages in creation of augmented reality involving virtual characters can be found in [64]. There, a complete model of the scenario is used to produce realistic images with virtual avatars, see Fig. 2.9. In that paper a list of problems related is stated, although most of them are related to computer graphic features, such as illumination or character realism, which are not tackled in this Thesis.

Figure 2.9: Example animation result from [64]

(a) (b) (c)

Figure 2.10: Example augmented reality results from current works. (a) Example result from [11] (b) Example result from [108].

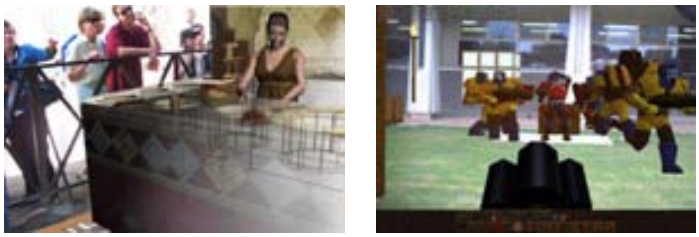
Interaction between real and virtual agents has been little considered previously [12, 41]. Balcisoy et al. showed the application into a virtual storytelling system, restricting virtual agents to perform a given script, converting them in directors of the scene [12]. More recently, Balcisoy et al. describe a mixed reality setup to mix virtual characters with the tracking of particular objects in the scene, see Fig. 2.10(a) [11].

Gelenbe et al. proposed an augmented reality system which combines computer vision with behavior–based object agents [41]. Behavior is modeled using a hierarchy of three behavior modules, but without considering the particular features of human motion and behavior. The occlusion between real and virtual is tackled only with static objects, for instance a tree. All the static objects that may occlude virtual characters are segmented from the background by generating a virtual representation of the scenario.

Zhang et al. presented a method to merge virtual objects into video sequences recorded with a freely moving camera [108]. Multiple views of the scenario are matched by means of fusing features extracted using the SIFT algorithm. In order to integrate the virtual objects consistently, the 3D scene model is reconstructed by means of an extended multi–baseline algorithm. The method is consistent according to illumination and shadows, but it does not tackle occlusions with real moving agents, see Fig. 2.10.(b).

The use of computer vision techniques in augmented reality has also been re-

Figure 2.11: Example result from [61]. Several virtual Ewoks are defeated by a virtual Darth Vader on a desk.



(a) (b)

Figure 2.12: (a) Example result from [78]. (b) Example result from [84].

cently confronted by Douze et al. [33], where moving targets are tracked from image sequences and merged into other real or virtual environments, see Fig. 2.10.(c). However, the method does not consider the animation of behavioral virtual agents in the resulting sequence.

Klein and Murray adapt SLAM algorithms developed for robotic exploration into the AR field [61]. Using images taken from calibrated hand–held cameras, they collect thousands of feature points that enable the estimation of a dominant ground–plane. Then, they use this information to add virtual objects over the ground–plane that develop some predefined behavior, see Fig. 2.11. The method keeps a correct estimation of the ground–plane as the camera moves, thereby maintaining a consistent existence of virtual objects in the image sequence.

The generation of augmented reality has received much attention given the wide set of potential applications, which have led to the publication of many contributions in different areas. For instance, Papagiannakis et al. use AR for the reconstruction of ancient cities in [78]. The proposed method presents virtual actors that introduce visitors of ancient locations into the world of fresco paintings, by providing these actors with dramaturgical behaviors. Augmented reality can be also very helpful to aid modeling processes, since it can provide additional information that can not be obtained by just observing the scenario. In [55], Barakonyi et al. use *ubiquitous* agents in AR displays in order to guide the user to configure interfaces, e.g. repairing a lego construction. Torre et al. used AR to simulate a virtual opponent in checkers competition [96]. The method tracks the checkers board to generate a consistent location of the virtual agent. Hugues et al. presented some examples of AR systems applied to military training, see Fig. 2.12.(a). Finally, the use of AR has been

massively investigated within the game and cinema industries, since it can provide impressive results using a reduced amount of resources. For instance, Piekarski and Thomas presented an augmented reality version of the popular game Quake in [84], where the player can shoot to virtual monsters in a real outdoor environment, see 2.12.(b).

2.3.2 Performance Evaluation of Tracking

Nowadays, research in multi–object tracking algorithms has achieved great results when tracking non–grouped targets in few frames image sequences including soft illumination changes. However, several problems, inherent to computer vision in general, are still unsolved and constitute a big challenge towards an unconstrained multiple–target tracking, namely long occlusions, grouping disambiguation and camouflage.

One typical drawback comes up when testing the tracking algorithm in a different environment or image sequence [14]. As long as difficult frames appear in the new image sequence, some modifications must be done to the algorithm (parameters, thresholds) in order to achieve good results. However, these modifications could damage the results obtained in former image sequences. Instead of using completely new sequences, it would be useful to have a method to gradually increase the difficulty of a given sequence. However, when recording a new sequence, even in the same scenario, we are exposed to several condition changes due to illumination, weather, or scenario configuration. In addition, it is sometimes hard to record image sequences containing crowds of people in public urban environments for legal or security reasons.

Related work in the field of performance evaluation for object tracking can roughly be classified into different semantic classes as they specifically address one or more of the following semantic levels [87]: pixel-level [2, 73], frame-level [14, 2], object trajectory level [14, 103], and behaviors or higher level events [31]. Bashir and Porikli [14] presented a set of unbiased metrics on the frame and object level which leaves the final evaluation to the community. However, the total number of 48 different metrics makes the interpretation difficult. Aguilera et al. [2] presented an evaluation method for the frame and pixel level, all based on segmentation. The pixel-level metric is currently used for the online service called PETS metrics[107]. Wu et al. [103] evaluate their body part detector-based tracker using five criteria on the trajectory level which cover most of the typical errors they observed. Furthermore, occlusion events were separately evaluated defining short and long scene or object occlusions. The metric then gives the number of successful handled occlusions against all occlusions of a certain category by dataset. Desurmont et al. [31] presented a general performance evaluation metric for frequent *high level* events where they use dynamic programming to specifically address the problem of automatic realignment between results and ground truth. Their definition of an event is however very much limited to the detection of blobs crossing predefined lines in order to count people passing by.

2.4 Resum

Malgrat ser una línia de recerca relativament nova, un gran nombre de contribucions relacionades amb l'anàlisi de comportaments en seqüències d'imatges han estat

publicades durant la darrera dècada. Aquest tema presenta indubtablement un gran desafiament i està rebent importants quantitats de subvencions tant públiques com privades, donat el seu conjunt potencial d'aplicacions. Tanmateix, la majoria de contribucions estàn destinades a analitzar un entorn específic o el comportament d'un agent en particular, provocant una manca d'usabilitat per a situacions generals.

En aquest capítol hem presentat l'estat de l'art en els diferents temes tractats en aquesta Tesi. En primer lloc, repassem els principals mètodes presentats darrerament per a l'aprenentatge automàtic de models de comportament a partir d'un conjunt d'aprenentatge. Seguidament, estudiem l'aplicació dins l'anàlisi de comportament de les eines més usades en la modelització de processos dinàmics, com per exemple els Models de Markov Ocults (HMMs) o les xarxes bayesianes (BNs). Seguidament també presentem els mètodes d'anàlisi de comportament basats en coneixement semàntic predefinit. En quant a la generació d'agents virtuals, presentem diferents mètodes per a modelar comportament sintètic i exemples d'aplicacions de realitat augmentada, sobretot combinant agents humans reals i virtuals.

Chapter 3

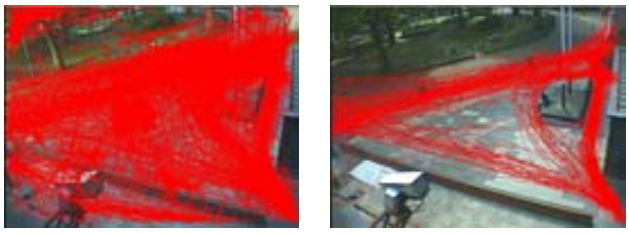
Behavior Learning

This chapter discusses two different strategies to automatically acquire behavior patterns from a video-surveillance scenario, taking as input the results of computer vision techniques applied to image sequences. As introduced in Chapter 1, the data available to attempt such a task is basically the set of typical features obtained from multiple–target tracking systems, thereby limiting the subsequent learning approaches in terms of the semantic classification of trajectories. The objective is to find a description of typical motion within the scenario. This is so–called in different ways along the recent literature: *semantic regions* [58], *statistical motion patterns* [52], *semantic scene models* [104, 67], *trajectory prototypes* [81, 9], etc. This is due basically to the application that each work performs to the obtained description. The common applications are anomaly detection and behavior analysis. A significant conclusion of the recent literature is that learning a scenario model has become the same task as learning a target behavior model, since the characteristics obtained for a given scenario describe the motion patterns expected to be observed, i.e the behavior of agents in terms of speed, orientation, and size.

In this chapter the aim is to obtain a description of agent motion within a scenario in terms of the paths that are mostly observed during a training period. Two approaches are presented, so–called On–line Scenario Modeling and Offline Scenario Modeling, respectively:

- The On–line learning method generates a hierarchy of trajectory clusters by processing each observation from tracking independently. This method does not need to process complete trajectories and can lead to intermediate results. However, On–line clustering does not consider the semantic meaning of complete trajectories, and noisy trajectories can be wrongly considered to update the model. In order to solve this drawback, a set of prunning methods are presented and discussed.

- The Off-line learning method follows the classical methodology of analyzing an existing training set [58]. The main advantage is that, since the method processes complete trajectories, the typical entry and exit areas can be automatically learnt and thus noisy or anomalous trajectories can be removed during



(a) (b)

Figure 3.1: (a) Original trajectory dataset (4250 trajectories). (b)) Trajectory dataset after removing tracking failure noise and semistationary motion noise [67].

the training process. The resulting model is then applied to perform anomaly detection on new observed trajectories.

Next, the obtained scenario models are used to detect anomalies in new observed trajectories. First, we define a taxonomy of anomalies in terms of the level of deviation from the models, which represent *normal* behavior. Recall that, while the training of the model can be performed by either the On–line or the Off–line Scenario Modeling methods, the subsequent anomaly detection system must have on–line performance in order to be reliable in video–surveillance and other similar applications. To this end, we propose probabilistic framework to incrementally classify trajectories between different kinds of anomalies, as new observations are acquired from the tracking system.

3.1 Preprocessing the trajectory dataset

The methods proposed in this chapter are intended to learn using large recording datasets without any previous handmade trajectory selection. Therefore, the initial trajectory dataset might contain spureous data, see Fig. 3.1.(a). Summarizing, the most relevant noise problems are due to:

- Tracking failure noise due to the failure of the motion-tracking algorithm to track a target successfully for its whole activity in the scene. It may appear in the form of false trajectories (trajectories where motion histories of more than one target have been mixed), or split trajectories (trajectories that represent only a portion of the motion history of a target).

- If sources of semistationary motion noise are present (such as trees, curtains, window reflections), then an apparent high activity is detected in the vicinity of the source of the semistationary motion noise.

The first and second noise problems are solved by applying the multistep learning algorithm of Entry and Exit zones, by Makris et al. [67]. The algorithm uses Expectation Maximization (EM) to find the set of entry (S) and exit (E) zones, represented by Gaussian Mixture Models (GMM). Since the number of entry and exit areas is

unknown beforehand, it is overestimated by the algorithm in an initial clustering. Afterwards, the resulting GMM is pruned using a density criterion and the EM is run again. Subsequently, S and E allow to obtain a subset $T' \subseteq T$ of trajectories that begin in some entry area of S and end in some exit area of E:

$$T' = \{t \in T | \exists_{s \in S, e \in E} : begins(t, s) \ \wedge \ ends(t, e)\} \tag{3.1}$$

3.2 On–line scenario modeling

This section presents an on–line clustering algorithm to construct a cluster database from trajectories obtained by a multiple–target tracking system. In our case, we will use the system described in [88]. The term *on–line* means that the algorithm updates the model using the information obtained for a single frame step, instead of requiring a complete trajectory (intermediate on–line) or a complete trajectory set (offline).

Trajectories are defined as sequences of observations for a given target over time:

$$t = \{o_1, \ldots, o_n\} \tag{3.2}$$

Each observation is formed by the center of the surrounding ellipse of the target. In addition, uncertainty is represented by a zero–mean gaussian noise:

$$o_i = (x_i, y_i) + \mathcal{N}(0, \sigma) \tag{3.3}$$

where Σ_I is the diagonal, simmetric covariance matrix, which needs to be converted to ground–plane coordinates, in order to obtain the real noise. We obtain the ground–plane covariance Σ_w by using an homography H, which maps image to ground plane coordinates:

$$\sigma_x = d(H \times \begin{bmatrix} x \\ y \end{bmatrix}, H \times \begin{bmatrix} x + \sigma \\ y \end{bmatrix}), \tag{3.4}$$

$$\sigma_y = d(H \times \begin{bmatrix} x \\ y \end{bmatrix}, H \times \begin{bmatrix} x \\ y + \sigma \end{bmatrix}), \tag{3.5}$$

$$\Sigma_w = \begin{pmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{pmatrix} \tag{3.6}$$

where d is the Euclidean Distance.

Subsequently, a cluster is defined as an accumulation of trajectories, and is based on the uncertainty of the observations. Thus, clusters are represented as a sequence of cluster nodes $C = \{c_1, \ldots, c_m\}$ where $c_i = (\mu_i, \Sigma_{i_w})$, see Fig. 3.2.

From these clusters we construct a tree representation in order to establish spatio–temporal relationships of precedence between clusters. Thus, given a cluster C_k and a list of clusters $L = \{C_1, \ldots, C_n\}$, if every path beginning in C_k finishes in one cluster of L, this is represented in this work by means of tree–like structures, so–called *cluster trees*, see Fig. 3.3. In this framework, the childhood relationship (C_i, C_j) indicates that C_i absolutely precedes C_j.

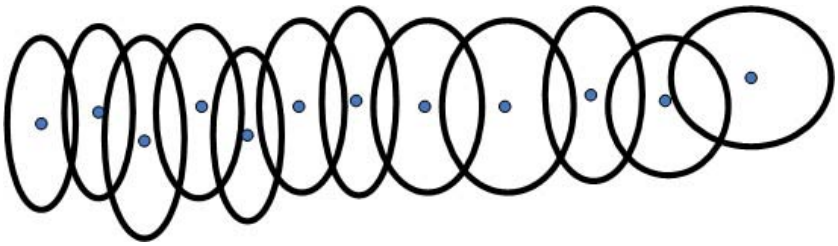


Figure 3.2: Cluster constructed from incoming trajectories. Dark circles represent the mean $\{x, y\}$ center of each cluster node and white circles represent their covariance matrix Σ.

Figure 3.3: Example of obtained clustering of trajectories and the subsequent tree representation.

3.2.1 Distance Measurement and Cluster Update

In order to check if a trajectory is matching a cluster, the distance between the cluster and each new observation is measured. Nonetheless, although distance is measured in terms of spatial proximity, time information must be used in order to compare the agent position only with those cluster nodes which are close in time. Since each cluster node has different variance for each axis, we define:

$$D\left(o_i, C_k\right) = \min_j \left(mahal\left(o_i, c_j\right)\right), j \in \{\lfloor(1-\delta)i\rfloor \ldots \lceil(1+\delta)i\rceil\} \tag{3.7}$$

where the $mahal$ computes the Mahalanobis distance between an observation o and a cluster node $c_j = (\mu_j, \Sigma_j)$, considering the distribution in c_j:

$$mahal(o, \mu, \Sigma) = \sqrt{(o-\mu)^T * \Sigma^{-1} \Delta (o-\mu)^T} \tag{3.8}$$

When observation o has been captured, its position is compared with cluster nodes that represent a time period close to i. The δ parameter in Eq. 3.7 defines a sliding temporal window centered in the time step i. This window is required in order to compare the distances between the observation and cluster nodes that are close in time. The temporal window is then clipped in the range $(1 \ldots ||C||)$ and, if it completely falls outside this range, the distance D is set to ∞. Note that the values $mahal(o, c)$ (i) can be computed as new observations are generated by the tracking system, and (ii) can be used to detect if the trajectory is moving away from the cluster.

A cluster node is updated when a new observation is incorporated. To do so, we once more represent the contribution of that observation as a gaussian having the

initial diagonal covariance matrix. The cluster is updated by computing a normalized sum of the two gaussians, with respect to the number of observations contained in each cluster node.

$$\mathcal{N}_{N+1}(\mu_{N+1}, \Sigma_{N+1}) = \mathcal{N}_N(\mu_N, \Sigma_N) + \mathcal{N}_1(\mu_1, \Sigma_1) \quad (3.9)$$

$$\mu_{N+1} = \frac{(N+1)\mu_N + \mu_1}{N} \quad (3.10)$$

$$\Sigma_{N+1} = \frac{(N+1)\Sigma_N + \Sigma_1}{N} \quad (3.11)$$

3.2.2 Clustering algorithm

The clustering algorithm is sketched in Fig. 3.4. The input is a new observation o, consisting of a trajectory identifier id, the center of the bounding box (x, y) and the uncertainty Σ. When a new trajectory is detected by the tracking system, it does not exist in the trajectory database. In this case, the trajectory is added and its initial position is compared against the entrance points, represented by the root nodes of the cluster tree set. If a match is found, the corresponding cluster is updated and associated to the trajectory, whose status is set to $FOLLOWING$. Otherwise, a new cluster tree is created with a root cluster containing the initial position of the trajectory, and the trajectory status is set to $CREATING$. However, if the trajectory already existed in the database and its status is $FOLLOWING$, the distance from the agent status to its related cluster C is computed. If the distance is small enough, the trajectory is considered to be still following C, and the nearest cluster node of C is updated using Eq. 3.9. Otherwise, the algorithm tests whether the trajectory matches one cluster of the children of C. If so, the related cluster is changed by the correspondent child. In case that no child cluster is close enough, a bifurcation is created at the nearest cluster node of C, see Fig. 3.5. This fact implies the creation of two child clusters C_t (a tail cluster split from C) and C_{new}, a new cluster assigned to the trajectory, whose status is now set to $CREATING$.

3.2.3 Generating a Conceptual Model of the Scenario

The on–line procedure described above obtains a quantitative representation of the typical development of the paths between two significant areas of the scenario. However, no conceptual knowledge has been inferred so far, since no qualitative data has been extracted from the clusters. In this work we will focus on two concepts that can be inferred by means of the clusters: speed and orientation changes, which will give a semantic description on some parts of the scenario. Also, we next describe how to deal properly with outliers and atypical behaviors observed in the scene. In first subsection we briefly introduce methods to prune the database and subsequently we describe the conceptualization of the resulting model.

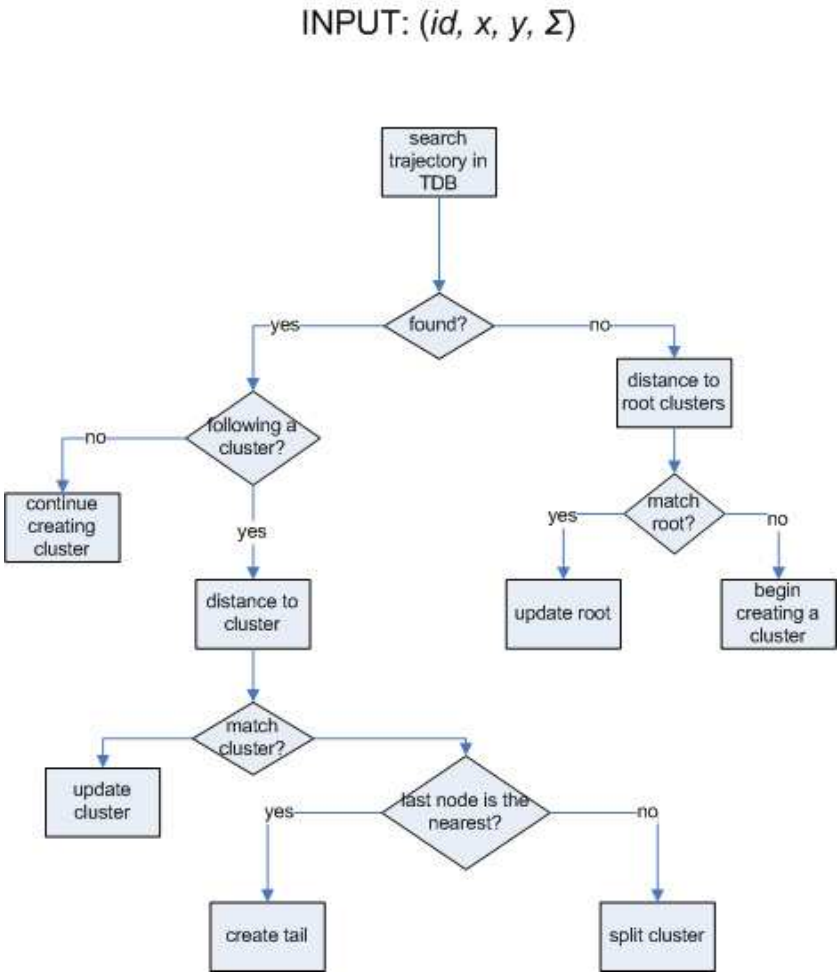
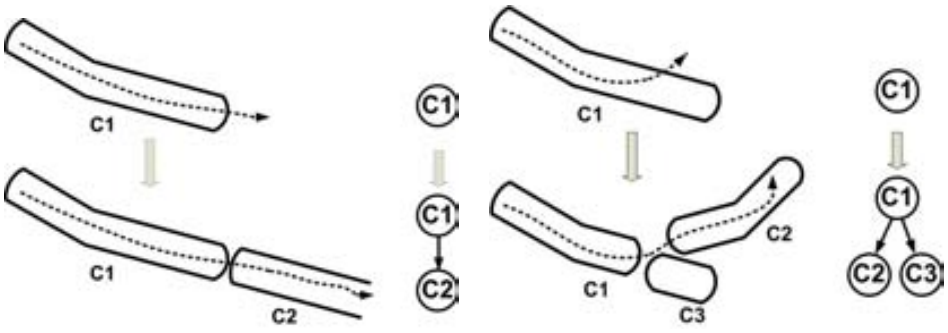


Figure 3.4: On–line clustering algorithm.



(a) (b)

Figure 3.5: Situations producing the creation of new clusters. (a) New cluster created since the trajectory goes further than its followed cluster. (b) Split situation when a trajectory is not within the current cluster anymore. A new cluster is created and the previous is divided into two at the splitting point.

Cluster Database Management

Human trajectories are diffuse, unpredictable, and they are exposed to unforeesable changes in orientation and speed. Therefore, establishing a set of paths in a human environment requires to apply a maintenance process to the database when the number of clusters grows. This maintenance consists of analyzing the tree representation in order to reduce the number of clusters by keeping only those containing the maximum number of trajectories. To this end, three different techniques are presented: cluster tree prunning, sibling merging, and child concatenation.

Cluster Tree Prunning Prunning the cluster tree preserves of maintaining clusters that either have a low update frequency (abnormal behavior) or they have not been updated for a long time period (indicating that the environment has changed and may be the path is not available anymore).

On the one hand, frequency is computed recursively to each branch of the cluster tree set, beginning at the root. Subtrees whose root update frequency is below a threshold τ are removed from the cluster tree set. On the other hand, temporal prunning requires to maintain information about last updates. Subtrees not being updated for a long period are removed.

Sibling Merging Sometimes two clusters with the same parent (*sibling* clusters) are very similar and represent the same path. This happens when a new trajectory does not match any existing cluster, thereby creating a new one. Despite having different starts, the existing and the new cluster are very similar and could be merged into a single cluster.

To this end, similar sibling clusters are searched and merged, combining two similarity measurement techniques:

- Distance between clusters: This similarity measure returns the distance measure

between two clusters $C_i = \{c_{i_1}, \ldots, c_{i_n}\}$ and $C_j = \{c_{j_1}, \ldots, c_{j_m}\}$ as the mean of the normalized distances of every point C_{i_k} from the nearest point of C_j found inside the sliding temporal window centered at k.

$$D\left(C_i, C_j\right) = \frac{1}{n} \sum_{k=1}^{n} d\left(c_{i_k}, C_j\right) \tag{3.12}$$

where $d\left(t_i, C\right)$ is the distance measure defined in Eq. 3.7.

- Cluster–to–cluster *correspondence*: Given two clusters $C_i = \{c_{i_1}, \ldots, c_{i_n}\}$ and $C_j = \{c_{j_1}, \ldots, c_{j_m}\}$, the correspondence interval between C_i and C_j is defined as the pair of sequences $S_i \subseteq C_i$ and $S_j \subseteq C_j$ so that:

$$\begin{aligned} S_i &= \{c_{i_k}, \ldots, c_{i_l}\}, 1 \leq k < l \leq \|C_i\| \\ S_j &= \{c_{j_p}, \ldots, c_{j_q}\}, 1 \leq p < q \leq \|C_j\| \\ \|S_i\| &= \|S_j\| \\ \forall_x &: 1 \leq x \leq \|S_i\| : mahal\left(S_{i_x}, S_{j_x}\right) \leq 2\sigma \end{aligned} \tag{3.13}$$

  The correspondence interval is computed for each pair (C_i, C_j) of sibling clusters. If the size of the correspondence interval is relevant according to the whole cluster size, then C_i and C_j are merged as shown in Fig. 3.6. The correspondent part is merged into a single cluster C_{ij}, and non–correspondent tails C_{i_t} and C_{j_t} are set children of C_{ij}. Finally, children of C_i and C_j are set children of C_{i_t} and C_{j_t}, respectively.

These similarity measures are applied to each cluster by following a fixed–point algorithm: the similarity between sibling clusters is continuously measured and possible merges are performed until no more similarities are found. In each loop, merging is measured recursively: for each node of the tree, merging is first attempted to its children and then to it.

This procedure allows the same cluster to be fully or partly mergeable with two different clusters. However, only the merging operation with the greatest similarity value is executed at each loop. The rest of mergeable clusters will be again compared to the new merged cluster in future loops of the algorithm.

Concatenation of Single Childs After a large number of processed trajectories, some tree nodes might have a single child subtree. This can be produced either because of tree prunning, sibling merging, or because similar trajectories ended at different points and produced tail clusters. These clusters are concatenated to their parents in order to obtain a single cluster.

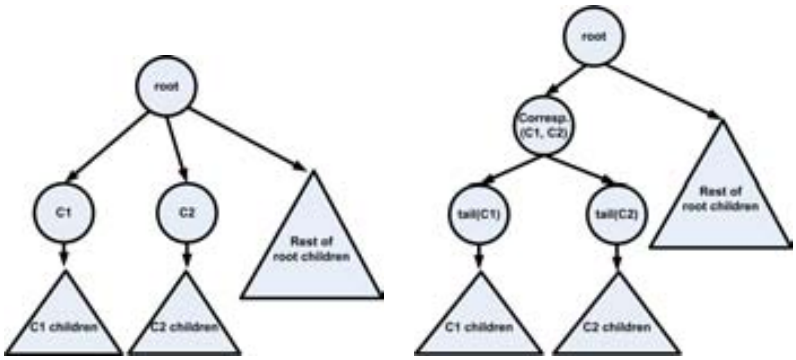


Figure 3.6: Cluster–to–cluster correspondence. (a) Before reestructuration, correspondence between C_1 and C_2 is computed. (b) Merging is performed and the tails are assigned to their children.

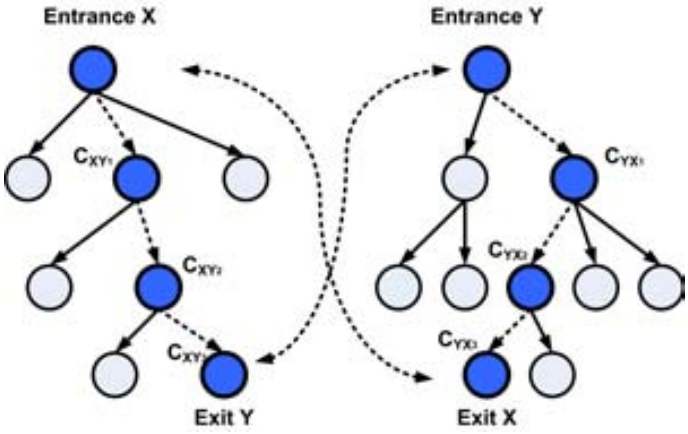


Figure 3.7: Finding double way paths.

Cluster Relation Graphs

Given a cluster tree T, its root node C_r is an *entrance* point to the scenario and the leaf nodes of T contain the *exit* points from the scenario. Thus, each sequence of nodes from the root to leaf nodes defines a *path* in the scenario from one entrance point to one exit point. However, due to inherent features of human behavior within a scenario, entrance points in one cluster tree can be exit points of another cluster tree, and viceversa. Thus, given an entrance point A and an exit point B, the path from A to B is defined as the sequence of clusters beginning in A and finishing in B. Next, entrance points are compared to the exit points in order to obtain double way paths, see Fig. 3.7. Therefore, the goal of this procedure is to establish a global relation between entrance and exit points from the scenario, so that, given two entrance/exit

points A and B, there is a single path to go from A to B. To this end, these cluster trees are fused into a *Cluster Relation Graph* (CRG) $G = (V, E)$ where $V = \{v_i\}$ contains spatial portions of the scenario and $E \subseteq V \times V$ defines the continguity relations between portions of the scenario.

The procedure explained in this section generates a cluster relation graph by fusing the set of cluster trees obtained during the learning step. Although the model can be generated at any time since the first trajectory is clustered, the accuracy and representativity of the model depends on the number of clustered trajectories and the maintenance operations performed so far. As the number of processed trajectories increase, outlier clusters (representing abnormal trajectories) can be more easily identified and removed from the cluster set.

The CRG is initialized by transforming each entrance/exit point of the scenario into a vertex. To this end, entrance points are found by taking the root nodes of each cluster tree, and exit points are obtained by taking the last position of leaf nodes in the cluster tree.

Given two entrance points X and Y, represented by two root nodes, the sequence of clusters that form the path between X and Y, ($S_{X\to Y}$ and $S_{Y\to X}$), are searched and their correspondence is computed.

$$\begin{aligned} S_{X\to Y} &= C_{XY_1}, \ldots, C_{XY_n} \\ S_{Y\to X} &= C_{YX_1}, \ldots, C_{YX_m} \\ path(X,Y) &= \{X, V_1, \ldots, V_k, Y\}, \qquad k = max\,(n,m) \end{aligned} \tag{3.14}$$

The path between X and Y in the relation graph contains as many vertices as the maximum number of clusters in the sequences $S_{X\to Y}$ and $S_{Y\to X}$. When the number of clusters is different, e.g. $\|S_{X\to Y}\| > \|S_{Y\to X}\|$, some clusters of $S_{Y\to X}$ will be splitted in order to merge with one cluster of $S_{X\to Y}$.

Fig. 3.8 shows the vertex creation. The example scenario contains three entrance points $\{A, B, C\}$ and the paths between them are formed as follows:

$$\begin{aligned} S_{A\to B} &= \{C_1, C_2, C_3\} \\ S_{B\to A} &= \{C_7, C_6, C_5\} \\ S_{A\to C} &= \{C_1, C_2, C_4\} \\ S_{C\to A} &= \{C_{13}, C_{12}, C_{10}, C_9\} \\ S_{B\to C} &= \{C_7, C_8\} \\ S_{C\to B} &= \{C_{13}, C_{12}, C_{10}, C_9\} \end{aligned}$$

The procedure starts by processing $S_{A\to B}$ and $S_{B\to A}$. Since the maximum number of clusters is 3, there will be a sequence of 3 vertices $\{v_1, v_2, v_3\}$ between A and B. Each vertex represents a list of clusters representing the same region of the scenario. To this end, the correspondence between each pair of clusters (C_i, C_j), $C_i \in S_{A\to B}$, $C_j \in S_{B\to A}$ is computed to find which cluster in $S_{A\to B}$ is most similar to which

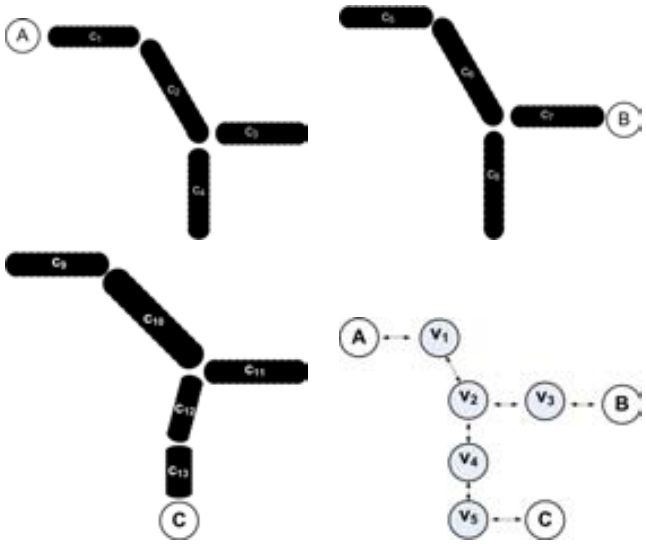


Figure 3.8: Example of Cluster Relation Graph creation. (a) Tree for the entrance point A (b) Tree for the entrance point B (c) Tree for the entrance point C (d) Resulting conceptual graph.

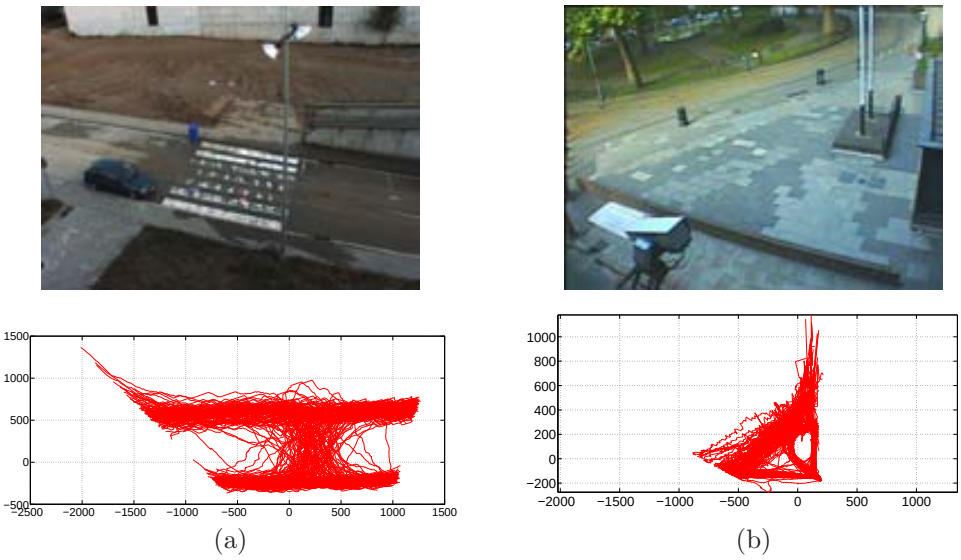
cluster in $S_{B\to A}$.

$$\begin{aligned} v_1 &= (C_1, C_5) \\ v_2 &= (C_2, C_6) \\ v_3 &= (C_3, C_7) \end{aligned}$$

As seen above, each pair of correspondent clusters creates a new vertex in the graph. Next, the procedure continues by processing $S_{A\to C}$ and $S_{C\to A}$. In this case, A and C are separated by 4 vertices, see Eq. 3.14. After computing the correspondence measure, the highest correspondence for clusters C_{12} and C_{13} is found to be C_4. Consequently, C_4 is splitted into C_{4_1}, C_{4_2} and thus $S_{A\to C} = \{C_1, C_2, C_{4_1}, C_{4_2}\}$. On the other hand, C_1 and C_2 are merged into already existing vertices:

$$\begin{aligned} v_1 &= (C_1, C_5, C_9) \\ v_2 &= (C_2, C_6, C_{10}) \\ v_3 &= (C_3, C_7) \\ v_4 &= (C_{4_1}, C_{12}) \\ v_5 &= (C_{4_2}, C_{13}) \end{aligned}$$

Finally, the correspondence between $S_{B\to C}$ and $S_{C\to B}$ is processed. The final vertices list is



(a) (b)

Figure 3.9: Scenarios used in the experiments: (a) Hermes Outdoor (b) Kingston Dataset. The trajectories are shown in ground–plane coordinates after applying the multistep algorithm by Makris et al [67], which erases noisy trajectories from stationary motion noise and tracking failures.

$$
\begin{aligned}
v_1 &= (C_1, C_5, C_9) \\
v_2 &= (C_2, C_6, C_{10}) \\
v_3 &= (C_3, C_7, C_{11}) \\
v_4 &= (C_{4_1}, C_{12}, C_{8_1}) \\
v_5 &= (C_{4_2}, C_{13}, C_{8_2})
\end{aligned}
$$

The resulting relation graph is shown in Fig. 3.8.(d). In this graph, vertices represent the identified portions of the scenario and the edges show the transitions an upcoming trajectory is expected to perform, in order to be considered *normal*. Therefore, such trajectories will be considered as *normal* behavior, and those performing different transitions as *abnormal* behavior.

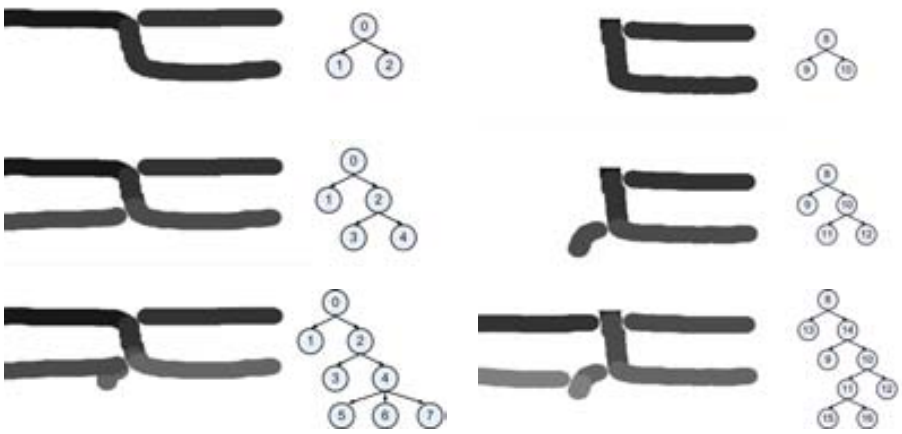
3.2.4 Experiments

The clustering algorithm has been tested in two different scenarios, so–called HERMES Outdoor and Kingston Dataset, see Fig. 3.9. These scenarios comprise several difficulties:

1. No phisical limitations to the possible set of trajectories. Human agents can walk almost in all the points of the scenario.

(a) (b)

Figure 3.10: Example of tracking results.



(a) (b)

Figure 3.11: Evolution of the cluster tree corresponding to entrance points in the HERMES Outdoor scenario. The gray level indicates the depth level in the tree, being the root of the tree painted in black. The original cluster identifiers have been changed due to clarity reasons. (a) Entrance point 1. (b) Entrance point 2.

2. No predefined behavior patterns are supposed to be followed.

3. Human trajectory performance depends on other agents moving at the same time, like other pedestrians or vehicles.

A single calibrated monocular camera has been used to acquire image data and real–time clustering has been implemented using MATLAB. Executions were performed using a Pentium 4 processor with 2GB of RAM memory and running at 3Ghz.

Clustering

For the HERMES Outdoor scenario, 223 trajectories have been tracked and clustered. Similarly, 120 trajectories were considered for the Kingston Dataset. The target po-

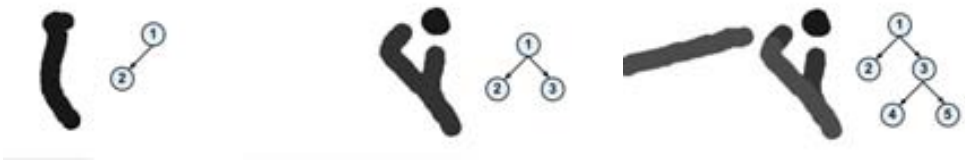


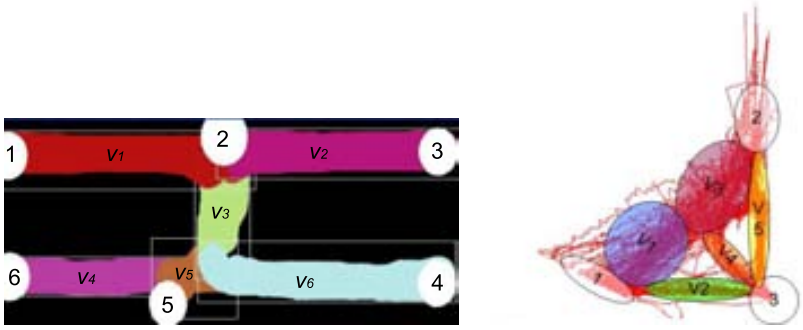
Figure 3.12: Evolution of the cluster tree corresponding to entrance points in the Kingston Dataset scenario. The gray level indicates the depth level in the tree, being the root of the tree painted in black. The original cluster identifiers have been changed due to clarity reasons.

sitions have been processed by the time they were returned from the tracking system, whose results are shown in Fig. 3.10. To this end, the parameters σ, α, and δ have been tuned to cope with the human charasteritics, the tracking system performance and the dimension units of the scenario. The variance value to construct the initial covariance matrices has been assigned $\sigma^2 = 100$, which assumes a standard deviation of 10 pixels in each observation on a 1392×1040 image plane. This variance is thus growing or diminishing depending on the actual position in ground plane coordinates. Each cluster node is initially weighted as representing $N = 100$ observations, so each update is a sum of gaussians with a low update rank. This is an appropiate because since lower updates made no effect on the cluster and higher values made some cluster nodes to be updated continuously and growed to match every trajectory in the scenario. The rest of the parameters used, $\delta = 0.5$ and $\tau = 5\%$ were finally chosen after testing different ranges of values.

The maintenance of the cluster database, i.e. prunning, concatenation, and merge, has been applied after each 30 processed trajectories. As a result, clusters produced by abnormal trajectories like the one shown in Fig. 3.10.b) have been removed from the cluster database. Fig. 3.11 shows the evolution of the cluster database for the cluster trees related to two entrance points of the Hermes Outdoor scenario, after several maintenance processes. Similarly, the evolution of the cluster tree represented by a entrance point in the Kingston Dataset Scenario is depicted in Fig. 3.12.

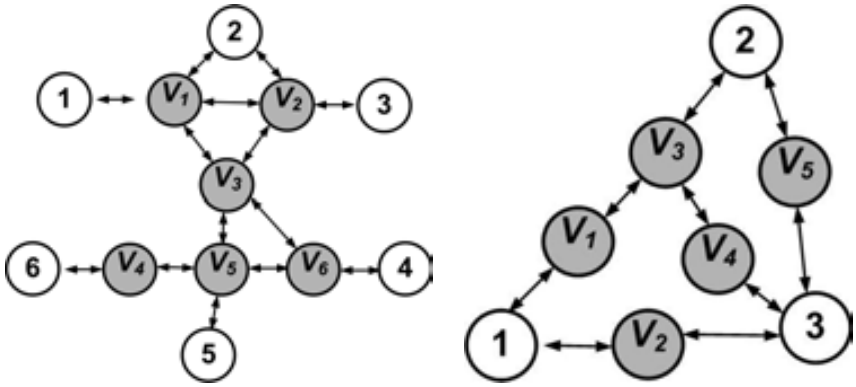
Conceptual Knowledge

Fig. 3.13 shows the obtained CRGs for the HERMES Outdoor and Kingston Dataset scenarios. Also, a graph representation of the resulting CRGs is depicted in Fig. 3.14. For the Hermes Outdoor scenario, the resulting CRG contains 12 vertices, being 6 of them entry/exit points of the scenario. As a result of the double way path computation, all the transitions between vertices have finally become bidirectional. By comparing the graph with the original scenario, we can assess the semantic labels of the vertices, being $\{v_1, v_2, v_4, v_5, v_6\}$ the *sidewalk* regions and v_3 the *crosswalk*. Note that the resulting conceptual model establishes a strong separation between the entry points $1, 2, 3$ and $4, 5, 6$, and normal behavior implies to cross v_3, i.e. the crosswalk, to change from one sidewalk to another. Another remarkable conclusion from this result is that at first glance the scenario seems to be simmetric. However, trajectories



(a) (b)

Figure 3.13: Cluster representation of the CRG obtained from the fusion of the learned cluster tree set. The results are shown in ground–plane coordinates for the two scenarios: (a) Hermes Outdoor (b) Kingston Dataset.



(a) (b)

Figure 3.14: Graph representation of the CRGs obtained from the fusion of the learned cluster set for the two scenarios: (a) Hermes Outdoor (b) Kingston Dataset.

moving from any position inside either v_1 or v_2 to v_3 are considered normal, v_4 has no direct transition to v_3, so trajectories performing this transition are considered abnormal. This fact is determined by the trajectories considered during the learning step, and their order of appearence.

Note that the resulting structure models possible paths not existing in the training set, for example the path from the entry point 1 to the exit point 6.

On the other hand, the CRG obtained for the Kingston Dataset scenario contains three entry and exit points, and five internal vertices, which clearly identify the normal routes to pass through the scenario. However, this scenario lacks of semantically diferentiated zones, since almost all the ground plane is a pedestrian environment. Although each node has bidirectional relationship with its neighbors, not all possible combinations of neighbor vertices describes a normal path in the scenario. For

Scenario	# Clusters bef.	# Clusters aft.	% Obs.	% Correct paths
Hermes Outdoor	35.4 ± 10.3	11.5 ± 2.5	74.32 ± 7.2	86.5 ± 2.3
Kingston dataset	134.3 ± 8.6	8.4 ± 2.1	62.5 ± 11.3	82.5 ± 5.2

Table 3.1
Evaluation of the size of the cluster database and the subsequent CRG for a set of 150 experiments in each scenario.

instance, the path $(1 \rightarrow v_1 \rightarrow v_3 \rightarrow v_4 \rightarrow 3)$, in the resulting CRG, has not been inferred from the database. Therefore, the resulting CRG allows to recognize two types of anomalies from incoming trajectories:

- Trajectories that perform a path represented in the CRG but not in the database, e.g. the path $(1 \rightarrow v_1 \rightarrow v_3 \rightarrow v_4 \rightarrow 3)$.
- Trajectories that perform a path which results in an unobserved sequence of vertices of the CRG. For instance, the path $(1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow 2)$ is not a modeled path in the CRG, since there is not neighborhood connection between v_2 and v_4.

A quantitative analysis of the clustering and conceptualization is reported in Table 3.1. This experiment has been done by randomly choosing 150 permutations of the each dataset and simulating an on–line development of each permutation of trajectories. The objective of this experiment is to analyze how the order of the trajectories affect to the resulting cluster database, and how the cleaning procedures described in this work help to reduce the size of the database. Finally, the experiment compares the results obtained for each scenario.

Although the amount of clean trajectories is similar in each scenario, the experiment showed that the noise contained in the Kingston Dataset produced a very high number of clusters, since stationary motion trajectories never followed any of the existing cluster trees. However, the cleaning procedures, specially the noise removing procedure described in previous section, considerably reduces the number of outlier clusters.

The last two columns of Table 3.1 reports a comparison between the obtained representations and a manually annotated ground truth of the conceptual model, for each scenario. To this end, the number of entry, exit and paths have been annotated and a set of points for each path have been selected. The column *% Observations* contains the average number of points that were correctly assigned to existing paths in the database. Finally, the column *% Correct paths* compares the obtained path list with the manually generated ground truth. The results of these two values are justified as follows: The point comparison column gives better classification rate to the Hermes Outdoor scenario, since the paths are very stilized and hardly overlap, while the Kingston Dataset represents a set of very overlapped paths, producing errors when classifying each point of the experiment. On the other hand, the number

of paths correctly detected in the approach have been similar, since the previously described disadvantage is corrected because the Kingston Dataset scenario is conceptually simpler than the Hermes Outdoor scenario.

3.2.5 Discussion

The presented method constructs a conceptual model by means of on–line trajectory clustering. The main advantage of this approach is the capability to work with incomplete trajectories, which allows to get intermediate results and avoids preprocessing a trajectory set. The method has shown good results in stylized scenes where paths are well structured. This is due to the observed deviation from the trajectories of the same path is minimized and, therefore, a small set of clusters is generated out of them. For instance, a roadway scene involving vehicles can be accurately modeled using this technique, since it covers with the desirable requirements. Firstly, roads establish well-defined and restricted paths. Secondly, vehicles represent rigid objects and therefore their behavior is mostly defined by their speed and orientation.

As long as the scenario complexity rises, specially in human–populated scenarios, the number of cluster might be high since there is no clue about the final development of a trajectory given the observations time–up–to–now. Knowing the ending area beforehand would allow to relate the trajectory with any of the current paths, thereby avoiding the creation of duplicated clusters. The following sections propose an alternative approach to generate a scenario model from a set of existing trajectories, in which the sets of entry and exit areas are learnt for modeling typical routes within the scenario.

3.3 Off–line Scenario Modeling

In this section we present our proposed algorithm to create routes from the previously described trajectory set T'.

3.3.1 Scenario Model

A *route* $R_{s,e}$ between two areas $(s, e) \in S \times E$ is defined as a *normal* way to go from s to e and is represented by a Gaussian Mixture Model (GMM):

$$R_{s,e} = \{G_1, \ldots, G_k\} \tag{3.15}$$

Due to several reasons, such as scene restrictions or speed variations, there could be more than one route between a pair $(s, e) \in S \times E$. A *path* $P_{s,e}$ contains the routes from s to e:

$$P_{s,e} = \{R^1_{s,e}, \ldots, R^U_{s,e}\} \tag{3.16}$$

Finally, our scenario model consists of the sequence of paths between each pair $(s, o) \in S \times E$:

$$\mathcal{M} = \{P_{s,e} | s \in S, e \in E\} \tag{3.17}$$

3.3.2 Clustering Entry and Exit Areas

A region r of the scenario is considered to be an *entry* of *exit* area if trajectories begin or end in that points. To this end, the first positions of the trajectory set are clustered to find the entry areas, as explained as follows.

The objective is to find a set E which contains the start points to the scenario. The fact that the exact number of entry areas is unknown beforehand discourages the use of parameter–specific clustering algorithms like k–means, although research done in that subject [29]. In order to avoid specifying the number of clusters, two different approaches have been tested.

Initially we have used the *Quality Threshold* (QT) clustering algorithm [51] in a similar way to [9]. The QT clustering is based on establishing the maximum distance between points of the same cluster. In our case , since the scenario has been calibrated, the distances between entry areas are expressed in meters. Using this information, we set a distance constraint D, the maximum distance allowed between two entry areas to belong to the same cluster.

$$\begin{aligned} S &= \{s_1, \ldots, s_k\} \\ s_i &= \{t_1, \ldots, t_m\}, \forall_{t_{p_1}, t_{q_1} \in s_i} dist(t_{p_1}, t_{q_1}) \leq D \end{aligned} \tag{3.18}$$

Following Eq. (3.18), the QT-clustering algorithm divides the training trajectory set T into k disjoint subsets. For each subset s_i, the positions must be separated by a maximum distance D (called *complete linkage* clustering). Afterwards, a centroid is computed for each subset s_i:

$$\begin{aligned} C_S &= \{c_{s_1}, \ldots, c_{s_k}\} \\ c_{s_i} &= \sum t_j \in s_i / k \end{aligned} \tag{3.19}$$

However, this measure requires that the trajectories are estimated over a common ground plane, which may not be realistic assumption in some scenarios. Furthermore, the QT clustering is too much demanding in computing time, and this issue encouraged to look for a different approach. The same procedure is followed to find the exit areas, but using the last positions of the trajectory set.

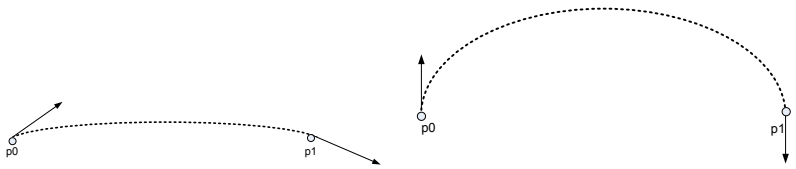
Subsequently, the sets of entry and exit areas have been estimated using the approach by Makris et al. [67], as explained previously in Section 3.1, see Fig. 3.15.

3.3.3 Smoothing Trajectories

Depending on the accuracy of the tracking algorithm and also the scenario conditions, trajectories might lack of smoothness, being irrealistic representations of the actual target motion over the scene. This problem is solved by representing each trajectory of T' using a continuous function model that allows to get rid of the inaccuracy coming from the tracking system. In order to solve such a problem, trajectories in T' are converted into a spline representation [30]. The main advantage over the initial structure of the trajectory is that the spline acts as a continuous function which takes values from $[0 \ldots 1]$, allowing to sample points with any required precision.

(a) (b)

Figure 3.15: Detected entry and exit areas from (a) Kingston Dataset and (b) HERMES Outdoor.



(a) (b)

Figure 3.16: The speed vectors completely determine the curve of the spline between the points p_0 and p_1. In these two samples it can be seen how to obtain different splines by just varying the derivative vectors.

The spline representation of a trajectory $t = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ has to deal with the features of human motion, i.e. complex curve shapes that might be modeled. Hence, a single spline is not enough to model all types of trajectories. Instead, we use a concatenation of simple splines, or *B-splines*. Thus, the spline representation of t is called $sp(t)$, being $sp(t) = \{bsp_1, \ldots, bsp_d\}$ the sequence of d control points.

In order to ease the notation, let us consider that one b–spline models a segment of a trajectory, e.g $\{x_p, \ldots, x_q\}$. The construction of the b–spline needs of two control points p_0 and p_1, which coincide with the starting and ending point of the spline curve. However, the curvature of the spline is determined by the *speed* in the entrance and exit of the spline, see Fig. 3.16. This speed is represented by two normalized vectors v_0 and v_1, i.e. the derivatives of the spline at p_0 and p_1, see Fig. 3.17(b). In the figure, the vectors v_1 and v_2 are obtained by joining p_0 with the next two positions in the trajectory. The derivative obtained for p_1 at one b–spline will be the same as that for p_0 in next b–spline. This allows us to obtain a continuous and smooth concatenation of b–splines.

Let $T_{ij} = \{\tau_1, \ldots, \tau_q\}$ be the trajectories from T that begin in the entry area s_i

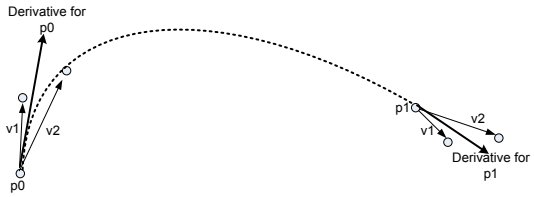


Figure 3.17: Computing the derivatives for a simple b–spline. The average of the vectors formed by p_0 and the subsequent trajectory points are normalized to obtain the derivative of the spline in p_0. The same is done for p_1, and that derivative is kept equal for the next spline.

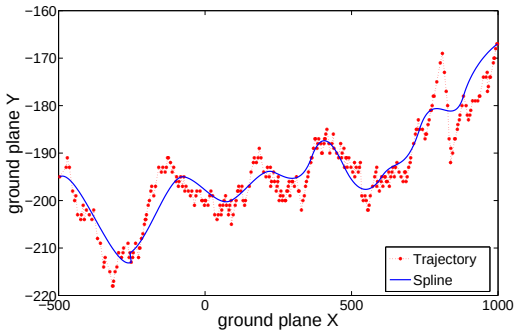


Figure 3.18: Example of trajectory and its spline representation.

and finish at the exit area e_j. Control points will maintain a spatial and temporal consistency between splines. To this end, splines are created using K control points, being K fixed for all the trajectories in the same T_{ij}. Fig. 3.18 shows an example trajectory and its conversion to the spline representation.

The required number of cubic splines is automatically decided by computing the error between the original trajectory and the computed spline sequence:

$$error(t, \mathrm{s}(t)) = \sum_{(x,y)\in t} d((x,y), \mathrm{s}(t)) \tag{3.20}$$

where d computes the minimum Euclidean distance between (x, y) and a set of points sampled from $\mathrm{s}(t)$.

This representation allows to sample intermediate points with any precision while maintaining the temporal consistency of the original trajectory. Moreover, this representation only requires to store the spline information, i.e. the intermediate control points and derivatives, reducing disk storage demand when the trajectory set is very large.

3.3.4 Learning algorithm

The following procedure is applied for each pair $(s, e) \in S \times E$. Let $T_{s,e} = \{t_1, \ldots, t_N\} \subseteq T'$ be the set of trajectories starting at s and ending at e. Each trajectory t_i is repre-

sented by a sequence $r(t_i)$ of K equally spaced control points, obtained by sampling from $s(t_i)$:

$$r(t_i) = \{p_1^i, \ldots, p_K^i\} \tag{3.21}$$

where p_k^i corresponds to the sample point k/K of $s(t_i)$.

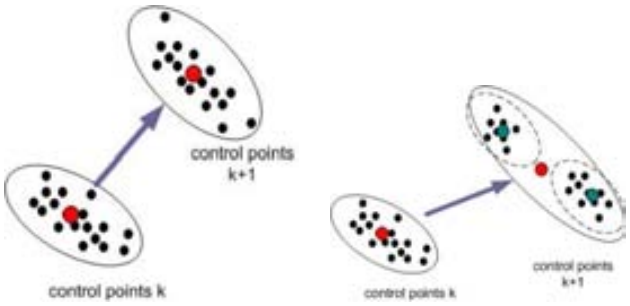
The learning algorithm shown in Algorithm 1 produces a progressive splitting of the trajectories, in order to create those routes $R^1 \ldots R^m$ that best describe the path between a pair of entry and exit areas.

Algorithm 1 Route Modeling Algorithm for a pair $(s, e) \in S \times E$

```
Initialize P_{s,e} with a single route R^1
for k = 1 to K do
  nc ← |P_{s,e}|
  for c = 1 to nc do
    list ← points(k, c)
    G ← GMM(list, 1)
    (G_1, G_2) ← GMM(list, 2)
    if δ(G) > (δ(G_1) + δ(G_2)) * α/2 then
      create R^{c1}, R^{c2} from R^c
      add G_1 to R^{c1}
      add G_2 to R^{c2}
      split points(k, c) according to G_1 and G_2
      substitute R^c with R^{c1}, R^{c2} in P_{s,e}
    else
      add G to R^c
    end if
  end for
end for
```

The input of the algorithm consists of the set of trajectories starting in an entry point $s \in S$ and ending in an exit point $e \in E$. These trajectories are represented in a $K \times N$ matrix, where N is the number of trajectories and K is the number of control points that have been sampled for each trajectory, given the spline representation explained before. Each trajectory is associated to one of the routes that form the path $P_{s,e}$. This is represented by $points(k, c)$, which contains the list of k–th points of the trajectories being currently associated to the route R^c. As an initialization, the algorithm considers that all the trajectories are following the same route R^1.

Then, the matrix is traversed row–wise, from $k = 1$ to K, considering at each loop the current routes and the k–th control points of the trajectories associated to each route. Given a route $R^c \in P_{s,e}$, the k–th control points of each associated trajectory are obtained by $list = points(k, c)$. Now, the number of gaussian components that best represent *list* must be decided. In order to simplify the algorithm, the assumption that routes will be separated in at most two sub–routes has been taken, although there may be better representations in particular scenarios using a higher number of components. Thus, *list* is modeled using a one–component GMM and a



(a) (b)

Figure 3.19: Possible situations in the learning algorithm. (a) The density of a single gaussian is enough to represent the $k-th$ set of control points. (b) Two gaussians represent better the control points and then the current route is split into two subroutes.

two–component GMM. In order to decide which model better represents the data, the degree of occupation inside the components is measured by means of a density criterion:

$$\delta(G) = \frac{w}{\pi \cdot \sqrt{|\Sigma|}} \tag{3.22}$$

where w and Σ are the prior probability and the covariance matrix of G, respectively. If the mean density at the two components GMM (G_1, G_2) is higher than the density at G, bounded by a factor α, then the current route R^c is splitted into two new routes R^{c_1} and R^{c_2}, see Fig. 3.19. Also, the trajectories that were associated to R^c will be now associated to one of the new routes, depending on the proximity of their k–th control points to each of the components G_1 or G_2. Since in the current loop cycle the route R^c has been split into two new routes, R^c is replaced by R^{c_1} and R^{c_2} in the remaining of the loop.

After applying the algorithm to each pair of entry and exit areas, $\mathcal{M}$ contains the set of normal paths that trajectories should pass in the future. Thus, a trajectory deviating from $\mathcal{M}$ will be considered as an anomaly. However, current distinction between *normal* and *anomalous* trajectory can be enhanced by recognizing different types of anomalies, as explained next.

3.4 Anomaly Detection

Nowadays, the detection of anomalies in video sequences is considered a hot topic in video understanding research [47]. This issue is caused not by the difficulty of implementing an anomaly detector, but because it is unclear which is the best definition of *anomaly*. On the one hand, the concept of anomaly is usually related in video–surveillance to suspicious or dangerous behaviors, i.e. those for which an alarm should be fired when detected. Unfortunately, concepts like *dangerous* are very difficult to automatically learn without the explicit help of a–priori semantic knowledge about

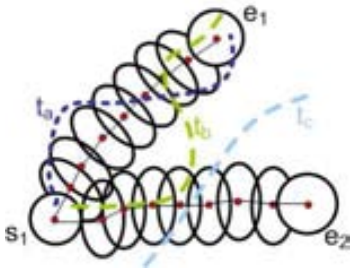


Figure 3.20: Schema of the types of anomalies detected in our approach. The trajectory t_a performs a well–known route from s_1 to e_1, however it is marked as SA since some parts fall out of the route. t_b performs a route from s_1 to e_1. However, the first half of the trajectory was directed towards e_2, so it has been labeled as IA. Finally, t_c is very far from following any of the existing routes, so it is labeled as HA.

the context. Existing top–down approaches take advantage of this a–priori knowledge to identify suspicious behavior [22] or to generate conceptual descriptions of the detected behavior patterns [7]. However, these approaches are scenario oriented and are designed to recognize or describe a reduced set of specific behaviors.

On the other hand, from a statistic point of view, normal behavior occurs more frequently than anomalous behavior. This is the main assumption of all recent works performed in anomaly detection research [13, 52, 67, 83]. Since an anomaly is a deviation from what is considered normal, these two concepts, normality and anomaly, are complementary. In the video–surveillance domain, the standard learning procedure uses observations extracted by a motion tracking algorithm over a continuous recording to build a model of the scenario that will determine somehow the normality or abnormality of new observations.

3.4.1 Taxonomy of Anomalies

Our proposal first focuses on the definition of anomalies in terms of a hierarchy of deviations from a previously learnt model. In this work we differentiate among three semantically different anomalies, namely *Soft* (SA), *Intermediate* (IA), and *Hard* (HA) anomaly. Each anomaly is related with a level of deviation between the learnt model $\mathcal{M}$ and new trajectory observations. In essence, a SA represents a slight deviation from the parameters of a learnt pattern inside the model, e.g. a person running, stopping, walking backwards for a while, etc. On the other hand, a HA is the observation of a event which is completely outside the model, e.g. a person appearing from an unobserved entry point. These two anomalies have been recognized using different approaches, see Table 2.1 in Chapter 2, but there still exists a semantic gap between them. To fill this gap, the IA represents the observations that deviate from any learnt pattern but they are still fit inside the model, e.g. a person that, walks from A to B, but somewhen changes to C.

Summarizing, we differentiate three kinds of anomaly with respect to the current trajectory over the scenario, see Fig. 3.20:

- *Soft Anomaly* (SA): Some parts of a trajectory are classified as SA if they follow a modeled path, but there are sudden changes in speed or orientation that differ from the learnt routes from s to e.

- *Intermediate Anomaly* (IA): A trajectory is classified as IA the most probable path $P_{s,e}$ has changed during the trajectory development for a whole window ω.

- *Hard Anomaly* (HA): A trajectory is classified as HA if it has performed a completely unobserved path. This can be caused because the trajectory started from a entry point $e' \notin E$ or because the probability of any path beginning in the actual entry point is too low for the whole window ω.

This description of anomalies represent different degrees of deviation between a new observation and the learnt model. Indeed, a SA can be considered as a route deviation inside a path. A IA detects path deviations inside the same model $\mathcal{M}$. Finally, a HA represents a complete deviation from $\mathcal{M}$. Note that this representation could be extended if there were several models $M_1, \ldots, M_m$ instead of a single one.

3.4.2 On–line labeling of trajectories

Trajectories are on–line classified in one of the four considered classes, namely *normal*, *SA*, *IA*, and *HA*, as long as new observations are obtained by the tracking system. In our method, trajectories do not need to be completely observed before deciding about its normality or abnormality. However, this on–line classification enables to modify the classification of a trajectory over time. On the other hand, a given trajectory may not be classified with a single label; it could have *normal* and *anomalous* parts. These parts will be referred in the following as trajectory segments.

The assignment of labels is based on a measure of the probability for a trajectory to belong to the what is considered a *normal* behavior. In that case, the model $\mathcal{M}$ obtained by the method in Section 3.3 is used. However, the proposed hierarchy of anomalies is not dependant on the precise model, and could be evaluated using alternative approaches.

Firstly, the prior probability that a pixel location (x, y) is a part of a modeled path, $p(P_{s,e}|(x, y))$, is obtained by computing a probabilistic map of the image plane. These probabilities are stored in a $\mathrm{H} \times \mathrm{W} \times |\mathcal{M}|$ matrix Θ, where $\mathrm{H} \times \mathrm{W}$ are the number of pixels of the image and $|\mathcal{M}|$ is the number of paths:

$$\Theta_{x,y} = p(P_{s,e}|(x, y)) = max\left(p(G_j|(x, y)\right) \tag{3.23}$$

where G_j belongs to some route $R^u_{s,e} \in P_{s,e}$.

The information provided by a single observation can be interpreted in several ways, since it does not carry information about past development. This problem is tackled by assuming correlation with the previous frame steps and the system maintains a temporal window ω with the last $|\omega|$ observations . Thus, the probability for a trajectory to be in a given path is computed recursively, by combining the prior

probability described in Eq. (3.23) with the probabilites computed in previous frame steps:

$$p(P_{s,e}|\Theta,\omega,(x,y)) = \alpha * \Theta_{x,y} + (1-\alpha) * \frac{\sum_i^{|w|} p(P_{s,e}|\Theta,w_i)}{|w|} \tag{3.24}$$

where w_i represents the i–th observation in ω and α is an appropiate update factor.

The on–line anomaly detection algorithm is explained next. When a new trajectory t is detected, the initial observation is compared to the entry areas in S, previously learnt. If there is no entry point close enough, then t is labeled as HA, since there is no path in $\mathcal{M}$ to explain this observation. Otherwise, assuming some $s \in S$ as the entry area of t, the probabilities for the trajectory to follow paths beginning in e are computed using Eq. (3.24) for each new observation (x, y). Since paths sharing the same entry area can be partially overlapped, the probabilities may remain similar for a certain period of time. When this happens, t is not assumed to be following any of the paths. However, since all the possible paths remain inside $\mathcal{M}$, the trajectory is labeled as *normal* while this condition holds.

When paths split towards different directions, after $|\omega|$ frames a path $P_{s,e}$ may eventually get a high probability compared to the others. In such a case, the trajectory will be marked to be following $P_{s,e}$ for the next frames. Otherwise, if the trajectory takes a direction which makes all the paths to have a low probability, this means that the trajectory is not performing any of the modeled paths, and it will be marked as HA after $|\omega|$ frames. This situation will hold unless the trajectory *corrects* its development and returns back to a known path, which will increase its probability. If the trajectory keeps developing normally inside the path, the rest of the trajectory will be labeled as *normal*.

Let us consider that the trajectory is developing normally inside a path $P_{s,e}$, which means that the probability of being in $P_{s,e}$ is very high compared to any other path that started in s. Suppose that probabilities become similar again. While this situation holds, uncertainty about final trajectory development is high, but the trajectory is kept to be following the previous path. However, if another path $P_{s,e'}$ gets a dominant probability, then it is unclear which segment of the trajectory will be finally the anomalous one, and thus both anomalies must be considered. The trajectory segment where $P_{s,e}$ was dominant is labeled as IA, and so does the segment starting at that particular time step. The uncertainty about the actual anomaly will not be solved until the trajectory ends in either e or e'. If the trajectory finally ends in e' then the only segment labeled as IA will be that in which $P_{s,e}$ was dominant. On the other hand, if $P_{s,e}$ gets again the dominant probability and the trajectory finally ends up in e, then the segment where $P_{s,e'}$ will be labeled as IA.

Now let us assume that the trajectory is inside a path $P_{s,e}$, avoiding hard and soft anomalies. We select the gaussian G_{j^*} that better represents the current observation:

$$j^* = \underset{j}{\text{argmax}}\ p(G_j|(x,y)) \tag{3.25}$$

In order to check whether the trajectory is developing normally given the current path $P_{s,e}$, we compute the sum of increments of the probability that ω is performing a

path between G_{j^*-1} and G_{j^*+1}, which are the previous and the next Gaussian Model of G_{j^*}, respectively. This information is encoded into a *normality* factor $F(\omega)$:

$$F(\omega) = \begin{cases} \sum_i^{|\omega|} -\Delta p(G_{j^*-1}|\omega_i) & \text{if } h(G_{j^*}, \omega) > 0 \\ \sum_i^{|\omega|} \Delta p(G_{j^*+1}|\omega_i)) & \text{otherwise} \end{cases} \tag{3.26}$$

where $\Delta p(G|\omega_i) = p(G|\omega_i) - p(G|\omega_{i-1})$, and $h(G_{j^*}, \omega)$ measures the direction of ω towards G_{j^*}:

$$h(G_{j^*}, \omega) = p(G_{j^*}|\omega_{|\omega|}) - p(G_{j^*}|\omega_1) \tag{3.27}$$

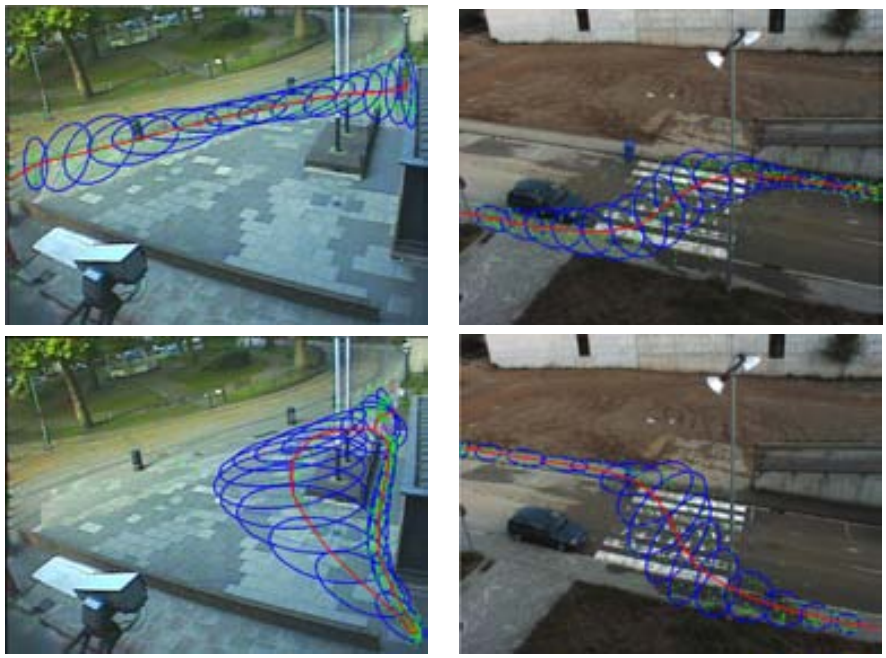
A positive value of $F(\omega)$ means that the last ω frames represent a normal sequence of observations given the model $\mathcal{M}$. Otherwise, the trajectory is slightly deviating from the model parameters, so we annotate subsequent observations as SA until the trajectory recovers from the deviation. Note that the value of $F(\omega)$ is only representative when the trajectory is following a path, i.e., the highest path has kept stable during the whole last ω. Hence, if the trajectory is labeled as IA or a HA, the value of $F(\omega)$ does not provide any extra information.

3.4.3 Experiments

In this section we show the performance of the offline learning approach by analyzing the HERMES Outdoor and Kingston Dataset scenarios, which have been previously introduced in Section 3.2.4. In previous sections we have shown how the initial trajectory dataset T has been refined by learning the entry and exit areas of the scenario, so a subset T' has been obtained. Moreover, we have used a spline representation of the trajectories in T'.

For our experiments we have tested a wide set of values for the parameters K, the number of control points per path, and ω, the size of the temporal window. In the following results we have used the values $K = 10$ and $|\omega| = 20$. In Fig. 3.21.(a), the results for two paths are depicted. Moreover, Fig. 3.21.(b) depicts the probabilistic maps Θ for the modeled paths. Finally, the resulting model $\mathcal{M}$ is depicted in Fig. 3.22.

Regarding anomaly probabilities, a path P_1 is considered to be qualitatively more probable than another path P_2 if $p(P_1|\Theta, \omega, (x, y)) > 1.5 * p(P_2|\Theta, \omega, (x, y))$. Finally, in order to consider a trajectory as HA, we have set that all paths should have a probability less than 0.5. In Fig. 3.23 we show four examples of different result. The first column shows the trajectory in the image plane, being the orientation marked by the black arrow. The second and third columns show the evolution of $p(P_{s,e}|\Theta, \omega, (x, y))$ and $F(\omega)$, respectively, over time for the possible paths given the computed entry point. The case (a) represents a trajectory without any anomaly, since it has a single high probable path over all the trajectory, and $F(\omega)$ keeps greater than zero. The case (b) represents also has a single high probable path over all the trajectory, however $F(\omega) < 0$ in the interval when the trajectory is going backwards, and this interval is marked as SA. Case (c) is an example of IA, because the most probable path has changed but has kept stable for a long time. Note that it is unclear which part of the



(a) (b)

Figure 3.21: (a) Examples of route models using our learning procedure, depicted in Alg. 1. (b) Graphical representation of the probabilistic maps Θ of the paths.

trajectory (the intervals where $P_{3,2}$ and $P_{3,1}$ are higher) should be considered anomalous. Indeed, both parts will be considered anomalous until the trajectory eventually finishes in exit point 1, when the anomaly will be assigned to the part where $P_{3,2}$ was higher. Finally, case (d) represents a HA, because the trajectory goes far from any modeled path from the detected entry point (2).

3.5 Discussion

This chapter has introduced two approaches towards learning scene models from a initial set of trajectories. The first introduced method performs an on–line learning, updating the model at each frame step given the observations provided by the target tracking system. On the other hand, the second method generates a consistent model of the scenario by dealing with the trajectory set as a whole. As been explained along this chapter, both methods show advantages and drawbacks, although the second method seems to be more convenient, since it performs less assumptions over the scenario structure.

The current bottleneck in learning scenario models in video–surveillance scenarios is the lack of proper training sets, richs in different data types, like those listed in Chapter 1: target, body, and face tracking. Nowadays, the learning is performed

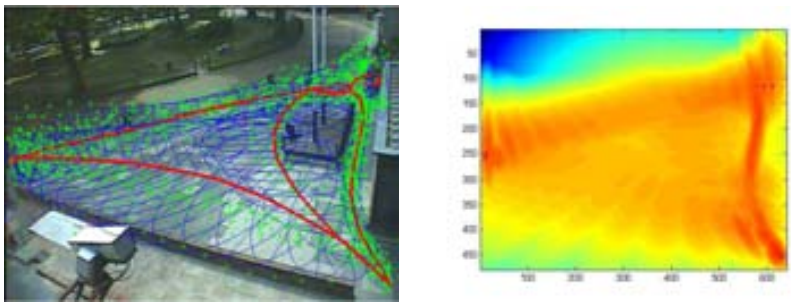


Figure 3.22: (a) Model $\mathcal{M}$ obtained by running Algorithm 1 over the trajectory dataset T' of Fig. 3.1.(d). (b) Graphical representation of the probabilistic maps Θ of the paths in $\mathcal{M}$

using only target trajectories, which are composed by spatial coordinates, speed, orientation and blob sizes. Given these quantitative data, learnt models basically describe information related to the motion pattern performed by some kinds of agents at different locations within the scenario. But, no inference about other factors that affect behavior can be analyzed, e.g. mood, which would help to generate a more detailed description of a new observation.

On the other hand, our proposed methods use quantitative information to learn the model, avoiding the use of a–priori knowledge about the scenario. Designing very generic methods entails a wide applicability over a variate set of possible scenarios. However, the lack of qualitative information prevents from obtaining a high–level, semantic representation of the result. In other words, it is impossible to distinguish two semantic regions if the same motion pattern has been observed during the training period, i.e., the same speed, orientation, etc. But human behavior could be interpreted in a very different way depending on its location. For instance, the sudden stop of a pedestrian could be considered anomalous behavior, since this stop has not been observed in the training period. This fact could be a dangerous situation depending on the semantics of the regions, however the notion of anomaly may not be related to the concept of danger. Thus, the pedestrian stopping on the sidewalk is less dangerous than stopping on the road.

For these reasons, in order to obtain a semantic description of human behavior in video, some prior semantic knowledge should be considered about the types of behavior that are expected to observe in a particular scenario. Next chapter describes an alternative approach to describe and manage prior knowledge, and combine it with quantitative data obtained from vision systems in order to produce high–level reasoning about the contents of an image sequence.

3.6 Resum

Aquest capítol introdueix dos enfocs per a l'aprenentatge de models d'escena a partir d'un conjunt de trajectòries. El primer mètode explicat realitza l'aprenentatge on–

line, actualitzant el model quan arriba una nova observació de la trajectòria d'un agent. D'altra banda, el segon mètode processa un conjunt de trajectòries a la vegada per generar un model consistent de l'escena.

Els models obtinguts representen el conjunt de camins *normals* en un escenari concret. Els camins representen diferents rutes per anar des d'un punt d'entrada a un punt de sortida de l'escenari. A més, les rutes estàn formades per cadenes de models gaussians, que indiquen la trajectòria normal d'un agent que camina per la ruta.

A partir dels models apresos, hem definit una jerarquía de d'anomalies que permeten obtenir una classificació de noves observacions en diferents classes, depenent de la seva proximitat als models apresos. Així, considerem com a *anomalia forta* un agent caminant fora de cap camí après, *anomalia intermitja* a un agent que en algun moment s'ha desviat completament però que finalment ha caminat per punts d'entrada / sortida coneguts i, finalment, una *anomalia lleugera* consisteix en una petita desviació d'una ruta durant un petit interval de temps.

Un dels inconvenients actuals en l'aprenentatge de models d'escena en l'entorn de video-vigilància és la manca de conjunts d'entrenament rics en dades referents els tipus d'inputs descrits a la introducció: target, body i face tracking. Actualment, l'aprenentatge es realitza usant només trajectòries que contenen, com a molt, les posicions, velocitats, orientacions i mides dels objectes que es mouen dins l'escenari. Donades aquestes dades, els models apresos descriuen informació relativa al tipus de moviment que alguns tipus d'agents realitzen a diferents regions de l'escenari. En canvi, no se'n pot fer una inferència sobre altres factors del comportament que ajudarien a generar una descripció més detallada d'una nova observació.

És per això que, de cara a obtenir una descripció semàntica del comportament humà en video, cal aportar un coneixement a–priori que indiqui quines regions semàntiques hi ha a l'escenari i quin tipus de comportament s'espera trobar. El següent capítol discuteix un enfoc alternatiu per a representar el coneixement sobre el comportament humà en un escenari concret per tal d'obtenir descripcions d'alt nivell del contingut d'una seqüència d'imatges.

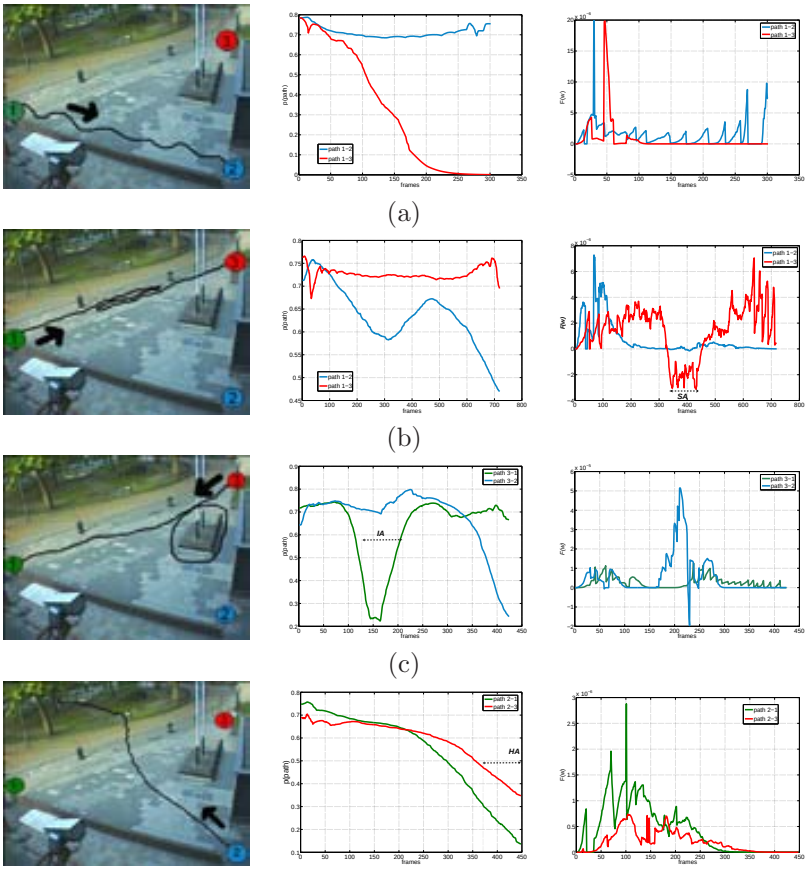


Figure 3.23: Examples of trajectories used for detection of anomalies. (a) Normal trajectory (b) Soft Anomaly (c) Intermediate Anomaly (d) Hard Anomaly. More details in the text.

Chapter 4

Behavior Modeling using Prior Knowledge

Previous chapter showed that current pure learning techniques do not cope with the semantic gap explained in the introduction. The resulting models obtained from bottom–up techniques are quantitatively accurate, but little semantic explanation can be extracted without a–priori semantic knowledge.

Since complex semantics can not be derived from pure statistical analysis, some scenario–dependent *Commonsense Knowledge Base* is required in order to address the semantic gap. This chapter presents a complete framework which identifies the knowledge required for behavior modeling. The framework is composed of a unified data representation scheme, the Fuzzy Metric Temporal Logic (FMTL) formalism, and a behavior modeling tool, the Situation Graph Tree (SGT). On the one hand, F-Limette is a *FMTL*–based inference engine based on fuzzy temporal predicates, which in the following will be also referred as *concepts*. On the other hand, SGTs describe human behavior as a combination of concepts instead of numerical data. The following sections fully explain FMTL, SGTs, and their combination towards analyzing observed behaviors in image sequences.

4.1 Inferring Concepts from the Scene

This section describes the management and processing of the numerical results obtained by means of tracking and other low–level processes. Reasoning on these results has to address three challenging issues:

1. Concepts extracted from image sequences recording time-variant scenes might be valid only during certain time intervals.

2. The estimation of numerical knowledge from image sequences involves uncertainty.

3. More uncertainty is included from associating conceptual attributes with geometric quantities, for example *abnormal behavior* speed, due to the inherent

vagueness of many concepts.

In order to address these issues, concepts can be well represented not by using quantitative data but with fuzzy temporal logic predicates. To this end, we use the Fuzzy Metric Temporal Logic (FMTL) formalism [90], which consists of a rule–based inference engine in which conventional logic formalisms are extended by a fuzzy and temporal component. In terms of notation, FMTL is similar to the well known reasoning engine PROLOG [27]. However, while the latter is based on resolution calculus and depth search, FMTL uses tableaux calculus, and provides several inference strategies (depth search, breadth search, beam search) which can be selected by the user.

In addition, each FMTL predicate has a temporal validity expressed by a time interval $(t_1, t_2) \in \mathcal{Z}$. The following expression:

$$t_1 : t_2 \; ! \; performing\_activity(\ldots). \tag{4.1}$$

indicates that the predicate $performing\_activity$ is valid for the time interval (t_1, t_2).

The *state vector* of an agent refers to the quantitative information obtained for that agent in a single frame step t:

1. The 2-D spatial position *(x,y)* of the agent *Agent* in the ground–plane, in addition to the velocity *Vel* and the orientation *Or*. These three parameters are called the *spatial status* of the agent, and constitutes the typical result ouptut by motion tracking systems.

2. A label *aLabel* which describes the action being performed by the agent, inferred with respect to the velocity *vel*. Thus, we may differentiate between *walking*, *standing* or *running*.

All this knowledge is comprised in the following attribute scheme, or *state vector of the agent*:

$$has\_status(Agent, X, Y, Or, Vel, aLabel). \tag{4.2}$$

FMTL membership functions encode the a–priori knowledge about the relation between the numerical estimation for a human feature as obtained by a vision system and the conceptual values used to describe qualitatively this estimation. As an example of FMTL membership function, the performance of the *has_speed(Agent, Concept)* function is explained next. This function associates a fuzzy speed value given the numerical value obtained from tracking. Given a state vector of an agent in a given time t, *has_status(Agent,_,_,_,Vel,_,_)*[1], the output fuzzy value *Concept* is assigned depending on the numerical value *Vel*.

```
has_speed(Agent,Concept) :-
        has_status(Agent,X,Y,Theta,Vel,A,S) ,
        associate_speed(Vel,Concept).
```

[1]In logic programming, the symbol '_' means that the corresponding value is not relevant for the predicate to be true or false.

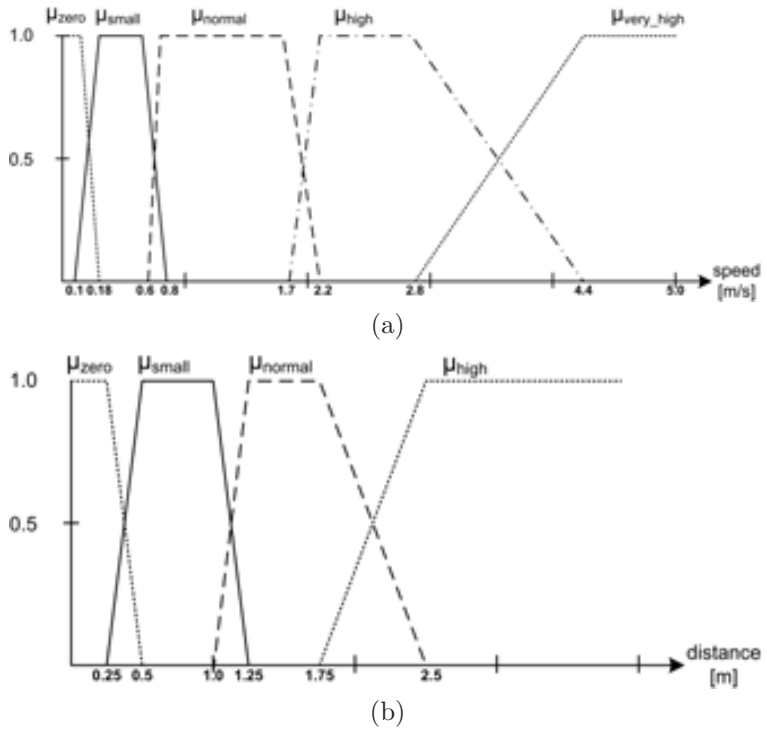


Figure 4.1: (a) Discretization of continuous speed values into a set of intervals. The graph shows the fuzzy membership functions $\mu_{speedvalue}$ for the subset (*zero, small, normal, high, very_high*) of discrete conceptual speed values. (b) Discretization of continuous distance values.

The function *degreeOfValidity(Vel, A, B, C, D)* computes the degree of confidence for the numerical value *Vel* to be in the trapezoid formed by the values *(A,B,C,D)*, see Fig. 4.1. Inside the function *associate_speed(Vel, Value)*, the fuzzy value with the highest degree of confidence is assigned to the output value *Concept*:

```
associate_speed(Vel,Concept) :-
        degreeOfValidity(Vel,-0.83,-0.27,0.10,0.15) , Concept = zero ;
        degreeOfValidity(Vel,0.10,0.15,0.6,0.83) , Concept = small ;
        degreeOfValidity(Vel,0.6,0.83,1.7,2.2) , Concept = normal ;
        degreeOfValidity(Vel,1.7,2.2,2.8,4.4) , Concept = high ;
        degreeOfValidity(Vel,2.8,4.4,6.0,12.0) , Concept = very_high ;
        degreeOfValidity(Vel,0.28,0.83,100.0,100.0) , Concept = moving.
```

The Commonsense Knowledge Base consists of a terminology of FMTL predicates related to the information to be described about the scene. Therefore, the terminology actually restricts the discourse domain, and consists of rules and facts regarding the state of the agent, its relationship with the environment, and information about the

context. Note that the problem domain itself provides the knowledge required to design such a terminology.

Based on the quantitative information of the state vector of the agent obtained from tracking procedures, the aim is to associate conceptual interpretations for such numerical data. For this purpose, we next address three different strategies to accomplish such an abstraction process, according to the source of knowledge which is exploited to generate the qualitative description[2].

4.1.1 About the spatial information of an agent within the scene

Quantitative state parameters are associated to concepts like *moving*, *small*, *left*, or *briefly* with a fuzzy degree of validity characterizing how good a concept matches the numerical quantity. As a result, the speed and orientation parameters of the state vector will be associated to fuzzy attributes, thus allowing the instantiation of logic predicates such as:

$$has\_speed(Agent, Value),$$
$$has\_direction(Agent, Value).$$

4.1.2 About the relationship of an agent with respect to its environment

Spatial relations are derived by considering the positions of the agents and other static objects in the scene. This description is implemented by applying a distance function between the positions of different agents/objects in the scene. Subsequently, a discretization of the resulting distance value is obtained by using Fuzzy Logic:

$$is\_alone(Agent, Proximity),$$
$$has\_distance(Agent, Patiens, Value).$$

Moreover, behavior analysis requires of an explicit reference to the spatial context, i.e., a conceptual model of the scene. Such a model allows to infer the relationship of the agent with respect (predefined) static objects of the scene, and to associate *facts* to specific locations within the scene. The scene is divided into polygonally bounded *segments*, which describe the possible positions in which an agent can be found. Each *segment* has a label which determines the conceptual description associated to such a segment. For example, the predicate $on\_road(Agent, WLine)$ checks whether the spatial position of the agent *Agent* in the scene is inside the specific *road* segment defined as *WLine*.

4.1.3 About the action which an agent is performing

An action label is associated using Fuzzy Logic to the state of the agent, depending on the agent velocity. Thus, we can distinguish between three different actions, namely

[2]In the predicates described next, capitalized words are considered variables and lower case ones are considered to denote constant values

running, *walking* or *standing*. These fuzzy attributes allow the posture status to incorporate a conceptual term, i.e. the label of the recognized action:

$$is\_performing(Agent, aLabel).$$
$$is\_running(Agent).$$
$$is\_waving(Agent).$$

This posture status can also reflect additional information about performance characteristics, such as the *style* or the *relative velocity* in which the action is being played, for example.

4.2 Situation Graph Trees

Situation Graph Trees [7] provide a deterministic formalism to represent the knowledge required for human behavior modeling, where *behavior* refers to human agent trajectories which acquire a meaning in a specific scene. These trajectories contain information about human agent features, see Eq. (4.2), for each time step. Based on this information, logic predicates are instantiated by employing FMTL. As a result, temporally and conceptually isolated logic statements are instantiated for each frame. Using SGTs, this knowledge is further embedded into a temporal and conceptual context.

The concept of *generically_describable_situation* presented in [71] is the basic unit of SGTs: a situation consists of an agent state, together with the potential reactions that such an agent can perform in such a state. SGTs organize the set of plausible situations into a temporal and conceptual hierarchy. Thus, on the one hand, SGTs can represent the temporal evolution of situations, and a set of potential successor situations are specified for each situation. That means, predicate evaluation is performed in a goal-oriented manner: given a situation, only its successors will be evaluated in the next time step. On the other hand, each situation can be described in a conceptually more detailed way, thus allowing to establish conceptual descriptions with a certain level of abstraction and specificity.

The basic component of SGTs is the *situation scheme*, which embeds the quantitative knowledge for a given agent at each frame step. These situation schemes are separated into two parts, see Fig. 4.2: the *state scheme* and the *reaction scheme*. On the one hand, the state scheme refers to logic predicates about the state of the agent. On the other hand, the reaction scheme describes the responses (in terms of logic predicates, too) that an agent is supposed or expected to perform when the predicates of the state scheme are satisfied. In this case, the traversal process instantiates the status part of the situation for a particular agent.

Situation schemes are connected by directed edges, called *prediction edges*, which define the temporal successor relationship within the sequence. Thus, transitions between situations express a temporal change from one situation to another: if an agent has been recognized to instantiate the situation from which a prediction edge starts, one probable next situation for that agent could be the situation to which the prediction edge points. Of course, an agent can persist in a single situation for more than one time step: those prediction edges from one situation to itself are referred as

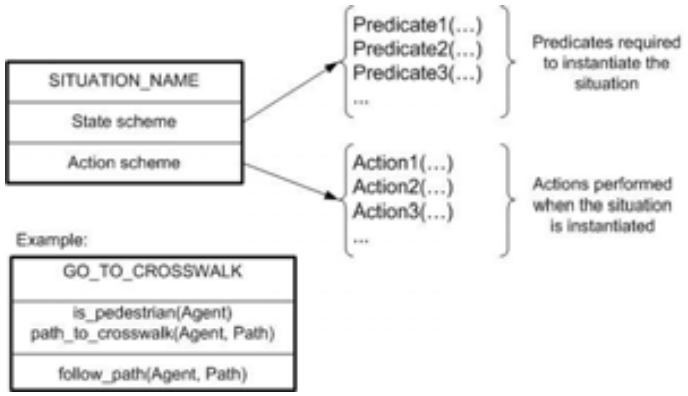


Figure 4.2: The situation scheme is the basic element of Situation Graph Trees.

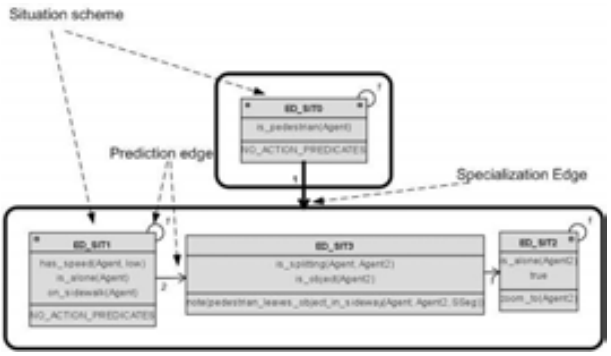


Figure 4.3: Example of Situation Graph Tree

self-prediction edges.

Situation Graphs are built by embedding single situation schemes into temporal sequences of other schemes or, in other words, by linking situation schemes using prediction edges. Thus, the resulting graph is directed and can comprise cycles. Situations within situation graphs can be marked as *start situation* and/or *end situation*. Each path from a start situation to an end situation defines a sequence of situations represented by the situation graph, see Fig. 4.3.

Also, situation graphs can particularize superordinate situation schemes by using *particularization edges*. These edges allow to describe a situation in a conceptually or temporally more detailed manner. The situation scheme particularized is called *parent situation scheme* of the particularizing situation graph. Therefore, situation schemes and their particularization edges build tree-like structures, which are called Situation Graph Trees (SGT).

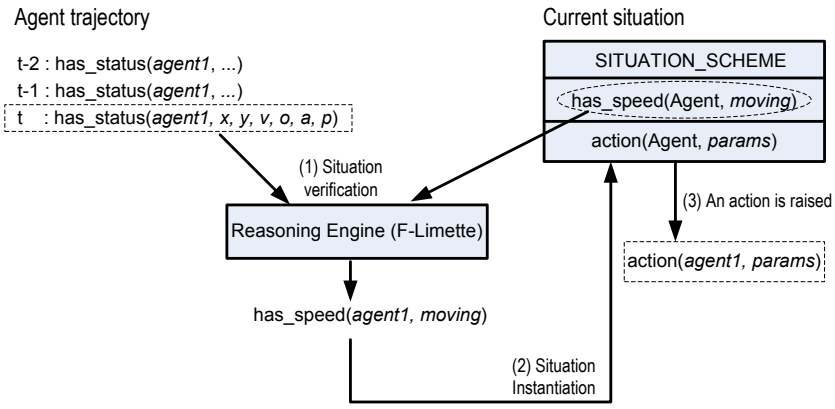


Figure 4.4: Situation instantiation in a Situation Graph Tree.

4.3 Evaluation of Human Behaviors in Image Sequences

SGTs can be used to recognize those situations which can be instantiated for an observed agent by applying the so–called *graph traversal* [49]. The goal is to determine the most particularized situation which can be instantiated by considering the FMTHL predicates that are true at each time step. This traversal of the SGT is applied by considering the knowledge encoded in the form of prediction and particularization edges. The procedure is described as follows.

The recognition of situations is started at the root situation graph. Here, a start situation is searched for which each logic predicate of the state scheme is satisfied for the actual inspected point of time. If no such situation scheme can be found, the traversal fails. Otherwise the traversal is instantiating that situation. If this situation scheme is further particularized by any situation graph, these graphs are again searched for those start situation schemes whose state predicates are also satisfied. Again, if such a situation scheme can be found, it is instantiated. Using this procedure, the most particular situation scheme will be instantiated. In other words, for each point in time, an agent is instantiating situation schemes on several levels of detail, each on a specialization *path* from a scheme in the root graph to the most particular case.

Concerning the reaction scheme, two different cases are considered when applying the graph traversal: each reaction predicate can be marked as *incremental* or *non-incremental*. Incremental reactions of a situation scheme are executed whenever this situation scheme lies on a particularization path of schemes being currently instantiated by the agent. On the other hand, non-incremental reactions are only executed whenever the situation is the most particular one on a path of schemes that an agent is instantiating.

For the next point in time, only those situation schemes connected by prediction edges are investigated. In other words, the next predicted situation scheme on the

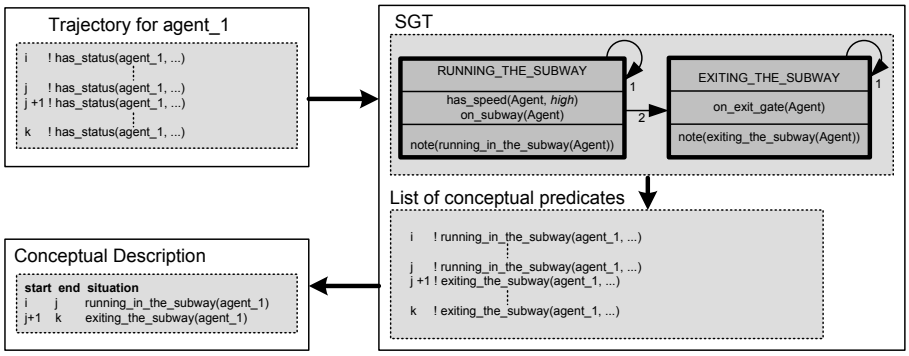


Figure 4.5: Generation of conceptual descriptions from image sequences using the framework SGT – FMTL.

same level of detail is first searched. If no scheme can be instantiated, two different cases are considered. On the one hand, if the situation scheme last instantiated is an end–situation, prediction on this level is finished and the traversal continues in the parent situation of the actual graph. On the other hand, if no end situation was reached in the actual graph, the traversal ends.

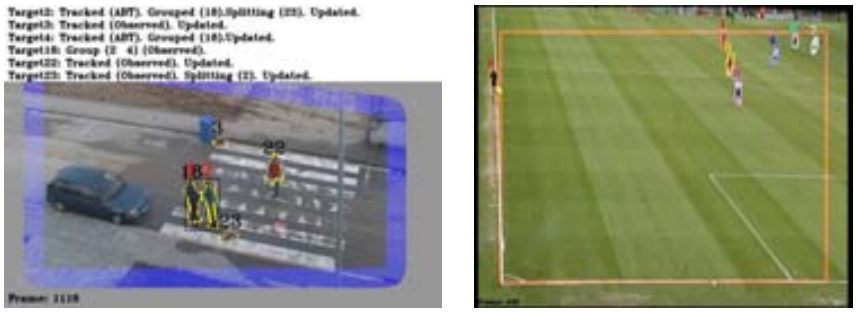
4.3.1 Description of Observed Behaviors

The conceptual description of an image sequence includes an explanation of observed behaviors, consisting on a list of conceptual predicates indexed by the frame interval where these predicates have been valid. In order to generate such indexing predicates, the SGT is designed with *note(P_c)* predicates at the reaction scheme. This predicate generates a conceptual predicate P_c when the situation is instantiated. When the same predicate is generated repeatedly during frames $i \ldots j$, then these are grouped into a single reference:

$$i : j \ ! \ P_c(agent) \tag{4.3}$$

, thereby generating an explanation of agent behavior between frames i and j.

Figure 4.5 shows an example of the overall process followed to obtain this conceptual predicates from the geometrical data obtained from tracking. Given an agent *agent_1*, a list of predicates *has_status(...)* is generated by the tracking system. In this case example, the predicate list represents the frame interval $(i, \ldots, k)$. When frame i is analyzed, the SGT is inspecting the start situation scheme *RUNNING THE SUBWAY*. The state scheme of that situation is satisfied from i-th to the j–th frame and, thus, the reaction scheme generates the predicate *note(running_in_the_subway(...))* for each frame in this interval. When the $(j+1)$–th state vector causes this state scheme to fail, the next predicted situation is inspected (*EXITING_THE_SUBWAY*). This state scheme is satisfied during the interval $(j+1, \ldots, k)$, and then the predicate *note(exiting_the_subway(...))* is generated during this interval. Unique predicates are grouped into a semantic description.



(a) (b)

Figure 4.6: Tracking results (a) Pedestrian crossing sequence (b) Football sequence.

4.4 Experiments

In this section we describe the full top–down system applied to two different discourse domains. The aim is to demonstrate the feasibility of top–down frameworks to exploit the semantics of each discourse domain. The performence of the system in the video–surveillance domain is tested using image sequences from the HERMES Outdoor scenario[3]. In the first experiment, the objective is the study of pedestrian behaviors and the interaction between pedestrians and vehicles. Next, the second experiment focuses on the analysis of sport video sequences to obtain descriptions of sport matches . To this end, a football sequence from the VS-PETS 2003 database [77] has been used to evaluate football player behavior. Some samples of tracking results from both sequences can be seen in Fig. 4.6.

For each application domain a conceptual model of the environment has been designed, see Fig. 4.7. The scenario is represented in ground–plane coordinates, which allows the reasoning system to receive 3D information from different calibrated cameras. Moreover, this allows the system to work with centimeters instead of pixels.

The design of the experiments takes into account the main differences between both scenarios:

1. In the HERMES Outdoor scenario, there is no a–priori information about the number of agents that will appear. Also, there are at least two kinds of agents that will interact, i.e. pedestrians and vehicles. On the contrary, the Football scenario leads to make some assumptions about the number and type of agents. In a normal configuration, a football match contains 22 playes and a referee, who are also known beforehand. We must consider the ball as an active agent as well, since its movement determines also the behavior of the rest of agents.

2. Regarding agent interactions, there are no a–priori assumptions about the relationships between agents in the Outdoor Scenario. These relationships could go from the simplest, no relation at all, to more complex relations like friendship, work, enemies, etc. On the other hand, relationships can be well established

[3]Information about the HERMES project can be found at http://www.hermes-project.eu

(a)

(b)

Figure 4.7: Conceptual scene models. (a) Pedestrian crossing scenario divided into semantic regions (b) Football scenario. The semantic labels specify the defense fields for the teams A and B. Note that each semantic region can be referred as attacking regions for the opposite team. For instance, the left–defense region of team A is the right–attack region of team B.

in the Football scenario. The initial 23 human agents are divided into two teams plus the referee, and assumptions can be made between inter–team and intra–team interactions.

3. Finally, the set of expected behaviors are also different between these two scenarios. While the HERMES Outdoor scenario allows to develop a wide set of possible behaviors, being some of them are specific of this particular domain (e.g. crossing the road), and being the rest a raw set of human behaviors that can take place unexpectedly (e.g. meeting another pedestrian), the Football scenario entails a limited set of this specific sport domain, whose rules are well known and can be easily reproduced and recognized,.

This list of scenario features allows to bound the degree of complexity of each proposed scenario. Even in a very restricted domain, like the HERMES Outdoor scenario, the set of possible behaviors is wide and confusing. Therefore, we are forced

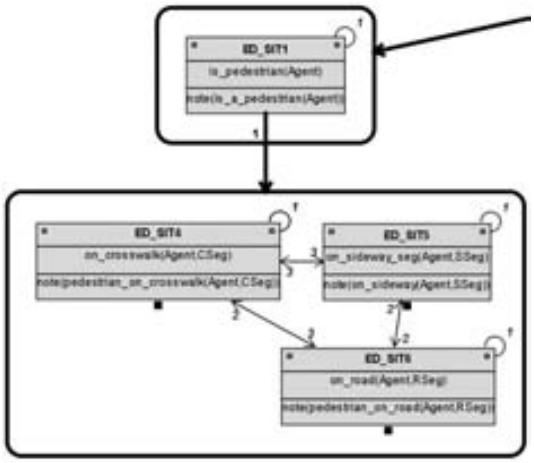
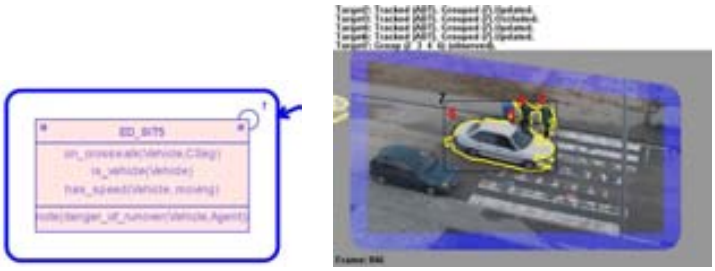


Figure 4.8: SGT describing general pedestrian behavior within the HERMES Outdoor scenario.



(a) (b)

Figure 4.9: Describing interaction between vehicles and human beings. (a) Runover situation. This situation is a specialization of the ED_SIT2 situation of Fig. 4.8. (b) Two human agents are crossing the crosswalk while a car is approaching. The system triggers a *danger–of–runover* alarm.

to specify an a–priori subset of behaviors that will be recognized.

4.4.1 Experiment 1: Understanding the Semantics of Agent Interaction

Experiments in the HERMES Outdoor scenario have coped with complex human behaviors and interactions between pedestrian and vehicle agents. The semantic analysis includes interactions between pedestrian and (i) other moving objects, such as vehicles; (ii) static objects, such as picking up a left bag; and (iii) other pedestrians, e.g chasing.

A general SGT is designed to model pedestrian location at each frame step, see

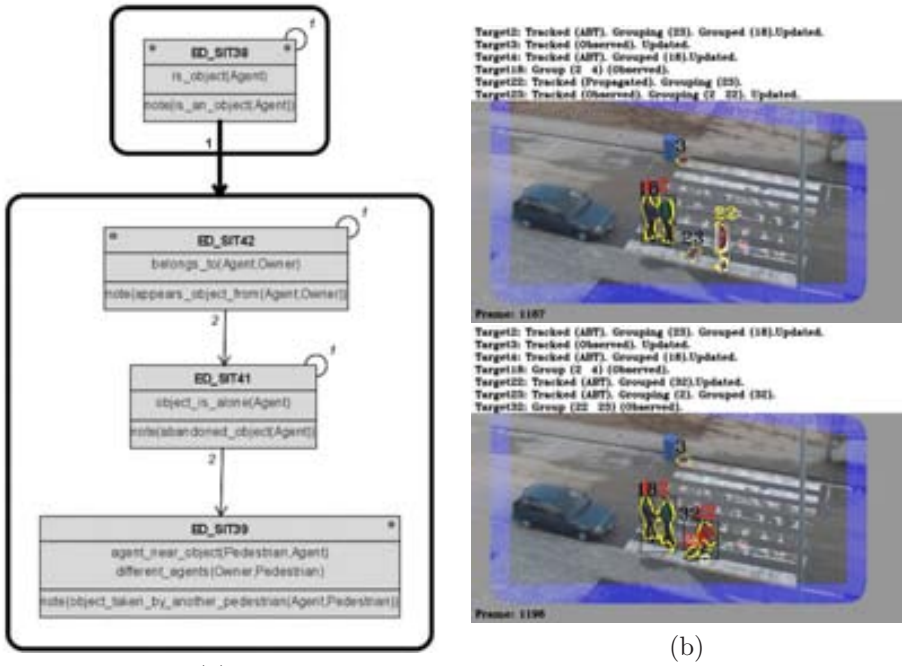


Figure 4.10: Describing interaction between human agents and objects. Further details in text.

Fig. 4.8. Given the conceptual scene model, the agent location is inferred using the spatial coordinates obtained from tracking. Hence, a traversal of this SGT produces a conceptual description of agent motion within the scenario. This SGT constitutes the starting point towards the recognition of more specific situations, as explained next.

First, the interaction between vehicles and pedestrians is evaluated by considering a dangerous event, the possibility of runover. This event is defined as the concurrent existence of a vehicle and a pedestrian moving within a pedestrian crossing. Such event is modeled by specializing the situation ED_SIT4 of the general SGT in Fig. 4.8. The new situation, needs to satisfy three FMTL predicates, namely *on_crosswalk(Vehicle,CSeg)*, *is_vehicle(Vehicle)*, and *has_speed(Vehicle, moving)*, which mean that $Vehicle$ is a vehicle agent that is moving along the crosswalk, see Fig. 4.9(a). Since the new situation is a specialization of ED_SIT4, it also carries the condition from its parent, which is that *Agent* is a pedestrian also moving along the crosswalk. While this situation remains true, the system triggers the predicate *danger_of_runover*. An example frame of this situation is shown in Fig. 4.9.(b).

Secondly, the interaction between pedestrians and static objects is inspected. To this end, a new SGT has been modeled in order to describe the behavior of a static object in the scenario. For instance, static objects can *carried* by other agent in

the scene. However, we would like to have a description of the scenario centered on objects, in order to study their relationships with other agents along time.

Fig. 4.10.(a) shows a SGT modeling the interaction between pedestrians and objects from the scenario. In the vision subsystem, objects are not detected until they separate from their owner, thus becoming a new moving target in the tracking system. When it occurs, the low–level predicate *is_splitting(A,B,C)* indicates that the agent A has been divided into agents B and C. This predicate allows to instantiate a higher–level predicate, *belongs_to(Agent, Owner)* which will be permanently satisfied from the time step when the low–level predicate was instantiated:

```
always(belongs_to(Object, Owner):-
   event past (is_splitting(Owner, Object, Owner);
               is_splitting(Owner, Owner, Object)),
   is_pedestrian(Owner)).
```

The prefix *event past* indicates that the subsequent predicate was satisfied at some time step in the past. Therefore, if *Object* and *Owner* were created by the separation of another agent at time t, this predicate will remain true from $t + 1$ on.

Situation ED_SIT41 describes the situation where the object has been abandoned in the scenario. This is inferred when the owner of the object maintains a long distance to the object and the predicate *object_is_alone* is instantiated:

```
always(object_is_alone(Object):-
    belongs_to(Object, Owner)
    not(agent_near_object(Agent, Object)).
```

and the predicate *agent_near_object* is defined in terms of the fuzzy membership function *has_distance* described in Section 4.1:

```
always(agent_near_object(Agent, Object):-
    have_distance(Agent, Object, small),
    is_pedestrian(Agent),
    is_object(Object)).
```

Third situation scheme, ED_SIT39, models the situation where another pedestrian, different from the owner, takes the object. The predicate *agent_takes_object* is defined in terms of the low–level predicate *is_grouping*, instantiated when two targets are grouped into a single one in the tracking level.

```
always(agent_takes_object(Agent, Object):-
    is_pedestrian(Agent),
    is_object(Object),
    have_distance(Agent, Object, zero),
    (is_grouping2(Group, Agent, Object);
     is_grouping2(Group, Object, Agent))).
```

Note that we impose that the ground–plane distance between the object and the pedestrian must be zero, in order to avoid as much as possible the instantiation of this predicate due to an eventual occlusion in the image plane.

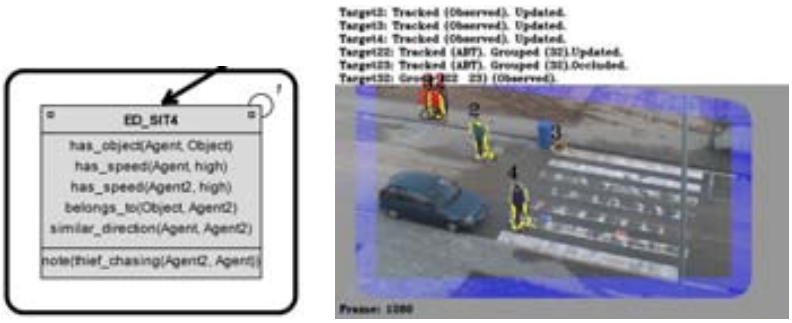


Figure 4.11: Describing interaction between human agents: The first agent has taken an object belonging to another agent. The system outputs the note *thief chasing*.

In the case of Fig 4.10.(b) object 23 appeared in the scenario and previous reasoning has stated that the object belongs to pedestrian 2. At frame 1167 the distance between object 23 and agent 2 allows the instantiation of situation ED_SIT41. Later, in frame 1196, agent 22 takes the object, instantiating situation ED_SIT39. This makes the system output the conceptual description *object_taken_by_another_pedestrian*. Nevertheless, no further reactions can be taken at this precise time step because there are possible interpretations for this situation, i.e., relationship between agent 2 and agent 23 is not known.

Lastly, Fig. 4.11 shows a specialization of the situation ED_39 in Fig. 4.10, to describe a case of interaction between pedestrians. Such scheme describes the owner of the object chasing the pedestrian that has taken his object. If both pedestrians are running in similar direction, deduced by the predicate *has_direction* described in Section 4.1, and the owner is following the thief, the system outputs the conceptual description *thief_chasing*.

The complete behavior analysis of the HERMES Outdoor scenario is depicted in Fig. 4.12. There the conceptual descriptions are indexed by the frame step of instantiation. Blue descriptions refer to the entrance of new moving agents inside the scenario and their classification between vehicle or pedestrian. Red descriptions refer to the complex behaviors described so far.

4.4.2 Experiment 2: Semantic Video Annotation of Sport Image Sequences

Our second experiment focuses on the recognition of high–level events in the domain of a football match. The SGT depicted in Fig. 4.13 represents a simplified version of the model used to represent football player behavior. This SGT copes with the most general interaction between one player and the ball, such as driving the ball (ED_SIT1), shooting/passing the ball (ED_SIT3) and the possible end situations given that the ball is alone within the field: (i) ball in goal (ED_SIT5) (ii) ball out of field (ED_SIT4) (iii) ball taken by a player of the same team (ED_SIT8), and (iv) ball taken

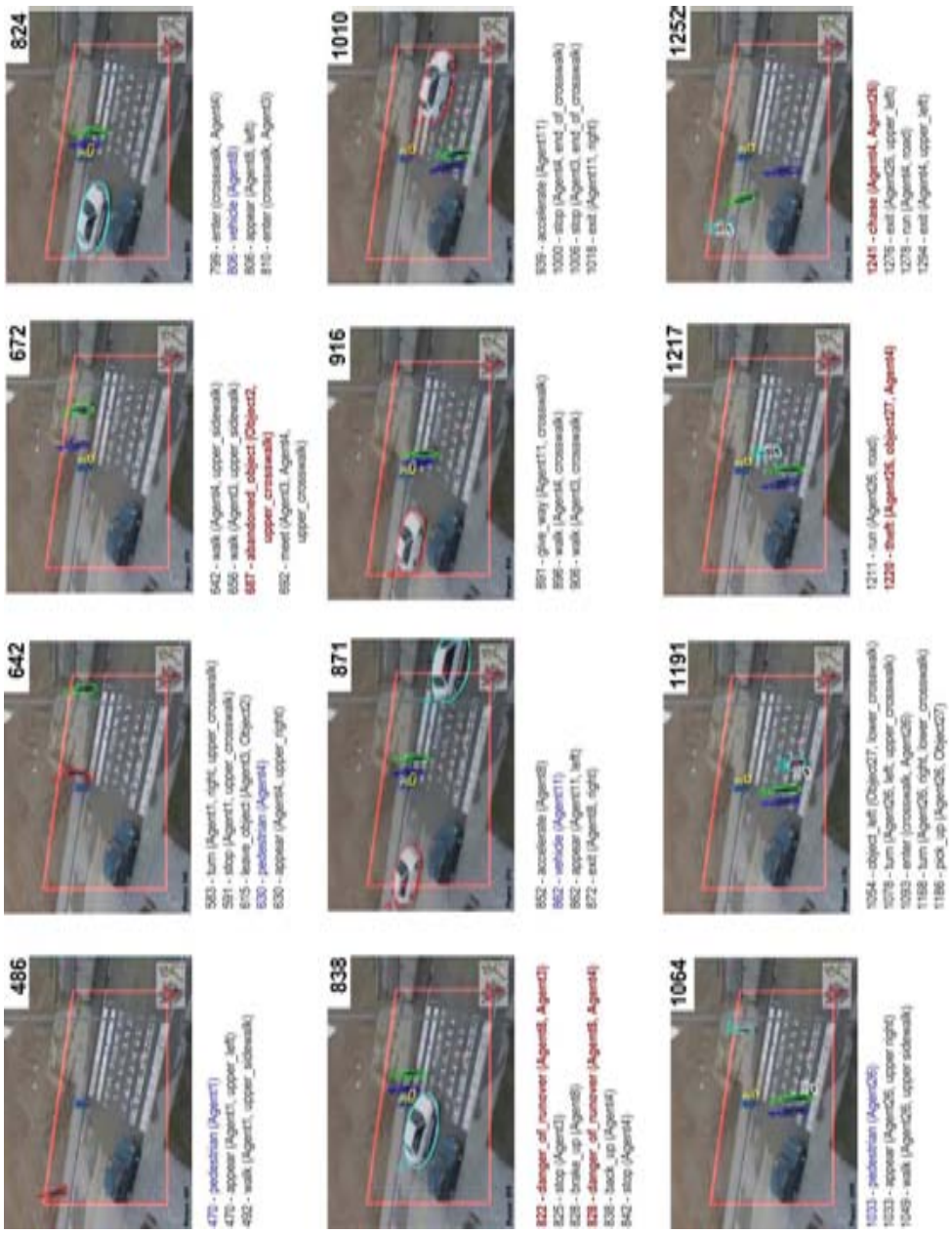


Figure 4.12: Complete sequence of conceptual descriptions in the Outdoor Scenario.

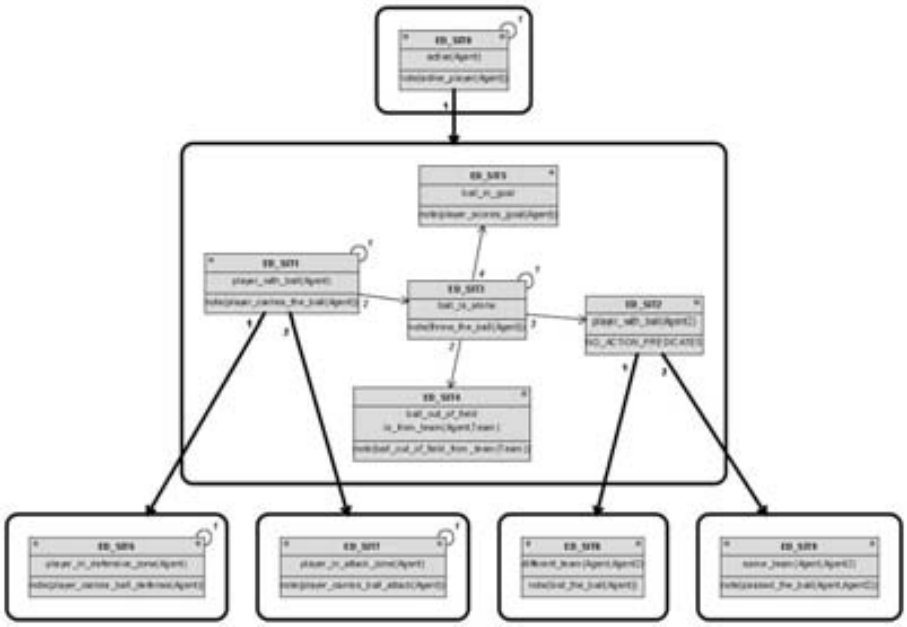
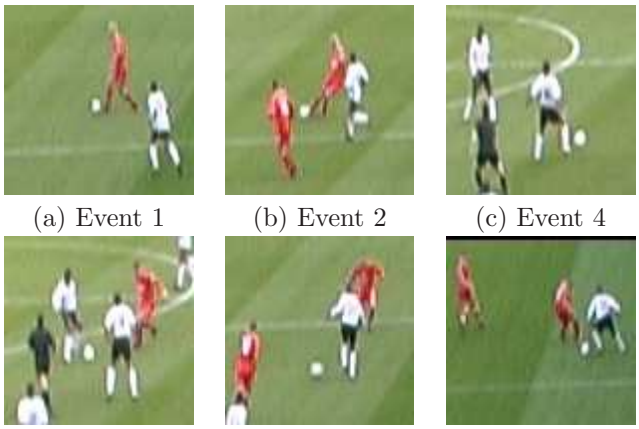


Figure 4.13: SGT for football player description. Further details in the text.

by a player of the opposite team (ED_SIT9). Note that the predicates concerning the ball, e.g. *ball_in_goal*, or *ball_is_alone* do not carry parameters, since the ball is unique inside a football field, and therefore there is no need to refer the ball with a name.

This SGT is assigned to each player, so that the traversal describes the interaction between one player and the ball. The traversal finishes when the situation ED_SIT2 is instantiated, which means that another player has got the ball. In that precise moment, the SGT associated to that player begins the traversal, and so on. Table 4.1 shows the results obtained processing the sequences VS–PETS 2003. Since these sequences are recorded not for human behavior analysis but for tracking evaluation, not too complex behavior patterns are observed. However, a fair conceptual description can be obtained for the sequence *Testing_camera_3* in the frame interval $[187, 465]$. This table is created by joining the results obtained by traversing the SGT of Fig. 4.13 for each agent involved in that sequence. A complete description of the game performance is obtained: The ball is initially carried by the team B but a failed pass is caught by the team A. Next, the ball is passed through different players and the last player reaches the attacking field.

Events $1, 2, 3$ were obtained in the traversal for agent *player_teamB_4* and the events $4, 5$ come from the traversal for agent *player_teamA_2*. Next, events $6, 7, 8$ are obtained for the agent *player_teamA_5*. Finally, events 9,10,11 are obtained for the agent *player_teamA_4*. Note that events $1, 4, 6, 9, 10$ denote the interaction of the player with the environment (attack or defense). Situations ED_SIT6 and ED_SIT7 could be further specialized into situation graphs that completely describe the agent



(a) Event 1 (b) Event 2 (c) Event 4

(d) Event 5 (e) Event 8 (f) Event 10

Figure 4.14: Sequence of the behavior explained in Table 4.1. The events 1, 2, 4, 5, 8, 10 correspond to Fig. 4.14.(a),(b),(c),(d),(e),(f), respectively.

position over the football field, like in the SGT of Fig. 4.8. Fig. 4.14 shows the key–frames of the sequence, where the events above mentioned can be observed; namely the events 1, 2, 4, 5, 8, 10 correspond to Fig. 4.14.(a),(b),(c),(d),(e),(f), respectively.

Start	Situation	Event Num.
187	player_carries_ball_defense(player_teamB_4)	1
279	threw_the_ball(player_teamB_4)	2
296	lost_the_ball(player_teamB_4)	3
296	player_carries_ball_defense(player_teamA_2)	4
305	passed_the_ball(player_teamA_2, player_teamA_5)	5
305	player_carries_ball_defense(player_teamA_5)	6
332	threw_the_ball(player_teamA_5)	7
357	passed_the_ball(player_teamA_5, player_teamA_4)	8
357	player_carries_ball_defense(player_teamA_4)	9
403	player_carries_ball_attack(player_teamA_4)	10
465	threw_the_ball(player_teamA_4)	11

Table 4.1

Sequence of conceptual descriptions obtained for the frame interval $[187, 465]$ in the VS–PETS image sequence. More details in the text.

4.5 Discussion

This chapter has presented a complete framework for the analysis of behavior in image sequences, specially focused in human beings. The challenge represented by the complexity and unpredictability of human behavior has been addressed by using a conceptual approach which discretizes the quantitative data obtained by vision systems into concepts, later combined to describe behavior patterns.

The proposed framework cope with the Conceptual Integration and Behavior Interpretation Levels of the Human Sequence Evaluation scheme, introduced in Section 1.4. The input data supplied to the framework has passed all the lower levels of HSE, and each of them may introduce errors or noise to the quantitative data. Therefore, the accuracy of the proposed approach is not dependant only on itself, but also on the accuracy of the lower levels, i.e. image acquisition, segmentation and tracking. These lower levels entail problems that are not completely solved for every type of scenario. For instance, sudden illumination changes still cause problems in the segmentation, while long occlusions can lead to lose the tracks of a target in the tracking process.

An important advantage of this framework is the feasibility for the incorporation of additional knowledge from a netwoork of cooperative cameras and alternative vision processes, like those described in Section 1.5.3. Thus, predicates relative to face expressions or more detailed body actions would extend the level of specificity of the described behaviors, but without the need to change the behavior models obtained so far.

On the other hand, the framework has to minimize the need of scene–dependent knowledge. This means that new discourse domains will require to specify new knowledge, i.e. the design of new FMTL predicates and new SGTs must be performed by a human expert. Even, a different scenario from the same domain will require some modifications, e.g. the conceptual scene model. Nevertheless, SGTs base their definition on semantic predicates, thereby letting to an easy understanding of the results by human readers. In fact, this contrasts with the results by pure statistical methods, which, despite of representing automatically learnt behavior models, are difficult to interpret since are not based on a semantic interpretation of the scenario.

4.6 Resum

L'aprenentatge automàtic a partir d'observacions té com a principal avantatge que els models de comportament obtinguts representen fidelment les dades usades per a l'aprenentatge, i per tant estan exempts de la desviació subjectiva produida pel disseny d'un expert humà. En canvi, la informació obtinguda és purament quantitativa i no permet extreure una explicació semàntica del resultat. Els models obtinguts representen el comportament normal dins d'una escena determinada, i tot el que sigui diferent és considerat una anomalia.

En aquest capítol proposem un model de comportament determinista que permet definir acuradament els comportaments esperats en un escenari, fent us de coneixement semàntic definit prèviament. El model està format per dos eines encarregades de convertir la informació quantitativa generada per un sistema de visió, en informació qualitativa, o conceptes, entenibles per un humà i que permeten explicar

semànticament les observacions. En primer lloc, fem servir el motor de raonament Fuzzy Metric Temporal Logic (FMTL) que instància un conjunt de predicats d'alt nivell a partir de les observacions. En segon lloc, construim Arbres de Grafs de Situacions (AGS) que permeten organitzar aquests predicats per representar comportaments.

Per tal de demostrar l'eficàcia del model proposat, hem construit models en dos camps d'aplicació diferents. En primer lloc, hem generat models de comportament per a entorns de video–vigilància oberts. Mostrem descripcions conceptuals d'alt nivell per a comportaments observats en un entorn urbà, com per exemple *robatori* o *risc d'atropellament*. En segon lloc, utilitzem nous models de comportament dins l'entorn d'esdeveniments esportius, generant permetent la generació automàtica de comentaris sobre les jugades observades.

Chapter 5

Augmenting Reality with Virtual Behaviors

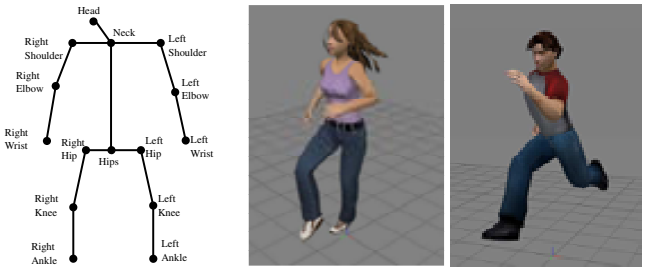
A desirable feature of every modeling formalism is that model instances can not only be represented and recognized, but also synthetically generated. This chapter demonstrates the feasibility of the previous framework to simulate behavioral synthethic agents in a virtual environment.

Next sections present an efficient and general approach to combine virtual agents with real agents in a real world image sequence, using the FMTL + SGT framework. Real agents are detected using a motion–based multi–object tracking algorithm and joined with synthetically generated virtual agents in real–time.

5.1 Virtual Agent Modeling

This section adapts the FMTL + SGT framework described in last chapter towards the creation of synthetic instances of agent trajectories. As explained before, the SGT is traversed given the quantitative information obtained from tracking at each frame step, which can instantiate situation schemes and can raise reactions. In previous chapter these were basically semantic annotations of observed behavior. Next, these reactions will be used to generate synthetic behaviors. Given an initial configuration for a virtual agent, the system will recursively generate the activities of an agent within its environment.

The generation of synthetic trajectories requires to adapt the prior knowledge initially designed for behavior recognition. Thus, each virtual agent behavior is modeled using three different generation processes, enclosed in the SDL, BIL, and CIL levels of the HSE, see Section 1.4. Firstly, the physical *motion* of the virtual agent is defined in short term, e.g. human actions like *walk, bend*. Secondly, the virtual agent *activity* represents the intermediate goals that each agent should accomplish in order to satisfy a particular objective, such as *going to a particular position*. Finally, the virtual agent *behavior* describes the capabilities of the agent to reach the goals specified by its activity, thereby interacting with other agents, static objects and different locations in the scene. Examples comprise *crossing a crosswalk* or *catching a bus*.



(a) (b)

Figure 5.1: (a) Generic human body model represented using a stick figure similar to [26], here composed of twelve limbs and fifteen joints. (b) Different human models used performing *dancing* and *running* actions.

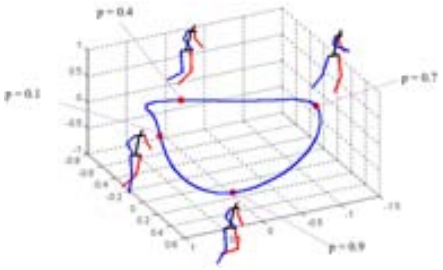


Figure 5.2: *p–actions* computed in the *aRun aSpace*, see [46] for details: by varying the parameter pose p, we actually move along the manifold, thus obtaining the temporal evolution of the human body posture during the prototypical performance of any learnt action.

5.1.1 SGT Traversal for Motion Generation

The movement of the agent can be considered as the low–level behavior: it defines the physical actions that can be performed by a virtual agent. An agent exists inside a virtual environment during a period of time steps. The information of agent A at t is defined by the *state vector* of the agent s_t^A:

$$s_t^A = (x, y, o, v, a, p) \tag{5.1}$$

This state vector embeds the 2-D spatial position (x, y) in the ground plane, in addition to velocity v and orientation o. The parameter a refers to the human action, e.g. *walking* or *running*. Finally, the parameter $p \in [0, 1]$ refers to the human body posture given a, as explained next.

A human action is defined as a discrete sequence of movements of body parts. In this work we use a human model based on the stick figure, see Fig. 5.1. The learnt sequence of movements for a particular action is called the prototipical action or *p–action* and is represented by a parameter $p \in [0, 1]$ where $p = 0$ and $p = 1$ indicate the start and end of an action, respectively. Using p–actions, we can model both cyclic actions, e.g. *walk* or *run*, and non–cyclic ones, e.g. *wave* or *bend* [47].

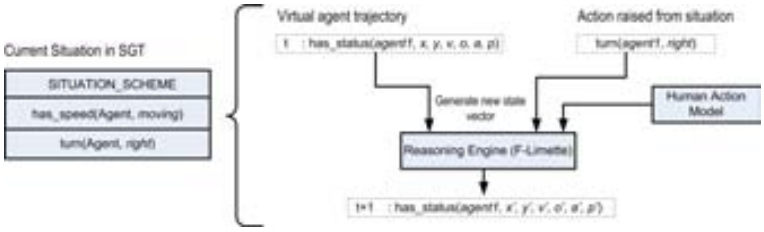


Figure 5.3: Scheme of the process since the reaction predicate is raised from the traversal until the new state vector is generated.

The *trajectory* of a virtual agent during its existence within the virtual environment is defined as the sequence of state vectors over time:

$$T^A = \{s_1^A, \ldots, s_n^A\} \tag{5.2}$$

A proper representation for behaviors allows to synthetically generate the trajectory for a virtual agent by taking into account the previously mentioned interactions. This is accomplished by enhancing the use of the SGT formalism presented in previous chapter. The reaction scheme, used to generate conceptual descriptions for observed behaviors, will now be used to modify the agent status for future frame steps. These generated frames are fed back to the SGT as a feed–back, and the traversal considers them in next traversal loops.

This recursive procedure is depicted in Fig. 5.3. The traversal starts with an initial status of a virtual agent, containing its position, orientation, speed, and action at the very first time. Then, for each time step t, the traversal takes the current agent status S_t^A to generate the next one S_{t+1}^A. In the example used in Fig. 5.3, the situation instantiated at time t generates the action predicate *turn(agent1, right)*. This predicate will modify the agent status so that the agent will be turning to the right for the next frame steps. The computation of S_{t+1}^A is based on S_t^A and the p–action modeled previously. The semantic concept *right* is converted into a numerical value by combining the current orientation o and speed v, and is used to generate the new position (x', y'), speed v', and orientation o' for the next time step. The obtained values are used to construct the agent status for the time step $t + 1$ and will be used as input for time step $t + 2$ in the next traversal loop.

Action predicates, like *turn* determine particular movements and actions for a virtual agent. This is achieved by modifying its position, velocity, orientation, and action, for example:

- `accelerate(Agent, Value)` modifies the velocity of the agent for the next time step. The fuzzy concept *Value* describes the discretized value of speed that the agent will reach in some future time step.

- `change_to_performing(Agent, Action)`: it makes the agent change from its current physical action to *Action*, by varying the pose parameter p, as described in the Movement Level.

(a) (b)

Figure 5.4: Simple behavior to describe the SGT traversal in behavior simulation. (a) SGT model of the *walking in spiral* behavior. (b) Several virtual agents walking in spiral while performing different actions.

- `follow(Agent, Agent2)` makes agent *Agent* move towards the location of agent *Agent2*. A path between *Agent* and *Agent2* is computed and the next status contains the first step towards following this path.

5.1.2 Activity Generation

In this process we define the conceptual knowledge required to specify the goals that a virtual agent is intended to accomplish. Such knowledge is classified into two parts:

1. Facts that are true, even if only for a time interval, e.g. being at a particular position or performing some movement.

2. Objectives to be accomplished in middle term, e.g. going to a position, changing to another action.

So, objectives are defined using predicates that require an adaptation of the agent trajectory to be valid in future time steps. For instance, given an agent with state vector $s_t^A = (x_t, y_t, v_t, o_t, a_t, p_t)$, the predicate *go_to_location(A, Location)* computes the shortest trajectory $\{s_{t+1}^A, \ldots, s_n^A\}$ to go to *Location* and infers the next position (x_{t+1}, y_{t+1}) of such a trajectory, according to the current speed value v_t.

The activity generation is explained using the SGT depicted in Fig. 5.4.(a), which models the example *spiral* behavior. Placing a virtual agent at some starting position, the agent will keep walking creating a spiral trajectory, as shown in Fig. 5.4.(b). The SGT traversal starts by instantiating the most general situation (*AGENT_ACTIVE*) in the first frame step. Furthermore, the specialized situation *LOOPING*, will be also instantiated since its two state predicates, i.e. *has_speed* and *inside_scene*, are satisfied. While those conditions hold, the reaction scheme of that situation will be executed every frame. Hence, the action predicates *turn* and *accelerate* will modify the orientation and speed of the agent, thus generating state vector for next time steps. When the position of the agent falls out of the scenario boundary at some time step t, the traversal instantiates the situation *OUT_OF_BOUNDS* and the traversal finishes

since no more prediction edges are available to other situations. Given the result of the traversal, a synthetic image sequence I^v is rendered by a 3D rendering engine, containing the silhouette of the virtual agent over the virtual environment. The *spiral* behavior performs isolated actions, which can be solved withouth considering the interaction with the environment.

5.1.3 Behavior Generation

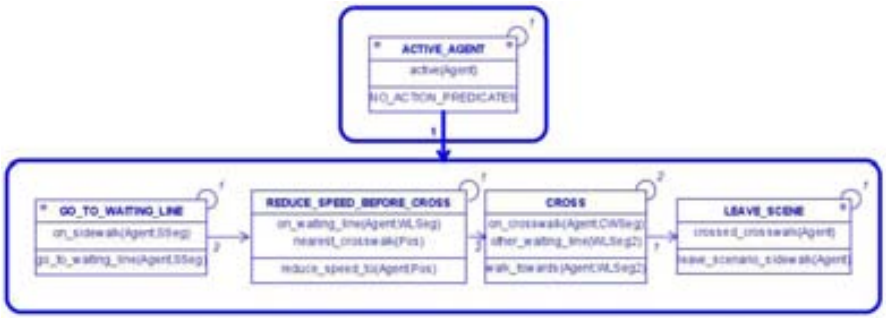
In order to define the consistent co–existence of agents inside a virtual environment, the previously defined level does not cover our requirements, because:

- Complex behaviors, defined as a sequence of activities, need to be specified in a higher level of abstraction, such as *catching_the_bus* or *crossing_the_crosswalk*.
- The existence of other agents in the virtual environment can affect the agent activity due to interactions and therefore some higher level mechanism must be provided to manage changes of and transitions between goals.

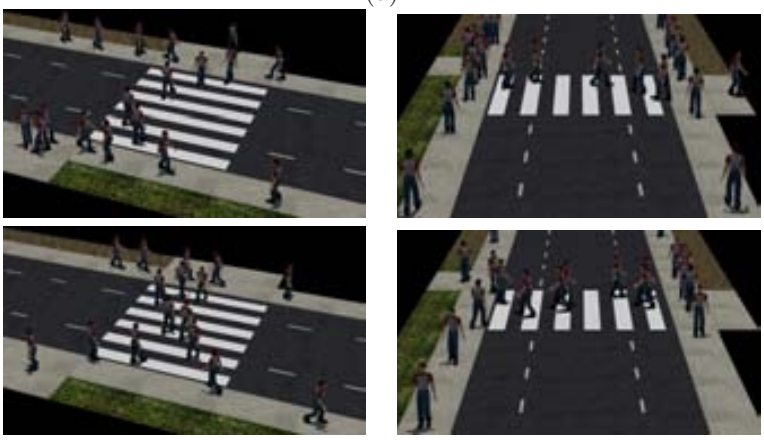
We consider two types of interactions: (i) interactions between agents, in which every virtual agent knows about the existence of other agents and adapts its behavior according to them; and (ii) interactions with the environment, in which a virtual agent acts depending not only on its spatial coordinates but also on the semantics of that location. In order to assign semantic properties to different regions of the virtual environment, we use the conceptual scene model introduced in Chapter 4, Section 4.1.2, thereby dividing the ground–plane into semantic regions.

Regarding path planning, action predicates already take into account the semantics of the agent's position and the goal the agent is going to achieve, planned by the behavior. The family of predicates *go_to_location* compute the minimum path, in terms of a list of segments, to go from the current agent position to a particular *location* segment. If no further restrictions are imposed, the path computation directly takes neighbour segments that most approximate to the goal location. However, some restrictions in terms of the semantic regions crossed can be explicitly specified. For instance, if the path towards a segment *seg_1* should contain only segments of type *type_1*, only those neighbor segments whose type is *type_1* would be considered to create the path.

The behavior generation is fully tested on the following SGT, depicted in Fig. 5.5, describing the behavior *cross_street*. Such behavior implies the accomplishment of several goals, related to specific semantic regions of the scenario. A virtual agent is initially located at a sideway region, specified by the situation *GO_TO_WAITING_LINE*. While this situation is instantiated, the agent performs a path towards the waiting line, i.e. a narrow region between the sidewalk and the crosswalk. Then, the behavior has been designed in a way that the agent stops before crossing the road, to allow possible vehicles to notice its action. Thus, the situation *REDUCE_SPEED_BEFORE_CROSSING* forces the agent to reduce its speed to eventually stop just after entering the crosswalk. Subsequently, the *CROSS* situation is instantiated and the path towards the contrary waiting line is performed by the agent by means of the predicate *walk_towards(...)*. Finally, the agent arrives to the contrary waiting



(a)



(b)

Figure 5.5: (a) SGT describing the *cross_street* behavior, used to test the agent interaction with the environment. (b) Simulatin of a crowd of virtual agents performing the *cross_street* behavior.

line and leaves the scenario walking along the sidewalk. Fig. 5.5.(b) shows example frames obtained by simulating a crowd of simultaneous agents performing the *cross_street* behavior.

5.2 Augmented Sequences with Virtual Agents

We have demonstrated so far the feasibility of the FMTL–SGT framework in the field of behavior modeling. On the one hand, SGTs have been used to recognize and interpret behavior patterns observed in image sequences. On the other hand, SGTs have led to generate synthetic image sequences involving virtual agents. Subsequently, the next challenging step is to combine both capabilities towards the generation of augmented image sequences. The objective is to include virtual agents into real image sequences in a consistent manner, so that virtual agents interact not only with other virtual agents, but also with real agents that are performing a simultaneous activity in the scenario. The process is done on–line in two steps: (i) real agent informa-

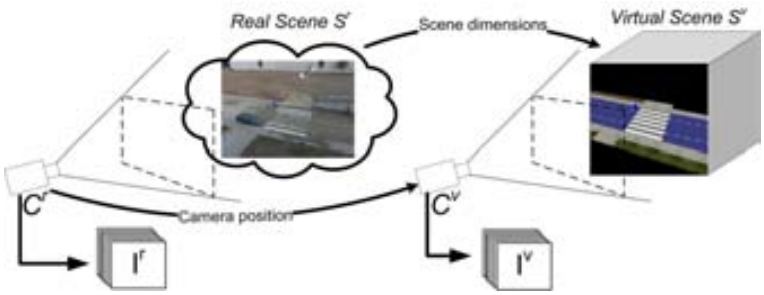


Figure 5.6: Overall scheme of the virtual representation of a real scenario. The virtual scenario is modeled using the measurements from the camera calibration.

tion is acquired at each frame step, so that can be taken by the behavior models governing the virtual agents, and (ii) virtual agents are rendered inside the image sequence taking into account the moving real agents, in order to create consistent spatial occlusions. The proposed approach has applications in several domains, including cinema, computer games, human computer interaction. In last section, we propose two applications in the fields of tracking evaluation and virtual storytelling.

5.2.1 Converting real agents into virtual agents

The straightforward way to integrate real agents into a virtual environment is translating their information in a way that there are no differences between virtual and real agents. Thus, virtual agents will interact with other agents independently from whether they are real or virtual. This process implies the acquisition of real agent data from image sequences and its conversion into sequences of state vectors. Such information contains the values listed in Eq. (5.1), required to reason about real agent behavior, and the estimated silhouette, compulsory for the consistent creation of augmented sequences. Real agent information is acquired by means of a real–time tracking algorithm based on efficient background substraction [4, 88].

Subsequently, the virtual environment must be adapted in order to match with the real scenario. Let S^r be a real scenario and let C^r be a calibrated static camera[1]. In order to have a consistent correspondence between real and virtual perspective view, a virtual scenario S^v is modeled by using the real scene dimensions obtained with the calibration of C^r, see Fig. 5.6. Finally, a virtual camera C^v is located in S^v at the same location of C^r in S^r.

The overall procedure to combine real and virtual agents is depicted in Fig. 5.7 and is explained next. An image sequence I^r is recorded using C^r and processed by the tracker. At time step t, the tracker generates a list of k targets' position coordinates over the image plane and their estimated silhouette, which are stored in I^{ra}. We obtain the ground plane representation of the k agents' positions O_t^r by applying a 2D homography using the DLT algorithm [50], thus obtaining:

$$O_t^r = \{\{x_t^1, y_t^1\}, \ldots, \{x_t^k, y_t^k\}\} \tag{5.3}$$

[1]The superindexes r and v denote *real* and *virtual*, respectively.

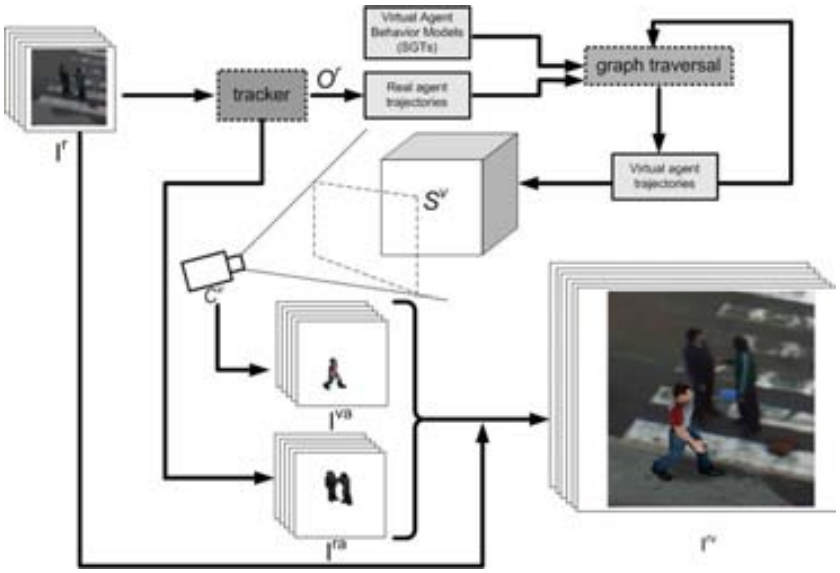


Figure 5.7: Generation of an augmented reality image sequence I^{rv}. More details in the text.

which embeds the 2D ground–plane position for the k targets i at time step t.

Next, position coordinates are converted into state vectors, i.e. *has_status* predicates, and incorporated to the list of state vectors for the current time step. Thus, the information obtained by the tracker has the same structure as that generated by the traversal of a SGT and will be considered as a virtual agent by the actual ones.

$$t \; ! \; has\_status(agent_i, x_t^i, y_t^i, o_t^i, v_t^i, a_t^i, p_t^i) \tag{5.4}$$

where the values o_t^i, v_t^i are computed using the previous agent location (x_{t-1}^i, y_{t-1}^i). The action a_t^i is valued either *standing*, *walking*, or *running*, depending on v_t^i, and the pose parameter p_t^i is incremented frame by frame, looping in the range $[0, 1]$. Finally, at time step $t+1$, the graph traversal will receive the data from both real and virtual agents at time t. As a result, the system renders the silhouette of each virtual agent in intermediate images I^{ra}.

5.2.2 Composing the augmented sequence

Once all agents have been processed at time t, a new frame is rendered containing the silhouettes of all the virtual and real agents. In order to provide the resulting image sequence with consistence and realism inside the 3D environment, occlusions between virtual and real agents must be taken into account. Let $A_t = \{a_1, \ldots, a_n\}$ be the list of agents that are moving inside the scenario and let $P_t = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the list of their positions on the ground plane. The agents are sorted in terms of the distances between their current position and the position of the camera C^r. Fig. 5.8 shows one real agent and one virtual agent at distances $d1$ and $d2$ from the camera,

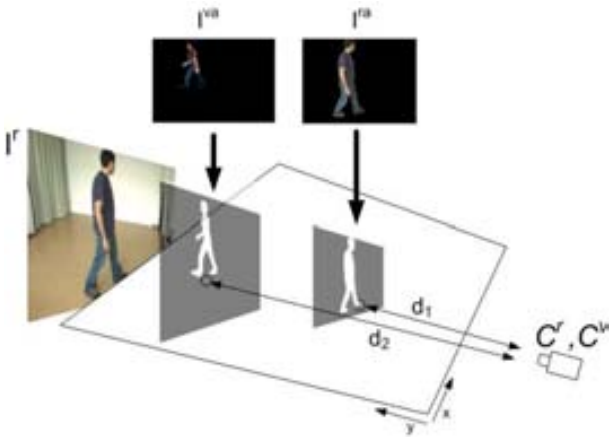


Figure 5.8: Ordering agents by the distance between their position and the camera. More details in the text.

respectively. Since $d2 > d1$, the agents can be sorted in depth. Their silhouette images (I^{ra} and I^{va}) are plotted on the image background and hence agents occlude each other in a consistent manner.

5.3 Experiments

This section describes the experiments performed in order to evaluate the performance of the proposed framework, by simulating virtual agents real indoor and outdoor image sequences.

5.3.1 The *Circle Activity* sequence

The first experiment tests the silhouette estimation and the augmented sequence composition over a simple sequence called *Circle Activity*, where a real agent is walking in circles in a room. The sequence is augmented with a virtual agent, whose behavior consists on walking in circles in front of the real agent. The SGT used to model such behavior is similar to that of the *spiral* behavior, used in Section 5.1.3. Fig. 5.9 shows example frames of the resulting sequence. In frames (a) and (b), the image composition is consistent with the occlusions and with the camera perspective. However, in (c) the leg of the virtual agent appears in front of the real agent, although the real agent is closer to the camera position and is thus rendered after the virtual agent. This is due to errors in the segmentation, in particular due to the similarity between skin color and floor color, which produces a camouflage and thus the silhouette is wrongly estimated.

(a) (b) (c)

Figure 5.9: Example frames of the augmented *Circle Activity* sequence.

5.3.2 The *Police Behavior* sequence

The purpose of the second experiment is demonstrate the feasibility of the SGT in modeling human and vehicle behavior taking into account external affectors produced by real agents in the scenario. Moreover, this experiment tests the extensibility of the approach to work not only with tracking information, i.e. real agent positions and silhouettes, but also with additional information about body action. To this end, a sequence has been recorded in the real urban scenario introduced in last chapter. In this second sequence, called the *Police Behavior* sequence, a single real agent acts as a policeman, who is recorded while giving driving directions near the pedestrian crossing, thereby allowing the pass of vehicles and pedestrians alternatively at certain time intervals.

The position of the policeman is tracked over time and its action is recognized using Motion History Images [17]. In an MHI H, pixel intensity is a function of the temporal history of motion at that point. The result is a scalar–valued image where more recently moving pixels are brighter. We then develop the recognition method matching temporal templates against stored instances of multiple views for known actions. The method automatically performs temporal segmentation, is invariant to linear changes in speed, and runs in real–time on standard platforms. As a result, the recognized actions are converted into FMTL predicates, see Fig. 5.10:

- $police\_orders\_stop(Policeman)$ indicates that the policeman is authorizing the pass to pedestrians, and vehicles must stop.
- $police\_orders\_pass(Policeman)$ indicates that the policeman is authorizing the pass to vehicles, and therefore pedestrians must stop.

Regarding the scene model, four semantic labels have been considered, namely *road*, *crosswalk*, *sidewalk*, and *waiting line* (the area separating sidewalk and crosswalk). The following SGTs have been designed to model pedestrian and vehicle behavior, respectively, taking into account the policeman actions:

- *Crossing the Crosswalk (CC)*, depicted in Fig. 5.11: A virtual pedestrian is located somewhere in the scenario. After instantiating the general situation *sit_ED_SIT0*, two possible specializations are available as starting situations: *sit_ED_2*, if the agent is on the sidewalk, or *sit_ED_3*, if the agent is already in

(a) (b)

Figure 5.10: The actions performed by the policeman in the *Police* sequence allow to instantiate two predicates: (a) $police\_orders\_stop(Police)$ (b) $police\_orders\_pass(Police)$.

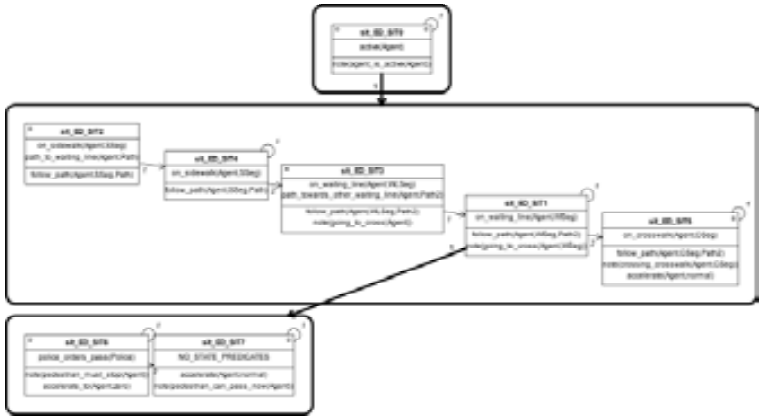


Figure 5.11: Pedestrian behavior for the Police experiment (CC). The predicates $police\_orders\_stop$ and $police\_orders\_pass$ are the police action recognized for each time step.

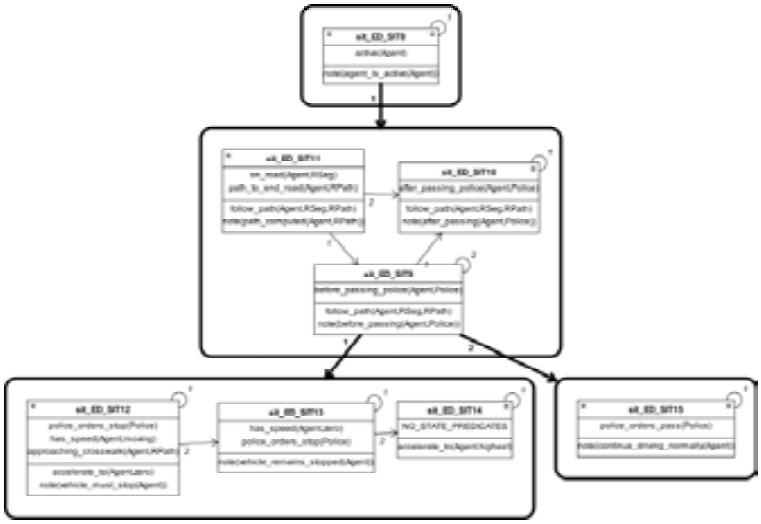
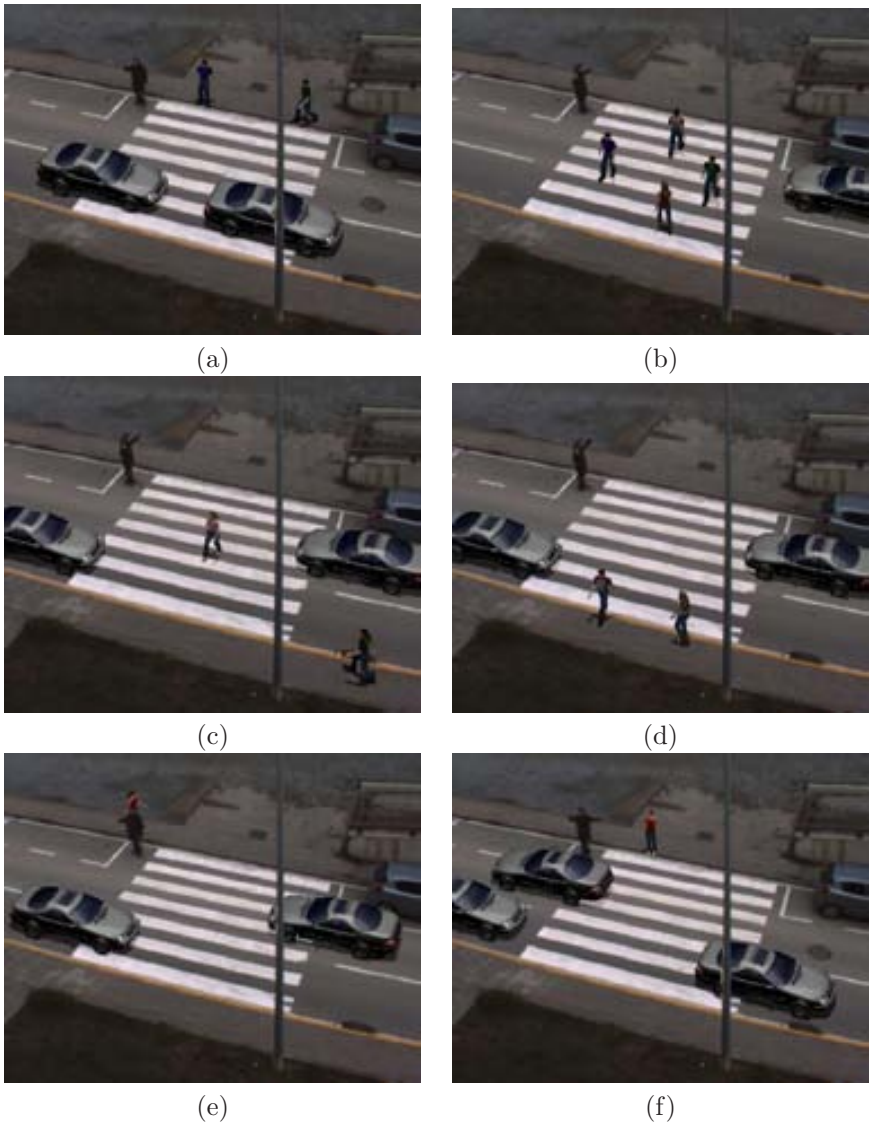


Figure 5.12: Vehicle behavior for the *Police* experiment *DR*. The predicate *police_orders_stop* represents the police action authorizing the pass to pedestrians. In that case, the other action (the police authorizing the pass to the vehicle) is not considered for this model since no specific behavior needs to be performed when such action is recognized.

the waiting line. In the first case, the predicate *path_to_waiting_line* computes the minimum path for the agent to reach the waiting line and the agent follows the path for the following time steps. When in the waiting line, the predicate *path_towards_other_waiting_line* computes the path to cross the crosswalk. Then, situation *sit_ED_SIT1* is instantiated and the possible specialization is tested. If the predicate *police_orders_pass* is satisfied for that precise time step, the situation *sit_ED_6* is instantiated, producing the agent to stop before crossing. When the policeman action changes, the agent accelerates and follows previously computed path, reaching the other waiting line and eventually leaving the scene. On the other hand, if the predicate *police_orders_pass* was not satisfied when situation *sit_ED_SIT1* was instantiated, the agent would continue the path without stopping.

- *Driving on the Road (DR)*, depicted in Fig. 5.12: A virtual vehicle appears from one of the road limits. The situation *sit_ED_SIT11* is instantiated and the predicate *path_to_end_road* computes the path to reach the end of the road. Then, if the vehicle has not passed the policeman, then the *sit_ED_SIT9* is instantiated and two possible specializations are available. On the one hand, the vehicle stops before crossing the crosswalk if the predicate *police_orders_stop* is satisfied at the current time step (*sit_ED_SIT12*), and the vehicle is approaching the crosswalk. In any other case, the vehicle continues driving normally by the road until it reaches the end of the scenario.

Fig. 5.13 shows sample frames obtained after simulating 20 virtual agents, 10



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

Figure 5.13: Result frames of the augmented *Police* sequence.

vehicles and 10 humans, in the *Police* sequence: (a) The policeman grants the pass to pedestrians and vehicles stop before the crosswalk; (b) Vehicles are granted to pass and pedestrians wait in the waiting line (c) Policeman silhouette is consistently maintained in the augmented sequence.

Finally, another sequence, so–called *Outdoor* sequence, has been recorded in order to refine interaction between virtual agents. Thus, the *CC* behavior has been specialized as follows: When being in a crosswalk, the virtual agent checks if there is a vehicle crossing at the same time. If so, the virtual agent raises the arm as a complaint. Fig. 5.14.(a) shows a sample frame of the real sequence I^r. The result of tracking is shown in Fig. 5.14.(b). A virtual agent has been added to the sequence and it arrives to the crosswalk while one real vehicle is crossing. The agent detects a danger of runover situation and then behaves as explained above, see Fig. 5.14.(c). Since the system is working in ground–plane coordinates, we have independence from the point of view, so a multi–camera system can reproduce the same result, as shown in Fig. 5.14.(d).

5.3.3 Quantitative Evaluation

The results shown in this section have been performed using a Pentium D 3.21 GHz with 2GB of RAM, and the system has been developed using *C++* and the *OpenGL* library. The augmented sequences have been obtained using mid–resolution image sequences (696×520 pixels). The system has been quantitatively evaluated in terms of accuracy and scalability.

On the one hand, the overall performance of the system is highly dependent on the data acquisition from the real world, performed by the segmentation and multi–object tracking system. In fact, the estimation of real agent silhouettes conditions on the resulting augmented sequence. The accuracy of the silhouette estimation is analyzed using a manually labeled ground–truth from the *Outdoor* sequence, and the percentage of false positives (FP) and false negatives (FN) is computed as follows:

$$\begin{aligned} FN(\%) &= (\#I_{FN}/\#I_{GT}) * 100 \\ FP(\%) &= (\#I_{FP}/\#I_{GT}) * 100 \end{aligned} \tag{5.5}$$

where I_{GT} is the ground–truth image, and I_{FP}, I_{FN} contain the false positives and negatives respectively.

The obtained FP and FN are shown in Fig. 5.15, and have been compared to other existing approaches in background subtraction [60, 93] by using 4 image sequences sampled from the *Outdoor* sequence.

Subsequently, the scalability of the AR generation is tested, where AR generation involves (i) simulation and rendering of each virtual agent, and (ii) composition of the augmented sequence. Fig. 5.16 shows a serie of speed tests performed by varying the number of virtual agents and the quality of rendering in the *Police* sequence. The human and vehicle agents performed the *CC* and *DR* behaviors, respectively, and all agents appeared in the scenario simultaneously. The maximum frame rate of the system ($25fps$) is achieved in most of the cases. However, as the number of

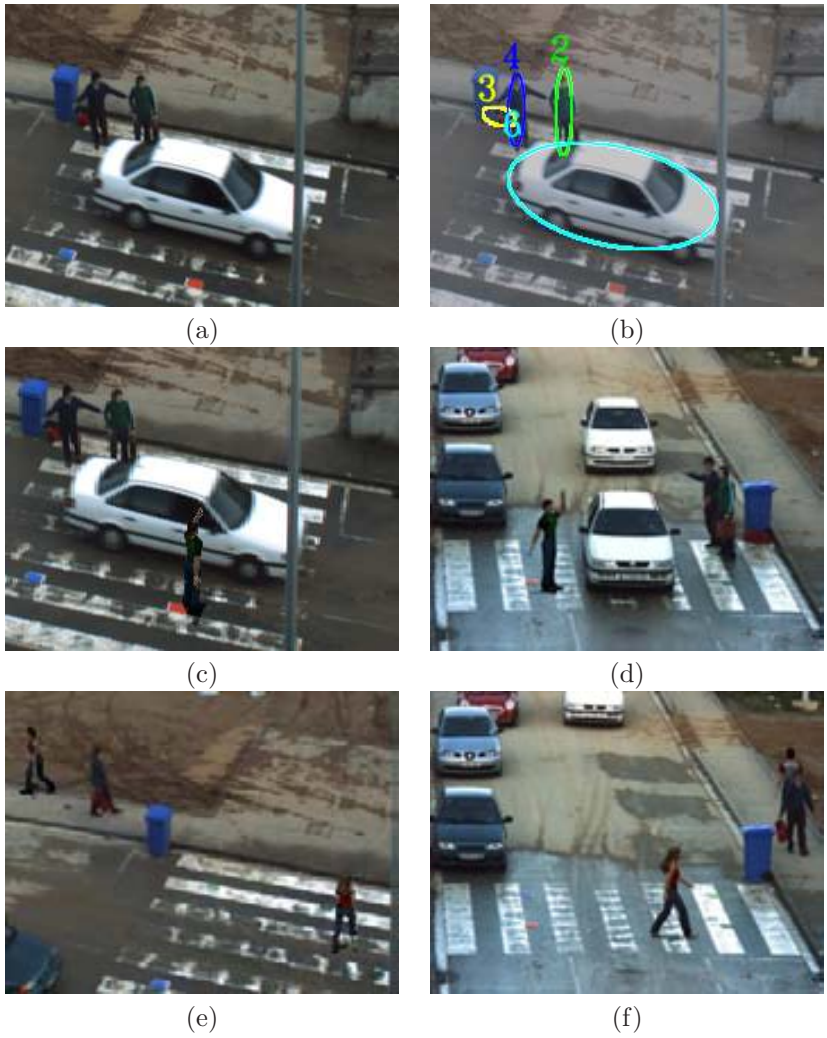
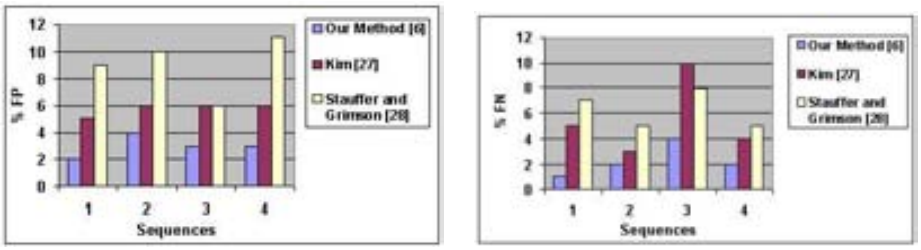


Figure 5.14: Result frames of the steps shown in Fig. 5.7. (a) Original frame (b) Graphical output of the tracking algorithm (c) Augmented frame with one virtual agent (d) Same frame from another point of view. (e-f) Another example from different views, which shows the spatial consistency between real and virtual agents. More details in the text.



(a) (b)

Figure 5.15: Silhouette segmentation errors (a) False Positives (FP) (b) False Negatives (FN)

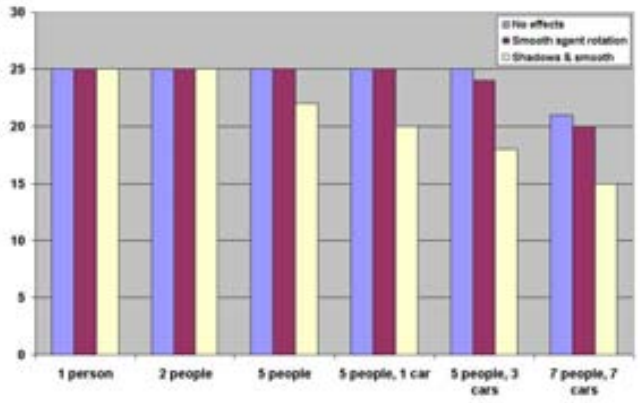


Figure 5.16: Evaluation of the rendering frame–rate by increasing the number of simultaneous agents.

agents grows, the SGT traversal needs to recompute paths to avoid collisions, thereby decreasing the rendering frame–rate.

5.4 Applications: Evaluation of Tracking

In this work we focus on the evaluation of tracking systems specialized in open–world image sequences. Within this context, the tracked targets are expected to be the typical elements in urban environments, i.e. pedestrian, animals and vehicles.

Nowadays, research in multi–object tracking algorithms has achieved great results when tracking non–grouped targets in few frames image sequences including non–severe illumination changes. However, several problems, inherent to computer vision in general, are still unsolved and constitute a big challenge towards an unconstrained multiple–target tracking, namely long occlusions, grouping disambiguation and camouflage. So, performance evaluation of multi–objects tracking algorithms

for video–surveillance has recently received significant attention [107, 98]. Indeed, evaluation metrics for surveillance are almost as numerous as multi–object tracking methods themselves.

One typical drawback comes up when evaluatin the tracking algorithm in a different environment or image sequence [14]. As long as difficult frames appear in the new image sequence, some modifications must be done to the algorithm (parameters, thresholds) in order to achieve good results. However, these modifications could damage the results obtained in former image sequences. Instead of using completely new sequences, it would be useful to have a method to gradually increase the difficulty of a given sequence. However, when recording a new sequence, even in the same scenario, we are exposed to several condition changes due to illumination, weather, or scenario configuration. In addition, it is sometimes hard to record image sequences containing crowds of people in public urban environments for legal or security reasons.

5.4.1 Evaluation Framework

This section describes two different methods to evaluate the performance of tracking algorithms. Firstly we define the elements involved in our framework. Let $Tracker$ be the tracking algorithm to evaluate and let S^r be a real scenario. An image sequence I^r is recorded using a static camera C^r. Let O^r be the output obtained when processing I^r by $Tracker$. Although this output may vary from different trackers, most of them include either a list of target positions for each frame, *frame–oriented* [98], or a list of targets including their tracked trajectory, *target–oriented* [45]. In this work we assume a target–oriented output, therefore

$$O^r = \{T_1^r, \ldots, T_n^r\} \tag{5.6}$$

is a list of tracked targets where $T_i^r = \{t_{i_1}, \ldots, t_{i_k}\}$ is the sequence of position coordinates in each frame step for the target i, see Eqs. 5.1,5.3.

Subsequently, a synthetic scenario S^v is modeled by using the real scene dimensions obtained with the calibration of C^r, see Fig. 5.6. Finally, a virtual camera C^v is located in S^v at the exact location of C^r in S^r.

In order to generate a synthetic sequence with moving targets, a set of virtual agents is added to the created environment. Thus, two types of moving targets are considered so far, pedestrians and vehicles. The development of a virtual agent inside the environment is determined by one of the following strategies:

- *Trajectory–based*:a predefined, possibly manually provided trajectory $T = \{t_1, \ldots, t_k\}$ is assigned to the virtual agent, where t_i defines the agent position in the frame i.

- *Behavior–based*: virtual agents are provided with SGT–based behavior models which automatically generate the trajectories. This strategy better represents the desired simulation since it is aware of interactions between the agent and other participants in the scenario.

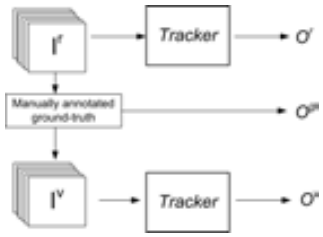


Figure 5.17: Comparison of two outputs of the tracker with ground-truth trajectories (O^{gt}). O^r is the output given by a real image sequence I^r and O^v is the output obtained from processing the synthetic image sequence I^v, which has been generated using n virtual agents performing the ground–truth trajectories $O^{gt} = \{T_1^{gt}, \ldots, T_n^{gt}\}$.

Generation of synthetic sequences using the ground truth

A common strategy to evaluate tracking performance is to compare an output with the corresponding ground–truth data. In our case, the ground–truth is defined as a list of targets obtained by means of manual annotation of the trajectories observed in I^r, and it is considered as the ideal output O^{gt} of the tracker:

$$O^{gt} = \{T_1^{gt}, \ldots, T_n^{gt}\} \tag{5.7}$$

where n corresponds to the number of targets. Given the ground–truth of I^r, a synthetic image sequence is generated by modeling a set of virtual agents performing the trajectories stored in O^{gt}. The resulting sequence, namely I^v, is processed by the tracker and an output O^v is obtained, see Fig. 5.17.

The evaluation metric is based on the comparison between the ground–truth O^{gt} and the outputs O^r and O^v which are the results obtained by the tracker when processing the real sequence and the virtual sequence, respectively. In order to perform a qualitative evaluation of the results, a list of events is extracted by analyzing the trajectories contained in O^{gt}, O^r and O^v. These events can be related either to low or high–level feature events. On the one hand, events like the target interaction with the scene boundaries (*entering* or *leaving* the scene) or occlusions between agents (*start / end* occlusion) require to analyze and compare the trajectory positions over time. On the other hand, higher level events require a semantic evaluation of the trajectories [49]. This implies to deal with semantic knowledge about human behavior in the selected environment, as well as to know the semantic explanation of every part of the scene [10]. For example, if a part of the scenario is labeled as *crosswalk*, a high–level event can be defined as *entering / exiting the crosswalk*.

Augmenting scene complexity by adding virtual agents

In order to permit a gradual increase of complexity of a previously recorded image sequence I^r recorded inside a real scenario S^r, the number of moving targets should be increased. Thus, the more of agents involved in the scenario, the more complex the image sequence will be. However, the initial conditions of the scenario must be kept in order to avoid distortions prompted by e.g. illumination changes or alterations of

the background configuration. To this end, a set of m virtual agents $\{A_1, \ldots, A_m\}$ is modeled, either by assigning predefined trajectories or by simulating behavior models, and those are rendered into a synthetic image sequence I^{va}. Finally, I^r and I^{va} are fused into a new image sequence I^{rv}, containing real and virtual agents in the real scenario S^r, see Fig. 5.7. This resulting image sequence increases complexity in terms of occlusions, camouflages, and events.

The obtained image sequence is processed by the tracker, thus generating an output O^{rv}. The evaluation of the results implies the comparison with a ground–truth O^{gtrv}, which is a combination of the original ground–truth O^{gt} and the trajectories $\{t^{A_1}, \ldots, t^{A_m}\}$ performed by the virtual agents:

$$O^{gtrv} = \{t_1^{gt}, \ldots, t_n^{gt}, t^{A_1}, \ldots, t^{A_m}\} \tag{5.8}$$

Finally, given O^{rv} and its corresponding ground–truth O^{gtrv}, the event–based metric explained above is applied to evaluate the tracker performance.

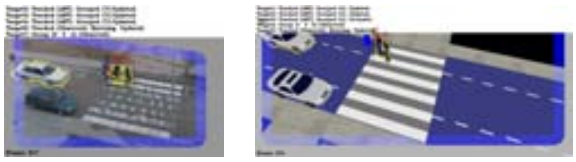
5.4.2 Case Study: Evaluation in a pedestrian environment

The architecture of the analyzed tracking algorithm is based on a modular and hierarchically-organized system. A set of co-operating modules, which work following both bottom-up and top-down paradigms, are distributed through three levels. Each level is devoted to one of the main different tasks to be performed: Target Detection, Low-Level Tracking (LLT), and High-Level Tracking (HLT) [88]. Since high-level analysis of motion is a critical issue, a principled management system is embedded to control the switching among different operation modes, namely motion-based tracking and appearance-based tracking. This tracking algorithm has the ability to track numerous targets while they group and split while maximising the discrimination between the targets and potential distracters.

Virtual Environment

The proposed tracking evaluation approach has been applied into the pedestrian crossing scenario, introduced in Section 5.3.2. In order to obtain augmented reality image sequences on the selected scenario, three simple behavior models related to pedestrian crossings have been defined:

- *Walking on the Sidewalk (WS)*: A virtual human agent is located over the scenario. It reaches the closest sidewalk and exits the scenario walking to the farthest point of the sidewalk.
- *Crossing the Crosswalk (CC)*: A virtual human agent is located over the scenario. It first reaches a sidewalk (if in another region) and then moves to the crosswalk. After crossing it, the agent leaves the scene walking through the opposite sidewalk.
- *Driving on the Road (DR)*: A virtual vehicle agent is located over the scenario. It reaches the road (if in another region) and finally leaves the scene driving to the farthers point of the road.



(a) (b)

Figure 5.18: Results of the tracking performed to (a) Real image sequence I^r (b) Synthetic image sequence I^v generated using the ground–truth trajectories from I^r.

Events	Ground–truth O^{gt}	Output O^r	Output O^v
Entering Scene	6	6	6
Exiting Scene	4	4	4
Starting Occlusion	5	5	5
Ending Occlusion	5	5	5
Entering Crosswalk	6	6	7
Exiting the Crosswalk	5	5	6

Table 5.1
EVENT RECOGNITION RESULTS IN BOTH THE REAL AND VIRTUAL SEQUENCES, COMPARED TO THE HAND–LABELLED GROUND–TRUTH

Evaluation Results

Two different experiments have been applied to the real scenario and its conversion to virtual environment. On the one hand, the real image sequence I^r and its correspondent synthetic image sequence I^v have been processed by the tracker, see Fig. 5.18. The outputs obtained have been compared in terms of low–level and high–level events and the results are shown in Table 5.1. As can be observed, the tracker matched the ground–truth with the real sequence. However, it has detected a false positive for the events *Entering / Exiting crosswalk* in the synthetic sequence, produced by little differences in the measurement of the scene dimensions.

On the other hand, a new image sequence I^{rv} has been obtained by fusing the original sequence I^r with a synthetic image sequence I^{va} generated by simulating 30 virtual agents. 15 agents were assigned the *CC* behavior, 10 were assigned the *WS* behavior, and 5 were assigned the *DR* behavior. This sequence has been then input to the tracker and an output O^{rv} has been obtained. In order to balance this result, a new ground–truth has been computed, joining O^{gt} with the trajectories generated after behavior simulation. The results are depicted in Table 5.2. The *entering / exiting scene* events were successfully recognized given the tracking results. However, due to camouflage, the number of occlusions detected is higher than the annotated ground–truth. Finally, most of the high–level events *entering / exiting crosswalk* have been detected.

(a) (b) (c)

Figure 5.19: Tracking results in the augmented reality sequence. Example frames of the obtained image sequence containing both real and virtual agents.

Events	Ground–truth O^{gtrv}	Augmented Reality Tracking O^{rv}
Entering Scene	36	36
Exiting Scene	35	35
Starting Occlusion	17	21
Ending Occlusion	17	21
Entering Crosswalk	20	19
Exiting the Crosswalk	20	19

Table 5.2
EVENT RECOGNITION RESULTS FOR THE TRACKER T PROCESSING THE AUGMENTED REALITY SEQUENCE, COMPARED TO THE AUGMENTED REALITY GROUND–TRUTH, WHICH HAS BEEN OBTAINED BY JOINING THE HAND–LABELLED GROUND–TRUTH OF THE REAL SEQUENCE WITH THE TRAJECTORIES GENERATED BY THE VIRTUAL AGENTS.

5.5 Applications: Interactive Virtual Storytelling

Virtual storytelling tools facilitate external users to generate dynamic content such as image or video sequences, by writing natural language plots related to certain scenarios, see Fig. 5.20. The main concern of this application is to provide a flexible and natural solution to produce generally complex sequences automatically. One of the main current challenges on these fields consists of bringing complex high-level modeling closer to the final users, so that it becomes both intuitive and powerful for them to automatically generate mixed scenes. The interactivity of the users is thus a key factor on these research lines.

This work is an extension of the general approach presented in Section 5.2 and is based on an interface that allows users to extend real image sequences with virtual agents, by introducing goals for them at specific points along the video timeline. Once a plotline is given for a certain time-step, the virtual agent tries to accomplish its goal by following a defined scene model. The user can interactively extend the behaviors of the agents in an easy and flexible way, being enabled to define arbitrarily complex occurrences. In order to improve the naturalness of the virtual agents while performing their goals, we base their concrete trajectories on the patterns learnt from observed real agents in the scenario.

In this section, we first introduce the Natural Language Understanding (NLU) module, which converts the plotlines into a set of specific goals. Subsequently, goals

Figure 5.20: The goal of Virtual Storytelling is converting a text into an image sequence.

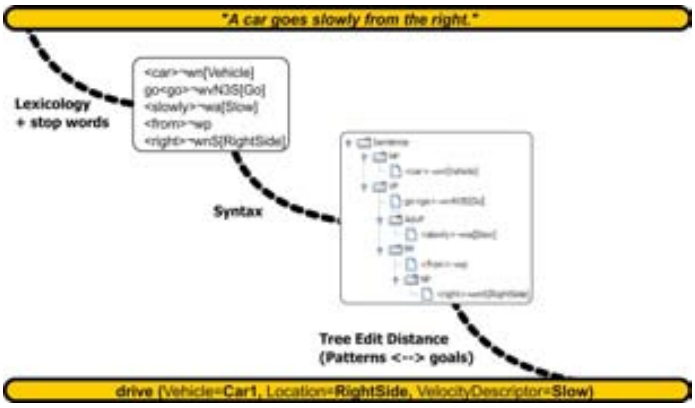


Figure 5.21: Example showing the layout of the NLU module, in which a plotline is converted into the most reasonable conceptual predicate available. Left side of the figure shows the input text and the intermediate results of the analysis until the final predicate. Right side contains rules and patterns applied for this specific example.

are structured into SGTs thereby defining virtual agent behavior. Then, we introduce a sampling technique to produce virtual agent trajectories, based on a previously learnt behavior model. Finally, we show experiments in two different scenarios to demonstrate the flexibility of the proposed approach.

5.5.1 From Natural Language to Goals

NLU (Natural Language Understanding) has usually been regarded as a process of hypothesis management that decides for the most probable interpretation of a linguistic input [86]. Following this idea, the NLU module links plotline sentences to their most accurate interpretations in the domain of interest, in form of high-level predicates referring to known concepts or instances within the scene.

Once a proper formatting has been applied, an input sentence is analyzed through a sequence of 3 processes: first, a morphological parser tags each input word with linguistic features depending on the context of apparition. Secondly, a syntactic/semantic parser recursively builds a dependency tree for the tagged sentence. Finally, the resulting dependency tree, already having ontological references, is assigned to the most

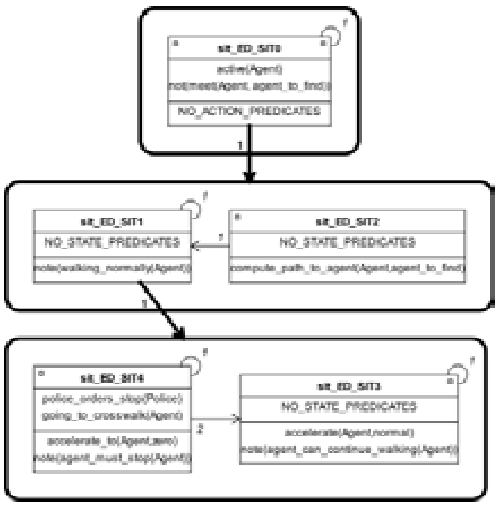


Figure 5.22: Example of SGT for Virtual Storytelling.

related high-level predicate found [38, 36]. Fig. 5.21 depicts the three referred process in the processing of the sentence *A car goes slowly from the right*. The morphological and syntactical processes convert the NL sentence into a tree representation, which is finally converted into a goal predicate measuring the distances from the tree to the predicates stored in the ontology.

Subsequently, each plotline predicate produced by the NLU module instantiates a high-level event, which must be converted into a list of explicit spatiotemporal actions. At this point, we are interested in providing the user with a device to define behavioral patterns for the agents, still keeping it an intuitive solution with interactive operability. The proposed conceptual planner is based on the framework formed by the reasoning engine FMTL and the behavior modeling tool SGT, described in Chapter 4.

Finally, each high-level predicate is decomposed into a temporal sequence of lower-level objectives. For instance, we may want to define a pedestrian situation *"P1 meets P2"* as the sequence (i) *"P1 reaches P2"*, and (ii) *"P1 and P2 face each other"*, or translated into FMTL predicates:

$$meet(P1, P2) \vdash go(P1, P2) \rightarrow faceTowards(P1, P2) \vee faceTowards(P2, P1) \quad (5.9)$$

Path Manager

The final step of the top-down process requires deciding about the trajectories to be performed by the virtual agents. Concretely, storytelling plots cannot fully specify the motion patterns of these agents, out of few conceptual scene locations considered by the domain. We solve this by sampling normal trajectories from the statistical route models learnt using the Off–line Scenario Learning algorithm described in Chapter 3, Section 3.3.

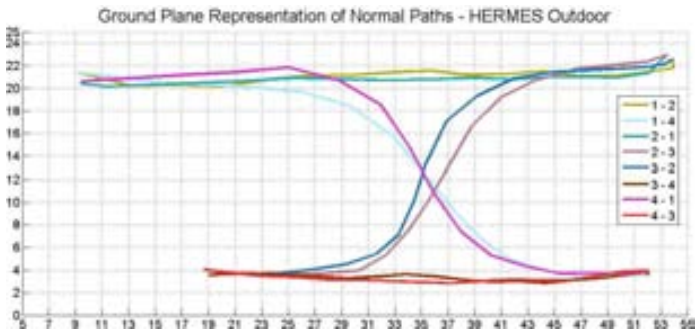


Figure 5.23: Roadmap for the HERMES Outdoor Scenario.

Hence, the route models contained in $\mathcal{M}$ are treated as a roadmap which drive agent trajectories within the scenario. Fig. 5.23 depicts the roadmap obtained for the HERMES Outdoor Scenario. Fig. 5.24.(b) shows a example case of sampling from M to obtain a path from the depicted start and goal locations. The red, blue and green circles represent intersections, splits and merge of paths, respectively. The magenta curve represents a spline.

Given an initial position (x_0, y_0) and a goal position (x_1, y_1), we attempt to create a path from (x_0, y_0) to (x_1, y_1) based on the route models contained in $\mathcal{M}$. The recursive algorithm is explained as follows:

1. Search for a route R_c from the routes $P \in \mathcal{M}$ starting near (x_0, y_0) and ending near (x_1, y_1).

2. If a match is found, then we return a route sampled from R_c.

3. Otherwise, we look for intersections between the paths starting near (x_0, y_0) and the paths ending near (x_1, y_2)

4. If there is a match, we return a route computed as the proper combination of intersected paths.

5. Otherwise, we create the list $S = \{(x_1^I, y_1^I), \ldots, (x_n^I, y_n^I)\}$ formed by all the intersections between paths starting in (x_0, y_0) and the rest of the routes in $\mathcal{M}$. Then, we apply recursively $1 - 3$ for each position $(x^I, y^I) \in S$.

After applying this algorithm, the output contains a combination of paths in $\mathcal{M}$, starting and ending near the desired positions. The resulting path maintains the same structure as the original paths in $\mathcal{M}$, and therefore consist of a sequence of gassian models:

$$R_c = \{G_1, \ldots, G_k\} \tag{5.10}$$

The gaussian models in R_c are sampled to generate a sequence of control points $(p_1, \ldots, p_n)$. An intermediate trajectory is obtained by fitting a spline into $(p_1, \ldots, p_n)$.

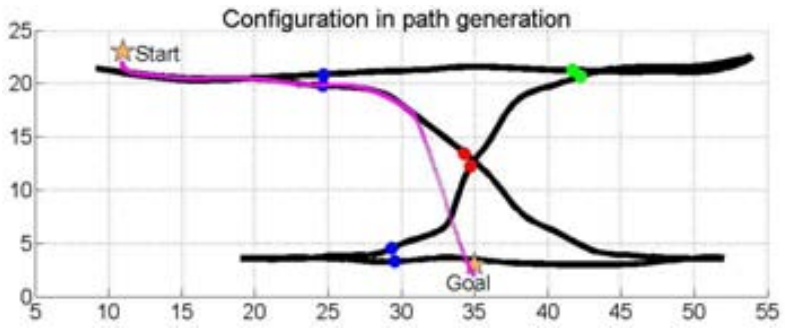


Figure 5.24: Example of path generation.

Figure 5.25: Selected frames from the first VST experiment.

Finally, the spline is sampled again to compute the virtual agent position for each time step in the image sequence.

It may happen that no path has been generated, since there is no matching combination of paths in M. In that case, the path would be generated in a deterministic way, as explained in Section 5.1.3.

Experiments

We have conducted two experiments in order to test the feasibility of the proposed framework. Those experiments comprise the generation of augmented sequences in the HERMES Outdoor and Kingston Dataset scenarios. An external user accesses the NL interface to provide plots for virtual agents. Then, the user receives the augmented scene and is allowed to interactively change the plots or models towards a desired solution.

The first experiment includes several pedestrians walking around an open scenario. Since the system has learnt from the observed footage of real agents in the location, virtual agents incorporate knowledge about typical trajectories to reach to any point. Virtual agents select the shorter of the learnt paths, or the one that avoids collisions with other agents, depending on the circumstances. The plot tells the agent to go to different zones A, B, or C at different moments of time. Fig. 5.25 shows three snapshots of a example augmented sequence obtained through the concurrent simulation of virtual agents.

The second experiment relies on the influence of real world into virtual agent

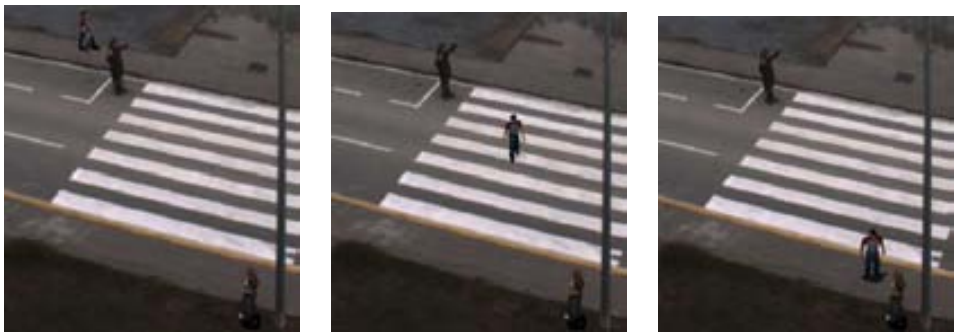


Figure 5.26: Selected frames from the second VST experiment.

decisions. To this end, we use the previously described *Police* sequence, where a real police agent gives way to vehicles or to pedestrians. There, a virtual agent can either wait in the sidewalk or directly enter the crosswalk, in order to meet somebody in the opposite sidewalk. Depending on the action being performed by the police, the virtual agent will have to wait before crossing the crosswalk.

Fig. 5.26 shows three snapshots of the augmented sequence using the path generated in Fig. 5.24. The plot used for that sequence is as follows: "*A person is standing at the upper crosswalk. A second pedestrian appears by the lower left side. He meets with the first pedestrian.*".

5.6 Discussion

This chapter has demonstrated the capabilities of the framework formed by FMTL and SGTs to build behavior models not only to recognize and understand activity in image sequences, but also to generate synthetic activity in terms of virtual agents. The resulting simulation has led to combine virtual and real agents into consistent augmented image sequences where virtual agents adapt their behavior depending on the activity observed in the scenario.

Virtual agent behavior has been modeled based on intermediate goals, which are expressed in terms of FMTL predicates. Those goals are hierarchically and sequentially organized within a Situation Graph Tree. On the one hand, the sequential organization of goals has enabled the specification of temporally ordered activities, but without assigning a predefined duration to each event. On the other hand, the hierarchical representation of situations has permitted modeling behaviors with any desired level of specificity. Moreover, FMTL represents a homogeneous interface to represent any sort of input data that can help to enrich the modeled behaviors, e.g. the predicates used to indicate policeman actions in Section 5.3.2.

The consistence of the subsequent image composition is highly dependent on the results obtained by the segmentation and tracking processes, devoted to obtain the real agent silhouettes and their positions on the ground plane. We have shown results in different sequences, where the current state–of–the–art methods show good performance. However, there still exist problems that are currently under research in the

image processing community, such as long–occlusions, crowded scenarios, cluttered environments and camouflage.

The proposed augmented reality framework has a wide set of potential applications, and in this chapter we have shown two in different domains. First, we have defined a new tracking evaluation method based on behavioral virtual agents that augment the complexity of a given scenario. Second, an extension of the framework has been applied to interactive virtual storytelling, producing image sequences from a Natural Language input.

5.7 Resum

Aquest capítol ha demostrat l'aplicació de l'entorn format per FMTL i AGS per construir models de comportament que serveixen no només per a reconèixer sino també per generar activitat sintètica en agents virtuals. La creació d'aquests agents i la seva simulació en entorns virtuals ha permés combinar agents virtuals i reals en seqüencies d'imatges augmentades, on els agents virtuals modifiquen el seu comportament depenent de l'activitat observada en un escenari real.

El comportament dels agents virtuals està basat en l'aconseguiment de fites intermitges, definides mitjançant predicats FMTL. Aquestes fites s'organitzen jeràrquica i sequencialment mitjançant Arbres de Grafs de Situacions (AGS). En primer lloc, l'organització sequencial de les fites permet l'especificació d'activitats ordenades temporalment, sense assignar una durada específica a cada event. D'altra banda, la representació jeràrquica de les situacions permet modelar comportaments amb el nivell desitjat d'especificitat. A més, FMTL representa una interfície homogènia per a representar qualsevol tipus de dada d'entrada que ajudi a enriquir els comportaments modelats, per exemple els predicats que representen les accions del policía dins la Secció 5.3.2.

La consistència de la composició d'imatges per a la seqüència augmentada depen dles resultats obtinguts pels sistemes de segmentació i seguiment, encarregats d'estimar la posició i la silueta dels agents reals dins l'escenari. Hem mostrat resultats en diverses seqüències, on els mètodes actuals de segmentació i seguiment mostren un bon funcionament. Tanmateix, encara existeixen problemes, com per exemple llargues oclusions o camuflatge, per als quals no s'ha trobat una solució general.

L'entorn de realitat augmentada proposat té un gran conjunt potencial d'aplicacions, de les quals n'hem mostrat dues en aquest capítol. En primer lloc, hem definit un mètode per avaluar el funcionament de mètodes de seguiment en seqüències d'imatges. El mètode està basat en agents virtuals que exhibeixen comportaments, afegint gradualment complexitat a un escenari concret. Per altra banda, hem proposat una extensió dins del marc de la narrativa virtual, per tal de generar seqüències d'imatges a partir d'un guió expressat en llenguatge natural.

Chapter 6

Concluding Remarks

6.1 Summary

In this thesis, we have focused on developing behavior models for the analysis and generation of image sequences. Initial chapters have been devoted to tackle the problem of analyzing the observed behaviors in video–surveillance scenarios. Consequently, two different approaches for behavior analysis in images sequences have been described and confronted.

First, the challenge of learning behavior models from the observation of activity in surveillance scenarios has been faced. To this end, we proposed a statistical method to learn motion patterns and to build up a *normal* behavior model in particular regions of the scenario. Using this model, we have been able to differentiate between normal and abnormal behavior in future observations. Moreover, the obtained model allowed to predict the development of incoming observations under normal conditions. Finally, the model we sampled from the obtained models in order to generate synthetic instances of normal behavior.

However, we have shown that the unsupervised learning of behavior models from quantitative data does not take care of the inherent semantics of human behavior. In order to tackle this issue, we have proposed to use a deterministic top–down approach for the analysis of human behavior which takes advantage of prior semantic knowledge about the enviornment and about the behaviors that we expect to observe there. Thus, we used the framework formed by the reasoning formalism called Fuzzy Metric Temporal Logic and the behavior modeling formalism called Situation Graph Tree for the recognition and description of a predefined set of human behaviors in outdoor scenarios. The applicability of the proposed approach has been tested into different discourse domains: video–surveillance and sport shows.

Subsequently, we have exploited the reaction capabilities of the SGT formalism to generate synthetic instances of behaviors and thus simulate synthetic agents in virtual environments. We have described a robust framework to simulate the development of concurrent virtual agents that are able to interact and show complex behaviors in a selected environment. A further step has been done to combine the SGTs for recognition with the SGTs for simulation , thereby obtaining interactions between

real and virtual agents. This has led to the generation of augmented reality (AR) sequences in where real and virtual agents exist in a consistent manner.

The unified approach to produce AR sequences has been tested on two application domains. First, we have described a virtual storytelling system that converts a natural language explanation into its representation in images. A natural language understanding (NLU) module is used to convert sentences into predicates that represent the goals to be accomplished by specific virtual agents. The acchievement of these goals generated a virtual story and finally rendered into an augmented image sequence. Second, we faced the problem of tracking evaluation by providing a mechanism to gradually increase the complexity of a given scenario. This is achieved with the addition of behavior–based virtual agents into a real scenario, thereby generating new targets to be observed by the tracker. Since being synthetically generated, these new agents automatically generate a ground–truth that enables a quantitative evaluation of the tracking results.

Concluding, in this Thesis we have explored the current open issues in the long path towards behavior analysis in image sequences and we have proposed a contribution to tackle each of them.

6.2 Discussion and Future Directions

The unavoidable influence of subjectivity in human behavior makes statistical approaches limited to provide an semantic explanation of the activity observed in an uncontrolled environment. However, explicit behavior models depend on the expertise of a human designer. An hybrid method which maintains a semantic layer and keeps learning from the environment seems to be the long–term solution to this problem.

6.2.1 Learning Behavior Models

A considerable part of this Thesis has been devoted to the automatic learning of behavior models by analizing the observations obtained by tracking systems. The generated patterns constitute the definition of normal behavior in the studied scenario. Given the structure of our scenario model, we have presented a proper holistic definition of anomaly in the video–surveillance domain. We have proposed a hierarchical representation of anomalies in terms of Soft Anomalies, Intermediate Anomalies and Hard Anomalies, which are detected depending on the degree of deviation from the learnt model. The classification of incoming observations using this hierarchy provides a richer explanation of the observation than other existing approaches.

All existing approaches, including our proposed methods, are based on the statistical definition of normality and anomaly: Those observations that occur more frequently are considered *normal*, and observations distant from the normal model are considered abnormal. However, considering the video–surveillance context, the obtained model should be able to differentiate between normal and abnormal events in terms of security: Abnormal events should coincide with the *dangerours* events in order to fire an alarm when those are detected. Obviously, the notion of *riskiness* is human–subjective and must be informed beforehand and might be scenario–

dependant. Filling the gap between the two definitions of anomaly constitutes currently the most important open issue for future work in the behavior learning area.

6.2.2 Behavior Description in Image Sequences

The presented top–down framework for the recognition and description of behaviors in image sequences has a strong semantic layer, thus enabling the subsequenct transformation of the predicate list into a natural language explanation [37]. As we have already explained in Chapter 4, the inclusion of semantics in the behavior model enables the subjective distinction between normal and abnormal activities, thus filling the gap described in previous section.

However, the use of predefined semantic knowledge entails some limitations that could restrict the set of application domains and must be therefore taken into account:

- The accuracy of the behavior models depends on human expertise on the selected environment. Also, the level of specificity used for the design of the FMTL predicates and SGTs will condition on the distinction between similar behaviors, e.g. *fighting* or *meeting*. Moreover, the performance evaluation of the framework is acchieved by means of visual inspection, since the correct identification of behaviors in the image sequences is subjective.

- The framework has no learning capabilities, and behavior models will remain unmodified along time. Possible changes in the environment do not affect the models. A reaction of the system to observed patterns could be desirable in order to evolve towards a more robust representation of behaviors.

- The framework is scenario–dependant. The selected set of behavior patterns are especially designed for a given scenario, and might not be applicable to other scenarios, even if they belong to the same discourse domain. For instance, behaviors in two different pedestrian crossing scenarios could differ due to specific environmental conditions (e.g. the existence or inexistence of traffic lights).

- The performance of the system is obviously depending on the accuracy of data acquisition by the computer vision systems, i.e. the detection and tracking processes. The fuzzy component of the FMTL predicates help to reduce the uncertainty produced by the sensory gap. However, there still exist unsolved problems in the field of multi-object tracking, described in Chapter 1, that may produce extremely noisy results, thereby affecting to the resulting behavior descriptions.

The above listed features must be analized while considering the potential applications of the proposed approach. From a scientific point of view, there is still a long ground towards building a complete system combining semantic knowledge with an unsupervised learning capability. Nowadays, researchers are boosting the use of ontologies to standarize a unified representation of common sense knowledge, in order to minimize the amount of prior knowledge that has to be predefined for each different scenario [1].

On the other hand, from a commercial point of view, behavior analysis systems in video–surveillance are usually devoted to a single and controlled environment. Moreover, the set of behaviors is predefined and a differentiation between normal and abnormal behaviors is provided by the client. This fact questions the need of automatically learn behaviors instead of predefining them. In fact, most of the existing learning approaches need to acquire hundreds of observations to use them as a training set. Therefore, the use of predefined behavior models seems to be the most suitable mechanism given the requirements of the potential commercial applications.

Nevertheless, an open issue that must be investigated is how to provide such a deterministic framework with a proper response to the uncertainty carried from the computer vision systems. Nowadays, the fuzzy component of the reasoning engine FMTL is capable to deal with a certain level of noise, but sometimes the environmental conditions make the tracking system fail, thereby losing tracks or confusing two or more tracks due to long occlusions, weather conditions or crowded environments.

6.2.3 Generation of Augmented Sequences

In this Thesis we have focused on the generation of augmented sequences by combining behavioral virtual agents with tracked real agents in real environments. The framework formed by SGTs and FMTL has shown to be feasible for both describing observed behavior in image sequences and generating synthetic behavior of virtual agents.

Future work will be devoted to improve the different topics discussed in Chapter 5. First, we need to provide our framework with more realism in terms of illumination and shadows, as stated in [64]. Also, silhouette estimation in tracking algorithms remains as an open problem when dealing with long occlusions or camouflage, and research towards this end will improve the composition of the augmented sequences.

An important question issued about the proposed method is the lack of feedback provided to the real agent in the augmented sequences. Since virtual agents are added on–line, real agents do not know about their existence. Nevertheless, this problem, which is has not been tackled in this Thesis, can be solved using external devices to provide real agents with knowledge about virtual agents [35, 25]. For instance, a combination of wearable cameras and video glasses would allow the generation of augmented sequences from the point of view of the real agent, see Fig. 6.1. Since the agent positions are expressed in real world coordinates, the augmented sequences can be generated with independence from the point of view. Therefore, the images taken by a wearable camera, whose position is also known in the ground–plane, can be augmented by rendering the virtual agents that currently exist in the environment. The augmented sequence is then transmitted to the video glasses, making the real agent feel that is really interacting with the virtual agents.

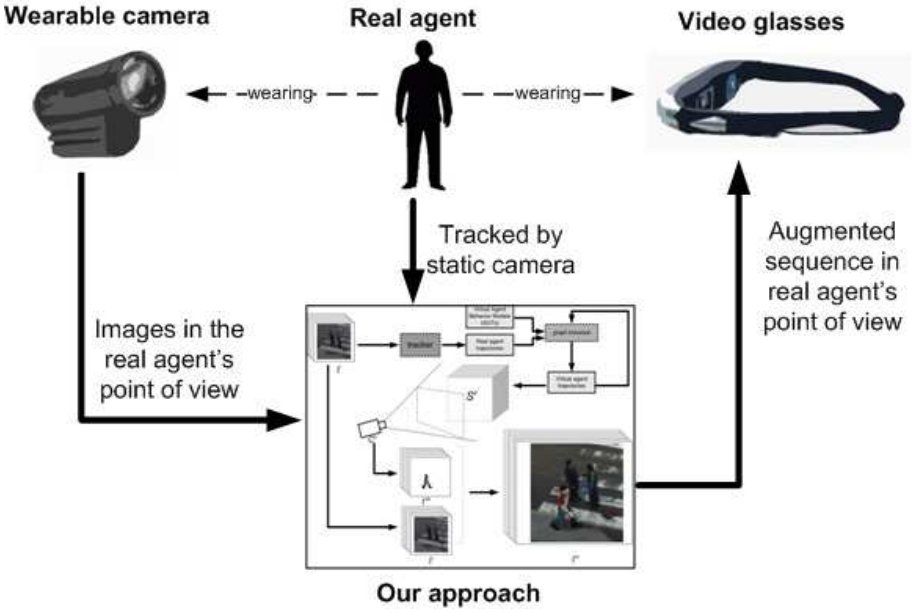


Figure 6.1: Scheme of the extension of our proposed approach in augmented reality in order to provide a feedback to the real agents. More details in the text.

Appendix A

Terminology of Predicates for Behavior Modeling

In this Appendix, we describe the conceptual predicates used for the Situation Graph Trees presented in Chapters 4 and 5. Basically, state predicates described here refer to human actions and interactions with other agents or with modeled static objects within the scene.

A.1 State Predicates Concerning Only the Actor

A.1.1 Spatial Predicates

active(Agent) : this predicate states that the agent **Agent** is presently active, which means on the one hand that there is some state information for that agent and, on the other hand, that this agent is being observed.

has_speed(Agent, Value) : this predicate states that an agent **Agent** has a speed which is associated to the conceptual speed value of **Value**. Acceptable conceptual speed values are *zero*, *small*, *normal*, *high*, *very_high*, and *moving*.

has_direction(Agent, Value) : the agent **Agent** is altering its direction in a way which is associated to the conceptual description **Value**. Possible conceptual descriptions are *straight*, *right*, *left*, and *not_straight*. All these values are interpreted in the agent's ego-centric view.

A.1.2 Posture Predicates

is_performing(Agent, Value) : the agent **Agent** is performing an action which is associated with the conceptual description **Value**. Acceptable conceptual action descriptions are *aBend*, *aJump*, *aKick*, *aRun*, *aSit*, *aSkip*, *aSquat*, *aTumble*, and *aWalk*.

is_standing(Agent) : the agent **Agent** is standing in an upright position. Commonly (not always), no spatial motion is observed for the agent.

is_performing_transition(Agent) : the agent **Agent** is performing a transition between two different actions (in the *thief* sequence, for example, there is a transition from *aBend* to *aRun*).

has_finished(Agent) : the agent **Agent** has finished to perform a given non-cyclic action (like *aBend* or *aJump*), or the agent has stopped to perform a cyclic action (like *aRun* or *aWalk*).

A.1.3 State Predicates Concerning Another Actor

is_alone(Agent, Proximity) : the agent **Agent** is alone within a circle centered on the agent's position. The radius of the circle is defined by the conceptual description **Proximity**. Possible conceptual descriptions are *nearby*, *halfway*, and *faraway*.

have_distance_change(Agent, Patiens, Value) : the agent **Agent** and another agent **Patiens** are moving in such a way that their distance is changing. The change of their distances is associated with the conceptual value **Value**, whose possible values are *constant*, *diminishing* and *growing*.

have_distance(Agent, Patiens, Value) : the agent **Agent** and another agent **Patiens** are separated by a distance associated with the conceptual value **Value**, whose admissible values are *small*, *normal*, and *high*.

A.2 State Predicates Concerning Only a Location in the Scene

location(Loc, X, Y) : the coordinates **(X, Y)** are assigned to the abstract location identifier **Loc**.

is_free(S_seg) : the segment **S_seg** is not occupied by any agent.

A.3 State Predicates Concerning the Actor and its Location

path_towards_waiting_line(Agent, WLine, WLPath) : from the position of the agent **Agent**, **WLine** is the next *waiting_line* segment. Then, **WLPath** holds the list of segments constituing the path from the agent to **WLine**. The last segment of **WLPath** is labeled as a *waiting_line* segment.

path_towards_specific_seg(Agent, Seg, Path) : from the current position of the agent *Agent*, this predicate constructs the straightest path towards the specific segment *Seg*, which is stored in *Path*

nearest_segment_of_type(Agent, Type, Seg, Dmin) : this predicate computes the nearest segment of type *Type* and stores it into *Seg*. The predicate also stores the distance from the current position of *Agent* to *Seg* in *Dmin*.

nearest_disconnected_segment_same_type(Agent, Seg) : this predicate computes the nearest segment that shares the type with the segment where *Agent* is currently located, but has different segments separating them. The predicate stores the segment into *Seg*.

nearest_crosswalk(Agent, Dist) : this predicate computes the distance between the current location of *Agent* and the nearest segment of type *crosswalk*, and stores it into *Dist*.

is_walking_through(Agent, Path) : the agent **Agent** is positioned on a segment which belongs to the path **Path**.

on_waiting_line(Agent, WLine) : the agent **Agent** is positioned on the segment **WLine**, which is labeled as a *waiting_line*.

on_sideway(Agent, SSeg) : the agent **Agent** is positioned on the segment **SSeg**, which is labeled as a *sidewalk*.

on_crosswalk(Agent, CSeg) : the agent **Agent** is positioned on the segment **CSeg**, which is labeled as a *crosswalk*.

on_road(Agent, RSeg) : the agent **Agent** is positioned on the segment **RSeg**, which is labeled as a *road*.

no_obstacle_ahead(Agent, S_seg) : the agent **Agent** can go towards the segment **S_seg**, because such a segment is not being presently occupied by any agent.

obstacle_ahead(Agent, Patiens, S_seg) : the agent **Agent** can not go towards the segment **S_seg**, because the agent **Patiens** presently resides on that segment.

A.4 Reaction Predicates

follow_path(Agent, S_seg, Path) : this predicate will modify the direction of the agent **Agent** in order to go from **S_seg** towards its next segment in the segment list of **Path**.

turn(Agent, Value) : this predicate will modify the direction of the agent **Agent** depending on the orientation value **Value**. Possible conceptual descriptions of **Value** are *left*, and *right*.

accelerate(Agent, Value) : this predicate will modify the velocity of the agent **Agent** depending on the acceleration value **Value**. Possible conceptual descriptions of **Value** are *high*, *normal*, and *small*.

accelerate_to(Agent, Value) : this predicate will modify the velocity of the agent **Agent** in order to reach progressively the qualitative value of **Value**. Possible conceptual descriptions of **Value** are *highest*, *very_high*, *high*, *normal*, *small*, and *zero*.

go_on_performing(Agent, ALabel) : this predicate will increment the posture parameter of the agent **Agent** in order to adopt the next posture described by the **ALabel** *p–action* (that means, to increment the arc length parameter of the *p–action* corresponding to **ALabel**).

change_performing(Agent, ANewLabel) : this predicate will increment the posture parameter of the agent **Agent** in order to change from the current action to the **ANewLabel** action. That means, to sample those points belonging to a spline which interpolates between the current posture parameter and the first posture of the **ANewLabel** *p–action*.

wait(Agent, Duration) : this predicate keeps the agent **Agent** performing the *aStand* action during the period determined by the qualitative value **Duration**. Possible conceptual descriptions of **Duration** are *long*, *normal*, and *briefly*.

keep_on_performing(Agent, Duration) : this predicate keeps the agent **Agent** performing cyclic actions (like *aWalk* and *aRun*) during the period determined by the qualitative value **Duration**. Possible conceptual descriptions of **Duration** are *long*, *normal*, and *briefly*.

freeze(Agent, Duration) : this predicate maintains the same posture parameter for the agent **Agent** during the period determined by the qualitative value **Duration**. Possible conceptual descriptions of **Duration** are *long*, *normal*, and *briefly*.

A.5 Low–level predicates extracted from tracking

The following predicates are used to distinguish the different agents considered in the HERMES Outdoor scenario. They are continuously instantiated while the target keeps tracked and are estimated by analyzing the bounding box of the tracked targets.

is_pedestrian(Agent) : this predicate states that the agent *Agent* is a pedestrian.

is_vehicle(Agent) : this predicate states that the agent *Agent* is a vehicle.

is_object(Agent) : this predicate states that the agent *Agent* is a static object that might be previously carried by a pedestrian.

The next two predicates are provided by our tracking system [88] and indicate the creation or split of a group of agents due to occlusions.

is_splitting2(Owner, Agent, Owner) : This predicate states that a new agent *Agent* has appeared due to the split of the agent *Owner* into two objects.

is_grouping2(Agent, Agent2, Agent)) : This predicate states that the agents *Agent* and *Agent2* are grouped into a single agent *Agent*.

A.6 High–level predicates for behavior understanding

The following predicates are inferred by means of a combination of previously listed predicates and constitute a higher level of abstraction for the recognition of complex behaviors in the HERMES Outdoor scenario:

near_other_pedestrian(Agent1, Agent2) : This predicate states that the agent *Agent1* is near a pedestrian *Agent2* at a given time step. Therefore, this implies that the predicates is_pedestrian(Agent2) and have_distance(Agent1,Agent2, small) must be also satisfied.

belongs_to(Agent, Owner) : This predicate states that the agent *Agent* is property of *Owner*. The property is stated by means of the low–level predicate is_splitting(...), which indicates that *Agent* appeared as a split of *Owner*.

leaves_object(Agent, Object) : This predicate states that the agent *Agent* has left an object in the scene. This is a result of combining the predicates is_object(Object) and the a past instantiation of belongs_to(Agent, Object).

object_is_alone(Object) : This predicate states that the agent *Object* is an object (and therefore the predicate is_object(Object) is satisfied) and does not have any other agent near it.

agent_near_object(Agent, Object) : This predicate states that the agent *Object* is an object (and therefore the predicate is_object(Object) is satisfied) and the agent *Agent* is near it.

no_other_agents_near_object(Agent, Object) : This predicate states that the agent *Object* is an object (and therefore the predicate is_object(Object) is satisfied) and the agent *Agent* is the only agent near it.

agent_takes_object(Agent, Object) : This predicate states that the agent *Agent* takes the object *Object*. It uses the low–level predicates is_object(Object) to indicate that *Object* is an object and is_grouping(...) to indicate that *Agent* is a group that contains *Object*.

carries_object(Agent, Object) : This predicate states that the agent *Agent* is carrying *Object*. This implies a past satisfaction of the predicate agent_takes_object.

going_in_similar_direction(Agent1, Agent2) : This predicate states that *Agent1* and *Agent2* are moving in a similar direction. It uses the orientation of both agent to compute a fuzzy degree of similarity.

is_chasing(Ped, Thief) : This predicate states that *Ped* is chasing *Thief*. It is based on the fact that both *Ped* and *Thief* are moving fast, following the same direction and *Ped* is behind *Thief* given that direction.

danger_of_runover(Ped, Vehicle) : This predicate states that there the agent *Vehicle* may runover *Ped*. It is instantiated when *Ped* is a pedestrian, *Vehicle* is a vehicle and both are moving along the crosswalk in the same time step.

Subsequently, the following list of predicates describe a set of high–level situations that describe football player behavior in the football sequence of VS-PETS 2003. Note that the ball is not included as a parameter in the predicates, since we assume to have only one agent in the scene represented the ball, and is indicated with the predicate *is_ball(Agent)*.

player_with_ball(Agent) : This predicate states that the player *Agent* has the ball controlled at the current time step. This entails a past satisfaction of the low–level predicate *is_grouping* between *Agent* and the ball.

is_from_team(Agent,Team : This predicate states that the player *Agent* belongs to the team *Team*, which in the case of football will be either *team1* or *team2*. Teams are distinguished based on the color model of their clothing.

player_in_defensive_zone(Agent) : This predicate states that the player *Agent* is located in its deffensive zone, according to the team that *Agent* belongs to.

player_in_attack_zone(Agent) : This predicate states that the player *Agent* is located in its attacking zone, according to the team that *Agent* belongs to.

different_team(Agent, Agent2) : This predicate states that the players *Agent* and *Agent2* belong to opposite teams.

same_team(Agent, Agent2) : This predicate states that the players *Agent* and *Agent2* belong to the same team.

ball_is_alone : This predicate states that there are no players near enough to the ball in the current time step.

ball_in_goal : This predicate states that the ball is located in a segment of type *goal*.

A.7 Defining the Conceptual Scene Model

A.7.1 (Factual) Predicates

point($<x>, <y>, <p>$) : the individual $<p>$ is a point in the plane with coordinates $< x >$ and $< y >$, which are floating point numbers.

line($<p1>, <p2>, <l>$) : the individual $<l>$ constitutes a line between two points, given as the two individuals $<p1>$ and $<p2>$.

segment_of_lane($<l1>, <l2>, <seg>$) : the two lines given by the individuals $<l1>$ and $<l2>$ define a segment which is given by the individual $<seg>$.

sideway_seg($<seg>$) : the individual $<seg>$ constitutes a sideway segment.

waiting_line($<seg>$) : the individual $<seg>$ constitutes a *waiting_line* segment.

road_seg($<seg>$) : the individual $<seg>$ is labeled to be a *road* segment.

crosswalk_seg($<seg>$) : the individual $<seg>$ constitutes an *crosswalk* segment.

A.7.2 (Factual) *Precomputed* Predicates

The following predicates define facts which can be precomputed from the defined conceptual model. The precomputation is done only for better performance during the actual SGT-traversal.

lseg_beside$(<seg1>, <seg2>)$: the segments $<seg1>$ and $<seg2>$ lie next to each other (such that an agent could change from one segment to the other).

lseg_in_front$(<seg1>, <seg2>)$: the segment $<seg1>$ lies in front of the segment $<seg2>$.

lseg_behind$(<seg1>, <seg2>)$: the segment $<seg1>$ lies behind the segment $<seg2>$.

Bibliography

[1] Wordnet lexical database. http://wordnet.princeton.edu/.

[2] J. Aguilera, H. Wildenauer, M. Kampel, M. Borg, D. Thirde, and J. Ferryman. Evaluation of motion segmentation quality for aircraft activity surveillance. In *ICCCN '05*, pages 293–300, Washington, DC, USA, 2005.

[3] M. Albanese, R. Chellappa, V. Moscato, A. Picariello, V.S. Subrahmanian, P. Turaga, and O. Udrea. A constrained probabilistic petri net framework for human activity detection in video. *IEEE Transactions on Multimedia*, 10(6):982–996, October 2008.

[4] A. Amato, M. Mozerov, I. Huerta, J. Gonzàlez, and J.J. Villanueva. Background subtraction technique based on chromaticity and intensity patterns. In *19th International Conference on Pattern Recognition (ICPR'08)*, Tampa, Florida, USA, 2008.

[5] N. Anjum and A. Cavallaro. Single camera calibration for trajectory-based behavior analysis. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2007*, pages 147–152, London,UK, September 2007.

[6] G. Antonini and J.P. Thiran. Counting pedestrians in video sequences using trajectory clustering. *IEEE Trans. on Circuits Syst. Video Technol.*, 8(16):1009–1020, 2006.

[7] M. Arens and H.-H. Nagel. Behavioral knowledge representation for the understanding and creation of video sequences. In *Proceedings of the 26th German Conference on Artificial Intelligence (KI-2003)*, pages 149–163. LNAI, Springler-Verlag: Berlin, Heidelberg, New York/NY, September 2003.

[8] N.I. Badler, M.S. Palmer, and R. Bindiganavale. Animation control for real-time virtual humans. *Communications of the ACM*, 42:64–73, 1999.

[9] P. Baiget, E. Sommerlade, I. Reid, and J. Gonzàlez. Finding prototypes to estimate trajectory development in outdoor scenarios. In *Proceedings of the 1st THEMIS Workshop*, pages 27–34, Leeds, UK, 2008.

[10] P. Baiget, J. Soto, X. Roca, and J. Gonzàlez. Automatic generation of computer animated sequences based on human behavior modeling. In *10th International*

Conference in Computer Graphics and Artificial Intelligence, Athens, Greece, May, 2007.

[11] S. Balcisoy, M. Kallmann, R. Torre, P. Fua, and D. Thalmann. Interaction techniques with virtual humans in mixed environments. In *5th IEEE EMBS International Summer School on Biomedical Imaging, 2002*, Ile de Berder, Bretagne, France, 2002.

[12] S. Balcisoy and D. Thalmann. Interaction between real and virtual humans in augmented reality. *Computer Animation '97*, pages 31–38, Jun 1997.

[13] A. Basharat, A. Gritai, and M. Shah. Learning object motion patterns for anomaly detection and improved object detection. In *IEEE Conference on CVPR*, pages 1–8, Anchorage, AK, USA, 2008.

[14] F. Bashir and F. Porikli. Performance evaluation of object detection and tracking systems. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, New York, USA, June, 2006.

[15] B. Benfold and I. Reid. Colour invariant head pose classification in low resolution video. In *19th British Machine Vision Conference*, Leeds, UK, September 2008.

[16] B. M. Blumberg and T.A. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of SIGGRAPH '95*, pages 47–54, New York, NY, USA, 1995. ACM.

[17] A. Bobick and J.W. Davis. The recognition of human movement using temporal templates. *IEEE Trans. on PAMI*, 23(3):257–267, Mar 2001.

[18] A. Bobick and A. D. Wilson. A state-based technique to the representation and recognition of gesture. *IEEE Trans. on PAMI*, 19(12):231–236, 1997.

[19] A. Borzin, E. Rivlin, and M. Rudzsky. Representation and recognition of agent interactions using marking analysis in generalized stochastic petri nets. In *International Workshop on Content-Based Multimedia Indexing '07.*, pages 33–39, Bordeaux, France, June 2007.

[20] M. Brand. Understanding manipulation in video. In *Int. Conf. on Automatic Face and Gesture Recognition*, pages 94–99, 1996.

[21] M. Brand and V. Kettnaker. Discovery and segmentation of activities in video. *IEEE Trans. on PAMI*, 22(8):844–851, 2000.

[22] F. Bremond, M. Thonnat, and M. Zuniga. Video understanding framework for automatic behavior recognition. *Behavior Research Methods*, 3(38):416–426, 2006.

[23] H. Buxton. Learning and understanding dynamic scene activity: A review. *Image and Vision Computing*, 21(1):125–136, 2002.

[24] S. Calderara, R. Cucchiara, and A. Prati. Detection of abnormal behaviors using a mixture of von mises distributions. In *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pages 147–152, September 2007.

[25] P. Chaudhuri, G. Papagiannakis, and N. Magnenat-Thalmann. Self adaptive animation based on user perspective. *The Visual Computer*, pages 525–533, July 2008.

[26] J. Cheng and M.F. Moura. Capture and represention of human walking in live video sequences. *IEEE Transactions on Multimedia*, 1(2):144–156, 1999.

[27] A. Colmerauer. Introduction to prolog III. *Communications of the ACM*, 33(7):68–90, 1990.

[28] T. Conde and D. Thalmann. Autonomous virtual agents learning a cognitive model and evolving. In *International Workshop on Intelligent Virtual Agents 2005*, pages 88–98, London, UK, 2005. Springer-Verlag.

[29] K. Daniels and C. Giraud-Carrier. Learning the threshold in hierarchical agglomerative clustering. *5th International Conference on Machine Learning and Applications 2006*, pages 270–278, December 2006.

[30] C. de Boor. *A practical guide to splines*. Springer-Verlag, New York, USA, 1978.

[31] X. Desurmont, R. Sebbe, F. Martin, C. Machy, and J.-F. Delaigle. Performance evaluation of frequent events detection systems. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, New York, USA, June, 2006.

[32] F. Dornaika and F. Davoine. Simultaneous facial action tracking and expression recognition in the presence of head motion. *Int. Journal on Computer Vision*, 76(3):257–281, 2008.

[33] M. Douze and V. Charvillat. Real-time generation of augmented video sequences by background tracking. *Computer Animation and Virtual Worlds*, 17(5):537–550, 2006.

[34] T.V. Duong, H.B. Bui, D.Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *IEEE CVPR '05*, volume 1, pages 838–845, San Diego, USA, June, 2005.

[35] A. Egges, G. Papagiannakis, and N. Magnenat-Thalmann. Presence and interaction in mixed reality environments. *The Visual Computer*, 23(5):317–333, May 2007.

[36] C. Fernández, P. Baiget, and J. Gonzàlez. Mixed–initiative authoring for augmented scene modeling. In *22nd Annual Conference on Computer Animation and Social Agents (CASA '09)*, Amsterdam, The Netherlands, June, 2009.

[37] C. Fernández, P. Baiget, X. Roca, and J. Gonzàlez. Natural language descriptions of human behavior from video sequences. In *In 30th Annual German Conference on Artificial Intelligence (KI-2007)*, Osnabrück, Germany, 2007.

[38] C. Fernández, P. Baiget, X. Roca, and J. Gonzàlez. Interpretation of complex situations in a cognitive surveillance framework. *Signal Processing. Special issue on Semantic Analysis for Interactive Multimedia Services*, page To appear, 2008.

[39] J. H. Fernyhough, A. G. Cohn, and D. Hogg. Generation of semantic regions from image sequences. In *ECCV '96*, pages 475–484, London, UK, 1996. Springer-Verlag.

[40] A. Galata, N. Johnson, and D. Hogg. Learning variable-length markov models of behavior. *Computer Vision and Image Understanding*, 81(3):398–413, 2001.

[41] E. Gelenbe, K. Hussain, and V. Kaptan. Simulating autonomous agents in augmented reality. *Journal of Systems and Software*, 74(3):255–268, 2005.

[42] N. Ghanem, D. DeMenthon, D. Doermann, and L. Davis. Representation and recognition of events in surveillance video using petri nets. In *CVPR Workshop '04*, page 112, Washington, DC, USA, 2004. IEEE Computer Society.

[43] A. S. Glassner, J. Arvo, R. L. Cook, E. Haines, P. Hanrahan, P. Heckbert, and D. B. Kirk. *An Introduction to Ray Tracing*. Academic Press, London, 1989.

[44] J. Gonzàlez. *Human Sequence Evaluation: The Key-frame Approach*. PhD thesis, Universitat Autònoma de Barcelona, Spain, 2004.

[45] J. Gonzàlez, F. X. Roca, and J. J. Villanueva. Hermes: A research project on human sequence evaluation. In *Computational Vision and Medical Image Processing (VipIMAGE'2007)*, Porto, Portugal, 2007.

[46] J. Gonzàlez, X. Varona, X. Roca, and J. J. Villanueva. Automatic keyframing of human actions for computer animation. In *Proceedings of the 1st Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2003)*, pages 287–296, Port d'Andratx, Mallorca, Spain, 2003.

[47] J. Gonzàlez, D. Rowe, J. Varona, and X. Roca. Understanding dynamic scenes based on human sequence evaluation. *Image and Vision Computing*, DOI: 10.1016/j.imavis.2008.02.004:–, 2008.

[48] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE Trans. on PAMI*, 29(12):2247–2253, December 2007.

[49] M. Haag and H.-H. Nagel. Incremental recognition of traffic situations from video image sequences. *Image and Vision Computing*, 18(2):137–153, 2000.

[50] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, USA, 2003.

[51] L.J. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identification and analysis of coexpressed genes. *Genome Research*, 11:1106–1115, 1999.

[52] W. Hu, X. Xiao, Z. Fu, and D. Xie. A system for learning statistical motion patterns. *IEEE Trans. on PAMI*, 28(9):1450–1464, 2006.

[53] C. E. Hughes, C. B. Stapleton, D. E. Hughes, and E. M. Smith. Mixed reality in education, entertainment, and training. *IEEE Computer Graphics and Applications*, 25(6):24–30, 2005.

[54] EU Project IST-045547. Vidi-video: Interactive semantic video search with a large thesaurus of machine-learned audio-visual concepts. http://www.vidivideo.it.

[55] Dieter Schmalstieg István Barakonyi. Augmented reality agents for user interface adaptation. *Computer Animation and Virtual Worlds*, 19(1):23–35, 2008.

[56] Y.A. Ivanov and A.F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. on PAMI*, 22(8):852–872, 2000.

[57] F. Jiang, Y. Wu, and A. K.Katsaggelos. A hierarchical bayesian network for event recognition of human actions and interactions. *IEEE Transactions on Image Processing*, 18(4):907–913, April 2009.

[58] N. Johnson and D. C. Hogg. Learning the distribution of object trajectories for event recognition. In *British Machine Vision Conference '95*, pages 583–592, Surrey, UK, 1995.

[59] Z. Kasap and N. Magnenat-Thalmann. Intelligent virtual humans with autonomy and personality: State of the art. *Intelligent Decision Technologies*, 1(1-2):3–15, 2007.

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

[61] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[62] A. Kojima, T. Tamura, and K. Fukunaga. Natural language description of human activities from video images based on concept hierarchy of actions. *International Journal of Computer Vision*, 50(2):171–184, 2002.

[63] I. Kotsia and I. Pitas. Facial expression recognition in image sequences using geometric deformation features and support vector machines. *IEEE Transactions on Image Processing*, 16(1):172–187, Jan. 2007.

[64] N. Magnenat and D. Thalmann. Animating virtual actors in real environments. *Multimedia Systems*, 5(2):113–125, 1997.

[65] N. Magnenat-Thalmann and D. Thalmann. Virtual humans: thirty years of research, what next? *The Visual Computer*, 21(12):997–1015, December 2005.

[66] R. Maiocchi. 3-d character animation using motion capture. In Prentice-Hall, editor, *Interactive Computer Animation*, pages 10–39, London, UK, 1996.

[67] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Trans. on Systems Man and Cybernetics–Part B*, 35(3):397–408, June 2005.

[68] D. Minnen, I. Essa, and T. Starner. Expectation grammars: Leveraging high-level expectations for activity recognition. In *IEEE CVPR '03*, volume 2, pages 626–632, Wisconsin, USA, June, 2003.

[69] T.B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2):90–126, 2006.

[70] A. Naftel and S. Khalid. Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. *IEEE Trans. on Multimedia Systems*, 3(12):45–52, 2006.

[71] H.-H. Nagel. From image sequences towards conceptual descriptions. *Image and Vision Computing*, 6(2):59–74, 1988.

[72] H.-H. Nagel. Steps toward a cognitive vision system. *AI Magazine, Special Cognitive Vision*, 25(2):31–50, 2004.

[73] A.T. Nghiem, F. Bremond, M. Thonnat, and R. Ma. A new evaluation approach for video processing algorithms. In *IEEE Workshop on Motion and Video Computing WMVC*, pages 15–15, 2007.

[74] Nam T. Nguyen, Dinh Q. Phung, Svetha Venkatesh, and Hung Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov models. In *IEEE CVPR '05*, pages 955–960, Washington, DC, USA, 2005. IEEE Computer Society.

[75] N.T. Nguyen, H.H. Bui, S. Venkatesh, and G. West. Recognising and monitoring high-level behaviours in complex spatial environments. In *IEEE CVPR '03*, volume 2, pages 620–625, Wisconsin, USA, 2003.

[76] N.M. Oliver, B. Rosario, and A.P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Trans. on PAMI*, 22(8):831–843, 2000.

[77] Joint IEEE International Workshop on Visual Surveillance, Performance Evaluation of Tracking, and Surveillance. Vs-pets 2003. http://www.cvg.rdg.ac.uk/VSPETS/.

[78] G. Papagiannakis, S. Schertenleib, B. O'Kennedy, M. Arevalo-Poizat, N. Magnenat-Thalmann, A. Stoddart, D. D. Thalmann, S. Geneva, and S. Lausanne. Mixing virtual and real scenes in the site of ancient Pompeii. *Computer Animation and Virtual Worlds*, 16(1):11–24, 2005.

[79] G. Papagiannakis, G. Singh, and N. Magnenat-Thalmann. A survey of mobile and wireless technologies for augmented reality systems. *Computer Animation and Virtual Worlds*, 19(1):3–22, 2008.

[80] S. Park and J.K. Aggarwal. A hierarchical bayesian network for event recognition of human actions and interactions. *Multimedia Systems*, 10(2):164–179, August 2004.

[81] L. Patino, H. Benhadda, E. Corvee, F. Bremond, and Monique Thonnat. Extraction of activity patterns on large video recordings. *IET Computer Vision*, 2(2):108–128, June 2008.

[82] K. Perlin and A. Goldberg. Improv: a system for scripting interactive actors in virtual worlds. In *Proceedings of SIGGRAPH '96*, pages 205–216, New York, NY, USA, 1996. ACM.

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

[84] W. Piekarski and B. Thomas. Arquake: The outdoor augmented reality gaming system. *Communications of the ACM*, 45(1):36–38, 2002.

[85] W. Piekarski and B. H. Thomas. An object-oriented software architecture for 3D mixed reality applications. In *2nd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR) '03*, page 247, Washington, DC, USA, 2003. IEEE Computer Society.

[86] E. Reiter and R. Dale. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge/UK, 2000.

[87] D. Roth, E. Koller-Meier, D. Rowe, T.B. Moeslund, and L. Van Gool. Event-based tracking evaluation metric. In *IEEE Workshop on Motion and Video Computing (WMVC)*, 2008.

[88] D. Rowe. *Towards Robuts Multiple–Target Tracking in Unconstrained Human–Populated Environments*. PhD thesis, Universitat Autònoma de Barcelona, Spain, 2008.

[89] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2003.

[90] K. Schäfer. Fuzzy spatio-temporal logic programming. In C. Brzoska, editor, *Proceedings of 7th Workshop in Temporal and Non-Classical Logics (IJCAI) '97*, pages 23–28, Nagoya, Japan, 1997.

[91] Id Software. Quake III arena. http://www.idsoftware.com/games/quake/quake3-arena/.

[92] Massive Software. Simulating life. http://www.massivesoftware.com.

[93] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *IEEE CVPR '99*, 2:–252 Vol. 2, 1999.

[94] W. Tambellini T. Conde and D. Thalmann. Behavioral animation of autonomous virtual agents helped by reinforcement. In *International Workshop on Intelligent Virtual Agents 2003. Proceedings (LNAI)*, volume 2792, pages 175–180, Kloster Irsee, Germany, 2003.

[95] D. Shreiner M. Woo T. Davis, J. Neider. *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Addison-Wesley, 2005.

[96] R. Torre, P. Fua, S. Balcisoy, M. Ponder, and D. Thalmann. Interaction between real and virtual humans: Playing checkers. In *Eurographics Workshop On Virtual Environments '00*, 2000.

[97] C. Town. Ontology-driven bayesian networks for dynamic scene understanding. In *IEEE CVPR '04*, volume 2, pages 626–632, Washington D.C., USA, 2004.

[98] CAVIAR: Context Aware Vision using Image-based Active Recognition. Ec's information society technology. http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

[99] N. Vaswani, A.R. Chowdbury, and R. Chellappa. Activity recognition using the dynamics of the configuration of interacting objects. In *IEEE CVPR '03*, volume 2, pages 633–640, Wisconsin, USA, 2003.

[100] H. Veeraraghavan, N. Papanikolopoulos, and P. Schrater. Learning dynamic event descriptions in image sequences. In *IEEE CVPR '07*, pages 1–6, June 2007.

[101] T. Wada and T. Matsuyama. Multiobject behavior recognition by event driven selective attention method. *IEEE Trans. on PAMI*, 22(8):873–887, 2000.

[102] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2):249–257, 2006.

[103] B. Wu and R. Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. In *IEEE CVPR '06*, pages 951–958, New York, USA, 2006.

[104] X.Wang, K. Tieu, and E. Grimson. Learning semantic scene models by trajectory analysis. In *Proceedings of the ECCV '06*, Graz, Austria, 2006.

[105] Y. Yacoob and M. Black. Parameterized modeling and recognition of activities. In *IEEE ICCV '98*, pages 120–127, Bombay, India, 1998.

[106] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, 2006.

[107] D.P. Young and J.M. Ferryman. Pets metrics: On-line performance evaluation service. *2nd Joint IEEE International Workshop on VS-PETS '05*, pages 317–324, 15-16 Oct. 2005.

[108] G. Zhang, X. Qin, X. An, W. Chen, and H. Bao. As-consistent-as-possible compositing of virtual objects and video sequences. *Computer Animation and Virtual Worlds*, 17(3-4):305–314, 2006.

[109] N. Zouba, F. Bremond, and M. Thonnat. Monitoring activities of daily living (adls) of elderly based on 3D key human postures. In *4th International Cognitive Vision Workshop, ICVW '08*, Santorini, Greece, May 12-15, 2008.

Publications

Journals

- Pau Baiget, Carles Fernández, Xavier Roca, Jordi Gonzàlez. Generation of augmented video sequences combining behavioral animation and multi–object tracking. Computer Animation and Virtual Worlds. In press. 2009.

- Pau Baiget, Carles Fernández, Xavier Roca, Jordi Gonzàlez. Dynamic Scene Conceptualization by Trajectory Clustering. Computer Vision and Image Understanding. Accepted with changes.

- Carles Fernández, Pau Baiget, Xavier Roca, Jordi Gonzàlez. Interpretation of Complex Situations in a Cognitive Surveillance Framework Signal Processing: Image Communication Journal, Special issue on 'Semantic Analysis for Interactive Multimedia Services'. Elsevier, 2008.

Conferences

- Carles Fernández, Pau Baiget, Jordi Gonzàlez Mixed-Initiative Authoring for Augmented Scene Modeling 22nd Annual Conference on Computer Animation and Social Agents (CASA 2009) Amsterdam, The Netherlands, June 2009

- Pau Baiget, Carles Fernández, Xavier Roca, Jordi Gonzàlez Observing Human Behavior in Image Sequences: the Video-Hermeneutics Challenge 3rd CVC Workshop: Progress of Research and Development (CVCRD'2008) Cerdanyola del Vallés, Barcelona, Spain, October 2008

- Carles Fernández, Pau Baiget, F. Xavier Roca, Jordi Gonzàlez Three Dialogue-based Challenges for Cognitive Vision Surveillance 3rd CVC Workshop: Progress of Research and Development (CVCRD'2008) Cerdanyola del Vallés, Barcelona, Spain, October 2008

- Carles Fernández, Pau Baiget, F. Xavier Roca, Jordi Gonzàlez Cognitive-Guided Semantic Exploitation in Video Surveillance Interfaces Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS Workshop) in conjunction with British Machine Vision Conference (BMVC'2008). **Best paper award**. Leeds, UK, September, 2008

- Pau Baiget, Eric Sommerlade, Ian Reid, Jordi Gonzàlez Finding Prototypes to Estimate Trajectory Development in Outdoor Scenarios Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS Workshop) in conjunction with British Machine Vision Conference (BMVC'2008) Leeds, UK, September, 2008

- Pau Baiget, Carles Fernández, Xavier Roca, Jordi Gonzàlez Autonomous Virtual Agents for Performance Evaluation of Tracking Algorithms 5th International Conference on Articulated Motion and Deformable Objects (AMDO'2008) **Best paper award** Andratx, Mallorca, Spain July, 2008

- Pau Baiget, Carles Fernández, Xavier Roca, Jordi Gonzàlez Automatic Learning of Conceptual Knowledge for the Interpretation of Human Behavior in Video Sequences 3rd Iberian Conference on Pattern Recognition and Image Analysis (Ibpria 2007) Girona, Spain, June 2007

- Pau Baiget, Joan Soto, Xavier Roca, Jordi Gonzàlez Automatic Generation of Computer Animated Sequences Based on Human Behavior Modeling 10th 3IA International Conference in Computer Graphics and Artificial Intelligence (3IA 2007) Athens, Greece, May 2007.

- Carles Fernández, Pau Baiget, Xavier Roca, Jordi Gonzàlez Natural Language Descriptions of Human Behavior from Video Sequences 30th Annual German Conference on Artificial Intelligence (KI-2007) Osnabrück, Germany, October, 2007

- Pau Baiget, Carles Fernández, Aariel Amato, F. Xavier Roca, Jordi Gonzàlez Constructing a Path Database for Scene Categorization 2nd CVC Workshop: Progress of Research and Development (CVCRD'2007) Cerdanyola del Vallés, Barcelona, Spain, October 2007

- Carles Fernández, Pau Baiget, F. Xavier Roca, Jordi Gonzàlez High-level Integration for Cognitive Vision Surveillance 2nd CVC Workshop: Progress of Research and Development (CVCRD'2007) Cerdanyola del Vallés, Barcelona, Spain, October 2007

- Carles Fernández, Pau Baiget, Xavier Roca, Jordi Gonzàlez Semantic Annotation of Complex Human Scenes for Multimedia Surveillance Tenth International Conference on Advances in AI (AI*IA 2007) Roma, Italy, September 2007

- Carles Fernández, Pau Baiget, Mikhail Mozerov, Jordi Gonzàlez Spanish Text Generation for Human Evaluation using FMTHL and DRS First CVC Workshop on the Progress of Research and Development (CVCRD 2006) Cerdanyola del Vallés, Barcelona, Spain, October 2006

- Javier Orozco, Pau Baiget, Xavier Roca, Jordi Gonzàlez Eyelids and Face Tracking in Real-Time The Sixth IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP 2006) Palma de Mallorca, Spain, August 2006

- Pau Baiget, Carles Fernández, Xavier Roca, Jordi Gonzàlez Interpretation of Human Motion in Image Sequences Using Situation Graph Trees First CVC Workshop on the Progress of Research and Development (CVCRD 2006) Cerdanyola del Vallés, Barcelona, Spain, October 2006

Technical Reports and Book Chapters

- Pau Baiget Interpretation of Human Behavior in Image Sequences CVC Technical Report #102, CVC (UAB) February 2007
- Carles Fernández, Pau Baiget, Xavier Roca, Jordi Gonzàlez Exploiting Natural Language Generation in Scene Interpretation Book chapter in Human-Centric Interfaces for Ambient Intelligence Elsevier Science and Technology Book Group, October 2009