Universitat
Autònoma
de Barcelona

Human Sequence Evaluation: the *Key-frame* Approach

A dissertation submitted by **Jordi Gonzàlez i Sabaté** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor en Informàtica**.

Bellaterra, May 25th, 2004

Director: **Dr. F. Xavier Roca i Marvà**
Universitat Autònoma de Barcelona
Dept. Informàtica & Centre de Visió per Computador
Co-director: **Dr. Javier Varona Gómez**
Universitat de les Illes Balears
Dept. Matemàtiques i Informàtica

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 84–933652–2–X

Printed by Ediciones Gráficas Rey, S.L.

Atsedenik ez drogarik gabe!

Observa con qué facilidad escribes
sobre pájaros. Pero ¿ cuántos has palpado
amorosamente con el calor de tus manos ?
¿ Cuántos han latido realmente
bajo la presión de tus dedos ?
¿ Acaso los has descrito
sin olvidar detalle como quien
conoce bien un cuerpo amado ?
¿ Los has liberado acaso
del peso de tus palabras ?

Juan Calzadilla

Agraïments

Diuen que l'home és la suma de l'ésser viu i del seu context. Per ésser viu em refereixo a qualsevol autor d'una Tesi, que arriba fins aquesta secció després d'anys de resoldre, dia a dia, petits reptes científics. Per altra banda, el seu context està format per aquelles persones que un guanya després d'anys de resoldre, dia a dia, petits reptes d'amistat. És per a tots aquells que m'han envoltat fins aquí a qui dedico aquest treball, sense ordre de preferència.

Molt especialment, als meus guies de viatge i amics, Xavi Roca i Javi Varona. He comprovat que tots dos es complementen a la perfecció. A Xavi, per haver cuidat de mi i haver-me donat la tranquil·litat que necessitava per a poder desenvolupar les meves paranòies. Per tenir sempre un moment, durant la seva carrera quotidiana, d'aturar-se per prendre un gin-tonic verbal. A Javi, el meu gurú espiritual i científic d'aquests anys: pels seus consells, idees, rectificacions i paciència. Per haver-me fet descobrir la il·lusió per la recerca, barrejada amb bon humor. Aquest optimisme, sense dubte, ha estat possible gràcies a la presència de l'Aina al seu costat.

Per descomptat, al Prof. J.J. Villanueva, qui, com a director del Centre de Visió per Computador, em va oferir l'oportunitat d'investigar dins d'un entorn científic privilegiat, així com per la seva ajuda personal en tot moment.

Also, I would like to express my most sincerely gratitude to Prof. H.-H. Nagel. He is actually the source of inspiration, and strength. His dedication to science still astonishes me. Likewise, I appreciate his patience with respect to my technical reports, which were converted to *red books* after his remarks. This Thesis is due to him. I would like to thank his incredible team: Frau Dietrich, Michael, Artur, Ralf and Markus. They really adopted me in Karlsruhe.

Als inquilins del CVC, una comunitat científica de veïns que funciona com una gran família quan es convoquen barbacoes regades amb (molt) Marqués de Cáceres. Som ja tanta gent que no caben en aquesta secció. Als companys de despatx d'aquests anys, molt especialment a l'Anna, Juanma, Xevi, i David. Compartir amb ells aquests anys m'ha permès ser testimoni de la seva superació personal, així com comprovar que una Tesi és el resultat d'un treball quotidià i rigorós. I als altres companys de recerca: Ramon Felip, Oriol, Robert, Botey, Dani, David, Fernando, Débora, ... per a què aviat arribin a la seva secció d'agraïments. Gràcies també per l'ajuda que m'han proporcionat els qui aconsegueixen que aquest Centre funcioni cada dia:

Mari Carmen, Montse, Raquel, Pedro, Masu, Pilar, Agustí, Maria José, Núria, ... sempre amb predisposició per ajudar. Els nivells de paciència que han de demostrar són exemplaritzants. Als altres integrants d'aquest edifici: Sílvia, Mireia, Miguel, Antonio, Juan Ramón, Eva, ... per estar sempre disposats a muntar dinars i sopars de germanor. També als Doctors de la casa, els quals moltes vegades m'han solucionat dubtes acadèmics, d'investigació i existencials: Maria, Cristina, Ramon, Felipe, Joan, Josep, Jordi, XavierB, Enric, Antonio, Javier, Ricardo, Petia, Gemma, Ernest, ... També vull destacar les ganes contagioses amb les que els nous doctorands han entrat en aquesta casa, com Ignasi, DaniR, i tota la resta que ara comencen l'aventura d'una Tesi.

Als meus referents laborals: Paco, Albert, Ernest, Pepín, Mòrbius, Nacho, Marco, Alex,.... És tot un plaer veure com van cumplint els seus somnis any rera any. Estan demostrant que hi ha molt per viure fora del CVC. Molt en especial al Paco i l'Albert, que m'han ajudat durant aquests anys a mantenir la seva amistat en la distància, així com a disfrutar dels petits plaers de la vida.

A l'Alaska, i a tots els que allí ens reuníem entre morros, bravas i chocos: Cristina, Nando, David, Ferran, Alex i Sergio. Pel coratge demostrat per sobreviure dins de la jungla capitalista. Molt especialment, als sopars de Nadal tan emotius amb Cinta i la Mon (amb Pere i Quique, respectivament).

A les dones del pis: Angelina, Sofia i Rosa. I a la xiqueta, Núria. Perquè m'han cuidat com a un germà durant aquests últims mesos de Tesi, quan més nerviós estava. Realment em sento orgullós d'haver compartit amb elles tots aquests anys de convivència divertida. No hi ha al món prou xocolata per a compensar-ho.

Al 25 de maig.

A la millor penya del món, Alcatràs: Javier, Nandot, Ricardo, Gersio, Jack, Jordi, Edu, Santi, Pau, Mono, JuanRa, Pepe, David, Pablo, Juan Antonio, Juanito i Vicent. Sense ells, tindria el fetge intacte. Amb ells, la festa i l'amistat tenen els seus millor referents. Disfrutar d'aquests companys de viatge és tota una sort i un repte *freak*.

Als meus pares, Juan Antonio i Marisa, els que més m'han ajudat en aquest gran repte personal, de tot cor. A ells, els agraïments s'han de repetir cada dia. Per tot el que han i hem passat junts.

Finalment, aquest treball està dedicat especialment a la memòria del meu germà Javier.

Resum

L'anàlisi de seqüències d'imatges on apareixen éssers humans permet desenvolupar múltiples aplicacions, però també comporta moltes dificultats. Aquest àmbit de recerca tan complexe s'anonema *Human Sequence Evaluation* (HSE). Un sistema HSE genèric transforma dades d'imatges en descripcions d'alt nivell, i viceversa. Per a assolir aquesta abstracció, descrivim una arquitectura modular per desenvolupar sistemes HSE, on cada mòdul es correspon amb un pas d'abstracció. Les contribucions de la investigació que es presenta a continuació s'emmarquen dins d'aquesta arquitectura. Per això s'estableix una taxonomia de moviment humà que guiï el disseny de models intermedis que permetin entendre els canvis produïts en una escena. Aquesta taxonomia inclou el concepte d'*acció*, que es defineix com una seqüència predeterminada de postures humanes.

En aquesta Tesi es proposa un nou model d'accions humanes que s'utilitza en aplicacions on es requereix representar el moviment humà. Les dades d'aprenentatge es corresponen amb postures humanes, on cada postura es defineix a partir d'un nou model del cos humà. Utilitzem moltes realitzacions d'una mateixa acció per construir un espai d'accions humanes, anomenat *aSpace*, on cada realització es representa mitjançant una corba paramètrica. Un cop calculada la mitjana de totes les realitzacions apreses, les postures més característiques de l'acció, anomenades *key-frames*, són seleccionades automàticament d'entre totes les postures que hi pertanyen. Els *key-frames* s'utilitzen per a construir el model final d'acció humana, anomenat *p–action*. El *p–action* és una corba que modelitza l'evolució temporal de la postura del cos durant l'execució prototípica d'una acció i s'utilitza per a implementar algorismes de reconeixement i síntesi d'accions humanes, així com per a analitzar execucions particulars d'accions. Així doncs, en primer lloc, describim un procediment de reconeixement d'accions humanes utilitzant els *key-frames* de cada model d'acció. En segon lloc, presentem un mètode per a realitzar la síntesi d'accions humanes. Donada únicament la durada de l'acció a sintetitzar, obtenim un moviment humà suau i realista. Per a això, el model *p–action* es descriu a partir de la longitud d'arc per tal d'assolir independència respecte a la velocitat d'execució. A més a més, la representació d'accions humanes permet modelitzar les postures que es corresponen a les transicions entre accions, sintetitzant així activitats. Per últim, establim un entorn de comparació per a analitzar les diferències entre realitzacions d'una mateixa acció. En concret, utilitzem l'*aSpace* per a establir una caracterització de l'estil de caminar a partir del gènere dels agents.

Per a concloure aquesta Tesi, afrontem la tasca d'incloure el nostre model d'accions humanes dins de l'entorn de treball del HSE. Per a això, utilitzem els *Situation Graph Trees* (SGTs) per modelitzar el coneixement necessari que ens permet representar el comportament humà. Adaptant el nostre model d'acció dins de la metodologia SGT, aconseguim generar descripcions conceptuals sobre el comportament d'un agent a partir de la informació quantitativa que s'obté de seqüències d'imatges. Finalment, exemplifiquem com obtenir una descripció del comportament humà dins d'una escena, així com la creació de comportaments sintètics per a agents virtuals.

Abstract

The analysis of image sequences involving human agents allows to develop multiple applications, but it implies also lots of difficulties. This challenging domain is referred here as *Human Sequence Evaluation* (HSE). A generic HSE system transforms image data into conceptual descriptions, and vice versa. This abstraction process is addressed by describing the HSE framework as a modular scheme, each module concerned to a specific task domain. The contributions of this investigation are discussed within this framework, and a human motion taxonomy is established to reflect the minimal abstraction steps required for HSE. This taxonomy includes the *action* term which denotes a learnt sequence of human postures.

This Thesis proposes a novel human action model used in different applications which require a representation for human movements. Several performances of a given action constitute the training data which is represented as a sequence of human postures. The learning postures are described using a novel human body model, and they are used to build a human action space, called *aSpace*, within which each human performance is represented as a parametric manifold. As each manifold is parameterized by the (normalized) temporal variation of the posture, the mean performance can be computed. Subsequently, the most characteristic postures for such an action, called *key-frames*, are selected automatically from the postures belonging to the mean performance. Key-frames are used to build the human action model, called *p–action*. A *p–action* represents the time evolution of the human body posture during the prototypical performance of a particular action, and is exploited to perform human action recognition and synthesis, and performance analysis. Firstly, we describe a human action recognition procedure by considering the *key-frame* set of each action model. Secondly, an algorithm for human action synthesis is presented. Realistic and smooth human motion is generated given only the temporal duration of the synthesized action. For this purpose, *p–actions* are parameterized by arc-length to achieve invariance to speed. Moreover, our proposed model for human actions is enhanced to represent postures corresponding to action transitions, thus allowing to synthesize human activities. Lastly, a comparison framework is established to analyse the differences between performances of the same action. Specifically, the *aSpace* representation is used to derive a proper characterization of the walking style in terms of the gender of the walker.

To conclude this investigation, we confront the task of embedding our human

action model within the HSE framework. For this purpose, Situation Graph Trees (SGTs) are used to model the knowledge required for human activity and behavior representation. By adapting our action model to the SGT methodology, we derive semantic primitives based on the quantitative information obtained from image sequences, and we also generate synthetic sequences based on the conceptual information embedded in activity and behavior models. We show examples of SGTs which infer the behavior of actors within a scene, and which generate synthetic behavior for virtual human agents.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the late 19th century, advances in photography and film helped to spur a revolution in the study and perception of human motion. For the first time, an individual could slow down time to a speed at which the different phases of a running or jumping motion were beautifully clear. Two early pioneers in this area were Eadweard James Muybridge and Etienne Jules Marey.

In 1878, Muybridge produced the first time-series photographs of a horse in motion. His books of photographs illustrating animal and human locomotion are still available, and are studied by students of art and science today [100]. While the time-series photographs taken by Muybridge were revolutionary, they did not afford precise measurement of motion over time.

Etienne Jules Marey developed a photographic technique to address this problem in the 1880s [29]. Marey's camera created multiple exposures of an actor at precisely timed intervals. Many instants of the motion were captured on the same photographic plate, so motion over time could be measured from a single photograph. Marey also used the combination of images and force measurements to address such questions as how soldiers might march with a heavy pack so as to minimize fatigue. In fact, Marey's scientific studies anticipated a rich field of human motion research related to multiple areas, which continues today.

Not long ago, computer vision research also began to study video sequences. Improvement of computer resources in terms of capacity and speed allowed to develop new algorithms which coped with the required amount of data. As a result, a new topic of research emerged: the analysis of motion in image sequences.

The analysis of image sequences involving human beings is commonly referred as *dealing with humans*. The goal is to develop a suitable human motion description from image data. This is usually addressed to accomplish two different tasks, namely human motion synthesis and recognition. On the one hand, the aim is to synthesize previously stored human motion information by using virtual actors [51, 96]. Examples of applications are synthetic films, virtual reality interfaces, athletic performance

analysis, and training of physically disabled persons. On the other hand, the goal is to generate textual descriptions about a given human body within a scene using recognized human motion patterns [63]. Typical applications are *video surveillance* and security systems, which are currently being used to monitor buildings or parking lots in order to detect intruders. To this end, textual descriptions about moving humans can be generated for further reasoning about observed conducts.

However, dealing with humans implies multiple difficulties. Typically, appearance variability is found in these sequences due to acquisition conditions, clothes, lighting and posture changes. Consequently, there is a large amount of literature involving human motion analysis [141]. Moreover, there is no unified framework which can cope with the particular requirements of both human synthesis and recognition. Human motion models used in computer animation or motion capture systems are required to be specific and accurate. The architecture is bottom-up, i.e. the representation is built from measurements. On the other hand, video surveillance systems require generic and robust human motion models. Typically, motion information is obtained from outdoor scenes, which implies to deal with noisy data and segmentation errors. These systems frequently operate top-down: measurements are obtained with respect to a predefined model, and the values for the model parameters are usually estimated using tracking procedures.

As a consequence, the procedures for human motion analysis are designed specifically considering the final application domain. Instead, we propose an unified framework for human motion analysis which can be successfully applied to confront both recognition and animation.

1.1 Basic Concepts

We first define the basic terms used throughout the Thesis by describing the knowledge required to develop a *video surveillance* application. The term video surveillance is defined in [36] as describing the actions and interactions of different *agents*, usually humans or vehicles, within complex scenes in real-time. This is achieved by means of *event* analysis. We consider an event as a noticeable change-of-interest in the scene, together with the conditions in which these changes are observed. Actually, we assume that motion analysis is mainly performed based on change detection[1].

Due to the complex task to be achieved by video surveillance systems, these are subdivided into *interconnected modules* to be robust enough in order to deal with noisy data and to recover from erroneous intermediate results generated by any module. Moreover, such a modular structure facilitates the incorporation of those requirements which are specific of a particular application domain.

As described in [102], video surveillance systems are a particular example for generic *Image Sequence Evaluation*. ISE systems accomplish two tasks. First, they

[1] This assumption is used basically when the acquisition device (e.g. a camera) remains stationary with respect to the scene being recorded. This configuration is typical of video surveillance systems.

transform image data into high-level descriptions. By using the term *high-level* we actually imply the existence of an abstraction process, which is reflected in ISE systems by means of intermediate representations. The second goal is to reason about these generated descriptions, which leads to system *reactions*. Generically, such reactions take the form of signal processes or conceptual terms. The former implies to activate different notice or warning signals (such as acoustic alarms, lights, or control signals to mechanisms of access) depending on the observed patterns of motion. A *conceptual description* requires to associate natural language components, such as verbs, nouns, adjectives, or entire texts, to observations of temporal events. Conceptual descriptions are preferred to be built in a suitable way (for example, in terms of logic representations), so further reasoning becomes feasible by means of algorithmic inference engines, for example.

As commented before, there is a set of issues which complicates the extraction of conceptual descriptions. In order to introduce these issues, we need first to define precisely the *domain* where our system is going to look for events. A *scene* is defined as the restricted portion of the 3-D world recorded by a sensor, commonly a camera, and it contains moving agents and static components such as buildings or trees. Thus, the input of our system consists of a sequence or sequences of frames, each frame comprising the 2-D projection of the recorded scene at a given time. Note that even before any processing step has been applied, the image data lacks some of the original information (about depth, for example), and other knowledge becomes unavailable (for instance, when an agent is being occluded by a tree) or modified (like the real sizes of the objects within the scene). This fact has to be taken into account as follows.

ISE systems should incorporate some kind of *a-priori* knowledge in order to correct such incompleteness of the available data. The a-priori knowledge sources are usually embedded in *models*, and they are preferred to be scene-dependent. Thus, knowledge provides the chance to free generic motion analysis methods from the characteristics of the specific scene for which these methods are applied. For example, suitable a-priori knowledge for video surveillance systems consists of:

- camera models to determine the best acquisition configuration,
- agent motion models to track occluded agents,
- scene models to predict occlusions due to static components, and
- calibration models to recover the 3-D positions of the agents and to obtain the real relative sizes of different objects.

All these scene-specific models help us to understand the requirements and failures of the system. Also, these models provide constraints and information that allow to optimize the processing time of the system, and to reduce the complexity of the source code. As a consequence, the a-priori knowledge should be determined firstly based on the problem domain and, transferred subsequently in terms of explicit models into part of the system to be designed.

In fact, a video surveillance system requires to confront different tasks. First, agents should be located robustly in the scene. After that, a motion description is generated for each agent by considering a subsequence of recorded frames. A sequence of motion descriptions constitutes a pattern to be compared with previously learned patterns. Moreover, the *history*[2] of recognized patterns can help to infer the developments in the scene. Lastly, a description of recorded developments is generated using natural language.

Next, an analysis of the problem domain helps to determine the required a-priori knowledge. First, agents need to be detected from the raw input image data. But locating agents precisely in outdoor scenes is quite difficult due to different climate conditions (illumination changes), variable background (structural changes) and occlusions (an agent dis/re-appears suddenly). Background modeling and visual tracking algorithms are the techniques commonly applied to supplement the measurements. In order to help both the locating and tracking processes, a representation of the agent is usually instantiated.

The next step describes the motion of the agent during several consecutive frames, which constitutes a *pattern of motion*. The motion representation is preferred to be scene-independent and non-ambiguous. Also, it should be general, in order to be able to represent all the feasible patterns to be reported by our system, and discriminative enough to enhance the distinctions between different patterns. Subsequently, despite the selected description, it is obvious that first it has to be clearly defined which patterns of motion should be reported by the system. These patterns are required to be represented in a proper way so that they can be recognized in image sequences. This implies a previous analysis of the pattern characteristics to be recognized in order to develop a proper representation. Furthermore, in order to achieve motion recognition, both a comparison measure and a confidence margin should be defined to select the closest, most similar pattern.

Conceptual descriptions for each agent are inferred from the recognized motion patterns. Specifically, a correspondence process is attained between the identified patterns and the knowledge about the task domain, which is called *integration*[3]. It is important to note that *understanding* a scene (defined as dealing with conceptual descriptions) requires of knowledge about the task domain, commonly referred to as *context*. By means of context, the problem domain is constrained, so interpretations are restricted to those which are consistent with such knowledge. Thus, only meaningful semantic descriptions are generated. Context can be built up from a combination of two sources of knowledge [125]: the physical environment (i.e. representation of its structural properties) and the symbolic world (i.e. representation of rules, prepositions, predicates and algorithms). For instance, the static components of a scene can be modeled by means of context. Such knowledge is used to help the locating process (by defining regions of interest), to compute 3-D features of the agents from image

[2]Here, the term *history* refers to a temporal sequence comprising both previously and currently recognized patterns.

[3]*Integration* in the video surveillance domain is defined as the transformation of numerical results in such a way that they become fully a part of a conceptual description. As discussed later, this transformation process incorporates *uncertainty* to the system.

coordinates (in order to estimate depth information and to instantiate 3-D models), and to predict and handle occlusions (due to buildings, for example).

In addition to context, *inference processes* are required, which supervise the states of the analysis process and the search space for hypotheses. Inference processes strongly depend on how knowledge base is represented.

There is a final issue not involved with human motion patterns neither with context: to assure a quick response when some abnormal situation occur, for example the detection of intruders in parking lots or of problems in the traffic flow. Warning notes generated much too late will interest no one. Consequently, real-time processing is a critical restriction of these systems, methods to be applied thus should be highly optimized in terms of processing time[4].

An example of ISE is provided in [61], where the patterns of motion of rigid agents (i.e., cars) in streets are evaluated. The motion description is represented by means of the car location, speed and orientation. Thus, they can infer conceptual descriptions such as *acceleration*, *stopping*, or *driving slowly*. Also, they locate the rigid agent in the 3-D scene, so its spatial relationships with other rigid objects and the static scene components can be inferred. In fact, by analysing the spatial relationships of an agent with respect to its environment, one can infer that a car might stop due to the existence of a halt sign, an intersection, or another car ahead; or one can state that a car is turning to the left through a lane.

In this Thesis, we are interested in describing the structured motion of an *articulated object* (i.e., a human agent, semantically referred to as *actor*) in terms of its posture and limb motion, in contrast to analyze solely the location, orientation and velocity parameters of a single rigid object within the scene. Particular gymnastic or yoga exercises, but also running, bending or walking provide examples of typical patterns of motion.

1.2 Human Sequence Evaluation Architecture

As described in [125], at the beginning of chapter 8, "knowledge based scene interpretation is a complex task, systems for performing this task tend to be complex, too". We have presented so far the knowledge required to describe motion patterns recognized within a recorded image sequence. Hereafter, we restrict our domain to human motion analysis: the term *Human Sequence Evaluation* (HSE) will denote the specific application domain of Image Sequence Evaluation in which the agents involved are actors.

However, when dealing with human agents, additional issues should be addressed. Since the motion of the human body will be described, it is obvious that a human body model is required. Another issue to be considered is *time* representation. Time

[4]However, real-time processing is not a compelling issue which should determine the design of the system: as soon as the computational expenses decrease, the preconditions change and our approach might become worthless compared to alternatives.

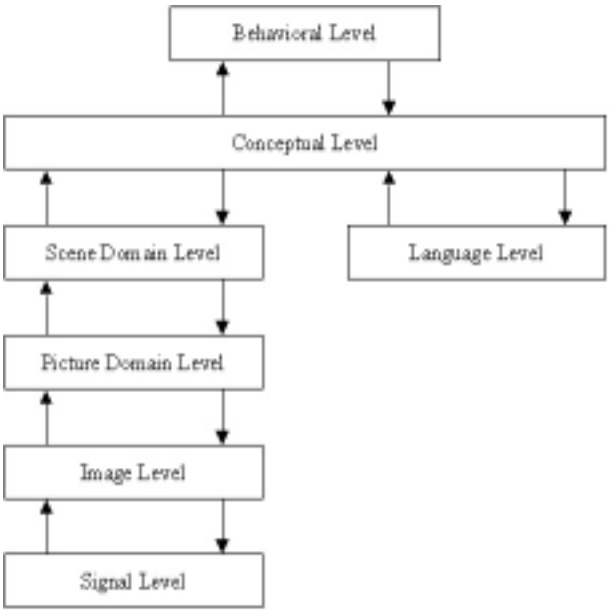


Figure 1.1: Modular scheme for Human Sequence Evaluation. See text for details.

should be modeled in order to determine the duration of a pattern, to model changes of the attitude during a performance and to find temporal relationships between patterns. Moreover, context should be represented in order to derive plausible interpretations and predict future responses properly. Also, context helps to describe interactions between actors (such as meet and split, or meet and continue together) and to distinguish active from passive agents.

All these issues are based on knowledge of different nature. In order to cope with them, Kanade suggested in [79] to describe these systems by means of a modular scheme. As a result, an architecture which differentiates between a *Picture Domain* and a *Scene Domain* was first proposed. Following the same strategy and considering the structure proposed in [102] for ISE, Fig. 1.1 shows the architecture required to perform HSE [5].

The interpretation of human motion is treated as a transformation process between raw video signals and high-level, qualitative descriptions. This integration is subdivided into layers and in each layer, both suitable representations and context are considered. In fact, the combination of knowledge provided by models and context, together with assumptions and restrictions, are exploited at each level to design robust and useful HSE systems.

Next, we detail the first four layers of the HSE architecture. The first level is called the *Signal Level* (SL). This layer provides image data and information about camera parameters, for example in order to perform calibration. Thus, it is possible to

[5] The scheme presented in [102] is also an extension of [79].

represent the viewing conditions, such as the recording technique, the viewpoint and the image resolution. The camera configuration can be also modified, thus allowing to vary the recording conditions (for example, when the camera zooms in on an actor in the scene). A camera interface is required in this layer to receive and send information from and to the camera, respectively.

The next level is called the *Image Level* (IL). Here, a sequence of raw image data, called low-level image *measurements*, is processed in order to locate the actors in each image. As a result, information about moving or foreground regions is provided to the following layer, the *Picture Domain Level* (PDL) in the form of moving blobs, for example.

At the PDL, 2-D representations of the actor are calculated based on the regions segmented at the Image Level. These representations constitute the *Picture Domain Cues* (PDC), for example, the bounding box or the centroid of the blob. Possible segmentation errors generated at the Image Level should be addressed here by taking means of tracking. The tracking process computes the estimated geometric representation (called the *state of the agent*) of the actor in the image at the next time step. The tracking process requires a motion model, and the expected state of the actor can be sent back to the Image Level in order to improve the segmentation process. Furthermore, the picture-based representation of the actor can be enhanced[6] in order to model human attitudes in the time domain. Thus, it is possible to model human motion patterns and, subsequently, to perform picture-based motion recognition. Note that models and algorithms used at this level are based only on features observed in the picture, such as line segments, regions, or intensity values.

At the *Scene Domain Level* (SDL), models and algorithms are based on 3-D configurations of objects in the scene. This level requires a calibration process which can include two or more cameras[7]. As a result, actors are treated as 3-D bodies, so the definition of a predefined 3-D human model is required. These 3-D models, together with other scene-domain-based features (such as surface orientations, reflectance or lighting conditions), constitute the so-called *Scene Domain Cues* (SDC). When time is involved, we can compute 3-D human motion characteristics of selected limbs and joints. Consequently, 3-D human motion evaluation is applied at the SDL by using suitable SDCs. Also, it is possible to perform tracking at this level by using a 3-D motion model. This process is commonly called *motion capture*. Extended over time, the resulting representation allows to perform scene-based human motion recognition, that is, motion recognition based on configurations of the 3-D human body model over time. Also at this level, we can use the knowledge about the context (such as the 3-D model of the scene) in order to improve the pattern recognition rate or to simplify the instantiation process of the 3-D human body model.

Information about the agent at the PDL is made available to the SDL, and vice

[6] In the sense of being transformed to a suitable representation which can model temporal variations of the posture.

[7] Of course, depending on the restrictions and assumptions of the system (for example, restricted to an orthographic view or by requiring a previous initialization of the 3-D human body model), the instantiation process of a 3-D human body model can be simplified.

versa: a critical issue is the question of how to associate the PDC to the SDC. When dealing with non-rigid bodies like humans, a correspondence process between components of 2-D representations of the Picture Domain, such as edges, and components of the 3-D human body model at the Scene Domain, such as cylinders, is not straightforward[8]. Commonly, once the 3-D model has been instantiated, it is projected into the image to assist the extraction of PDCs when self-occlusions of limbs or occlusions of part of the body due to static components of the scene (such as trees) occur. Subsequently, the projected 3-D human body model can be used to perform motion recognition at the PDL. This procedure is commonly referred to as $2\frac{1}{2}$D [96].

Note that human tracking and human motion recognition can be performed at both the PDL and the SDL. One advantage of dealing with PDC-based approaches is that they are easily adapted to different scenes, but the recognition process depends strongly on the viewpoint of the camera: 2-D single-view representations can turn out to be ambiguous. SDC-based approaches offer a more accurate description of human motion, but their computational cost can be prohibitive for a certain type of applications, such as video surveillance.

The information obtained at the SDL is forwarded to the *Conceptual Level* (CL) in order to determine tentative descriptions at each time step. Typically, conceptual descriptions are characterized using knowledge about the scene and about the actor. Knowledge about the scene refers to predefined facts concerning a conceptual scene model. The knowledge about the actor comprises all the facts about an agent which can be derived from the geometric information obtained at the IL, PDL and SDL, such as:

- the spatial information of the actor within the scene,
- the relationship of the actor with respect its environment, and
- the recognized motion pattern.

This knowledge is referred to as the *situation* of a given actor [101]. Integration is applied at this level: the set of parameter values which determines the configurations of the human body model over time are associated to a conceptual representation which should be adequate with respect to a required task. Integration implies the transition from pattern recognition to descriptive processes. As a consequence, there exists a data flow in two directions in order to restrict combinatorial explosion of data and reproduction of errors:

- bottom-up data flow corresponds to potential situation descriptions (i.e. hypotheses based on expectations) derived from the motion analysis, and
- top-down data flow is required for hypothesis verification.

That is, the goal is to infer plausible situation hypotheses, but also to *verify* them. The Conceptual Layer may require additional information from the SL, IL, PDL, and

[8] This *bridge* process is referred as the *Physical Layer* in [79].

SDL in order to select the most adequate concept for the current situation, and to reduce the system *uncertainty*[9] about such a situation. For example, if an actor has stopped in front of a traffic light, we can implicitly assume that the actor is waiting to cross. However, we can not verify it: maybe the actor has stopped to ask for a particular street.

The current situation is fed back to the *Behavioral Level* (BL). At this level, the expected temporal evolution of situations are modeled a-priori in order to perform spatio-temporal reasoning, see [61] for details in the traffic surveillance domain. So, given a situation, the set of possible successor situations can be determined, thus allowing the system:

- to reason about the behavior and intention of the actor,
- to provide instantiated situations for description generation using natural language text,
- to determine the set of possible successor configurations of the actor in order to assist the IL, PDL, and SDL at the next time step, and
- to evaluate whether an instantiated situation raises *suspicion(s)* in an observer.

The fourth issue also implies to determine the system *reactions* when an specific situation is instantiated from image data. Three kinds of system reactions are commonly considered. First, reactions implemented in terms of control signals. In this case, they are fed back to the SL in order, for example, to activate warning signals or to close doors. Second, reactions which produce geometric information related to the observed actor within the scene. This information is forwarded to the IL, PDL and SDL in order to assist segmentation and tracking procedures, for example. Also, such geometric information can be used to re-visualize recognized patterns of motion. Third, reactions can produce natural language formulations to be reported to human operators. In this case, system reactions are forwarded to the *Language Level* (LL), which associates the situation representation with natural language text. In other words, the description of the situation is converted into logic and coherent statements.

To sum up, evaluation embeds two separated tasks: first, to obtain a geometric description of the scene in terms of quantitative values (task related to the Signal, Image, Picture Domain, and Scene Domain levels); and second, to associate the geometric description with qualitative concepts within a proper knowledge representation for conceptual description, synthetic generation and deep reasoning about observed conducts (task related to the Conceptual, Language, and Behavioral levels). Note that there is an explicit cooperation between different scientific fields: Pattern Analysis and Artificial Intelligence.

[9] A description of an image sequence is a quite arbitrary series of statements, due to the fact that the system *interprets* the facts shown by the sequence. The term *uncertainty* refers to the inability of categorizing in a precise way a particular instantiation of the situation.

1.2.1 Sources of Knowledge

Once a modular description of an HSE system has been presented, the required sources of knowledge will be identified. In analogy to [60], this analysis will determine the requirements of generic HSE systems in order to detect possible lacks of constraints, and the causes of possible system breakdowns. But even more important, by formulating the sources of knowledge as explicit models, it is possible to design human motion analysis procedures which are independent of the scene in which they are applied. In fact, evaluation in different scenes will imply the substitution of selected models but not a substantial change in the system architecture. Thus, it is feasible to modify, substitute, and evaluate each model independently of the rest of the system. Also, when additional knowledge is required, it is easier to be incorporated by means of an explicit model.

We here describe generically which would be the required sources of knowledge for a generic HSE system, and the means by which they are provided.

The Signal Level

A *camera model* will allow us to change the recording conditions by varying the camera parameters (for modification of the resolution or the viewpoint, for example). This can be useful when a human is detected far away in a scene, so a posterior zoom can provide the system with a more detailed image. A *digitizer model* is necessary in order to transform the output signals of the camera to image pixel values (the so-called *measurements*). Thus, we determine the *format* of the image data, such as the available rank of pixel values (for example, 8 bits per pixel), and whether the image is gray-level or it contains color information. This model depends on the characteristics of the camera (infra-red, BW, color, or progressive cameras). Lastly, a *decoder model* is required when the image is provided in some encoded format (for example, due to compression or security requirements).

Moreover, an *actuator model* can be implemented in order to activate possible signal-based re-actions of the system, such as generating warning alarms or closing/opening admittance doors, depending on the current observations within the scene.

The Image Level

Once the image data is available, the system should detect moving or foreground regions[10]. Motion analysis or a suitable background model are employed, respectively in order to aid the segmentation process of actor images in digitized video frames. In both cases, we can use the knowledge about the continuity of lines to find lines;

[10] Foreground regions are defined with respect to a background model: those regions which do not correspond to the background model are considered as foreground regions. Thus, there are moving regions, such as the leaves of a tree due to wind, which can be modeled as background, and motionless regions, such as an actor stopped in front of a traffic light, which can be detected as foreground.

the texture primitives for texture classification; the homogeneity of regions to build areas; or the gray level changes for edge detection.

Also, knowledge about color metrics can be applied to image color transformation (for example, RGB to HSL), depending on the advantages that color information provides to the system.

The Picture Domain Level

At this level, image features are manipulated in order to obtain a 2-D representation of the actor which constitutes the *2-D human body model*. The representation relies on the results of the segmentation process at the Image Level. Silhouettes and blobs provide examples of 2-D human body models.

The tracking process is performed by using a *2-D human motion model*, which is usually implemented in discrete form by means of difference equations. Thus, we are able to estimate the expected position of the human body in the image at the next time step. But also, the human body parts can be tracked separately when a suitable 2-D human body model is used (for example, 2-D models based on sticks or joints). Note that tracking is based purely on image features.

Once the human is successfully tracked, the set of parameters, which characterize the model, can be extended over time, thus allowing the analysis of the temporal evolution of the human body parameters in order to perform motion recognition. A representation of motion is built using two consecutive frames, which embeds motion characteristics, or the human pose, or both. As a result, motion patterns represent the variation of the actor posture during an explicitly defined time value, usually in a supervised manner. A similarity measure and a confidence margin should be defined to implement recognition.

Context can provide restrictions which will improve the tracking and motion recognition results, for example by defining regions of interest, by restricting the plausible human body configurations (by means of *dynamic models* of human motion), or by providing information about the projected scene.

The Scene Domain Level

When 3-D knowledge is exploited, the 3-D configuration of the human body can be recovered and placed in the scene by means of a predefined *3-D human body model*. Consequently, 3-D human motion can be analysed. 3-D human body models can consist of sticks, but also of polyhedral, ellipsoidal, or cylindrical components.

A *correspondence model* links the representation in the Scene Domain with image features in the Picture Domain, usually edges or regions. On the one hand, the correspondence process fits the 3-D person model to the image, by transformation of scene points to image points. This process can be implemented in terms of chained functions which transform body part coordinates to image coordinates [139]. Differences be-

tween the projected model and the image features can be used to calculate corrections over the parameter values of the 3-D human body model. On the other hand, the configuration of the 3-D human body model is recovered, or rather estimated, from image features. Anatomical restrictions about allowed body configurations (for example, in terms of kinematic geometry of joint coordinates) can improve this estimation process.

A *calibration* process is required to infer 3-D positions within the scene from 2-D coordinates of the image. Also, we can derive the real sizes of objects in the depicted scene. Therefore, calibration requires a camera model at the SL.

Tracking is applied to a particular instantiation of the 3-D human body model. As before, a *3-D human motion model* is required to estimate the next expected position and/or posture of the actor at the next time step. Restrictions due to physically plausible motion of the human body can be incorporated by means of kinematics of human motion, thus allowing to improve the tracking process in terms of time, accuracy and reliability issues. A *3-D scene model* may be incorporated to detect occlusions due to the static components of the scene, and to provide context information for whole-body tracking.

Extension of the human motion analysis to the temporal evolution of 3-D human body configurations enable motion recognition. The 3-D body representation is usually simplified by considering a reduced set of body characteristics which will be used to learn and recognize patterns of motion. For example, if the human body is described in terms of joints and limbs, we can just analyse the angle variations of a reduced set of joints in order to best characterize a motion. Similarly to the PDL, a similarity measure and a confidence margin should be precisely determined for recognition purposes.

Although the human motion analysis presented in this Thesis is achieved completely at the SDL, there are publications which project the 3-D human body model into the image, in order to perform motion recognition at the PDL. These $2\frac{1}{2}$D approaches usually require the set of possible poses to be defined a-priori.

The Conceptual Level

We assume that interpretations inferred from the actor are characterized by the combination of three sources of knowledge:

- The spatial information of the actor within the scene, such as the position, velocity and orientation in the floor plane, for example.
- The relationship of the actor with respect to its environment, such as the presence of other actors nearby or the proximity of static objects represented in a conceptual scene model.
- The recognized motion pattern which the actor is performing.

As described before, this knowledge at a given point in time is called *situation*. Based on the geometrical information obtained at the SDL, the aim is to apply *integration* to associate a conceptual interpretation to the resulting numerical values. This integration process performs an *abstraction* step to convert quantitative data to qualitative knowledge (for example, in terms of logic predicates), which can be achieved using different strategies.

For example, we can abstract quantitative state parameters by associating 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, quantitative values are associated to fuzzy attributes.

Also, we can derive spatial relations by considering the position of the actor in the scene. In this case, a *conceptual scene model* is required to describe the spatial coordinates of the actor with respect to static objects, other actors, and specific locations within the scene. Integration can be implemented by applying a distance function between the positions of the actors/objects in the scene. Also, a discretization of the resulting distance value can be obtained by using Fuzzy Logic.

Lastly, we can associate a label to predefined motion patterns. For this purpose, motion recognition algorithms are addressed to derive the concept associated to a recognized motion pattern. Therefore, we can provide the label of the recognized pattern to the CL when such an pattern is finished. Alternatively, we can derive the label of a *tentatively recognized* pattern *while* it is being observed. In this case, several possible labels can be instantiated at the same time, due to the fact that different patterns of motion can exhibit some identical features during their performances.

But the situation can also reflect additional information about performance characteristics, such as the *style* of the detected motion, for example[11]. In this case, integration is required, too.

The Behavioral Level

The temporal evolution of recognized situations are modeled at the Behavior Level (BL), which addresses three main issues. First, the aim is to infer the situation, i.e., a qualitative description of observed motion in a temporal and spatial context. Second, the generation of synthetic motion patterns from previously generated qualitative descriptions is also handled here. Third, it is possible to *reason* about what is happening in the scene using comprehensible terms. That is, we consider here that there is human-computer interaction: users can ask the HSE system about the scene, and the system can generate scene descriptions after reasoning. *Evaluation*, in the sense of explanation, helps to make easy to understand such systems by non-expert users.

Context helps to reduce the inherent uncertainty about the currently prevailing situation. A *scene model* (2-D or 3-D) provides useful information for spatial reason-

[11] A *style* is defined as a manner or way of human performing. In fact, some manners of performing are typical of particular actors.

ing, and the position of the agent within the scene can help to infer the conduct and the goal of such an agent. For example, in order to establish that an actor has come inside a building, we can model the positions in the scene of visible doors, i.e., our regions of interest. An actor which disappears at those positions leads to semantic interpretation. Another example is related to human occlusions by static components of the scene: occlusions can be predictable, thus improving the tracking process (in terms of robustness) so interpretation can be achieved.

Also at this level, a *knowledge representation language* is designed in order to reason about the motion and intention of actors. Although techniques for evaluation depend on the selected knowledge representation language[12], there are general requirements to take into account.

A first requirement is the evaluation of the performance of the system. Thus, it is possible to *justify* the results of the system reasoning process by controlling the inference process which is applied. The aim is:

- to trace the computational process which generates the result,
- to determine the internal information requested by the system, and
- to reason about the selection of particular reactions by the system.

Thus, it would become easier to *debug* the system. Consequently, there is a reduction of implementation errors. Also, as a result of this debugging process, the designer can decide to incorporate extra knowledge (by means of models, restrictions, and assumptions) in order to improve the performance of the system in terms of reliability. So there is an increase of *confidence* of the users in the results reported by the HSE system.

Another requirement is the evaluation of the strategies of the system. This issue is related to the capability of asking about how the system would address the resolution of a particular problem, in terms of the inference steps to be applied. This capability asks for the necessity of control algorithms, which determine the inference processes to be activated in order to answer the user.

A third requirement is the evaluation of the knowledge used by the system. Therefore, the user can ask about the content of the knowledge base. Thus, it is possible to obtain a description of *what* is happening in the scene, and even *why*: the system can reason about the availability or no availability of particular knowledge. As context helps to infer the knowledge of the system, the user is actually learning about the context specifications while asking the system: in other words, there is a *training* process of the user in the specific task domain in which the system is addressed.

Evaluation of the knowledge deployed by HSE systems can be implemented using two strategies [7]:

[12] In the literature, semantic networks, logical, and procedural schemes are considered as suitable knowledge representation languages.

- let the HSE system generate a synthetic image sequence using the textual description obtained from a previously recorded image sequence. Both synthetic and original sequences can be compared to evaluate the HSE system knowledge.
- let the HSE system generate the textual description of the synthetic image sequence. Subsequently, comparison of the original and the synthetic textual descriptions can be made to detect deficiencies in the HSE system knowledge.

Situation descriptions generated from image sequences are provided to the last level, the Language Level (LL).

The Language Level

At this level, natural language text descriptions are generated for situations, which involves the generation of logic and coherent statements by means of natural language formulations. A suitable *grammar* needs to be designed to combine the conceptual terms of the current situation into meaningful statements.

The situation of an actor describes qualitatively the spatial configuration of such an actor within the scene, the configuration of the human body (with respect to pose or motion characteristics), and the goal of the actor. This qualitative information generated at the CL is used to associate the quantitative and geometric terms obtained at the SDL with nouns, motion verbs, adverbs and adjectives at the LL.

Furthermore, the temporal and spatial evolution of situations modeled at the BL will provide information about the relation of manner or quality, place, time, degree, number, cause, opposition, affirmation, or denial between situations and agents: temporal transitions and spatial relationships between instantiated situations serve to connect properly the statements by means of *adverbs*.

Next, the relationships between the Language Level and the previously described levels will be pointed out. The user interacts[13] with the system through the LL. As a result, inference processes are activated at the BL to reason about instantiated situations at the CL. Evaluation of additional numerical results provided by the IL, PDL, and SDL can also be required to handle uncertainty. The result of the inference process is fed forward to the LL in order to generate a natural language description embedding the answer of the system.

Since the two main goals of HSE are the description and generation of motion patterns, integration is required to derive a qualitative description from recognized motion patterns, and also to generate a synthetic film during which a particular motion pattern is visualized. This abstraction process is addressed next by establishing a suitable human motion taxonomy. This taxonomy reflects the intermediate steps required to implement the transformation of geometric information at the SDL into conceptual terms at the BL.

[13]The user can interact with the system through menus or using query languages. A *user interface* is required to make this interactive process easy.

1.2.2 A Human Motion Taxonomy

Literature reviews classify existing approaches based on different taxonomies. Usually, the classification criteria are based on geometrical or methodological issues. On the one hand, different reviews focus the problem of defining a taxonomy to organize existing publications in terms of the geometric representations implemented [1, 33, 51, 96]. On the other hand, previous publications can be classified into categories by using methodological criteria instead (for example, model-based versus feature-based), in a similar way to that presented in [141]. Alternatively, the classification presented in [32] reviews the literature in terms of the models considered.

Since HSE associates geometric information with conceptual statements, we are interested in establishing a motion taxonomy using knowledge concepts in order to establish a knowledge-based classification of motion representations which will guide integration implementation. However, few taxonomies based on knowledge have been proposed in the literature.

For example, the taxonomy proposed by Nagel in [101] is suitable to perform conceptual description generation. The levels are called *change*, *event*, *verb*, *episode*, and *history*: a change is a discernible difference in a sequence; an event is a change which is considered as a primitive of more complex descriptions; a verb describes some activity; an episode describes complex motions, which may involve several actions; and a history is an extended sequence of related activities. These terms are more related to story understanding, because the goal is the generation of conceptual descriptions by means of natural language. These descriptions are more suitable to perform logic reasoning about the scene without human intervention and without reliance on domain specific knowledge. It is also possible to infer interactions between multiple objects (both static and moving) by means of their spatial and temporal relationships. However, Nagel's classification is mainly derived to perform reasoning, so it is not obvious how to address intermediate representations of complex human motion patterns according to his taxonomy.

Centered in human agents, different description levels are suggested by Bobick in [20]. In this paper, several publications are also classified according to the degree of knowledge required to describe semantically the scene. Bobick described three levels: *movement*, *activity* and *action*. A movement is a predictable motion that can be defined by a space-time trajectory in a proper feature space, so no knowledge other than motion is needed. An activity is *composed* of a sequence of movements. Here, different approaches to model temporal relationships between movements are discussed. An action involves semantic knowledge related to the context of motion, that means, inferences rely on domain specific knowledge. This classification is suitable to perform comparison between existing motion recognition publications. Furthermore, it makes explicit the requirement of deriving time and causal relations from statistics of image motion. Reasoning about the scene can only be achieved if perceptual implications are observed. Therefore, reasoning is mainly restricted to time dependencies between actions.

In this Thesis, a combination of both taxonomies is used. We want to describe

our approach as a human motion analysis process embedded in a generic Human Sequence Evaluation framework. So, on the one hand, Nagel's classification will assist our system to generate semantic descriptions suitable to perform deep reasoning. And, on the other hand, we adopt Bobick's terms in order to design intermediate representations appropriate to perform complex human motion analysis. These levels are called *movement*, *action*, *activity* and *behavior*, which are described next[14].

A *movement* represents a change of human posture between consecutive frames[15]. No knowledge about the context is required at this point. Human movement analysis is found to be used as a constraint factor to improve the tracking of human body parts. So an appropriate human body model is needed to be pre-defined. Also, it is considered as the lowest stage of more complex human motion descriptions [22]: movements can be characterized in terms of a space-time function, so different performances of the same movement recorded from the same viewpoint will have similar characterizations.

We consider a human *action* as a process of human performing that has a significance to our system. A process of human performing refers to a predefined pattern of motion. Significance means that these actions will be learned so afterwards their recognition can be properly achieved. Using the previous definition of movement, we define an action as a temporal series of human movements. As a consequence, a method is required for handling time. This requirement involves a proper way to determine the temporal relationship between these constituent movements. That means, the temporal order of the movements will actually determine the action which is being performed. Furthermore, variability in the duration of these movements should also be considered: the recognition of an action should be independent of its temporal extent. Typical actions are running, jumping, grasping or bending. Using conceptual terms, actions can be described by *motion* verbs. No reference to the context needs to be addressed in order to characterize an action: the recognition of an action relies on the recognition of its constitutive movements. It is important to note that, at this point, the *goal* of an actor is *implicit* in the performance of an action, so the knowledge of the context is required in order to determine such a goal[16].

An *activity* is composed of a sequence of two or several human actions, plus the transition between consecutive actions. A transition refers to the variation of the human posture from one finished action to the next one described by the activity. A motivating example is the so-called *thief sequence* [27], in which a "thief" walks, stands up straight to look around, bends to pick up (an object), and then runs away. No reference to the context is applied here: an activity is *generically* recognized as the consequence of the recognition of its constitutive actions. An activity can be expressed in terms of a *series* of motion verbs.

Lastly, the term *behavior* refers to one or several activities which acquire a meaning

[14]These terms can be defined in a different way using different names. Furthermore, additional levels can be added. But this taxonomy is adequate enough to present the analysis of human motion to be studied here.

[15]A movement can also represent a change of place, for example during jumping actions.

[16]That means, the goal of an action is the logical consequence of its execution.

in an specific context. For example, we can model a waving arms activity. Recognizing such an activity in a football game (ball request) has a different meaning that in a subway station (help request). That is, the description of the temporal changes of an actor is combined with the relationship of such an actor with respect to its environment. Context information will provide the required knowledge necessary to generate descriptions about the actor in a particular scene and to express the recognized motion verbs related to the scene components (the static objects and other actors involved in the activity). Moreover, the context will also help to determine the *goal* of the activity. We consider that the goal, if it is not known in advance, can be inferred from the set of interpretations of the system in a known context. If the goal is known in advance, the context provides the information required to state whether the goal has been successfully accomplished.

So the behavior of an actor is characterized by the combination of:

- a description of its motion, i.e. the action which is being performed,
- a description of its spatial configuration within the scene and with respect to other actors, and
- a description of its goal, known in advanced or inferred from the context.

This knowledge in a single point in time is called *situation* in [101]. Using this term in our taxonomy, a behavior corresponds to a temporal sequence of situations. Therefore, a situation corresponds to a movement which acquire a meaning in a specific context, and determines the conceptual descriptions to be inferred from image data.

1.3 Situation Graph Trees

A survey of systems creating high-level descriptions from image sequences is presented in [32]: several procedures can be chosen to develop a motion understanding system, depending on the requirements of the discourse domain. Motion understanding systems rely on knowledge based on the geometry of the scene and the numerical data obtained from pattern analysis procedures. In our case, additional requirements are found to be satisfied by analyzing the architecture studied before: we explicitly obey the modular scheme of HSE.

In this Thesis, our procedure for human action analysis is designed and implemented at the SDL, thus allowing to determine 3-D spatial relationships of actors and predefined objects. Therefore, we distinguish between the numerical representations obtained at the SDL and the situations instantiated at the CL. We assume that the geometrical information obtained at the SDL is provided to the CL uninterruptedly over time. Furthermore, plausible situations should be organized into a temporal and conceptual hierarchy at the BL in order to generate behavior descriptions.

The generation of textual descriptions from image sequences is an important requirement. But also, the inversion of this process, i.e. the generation of synthetic sequences should be feasible by using the conceptual descriptions instantiated at both CL and BL. Therefore, algorithms for synthetic video generation should rely on knowledge very similar to the knowledge required for video description. Furthermore, the generation of video sequences should be deterministically defined. That is to say, the animator should be able to build activity and behavior representations by establishing which, and in which order, human actions are synthesized. The last requirement is to support not only the execution, but in addition the diagnosis steps during the continuous development and test of HSE systems.

Similar requirements can be found in [61], where a system for recognition of traffic situations is presented. Next, the characteristics of the deterministic formalism used in such a paper are pointed out.

Based on the geometrical information at the SDL, logic predicates are instantiated by employing a *Fuzzy Metric Temporal Horn Logic* (FMTHL). This predicate logic language treats dynamic occurrences, uncertainties of the state estimation process, and intrinsic vagueness of conceptual terms in an unified manner, see [126] for details. As a result, temporally and conceptually isolated logic statements are instantiated for each frame at the CL.

Subsequently, these statements are embedded into a temporal and conceptual context by using *Situation Graph Trees* (SGTs). For this purpose, the concept of *generically_describable_situation* presented in [101] is used: 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 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 at 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.

As SGTs are defined using a deterministic formalism, we can establish which sequence of situations should be investigated during both descriptive and generative procedures. Furthermore, SGTs can be converted into FMTHL-logic programs. Thus, the behavioral knowledge and the situation analysis is based on formal logic inference.

Consequently, we will use the SGT formalism for behavior representation at the BL of the HSE architecture. Therefore, the numerical information obtained at the SDL should be converted into isolated logic predicates at the CL by applying integration. SGTs will determine at the BL the sequence of situations, which should be instantiated over time, in order to perform spatio-temporal reasoning for human behavior analysis.

This implies to address two issues. First, as SGTs are based on logic predicates, we adapt the terminology used in [61] to be applied in our discourse domain. A *terminology* consists of logic-rules and facts which regard the actor itself, the actor

Figure 1.2: A sequence of frames depicting an athlete throwing a javelin. Training athletes attempts to reach predetermined attitudes during the action in order to best execute the throw.

and other actors/objects, the actor and specific scene locations, and the knowledge about the scene. Also, the terminology should define the possible set of reactions that any actor can perform in our discourse domain. Second, once a suitable logic-based terminology is defined, the next step will be centered on defining SGTs which represent expected behaviors at specific scenes. Subsequently, human behavior description and generation will be achieved by using the same knowledge representation.

1.4 The *Key-frame* Approach

The key point of this Thesis consists in defining a proper human action model at the SDL so that human actions can be learnt and used in different applications, such as human action recognition and synthesis. In order to accomplish this task, our human action model will be based on the *key-frame* concept.

Fig. 1.2 illustrates the idea and motivates the development of this approach. In this sequence, five attitudes are presented which occur while throwing a javelin. The goal of this action is to hurl the javelin as far as possible. Thus, training athletes consists in teaching a throwing technique. This optimum[17] technique consists of reaching predetermined attitudes which will provide the athlete with the greatest strength at the instant of hurl. So, in order to perform such a complex series of movements, the prototypical action is decomposed into key postures, as the figure reflects. Then, the training of the athlete is driven to reach these postures at a particular phase during the entire process of throwing a javelin in order to be best positioned for reaching the next posture[18].

We derive several conclusions from the previous example. A representation of a prototypical action should embed the *essential* attitudes of such an action, in order to be suitable to be used for learning purposes. The term essential means here that in order to perform an action, the human body should adopt, at the very least, these postures. An action is then described in terms of time-ordered essential postures. As the recognition of these postures during a sequence allows the recognition of the entire

[17] The term *optimum* here refers to best execute a given action. Usually, optimum techniques are usually theoretically described by sport trainers, where the essential postures of the action execution are established. Then, an action executed optimally can be referred as *prototypical*.

[18] In this example, each movement is executed to reach an attitude from which the next movement can be executed optimally. Concatenation of these optimal movements (or attitudes) will provide the maximum impulse to the javelin.

action, these essential attitudes are considered as the *reference* postures of such an action. The term reference denotes the idea that these essential postures will become the means of identification, that is, they will be suitable to be used for recognition purposes. These postures characterize the action so, given a reference posture, the next reference posture can be determined, and *expected*.

We consider that in order to optimally execute an action, each movement is goal-oriented to settle the human posture to be best placed to perform the next movement[19]. This effect is also used in human gesture recognition (the end of the current gesture is influenced by the motion of the gesture planned to be executed next), and is similar to coarticulation in speech [23]. That means that some kind of prediction can be applied while executing an action. These characteristics motivate the key-frame concept and justify such a proposal for action representation: an action is divided into *segments*, where the reference attitudes mark the beginning and the end of each segment. We denote the images of these reference postures as the *key-frames*[20] of an action.

So an action can be represented by means of a sequence of key-frames or, in other words, body posture configurations. An obvious extension is to enhance this representation with the characteristics of motion transitions between key-frames. Time is represented by considering the order of the key-frame sequence. We do not use a quantitative description (for example, the overall number of frames or seconds that an action lasts) but a qualitative, relative representation. In fact, the key-frames can be found non uniformly with respect to the overall number of postures which constitute the performance of an action.

Another advantage of action models based on key-frames is the ability to perform on-line recognition, that is, an action is recognized *while* its performance: we are actually *parsing* the execution of an action, so recognizing its key-frames will involve recognizing the action. Strictly speaking, it is possible to *hypothesize* about the action currently being executed when recognizing a key-frame. Several action hypotheses are possible at each time step, and the number of valid hypotheses decreases when detecting further key-frames. Also, key-frames provide facilities to human action synthesis, by means of interpolation techniques. This advantage will be exploited when generating animated sequences of virtual agents.

In this Thesis, we define a procedure for extracting automatically the key-frames from a sequence. Four issues, already introduced before, will be involved in this process. First, the selection of a suitable representation of the body posture which will enhance its anatomical properties in order to maximize variations between different body postures. Second, a proper algorithm which selects automatically those postures of an image sequence which are considered as the most characteristic ones according to predefined criteria. Third, to use the key-frame approach in different applications

[19] This goal-oriented treatment of movements can be found in many Oriental arts. *Tai-Chi* is a nice, motivating example: there are a number of so-called *forms* which consist of a sequence of movements. The aim of Tai-Chi is the precise execution of these forms. Many practitioners notice benefits in terms of correcting poor postural, alignment, or movement patterns which can contribute to tension or injury.

[20] This term in the human motion analysis domain was coined by Akita [2].

which are implemented in order to validate our action model. And lastly, to adapt the key-frame approach in the SGT methodology in order to perform integration and, consequently, HSE.

1.5 Contributions and Thesis Outline

The goal of a Human Sequence Evaluation system is to infer the behavior of a human agent within a scene, and also to validate inferred behavior descriptions. That means, knowledge about the agent behavior should be represented in a proper way which permits to generate a description of agent behaviors, and to create a synthetic film in which a recognized behavior is visualized. Consequently, the transformation between the geometric information obtained at the SDL and the conceptual terms modeled in the form of a-priori knowledge representation at the BL should be also addressed in this Thesis. Specifically, we address the integration process by presenting the following contributions:

- A human motion taxonomy is established: a hierarchy of human motion terms is presented, which helps to determine the knowledge required to perform Human Sequence Evaluation.
- A novel 3-D human body model is defined to properly represent the variation of the human posture during an action performance.
- A novel human action model is presented in which the human posture variation, observed during several action performances, can be described in a compact, accurate and specific manner. This human action model provides facilities to human action recognition, synthesis, and analysis.
- A human action recognition procedure is described, which assigns an input posture to the most likely action(s). Therefore, the geometric representation of the human posture is converted into a conceptual attribute, i.e. the recognized action label.
- An algorithm for human action and activity synthesis is presented. Automatic generation of realistic human motion is achieved. Furthermore, transition between actions can be synthesized in order to visualize activities.
- A comparison framework is introduced in order to describe any action performance with respect to its corresponding action model. Also, a synchronization procedure is presented to derive a characterization of the action style in terms of the gender of the actor.
- The SGT methodology is adapted to generate textual descriptions from observed human behavior. Also, the automatic generation of synthetic films using SGTs is addressed. The goal is to establish a link between the conceptual action label and its corresponding geometric human action model, thus allowing to re-visualize recognized behaviors using virtual actors.

These points look for your considerations, and they are presented within this Thesis as follows. By obeying our human motion taxonomy, we describe in chapter 2 different papers which address specific issues of HSE. The aim is to examine existing human motion models, and the tasks which can be accomplished by using such representations. In chapter 3, we present our novel human action model, called *p–action*, which is based on a suitable representation of the human posture. Such an action model is built upon the most characteristic human body postures found during several performances, which are called the *key-frames*. Subsequently, our human action model is used in chapter 4 to implement three different applications, namely human action recognition, human activity synthesis, and human performance analysis. In chapter 5, we confront the task of embedding our human action model within the HSE framework. Subsequently, we show how to infer behavior descriptions of agents within a scene, and also how to generate synthetic behaviors for virtual human agents. Lastly, chapter 6 summarizes the main contributions of this Thesis and discusses possible avenues of future research.

Chapter 2

Related Work

As described in the previous chapter, several issues should be contemplated to develop a Human Sequence Evaluation (HSE) system. This is reflected in the literature: a huge number of papers confronts some of the levels of the general scheme shown in Fig. 1.1, but rarely all of them. However, the number of human behavior analysis algorithms is increasing each day, thus reflecting current significance of HSE systems in the computer vision community [36]. In this chapter, an overview of papers, which develop strategies for human motion analysis at different abstraction levels, is presented to the reader to illustrate integration, or how information can be transformed from image data to conceptual interpretations.

Literature reviews classify human motion approaches based on different criteria. Early surveys were centered on methods based only on geometrical models [1, 33, 51, 96]. That means, most referred papers implement human motion analysis at the Picture and/or Scene Domain Levels. More recently, human motion surveys began to include papers centered on high-level vision issues, that is, approaches whose methodology were based on conceptual representations for behavior reasoning [32, 141]. Therefore, both Conceptual and Behavior Levels began to be exploited using the quantitative information acquired from image sequences.

Following this tendency, we present a survey based on the terms considered for our human motion taxonomy, namely movement, action, activity, and behavior. In fact, our taxonomy reflects the minimal abstraction steps required to derive conceptual descriptions from pixel values. As the taxonomy explicitly differentiates between quantitative and qualitative knowledge, we demonstrate the complexity of a generic HSE system by showing which models, suppositions, and restrictions are usually considered at each abstraction step.

Next, we survey human motion analysis papers based only on geometrical knowledge. For this purpose, movement and action representations will be described according to both the degree of knowledge exploited and the characteristics of the methodology used.

Subsequently, we revisit different approaches for activity and behavior analysis. That is to say, we will detail the degree of knowledge used in the literature to represent the transition between different actions, and to employ the information of the context in order to assist human behavior modeling, respectively. Despite the fact that some progress has been made in behavior understanding, the aim is to show that conceptual interpretation still relies on a proper action representation.

2.1 Movement Descriptions: Analysis from Consecutive Images

The basic task of HSE systems is centered on the detection of actors within the scene by means of event analysis, here called *movement segmentation*. Event detection generates movement representations between consecutive frames, which are built using two different approaches. On the one hand, event analysis is applied with respect to a predefined *background model*. That is, actors correspond to foreground regions. On the other hand, event analysis can be based on *moving regions*. That is, motion information is extracted from two consecutive frames. Considering both approaches, information about foreground and moving regions can be combined to enhance the movement representation.

However, issues like clutter, shadows and changes in illumination conditions are important sources of noise which should be confronted. Indeed, multiple restrictions and assumptions are often contemplated in order to reduce the complexity of generic HSE systems, and to test behavior analysis algorithms based on movement representations obtained robustly[1].

Additionally, *tracking* procedures are usually incorporated in order to reduce segmentation errors. Tracking techniques embed knowledge about the actor, such as its observed motion, appearance, or shape. Using this knowledge, a *prediction* of selected image features is generated at each time step. Afterwards, a *correction* criterion is applied to evaluate such a prediction. Commonly, tracking is used to detect robustly actors within the scene by applying an *human motion model* for prediction and a suitable *likelihood function* for correction.

As the performance of higher levels of the HSE architecture is strongly dependent on the accuracy of movement segmentation, some human motion understanding publications prefer to avoid errors in the input data. For example, a typical domain in which no errors in movement detection can be found is motion capture involving markers on the body[2]. Consequently, a human movement model is automatically and accurately built from image data, and human behavior analysis is applied in scenes without clutter and well-defined agent silhouettes after automatic segmentation.

[1] Commonly used assumptions are [96]: no camera motion; only one person in the scene; slow and continuous movements; movements parallel to the camera-plane; known camera parameters; and known start pose.

[2] Motion capture is the process of recording a live motion event to obtain a 3-D representation of the performance.

2.1.1 Movement Segmentation

As commented in [90], segmentation performance evaluation remains subjective: most papers show few results that "look good", without specifying whether they are typical or the best ones obtained. When reviewing the literature in order to choose and implement the most suitable human movement segmentation algorithm according to our demands, one should analyze the recording conditions and the characteristics of the scene. Similar requirements will produce, in principle, similar results. For example, issues like the size of the agents with respect to the whole field of view; the characteristics used to perform segmentation, for example color or texture information; the presence or absence of noticeable clutter; the nature of the scene, i.e., outdoor or indoor scenes; and the degree of refinement required can help to determine the segmentation algorithm to be developed.

Conventionally, two different approaches are found in the literature, namely, *background modeling* and *motion detection*. The former refers to implement a suitable background model of the scene to determine foreground regions. Afterwards, the movement representation is built on two consecutive foreground actors. Alternatively, motion detection refers to motion information extracted from consecutive foreground regions, which is usually achieved by applying motion correspondence or optical flow. As a result, a movement is described in terms of a proper motion characterization.

Background Modeling

This approach defines the foreground regions with respect to a background model: regions which do not correspond to the background model are considered as foreground regions, or *agents-of-interest*. The main issue is to determine what is background and what is foreground by using an explicit (usually adaptative) background model. So we can keep on locating an actor all the time, even if it is motionless. Consequently, there are moving regions, such as the leaves of a tree due to wind, which can be modeled as background, and motionless regions, such as an actor stopped in front of a traffic light, which can be detected as foreground. However, this approach is very sensitive to clutter, shadows, illumination conditions, and physical changes in the scene.

Most referred publications use a background modeling-based approach [47, 63, 107, 133, 143]. For example, a probabilistic color-based background model is used in *Pfinder*, developed by Wren et al. [143]. An indoor scene is modeled as a textured surface, where each background pixel is represented using low-order statistics, by means of a mean color value and a Gaussian distribution about this mean. It is assumed that a discrepancy with the background is due to a human. However, small changes in the background pixels color values are allowed, and only one actor, in an upright standing posture, is assumed to be in the scene at a time.

Haritaoglu et al. [62] presented W^4, a real-time system that locates and tracks people in outdoor scenes. Like Pfinder, W^4 uses a statistical background model. Unlike Pfinder, which uses a single Gaussian distribution of color for each pixel, W^4 uses a bimodal distribution of intensity for each pixel. Considering a static

background, the scene is modeled by computing three images: two consisting of the maximum and minimum intensity values for each pixel of the image during a training period, and the third one consisting of the maximum difference between consecutive frames. As a result, foreground regions are represented as blobs of neighboring pixels.

Another method is described by Oliver et al. [107], called *Eigenbackground*. Unlike Pfinder and W^4, foreground regions are required to be small with respect to the field of view. First, the mean background image and its covariance matrix is calculated. After applying Principal Component Analysis (PCA), the background model is represented by the set of largest eigenvalue eigenvectors. Unlike the previous approaches, which computes a distribution for each pixel, the scene model corresponds to the entire background image. Thus, small movements of the camera and the background are well described. Pixels corresponding to moving regions do not have a significant contribution to the background model, since they do not appear at the same location in the image, and they are typically small.

Stauffer et al. in [133] described an adaptive background model, which is an extension of that used in *Pfinder*. The intensity or color values that each pixel exhibits along several frames is modeled by means of an adaptive *Mixture of Gaussians* distributions (MoG). Using an adaptation procedure, an actor will be included in the background model if it remains motionless long enough, and it will be excluded if it moves. However, how fast the model adapts to changes needs to be determined to obtain high detection sensitivity[3].

In order to achieve high sensitive detection using a model which adapts quickly to changes, Elgammal et al. in [47] proposed a non-parametric background model which is well suited for outdoor scenes. The background is not required to be perfectly static, and shadow suppression is handled. As in [133], the model considers the most recent pixel values to compute a statistical model. But here, density estimation is achieved by using a Normal kernel function, which is a generalization of a MoG. However, several heuristics are established to handle false positives and false negatives. Also, this approach requires considerably memory resources and processing time.

Summarizing, approaches presented in this section can be used to build a movement model represented as a consecutive pair of segmented agents. Therefore, such a model can be extended over time to build an action model represented as a sequence of segmented postures.

Motion Detection

Motion-based approaches only pay attention to agents in motion, so motionless humans can not be detected. This approach is consistent with the idea that surveillance systems is only interested in what is happening in the scene by means of event analysis, where the basis of event analysis is change detection. Commonly, change detection is assumed to be due to motion, and is segmented by means of temporal differencing

[3]If the background model adapts too slowly to changes, an inaccurate model will be built. On the other hand, if the model adapts too quickly, the target will be included in the model.

or optical flow analysis.

Temporal differencing can be considered as the simplest way to apply change detection, and it can achieve real-time performance. This approach uses pixel-wise differences between two [87] or three [118] consecutive image frames. However, only motion regions are extracted instead of the whole moving object. Thus, it is difficult to extract agent features from these motion regions. Also, since the difference between two frames is sensitive to noise, false positives cannot be suppressed effectively. In [87], for example, temporal differencing is used in outdoor scenes. The absolute difference between two consecutive images is calculated, and a threshold function determines movement segmentation. A connected component criterion is used to cluster moving sections into moving regions. However, this approximation has a very limited ability to represent variations in the human posture.

On the other hand, optical flow analysis describes coherent motion of points of image features between frames; for a review, see [13, 95]. Thus, the characteristics of flow vectors of moving objects facilitate the segregation of objects from the background. For example, Bregler in [30] represents each pixel by its optical flow. Afterwards, blobs are built by grouping pixels with similar flow vectors, thus obtaining blobs having coherent motion. Each blob is represented as a mixture of multivariate Gaussians, where each single Gaussian encodes the coherent motion of that blob.

An advantage of optical flow algorithms is that moving agents are segmented even with camera motion. However, it is not feasible for nonrigid object extraction since the movements of the body parts are different. Also, optical flow is difficult to be applied in real-time domain, due to its expensive computation. Furthermore, in some situations, optical flow analysis is very unstable: due to the aperture problem, only the normal flow (the component parallel to the gradient) can be computed.

Optical flow is actually used in different feature-based tracking papers for motion correspondence, due to the fact that algorithms based on 2-D features, such as corners, do not suffer from the aperture problem. Therefore, optical flow is more appropriate to obtain suitable motion descriptions of image features which can be tracked and learnt, rather than solely classifying each pixel into motion or motionless regions.

Summarizing, motion information is used to detect predetermined image features of the human posture, upon which more complex human motion models are built. Note that these features should embed facilities for human tracking or human action modeling [31, 48, 68, 81, 122, 146].

2.1.2 Human Tracking

A *movement* is defined as a noticeable change of human posture or place between consecutive frames. As described in the preceding section, human movements can be represented using two consecutive human posture, or a motion characterization of predetermined features of the human posture, such as the centroid, the silhouette, or the joints of a predefined stick figure, for example.

However, segmentation errors can lead to an inaccurate movement description. Errors due to image noise and partial occlusions of the actor change, possibly in an extreme way, the observed human posture. Even worst, the actor can be completely occluded due to static components of the scene or foreground objects. In these cases, the parameters of the movement model are updated considering erroneous or incomplete image information.

Consequently, tracking procedures are incorporated to perform movement segmentation robustly. The goal is to process image data, called *measurements*, in order to obtain a minimum error estimate of the state of an agent by utilizing a dynamical model, predefined initial conditions, and assumed statistics of system noises and measurement errors. This tracking process can be considered as an estimation algorithm over time, also called *temporal filtering*. Thus, it is possible to maintain the parameters of the human body model when measurements are unreliable or unavailable. The advantage of such estimators relies in the minimization of the estimation error in a well defined statistical sense. Also, they utilize all measurement data plus prior knowledge about the system. Potential disadvantages of these methods are their sensitivity to erroneous a-priori models, and the inherent computational burden.

Several papers already review different human tracking procedures by considering different geometrical or methodological criteria [1, 32, 33, 51, 96, 140, 141]. Alternatively, the aim of this section is centered on describing which knowledge is usually pre-determined in the literature to perform tracking.

Knowledge for Human Tracking

In recent years, novel tracking techniques are defined based on an hypothesis/validation principle [39]. Consequently, the tracking process is described using a probabilistic scheme, which is based on the Bayes' rule [30, 73, 130, 134]. The goal is to estimate the *state of the agent* using a combination of hypothesis generation and validation. The generation of new hypotheses is modeled using a prior probability distribution called the *human motion model* hereafter, and the validation of such hypotheses is attained using *likelihood functions*.

Consequently, tracking algorithms require to define [12]: the state of the agent, which describes the characteristics of the agent being tracked; the human motion model, which provides a prediction cue for the state of the agent at next time step; and the likelihood function, which defines the relationship between the state of the agent and the measurements obtained from input devices, such as cameras. These topics should be carefully designed in order to develop a tracking procedure specific for our problem domain.

The state of the agent is a quantitative vector which corresponds to a predetermined set of features derived from the actor being tracked. Such features should be selected accurately, depending on the functionalities provided by the system: the state of the agent can rely either on a pre-defined human body model, or on low-level image characteristics. That means, there is a trade-off between complex high-level

human body models which supply accurate body posture configurations, and simple low-level human body representations which can be robustly extracted from noisy images.

In any case, the state of the agent can not be determined directly in real-world applications. Instead, a set of *measurements* of the state of the agent is obtained from at least one camera. Thus, tracking is posed as an estimation problem, where the state of the agent is estimated from measurements following two approaches, namely *model-driven* and *data-driven* procedures [51]. The former refers to a predefined body model which determines the image characteristics to be tracked. The latter refers to predefined image features and heuristics which represent the actor. Methodologically speaking, model-driven and data-driven methods can be referred to as top-down and bottom-up approaches, respectively [147]. Both approaches perform a correspondence measure between measurements and the state of the agent using the prediction stemmed from the motion model.

The human motion model describes how the agent is moving within the scene, thus allowing to estimate the next expected position and/or configuration of the human body at next time step[4]. The goal is to reduce the set of plausible configurations of the agent state at next time step, once we have estimated the current agent state from image measurements.

Commonly, constant velocity and acceleration is assumed, and the human motion model is implemented in discrete form through difference equations. Using probabilistic terms, these linear dynamical models can be expressed by means of Gaussian densities. The parameters of such densities are determined during a training step to describe the motion of a given agent. This learning process can be implemented off-line, that is, the parameters of the motion model are determined beforehand to improve the tracking performance. Alternatively, the human motion model can be learnt on-line by means of automatic learning procedures, for example by using the principle of *maximum likelihood* [17]. Both learning processes search for the optimum parameters which maximize a predefined likelihood function.

Likelihood functions relate the motion model with the state parameters of the agent. In fact, these functions can be considered as probability measures, i.e., the probability that a given state generates a given measurement. This correspondence measure *corrects* the predicted state obtained from the human motion model. Therefore, likelihood functions are determined by the complexity of the state of the agent and the efficiency of the movement segmentation step.

In data-driven approaches, the validation step is attained by using correlation, distance measures, or suitable objective functions. Likewise, in model-driven approaches, the correspondence between measurements and the state is achieved by means of a matching procedure between image data and model data. Therefore, low-level human body models are easier to match to image features, and high-level human body models are more difficult to be compared with image features [1].

[4]However, in some domains, actors move so erratically that motion is impossible to be modeled [72]. In such cases, a suitable alternative is *data association* [12].

Human Body Models for Video Surveillance

Depending on the application, the human body is represented based on different criteria. We consider here that such a model is driven to assist video surveillance applications, that is to say, the interesting parameters of the body model should allow to infer the presence or non-presence of human actors, the spatial position of detected actors, and the separation of individuals in the case of multiple humans tracking. Therefore, tracking requires of a compact representation adequate enough to deal with locating errors due to occlusions, background clutter and noise.

Commonly, video surveillance does not require an accurate representation of the human agent, because each additional parameter of the body model increases the complexity of the tracker. Consequently, data-driven human body models are usually considered. Based on the final purpose of surveillance systems, the spatial information about each agent is well-represented using simple image features, such as points, lines, or regions. However, it is possible to enhance these models with pixel characteristics, such as color [92, 114], appearance [18, 137], or even their motion [30]. Also, heuristics can help to determine the agent posture or even its body parts [63]. These low-level descriptions can be extracted robustly from noisy and incomplete data.

Most popular representations are blobs [30, 74, 86, 143] or sub-features of blobs, such as the centroid [111], median [63] or bounding box [104]. Blobs are constructed by grouping pixels having similar characteristics, such as color or coherent flow[5].

In [111], pixels corresponding to independently moving objects are detected. These pixels are then grouped using spatial clustering methods. Tracking is performed using the centroid of the resulting moving blob. As the centroid is affected by the large motions of the extremities, the height of the agent is assumed to be constant. This approach assume only one moving object in the image. Motion field between successive frames is used to segment and track the actor, detect scale changes and compensate for translation and scaling. In [63], the median of the resulting segmented blob is tracked, which is not affected by large motions of the extremities which tend to influence the centroid significantly. In [120], two points are the features selected for tracking, which correspond to the two opposite corner of the blob's bounding box. Wren et al. [143] represent the human body as a set of blobs, each one described in terms of spatial and color information. The actor model represents the global aspects of the shape, and is built by identifying the head, hands and feet locations using 2-D contour shape analysis. A initialization step is required.

Note that low-level features can not be used to analyze internal features of the actor, so neither body parts can be tracked nor body pose can be determined accurately. In order to describe motion patterns with communicative purposes, model-driven procedures are used instead, which require of a predefined synthetic model, such as stick figures, 2-D contours, and volumetric models [1, 141]. Usually, a initialization step is required, and the aim is to maintain and update the parameters of the human body model using image features. Therefore, a matching process is required to compare

[5]This procedure is inspired in the investigation about the human visual system performed by the Gestalt school in the 1930s [82].

how well the body model fits the human tracked in the image.

Errors due to locating and tracking processes are commonly estimated and corrected without considering the actual behavior of the actor. However, once an action is recognized, the movement information about such an action could help to determine the next expected body posture configuration, thus reducing the search space of the tracking algorithm [19].

2.2 Action Descriptions: Analysis from Image Sequences

We define a human action as a learnt sequence of human movements. The term *learnt* denotes the process during which a predefined sequence of movements is semantically associated to a *verb*[6]. So human actions are described using motion verbs, such as *walking*, *jumping*, or *bending*. Thus, image sequences can be compared against learnt motion patterns to describe which verb is being observed. Consequently, human actions constitute the semantic basis for motion understanding.

In the literature, an action is commonly determined beforehand, that means, *supervised* learning procedures are used. Therefore, during a learning stage, the sequence of frames which constitutes a given action is exactly determined[7]. Movement patterns are extracted from that sequence, and processed afterwards to develop a suitable action model. The term *suitability* is defined with respect to predefined criteria, such as:

Compactness : to require few parameters to describe each action, thus improving the performance and speed of human action evaluation algorithms.

Accuracy : to include any movement which is plausible to appear in any performance of a given action. Accuracy is often poor where not enough training data is provided.

Specificity : to exclude those movements which are not likely to appear during a learnt action.

Temporal invariance : to be invariant to nonuniform changes in the speed of any action performance to be learnt.

Order-preserving : to include the order in which the variation of the movement has been observed during an action performance.

[6] Verbs are the most natural, best-understood and most readily accessible way for communicating motion and time-varying concepts [136].

[7] When considering cyclic actions such as running or walking for example, repetitive motion detection can help to determine the sequence of frames which constitutes the action to be learnt [4, 30, 35, 111, 128].

So the aim is to model properly the variation of the human movement, but also to design a human action representation useful for HSE purposes. That means that, on the one hand, the action model should be instantiated from low-level image cues. But also, the action model should be useful for high-level tasks, such as motion understanding, human synthesis, or performance analysis, for example.

HSE requires human action models with facilities for motion understanding, which involves both human action recognition and description. That means, the action model should maximize existing dissimilarities between their constituents movements. Also, a comparison measure between action descriptions should be defined to implement human action classification. Such a comparison depends on the type of features used to describe motion patterns, and is usually implemented by means of distance calculations between an action model and an unknown sequence.

Moreover, HSE requires human action descriptions with facilities to motion synthesis and performance analysis. That means, the human action description should provide both anatomical meaning and accurate measurements. Therefore, video sequences involving virtual agents can be generated, during which human action descriptions are re-visualized for evaluation purposes. Also, the accurate analysis of a particular performance should be addressed to determine "how well" a performance has been executed with respect to a reference action model, and to derive action characteristics such as the style, for example.

As commented before, an action model can be built using a sequence of either motion measurements or human postures. Next, we detail different human body models which are used in the literature for human action modeling. Subsequently, we survey different human action models by analyzing the previously described requirements.

2.2.1 Human Body Models for Motion Understanding

As commented in [33], "to properly study human motion, good body models must be defined". In fact, human motion is divided into two components: *absolute* motion and *relative* motion. Absolute motion is defined as the global movement of the human body with respect to the observer, also called *locomotion*. On the other hand, relative motion is the movement of one limb with respect to other limbs of the human body, which is the basis for human motion understanding.

Data-driven human body models are suitable to derive the absolute motion description of an observed agent. Such models are built based on low-level image features, such as points [63, 105, 111, 120, 127] and blobs [66, 72, 114, 143, 147]. But this set of parameters can be extended over time to model actions. Thus, the temporal evolution of human movements can be represented by means of motion trajectories. However, image features do not provide rich descriptions of the human body for relative motion analysis, though their extraction from images is robust.

In order to analyze the relative motion of an actor, model-driven procedures are best suited, due to the fact that they are based on high-level body models. Considering these models, the goal of tracking is to recover the full-body pose of the agent. Most

referred models are those based on stick figures [2, 21, 35, 45, 49, 59, 80, 110, 144], 2-D contours [63, 76, 85, 106, 118, 145], and volumetric models [15, 34, 52, 69, 77, 119, 130]. Consequently, human body segments can be represented using lines, 2-D ribbons, and 3-D volumes, respectively.

The stick figure representation consists of segments connected at their endpoints, and is defined in terms of a set of joints, usually grouped in a hierarchical manner. The characteristics of stick figures can be summarized as follows:

- The stick figure reflects the anatomical characteristics of the human body. Thus, facilities for action synthesis are embedded.
- Human body parts can be tracked separately by means of the joints. Angle predictions and constraints of individual joints can simplify the search space of tracking procedures.
- The human action can be represented as a sequence of postures, as motion trajectories of body parts, or as the variation of joint angles over time.
- Depth information can be determined. The stick figure can be set in correspondence with a more realistic and complex three-dimensional human body model. Thus, three-dimensional tracking can be implemented.
- The human body is modeled using a reduced number of parameters.
- A comparison measure between different human body poses can be established, due to the hierarchical nature of the stick figure representation.

The stick figure can be built from image data by means of skeletonization techniques. For example, Fujiyoshi et al. selected in [49] a body representation based on a real-time skeletonization process. Once a foreground object is detected, its centroid and boundary is computed. The centroid is assumed to lie inside the object boundary. Subsequently, a function is applied to each border point of the boundary to determine the distance from the centroid. Local maxima correspond to extremal points of the boundary, and the skeleton is built by connecting them to the centroid.

Stick figures can be defined a priori, too, by determining the number of joints and their degrees-of-freedom (DOF) considered for modeling. Cheng and Moura in [35], for example, represented the human body as a stick figure composed of 12 rigid body parts (head, torso, plus two primitives of arms and three primitives of legs), which is built in a hierarchical manner. As their paper is focused on human walking, kinematic and physical constraints are applied to reduce the human model to 14 DOF, and dynamic constraints of human walking restrict the search space of plausible motion parameters. Tracking of body parts is implemented using a gradient-based method. Furthermore, it is assumed that motion is restricted to a planar walking motion. The shape, texture and motion of the human are recovered for synthesis purposes.

On the other hand, 2-D contours are set to correspond to the human body projection in the image plane, so body models are restricted to the camera's angle. For

example, a cardboard body model is proposed in [76], in which the limbs of the human are represented by a set of connected planar patches. The motion of these patches are parameterized to deal with the analysis of articulated motion of human limbs. In [85], the outline of the human body is found using coincidence edges, which are edges common to an edge image and a difference image. Structural constraints, such as lengths and relations between different joints, are introduced and combined with shape constraints. A B-spline is used in [14] to represent the shape of the agent. Shape constraints are introduced, and a set of Kalman filters are used to predict and smooth the different body model parameters. Davis et al. [42] use a similar technique: each extracted contour is parameterized using a B-spline. Tracking is implemented using the Condensation algorithm, which is based on a template hierarchy, and the likelihood function is based on a multi-feature distance algorithm.

Volumetric models can be modeled as elliptical cylinders [31, 69, 119, 130], cones [43, 139], or super-quadrics [52, 77]; see [96] for a detailed review. In fact, complex 3-D volumetric models provide an accurate representation of the body pose, which achieves independence from the viewpoint, and handles self-occlusions and self-collisions of limbs.

However, volumetric models involve complex and expensive reconstruction algorithms [143], so kinematic and shape constraints of the human body are highly exploited. In fact, pose recovery is very unreliable and difficult to perform, so body model parameters are usually derived from image features. Therefore, a matching procedure is required to determine how well the model fits the human agent in the image [26, 35, 80, 110, 129]. This comparison measure is usually performed by means of a matching process between the three-dimensional model of synthesized data, and a suitable two-dimensional representation of image data. The goal is to maintain and update the parameters of the human body model at each time step.

Data-driven models are used to assist the instantiation of the synthetic model, too. In [59], Guo et al. compute the skeleton of the segmented agent, which is matched against a stick figure using an energy function. A potential field is introduced to reduce the complexity of the problem. Angle constraints of individual joints are used for prediction. In [69], the boundaries of the projected model limbs were compared to extracted edges in the image. Hard constraints on the motion of the model are determined. Rohr [119] presented a similar approach, but Kalman filters where used instead for estimating the pose between time steps. Edges are also considered in [52] where four calibrated cameras were used to image-model comparison. Wachter and Nagel [139] exploited edge and region information of the image to establish a matching process based on the iterative extended Kalman filter.

Model-driven approaches can incorporate known physical constraints of limbs and extremities of the body to perform location and tracking. In fact, the parts of the human body can move in quite independent ways, but human actions, such as walking, running, etc., are very constrained by factors including motion symmetries or dynamics. So they can be based on physics heuristics. That means, physical restrictions about dynamics and human structure can be easily embedded in such models to restrict the set of plausible postures that an agent can exhibit.

2.2.2 Human Action Modeling

As commented before, the basis of motion understanding is action recognition and description. Therefore, suitable models for human actions should be developed in order to recognize predefined motion patterns, and to produce high-level descriptions for the observed action. Typically, action recognition is posed as a time-varying data matching problem. Therefore, in order to deal with the inherent temporal and spatial variability of human action performances, suitable analytical methods have been used in the literature for matching time-varying data. Most referred algorithms are Dynamic Time Warping, Hidden Markov Models and Neural Networks [33, 141]. Such procedures model how an action description varies during a performance. In any case, the recognition performance relies mainly on the features used for action modeling.

Human action models can be built using two strategies, namely, *motion-based* or *appearance-based* approaches [22]. On the one hand, motion-based algorithms characterize the motion itself by encoding the variation over time of selected low-level image characteristics in a feature vector [33]. Such a vector can correspond to parameterized trajectories of tracked features over time, or to temporal templates built from image data and matched against stored models of known actions. Therefore, no pose reconstruction is required. On the other hand, appearance-based approaches describe motion patterns by considering the sequence of human postures exhibited during several action performances. Thus, the aim is to recover the posture of the human body using a pre-defined human body model, which is usually different from that used for whole-body tracking: anatomical information of the agent is also modeled, which is useful for synthesis and analysis purposes.

Human Action Representations based on Motion

Early research on Moving Light Display (MLD) [75] experiments suggested that many human gestures could be recognized solely by motion information. Therefore, motion-based action models require of motion extraction[8]. This process can be achieved by using two methods [33]: motion correspondence and optical flow. On the one hand, motion correspondence refers to characteristic features, which are tracked over time. On the other hand, techniques like optical flow and image differencing, which compute the displacement of each pixel between frames, are applied to extract human motion information.

Motion correspondence derives a trajectory from tracked image characteristics. Therefore, two steps are involved: the detection of predefined tokens in each frame, and the correspondence of such tokens from one frame to another. Suitable tokens are those which are distinctive enough for easy detection, and stable enough over time so they can be tracked. Examples of tokens are edges, corners, points and regions. The sequence of locations of such tokens can be represented as a motion trajectory, that

[8] In order to represent the relative motion of the human body, incidental motion is removed, such as camera motion and agent locomotion [22].

means, a vector valued function which describes the token location over time. This problem is computationally expensive, so several heuristics are introduced to limit the search space. Such qualitative constraints are usually transformed into cost functions in order to quantitatively determine optimal trajectories.

As parameterization of trajectories is better suited for computations, temporal trajectories are parameterized in the literature in terms of speed and direction, velocity, and spatiotemporal curvature [4, 23, 57, 115, 116, 118]. Mainly, parameterization of trajectories has been applied to gesture representation, but its extension to human action modeling is straightforward. Bobick and Wilson [23] model a gesture as a trajectory in $X \times Y \times T$ space, which is derived from a sequence of spatial hand positions extracted over time. Thus, once several training gesture trajectories have been computed, a *prototype curve* is derived by removing time as an axis, and by preserving the temporal coherence of the trajectory. Such a prototype curve, which is parameterized by arc length, is clustered using the *k-means* algorithm. By using a suitable distance metric, the cluster centers, called *states*, correspond to those points of the curve with most relevant variance of the gesture, independently of the temporal ordering. Consequently, the gesture is modeled as a sequence of such states. Rao and Shah in [116] model actions performed by a hand, based on spatiotemporal curvature of a trajectory. For each frame, the hand is located using skin detection, and its centroid is computed. Subsequently, the trajectory is built by joining the centroids of the hand of each frame. Next, *dynamic instants*, which correspond to important changes in motion characteristics (such as speed, direction, acceleration and curvature), are identified in the trajectory. The number and locations of such instants are exploited in both the learning and recognition processes. And this can lead to ambiguity due to the differences in the knowledge embedded in both models.

Alternatively, two dimensional motion information derived from optical flow analysis can be used to build more elaborate representations [22, 30, 76, 88, 112, 121]. In [22], Bobick et al. model 18 aerobic exercises by considering the temporal history of motion at each pixel. Specifically, a temporal template, which is composed of two components, is built by image differencing for a given viewpoint and a given action. Such components represent, on the one hand, where there is motion (called *Motion-Energy Image* or MEI) and, on the other hand, how motion is moving (called *Motion-History Image* or MHI), which implicitly represents the direction of motion. Both images are computed recursively, and used jointly to improve the recognition rate. Duration of each aerobic exercise is computed using a backward-looking variable time window. Ju et al. in [76] represent the walking action using 2-D parameterized models of optical flow. Human agents are represented using cardboard models, in which each limb of the model is described by a planar patch. Image motion of each rigid planar patch is determined through energy functions and represented using temporal curves. Subsequently, a *characteristic curve* is created for each action and body part viewed at a given angle using PCA. Thus, the dominant curve components are captured, which is the basis for action recognition. Polana and Nelson [112] compute the optical flow field between consecutive frames. Periodic human motion is assumed, thus allowing to model a single cycle for each action to be learnt. An action is represented as a spatio-temporal solid composed of $X \times Y \times T$ cells by partitioning the two

spatial dimensions into X, Y divisions, and the temporal dimension into T divisions. For each cell, the summed normal flow magnitude is computed. The resulting motion magnitude values of each cell constitute the high-dimensional vector used for action recognition.

Typically, motion-based approaches exhibit low computational complexity and simple implementation for action recognition procedures. However, the variations of the time interval of the movements influence the resulting action description. Furthermore, these action representations do not embed facilities to synthesis.

Human Action Representations based on Appearance

Appearance-based action models are built upon the sequence of static attitudes which an agent exhibits while performing a particular action. Such a sequence embeds the set of human body model configurations exhibited during a predetermined number of performances of the action to be modeled. Therefore, both a suitable representation for each human posture, and a procedure for the parameterization of the resulting sequence should be determined.

As discussed before, each body pose can be represented following two approaches, namely *data-driven* and *model-driven* procedures. The former refers to predefined image features and heuristics which represent the body model. The latter refers to a predefined body model which determines the image characteristics to be tracked. This differentiation is also found when reviewing human action models, too.

On the one hand, each static posture model can be built directly from low-level image features, such as pixel appearance [40], skeletons [3, 46, 49], body silhouettes [63, 78, 91, 147] and body contours [2].

For example, each human posture is modeled using skeletonization in [3]. Considering lateral view, each skeleton is then described in terms of three angles (torso, upper component of the leg and lower component of the leg). Therefore, an action is represented as a sequence of skeleton curves, that is, a sequence of three-element feature vectors.

Haritaoglu et al. [63] use temporal texture templates for action representation. Specifically, each action is represented as the average normalized horizontal and vertical projection templates computed from 4500 silhouettes of 7 people in 3 different views. Transitions between different actions are misclassified. Yamato et al. in [147] model an action as a sequence of feature vectors. Each feature vector is built as follows: given an image, the agent is first segmented. The resulting binarized image is divided into meshes. Each mesh size is related to the size of the agent, and the ratio of pixels corresponding to the agent in each mesh becomes an element of the feature vector. Subsequently, the feature vector sequence is transformed into a symbol sequence by vector quantization. This symbol sequence is used afterwards for human action learning and recognition.

An approach similar to that described by Bobick in [22] is presented in [91], where

motion information is calculated using an *infinite impulse response* filter. The idea is to represent motion by its recency: for each image of a given sequence, recent motion is represented brighter than older motion. Unlike [22], an action is represented as a sequence of normalized feature images, each image capturing the temporal changes in the sequence up to that image. Subsequently, each action is represented as a manifold in an eigenspace, whose points correspond to the different feature images the action goes through. By applying PCA, there is a reduction of dimensionality for each feature image to be learnt. Also, representing an action as a manifold provides better discrimination than modeling the action as a single point in eigenspace.

Human Action Representations based on Body Models

Alternatively, predefined human body models can be used instead, for example by means of stick figures or ribbons [15, 35, 59, 85]. By providing a synthetic body model, anatomical information and kinematic constraints are incorporated to the action model, thus allowing to derive rich and useful human motion descriptions. Such descriptions can be driven to assist different tasks of HSE systems, such as the tracking of limbs, the synthesis of motion, and the analysis of performances.

In the literature, most human action models are exclusively driven to perform action recognition. Due to the specificity of the task to be achieved, i.e., classification, the action model embeds characteristics which are discriminative enough for the set of actions to be recognized. However, as the qualitative paradigm points out, is reconstruction really necessary for human action recognition? [5]. That means, by representing an action as a sequence of body model configurations, additional issues need to be addressed, such as the segmentation of the agent, and, commonly, the identification of the individual body parts.

But, if only motion characteristics are embedded in the action model, we limit our HSE system to a specific application domain, namely action recognition. However, anatomical characteristics should be incorporated in the body model to perform HSE which can demand, additionally, action synthesis and performance analysis. For example, the stick model described by Cheng and Moura in [35] is used for multimedia applications, such as synthesis, video manipulation, or indexing.

Consequently, by representing an action as a sequence of parameter values of a predefined human body model, anatomical features can be included in the action representation, thus allowing to re-visualize a recognized action. For this purpose, we can consider different types of action models, depending on the task to be performed, for example one action model for recognition and one action model for synthesis. However, this approximation implies to establish a transformation procedure between both action representations, in order to *interconnect* both tasks.

Alternatively, we can establish a generic action model which can be used for different application domains, like the human body model presented in this Thesis. Ben-Aire et al. represent an action in [15] by a sequence of pose and velocity vectors for the major body parts, namely arms, legs and torso. The poses of the body

parts are described in terms of normalized angles to achieve invariance to body size. Subsequently, each body part representation is stored in a different hash table, thus enabling action recognition even when several body parts are occluded. As discussed before, Cheng and Moura [35] represent the walking action as a sequence of human postures. Thus, considering the average joint angle time series of several walking performances, human body part tracking is implemented using a template matching technique. In [59], human motion patterns are represented as a sequence of stick figure configurations obtained from the tracking process. Each stick figure vector embeds information about the length of the limbs and joint angles. In order to segment the duration of a given motion pattern, temporal periodicity is exploited. Thus, only periodic motions are modeled, specifically walking and running.

In the literature, video sequences used for action modeling are usually restricted to a very limited number of actions, performances, and agents. In fact, current research on action modeling is just in its infancy [141]. And, since human motion understanding highly relies on the performance of the human action recognition algorithm, procedures for the semantic interpretation of human behaviors are usually defined in highly constrained domains.

2.3 Activity Descriptions: Transitions between Actions

As commented in the last section, human action analysis is the basis for motion understanding. Human actions are associated to verbs, such as *walking*, *jumping*, or *bending*, so basic conceptual descriptions can be derived. These descriptions are combined/enhanced with the knowledge about the context in order to perform behavior *evaluation* or, in other words, deep reasoning about the behavior and intention of observed agents.

Consequently, there is an explicit relationship between two different fields of research, namely, pattern analysis and artificial intelligence. On the one hand, pattern analysis applied to the computer vision domain allows to develop different methodologies devoted to build a quantitative representation of human agents within the scene, by using the information obtained from video cameras. On the other hand, artificial intelligence allows to develop different methodologies devoted to instantiate and manipulate conceptual predicates for human motion interpretation and reasoning [123]. As a result, we can qualitatively evaluate what is happening in the scene.

As discussed in chapter 1, a critical issue of HSE is then centered on the *integration* process, that is to say, how to associate the quantitative information obtained from pattern analysis procedures, with the conceptual descriptions used for reasoning. Due to integration, the numerical information obtained from image sequences can be used to infer the behavior of an observed agent and, the other way round, conceptual knowledge can be used to assist action analysis processes.

We introduce integration by first discussing about the concept of *activity*, defined

as a sequence of consecutive actions. In fact, human activities embed characteristics of both action and behavior definitions. On the one hand, each component of a human activity is a discrete action primitive. On the other hand, a human activity also refers to the transition between consecutive actions. Therefore, conceptual knowledge is incorporated to establish a procedure which defines the transformation of the state parameters of the human agent in order to represent not only the complete sequence of actions predefined in the activity, but their transitions.

Most research in computer vision is centered on single action evaluation, and the analysis of continuous activity sequences is still a very recent domain. That means, activities are not considered as feasible motion patterns to be learnt.

At present, most interest in activity evaluation has been found in papers related to graphics and animation techniques [10, 93, 108]. In fact, advances in graphics hardware and rendering software has allowed to build visually rich worlds, creating possibilities for activity synthesis and analysis.

The creation of synthetic activities exhibiting realistic motion is a challenging issue. Current applications assemble static clips of motion created with traditional animation techniques such as motion capture or keyframing. Therefore, the assembly process requires making transitions between motions. These transitions may be difficult to create, such as a transition between a running and a lying-down action, or trivial, if the end of an action is identical to the beginning of the next action.

A sequence of actions and transitions between them can be represented as a graph where the edges are pieces of motions, and the nodes are choice points connecting motions, also called keyframes. A graph of this type is common in computer games, and is called *motion graph* [8, 84] or *move tree* [93]. Move trees are constructed by pre-planning actions with similar beginning and end keyframes. An animator then chooses the exact keyframes where transitions are to occur, and creates the transition motions.

The computer animation literature provides several procedures to build move trees. On the one hand, by specifying both an initial and a final body posture configuration, the sequence of intermediate body poses can be automatically found by using *interpolation* techniques [142]. In fact, the foundation of most animation is the interpolation of values, which is used when the animator directly controls how the object moves. On the other hand, more sophisticated techniques can be applied to generate motion with a desired degree of physical realism [67, 89]. In this case, the physical characteristics of the human body help to capture and generate realistic motion by considering several anatomical constraints and rules [54]. Note, therefore, that activity modeling involves a trade off between the required level of user-interactivity, and the physical realism obtained. These two issue are addressed next.

2.3.1 Creating Motion using Interpolation Techniques

Transitions between actions can be posed as an approximation problem, in which the state parameters of the human agent are interpolated to reach smoothly to a certain

posture of another action [55]. For this purpose, the term *key-frame* is applied: a key-frame is a variable whose value is set at a specific key frame, and from which values for the intermediate frames are interpolated according to a predefined procedure.

We assume hereafter that the human body consists of rigid segments connected at their endpoints, and is modeled in terms of a set of joints, usually grouped in a hierarchical manner. Given an activity model, a key-frame is set to correspond to the last human posture of a given action, and another key-frame refers to the body configuration of the initial posture of the next action in temporal order. Interpolation of the joint parameters of both key-frames will generate the sequence of intermediate body postures, or *in-betweens*.

Several issues need to be addressed to implement an interpolation procedure [108]:

Control points : constitute the set of spatial coordinates which describes the curve. Control points are used to compute a curve which pass through these points, or rather they are used to control the shape of the curve, so control points are approximated. The resulting curves are usually implemented using splines, such as Hermite splines or Cubic B-splines.

Complexity : refers to the underlying equations of the interpolating function. Typically, cubic polynomials provide smoothness while satisfying the constraints imposed by the control points. A polynomial whose order is lower than cubic may not fit smoothly the control points. A higher polynomial order does not provide any significant advantages and is more costly to evaluate.

Continuity : refers to the number of derivatives of the curve equations, which are continuous. Zeroth, first, and second-order continuity refers to the positional, tangential and curvature continuity of the values of the curve, respectively. Typically, first-order continuity at the control points guarantees the smoothness of the resulting curve.

Once the joint trajectories establish the sequence of spatial positions which drive the body to execute smoothly the transition between two consecutive actions, the *speed* at which the curve is traced out should be under control, too. For this purpose, the resulting trajectory curves are commonly parameterized to *arc length*, that is, the distance along the curve of interpolation.

Different procedures can be chosen to perform such a parameterization. For example, a length function can be computed analytically. However, as the relationship between the parameterizing variable and the arc length is usually non linear, there exists no analytical solution. Instead, a table of values can be created, which establishes the relationship between the parametric value and the arc length at several *sampled* points of the interpolated curve.

Once the curve is parameterized by arc length, it is possible to control the speed at which interpolating curves are traversed by means of *distance-time* functions. Such functions relates arc length values to time, which is independent of the form of the

curve itself. Therefore, we can generate a posture transition between two actions, which starts quickly, then continues slowly, and ends again quickly, for example.

Note that no anatomical information of the human body has been considered up to this point. Consequently, interpolation may produce non human-like figures while generating a transition between very different actions. To avoid these results, physical restrictions can be embedded into the body model in order to restrict the set of allowed configurations for the human body pose, as described next.

2.3.2 Creating Motion using Physical Constraints

Action transitions can be posed as a transformation problem, in which the human body model should adopt a sequence of body configurations in order to reach realistically to another posture. Incorporating physical knowledge to the body model allows to guide the motion of articulated figures in a more detailed and realistic way than using interpolation [108]. Such knowledge can generate movements due to forces, and avoid self-collisions, for example. Physical constraints are applied to structured body models, and allow to automatically establish *rules of motion*.

Rules of motion are defined by suitable *kinematic* and *dynamic* models. A kinematic model establishes human movements without considering the forces involved in motion [148]. In addition, dynamic models compute the underlying forces which produce a desired movement, such as gravity, friction, or collision, for example [10]. Reactions of the body to such forces are established automatically by a set of *equations of motion*, thus obtaining physically-based activities.

The human body model is commonly represented as a hierarchical set of linked rigid segments which correspond to the limbs of the body. Specifically, the body model is described using a tree-like structure, where the nodes refer to the joint angles of the body limbs, and the arcs establishes the hierarchical relationship between adjacent limbs. The term *forward kinematics* refers to the evaluation of this hierarchy to produce a human figure which reflects the current setting of the joint parameters. When dealing with complex articulated figures, such as humans, the analytical computation of these intermediate configurations can not be addressed by the interpolation techniques described before. Therefore, the evaluation of the sequence of intermediate body configurations, which drives the body from an initial body posture to the final desired posture, is often a trial-and-error process

To avoid such a tedious task, *inverse kinematics* is used instead [25]: given the initial and ending position of a joint, usually hands or feet, which is called *end effector*, the sequence of joint parameters required to attain the ending configuration are computed for each limb of the body. In this case, motion is incrementally generated by using the inverse or pseudo-inverse of the *Jacobian* matrix, where each term of the Jacobian relates the change of a specific joint to a specific change in the end effector.

Inverse kinematics can generate non human-like motion when multiple body configurations are possible. In fact, inverse kinematics lacks of anatomical restrictions for producing human-like configurations. In addition to inverse kinematics, we can

exploit the fact that the joints of the human body have physical limits. For example, the elbow can bend to approximately 20 degrees and extend to (at most) 160 degrees. More subtly, joint limits vary with the position of other joints, and further limits should be imposed to avoid self-collision of limbs.

When a more realistic activity is required, the underlying forces involved in motion should be incorporated. When applied to humans, forces induce accelerations which produce changes in the joint velocities. These velocities, in turn, produce changes in both the spatial position and the pose configuration of the human agent. The new position and configuration of the human agent give rise to new forces, which are computed at the next time step by means of the equations of motion.

However, as pointed out in [53], animators who build a particular activity often want more control over the motion than a total physical simulation model. Consequently, interpolation techniques (still) forms the foundation upon which most computer animation algorithms take place.

2.4 Behavior Descriptions: Playing with Context

The term *behavior* refers to one or several activities which acquire a meaning in a particular context. That means, the motion characteristics of an observed agent are combined with the knowledge about its environment in order to derive a semantic description about the recognized human action or activity. This is done by developing behavior models which operates in an specific domain, thereby restricting hypothesized interpretations to those that are *reasonable*.

To this end, artificial intelligence and natural language processing algorithms need to be addressed. Consequently, very few human motion analysis papers address the goal of human behavior modeling [141]. Only recently, mathematical models have been used to achieve behavior interpretation from very simply and specific human motion patterns. Specifically, the basis of motion understanding is (still) centered on *actions*, which are described using verbs.

For example, Intille and Bobick propose in [70] an automated annotation system for sport scenes using belief networks. They exploit visual evidences and temporal constraints to report the behaviors of moving players. Remagnino et al. present in [117] an event-based surveillance system which generates text-based descriptions from actions and interactions in 3D scenes. Alternatively, a state-transition model is used for generating textual descriptions of human behavior in [9]. The concept of *key-frame* is applied here to content-based video compression. Also, Kojima et al. addresses in [83] the generation of natural language descriptions of human behaviors. Once the agent is tracked using a model-based procedure, a motion-based human action model is built. The spatial relationship of the human agent with respect to its environment is evaluated to select the most suitable verbs for natural language generation.

Next, we describe the integration process between the agent state obtained from

pattern analysis procedures, and the qualitative predicates which are best-suited for reasoning. Subsequently, we discuss about the role of the context in HSE systems, and the knowledge required for behavior description.

2.4.1 Knowledge Integration

As discussed before, integration refers to the semantic gap between the quantitative information obtained from pattern analysis procedures, and the conceptual descriptions used for reasoning[9]. As a result of integration, numerical information obtained from image sequences can be used to instantiate qualitative predicates and, the other way round, conceptual knowledge can be used to assist pattern analysis processes. The steps required to achieve integration are detailed in this section.

A set of semantic features is first established, which is derived from the (numerical) state of the agent. Suitable features for HSE are determined by the nature of the human agent state's parameters which may refer to dynamical, positional, and postural properties of the actor. For example, motion verbs, such as *accelerating*, could be instantiated by evaluating the history of the spatial and speed parameters of the agent state.

To this end, the state of the agent is designed based on two different criteria: the predefined semantic features about the actor are derived from the numerical information of the agent state and, on the other hand, the agent state is robustly obtained by means of pattern analysis procedures. Complex agent states allow to derive complex semantic features, but they are more difficult to be extracted from raw image sequences.

Since semantic primitives are instantiated using the semantic features of the agent state, the meaning of a behavior becomes more specific as the number of semantic primitives increases. The complete set of semantic primitives is referred to as the *terminology of predicates* of our HSE system, which constitutes the basis of human behavior modeling.

Note that semantic interpretation may lead to *uncertainty*, due to the vagueness of the semantic concepts utilized, and the incompleteness, errors and noise in the agent state's parameters. Therefore, uncertainty prevents of categorizing in a precise way the integration of the agent state. In order to cope with the uncertainty aspects, integration can be *learnt* using a probabilistic framework: PCA and Mixtures of Gaussians (MoG) [98], Belief Networks (BN) [71, 117], and Hidden Markov Models (HMM) [28, 50] provide examples. A survey related to these tools is found in [32]. However, in certain multi-agent domains, the representation of all the possible behaviors using these models may become a quite complex process, and the estimation of the transition probabilities would be unreliable without a large amount of training data. There is also the question about how such tools would respond to a behavior which had not been presented to them before.

[9] Integration is also called the *inverse Hollywood problem* [83].

Alternatively, Fuzzy Metric Temporal Logic (FMTL) also copes with the temporal and uncertainty aspects of integration in a *goal-oriented* manner [126]. The main advantage of FMTL over the previously referred algorithms relies on the promise to support not only the execution, but in addition the diagnosis steps during the continuous development and test of HSE systems [61]. However, automatic learning capabilities are reduced.

Once the terminology is established and the uncertainty is modeled using either a mathematical or logical formalism, the semantic predicates are classified according to several criteria, such as the *specialization relationship* [83], the *semantic nature* [117] or the *temporal order* [71] of such predicates. Likewise, a suitable behavior model should explicitly represent/combine the specialization, temporal, and semantic relationships of its constituent conceptual predicates [101]. For this purpose, the semantic primitives involved in a behavior are organized into hierarchical structures, such as networks [71, 125] or trees [83, 139], which represent the modeled behavior.

To conclude, integration algorithms are incorporated into HSE systems to derive the set of semantic primitives of the behavior model, which are instantiated during an input video sequence. This set of instantiated primitives depends on the information embedded in the agent state, and on the knowledge about the context [125].

2.4.2 Knowledge about the Context

Context is usually defined as the set of all information about the problem domain which does not depend on time. Therefore, context information can provide the knowledge required to generate certain conceptual descriptions about an observed behavior in a particular scene. For example, we can relate the agent state to the scene components, such as predefined static objects of the scene or other actors. Also, the context can help to determine the *goal* of an activity. If the goal is known in advance, the context provides the information required in order to state if the goal has been successfully accomplished.

Here, we denote with the term *context* the set of strategies which attempts to reduce the computational cost of the search processes inherent in motion perception and conceptual reasoning. Therefore, the knowledge about the context is supplied to HSE systems to aid not only motion analysis, but integration. This is done by developing models which operates in an specific domain, thereby restricting a system's hypothesized interpretations to those that are reasonable, given the a-priori knowledge about that domain. Within a domain, many events are possible, but only a much smaller subset is truly likely.

As human behavior analysis should confront a variety of semantic topics, different sources of knowledge should be exploited and integrated. So the main issues are involved in how knowledge is represented, which knowledge is of relevance to the system, and how to incorporate such knowledge to the system. As commented in [61], the challenge is to use uniformly the knowledge of the context, while dealing with different geometrical representations and inference mechanisms.

Suitable context information can be established by considering *attentional* criteria [136]. Attention, defined by competition among hypotheses which chooses the strongest one to focus on, can operate successfully over a complex hierarchy of semantic primitives, and can eliminate a significant amount of search that a non-attentive process might otherwise require. To this end, attentional effects on motion are forms of search optimization.

Based on attention, the knowledge about the context is provided a-priori to assist low-level HSE processes, such as agent segmentation, human body tracking, or human action recognition. For this purpose, context should incorporate different *selection* and *prediction* criteria:

Selection : selection of objects, events, and tasks relevant for the problem domain; selection of detailed sub-regions for analysis; selection of spatial and feature dimensions of interest; and selection of operating parameters for low-level operations.

Prediction : prediction in time about where agents might appear; predicting the temporal sequence of motion patterns; predicting change in appearance over time; predicting the interactions with other actors; and predicting the gravity and other forces.

Consequently, the context provides restrictions and assumptions to infer robustly the agent state parameters from video sequences. Therefore, many authors which develop motion understanding procedures use the context information to solve easily low-level tasks, because these tasks are not the focus of the main contribution.

Moreover, context information can be applied to assist high-level tasks of HSE, such as human behavior reasoning. To this end, knowledge about the context provides HSE systems with conceptual models which characterize the scene, the semantic primitives, the task domain, the agent and its motion, and other *commonsense* knowledge [139]. These models are defined to restrict the vast amount of hypotheses which are plausible given only the state of the agent. Also, the context provides restrictions and assumptions to *reason* about the behavior inferred from quantitative data. All this knowledge should be integrated in the behavior model to generate *comprehensive* behavior descriptions.

For illustrative purposes, we discuss briefly the goals of the knowledge required to assist three particular application domains of HSE, namely, natural language description, synthetic video generation, and system performance evaluation. We refer the reader to [125] for different examples of application domains.

Most referred motion understanding papers generate textual descriptions from observed behavior of humans in a temporal and spatial context [71, 83, 117]. That is to say, natural language sentences are built upon the instantiated primitives of the behavior model. The most suitable sentence, which best explains the conceptual predicates instantiated so far, is generated. In this case, the context should help to derive the relationship between a (recognized) verb and its *gramatical attributes*, such as the subject, the object, and the goal.

Also, behavior models may allow the generation of synthetic video sequences, based on previously generated conceptual descriptions [7]. In fact, this process can be considered as an inversion of the natural language generation process: the behavior model generates a sequence of semantic primitives associated to (time-varying) geometric human body configurations. These body configurations are visualized using standard computer graphic procedures. For this purpose, the context should help to identify the semantic concepts of a particular behavior description in order to derive the semantic primitives.

An important requirement of HSE systems is the *evaluation* of the system performance itself [71, 101, 125]. That is, users can ask the system about the scene, and the system can generate natural language descriptions plus information about the reasoning process. For this purpose, we can combine the knowledge required for the generation of textual descriptions, with the knowledge required for the generation of synthetic sequences [7, 138]. Consequently, we can detect failures and lacks in the knowledge used for behavior modeling. System *evaluation*, in the sense of explanation, helps to make easy to understand HSE systems by non-expert users, and also to validate the generated behavior descriptions.

Without context, that is, without specific knowledge to guide processing, HSE in the general case has exponential complexity. So context turns out to be a powerful heuristic to limit search and make the overall problem tractable. However, assumptions used for HSE should avoid the fact that it is now almost universal to assume the unreasonable [136].

2.5 Conclusions

Reviewed motion understanding approaches rely on a proper human action model which commonly embeds facilities for recognition. In fact, the design/selection of the action model usually depends on the application domain. Most referred action representations are designed both at the PDL and/or SDL. Subsequently, semantic primitives are designed at the CL in order to cope with human behavior evaluation.

As behavior evaluation relies on the action model, HSE is implemented by designing an action model suitable for human action recognition and synthesis, and human performance analysis. To this end, an action should be represented as a predefined sequence of human postures, where each posture is defined in terms of a proper human body model with anatomical information.

We can avoid explicitly all the issues involved for movement segmentation by using automatic motion capture. Thus, integration can be addressed independently of the application domain. That means also that human activity and behavior modeling at the BL should rely on the knowledge embedded in the action representation. If such an action model is designed at the SDL, 3-D information can be incorporated, thus allowing to derive rich conceptual descriptions when integration.

Chapter 3

Human Action Modeling

We next describe our proposal on human action modeling. For this purpose, three main issues will be discussed and implemented.

First, we define a proper human body model which embeds facilities for both human posture description and human action synthesis. Our proposed body posture representation maximizes the differences between human postures, thus allowing action recognition. On the other hand, the body model reflects the anatomical structure of the human beings, thus embodying facilities to synthesis of actions.

Secondly, a human action space, called *aSpace*, is presented in order to show how the posture variation can be learnt and represented in a compact, specific, and accurate manner. The basic idea is to represent each performance, i.e., a predefined sequence of postures, as a parametric manifold. This space models the principal modes of variation of the posture exhibited during several action performances.

Lastly, a human action representation, called *p–action*, will be presented. Such a model is based on the most characteristic human body postures found during several action performances. These postures are found automatically by means of a predefined distance function, and are called *key-frames*. By using key-frames, *p–actions* represent the *prototypical performance* for a given action.

As demonstrated in the next chapter, the proposed model also allows to perform action recognition and synthesis, and human performance analysis by using an unified framework. Furthermore, the action model will constitute an important queue for integration within the HSE framework.

3.1 Procedure for Data Acquisition

Motion capture is the process of recording live movement and translating it into usable mathematical terms by tracking a number of key points or regions/segments in space

over time and combining them to obtain a three-dimensional representation of the performance [109].

There are different ways of capturing movement. Some systems use electromagnetic fields or ultrasound to track a group of sensors. Also mechanical systems are available, which are based on linked structures or armatures that use potentiometers to determine the rotation of each link. Others use cameras which digitize different views of the movement, which are then used to put together the position of the key points or markers, normally reflective.

Unfortunately, all the above systems are invasive with the agent which is performing the movement. Electromechanical suits are very invasive, and the systems have a low sampling rate. Magnetic trackers have also a low sampling rate, and performers are constrained by cables in most cases. Optical systems include minimal reflective markers which should be used to recover the relative motion of the agent. Despite the fact that optical systems are economically expensive, these have a higher frequency of capture, resulting in more samples per second. Also, the capture area is larger than the performance area of magnetic or electromechanical systems. By using reflective markers on the body, optical data is extremely accurate, and the performer is not constrained by cables.

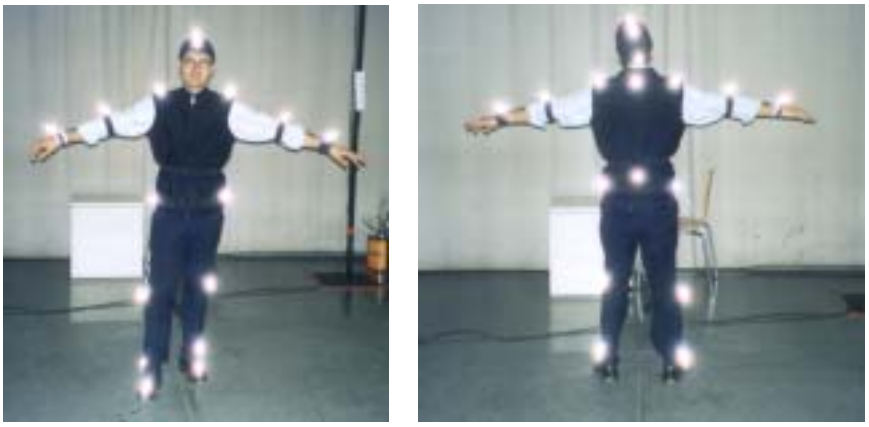
Therefore, an optical system is used to provide real training data to our algorithms. This system is based on six synchronized video cameras to record images. The optical system incorporates all the elements and equipment necessary for the automatic control of cameras and lights during the capture process. It also includes an advanced software pack for the reconstruction of movements and the effective treatment of occlusions.

In our experiments, the subject first placed a set of 19 reflective markers on the joints and other characteristic points of the body, see Fig. 3.1.**(a)** and **(b)**. These markers are small round pieces of plastic covered in reflective material. Subsequently, the agent is placed in a controlled environment (i.e., controlled illumination and reflective noise), where the capture will be carried out, see Fig. 3.1.**(c)**.

As a result, the accurate 3-D positions of the markers are obtained for each recorded frame, 30 frames per second. In our experiments, not all the 18 markers are considered to model human actions. In fact, we only process those markers which correspond to the joints of a predefined human body model. This model is presented next, and constitutes the basis of our proposed human action model.

3.2 Human Body Modeling

As an action is represented as a sequence of postures, a proper body model is required. In our experiments, the human body model considered is composed of twelve rigid body parts (hip, torso, shoulder, neck, two thighs, two legs, two arms and two forearms) and fifteen joints, see Fig. 3.2.**(a)**. These joints are structured in a hierarchical manner, where the root is located at the hip, see Fig. 3.2.**(b)**.



(a) **(b)**

(c)

Figure 3.1: Procedure for data acquisition. Figs. **(a)** and **(b)** shows the agent with the 19 markers on the joints and other characteristic points of its body. Figure **(c)** shows the scene where the motion capture system acquired the training samples.

Initially, each training sample $\mathbf{p}_s$ corresponds to the human body model defined as an ordered tuple of fifteen 3-D end-point coordinates of the twelve limbs listed above, each limb l defined by its end-points (x_i, y_i, z_i) and (x_j, y_j, z_j):

$$\mathbf{p}_s = (x_1, y_1, z_1, ..., x_{15}, y_{15}, z_{15})^T. \tag{3.1}$$

The variation of the human body is then represented as the sequence of human postures exhibited by the agent during an action performance, each posture defined as

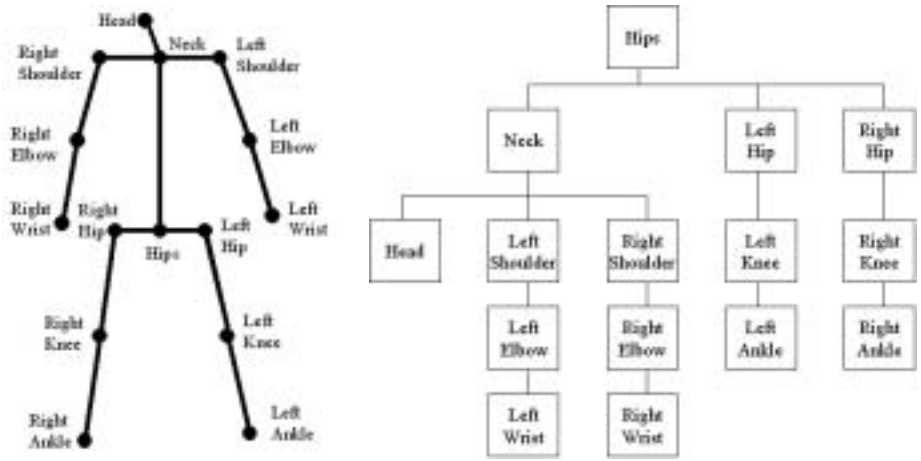


Figure 3.2: **(a)** Generic human body model represented using a stick figure similar to [35], here composed of twelve limbs and fifteen joints. **(b)** Hierarchy of the joints of the human body model. Once the 3-D position of these joints are obtained from a motion capture system, the human body representation is built conforming to this hierarchy.

in Eq. (3.1). However, this human body representation depends on the limb lengths of each recorded agent. Also, Cartesian coordinates are not ideal when considering linear approaches for posture variation modeling. These approches, which are based on linear statistics such as PCA, imply that only linear variations of the joints can be modeled independently [64]. Therefore, joints which exhibit non-linear variations, such as bending or pivoting, require of the combination of two or more linear modes of variation to be represented. As human body joints may exhibit complex non-linear variations, another body representation should be addressed.

Actually, coordinate systems which measure locations in terms of angles can cope with these non-linearities. For example, Figs. 3.3.**(a)** and **(b)** show the spherical and polar space coordinate systems respectively, which are more natural to be used for limb movement description [11]. Consequently, we define the human posture in terms of the angles of the limbs.

First, the spherical coordinate system is considered. A mapping function is used to transform the Cartesian coordinates of each limb l into spherical angles, namely *elevation* ϕ_l and *orientation* θ_l:

$$\begin{aligned} \phi_l &= \tan^{-1}\left(\frac{y_i - y_j}{\sqrt{(x_i - x_j)^2 + (z_i - z_j)^2}}\right), \\ \theta_l &= \tan^{-1}\left(\frac{x_i - x_j}{z_i - z_j}\right), \end{aligned} \tag{3.2}$$

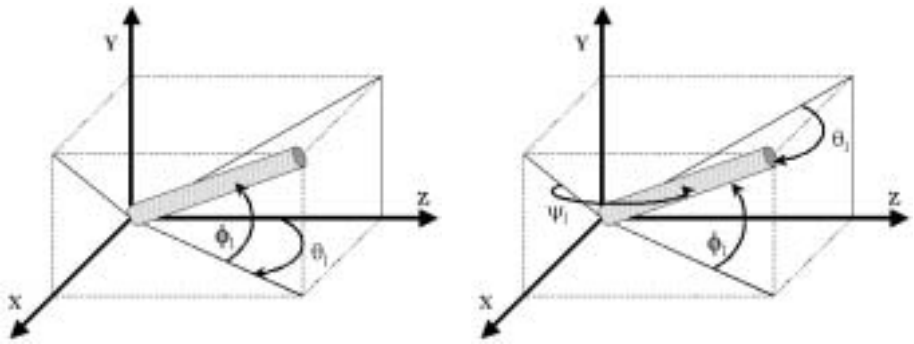


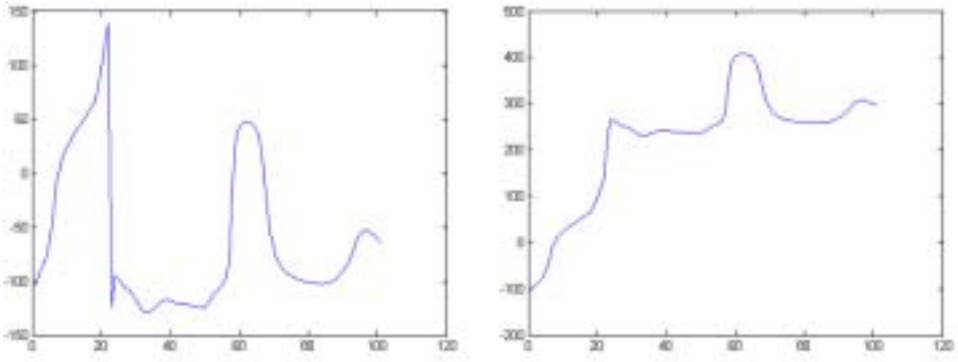
Figure 3.3: 3D coordinate systems suitable for limb movement description. **(a)** The spherical coordinate system describes a limb in terms of its elevation ϕ_l and orientation θ_l. **(b)** The polar space coordinate system describes a limb in terms of the elevation ϕ_l, latitude θ_l, and longitude ψ_l.

where denominators are prevented to be equal to zero. Elevation values are within $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. However, orientation values lie between the range of $[-\pi, \pi]$ (considering a bipolar range), and the angle discontinuity problem should be confronted. This is because a limb returns to the same position after each revolution of orientation. For example, we show in Fig. 3.4.**(a)** the orientation values corresponding to the left wrist during a bending performance. During this performance, the left arm hangs performing a circular swing. Despite of the fact that elevation values are stabilized near $-\frac{\pi}{2}$, orientation values *jump* between $-\pi$ and π. This discontinuity in the angle-time curve is not acceptable because it can cause problems in the computation of the human action model.

Fig. 3.4.**(b)** shows the resulting angle values by avoiding discontinuity jumps. However, note that the resulting continuous angle values are not restricted to lie between any determined range: a specific limb orientation can be mapped to several angle values $(\theta_l, 2\pi\theta_l, 4\pi\theta_l, \ldots)$. As the aim is to model angle variations over time, the range of training angle values should be limited.

Alternatively, a mapping to a higher dimensional space can be used to solve the discontinuity problem. That is, we model the limb orientation by using two different angles: we consider the polar space coordinate system which describes the orientation of a limb in terms of its latitude and longitude, see again Fig.3.3.**(b)**. As a result, the twelve independently moving limbs in the 3-D polar space have a total of thirty-six rotational DOFs which correspond to thirty-six absolute angles (w.r.t. the world coordinate system). So we compute the 3-D polar angles of a limb (i.e., elevation ϕ_l, latitude θ_l, and longitude ψ_l) as:

$$\phi_l \quad = \quad \tan^{-1}\left(\frac{y_i - y_j}{\sqrt{(x_i - x_j)^2 + (z_i - z_j)^2}}\right),$$



(a) **(b)**

Figure 3.4: Orientation values per frame during a bending performance. **(a)** As angle values lie between the range of $[-\pi, \pi]$, the angle discontinuity problem should be confronted. **(b)** Resulting angle values by avoiding discontinuity jumps.

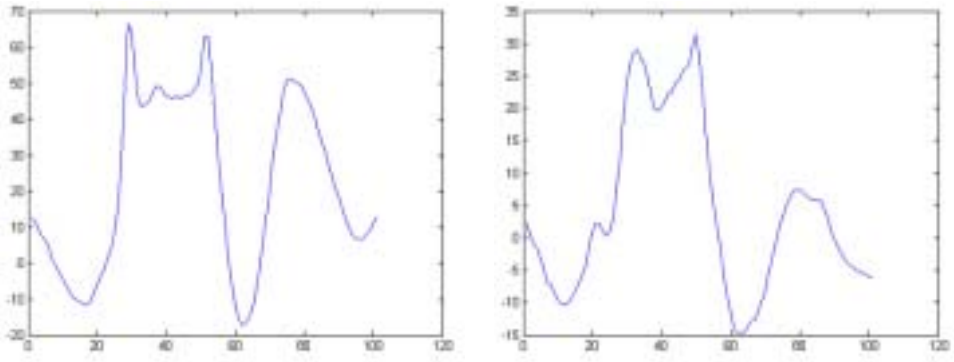
$$\begin{aligned} \theta_l &= \tan^{-1}\left(\frac{x_i - x_j}{\sqrt{(y_i - y_j)^2 + (z_i - z_j)^2}}\right), \\ \psi_l &= \tan^{-1}\left(\frac{z_i - z_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}\right), \end{aligned} \tag{3.3}$$

where denominators are also prevented to be equal to zero. Using this description, angle values lie between the range of $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, and the angle discontinuity problem is avoided, see Figs. 3.5.**(a)** and **(b)**.

The polar space representation may also include the length of each limb [11]. However, the size of the limbs is not a compelling issue when developing human action models. As a posture is described using angles, the human body model is independent of the size of the agent, thus generalizing our human action model to agents of different proportions.

Note that human actions are constrained movement patterns which involve to move the limbs of the body in a particular manner. That means, there is a relationship between the movement of different limbs while performing an action. In order to incorporate this relationship into the human action representation, we consider the hierarchy of Fig. 3.2.**(b)** in order to describe each limb with respect to its parent. That means, the relative angles between two adjacent limbs are next computed using the absolute angles of Eq. (3.3), see Fig. 3.6. Consequently, by describing the the human body using the relative angles of the limbs, we actually model the body as a hierarchical and articulated figure.

As a result, the model of the human body consists of thirty-six relative angles:



(a) **(b)**

Figure 3.5: Latitude θ_l **(a)** and longitude ψ_l **(b)** values per frame corresponding to the orientation value displayed in Fig. 3.4. As angle values lie between the range of $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, the angle discontinuity problem is avoided.

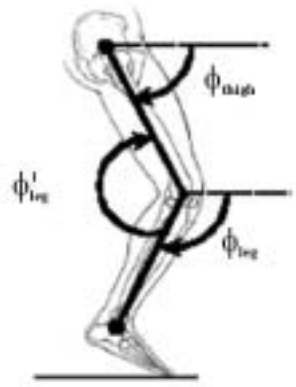


Figure 3.6: Our human body model is defined in terms of relative angles, following the hierarchy presented in this section.

$$\mathbf{\Delta}_s = (\phi'_1, \theta'_1, \psi'_1, \phi'_2, \theta'_2, \psi'_2, ..., \phi'_{12}, \theta'_{12}, \psi'_{12})^T. \tag{3.4}$$

Using this definition, we measure the *relative motion* of the human body. In order to measure the *global motion* of the agent within the scene, the variation of the height of the hip $\mathbf{u}_s$ over time is included in the model definition. In general, the spatial situation of the human body within the scene does not affect the execution of a performance. However, the variation of the height of the hip during an action performance can provide significant information about such an action, for example during jumping or tumbling. Thus, the real height value of the hip is normalized for

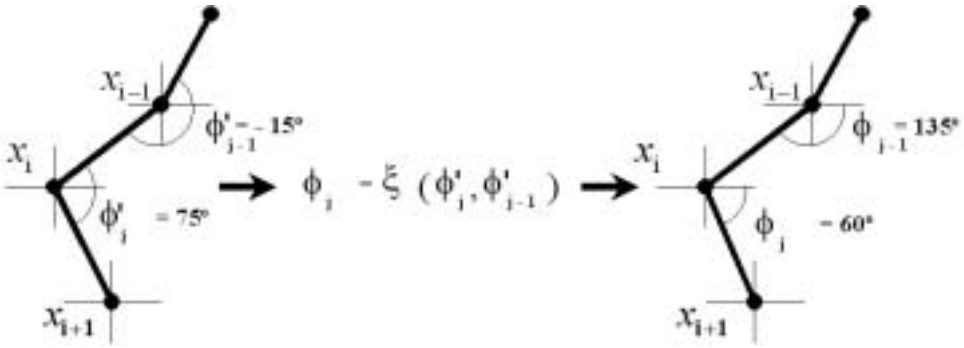


Figure 3.7: Given the hierarchy and the relative angles of the human body model, the absolute angle ϕ'_l of each limb can be computed. The function ξ transforms relative to absolute angles.

each captured agent in order to measure only the height variation.

Summarizing, our final human body model is defined in terms of thirty-six parameters and the variation of the height of the hip[1]:

$$\mathbf{x}_s = (\mathbf{u}_s, \mathbf{\Delta}_s)^T. \tag{3.5}$$

Table 3.1 reviews the complete human body modeling algorithm.

To conclude, we address how to build a graphical representation of the human body data. As the body model has been defined as a hierarchical tree, with the root located at the hip, we can compute incrementally the positions of the joints by using the relative angles of adjacent limbs. As shown in Fig. 3.7, the absolute angle values $(\phi_j, \theta_j, \psi_j)$ which defines the limb j, are calculated using the relative angles of both the limb j and its parent $j-1$, as defined in the body model hierarchy. Subsequently, the Cartesian coordinates $(x_{i+1}, y_{i+1}, z_{i+1})$ of the end-point $i+1$ of the limb j is computed using the end-point i of the limb parent $j-1$:

$$\begin{aligned} x_{i+1} &= x_i + r_j \ \sin\phi_j, \\ y_{i+1} &= y_i + r_j \ \sin\theta_j, \\ z_{i+1} &= z_i + r_j \ \sin\psi_j, \end{aligned} \tag{3.6}$$

where r_j is the (predefined) length of the limb j, and (x_i, y_i, z_i) refers to the end-point coordinates of the limb parent $j-1$, i.e., the Cartesian coordinates of the joint i which *links* both limbs.

[1] The following scheme can be enhanced to incorporate movement measurements of body joints as well.

Table 3.1: Human Body Modeling Algorithm

Input:

- Each training sample $\mathbf{p}_s$, defined as an ordered tuple of fifteen 3-D end-point coordinates of the twelve limbs, $\mathbf{p}_s = (x_1, y_1, z_1, ..., x_{15}, y_{15}, z_{15})^T$, where each limb l is defined by its end-points (x_i, y_i, z_i) and (x_j, y_j, z_j), according to the hierarchy of Fig. 3.2.**(b)**.

Algorithm:

1. Compute the 3-D polar absolute angles (elevation ϕ_l, latitude θ_l, and longitude ψ_l) of each limb l of the body model:

$$\begin{aligned}
\phi_l &= \tan^{-1}\left(\frac{y_i - y_j}{\sqrt{(x_i - x_j)^2 + (z_i - z_j)^2}}\right), \\
\theta_l &= \tan^{-1}\left(\frac{x_i - x_j}{\sqrt{(y_i - y_j)^2 + (z_i - z_j)^2}}\right), \\
\psi_l &= \tan^{-1}\left(\frac{z_i - z_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}\right),
\end{aligned}$$

   where denominators are prevented to be equal to zero.

2. Embed the resulting absolute angles into a vector which represents the configuration of the human body model: $(\phi_1, \theta_1, \psi_1, \phi_2, \theta_2, \psi_2, ..., \phi_{12}, \theta_{12}, \psi_{12})$.

3. Compute the 3-D relative angles for each limb, according to the hierarchy of Fig. 3.2.**(b)**: $\mathbf{\Delta}_s = (\phi'_1, \theta'_1, \psi'_1, \phi'_2, \theta'_2, \psi'_2, ..., \phi'_{12}, \theta'_{12}, \psi'_{12})^T$.

4. Compute the variation of the height of the hip, $\mathbf{u}_s$, which represents the *absolute motion* of the human body.

Output:

- The human body model, $\mathbf{x}_s = (\mathbf{u}_s, \mathbf{\Delta}_s)^T$.

In our experiments, we represent each human posture as a stick figure. This representation has the benefit of reflecting the anatomical structure of the human body. Therefore, the stick figure is a suitable representation to synthesize human movement. Furthermore, the stick figure allows to attain speed and simplicity of the computational processes activated for its evaluation.

3.3 Action Spaces: *aSpaces*

In order to design a human action model, several issues should be considered. The model should be compact, that is, it should be able to capture most of the variance of the training examples in a small number of parameters. Also, as dealing with non-rigid agents, non-linearities (due to non-linear movements of the limbs) should be handled. Furthermore, the model should be accurate, that is, the set of plausible postures that a human body can exhibit while performing the action should be modeled. However, the model of an action should be specific, in other words, a human action model should not generate postures which are not likely to be adopted during the performance of the action.

These requirements are also found in the literature when dealing with shape variability [64, 14, 26], which can be represented by means of Point Distribution Models (PDM) [38]. These statistical models are compact because they are based on PCA. PCA computes an orthogonal basis of the samples, the so-called eigenvectors. Therefore, the training data can be described as a linear combination of this basis. Usually, few eigenvectors are required, so a reduced dimensionality representation, compared with the dimensionality of the input data, is achieved. The distance between two points in the eigenspace can be considered as a measure of similarity. Actually, this distance corresponds to a measure of correlation between original samples [99]. Also, only plausible samples are generated by modifying, within limits, the principal modes of variation. These limits, related to the variance explained by each eigenvector, prevent the model to generate unallowable samples. PDM are suitable to be determined by automated training, because the training examples are not required to be either accurate or carefully selected. For further details on PDMs, see Appendix A.

Consequently, the PDM is a model which describes shape variability within a set of aligned shapes in a well-defined statistical manner. Adapting the procedure described in [64], each training sample of the PDM actually corresponds to the human body model as defined by Eq. (3.5). Only the variation of the normalized height of the hip will remain in the Cartesian domain. Using the motion capture system, we obtain the data samples of the PDM accurately, in order to develop our human action model by avoiding errors in the training data which would corrupt the reliability of the proposed action representation.

In our experiments, we have considered nine different elementary action types, which have been supplied to the system during a training process in a supervised manner. We named these actions as:

1. *aBend*,
2. *aJump*,
3. *aKick*,
4. *aRun*,
5. *aSit*,
6. *aSkip*,
7. *aSquat*,
8. *aTumble*,
9. *aWalk*,

and an activity called the *thief sequence* [27], which is a concatenation of *aWalk*, *aBend* and *aRun*. Also, we define the *aStand* action as the human body posture adopting an standing attitude.

In order to provide a proper learning set which is generic enough, the same action was performed five times (on average) by nine actors of different sizes. Note that each performance sequence is not compulsory to contain the same number of frames, that is, the duration of each performance may be different from that of other performances. As a result, approximately 45 performances per action were recorded. Taking into account that the acquisition rate is 30 fps, and that each performance lasts 5 seconds on average, we have acquired approximately seven thousand frames per action. Therefore, all actions embed 60 thousand posture configurations or training samples.

Each action, $\mathbf{A}$, is modeled using r different sequences, $\mathbf{A} = \{\mathbf{H}_1, \mathbf{H}_2, ..., \mathbf{H}_r\}$, where each sequence $\mathbf{H}_j$ of f_j human postures corresponds to a performance of such an action. Therefore, each $\mathbf{H}_j$ is composed of a set of body posture configurations $\mathbf{H}_j = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{f_j}\}$, where each $\mathbf{x}_i$ has dimensionality $n \times 1$ and is defined using the 37 parameters of Eq. (3.5).

The next step is concerned in building the action space $\mathbf{\Omega}^A$ by means of PCA, in which the action $\mathbf{A}$ will be represented. We denote such a feature space as *aSpace*. So given an action,

$$\mathbf{A} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_f\}, \tag{3.7}$$

where f refers to the overall number of training postures for this action:

$$f = \sum_{j=1}^{r} f_j. \tag{3.8}$$

Afterwards, we compute the mean posture for that action,

$$\bar{\mathbf{x}} = \frac{1}{f}\sum_{i=1}^{f}\mathbf{x}_i, \tag{3.9}$$

which is subtracted from each sample:

$$\hat{\mathbf{A}} = \{\mathbf{x}_1 - \bar{\mathbf{x}}, ..., \mathbf{x}_f - \bar{\mathbf{x}}\}. \tag{3.10}$$

Thus, it is guaranteed that the eigenvector with the largest eigenvalue corresponds to the dimension where the variance is maximum in a correlation sense. Next, the covariance matrix is computed as:

$$\Sigma = \frac{1}{f}\sum_{i=1}^{f}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T. \tag{3.11}$$

The eigenvalues λ_k and eigenvectors $\mathbf{e}_k$ of Σ are calculated by solving the eigenvector decomposition equation:

$$\lambda_k \mathbf{e}_k = \Sigma \mathbf{e}_k, \tag{3.12}$$

where each eigenvector, $\mathbf{e}_k$, corresponds to a mode of variation, and its corresponding eigenvalue λ_k is related to the variance explained by the eigenvector. In our case, note that each eigenvector reflects a mode of variation of the human posture during such an action.

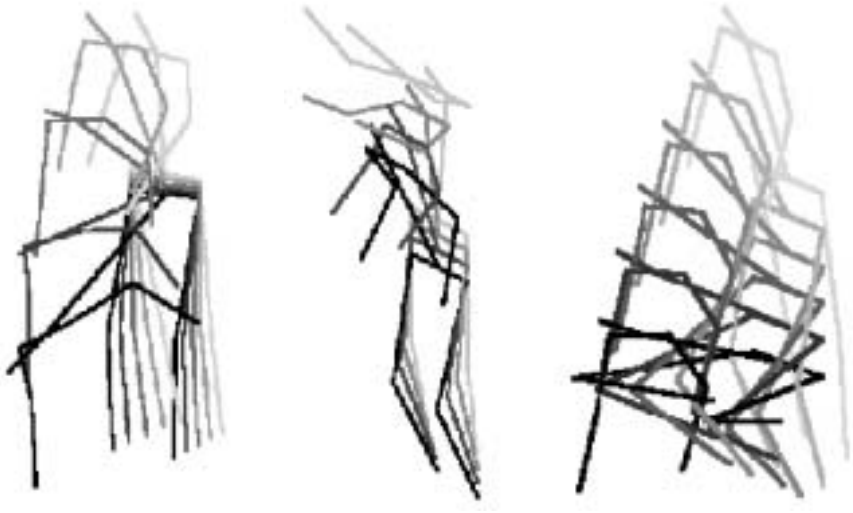
We preserve major linear correlations by considering the eigenvectors corresponding to the largest eigenvalues. By selecting the first m eigenvectors, $\{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_m\}$, we determine the dimensions of the action eigenspace. If we consider all n eigenvectors (the original size of the samples), the postures are exactly represented. When we just select $m \ll n$, a reduction in the dimensionality of the representation space is achieved, which involves a reconstruction error. So a critical issue is related to the choice of a proper number of dimensions of the eigenspace in order to retain the major variance of the input data or, in other words, when the squared reconstruction error decays and tends to become similar to the observation noise variance.

The value for m is commonly determined by eigenvalue thresholding. Consider the overall variance of the training samples, computed as the sum of the eigenvalues:

$$\lambda_T = \sum_{k=1}^{n}\lambda_k. \tag{3.13}$$

If we need to guarantee that the first m eigenvectors actually model, for example, 95% of the overall variance of the samples, we should choose m so that:

$$\frac{\sum_{k=1}^{m}\lambda_k}{\lambda_T} \geq 0.95. \tag{3.14}$$



(a) **(b)** **(c)**

Figure 3.8: Variation of the human posture while performing the *aBend* **(a)**, *aJump* **(b)** and *aTumble* **(c)** actions. These figures are generated by modifying the eigenvector associated with the largest eigenvalue of each *aSpace*.

Consequently, the smallest number m of eigenvalues which explains a large proportion of the total variance of the sequence of postures is chosen. As a result of Eq. (3.14), each modeled posture $\hat{\mathbf{x}}$ is found as a combination of:

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{E}\mathbf{b}, \tag{3.15}$$

where $\mathbf{E} = (\mathbf{e}_1, ..., \mathbf{e}_m)$ corresponds to the selected m eigenvectors, and $\mathbf{b} = (b_1, ..., b_m)^T$ is a vector of weights. Each weight is computed within suitable limits to generate new acceptable postures, similar to those in the training set. The limits are related to the variance of each variation mode, so each weight b_k is restricted to lie within:

$$-3\sqrt{\lambda_k} \leq b_k \leq 3\sqrt{\lambda_k}. \tag{3.16}$$

Consequently, all invalid postures are excluded automatically by obeying these limits. That is, the *aSpace* can be restricted to model only plausible human posture variations. Moreover, as the training data is obtained from a motion capture system, a realistic model of variation is provided.

Fig. 3.8 shows the principal modes of variation of the posture during the *aBend*, *aJump*, and *aTumble* actions, respectively. Once an *aSpace* is built for each action, these figures are generated by varying the vector of weights of Eq. (3.15) within the limits given by Eq. (3.16), corresponding to the largest eigenvalue. These figures demonstrate that each posture of a given action can be expressed in terms of a combination of the eigenvectors of the corresponding *aSpace* and a vector of weights.

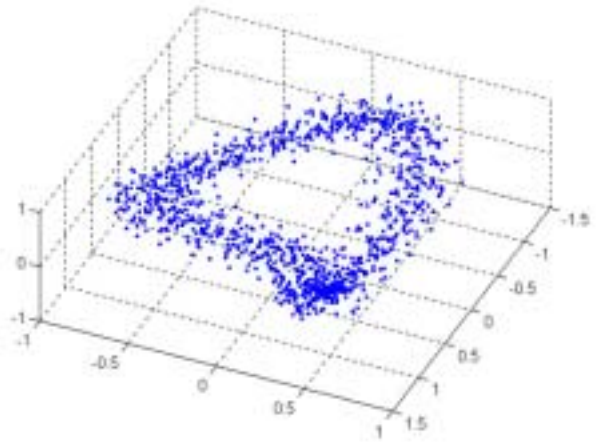


Figure 3.9: An example of an *aSpace*. Each point corresponds to the projection $\mathbf{y}_i$ of a posture configuration $\mathbf{x}_i$ presented during several *aRun* performances. Only the three eigenvectors associated with the three largest eigenvalues are displayed.

Furthermore, the eigenvectors with the largest eigenvalues are associated to the most important modes of variation of the human posture during an action.

Summarizing, we consider our human action space as the compact representation which consists of the orthogonal matrix $\mathbf{E}$ of eigenvectors, the diagonal matrix $\mathbf{\Lambda}$ of eigenvalues, and the mean posture $\bar{\mathbf{x}}$ of a given action:

$$\mathbf{\Omega}^A = (\mathbf{E}, \mathbf{\Lambda}, \bar{\mathbf{x}}). \tag{3.17}$$

3.4 Human Performance Representation

Using our action model, each performance is represented as a set of points in an eigenspace, each point corresponding to a human posture projection. This is achieved by projecting each learning posture $\mathbf{x}_i$ to the action space:

$$\mathbf{y}_i = [\mathbf{e}_1, ..., \mathbf{e}_m]^T (\mathbf{x}_i - \bar{\mathbf{x}}). \tag{3.18}$$

Thus, we obtain a set of discrete points $\mathbf{y}_i$ in the action space that represents the action class $\mathbf{\Omega}^A$, see Fig. 3.9. By projecting the set of human postures of a performance, we obtain a cloud of points wich corresponds to the projections of the postures exhibited during such a performance. As the acquisition rate of the camera is large enough to record smooth changes of the posture during consecutive frames, succeeding human postures become strongly correlated and, therefore, their projections into *aSpace* become close to one another.

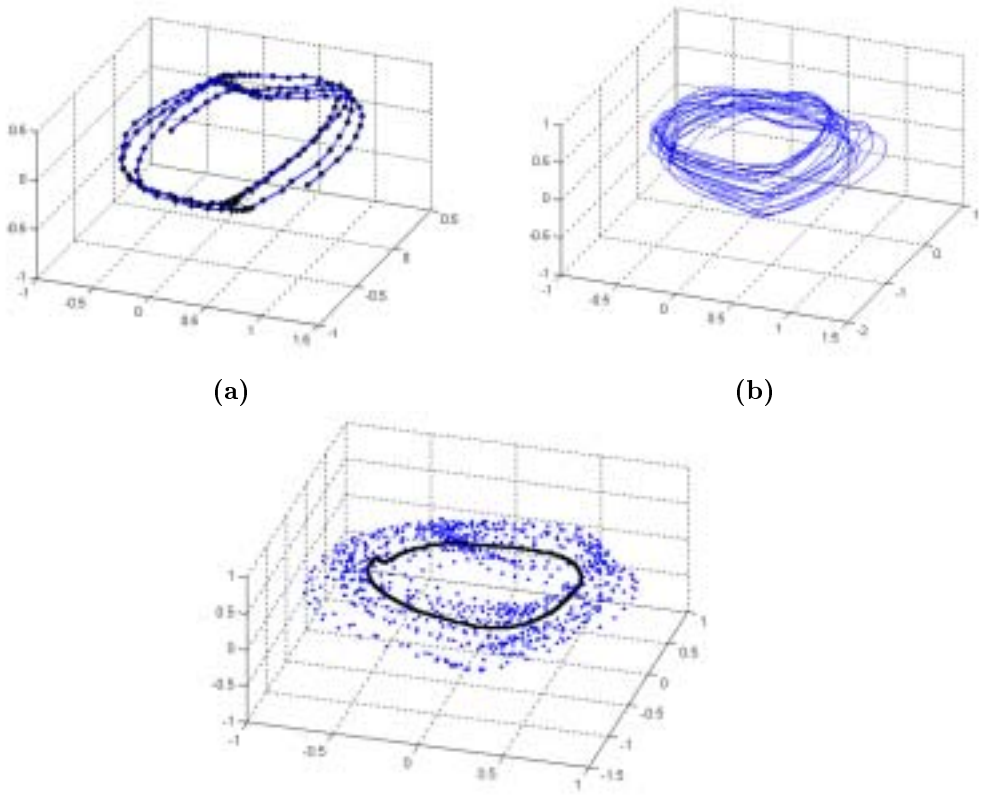


Figure 3.10: *aSpace* representation of the *aWalk* action. Only the three most prominent dimensions are displayed. **(a)** Manifold $\mathbf{g}_j(p)$ obtained by interpolation between the projections (black dots) of the postures of a single performance. **(b)** Manifolds obtained by interpolation between the projections of the postures of several performances. Each manifold corresponds to a performance. **(c)** Manifold $\mathbf{g}(p)$ obtained by interpolation between the means of pose distributions.

Consequently, a smoothly varying manifold in the *aSpace* can be computed [24, 99]. Therefore, we consider the projections $\mathbf{y}_i$ of a given a performance $\mathbf{H}_j$ as the control values for an interpolating curve $\mathbf{g}_j(p)$, which is computed using a standard cubic-spline interpolation algorithm [113], see Fig. 3.10.**(a)**. The parameter p refers to the temporal variation of the posture, which is normalized for each performance, that is, $p \in [0, 1]$. Thus, $p = 0$ refers to the beginning of a performance, and $p = 1$ refers to its termination. Usually, p is given in percentage: for example, $p = 50\%$ refers to the human body posture configuration at the middle of an action performance. Thus, by varying p, we actually move along the manifold. The idea of describing the time evolution of the action model by using the normalized pose has been also used to parameterize the walking action in [35].

As a result of interpolation, the manifold is described in terms of four matrices (namely, $\mathbf{B}_{\mathbf{g}_j}$, $\mathbf{C}_{\mathbf{g}_j}$, $\mathbf{D}_{\mathbf{g}_j}$ and $\mathbf{F}_{\mathbf{g}_j}$) of dimensionality $m \times 1$:

$$\mathbf{g}_j(p) = \mathbf{B}_{\mathbf{g}_j} p^3 + \mathbf{C}_{\mathbf{g}_j} p^2 + \mathbf{D}_{\mathbf{g}_j} p + \mathbf{F}_{\mathbf{g}_j}, \qquad p \in [0,1]. \tag{3.19}$$

Thus, each value of the parametric variable $p \in [0,1]$ determines the m coordinates $\mathbf{g}_j(p) = (y_1, ..., y_m)^T$ of the corresponding projection into *aSpace*. This process is repeated for each performance of the learning set, thus obtaining r manifolds. These manifolds are parameterized by the temporal variation of the posture, see Fig. 3.10.**(b)**:

$$\mathbf{g}_j(p), \qquad p \in [0,1], j = 1, ..., r. \tag{3.20}$$

As points lying in the manifold $\mathbf{g}_j(p)$ are indexed by their parameter p, it is possible to compute the mean postures of the r performances for each index p. First, we calculate:

$$\bar{\mathbf{g}}(p) = \frac{1}{r} \sum_{j=1}^{r} \mathbf{g}_j(p), \qquad p \in [0,1]. \tag{3.21}$$

Afterwards, the mean manifold is obtained by interpolating between these means:

$$\mathbf{g}(p) = \mathbf{B}_{\mathbf{g}} p^3 + \mathbf{C}_{\mathbf{g}} p^2 + \mathbf{D}_{\mathbf{g}} p + \mathbf{F}_{\mathbf{g}}, \qquad p \in [0,1]. \tag{3.22}$$

Fig. 3.10.**(c)** shows the point clouds in the *aSpace* corresponding to the *aWalk* action and its interpolated parametric curve $\mathbf{g}(p)$.

This performance representation is not influenced by its duration, expressed in seconds or number of frames. It is obvious that the posture configurations presented during a performance do not change with any variations of speed. So only the temporal variation of the human posture is modeled: our action model is centered on establishing how the human posture evolves to perform an action, instead of determining how much time a performance lasts.

Unfortunately, this resulting parametric manifold is influenced by the fact that any subject performs an action in the way he or she is used to. Furthermore, it is highly unlikely that the same actor performs the same action in an identical manner at different times. These problems affect the mean calculation for each index p. As a result, the manifold may comprise abrupt changes of direction. This rough property of the manifold is a direct consequence of the extreme variability of human posture configurations recorded during different performances of the same action.

So we need to determine how a *prototypical performance* representation can be derived by considering individual and highly variable performances as the learning set. And this is possible, since people can distinguish a given action despite of the performer. Therefore, our strategy is not centered on modeling the complete set of

Figure 3.11: Eadweard Muybridge, *The Human Figure in Motion*, frame extracted from Plate 23: Man running [100]. The action can be easily identified as someone running.

feasible postures which an actor could adopt while performing an action, because any new subject can include posture configurations not presented during the previous learnt performances.

A similar problem can be found in the computer animation domain, where the goal is to generate virtual figures exhibiting smooth and realistic movement. Commonly, animators define and draw a set of specific frames, called *key frames* or *extremes*, which assist the task of drawing the intermediate frames of the animated sequence. According to this procedure, we next analyze the key frame concept to incorporate it into our human action modeling approach.

3.5 Keyframing

In computer animation, the term *key-frame* has been generalized to apply to any variable whose value is set at specific frames and from which values for the intermediate frames are interpolated according to some prescribed procedure [108]. Key-frames are created by specifying the appropriate set of parameter values at a given time. The intermediate frames, called *in-betweens*, are obtained by smoothly interpolating the key-frames themselves when rendering.

Animators represent an action by means of a reduced set of postures which are characteristic enough. The concept *characteristic* is applied in the sense that these postures identify such an action, so they distinguish an action from all other actions. Within an action sequence there are few frames which contain most information about that action and whose combination provide the rest of the postures of the sequence. That means, there are postures which give enough information to state, by only considering these reduced group of postures, which action is being performed. As example, a single posture is shown in Fig. 3.11. Even with only one posture, one can recognize the action as someone running. This is due to the fact that such a posture is quite characteristic for a running action. Furthermore, this posture is not likely

to be found in other actions such as bending, waving arms, sitting or walking, for example. Thus, they can be used to discriminate between actions.

We have defined an action as a sequence of time-ordered body posture configurations. If one considers the complete set of body postures for an action sequence, it is obvious that the sequence contains redundant information: several postures are found to be repeated or quite similar in a correlation sense. A posterior analysis determines that there exists a more reduced group of frames that do not appear as frequently as the previous ones. Furthermore, these frames tend to correspond to transitions between groups of highly correlated frames. Also, they are found to identify that action.

Such capabilities argue for representing an action by selecting few frames from the entire sequence. Recently, an increasing number of papers which attempt to perform human action tracking and recognition by only considering few poses can be found in the literature. In [2], human movement tracking is applied during a gymnastic exercise which is known in advance. Tracking is performed by using a set of key-frames for each exercise, which are computed beforehand. Also in [15], few body poses are considered to achieve human action recognition. No criteria is used to select the key-frames, which are found randomly. In [135], human action tracking and recognition is performed by considering a set of stored key-frames. However, it is not clear how many key-frames are required for a given tennis stroke, due to the fact that they are provided beforehand. In [78], key-frames are called *exemplars*, which are selected from a sequence of frames using the k-means algorithm based on an error distortion measure. Exemplars are used to recognize humans from walking action sequences. Only equally spaced frames of a given sequence are considered in [91] to perform picture-based action recognition, so their human action model is not well-suited for synthesis purposes.

Alternatively, we present an algorithm which determines automatically not only the minimal number of postures, but also which are the most suitable postures for action representation. Consequently, only few human posture configurations will provide the *basis* of our final human action representation, which will constitute the *key-frame* set for that action.

3.6 Human Action Representation: *p–action*

Given an action **A**, our goal consists in the extraction of the most characteristic body posture configurations which will correspond to the set of key-frames for that action.

From a probabilistic point of view, we define characteristic postures as the least likely body postures exhibited during the action performances. As discussed before, quite a few characteristic body postures can be found when an action sequence is analyzed. This fact is simply a consequence of the very reduced number of frames where these postures appear.

So we need to represent the action in terms of a probability distribution in order

to compute the likelihood that a sample $\mathbf{x}_j$ is an instance of the action class $\mathbf{\Omega}^A$, that is, $P(\mathbf{x}_j|\mathbf{\Omega}^A)$. Low values actually correspond to less repetitive samples, that is, very characteristic postures for that action. Thus, by selecting those samples that are (locally) least likely, we assure that they provide high entropy of such an action class. On the other hand, similar samples correspond to repetitive frames and, therefore, exhibit high likelihood values.

As the *aSpace* is built based on PCA, such a space can also be used to compute the action class conditional density $P(\mathbf{x}_j|\mathbf{\Omega}^A)$. We assume that the *Mahalanobis distance* is a sufficient statistic for characterizing the likelihood:

$$d(\mathbf{x}_j) = (\mathbf{x}_j - \bar{\mathbf{x}})^T \Sigma (\mathbf{x}_j - \bar{\mathbf{x}}). \tag{3.23}$$

where $\bar{\mathbf{x}}$ corresponds to the mean posture defined by Eq. (3.9) and Σ to the covariance matrix defined by Eq. (3.11). Due to the diagonalized form of Σ, after finding its eigenvectors and eigenvalues, the previous expression can be computed as:

$$d(\mathbf{x}_j) = \sum_{i=1}^{n} \frac{{y_i}^2}{\lambda_i}, \tag{3.24}$$

where each y_i corresponds to the sample $\mathbf{x}_j$ projected to the action space and each λ_i is the i-th eigenvalue associated with the i-th eigenvector onto which the sample $\mathbf{x}_j$ is projected.

In order to build a prototypical action representation, we use the mean manifold which defines the mean performance. Once the mean manifold $\mathbf{g}(p)$ is established, we compute the likelihood values for the sequence of pose-ordered projections that lie in such a manifold [24, 97, 99]. Thus, extreme projections of the mean manifold are determined by computing the distance between each projection and the mean posture. That is, we apply Eq. (3.24) for each component of the manifold $\mathbf{g}(p)$. As a result, a distance measurement versus parameterized posture p is obtained, which is related to the likelihood of each projected posture. Note that this distance measurement is also related to important changes of direction of the mean manifold: since the elongation of each variation mode is taken into account, the distance function is also related to the curvature of the mean manifold.

For example, the graphic for the distance measure of the points belonging to the mean manifold computed in the *aBend aSpace* is shown in Fig. 3.12. Local Maxima of this function correspond to locally maximal distances or, in other words, to the least likely samples. These samples constitute the key-frames due to the fact that they correspond to the most characteristic postures of the action and, consequently, they provide most information about the action.

So each maximum of the distance function corresponds to a key-frame $\mathbf{k}_i$, where $i = 1, .., k$, and the number of key-frames k is determined by the number of maxima. Thus, we obtain the set of time-ordered key-frames for the action $\mathbf{A}$:

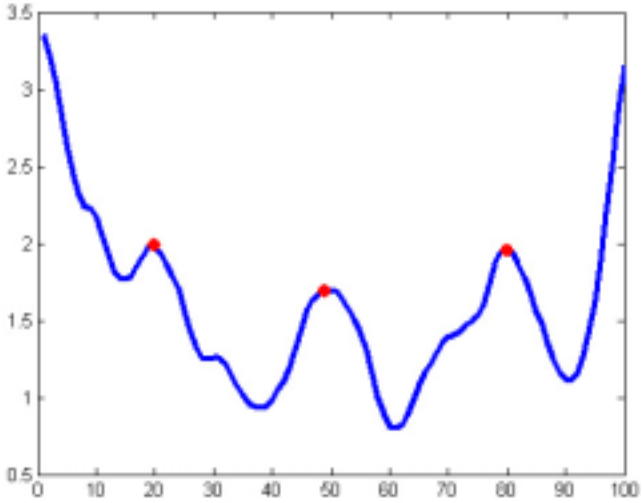
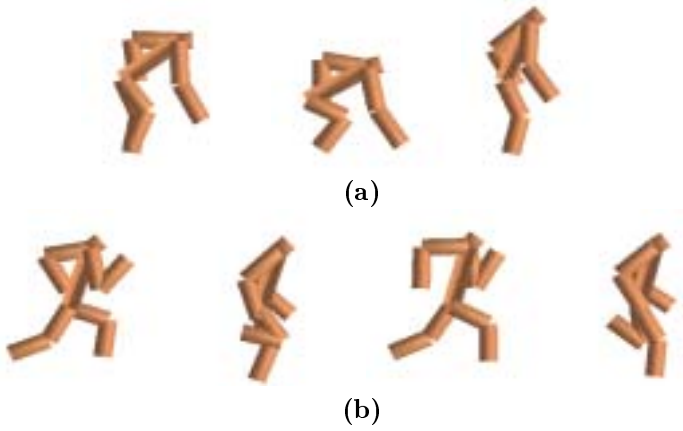


Figure 3.12: Distance measure after pose ordering applied to the points of the manifold in the *aBend aSpace*. Maxima (i.e., the key-frames) also correspond to important changes of direction of the manifold.



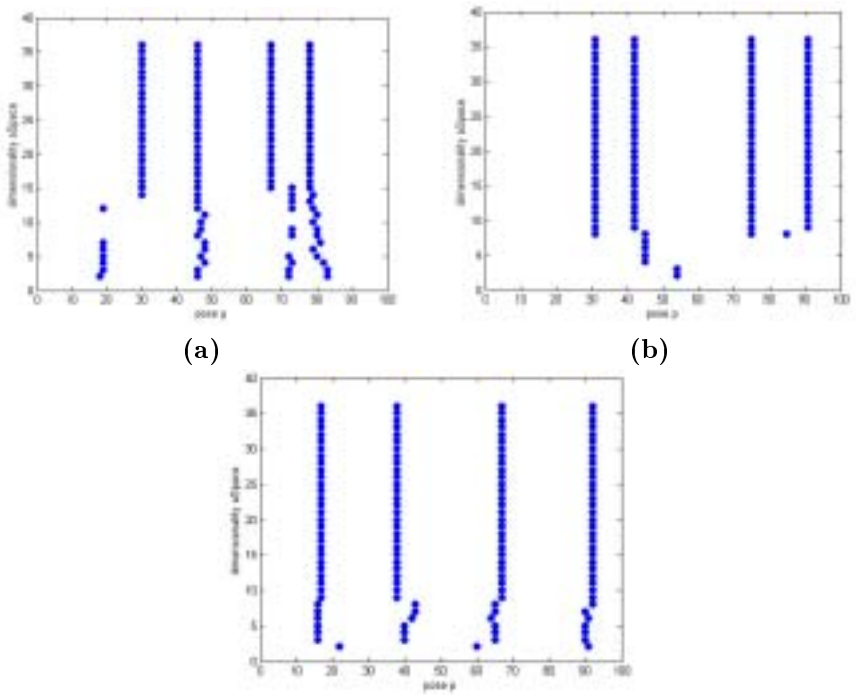
(a)

(b)

Figure 3.13: Postures corresponding to the peaks of the distance function for the *aBend* **(a)** and *aRun* **(b)** actions.

$$\mathbf{K}^A = \{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_k\}, \qquad \mathbf{k}_i \in \mathbf{g}(p). \tag{3.25}$$

Examples of key-frames obtained from the manifolds computed in the *aBend* and *aRun aSpaces* are shown in Fig. 3.13.



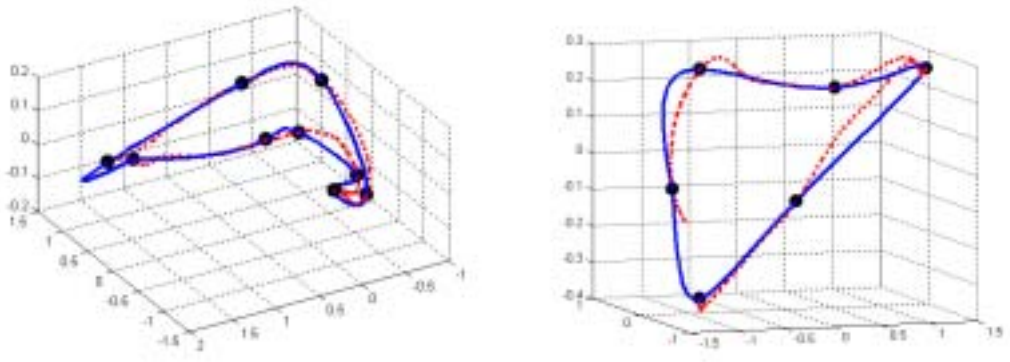
(a) **(b)**

(c)

Figure 3.14: The variation of the selected key-frames over the *aSpace* dimensionality is shown for *aTumble* **(a)**, *aSit* **(b)**, and *aRun* **(c)** actions. Dots represent the key-frames, i.e., the maxima of the distance function defined in Eq. (3.24).

In order to address a possible dimensionality reduction, due to the fact that not all the body parts move when an action is being executed, Fig. 3.14 shows the selected key-frames for the *aTumble*, *aSit*, and *aRun* mean manifolds by varying the dimensionality m of their corresponding *aSpaces*. Note that the key-frame selection process stabilizes in high dimensions. However, this stabilization is not achieved by considering an unique number of eigenvectors for all the modeled *aSpaces*. This is because the minimal number of eigenvectors required to represent a given action depends on the complexity of such an action, i.e., the variation of its constituent postures. For example, a complex action such as *aTumble* requires of 15 eigenvectors to capture the 95% of the overall variance of the training postures. On the other hand, the *aRun aSpace* is well-represented by considering only 9 dimensions.

Once the key-frames have been selected for a given action, $\mathbf{K}^A$, the prototypical performance representation is built as follows. By interpolating between the peaks of the distance function, a new manifold is computed within the *aSpace*. This new interpolated parametric curve is also parameterized by the pose p:



(a) **(b)**

Figure 3.15: Interpolation (blue solid curve) of the mean manifold (red dot curve) for *aSit* **(a)** and *aJump* **(b)** actions. Black dots represent the peaks of the distance function, i.e., the control points for interpolation.

$$\mathbf{f}^A(p) = \mathbf{B}^A p^3 + \mathbf{C}^A p^2 + \mathbf{D}^A p + \mathbf{F}^A, \qquad p \in [0,1], \tag{3.26}$$

which, as shown in Fig. 3.15, represents a manifold which *smooths* the manifold $\mathbf{g}(p)$: by using interpolation, we attain a reduction of the roughness inherited from the learning set. Thus, the posture variation over time is achieved to occur smoothly.

The projections of the key-frames found before correspond to k break points of this manifold, that is, $\mathbf{K}^A \in \mathbf{f}^A(p)$. Consequently, the curve is made up of transitions between key-frames, which represents how the body posture evolves from one key-frame to its next in temporal order.

We consider the manifold $\mathbf{f}^A(p)$ as the *prototypical performance* of the action $\mathbf{A}$. That means, p-ordered points belonging to such a manifold represent time-ordered postures exhibited during the prototypical performance of the action. We denote, therefore, $\mathbf{f}^A(p)$ as the *prototypical action representation* or *p–action*, see Fig. 3.16.

To conclude, our final human action representation $\mathbf{\Gamma}^A$ is defined as the combination of the *aSpace*, the *key-frames* and the *p–action* manifold:

$$\mathbf{\Gamma}^A = (\mathbf{\Omega}^A, \mathbf{K}^A, \mathbf{f}^A). \tag{3.27}$$

Table 3.2 reviews the complete human action modeling algorithm.

Once the human action model has been defined, its capabilities should be discussed. Consequently, the next chapter exploits the *aSpace* and *p–action* representations to implement procedures for human action recognition and synthesis, and performance analysis.

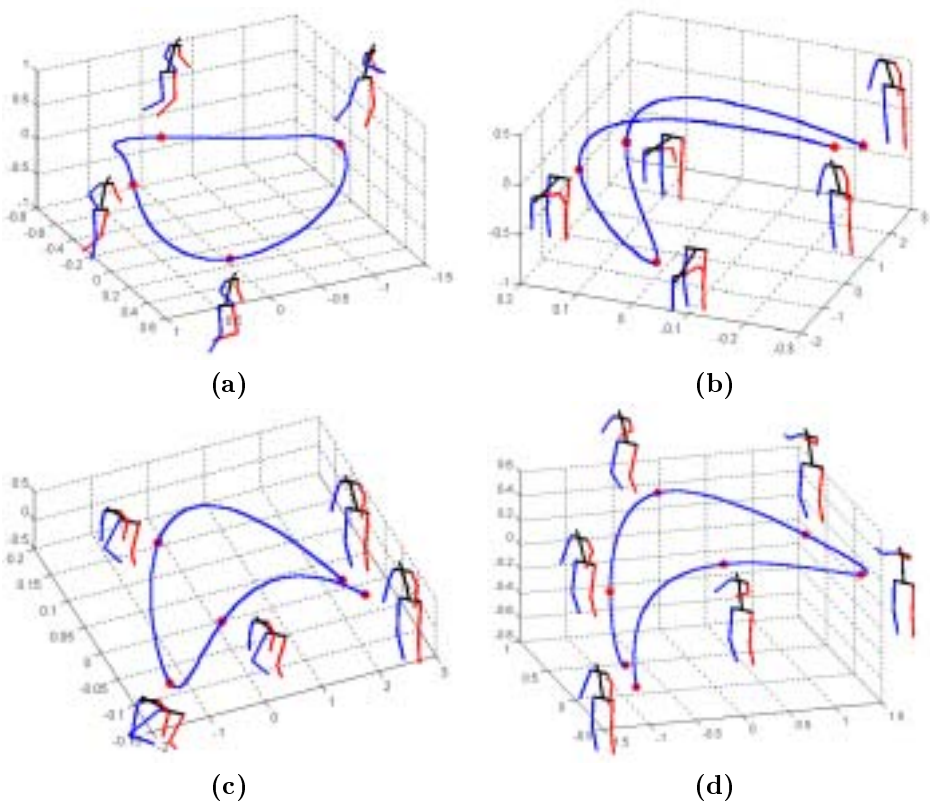


Figure 3.16: *p–actions* computed in the *aRun* **(a)**, *aBend* **(b)**, *aSquat* **(c)** and *aJump* **(d)** *aSpaces*: by varying the parameter pose p, we actually move along the manifold $\mathbf{f}^A(p)$, thus obtaining the temporal evolution of the human body posture during the prototypical performance of any learnt action.

Table 3.2: Human Action Modeling Algorithm

Input:

- Action $\mathbf{A}$ to be modeled, $\mathbf{A} = \{\mathbf{H}_1, \mathbf{H}_2, ..., \mathbf{H}_r\} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_f\}$, where each performance $\mathbf{H}_j$ is composed of a sequence of f_j postures, each posture $\mathbf{x}_i$ of $n \times 1$ dimensionality and defined by equation (3.5).

Algorithm:

1. Compute the mean posture, $\bar{\mathbf{x}} = \frac{1}{f} \sum_{i=1}^{f} \mathbf{x}_i$.
2. Compute the covariance matrix, $\Sigma = \frac{1}{f} \sum_{i=1}^{f} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$.
3. Obtain the eigenvalues λ_i and eigenvectors $\mathbf{e}_i$ of Σ by applying the eigenvector decomposition equation, $\lambda_i \mathbf{e}_i = \Sigma \mathbf{e}_i$.
4. Select the dimensionality of the *aSpace*, i.e., an m value so that $\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq 0.95$.
5. Define the *aSpace* of dimensionality m as $\mathbf{\Omega}^A = (\mathbf{E}, \mathbf{\Lambda}, \bar{\mathbf{x}})$.
6. Project each learning sample $\mathbf{x}_i$ in the *aSpace*, thus obtaining the projection $\mathbf{y}_i = [\mathbf{e}_1, ..., \mathbf{e}_m]^T (\mathbf{x}_i - \bar{\mathbf{x}})$.
7. Compute a manifold $\mathbf{g}_j(p)$ for each performance $\mathbf{H}_j$ by interpolating its projections, $\mathbf{g}_j(p) = \mathbf{B}_{\mathbf{g}_j} p^3 + \mathbf{C}_{\mathbf{g}_j} p^2 + \mathbf{D}_{\mathbf{g}_j} p + \mathbf{F}_{\mathbf{g}_j}$.
8. Calculate the mean posture for each pose p, $\bar{\mathbf{g}}(p) = \frac{1}{r} \sum_{j=1}^{r} \mathbf{g}_j(p)$.
9. Using interpolation between the mean postures of $\bar{\mathbf{g}}(p)$, compute the mean manifold $\mathbf{g}(p) = \mathbf{B}_{\mathbf{g}} p^3 + \mathbf{C}_{\mathbf{g}} p^2 + \mathbf{D}_{\mathbf{g}} p + \mathbf{F}_{\mathbf{g}}$.
10. By applying the distance function of Eq. (3.24), select the key-frame set of $\mathbf{g}(p)$, $\mathbf{K}^A = \{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_k\}$.
11. Interpolate between the key-frame set to obtain the prototypical action representation, or *p–action*, $\mathbf{f}^A(p) = \mathbf{B}^A p^3 + \mathbf{C}^A p^2 + \mathbf{D}^A p + \mathbf{F}^A$.

Output:

- The action model, $\mathbf{\Gamma}^A = (\mathbf{\Omega}^A, \mathbf{K}^A, \mathbf{f}^A)$.

Chapter 4

Applications

Last chapter focused on building a human action space, called *aSpace*, within which human actions are modeled as parametric manifolds, or *p–actions*. A *p–action* represents the temporal evolution of the human body posture during the prototypical performance of a modeled action.

The benefits of the *aSpace* and *p–action* representation are exploited by implementing three generic applications. First, we adapt the key-frame set to perform human action recognition. We present a procedure which compares manifolds using a proper distance measure and a suitable classification criteria. Second, the *p–action* representation is used to synthesize virtual sequences which reproduce a human action and a human activity. Due to the interpolation nature of *p–actions*, synthetic sequences exhibit smooth and realistic movement. Moreover, the variation of the human posture between consecutive actions is also synthesized. Lastly, we address the analysis of human actions by comparing different performances of the same action. By attaining independence from the speed at which performances are played, the *style* of human walking is analysed by establishing differences between a male and a female walkers.

4.1 Human Action Recognition

Once our human action representation has been established, we next present an approximation to human action recognition, here considered as a classification problem. In fact, we require an unique feature space to perform comparison between different actions. Therefore, all learnt human actions will be modeled as parametric manifolds in an unique *aSpace*, called *UaSpace*, following the procedure presented in chapter 3.

Subsequently, a human action recognition algorithm is presented, which classifies an unknown input sequence as belonging to an action class. We follow the method presented in [91], where recognition is achieved by comparing the manifold of a test

action with a set of reference manifolds. Lastly, experimental results are shown by comparing three different classification criteria.

4.1.1 The Universal *aSpace* or *UaSpace*

In order to represent all the learnt actions within the same reduced space, an universal *aSpace* is built, namely $\mathbf{\Omega}^U$ or *UaSpace*. In the *UaSpace*, each learnt action $\mathbf{A}_i$ is represented as a manifold $\mathbf{f}^{A_i}$ which has been obtained by interpolation of its *key-frame* set $\mathbf{K}^{A_i}$, as described in the previous chapter. Next, the procedure is described.

The training data $\mathbf{A}^U$ is composed of the set of actions $\mathbf{U} = \{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_u\}$ to be learnt, that is, $\mathbf{A}^U$ comprises r sequences of different actions:

$$\mathbf{A}^U = \{\mathbf{H}_1, \mathbf{H}_2, ..., \mathbf{H}_r\}. \tag{4.1}$$

Each sequence $\mathbf{H}_j$ of f_j frames corresponds to a performance of an action, which is composed of a sequence of human body configurations:

$$\mathbf{H}_j = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{f_j}\}, \tag{4.2}$$

where each $\mathbf{x}_i$ of dimensionality $n \times 1$ corresponds to the 37 values of the human body model described in the last chapter. Let f be the number of body posture configurations of $\mathbf{A}^U$:

$$f = \sum_{j=1}^{r} f_j. \tag{4.3}$$

Thus, the training data comprises a sequence of f body posture configurations:

$$\mathbf{A}^U = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_f\}. \tag{4.4}$$

Once the mean human posture $\bar{\mathbf{x}}^U$ (see Fig. 4.1) and the covariance matrix Σ^U of $\mathbf{A}^U$ are computed as described in the last chapter, the eigenvalues λ_i^U and eigenvectors $\mathbf{e}_i^U$ of Σ^U are calculated by solving the eigenvector decomposition equation.

However, not all the resulting eigenvectors contribute to model the variation of the training data. Next, we determine the number of eigenvectors m^U which capture most of the variation. Fig. 4.2.**(a)** shows the individual contribution of each eigenvector, which is computed as:

$$\frac{\lambda_i^U}{\sum_{k=1}^{n} \lambda_k^U}, \qquad i = 1, ..., n \tag{4.5}$$

It can be seen that past the 20th eigenvector, the contribution is less than 0.5%. On the other hand, Fig. 4.2.**(b)** shows the cumulative contribution of each eigenvector, calculated as:

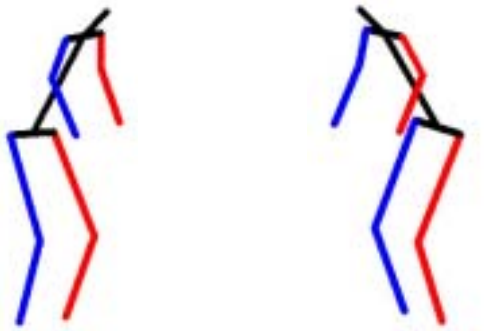
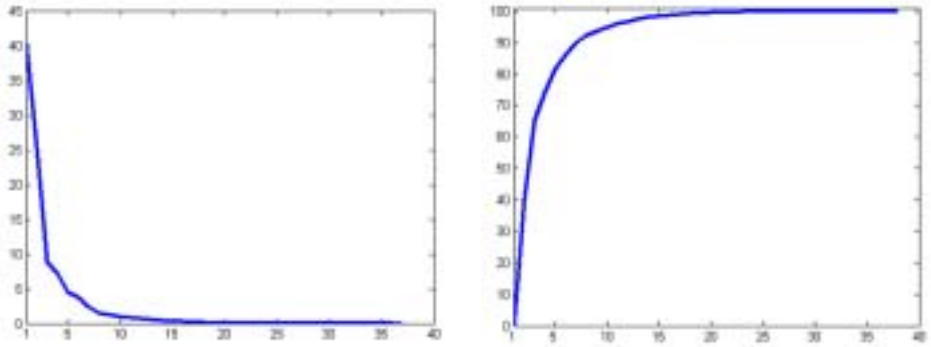


Figure 4.1: Two different views of the mean posture $\bar{\mathbf{x}}^U$ of the *UaSpace* $\mathbf{\Omega}^U$. Red and blue colors refer to the left and right side of the human body, respectively.



(a) **(b)**

Figure 4.2: Contribution of each eigenvector in the *UaSpace*. The individual contribution of each eigenvector is shown in **(a)**. The cumulative contribution is shown in **(b)**.

$$\frac{\sum_{k=1}^{i} \lambda_k^U}{\sum_{k=1}^{n} \lambda_k^U}, \qquad i = 1, ..., n \tag{4.6}$$

The curve increases rapidly during the first eigenvectors. It can be seen that the first 20 eigenvectors associated to the 20 largest eigenvalue capture more than 95% of the variation. Consequently, m^U is set to 20.

The resulting mean posture $\bar{\mathbf{x}}^U$, the orthogonal matrix $\mathbf{E}^U$ of m^U eigenvectors, and the diagonal matrix $\mathbf{\Lambda}^U$ of m^U eigenvalues of $\mathbf{A}^U$ constitute the *UaSpace* representation:

$$\mathbf{\Omega}^U = (\mathbf{E}^U, \mathbf{\Lambda}^U, \bar{\mathbf{x}}^U). \tag{4.7}$$

(a) **(b)** **(c)**

Figure 4.3: Modes of variation of *UaSpace*. Only the three eigenvectors associated to the three largest eigenvalues are shown. See text for details.

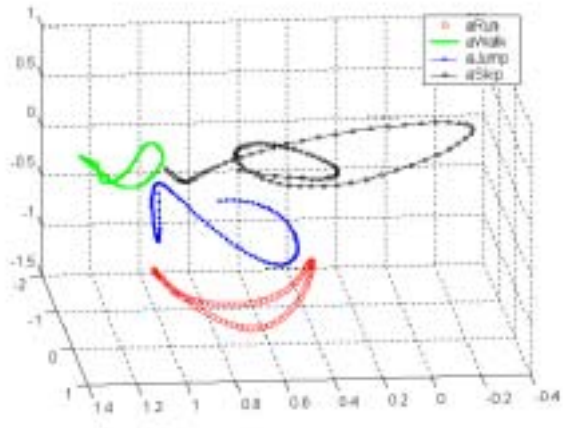
Fig. 4.3 displays the most important modes of variation of the human posture in the *UaSpace*. As expected, these modes of variation are found repeated for different learnt actions. Thus, Fig. 4.3.**(a)** reflects the variation of the posture exhibited during the *aBend*, *aSit*, and *aSquat* actions. Note that the most important variation mode corresponds to the inclination of the torso. Also, a smooth bending variation of the legs and thighs characterizes such actions. Fig. 4.3.**(b)** shows that the variation of the height of the hip is also modeled. In fact, this variation of the posture is exhibited in different actions, such as *aSit*, *aSquat*, and *aTumble* actions, during which folded legs and thighs are present. Fig. 4.3.**(c)** correspond to the third mode of variation which correspond to the movement of legs and arms recorded in the *aWalk*, *aSit* and *aJump* actions.

In the *UaSpace*, all the actions are represented as parametric manifolds, following the procedure described in chapter 3. First, for each action $\mathbf{A}_i$, its corresponding mean performance manifold $\mathbf{g}^{A_i}(p)$ is computed. Once the key-frame set $\mathbf{K}^{A_i}$ is established for each mean manifold, the prototypical performance or *p–action* $\mathbf{f}^{A_i}(p)$ is derived.

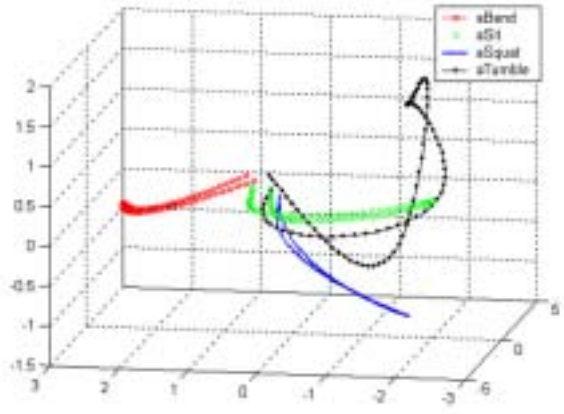
Note that each *p–action* manifold can be described in terms of a sequence of q projections $\mathbf{y}^{A_i}$:

$$\mathbf{f}^{A_i}(\mathbf{p}) = [\mathbf{y}_1^{A_i}, \mathbf{y}_2^{A_i}, ..., \mathbf{y}_q^{A_i}], \tag{4.8}$$

where each $\mathbf{y}_j^{A_i} = [y_1^{A_i}, y_2^{A_i}, ..., y_{m^U}^{A_i}]^T$ is composed of the m^U coefficients of the eigenvectors defined in $\mathbf{\Omega}^U$, and corresponds to a human body posture projected in the *UaSpace*. Such projections actually represent the *p–action* manifold sampled at specific $\mathbf{p} = \{p_1, p_2, ..., p_q\}$, each $p \in [0, 1]$.



(a)



(b)

Figure 4.4: Parametric curves or *p–actions* in the *UaSpace*. Only the three eigenvectors associated to the three largest eigenvalues are displayed.

To conclude, the representation of an action $\mathbf{A}_i$ is modeled within the *UaSpace* as:

$$\mathbf{\Gamma}^{A_i} = (\mathbf{\Omega}^U, \mathbf{K}^{A_i}, \mathbf{f}^{A_i}). \tag{4.9}$$

Fig. 4.4 shows the set of manifolds in the *UaSpace*, each manifold corresponding to a specific action. If the manifolds of two actions intersect, the intersection corresponds to very similar human postures of both actions. In fact, the distance between two points can be considered as a measure of similarity between human postures, as described in the next section.

4.1.2 Procedure for Human Action Recognition

Once the *UaSpace* is built, two issues need to be addressed in order to perform recognition following a pattern matching approach: on the one hand, to define a proper distance measure and, on the other hand, to establish a suitable comparison procedure for classification based on the selected distance measure.

Consider a new performance of an unknown action $\mathbf{A}_T$ to be recognized. We build its corresponding human performance representation:

$$\mathbf{\Gamma}^{A_T} = (\mathbf{\Omega}^U, \mathbf{K}^{A_T}, \mathbf{f}^{A_T}), \tag{4.10}$$

that is, the manifold $\mathbf{f}^{A_T}$ obtained by interpolation between the projections of $\mathbf{A}_T$ in the $\mathbf{\Omega}^U$ *UaSpace*, and the key-frame set $\mathbf{K}^{A_T}$.

First, we describe the distance measure between two manifolds, namely $\mathbf{f}^{A_i}$ and $\mathbf{f}^{A_T}$. Each manifold is described as a sequence of projections, that is:

$$\begin{aligned} \mathbf{f}^{A_i}(\mathbf{p}) &= [\quad \mathbf{y}_1^{A_i}, \mathbf{y}_2^{A_i}, ..., \mathbf{y}_{q_i}^{A_i} \quad], \\ \mathbf{f}^{A_T}(\mathbf{p}) &= [\quad \mathbf{y}_1^{A_T}, \mathbf{y}_2^{A_T}, ..., \mathbf{y}_{q_T}^{A_T} \quad], \end{aligned} \tag{4.11}$$

where q_i and q_T refer to the number of projections in the *UaSpace* of the reference and the test manifolds, respectively, which will be considered for comparison.

In fact, the distance between two projections in *UaSpace* is a measure of correlation between their corresponding human postures: the closer the points are, the more highly correlated are the postures [99]. Therefore, correlation is used very often as a measure of similarity between samples.

Thus, the distance between two manifolds is taken as the mean minimum distance between their correspondent normalized projections:

$$d(\mathbf{f}^{A_i}, \mathbf{f}^{A_T}) = \frac{1}{q_i} \sum_{l=1}^{q_i} \min_{1 \le j \le q_T} \left\| \frac{\mathbf{y}_l^{A_i}}{\| \mathbf{y}_l^{A_i} \|} - \frac{\mathbf{y}_j^{A_T}}{\| \mathbf{y}_j^{A_T} \|} \right\|. \tag{4.12}$$

To ensure symmetry, the final distance measure is defined as:

$$D(\mathbf{f}^{A_i}, \mathbf{f}^{A_T}) = \frac{1}{2} \left(d(\mathbf{f}^{A_i}, \mathbf{f}^{A_T}) + d(\mathbf{f}^{A_T}, \mathbf{f}^{A_i}) \right). \tag{4.13}$$

This distance measure is a variant of the Hausdorff metric (we use the mean of minima instead of the maximum of minima), and is invariant to temporal shifts; for further details, see [91].

We compute the distance measure of the new manifold with respect all learnt actions. Subsequently, the test manifold is classified as belonging to the action with

the nearest projections. Note that recognition only involves to compute a number of distances D equal to the number of action classes.

The previously defined distance measure will be computed over only few points of the manifold. However, different criteria can be established in order to select automatically these samples. For example, only equally spaced projections are considered in [91] to perform human action classification. Also, a human action recognition procedure is presented in [15], which is based on randomly selected projections. Alternatively, we propose to consider the key-frame set $\mathbf{K}^{A_i}$ as those samples used for classification. Next, we compare these different selection criteria.

Using the distance measure of Eq. (4.13), three different classifiers are considered:

Minimum Distance to Equally-spaced Projections (MDEP) : For each reference manifold, only equally spaced projections are considered to compute the distance function, as proposed in [91]. Thus, we choose $\mathbf{p}^e = \{p_1, p_2, ..., p_k\}$ of an action manifold $\mathbf{f}^{A_i}$, so that each p_j and p_{j+1} are equally spaced:

$$\mathbf{f}^{A_i}(\mathbf{p}^e) = [\mathbf{y}_1^{A_i}, \mathbf{y}_2^{A_i}, ..., \mathbf{y}_k^{A_i}], \qquad \mathbf{y}_j^{A_i} \in \mathbf{f}^{A_i}(p). \tag{4.14}$$

In our experiments, the number of projections is equal to the number of key-frames.

Minimum Distance to Randomly-spaced Projections (MDRP) : we apply the distance measure to randomly selected projections, as proposed in [15]. For each action manifold $\mathbf{f}^{A_i}$, we establish a sequence of $\mathbf{p}^r = \{p_1, p_2, ..., p_k\}$, so that each p_j is randomly selected. As a result, an action manifold is represented as:

$$\mathbf{f}^{A_i}(\mathbf{p}^r) = [\mathbf{y}_1^{A_i}, \mathbf{y}_2^{A_i}, ..., \mathbf{y}_k^{A_i}], \qquad \mathbf{y}_j^{A_i} \in \mathbf{f}^{A_i}(p). \tag{4.15}$$

We select as many projections as the number of key-frames.

Minimum Distance to Key-frames (MDK) : only the key-frame set, that is, $\mathbf{K}^{A_i} = \{\mathbf{k}_1^{A_i}, \mathbf{k}_2^{A_i}, ..., \mathbf{k}_k^{A_i}\}$ is considered for comparison. Therefore, the set of projections used for comparison, $\mathbf{p}^k$, embeds those p_j of the projections selected as key-frames:

$$\mathbf{f}^{A_i}(\mathbf{p}^k) = [\mathbf{k}_1^{A_i}, \mathbf{k}_2^{A_i}, ..., \mathbf{k}_k^{A_i}], \qquad \mathbf{k}_i^{A_i} \in \mathbf{f}^{A_i}(p). \tag{4.16}$$

4.1.3 Experimental Results

Considering the universal data set, composed of 405 video sequences, we used the samples of two performances for each action and subject for training (162 video sequences). This leaves a test data of 243 video sequences. The training instances were used to obtain the mean performance for each action and, subsequently, to compute the set of *p–actions* and the *UaSpace*.

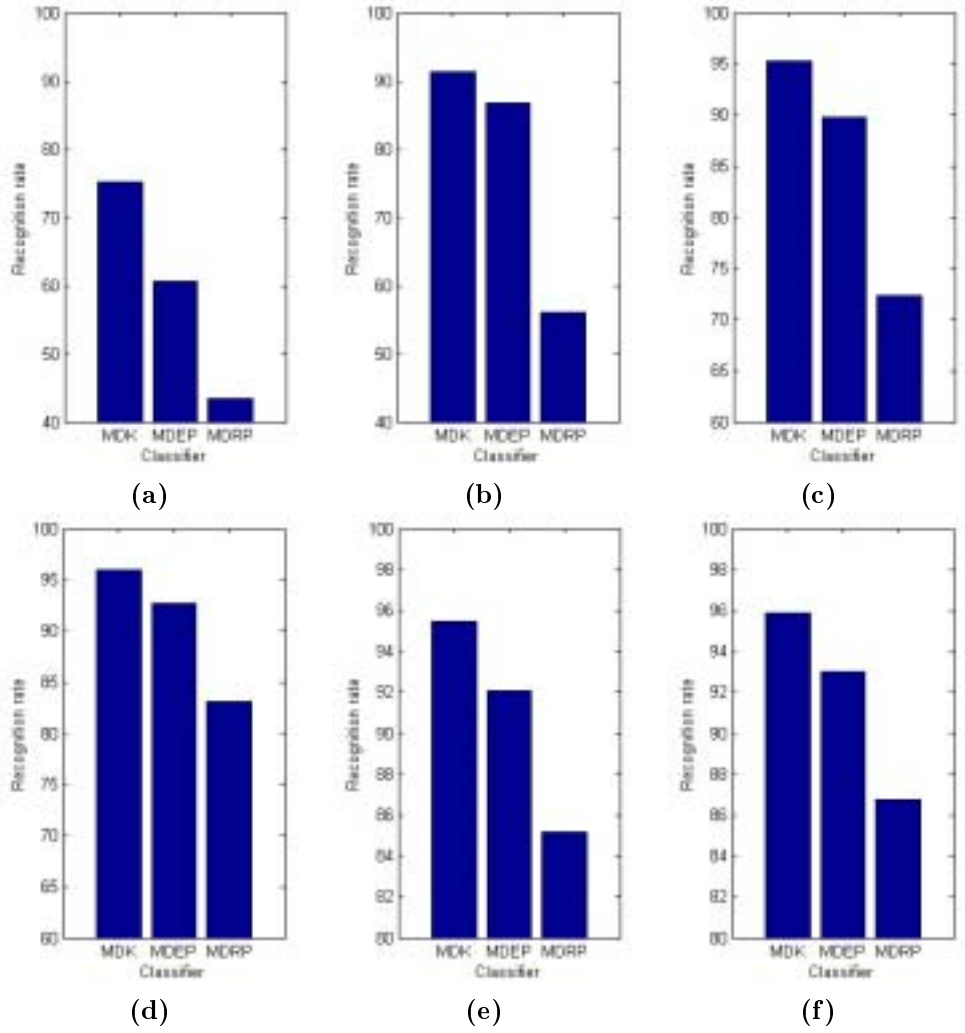


Figure 4.5: Recognition performance by varying the dimensionality of the *UaSpace*, specifically for $m^U = 5$ **(a)**, $m^U = 10$ **(b)**, $m^U = 15$ **(c)**, $m^U = 20$ **(d)**, $m^U = 25$ **(e)**, and $m^U = 30$ **(f)**.

Fig. 4.5 shows the recognition rate for these different classifiers, by varying the number of eigenvectors m^U considered to model the *UaSpace*. As m^U increases, the recognition rate improves and stabilizes past the 20th eigenvector. In higher dimensions, MDEP performance approaches to MDK results. This fact supports current tendency of selecting few (temporally) spaced postures for action recognition[1].

[1]However, MDEP may not provide those postures which best characterize an action sequence, that is to say, the postures which are best suited to be used for action synthesis and performance analysis.

Action	aBend	aJump	aSkip	aRun	aWalk	aKick	aSquat	aSit	aTumble
aBend	27	0	0	0	0	0	0	0	0
aJump	0	25	2	0	0	0	0	0	0
aSkip	0	0	27	0	0	0	0	0	0
aRun	0	0	0	27	0	0	0	0	0
aWalk	0	0	0	2	25	0	0	0	0
aKick	0	0	0	0	0	27	0	0	0
aSquat	0	0	0	0	0	0	23	1	3
aSit	0	0	0	0	0	0	1	26	0
aTumble	0	0	0	0	0	0	0	0	27

Table 4.1: Confusion Matrix

	aBend	aJump	aSkip	aRun	aWalk	aKick	aSquat	aSit	aTumble
Agent 1									
Agent 2		3							
Agent 3					4,4				
Agent 4		3							
Agent 5									
Agent 6									
Agent 7							8		
Agent 8							9	7	
Agent 9							9,9		

Table 4.2: Misclassified actions for each subject: the numbers indicate those actions chosen incorrectly (1: aBend, 9: aTumble)

MDK performance is slightly higher than MDEP, due to the fact that the most characteristic frames were used for comparison. At $m^U = 20$, the recognition rate using MDK reaches 96%, and it stabilizes. Therefore, by incorporating a selection criteria based on likelihood, recognition rate is increased. On the other hand, the MDRP criteria provides low recognition rates: the random selection of samples produces abrupt changes in the recognition rate, while the MDRP performance is about 10% and 30% below than using the MDK criterion.

In order to analyze why the recognition rate does not reach 100%, the confusion matrix is shown in Table 4.1 for $m^U = 20$ and using the MDK classifier. Misclassified actions were *aJump*, *aSkip*, *aSquat* and *aWalk*. Worst results were obtained by the *aSquat* action. This is due to the fact that such an action shares many similar postures with the *aSit* and *aTumble* actions.

Misclassified actions are shown per agent in Table 4.2 to demonstrate the singularity and variability of human performances. Three main conclusions can be derived. First, an action can be performed in a different way by the same agent. Consequently, there are misclassified performances per agent and action. Second, an action is performed in a different way by different agents. For example, two performances of the *aSquat* action are mistaken for $Agent7$ and $Agent8$ with the *aSit* and *aTumble* actions,

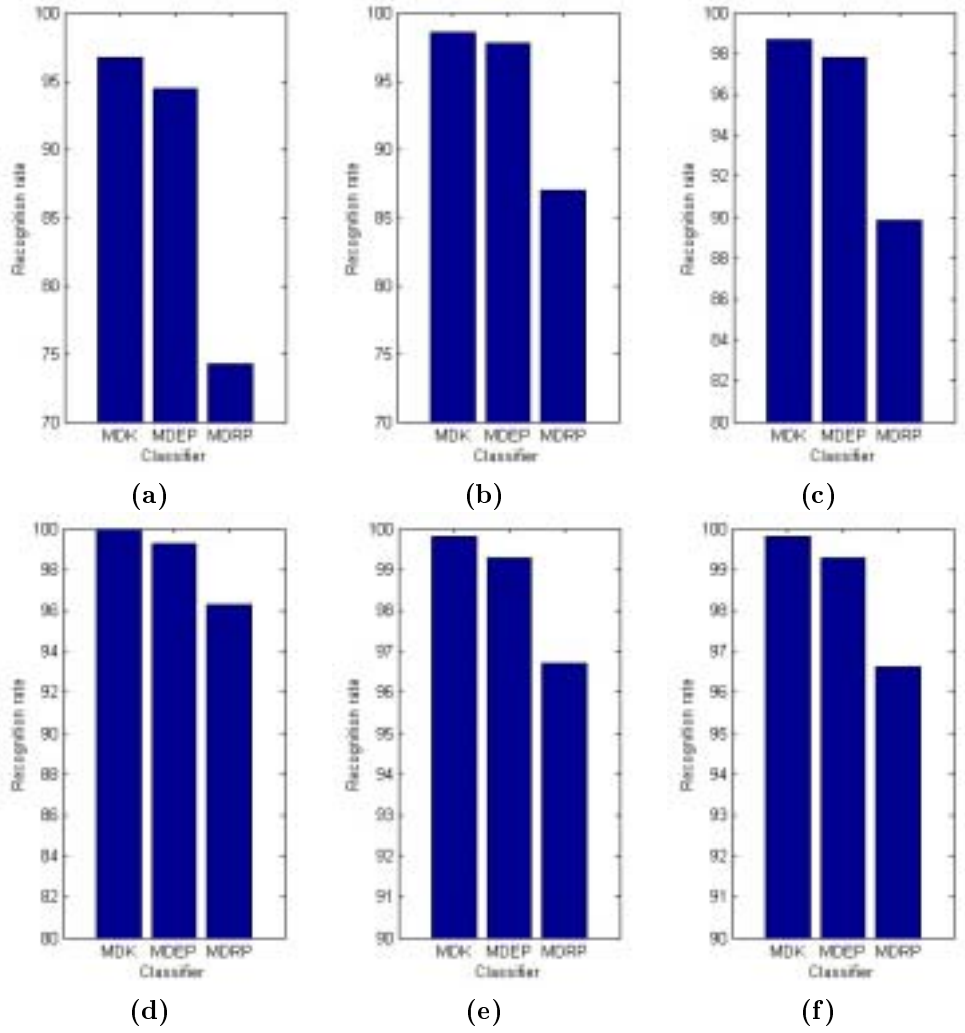


Figure 4.6: Recognition performance by considering the first three choices, and by varying the dimensionality of the *UaSpace*, specifically for $m^U = 5$ **(a)**, $m^U = 10$ **(b)**, $m^U = 15$ **(c)**, $m^U = 20$ **(d)**, $m^U = 25$ **(e)**, and $m^U = 30$ **(f)**.

respectively. Lastly, performances of a given action may exhibit postures very similar to another action, even whether such performances are performed by different agents. Thus, three *aSquat* performances of *Agent*8 and *Agent*9 are mistaken with *aTumble*. Also, two *aJump* performances of *Agent*2 and *Agent*4 are mistaken with *aSkip*.

When the correct action class is allowed to be within the first three choices, the recognition rate obviously increases. Fig. 4.6 shows the recognition performance for the three different classifiers, by varying the number of eigenvectors m^U. At $m^U = 20$, the recognition rate using MDK reaches 100%, and it stabilizes for higher dimensions.

To conclude, a balanced trade-off between recognition rate and reconstruction error is found by considering 20 eigenvectors for modeling the *UaSpace*. Moreover, this dimensionality reduction provides a human action representation which is computationally efficient.

4.2 Human Action Synthesis

Commonly, two steps are required for human animation [108]. First, the goal is to generate a rough version of the action required by the animator, which is independent of the animated character. As a result, a *prototypical* action is synthesized. Subsequently, the task of the animator is to *refine* the resulting sequence by adding small movement details and other corrections, in order to incorporate specific features for a given virtual character, such as realism or grace. Thus, different synthetic actors will exhibit different movements while performing the same action.

In this section, we confront the first task of generating a rough, primal sketch of an animated sequence, during which any (previously learnt) action is automatically synthesized. For this purpose, the human action model developed in the last chapter is exploited.

In order to design a human action synthesis procedure, we consider the *aSpace* of the action to be synthesized, and the human body postures of the prototypical performance of the action. This information is already embedded in our human action model $\mathbf{\Gamma}^A$, which defines the *aSpace*, the *key-frames*, and the *p–action* manifold of the action $\mathbf{A}$:

$$\mathbf{\Gamma}^A = (\mathbf{\Omega}^A, \mathbf{K}^A, \mathbf{f}^A). \tag{4.17}$$

Note that this representation embeds benefits for animation purposes. As the *p–action* is based on characteristic postures, i.e., the key-frame set, we guarantee that the viewer can recognize which action is being animated. Also, the interpolation nature of the *p–action* provides smoothness to the variation of the synthesized human posture. Moreover, the *p–action* is parameterized by the pose p, thus attaining a mechanism of control over our action representation. In fact, the parameter p specifies the temporal order of the postures during a new animated sequence.

However, in order to cope with the temporal variability inherent in action performances, the speed at which the parameter p is increased should be under control. This is not straightforward due to the unknown relationship between a change in p and the corresponding change in distance along the curve $\mathbf{f}^A$. Thus, an algorithm for stepping along the curve in equal increments is required, but also for speeding up and slowing down.

Next, we enhance our action model $\mathbf{\Gamma}^A$ by incorporating existing animation strategies which allow to control the speed at which synthetic performances are generated. Once the speed of the action performance is established, we describe a procedure

based on the *p–action*, which synthesizes a virtual agent performing an specific action. Lastly, we exploit the *UaSpace* representation in order to synthesize activities, that is, actions plus the *transitions* between consecutive actions. Within the *UaSpace*, interpolation between action manifolds will generate smooth and realistic *transitional* human postures.

4.2.1 Arc Length Parameterization of *p–actions*

As described before, by varying the parameter p of $\mathbf{f}^A$ by a constant amount does not mean that the resulting values will vary in a constant amount. Thus, if the human body posture is being interpolated by varying p at a constant rate, the human body postures generated will not necessarily represent a constant speed.

The control of this speed of interpolation is very important in creating effective animation. In fact, the speed of the interpolation may be non-linear, so that the change begins slowly, speeds up, and slows down into the next key-frame, for example.

Speed control is achieved by considering the distance along the curve of interpolation or, in other words, by establishing a reparameterization of the *p–action* by arc length [58]. Thus, the animator can specify the relative velocities at which the body posture variation occurs. For example, stepping along the curve at equally spaced intervals of arc length will result in a synthetic action at a constant speed.

So consider the manifold $\mathbf{f}^A(p)$ of an action model $\mathbf{\Gamma}^A$. We require a reparameterization in terms of the arc length s. Usually, this step is difficult or impossible for many types of curves, so several approximation techniques have been developed; see [108] for details. For our purposes, to establish the relationship between parametric values and arc length is enough. That is, it is not worth to require an arc length parameterization where a unit change in the parameter value results in a unit change in the curve length [124]. By describing how the length varies with the parametric variable, we could determine afterwards from that curve an arc length parameterization.

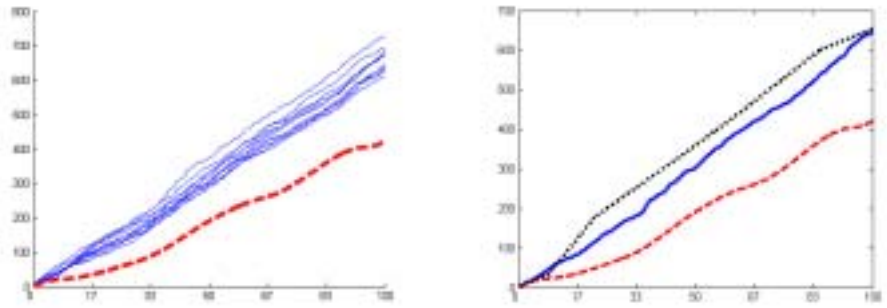
The easiest way for establishing the correspondence between p and s is to sample the curve at a multitude of parametric values p. Subsequently, their corresponding arc length values are then estimated by computing the length of the curve between adjacent points, as described next.

The length s of a curve from a point p_1 to p_2 is found by evaluating the arc length integral:

$$s = \int_{p_1}^{p_2} \left| \frac{d\mathbf{f}^A}{dp} \right| dp, \tag{4.18}$$

where:

$$\frac{d\mathbf{f}^A}{dp} = 3\ \mathbf{B}^A p^2 + 2\ \mathbf{C}^A p + \mathbf{D}^A, \qquad p \in [0,1]. \tag{4.19}$$



(a) **(b)**

Figure 4.7: Manifolds parameterized by arc length (parametric value p vs. arc length s). Left figure **(a)** represents several performances parameterized by arc length (blue solid lines). The red dashed curve represents the *p–action*, with a reduced total length due to the interpolation process between key-frames. Right figure **(b)** represents the manifold $\mathbf{f}^A(p)$ (red dashed line), a performance (blue solid line), and the same performance sampled at a different speed (black dotted line).

Consequently, the total length s_{end} of the curve is derived as:

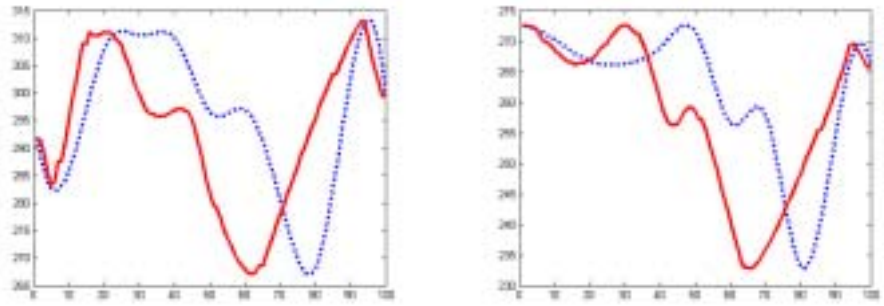
$$s_{end} = \int_0^1 \left| \frac{d\mathbf{f}^A}{dp} \right| dp. \tag{4.20}$$

Therefore, consider $\mathbf{p} = \{p_1, p_2, ..., p_n\}, p_i \in [0,1]$ the parametric values to be sampled and selected in increasing order. For each p_i, its corresponding arc length $s_i \in [0, s_{end}]$ is calculated using Eq. 4.18. This calculation is achieve recursively [108, 124]:

$$\begin{aligned} s_1 &= 0, \\ s_i &= s_{i-1} + \left| \frac{d\mathbf{f}^A(p_i)}{dp} \right| (p_i - p_{i-1}), \qquad i = 2, ..., n. \end{aligned} \tag{4.21}$$

As this is done, a table $\tau(i) =< p_i, s_i >$ is built up of arc lengths indexed by parametric values. Note that the arc length is a strictly monotonically increasing function of $\mathbf{p}$, and when $p_n = 1$, then $s_n = s_{end}$. Fig. 4.7.**(a)** represents several performances parameterized by arc length. Variation of the total arc length of such performances is due to the different postures exhibited. The red dashed curve represents the *p–action*, with a reduced total length due to the interpolation process between key-frames: the *p–action* manifold is smoother than any performance manifold.

The table τ is then searched in order to obtain the arc length s_i given a parametric value p_i. If the parametric value is not present in this table, the searched s for



(a) **(b)**

Figure 4.8: Comparison of both parameterizations (parametric value p and arc length s) for the *aBend p–action*. Left figure **(a)** represents the variation of the angle of the left lower arm when the *p–action* is sampled using p (blue dotted line) and s (red solid line) as parametric value. Right figure **(b)** corresponds to the angle of the left lower leg.

arc length value is interpolated between the arc lengths corresponding to entries on either side of the given parametric value. The reverse situation, i.e. how to obtain the parametric value corresponding to a given an arc length, is handled in a similar manner.

Once the *p–action* has been parameterized by arc length, it is possible to control the speed at which the curve is traversed. Fig. 4.7.**(b)** shows a new performance sampled at different speeds (blue solid and black dotted lines). For example, sampling at equally spaced values of arc length will evolve to constant-velocity movement.

In order to compare both parameterizations, arc length values are normalized to a range of unit length by applying:

$$s_i = \frac{s_i}{s_{end}}, \tag{4.22}$$

which will be still referred to as arc length. Fig. 4.8 shows the elevation values of the left lower arm and left lower leg during the *aBend p–action*, as a result of choosing equally spaced samples using p and s parameterizations. By considering the pose parameterization, the temporal evolution of the posture depends on the speed at which the performances have been recorded. By considering the arc length parameterization, the temporal evolution of the posture is achieved to occur at any user-specified speed.

However, speed is defined in the time space, which differs from the curve space. So it is required to relate arc length with time, that is, to produce a distance-time function $\eta(t)$,

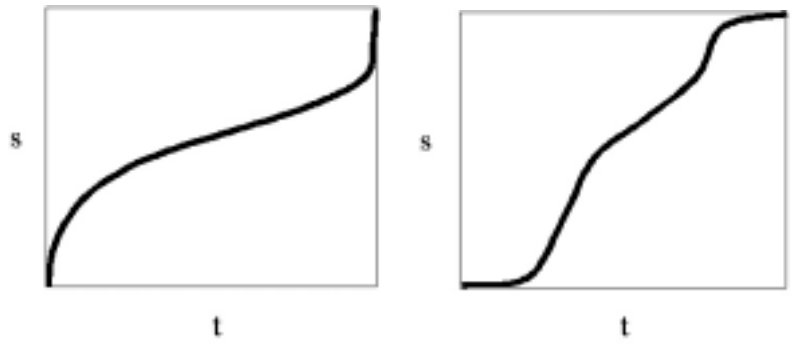


Figure 4.9: Sample distance-time functions. Left figure **(a)** represents a performance which starts and ends abruptly. Right figure **(b)** shows a performance which smoothly starts and stops.

$$\eta : T \longrightarrow S, \qquad \begin{aligned} T &= \{0, ..., t_{end}\}, \\ S &= \{0, ..., s_{end}\}. \end{aligned} \tag{4.23}$$

Therefore, given the temporal instant t_q, the corresponding distance traveled along the space curve, i.e. s_q, is obtained using $\eta(t_q)$. Subsequently, using the arc length table computed before, the final parametric value p_q is found.

Usually, distance-time functions are built analytically or by using a curve-editing tool. However, several assumptions are required to be contemplated [108]. For example, any distance-time function should be monotonic in t and continuous. In addition, it is commonly assumed that the entire arc length of the curve should be traversed during the given total time. This means that:

$$\begin{aligned} \eta(0) &= 0, \\ \eta(t_{end}) &= s_{end}, \end{aligned} \tag{4.24}$$

where $s_{end} = 1$ due to the normalization step described above, and $t_{end} = 1$. Obeying these normalization steps involves that distance-time curves can be reused with other *p–action* curves.

Examples of analytical distance-time functions are shown in Fig. 4.9. Abrupt starts and terminations correspond to discontinuities in speed or vertical tangents at the distance-time functions. Likewise, smooth starts and terminations correspond to horizontal tangents. Note that standard interpolation techniques can be used to aid the generation of distance-time curves.

For our experiments, we assume a constant increasing distance-time function or, in other words, a constant speed for synthesized performances.

4.2.2 Procedure for Human Action Synthesis

Many animation techniques are based on *key-frames*, in which animators define the key frames of the sequence to be animated [108]. In hand-drawn animation, the intermediate frames are drawn by mentally inferring the action between the keys. In computer animation, the term key-frame is applied to any variable whose value is set at specific key frames and from which values for the intermediate frames, called *in-betweens*, are obtained by smoothly interpolating the key-frames themselves when rendering.

Likewise, our human action model $\mathbf{\Gamma}^A$ embeds the *key-frame* set $\mathbf{K}^A$. As key-frames correspond to the most characteristic body postures for a given action, interpolation between key-frames results in a manifold which represent the prototypical action performance or *p–action*. As a result, the break points of this manifold within the *aSpace* correspond to the projections of the key-frames, and the manifold is made up of several segments. Each segment represents how the body posture evolves from one key-frame to its next in temporal order. Thus, the *p–action* consists of the key-frames plus the transition between these key-frames, i.e., the *in-betweens*.

As the *in-betweens* of the *p–action* have been obtained by interpolation, we achieve smooth variation of the body posture from one time step to the next. Thus, the action is synthesized in a realistic and visually pleasing manner. Also, as in-betweens actually lie in the *p–action*, the action is satisfactorily synthesized by moving along that manifold. And this is possible, due to the fact that the *p–action* is parameterized by the pose p. Thus, the temporal order of the synthetic postures during a new animated sequence can be established.

At this point, the speed at which synthetic actions are synthesized should be addressed. As the arc length parameterization allows speed control, a distance-time function will determine the sampling rate of the *p–action* manifold in terms of arc length.

So, given a human action model $\mathbf{\Gamma}^A = (\mathbf{\Omega}^A, \mathbf{K}^A, \mathbf{f}^A)$, the idea is to sample the manifold $\mathbf{f}^A$ at different $\mathbf{p} = \{p_1, p_2, ..., p_n\}, p_i \in [0, 1]$. This set is obtained by considering the entries $\mathbf{s} = \{s_1, s_2, ..., s_n\}$ of the arc length table $\tau(i) =< p_i, s_i >$, where each s_i has been obtained using the distance-time function $\eta(t_i), i = 1, ...n$. As a result, a sequence of projections are obtained, each projection $\mathbf{y}_i$ corresponding to a point of the *p–action* manifold:

$$\mathbf{f}^A(\mathbf{p}) = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n], \qquad \mathbf{y}_i \in \mathbf{f}^A(p), \tag{4.25}$$

where each projection correspond to a human posture configuration $\mathbf{x}_i$:

$$\mathbf{x}_i = \sum_{j=1}^{m} \mathbf{e}_j \mathbf{y}_{ij} + \bar{\mathbf{x}}, \qquad i = 1, ..., n, \tag{4.26}$$

where $\mathbf{E} = (\mathbf{e}_1, ..., \mathbf{e}_m)$ and $\bar{\mathbf{x}}$ are defined in $\mathbf{\Omega}^A$. Therefore, the i-th frame of the new

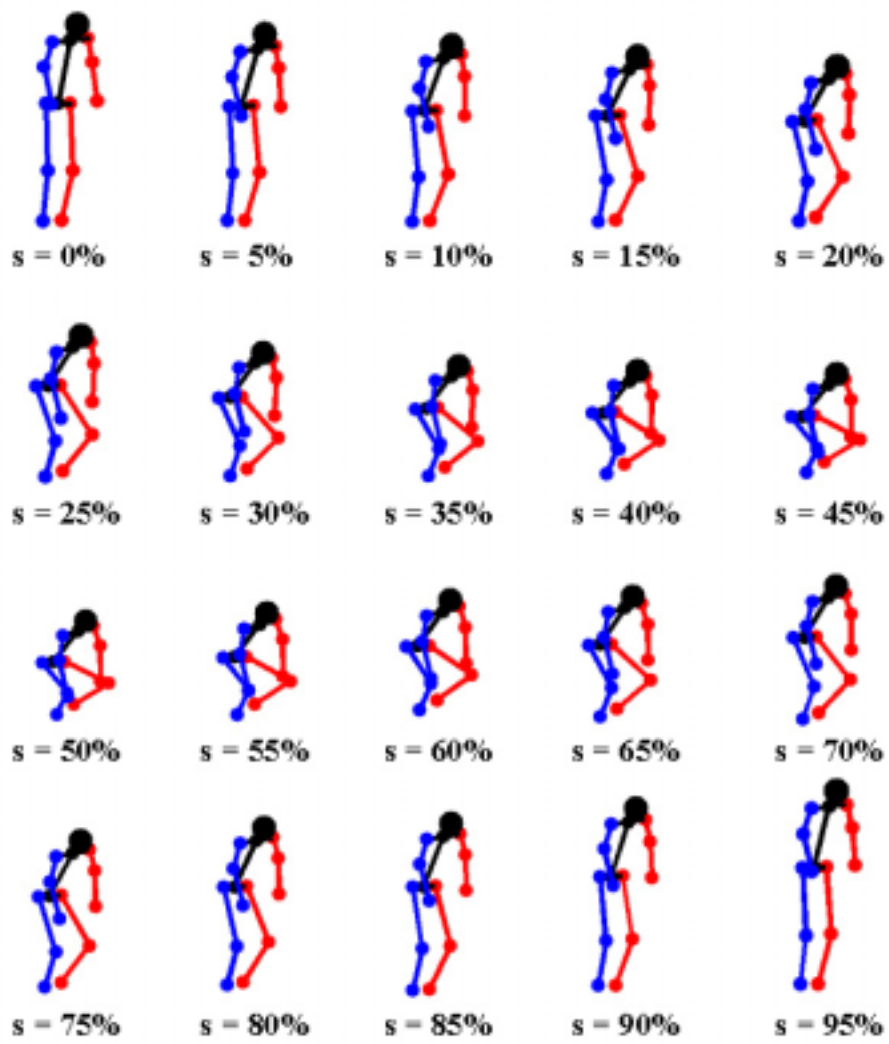


Figure 4.10: Stick figure models generated by varying the arc length s in the *aSquat aSpace*. Due to the cubic-spline interpolation step, the posture variation over time is achieved to occur smoothly.

sequence $\tilde{\mathbf{A}}$ synthesizes the i-th human posture configuration:

$$\tilde{\mathbf{A}} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}. \tag{4.27}$$

To conclude, a 3D human body representation can be built from each resulting $\mathbf{x}_i$ in order to generate the synthetic sequence, see Fig. 4.10. The complete algorithm is shown in Table 4.3.

Table 4.3: Human Action Synthesis Algorithm

Input:

- Action to be synthesized, i.e. $\mathbf{A}$.
- Number of frames of the synthetic sequence, n.
- Selected distance-time function $\eta(t)$.
- Arc length table τ.
- Action model $\mathbf{\Gamma}^A = (\mathbf{\Omega}^A, \mathbf{K}^A, \mathbf{f}^A)$.

Algorithm:

1. Generate a sequence of temporal instants $\mathbf{t} = \{t_1, ..., t_n\}$ in increasing order, where $t_1 = 0$ and $t_n = 1$.
2. Obtain their corresponding arc length value $\mathbf{s} = \{s_1, ..., s_n\}$, where $s_i = \eta(t_i)$.
3. Compute their corresponding posture-variation parameter $\mathbf{p} = \{p_1, ..., p_n\}$, using the arc length table,
$$\tau(i) =< p_i, s_i >, \qquad s_i \in \mathbf{s}.$$
4. Obtain their corresponding projection coordinates in the *aSpace* $\mathbf{\Omega}^A$, by using
$$\mathbf{y}_i = \mathbf{f}^A(p_i), \qquad \mathbf{y}_i = [y_1, ..., y_m]^T, \quad i = 1, ..., n.$$
5. Calculate the original human posture configuration $\mathbf{x}_i$, as follows:
$$\mathbf{x}_i = \sum_{j=1}^{m} \mathbf{e}_j \mathbf{y}_{ij} + \bar{\mathbf{x}}, \qquad i = 1, ..., n,$$
where $\mathbf{E}$ and $\bar{\mathbf{x}}$ are defined in $\mathbf{\Omega}^A$.

Output:

- The sequence of synthetic human body posture configurations, $\tilde{\mathbf{A}} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$.

Note that the *p–action* represents a generic human action performance, which embeds the most characteristic postures obtained from several performances of different agents. Consequently, specific action characteristics of individual performers are removed. However, we can restrict our action representation to model only those actions performed by a given agent, thus obtaining specific action patterns for *that* agent. That means, we can compute a different *aSpace* for each agent. Therefore, we could be able to synthesize several agents which exhibit differences while performing the same actions.

4.2.3 Procedure for Human Activity Synthesis

An activity is defined as a performance involving two or more actions, plus the transitions between consecutive actions. In order to synthesize activities, two issues should be addressed. On the one hand, to represent all the actions involved in the activity in an unique feature space. On the other hand, to generate those human postures which correspond to the transitions between consecutive actions. Note, however, that these *transitional* postures are not present in the learning set.

To confront these issues, we exploit the *UaSpace* representation presented before. Thus, all the actions involved in any activity sequence are represented in an unique feature space as parametric manifolds. Subsequently, interpolation between two projections of different action manifolds constitutes a path along which the human posture *varies* from a posture belonging to an action to another posture belonging to the next action of the activity sequence. Next, the procedure is described.

The training data $\mathbf{A}^U$ corresponds to the f human postures of the complete set of actions $\mathbf{U} = \{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_u\}$ to be learnt:

$$\mathbf{A}^U = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_f\}, \tag{4.28}$$

where each $\mathbf{x}_i$ corresponds to a human posture configuration. No learning samples corresponding to action transitions are included. Once the *UaSpace* is computed, $\mathbf{\Omega}^U = (\mathbf{E}^U, \mathbf{\Lambda}^U, \bar{\mathbf{x}}^U)$, each action $\mathbf{A}_i$ is modeled within the *UaSpace* as:

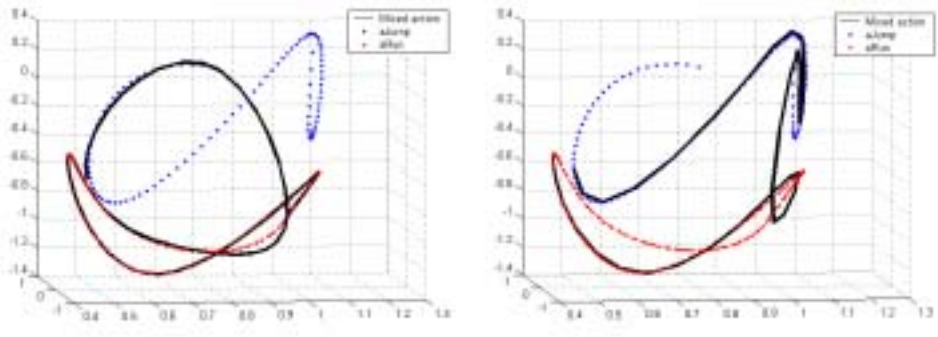
$$\mathbf{\Gamma}^{A_i} = (\mathbf{\Omega}^U, \mathbf{K}^{A_i}, \mathbf{f}^{A_i}). \tag{4.29}$$

Each modeled posture $\hat{\mathbf{x}}$ is found as a combination of:

$$\hat{\mathbf{x}} = \bar{\mathbf{x}}^U + \mathbf{E}^U\mathbf{b}, \tag{4.30}$$

where $\mathbf{b} = (b_1, ..., b_{m^U})^T$ is a vector of weights. Each weight is computed within suitable limits in order to generate new acceptable postures, similar to those in the training set. These limits are related to the variance of each variation mode, so each weight b_k is restricted to lie within:

$$-3\sqrt{\lambda_k^U} \leq b_k \leq 3\sqrt{\lambda_k^U}. \tag{4.31}$$



(a) **(b)**

Figure 4.11: The manifold corresponding to an activity (composed of *aRun*, *aJump*, and *aRun* again) is presented (solid curve). Figure (a) shows the resulting manifold by varying the arc length s from $s = 0\%$ to $s = 50\%$, and figure (b) shows the variation of s from $s = 50\%$ to $s = 90\%$. Due to the interpolation step, transitions between actions occur smoothly.

All invalid postures are excluded automatically by obeying these limits. That is, the *UaSpace* can be restricted to model only plausible human posture variations. Moreover, as the training data is obtained from a motion capture system, a realistic model of variation is provided.

Consequently, interpolation between two different postures within the *UaSpace* will generate a physically meaningful body posture variation only if Eq. (4.31) is obeyed. In other words, despite the fact that no postures corresponding to action transitions have been included in the training data $\mathbf{A}^U$, interpolation involves that the human posture variation from one action to another can be smoothly and automatically synthesized.

Two actions in the *UaSpace* are presented as dot curves in Fig. 4.11. Once the key-frames are selected, interpolation between these key-frames provides the body posture to vary smoothly from one action to the next in temporal order. Consequently, transitions between actions are feasible in a continuous and realistic manner.

Obeying the constraints described before, an activity $\mathbf{V}$ composed of M actions $\mathbf{V} = \{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_M\}$ is synthesized by considering the temporal-ordered set of key-frames $\mathbf{K}^V$ of the actions of $\mathbf{V}$:

$$\mathbf{K}^V = \{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_t\}, \qquad \mathbf{k}_j \in \mathbf{f}^{A_i}. \tag{4.32}$$

These selected key-frames represent the core of the new activity sequence. Specifically, they constitute the control points of a new manifold $\mathbf{f}^V$ which represents the sequence of human postures of the *prototypical performance* of the activity. This manifold, i.e. the in-betweens of the activity sequence, is generated by interpolation between key-frames.

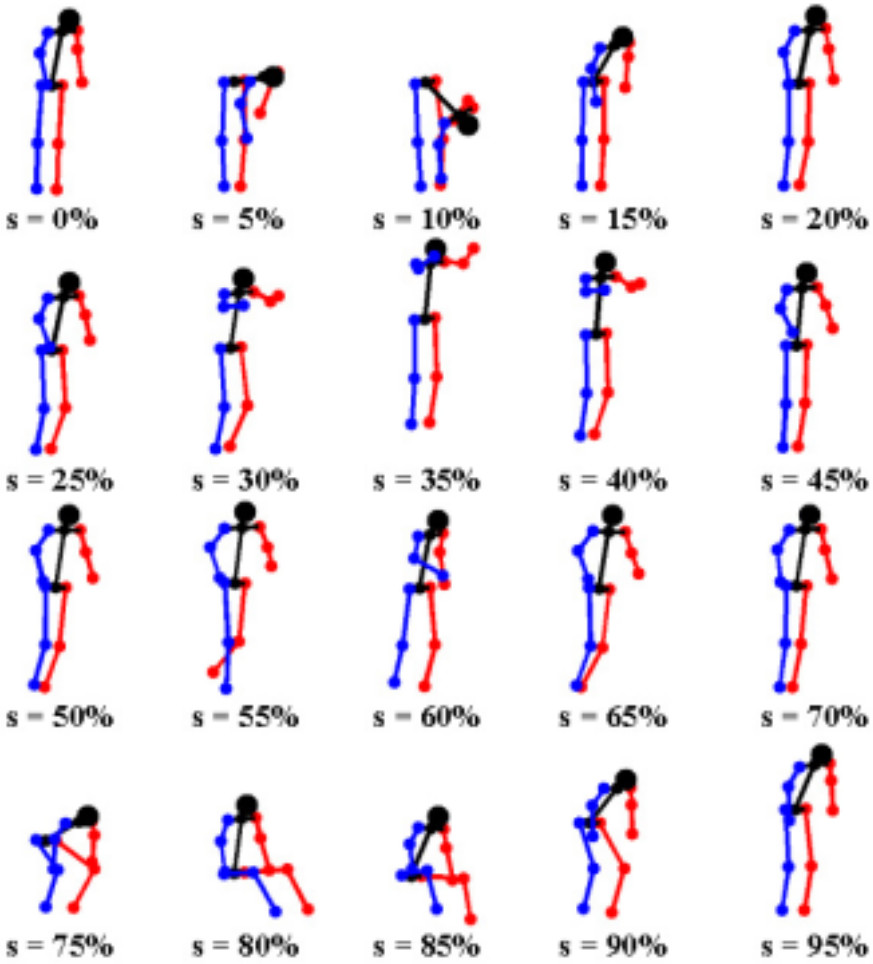


Figure 4.12: Activity synthesis: by varying the arc length s of an activity manifold in the *UaSpace*, the sequence of human postures exhibited during several mixed actions is generated. First row corresponds to *aBend*, second row to *aJump*, third row to *aRun*, and fourth row to *aTumble*. Note that $s = 20\%$, $s = 50\%$ and $s = 70\%$ correspond to postures of transitions between actions.

To summarize, the activity model is defined as:

$$\mathbf{\Gamma}^V = (\mathbf{\Omega}^U, \mathbf{K}^V, \mathbf{f}^V). \tag{4.33}$$

The algorithm described in Table 4.3 is applied here, too, to perform human activity synthesis by providing the activity model $\mathbf{\Gamma}^V$. The prototypical performance of an activity is shown in Fig. 4.12, which is composed of four different actions,

namely *aBend*, *aJump*, *aRun* and *aTumble*. Note that transition postures between consecutive actions, which were not learnt, are synthesized due to the interpolation nature of the activity manifold computation.

4.3 Human Action Analysis

There are several applications which are interested in human movement analysis. Training of physically disabled persons and athletic performance analysis, for example, require of a similarity measure between different performances of the same action in order to improve the efficiency of its execution. In other words, human movement analysis is considered here as a comparison framework which helps to determine "how well" a performance has been executed with respect to a reference action model.

Using our human action model, a synchronization algorithm is first presented to establish a comparison framework between the *p-action* representation and any performance of the same action. For this purpose, we require to attain independence from the speed at which performances are played.

Synchronization of performances allows to compare the variation of the angles of specific joints during several performances. Thus, the evolution of harmed articulations, like the knee or the elbow, can be studied during rehabilitation processes, for example. For this purpose, we describe lastly an analysis procedure applied to the walking action.

4.3.1 Human Performance Synchronization

The idea of synchronization arises from the assumption that any performance of a given action should present the key-frames of such an action. Therefore, the key-frame set is considered as the reference postures in order to adjust or *synchronize* any new performance to our action model. By considering the arc length parameterization, the aim is to sample the new performance and the *p–action* so that the key-frames are equally spaced in both manifolds.

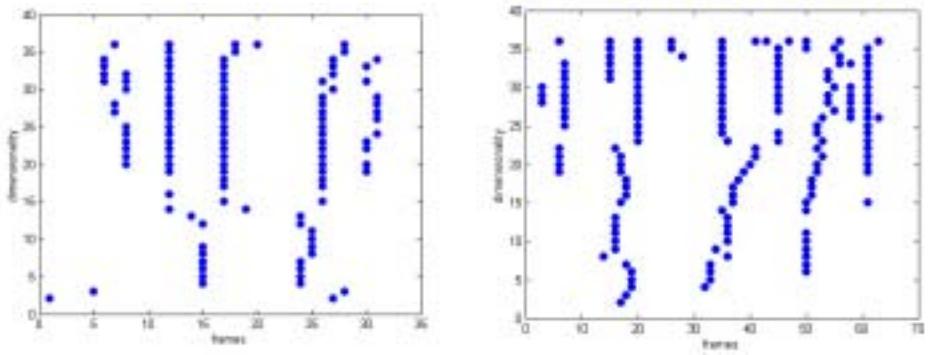
Consider a performance $\mathbf{H}$ of a given action $\mathbf{A}$. Such a performance is composed of a sequence of f_H human postures, $\mathbf{H} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{f_H}\}$, to be synchronized with respect to the $\mathbf{A}$ action model:

$$\mathbf{\Gamma}^A = (\mathbf{\Omega}^A, \mathbf{K}^A, \mathbf{f}^A). \tag{4.34}$$

We first build the human performance representation for $\mathbf{H}$:

$$\mathbf{\Gamma}^H = (\mathbf{\Omega}^A, \mathbf{K}^H, \mathbf{f}^H), \tag{4.35}$$

that is, the key-frame set $\mathbf{K}^H$ obtained by applying the key-frame selection criterion defined in the last chapter, and the manifold $\mathbf{f}^H$ obtained by interpolation between



(a) **(b)**

Figure 4.13: The selected key-frames of a single performance is shown for the *aJump* **(a)** and *aSkip* **(b)** actions. Dots represent the key-frames, i.e., the maxima of the distance function defined in chapter 3, by varying the *aSpace* dimensionality.

the key-frames.

Unfortunately, to determine the key-frame set $\mathbf{K}^H$ by using only the postures of a single performance is highly unstable. For example, the key-frames selected for an *aJump* and an *aSkip* performances are shown in Fig. 4.13 by varying the *aSpace* dimensionality. Due to the high degree of variability inherent in individual action performances, random key-frames can be found due to *noisy* movements. Thus, despite of the fact that some key-frames remain stable over dimensionality, additional key-frames appear isolated. Consequently, the key-frame selection criterion used in the last chapter will not be applied to single performances.

Alternatively, we can exploit the fact that we already know which action has been performed. So we next utilize the information embedded in the action model $\mathbf{\Gamma}^A$ in order to select the key-frame set $\mathbf{K}^H$. For this purpose, we adapt the human action recognition procedure presented before in order to find the key-frames $\mathbf{K}^A$ in the performance manifold $\mathbf{f}^H$. Thus, we define the distance between a key-frame $\mathbf{k}_i^A$ of $\mathbf{K}^A$, and a projection $\mathbf{y}_j^H$ of the performance manifold $\mathbf{f}^H$ as:

$$d(\mathbf{k}_i^A, \mathbf{y}_j^H) = \left\| \frac{\mathbf{k}_i^A}{\| \mathbf{k}_i^A \|} - \frac{\mathbf{y}_j^H}{\| \mathbf{y}_j^H \|} \right\|. \tag{4.36}$$

As the distance between two projections in the *aSpace* is a measure of correlation, the nearest projection $\mathbf{y}_j^H = \mathbf{f}^H(p_{\mathbf{k}_i}^H)$ to the i-th key-frame of the *p–action* $\mathbf{k}_i^A$ determines the i-th key-frame of the performance manifold $\mathbf{k}_i^H$:

$$p_{\mathbf{k}_i}^H = \arg\min_j d(\mathbf{k}_i^A, \mathbf{y}_j^H), \qquad j \in [1, h], \quad i = 1, \ldots, k. \tag{4.37}$$

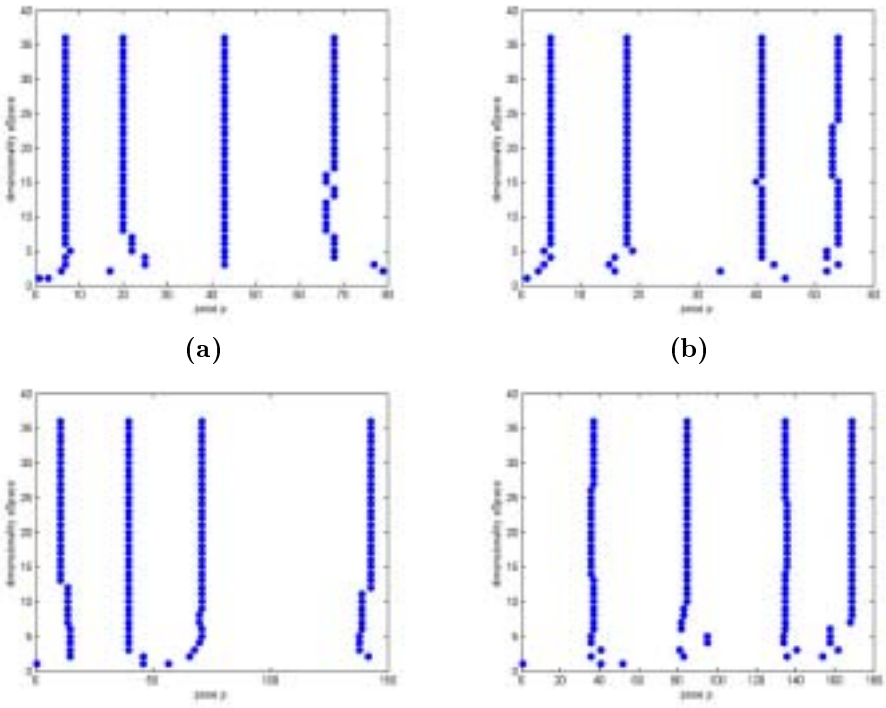


Figure 4.14: Selected key-frames are shown for two *aSkip* performances in **(a)** and **(b)**, and for two *aTumble* performances in **(c)** and **(d)**. Dots represent the key-frames, i.e., those projections in the performance manifold, whose distance is minimum to the key-frame set of the action model.

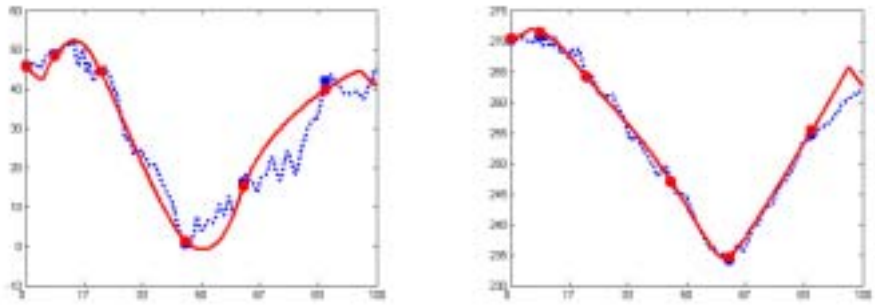
where $\mathbf{p}^H_{\mathbf{k}_i}$ refers to the pose parameter value which determines the i-th characteristic posture $\mathbf{k}^H_i$ in the performance manifold $\mathbf{f}^H$.

Fig. 4.14 shows the pose values of the selected key-frames for the *aSkip*, and *aTumble* performances by varying the dimensionality m of their corresponding *aSpaces*. As expected, the key-frame identification process stabilizes in high dimensions. The parametric value p is not normalized here to illustrate that the position of the key-frames within a performance sequence is highly variable.

Once the set of key-frame projections is established:

$$\mathbf{f}^H(\mathbf{p}^H_{\mathbf{k}}) = \{\mathbf{k}^H_1, \mathbf{k}^H_2, ..., \mathbf{k}^H_k\}, \tag{4.38}$$

we compute the arc lengths corresponding to the key-frames found in the performance manifold:



(a) (b)

Figure 4.15: Synchronization of a new performance with the *aBend p–action* $\mathbf{f}^A(p)$. The variation of the elevation angles for the right arm (a) and for the right upper leg (b) during a new performance (dotted line) and during the *p–action* (solid line) is shown. By adjusting the key-frames (dots), the correspondence for both manifolds is established.

$$\mathbf{s}_{\mathbf{k}}^H = \{s_{\mathbf{k}_1}^H, s_{\mathbf{k}_2}^H, ..., s_{\mathbf{k}_k}^H\}, \tag{4.39}$$

The next step is centered on determining the arc length values $\mathbf{s}_{\mathbf{k}}^A$ which correspond to the pose parameter values $\mathbf{p}_{\mathbf{k}}^A$ of the key-frame set $\mathbf{K}^A$ in the *p–action* $\mathbf{f}^A$:

$$\mathbf{s}_{\mathbf{k}}^A = \{s_{\mathbf{k}_1}^A, s_{\mathbf{k}_2}^A, ..., s_{\mathbf{k}_k}^A\}. \tag{4.40}$$

Subsequently, we generate a sequence of consecutive arc length values in the new performance so that $\mathbf{s}_{\mathbf{k}}^H$ be equally spaced as $\mathbf{s}_{\mathbf{k}}^A$. That is, a sequence of r consecutive equally spaced arc length values is taken between $s_{\mathbf{k}_i}^A$ and $s_{\mathbf{k}_{i+1}}^A$. Similarly, by considering r consecutive equally spaced arc length values between $s_{\mathbf{k}_i}^H$ and $s_{\mathbf{k}_{i+1}}^H$, we are actually sampling the new performance manifold so that the key-frames of both manifolds are time-coincident.

Fig. 4.15 shows an example of synchronization between a new performance and the *aBend p–action*. The variation of the angles of specific limbs of the human body model during a performance can be compared with respect to the *p–action*. Once the key-frames establish the correspondences for both manifolds, we modify the speed at which a performance is sampled so that its key-frames coincide in time with the key-frames of the *p–action*.

As a result of such a synchronization process, differences between a performance and the prototypical action can be derived by analyzing the resulting angle variation curves. Such differences can be associated with natural language terms related to speed, naturalness, or suddenness, for example. These terms can be used to enhance the description of a recognized action, as discussed next.

4.3.2 Analysis of Human Walking

Computational models of action style are relevant to several important application areas [41]. On the one hand, it helps to enhance the qualitative description provided by the human action recognition module. Thus, for example, it is important to generate style descriptions which best characterize an specific agent for identification purposes. Also, the style of a performance can help to establish ergonomic evaluation and athletic training procedures. Another application domain is to enhance the human action library by training different action models for different action styles, using the data acquired from a motion capture system. Thus, it should be possible to re-synthesize human performances exhibiting different postures.

In the literature, the most studied human action is *walking*. Human walking is a complex, structured, and constrained action, which involves to maintain the balance of the human body while transporting the figure from one place to another. The most exploited characteristic is the cyclic nature of walking, because it provides uniformity to the observed performance. In this section, we propose to use our human action model in the study of the *style* inherent in human walking performances, such as the gender, the walking pace, or the effects of carrying load, for example.

Specifically, we use the *aSpace* representation to establish a characterization of the walking style in terms of the gender of the walker. Such a characterization consists of a description of the variation of specific limb angles during performances played by agents of different gender. The aim is to compare performances to describe differences between male and female walkers.

Our training data set is composed of r sequences $\mathbf{A}^W = \{\mathbf{H}_1^W, \mathbf{H}_2^W, ..., \mathbf{H}_r^W\}$, each one corresponding to a cycle or *stride* of the *aWalk* action $\mathbf{A}^W$. Three males and three females were recorded, each one walking five times in circles. Each sequence $\mathbf{H}_j^W$ of f_j frames corresponds to a sequence of human body configurations:

$$\mathbf{H}_j^W = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{f_j}\}, \tag{4.41}$$

where each $\mathbf{x}_i$ stands for the 37 values of the human body model described in the last chapter, which was composed of twelve rigid body parts (hip, torso, shoulder, neck, two thighs, two legs, two arms and two forearms). Consequently, our human performance analysis is restricted to be applied to the variation of these twelve limbs.

Once the learning samples are available, we compute the *aSpace* representation of the *aWalk* action. First, the mean human posture $\bar{\mathbf{x}}^W$ and the covariance matrix Σ^W are calculated. Subsequently, the eigenvalues λ_i^W and eigenvectors $\mathbf{e}_i^W$ of Σ^W are found by solving the eigenvector decomposition equation. The individual contribution of each eigenvector determines that 95% of the variation of the training data is captured by the thirteen eigenvectors associated to the thirteen largest eigenvalues. So the resulting *aWalk aSpace* is defined as:

$$\mathbf{\Omega}^W = (\mathbf{E}^W, \mathbf{\Lambda}^W, \bar{\mathbf{x}}^W). \tag{4.42}$$

Figure 4.16: The three most important modes of variation of the *aWalk aSpace*.

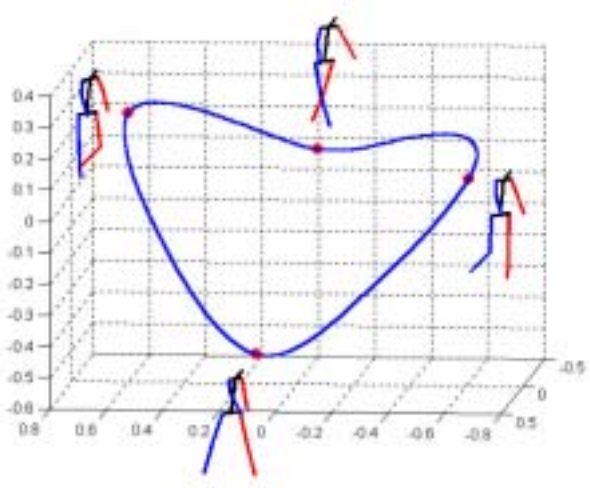
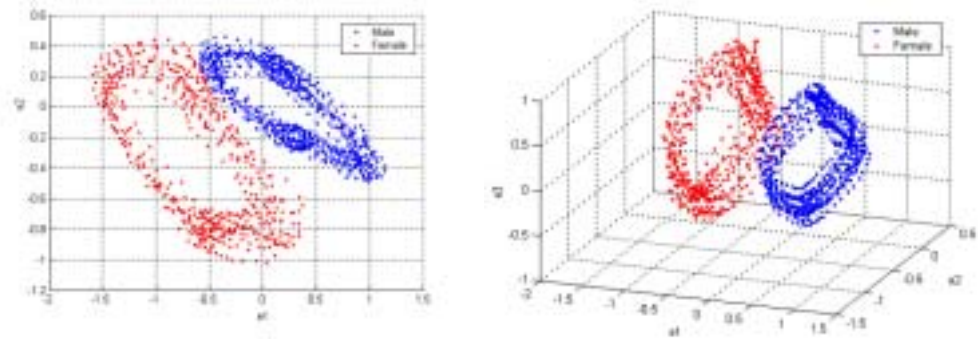


Figure 4.17: Prototypical performance manifold, or *p–action*, in the *aWalk aSpace*. Depicted human postures correspond to the key-frame set.

Fig. 4.16 shows the most relevant modes of variation of the human posture in the *aWalk aSpace*, which correspond to the three eigenvectors associated to the three largest eigenvalues. As expected, these modes of variation are mainly related to the movement of legs and arms.

Once the training human postures $\mathbf{x}_i$ are projected in the *aWalk aSpace*, the key-frame set $\mathbf{K}^W$ is found, and the *p–action* $\mathbf{f}^W$ is computed by interpolation between key-frames, see Fig. 4.17. Therefore, the *aWalk* action model is defined as:



(a) **(b)**

Figure 4.18: Male and female postures projected in the *aWalk aSpace*, by considering two **(a)** and three **(b)** eigenvectors for the *aSpace* representation.

$$\mathbf{\Gamma}^W = (\mathbf{\Omega}^W, \mathbf{K}^W, \mathbf{f}^W). \tag{4.43}$$

In order to compare performances played by male and female agents, we define two different training sets:

$$\begin{aligned} \mathbf{H}^{W_M} &= \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{f_M}\}, \\ \mathbf{H}^{W_F} &= \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{f_F}\}, \end{aligned} \tag{4.44}$$

that is, the set human postures exhibited during several *aWalk* performances for a male and a female agent, respectively. In our experiments, near 50 *aWalk* cycles per walker have been recorded. As a result, the training data is composed of near 1500 human posture configurations per agent.

Next, we project the human postures of $\mathbf{H}^{W_M}$ and $\mathbf{H}^{W_F}$ in the *aWalk aSpace*, as shown in Fig. 4.18. The cyclic nature of the *aWalk* action explains the resulting circular clouds of projections. Also, note that both performances do not *intersect*, that is, they do not exhibit the same set of human postures. This is due to the high variability inherent in human performances. Consequently, we can *identify* a posture as belonging to a male or female walker.

However, the scope of this section is not centered on determining a *discriminative* procedure between generic male and female walkers. Instead, we look for a comparison procedure to subsequently *evaluate* the variation of the angles of specific agents while performing the same action, in order to derive a characterization of the action *style*.

Following the procedure described in the last chapter, we use the projections of each walker to compute the performance representation for the male $\mathbf{\Gamma}^{W_M}$ and female $\mathbf{\Gamma}^{W_F}$ agents:

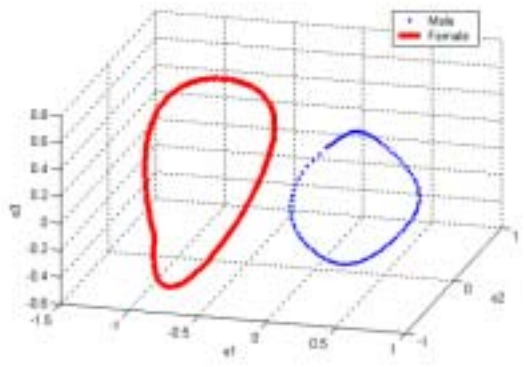


Figure 4.19: Male and female performance representations in the *aWalk aSpace*.

$$\begin{aligned} \mathbf{\Gamma}^{W_M} &= (\mathbf{\Omega}^W, \mathbf{K}^{W_M}, \mathbf{f}^{W_M}), \\ \mathbf{\Gamma}^{W_F} &= (\mathbf{\Omega}^W, \mathbf{K}^{W_F}, \mathbf{f}^{W_F}), \end{aligned} \tag{4.45}$$

where $\mathbf{f}^{W_M}$ and $\mathbf{f}^{W_F}$ refer to the male and female *p–actions*, respectively. These manifolds have been obtained by interpolation between the key-frames of their respective key-frame set, i.e., $\mathbf{K}^{W_M}$ and $\mathbf{K}^{W_F}$. These key-frames correspond to those postures in each manifold, which are most similar to the key-frame set $\mathbf{K}^W$ of the *aWalk* model $\mathbf{\Gamma}^W$. Fig. 4.19 shows the resulting *p–action* representations in the *aWalk aSpace*.

In order to compare the human posture variation for both performances, we sample both *p–actions* to describe each manifold as a sequence of projections, that is:

$$\begin{aligned} \mathbf{f}^{W_M}(\mathbf{p}) &= [\quad \mathbf{y}_1^{W_M}, \mathbf{y}_2^{W_M}, ..., \mathbf{y}_{q_M}^{W_M} \quad], \\ \mathbf{f}^{W_F}(\mathbf{p}) &= [\quad \mathbf{y}_1^{W_F}, \mathbf{y}_2^{W_F}, ..., \mathbf{y}_{q_F}^{W_F} \quad], \end{aligned} \tag{4.46}$$

where q_M and q_F refer to the number of projections considered for performance comparison. However, the sampling rate of both *p–actions* should be established in order to attain independence from the speed at which both performances have been played.

Therefore, both *p–actions* are parameterized by arc length and, subsequently, the synchronization procedure described in the last section is applied. Synchronization allows to modify the speed at which both manifolds are sampled so that their key-frames coincide in time with the key-frame set $\mathbf{K}^W$ of $\mathbf{\Gamma}^W$.

Once the male and female *p–actions* are synchronized, the angle variation for different limbs of the human body model can be analysed. Fig. 4.20.**(a)**, **(b)**, **(c)**, and **(d)** show the evolution of the elevation angle for four limbs of the human body model, namely the shoulder, torso, left arm, and right thigh, respectively.

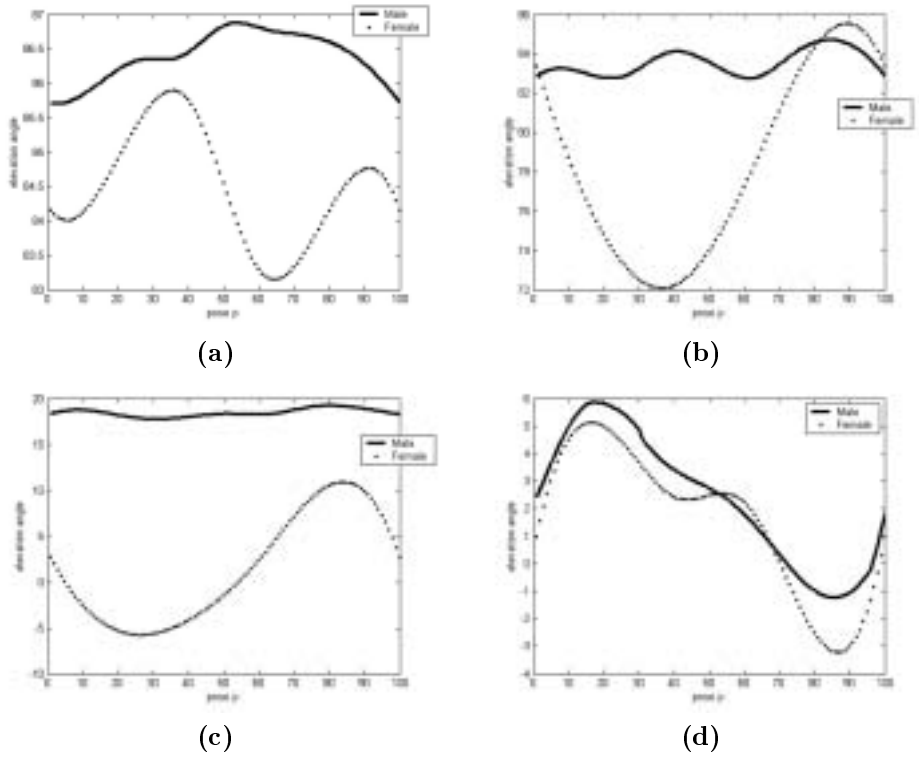


Figure 4.20: The elevation variation for the shoulder **(a)**, torso **(b)**, left arm **(c)**, and right thigh **(d)** limbs are depicted for a male and a female walker.

By comparing the depicted angle variation values of both walkers, several differences can be observed. The female walker moves her shoulder in a higher degree than the male shoulder. That is, the swing movement of the shoulder is more accentuated for the female. Also, the female bends the torso in a higher inclination degree. Therefore, the swing movement of the shoulder and torso for the male agent is less pronounced. The female walker also exhibits an emphasized swing movement in her left arm. On the contrary, the male agent does not show a relevant swing movement for his left arm. As expected, when the left arm swings backward, the right thigh swings forward, and vice versa. When comparing the angle variation of the right thigh for both walkers, few dissimilarities can be derived. In fact, most differences between the male and the female performances have been found in the elevation values of the limbs corresponding to the upper part of the human body.

Summarizing, a comparison framework has been presented which allows to evaluate the variation of the angles of specific human body limbs for different agents while performing the same action. This analysis of human actions helps to determine those human body model parameters which best characterize an specific action style. Con-

sequently, a suitable characterization of the action *style* can be built by analyzing the resulting angle values.

Using the characterization about action styles, human action recognition procedures can be enhanced by deriving style attributes about recognized performances. Also, we can enhance human action synthesis procedures by incorporating restrictions about a predefined action style, which the virtual agent should obey while reproducing the requested action.

Chapter 5

Towards a Cognitive Vision System

This chapter is focused on the most challenging abstraction process within the Human Sequence Evaluation framework: to *relate* qualitative knowledge at the Behavior Level (BL) with quantitative models at the Scene Domain Level (SDL). For this purpose, we establish an integration procedure addressed in two directions. The abstraction process from the SDL to the BL is defined as the generation of textual descriptions at the BL, based on the numerical results obtained at the SDL. Also, the inversion of this abstraction process is established as the automatic creation of synthetic sequences at the SDL, based on the conceptual knowledge modeled at the BL.

To confront such an abstraction task, and following the approach presented in [61], we use the *Situation Graph Tree* (SGT) formalism at the BL to model the knowledge required for human behavior analysis in an specific discourse domain. The basic component of SGTs is called *situation scheme* which comprises conceptual knowledge about a given actor for a given point in time. On the one hand, such a knowledge is based on a set of predefined conceptual predicates which *describe* at each time step the spatial position of the human agent within the scene, and also the posture exhibited. These qualitative statements are derived at the Conceptual Level (CL) using the quantitative information obtained at the SDL. On the other hand, situation schemes also embed conceptual predicates which *generate* an specific spatial and posture status for a synthetic agent. These reaction predicates are processed at the CL to derive the quantitative information required to synthesize motion for a virtual agent at the SDL.

Therefore, the abstraction process from the SDL to the BL implies to instantiate *descriptive* predicates at the CL, based on the information obtained at the SDL, in order to describe the behavior modeled at the BL. Likewise, the inversion process from the BL to the SDL implies to instantiate *generative* predicates at the CL in order to re-visualize at the SDL the behavior modeled at the BL. Both kinds of conceptual predicates constitute the *a–priori terminology* of our specific discourse domain, which is defined in Appendix B.

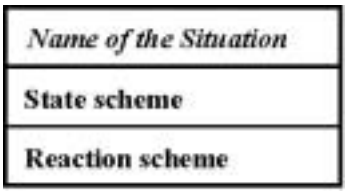


Figure 5.1: Each situation scheme is separated into two parts: the *state scheme* refers to the logical predicates which should be satisfied to instantiate such a situation; the *reaction scheme* refers to those logical predicates which will be generated when the situation is instantiated.

We show the suitability of SGTs by modeling a human activity and a human behavior. The human activity which will be described and synthesized using SGTs is called the *thief sequence* [27], in which a human agent walks, stands up straight to look around, bends to pick up an object, and then runs away. Subsequently, the behavior of an agent in a subway scene is also modeled for description and synthesis purposes. In this case, knowledge about the context is incorporated into the SGT. Despite the fact that these examples are quite trivial, the aim is to demonstrate that SGTs can cope with human behavior modeling, once human motion is properly expressed in terms of semantic predicates.

As we consider the same knowledge representation to perform both description and synthesis, i.e. the SGT formalism, evaluation of integration can be addressed using a two-step procedure [7]. First, a synthetic video is created using the textual descriptions generated for a given input sequence. Therefore, both synthetic and original *image sequences* can be compared to evaluate the knowledge used in HSE systems. Second, a textual description is generated for the resulting synthetic sequence. Consequently, we can compare both original and synthetic *textual descriptions* to detect deficiencies in the knowledge representation.

5.1 Using Situation Graph Trees

As commented in chapter 1, Situation Graph Trees (SGTs) provide a deterministic formalism to represent the knowledge required for human behavior evaluation, where *behavior* refers to activities which acquire a meaning in an specific scene.

The basic component of SGTs is the *situation scheme*, which is separated into two parts, see Fig. 5.1: 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 agent is said that *instantiates* the situation.

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 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 *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.

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 specialized is called *parent situation scheme* of the specializing situation graph. Therefore, situation schemes and their particularization edges build tree-like structures, which are called the Situation Graph Trees (SGT), see [7] for details.

An example of SGT is shown in Fig. 5.2. The root graph comprises only one situation scheme, namely **subway_behavior**. This scheme is particularized by another situation graph, comprising three schemes (**going_ticket_machine**, **walking_towards_ accessing_gate**, and **entering_subway**), whereby the first of these schemes is marked as start situation and the last one as end situation. Also, the first scheme is particularized further. Note that multiple particularizations/predictions are allowed.

SGTs can be used to recognize those situations which can be instantiated for an observed agent by applying the so called *graph traversal* [61]. The goal is to determine the most specialized situation which can be instantiated by considering the logic predicates generated at the CL at each time step. This traversal of the SGT is applied by considering the knowledge encoded in the form of prediction and specialization edges. The procedure is coarsely 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 agent is instantiating that situation. If this situation scheme is further specialized by any situation graph, these graphs are again searched for start situation schemes with state predicates also satisfied in the actual point of time. Again, if such a situation scheme can be found, it is instantiated by the actor. Using this procedure, the most special situation scheme which can be instantiated by the actor is found. In other words, for each point of time, an actor is instantiating situation schemes on several levels of detail, each on a *path* from a scheme in the root graph to the most special case.

Concerning the reaction scheme, two different cases are considered when apply-

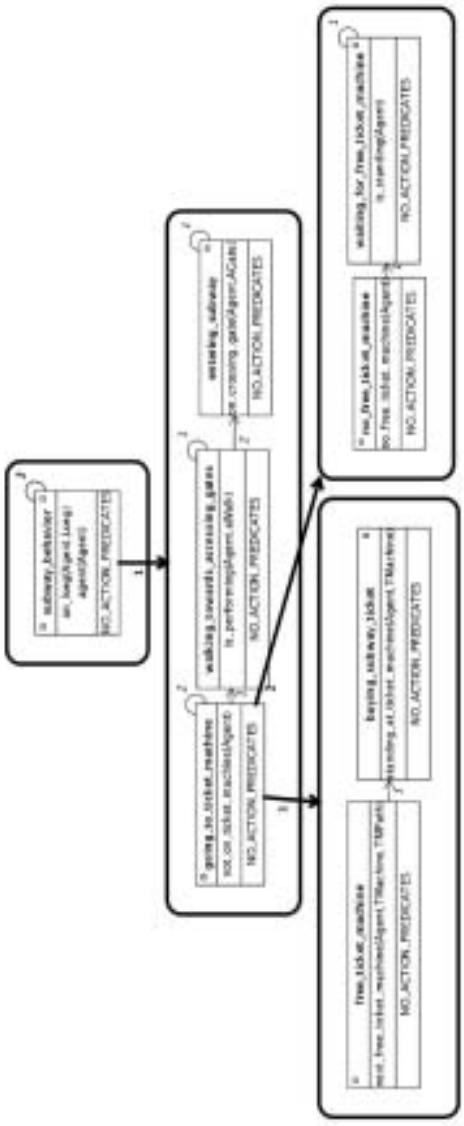


Figure 5.2: Example of SGT. Following the representation proposed in [7], situation schemes are shown as normal rectangles. Thin arrows stand for prediction edges, while bold arrows represent specialization edges. Circles in the right upper corner of situation schemes indicate a self-prediction of that scheme. Rounded rectangles depict situation graphs. Small rectangles to the left or right of the name of the situation indicate that the scheme is start or end situation, respectively.

ing 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 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 special one on a path of schemes that an actor is instantiating.

For the next point in time, only those situation schemes, which are connected by prediction edges to the situation scheme instantiated last, are investigated. Therefore, only those situation schemes of the same graph are searched. In other words, a prediction for a situation scheme on the same level of detail is 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, the failed prediction on this level is accepted 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 fails completely.

The traversal of SGTs is implemented by executing a logic-program obtained by an automatic translation of the SGT into rules of a *Fuzzy Metric Temporal Horn Logic* (FMTHL). By translating the SGT-traversal into a set of fuzzy logic rules, it is possible to operate with fuzzy truth values, which are essential to cope with the uncertainty arising in HSE systems. Furthermore, by translating the SGT-traversal into a set of metric temporal logic rules, it is possible to explicitly reason about time. These characteristics justify the utilization of SGTs within the HSE framework, which is addressed as follows.

A quantitative description of the state of the agent is obtained at the SDL for each time step. In our experiments, we assume that the following information is provided to the CL:

- The 2-D spatial position *pos* of the agent *Agent* in the floor plane, in addition to the velocity *vel* (in kilometers per hour) and the orientation *or* (in degrees). These three parameters are called the *spatial status* of the agent, which is generic for Image Sequence Evaluation systems. This information is commonly obtained using visual tracking procedures.

- The action label *aLabel* which the actor is performing. In addition, the human posture is also represented by the parameter *aPosture*. These two parameters are denoted as the *posture status* of the actor, which is especific of HSE systems. In fact, we associate the human action representation described in chapter 3 to the posture status of the agent. That is, given the *aLabel* parameter of the agent state, the *p–action* representation of the *aLabel* action is selected. Subsequently, the *aPosture* parameter establishes the arc length value of the *p–action* manifold, which corresponds to a particular human posture configuration. Therefore, incrementing the *aPosture* parameter implies to move along the *p–action* or, in other words, to reproduce the *aLabel* action.

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

$$has\_status(Agent, pos, or, vel, aLabel, aPosture). \quad (5.1)$$

Subsequently, the information embedded in the state of the agent is associated at the CL with concepts formulated in terms of logic predicates. These temporally isolated logic statements will be derived from both the state of the human agent and a predefined conceptual model of the scene, and they refer to three sources of knowledge:

- The spatial information of the actor within the scene, such as the position, velocity and orientation in the floor plane.
- The relationship of the actor with respect to its environment, such as the presence of other actors nearby or the proximity of static objects represented in the conceptual scene model.
- The action which the actor is performing, which is obtained using a suitable action recognition procedure. Also, the human posture representation for the observed actor can be considered.

For this purpose, we enhance the *terminology* of logic statements used in [61] to include predicates which *describe* the spatial information about the human agent within the scene, the properties of the actor with respect to its environment, and the action which the actor is performing. Our terminology can be found in Appendix B.

Descriptive predicates allow the instantiation of situation schemes, one situation per agent and frame. As situation schemes describe the behavior of a given agent for a single point in time, the graph traversal applied to SGTs involves to instantiate different situations over time, where the instantiation of each situation is based on the (temporally isolated) logic predicates derived at the CL.

The sequence of the resulting instantiated situations can be used both for description and synthesis. On the one hand, reaction predicates can be processed at the *Language Level* (LL) to formulate a natural language text describing the recognized behavior within the scene. On the other hand, reaction predicates can be used to generate video sequences, during which (previously modeled) human behaviors are automatically synthesized at the SDL. A proper human movement model should be implemented to modify the state of the agent according to instantiated reaction predicates. Therefore, as the state of the agent changes over time, we will deterministically predefine which sequence of situations should be instantiated in order to re-visualize predetermined behaviors.

5.2 Generation of Textual Descriptions

In order to model the behavior of human agents, we first define an a-priori terminology which consists of logic predicates related to the information to be described about

the scene. Therefore, the terminology actually restricts the discourse domain, and consists of logic-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.

According to the terminology described in Appendix B, we detail three different strategies to associate conceptual descriptions using the quantitative information of the agent state. Thus, semantic primitives related to three different sources of knowledge will be instantiated for a given point in time.

Subsequently, an SGT is used to represent a particular human behavior. For this purpose, it should be established the set of logic predicates which will be embedded both in the state and reaction schemes of each situation to be modeled. Two examples are provided to illustrate this process, i.e., a human activity and a human behavior.

We first represent the human activity called the *thief sequence*. The SGT is composed of predefined situations which define in a temporal (by means of the prediction edges) and specific (by means of the particularization edges) manner the sequence of situations which should be instantiated in order to describe such an activity. Each situation scheme embeds a set of state predicates which the agent should satisfy to instantiate the situation.

To conclude, an SGT is used to describe the behavior of human agents within a particular scene. In our experiments, a conceptual scene model of a predefined subway scene is developed in order to incorporate the knowledge about the context. In fact, the context helps to determine the relationship of the agent with respect its environment. We next describe the set of *plausible* situation schemes which can be instantiated within a subway station. After that, we organize the set of situations in a conceptually and temporally more detailed manner, according to the desired degree of specificity for the inferred textual descriptions.

5.2.1 From Quantitative to Qualitative Knowledge

Based on the quantitative information of the state vector of the agent at the SDL, the aim is to to associate conceptual interpretations for such numerical data at the CL. 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.

About the spatial information of an actor within a scene : to abstract the quantitative state parameters by associating 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:

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

About the relationship of an actor with respect to its environment : to derive spatial relations by considering the positions of the agents and other static objects in the scene. In this case, a conceptual scene model is required to describe the spatial coordinates of the agent with respect to static objects, other agents, and specific locations within 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:

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

Also, predicates concerning the spatial properties of the agent with respect to static objects (predefined at the conceptual scene model) can be instantiated. For example, the predicate:

$$on\_ticket\_machine(Agent, TMachine),$$

checks whether the spatial position of the agent *Agent* in the scene is inside the subway segment defined as *TMachine*.

About the action which an actor is performing : to associate an action label to the state of the agent, depending on the agent posture configuration. For this purpose, a variation of the action recognition algorithm proposed in chapter 4 is used, as described next. As a result of recognition, the posture status already embeds a conceptual term, i.e. the label of the recognized action:

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

But the 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. Thus, a conceptual interpretation using Fuzzy Logic may be required here, too.

The first two types of qualitative predicates refer to the *spatial status* of the agent, and they are generated by discretization of the numerical values of the agent state. Additionally, these predicates require of a conceptual scene model to relate the spatial information of the agent with respect to the scene.

The last item refers to those predicates related to the *posture status* of the agent, and the abstraction process can be implemented following two different approaches. On the one hand, we can provide the label of the recognized action at the CL when such an action is finished. In fact, most action recognition algorithms assign a label to an unknown input sequence only once the performance is finished[1]. However, this is not a compulsory assumption when designing action recognition procedures.

[1]When dealing with repetitive actions such as *aRun* or *aWalk*, the action term is generated when just a *cycle* is completed.

Instead, we derive the label of a *tentatively recognized* action while the performance is being executed, i.e., human action recognition is applied to each posture of the sequence. In this case, several possible labels can be instantiated at the same time, due to the fact that different actions can exhibit some identical postures during their performances, for example *aSquat* and *aTumble*. The abstraction process for the posture status is as follows.

Consider a human body posture model $\mathbf{x}_T$ defined as in chapter 3, and obtained from a motion capture system. Subsequently, the projection of $\mathbf{x}_T$ in the *UaSpace* $\mathbf{\Omega}^U$ is computed as described in chapter 4. Thus, we obtain $\mathbf{y}_T = [y_1, y_2, ..., y_{m^U}]^T$, where m^U correspond to the number of eigenvectors considered to represent $\mathbf{\Omega}^U$, in our experiments $m^U = 20$. Subsequently, we use the distance measure between two manifolds in order to compute the nearest *key-frame* $\mathbf{k}_i^A$ to $\mathbf{y}_T$:

$$d(\mathbf{k}_i^A, \mathbf{y}_T) = \left\| \frac{\mathbf{k}_i^A}{\| \mathbf{k}_i^A \|} - \frac{\mathbf{y}_T}{\| \mathbf{y}_T \|} \right\|. \tag{5.2}$$

Therefore, the nearest key-frame $\mathbf{k}^A$ is obtained by considering the minimum distance measure to all the key-frames of all modeled actions in $\mathbf{\Omega}^U$:

$$\mathbf{k}^A = \arg\min_i d(\mathbf{k}_i^A, \mathbf{y}_T), \qquad i = 1, \ldots, k_U. \tag{5.3}$$

where k_U refers to the number of key-frames considered for comparison. The last step assigns the action label $\mathbf{A}$ to the *aLabel* parameter of the the agent state. Also, the arc length value which corresponds to $\mathbf{k}^A$ in the *p–action* manifold $\mathbf{f}^A$ is assigned to the *aPosture* parameter of the agent state.

Different action labels can be assigned to the same input posture, due to that fact that different action manifolds can embed similar postures, like *aBend* and *aSit*, for example. In this case, a description for each possible action is generated at each time step. In our experiments, at the very most, the three nearest action labels determine the correct action.

Based on the final purpose of HSE systems, the aim is to infer at each time step *tentatively recognized* posture status parameters which can be verified once the action has *finished*. So we assume hereafter that the state of the agent is provided by the SDL to the CL at each time step.

5.2.2 Human Activity Description

Next, we present an SGT which models a human activity. Obeying the terminology, situation schemes are defined in terms of qualitative predicates which refer to the spatial position and the posture status of the agent. That means, no reference to the scene is required to model human activities.

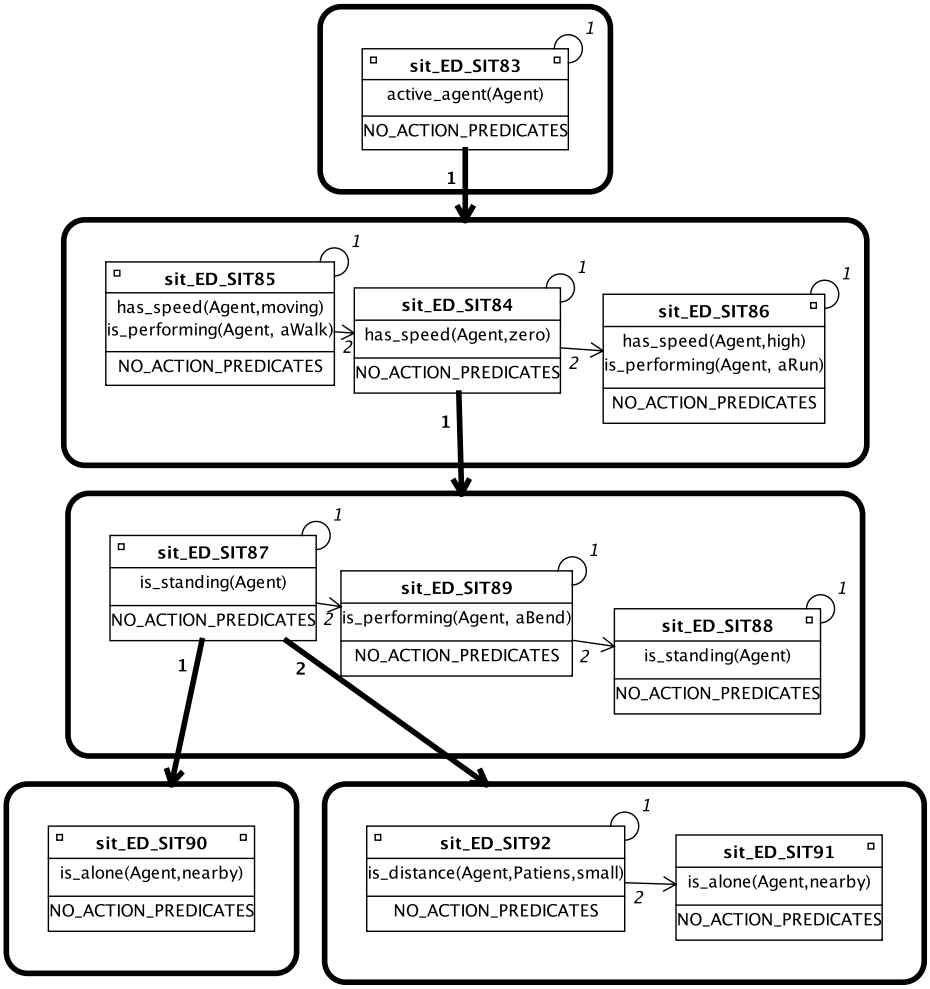


Figure 5.3: Situation Graph Tree modeling the *thief sequence* for descriptive purposes.

The SGT depicted in Fig. 5.3 represents the so-called *thief sequence*, in which a "thief" approaches an object, stands up straight to look around, bends to pick up the object, and then runs away.

The root graph comprises only one situation scheme, in which the predicate states that an agent is presently active, *active(Agent)*. That means, the state vector of the agent is available at the SDL. This scheme is particularized by another situation graph, comprising three schemes. The firs scheme (start-situation) is instantiated when the agent is (recognized as) walking, *is_performing(Agent,aWalk)*. Next situation in temporal order is instantiated when the agent is still, i.e. has no spatial speed, *has_speed(Agent,zero)*. Afterwards, the last scheme, referred to as end-

situation, is instantiated when the agent is recognized as performing the running action, *is_performing(Agent, aRun)*.

When the agent exhibits no movement, a particularized situation graph represents the pick-up or *aBend* action, *is_performing(Agent, aBend)*. But first, the agent looks around to check whether someone is near. This process is represented at the SGT by the start-situation *is_standing(Agent)*, which is particularized further. Note that different particularizations are allowed.

The first particularized situation graph states that the agent is actually alone: *is_alone (Agent, nearby)*. The second particularized situation graph checks if another agent, i.e the *patiens*, is positioned near enough, *is_distance (Agent, Patiens, small)*. If this situation is instantiated, the agent should wait until he is alone to pick up the object on the floor. This is reflected by its *self-prediction edge*. The end-situation is instantiated when the patiens has already left its neighborhood.

In this example, no reference to a-priori modeled static objects of the scene has been required. We only need to determine the relationship between the spatial position of observed agents. Next, we show a different example in which the relationship of the agent with respect to the static components of the scene helps to infer human behavior.

5.2.3 Human Behavior Description

In this section we show an example of SGT which can be used to describe human behavior within a subway station. 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. Thus, for example, if the agent is in front of a ticket machine, we *assume* that the agent is actually buying a ticket.

The conceptual scene model used is shown in Fig. 5.4. The scene is divided into polygonally bounded *subway_segments*, which describe the possible positions in which an agent can be found. Each *subway_segment* has a label which determines the conceptual description associated to such a segment. Thus, we distinguish (at least) six different types of segments, namely:

ticket_machine : subway segments placed in front of the ticket machines. If positioned on these segments, we assume that the agent is buying the ticket.

waiting_line : subway segments placed in front of the *ticket_machine* segments. The agent is expected to wait in these segments until a ticket machine is free.

inside_station : subway segments placed at the subway platform.

accessing_gate : subway segments placed at the accessing gates. Agents get into the subway platform through these segments.

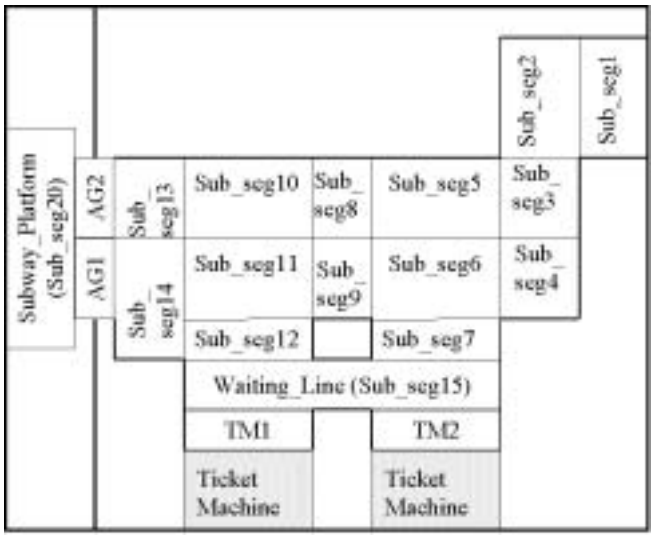


Figure 5.4: Conceptual subway scene model for human behavior description (see text for details).

entrance_segment : subway segments placed in front of the accessing gates. Agents get into the accessing gates through such segments.

sub_seg i : **i**-th subway segment through which the agent (normally) walks to get to accessing gates or to ticket machines.

Consequently, we can build predicates which relate the spatial position of the agent with respect to these segments. Fig. 5.5 depicts an SGT which allows to infer the behavior of agents within the subway station scene, as detailed next. In our experiments, four different agents were synthetically generated. Their trajectories in the *subway scene*, are depicted in Fig. 5.6.

The root graph comprises only one situation scheme, in which the predicate states that an agent is presently active (*active(Agent)*). This scheme is particularized by another situation graph, comprising four schemes. The start-situation is instantiated when the agent is not crossing the accessing gates. Therefore, the agent is expected to buy the subway ticket. The next situation in temporal order is instantiated when the agent is going to cross through the accessing gates, and the last-situation refers to the agent on the subway platform.

The start-situation (*not_on_entrance_segment(Agent)*) is particularized to describe the purchase of the ticket, which is again divided into four situation schemes:

- First, the agent looks for a ticket machine, *not_on_ticket_machine(Agent)*. This situation is specialized further to detect when the agent reaches the *waiting_line* segment: first, the agent is expected to walk towards the waiting

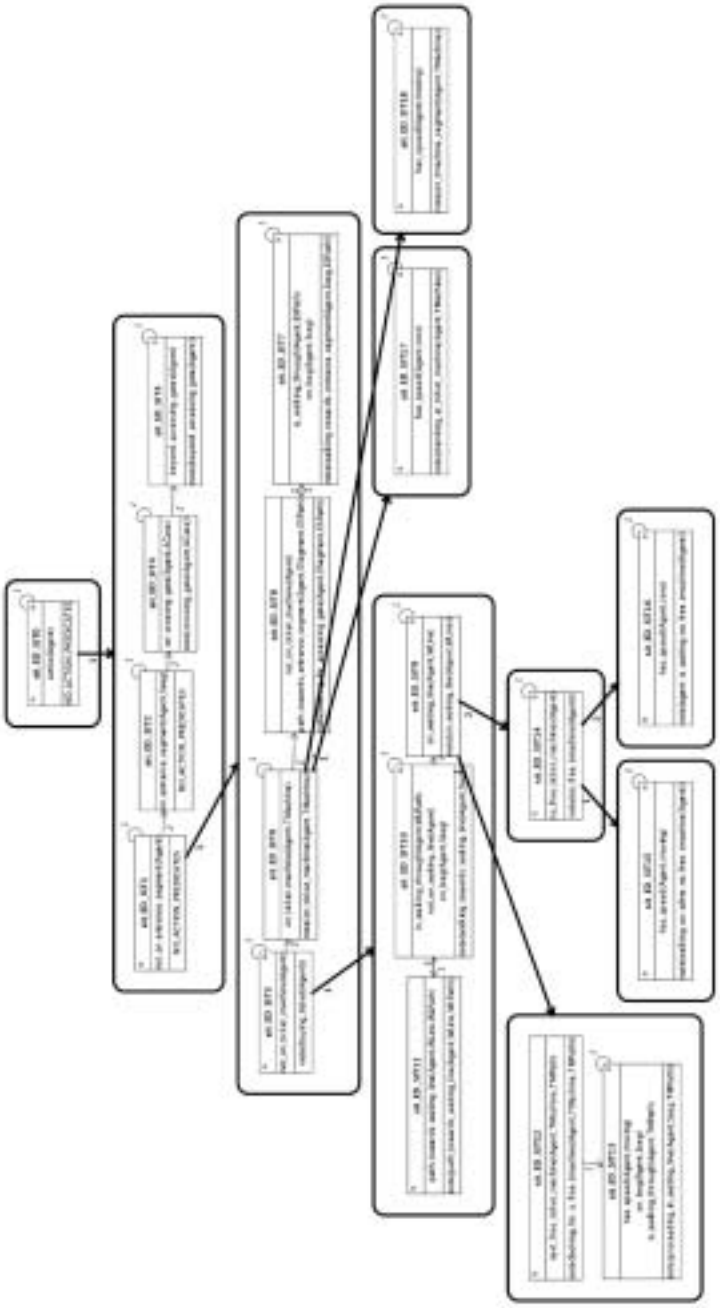


Figure 5.5: (Part of an) SGT for human behavior description in a subway station.

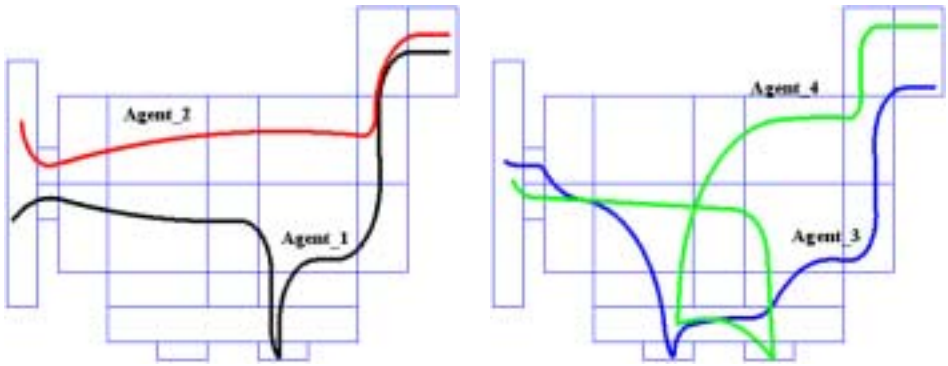


Figure 5.6: Trajectories of the agents within the subway scene. Agents **agent_1** and **agent_2** are shown in **(a)**. Agents **agent_3** and **agent_4** are shown in **(b)**.

line (*path_towards_waiting_line (Agent, WLine, WLPath)*), where *WLine* corresponds to the *waiting_line* segment, and *WLPath* holds the list of subway segments constituing the path from the agent to *WLine*. The next situation in temporal order checks whether the agent is following the list of segments of *WLPath*, *is_walking_through(Agent,WLPath)*. The last situation analyses the behavior of the agent at the *waiting_line* segment. This is done by considering two different specialized situation graphs. On the one hand, a free ticket machine segment is searched, *next_free_ticket_machine(Agent, TMPath, TMachine)*. As a result, *TMPath* refers to a sequence of subway segments which lead to the free ticket machine segment *TMachine*. On the other hand, if no free ticket machine segment is found (*no_free_ticket_machine(Agent)*), the agent should wait. During this waiting period, the agent can move (*has_speed(Agent,moving)*) or adopt a standing position, *has_speed(Agent,zero)*.

- Once the human agent reaches the ticket machine segment (*on_ticket_machine (Agent, TMachine)*), s/he is expected to exhibit no spatial movement only when the ticket is being bought, *has_speed(Agent,zero)*.

- When the agent departs from the ticket machine segment, s/he is expected to go towards a crossing gate segment. Consequently, a path towards the entrance segment is computed (*path_towards_entrance_segment (Agent, ESegment, ESPath)*), where *ESegment* corresponds to the *entrance_segment* segment, and *ESPath* holds the list of subway segments constituting the path from the current agent position to *ESegment*.

- The next situation in temporal order checks whether the agent is following the list of segments of *ESPath*, (*is_walking_through(Agent,ESPath)*.

Start	End	Situation
3	41	walking_towards_waiting_line(**agent_1**,lseg_1).
42	82	walking_towards_waiting_line(**agent_1**,lseg_2).
83	133	walking_towards_waiting_line(**agent_1**,lseg_3).
134	176	walking_towards_waiting_line(**agent_1**,lseg_4).
177	242	walking_towards_waiting_line(**agent_1**,lseg_6).
243	267	walking_towards_waiting_line(**agent_1**,lseg_7).
268	287	proceeding_through_waiting_line(**agent_1**,lseg_15).
288	295	walking_on_ticket_machine_segment(**agent_1**,lseg_16).
296	538	standing_at_ticket_machine(**agent_1**,lseg_16).
539	549	walking_on_ticket_machine_segment(**agent_1**,lseg_16).
550	569	walking_towards_entrance_segment(**agent_1**,lseg_15).
570	589	walking_towards_entrance_segment(**agent_1**,lseg_7).
590	617	walking_towards_entrance_segment(**agent_1**,lseg_6).
618	669	walking_towards_entrance_segment(**agent_1**,lseg_9).
670	770	walking_towards_entrance_segment(**agent_1**,lseg_11).
771	819	walking_on_entrance_segment(**agent_1**,lseg_14).
820	839	crossing_gate(**agent_1**,lseg_19).
840	868	walking_inside_subway_platform(**agent_1**,lseg_20).

Table 5.1: Sequence of textual descriptions generated for agent **agent_1** using the SGT of Fig. 5.5 and the trajectory data of Fig. 5.6.

Start	End	Situation
201	241	walking_towards_waiting_line(**agent_2**,lseg_1).
242	289	walking_towards_waiting_line(**agent_2**,lseg_2).
290	324	walking_towards_waiting_line(**agent_2**,lseg_3).
325	424	walking_towards_waiting_line(**agent_2**,lseg_5).
425	474	walking_towards_waiting_line(**agent_2**,lseg_8).
475	574	walking_towards_waiting_line(**agent_2**,lseg_10).
575	625	walking_on_entrance_segment(**agent_2**,lseg_13).
626	645	crossing_gate(**agent_2**,lseg_18).
646	675	walking_inside_subway_platform(**agent_2**,lseg_20).

Table 5.2: Sequence of textual descriptions generated for agent **agent_2** using the SGT of Fig. 5.5 and the trajectory data of Fig. 5.6.

The situation *on_entrance_segment(Agent,Sseg)* is instantiated when the agent reaches the entrance segment. Then, the agent gets into the subway platform through a crossing gate, *on_crossing_gate(Agent, AGate)*. Subsequently, the agent is expected to be placed in the subway platform segment, *on_subway_seg(Agent,subway_platform)*.

The textual descriptions generated for the four agents considered are shown in Tables 5.1, 5.2, 5.3, and 5.4.

Start	End	Situation
101	141	walking_towards_waiting_line(**agent_3**,lseg_1).
142	147	walking_towards_waiting_line(**agent_3**,lseg_2).
148	202	walking_towards_waiting_line(**agent_3**,lseg_3).
203	266	walking_towards_waiting_line(**agent_3**,lseg_4).
267	313	walking_towards_waiting_line(**agent_3**,lseg_6).
314	347	walking_towards_waiting_line(**agent_3**,lseg_7).
348	447	proceeding_through_waiting_line(**agent_3**,lseg_15).
448	455	walking_on_ticket_machine_segment(**agent_3**,lseg_17).
456	700	standing_at_ticket_machine(**agent_3**,lseg_17).
701	710	walking_on_ticket_machine_segment(**agent_3**,lseg_17).
711	731	walking_towards_entrance_segment(**agent_3**,lseg_15).
732	753	walking_towards_entrance_segment(**agent_3**,lseg_12).
754	825	walking_towards_entrance_segment(**agent_3**,lseg_11).
826	867	walking_on_entrance_segment(**agent_3**,lseg_14).
868	880	walking_on_entrance_segment(**agent_3**,lseg_13).
881	901	crossing_gate(**agent_3**,lseg_18).
902	920	walking_inside_subway_platform(**agent_3**,lseg_20).

Table 5.3: Sequence of textual descriptions generated for agent **agent_3** using the SGT of Fig. 5.5 and the trajectory data of Fig. 5.6.

5.3 Generation of Synthetic Image Sequences

In this section, we adapt the process presented in [103], where an inversion of the abstraction process presented so far is developed. The goal is to generate animated sequences from elementary conceptual descriptions. We restrict ourselves to develop an algorithmic approach addressed to build a synthetic sequence during which a virtual agent performs the modeled behavior.

For this purpose, we utilize the SGT-based knowledge representation to model the synthetic behavior of an agent. Thus, we can deterministically define which, and in which order, situation schemes should be instantiated to generate a particular behavior. Note that, when a situation is instantiated, reaction predicates are generated. So we will adapt these reaction predicates to modify the state of the agent according to the parameters of the agent state itself, and the relationship of the agent with respect to its environment.

The creation process of video sequences is as follows. Given an initial, quantitative state of the agent, conceptual terms are associated to the numerical values of such a state in order to instantiate logic predicates at the CL. Subsequently, the graph traversal is applied to the SGT modeled at the BL. Thus, the most special starting situation is searched for which each logic predicate of the state scheme is satisfied for the first point of time or, in other words, for which the set of logic predicates of the state scheme is found instantiated at the CL.

Start	End	Situation
141	184	walking_towards_waiting_line(**agent_4**,lseg_1).
185	250	walking_towards_waiting_line(**agent_4**,lseg_2).
251	274	walking_towards_waiting_line(**agent_4**,lseg_3).
275	375	walking_towards_waiting_line(**agent_4**,lseg_5).
376	424	walking_towards_waiting_line(**agent_4**,lseg_8).
425	442	walking_towards_waiting_line(**agent_4**,lseg_9).
443	480	walking_towards_waiting_line(**agent_4**,lseg_11).
481	500	walking_towards_waiting_line(**agent_4**,lseg_12).
501	510	proceeding_through_waiting_line(**agent_4**,lseg_15).
511	549	waiting_for_a_free_ticket_machine(**agent_4**).
550	634	proceeding_through_waiting_line(**agent_4**,lseg_15).
635	650	walking_on_ticket_machine_segment(**agent_4**,lseg_16).
651	740	standing_at_ticket_machine(**agent_4**,lseg_16).
741	750	walking_on_ticket_machine_segment(**agent_4**,lseg_16).
751	770	walking_towards_entrance_segment(**agent_4**,lseg_15).
771	790	walking_towards_entrance_segment(**agent_4**,lseg_7).
791	835	walking_towards_entrance_segment(**agent_4**,lseg_6).
836	885	walking_towards_entrance_segment(**agent_4**,lseg_9).
886	986	walking_towards_entrance_segment(**agent_4**,lseg_11).
987	1036	walking_on_entrance_segment(**agent_4**,lseg_14).
1037	1056	crossing_gate(**agent_4**,lseg_19).
1057	1070	walking_inside_subway_platform(**agent_4**,lseg_20).

Table 5.4: Sequence of textual descriptions generated for agent **agent_4** using the SGT of Fig. 5.5 and the trajectory data of Fig. 5.6.

Once the first situation scheme is instantiated, its corresponding reaction scheme is executed. The reaction predicates generated at the BL are processed at the CL in order to conceptually determine which parameter(s) of the agent state should be modified and how. For this purpose, a proper *human movement model* is developed to (generically) update the state of the human agent according to the reaction predicates activated by the graph traversal. Consequently, the graph traversal generates a sequence of reaction predicates which are processed by the human movement model in order to modify the position, orientation, speed, action and posture of the synthetic human agent.

As the new state of the agent is (possibly) different, a different situation scheme can be instantiated at next time step. As only those situation schemes, which are connected by prediction edges to the situation scheme instantiated last, are investigated, we can deterministically define which, and in which order, situation schemes will be executed by graph traversal. When graph traversal finishes, the resulting sequence of states, which has been generated by the human movement model, describes quantitatively the synthetic behavior of the agent within the scene, as explained next.

5.3.1 From Qualitative to Quantitative Knowledge

As described in the preceeding section, the state of the human agent is composed of five fields: position, orientation, speed, action and posture. As the set of possible reaction predicates should modify these parameters, we establish five types of reaction predicates, according to the parameter of the agent state which the human action model modifies:

- Reaction predicates related to both the agent's position and orientation. For example, the reaction predicate *follow_path(Agent, Subway_seg, Path)* modify the direction of the agent *Agent* in order to go from *Subway_seg* towards its next subway segment in the segment list of *Path*.

- Reaction predicates related to only the agent's orientation. For example, the predicate *turn(Agent, Value)* modifies the direction of the agent *Agent* depending on the orientation value *Value*. Possible conceptual descriptions of *Value* are *left*, and *right*.

- Reaction predicates related to the agent' speed. The predicate *accelerate(Agent, Value)*, for example, modifies the velocity of the agent by using *Agent* depending on the acceleration value *Value*. Also, the predicate *accelerate_to (Agent,Value)* modifies the velocity of the agent *Agent* in order to reach progressively the qualitative value of *Value*. Possible conceptual descriptions of *Value* are *high*, *normal*, and *small*.

- Reaction predicates related to the agent's action. In our experiments, three predicates were implemented. The predicate *go_on_performing(Agent, ALabel)* increments the parameter *aPosture* of the agent *Agent* in order to adopt the next posture described by the action *ALabel*. Also, the predicate *change_performing (Agent, ANewLabel)* increments the posture parameter of the agent *Agent* in order to change from the current action to the *ANewLabel* action. Lastly, the predicate *keep_on_performing(Agent, Duration)* keeps the agent *Agent* performing cyclic actions, as the *aWalk* and *aRun*) actions, during the period determined by the qualitative value *Duration*. Possible conceptual descriptions of *Duration* are *long*, *normal*, and *briefly*.

- Reaction predicates related to the agent's posture. The predicate *wait (Agent, Duration)* keeps the agent *Agent* performing the *aStand* action during the period determined by the qualitative value *Duration*. Also, the reaction predicate *freeze(Agent, Duration)* 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*.

These reaction predicates are processed by the human movement model, thus generating a sequence of synthetic agent states. Next, we present examples of synthetic sequences generated by these reaction predicates.

Frame	Agent State						
935	has_status(**agent_1**,	16.60,	17.04,	14.93,	0.16,	aBend,	0.0).
936	has_status(**agent_1**,	16.61,	17.04,	14.93,	0.13,	aBend,	0.05).
937	has_status(**agent_1**,	16.61,	17.04,	14.93,	0.11,	aBend,	0.1).
938	has_status(**agent_1**,	16.61,	17.04,	14.93,	0.09,	aBend,	0.15).
⋮	⋮						
953	has_status(**agent_1**,	16.61,	17.04,	14.93,	0.00,	aBend,	0.9).
954	has_status(**agent_1**,	16.61,	17.04,	14.93,	0.00,	aBend,	0.95).
955	has_status(**agent_1**,	16.61,	17.04,	14.93,	0.00,	aFinish,	0.0).

Table 5.5: Sequence of agent states which generates a synthetic *aBend* action. The agent state is defined as in (5.1).

Frame	Agent State						
4	has_status(**agent_1**,	1.04,	2.50,	1.0,	1.91,	aWalk,	0.05).
5	has_status(**agent_1**,	1.08,	2.50,	0.90,	1.83,	aWalk,	0.1).
6	has_status(**agent_1**,	1.11,	2.50,	0.82,	1.77,	aWalk,	0.15).
7	has_status(**agent_1**,	1.15,	2.50,	0.74,	1.72,	aWalk,	0.2).
⋮	⋮						
21	has_status(**agent_1**,	1.59,	2.50,	0.18,	1.51,	aWalk,	0.9).
22	has_status(**agent_1**,	1.62,	2.50,	0.16,	1.51,	aWalk,	0.95).
23	has_status(**agent_1**,	1.65,	2.50,	0.15,	1.51,	aWalk,	0.0).
24	has_status(**agent_1**,	1.68,	2.51,	0.13,	1.50,	aWalk,	0.05).

Table 5.6: Sequence of agent states which generates a synthetic *aWalk* action. The agent state is defined as in (5.1).

Table 5.5 shows the sequence of agent states which describes a synthetic *aBend* action. This sequence has been obtained by instantiating the *go_on_performing(Agent, aBend)* predicate. Given an action model $\mathbf{\Gamma}^{A_i}$, this predicate increments the *aPosture* parameter in order to increasingly sample the corresponding action manifold $\mathbf{f}^{A_i}$ or, in other words, to re-visualize the variation of the human posture while performing the selected action. Note that the increment value determines the speed at which the action is synthesized. When the *aPosture* parameter value reaches 1.0, the non-cyclic action manifold has been totally sampled. Consequently, the *aLabel* parameter value is set to *aFinish*.

When a cyclic action is synthesized, such as *aWalk* or *aRun*, the human movement model sets the *aPosture* parameter to 0.0 when it reaches 1.0. Thus, the cyclic action manifold is sampled repeatedly and continuously. Table 5.6 shows the sequence of agent states which generates a synthetic *aWalk* action. Such a sequence has been obtained by instantiating the *go_on_performing(Agent, aWalk)* predicate.

Next, the synthesis of activities is addressed by describing the agent states which correspond to action transitions. Two different cases are considered, depending on the cyclic/non-cyclic nature of the currently synthesized action. On the one hand, when

Frame	Agent State						
986	has_status(**agent_1**,	14.95,	19.04,	161.03,	5.98,	aRun,	0.65).
987	has_status(**agent_1**,	14.84,	19.07,	161.03,	4.90,	aRun2aStand,	-0.95).
988	has_status(**agent_1**,	14.74,	19.11,	161.03,	4.01,	aRun2aStand,	-0.85).
989	has_status(**agent_1**,	14.67,	19.13,	161.03,	3.28,	aRun2aStand,	-0.75).
⋮	⋮						
995	has_status(**agent_1**,	14.43,	19.21,	161.03,	0.98,	aRun2aStand,	-0.15).
996	has_status(**agent_1**,	14.41,	19.22,	161.03,	0.81,	aRun2aStand,	-0.05).
997	has_status(**agent_1**,	14.39,	19.23,	161.03,	0.66,	aStand,	0.0).

Table 5.7: Sequence of agent states which generates a synthetic action transition, i.e., from *aRun* to *aStand*. The agent state is defined as in (5.1).

the agent is performing cyclic actions, the transition to a new action starts when the situation scheme which holds the reaction predicate *go_on_performing(Agent, aWalk)* can not be instantiated at a certain time step, i.e. when the graph traversal fails in such a situation. Therefore, we can initiate action transitions depending on the spatial status of the agent. Moreover, we can also determine how long a cyclic action can last, regardless of the spatial status information. This is achieved by considering the *keep_on_performing(Agent, Duration)* predicate, which keeps the agent *Agent* performing cyclic actions during the time period determined by the qualitative value *Duration*.

Despite the way that action transitions are initiated, these are indicated in the state vector by using negative *aPosture* parameter values. An action transition between two action models $\mathbf{\Gamma}^{A_i}$ and $\mathbf{\Gamma}^{A_j}$ in the *UaSpace* is implemented as a manifold which interpolates between the current posture p_i of the manifold $\mathbf{f}^{A_i}$ and the initial posture $p_j = 0.0$ of $\mathbf{f}^{A_j}$. Thus, we represent the action transition as a interpolated curve from $\mathbf{f}^{A_i}(p_i)$ to $\mathbf{f}^{A_j}(p_j)$, which is called $\mathbf{f}^{A_i 2 A_j}$. Consequently, an action transition manifold obeys that:

$$\mathbf{f}^{A_i 2 A_j}(aPosture) = \begin{cases} \mathbf{f}^{A_i}(p_i), & if \quad aPosture = -1.0, \\ \mathbf{f}^{A_j}(p_j), & if \quad aPosture = 0.0, \end{cases} \tag{5.4}$$

and, as before, the action is synthesized by sampling the transition manifold $\mathbf{f}^{A_i 2 A_j}$ by incrementing the (negative) *aPosture* parameter value. Such an increment value actually determines the speed at which the action transition is synthesized, whether the arc length parameterization is applied. Furthermore, the *aLabel* parameter value indicates that a transition is being synthesized by establishing the syntax of the action label as $\mathbf{A}_i\mathbf{2}\mathbf{A}_j$. The sequence of agent states which generates the synthetic action transition from *aRun* to *aStand* is shown in Table 5.7. These agent states have been generated by instantiating the *keep_on_performing(Agent,briefly)* predicate, and subsequently (at frame 987) the *change_performing(Agent,aStand)* predicate.

Frame	Agent State							
253	has_status(**agent_1**,	4.62,	9.04,	37.26,	0.00,	aBend,	0.9).	
254	has_status(**agent_1**,	4.62,	9.04,	37.26,	0.00,	aBend,	0.95).	
255	has_status(**agent_1**,	4.62,	9.04,	37.26,	0.00,	aFinish,	0.0).	
256	has_status(**agent_1**,	4.63,	9.05,	37.26,	1.09,	aFinish2aRun,	-0.95).	
257	has_status(**agent_1**,	4.65,	9.05,	42.80,	1.98,	aFinish2aRun,	-0.85).	
258	has_status(**agent_1**,	4.68,	9.07,	54.78,	2.70,	aFinish2aRun,	-0.75).	
⋮	⋮							
265	has_status(**agent_1**,	4.80,	9.59,	106.17,	5.18,	aFinish2aRun,	-0.05).	
266	has_status(**agent_1**,	4.77,	9.69,	112.12,	5.33,	aRun,	0.0).	
267	has_status(**agent_1**,	4.72,	9.79,	117.12,	5.45,	aRun,	0.05).	

Table 5.8: Sequence of agent states which generates a synthetic action transition, i.e., from *aBend* to *aRun*. The agent state is defined as in (5.1).

On the other hand, when the agent is currently performing a non-cyclic action, the transition to a new action begins when the non-clycic action is finished or, in other words, when the *aPosture* parameter reaches 1.0 and the *aLabel* parameter is set to *aFinish*. As before, the action transition between an non-cyclic action $\mathbf{A}_i$ and a new action $\mathbf{A}_j$ is indicated in the state vector by using negative *aPosture* parameter values. Also, we model such an action transition as a interpolated curve from $\mathbf{f}^{A_i}(1.0)$ to $\mathbf{f}^{A_j}(0.0)$, also called $\mathbf{f}^{A_i 2 A_j}$. The sequence of agent states which generates the synthetic action transition from *aBend* to *aRun* is shown in Table 5.8. This sequence of agent states have been generated by instantiating the *go_on_performing(Agent,aBend)* predicate, and once the *aBend* action synthesis is finished (at frame 255), the *change_performing(Agent,aRun)* predicate is instantiated.

We next address the correspondence between the quantitative information of the state vector and the geometrical information of the human body model. Given the *aLabel* parameter of the agent state, the *p–action* representation of the *aLabel* action is selected. Subsequently, the *aPosture* value establishes the arc length parameter of the *p–action* manifold, which determines a particular human posture. Such a posture corresponds to the human body model defined in chapter 3. Therefore, incrementing the *aPosture* parameter implies to move along the *p–action* or, in other words, to synthesize the *aLabel* action. To conclude, the resulting human body configuration is visualized using a suitable representation, such as a stick figure.

5.3.2 Human Activity Synthesis

The SGT depicted in Fig. 5.7 generates the so-called *thief sequence*, in which, as commented before, a "thief" approaches an object, stands up straight to look around, bends to pick up the object, and then runs away.

The root graph comprises only one situation scheme, in which the predicate states that an agent is presently active, (*active(Agent)*. This scheme is particularized by

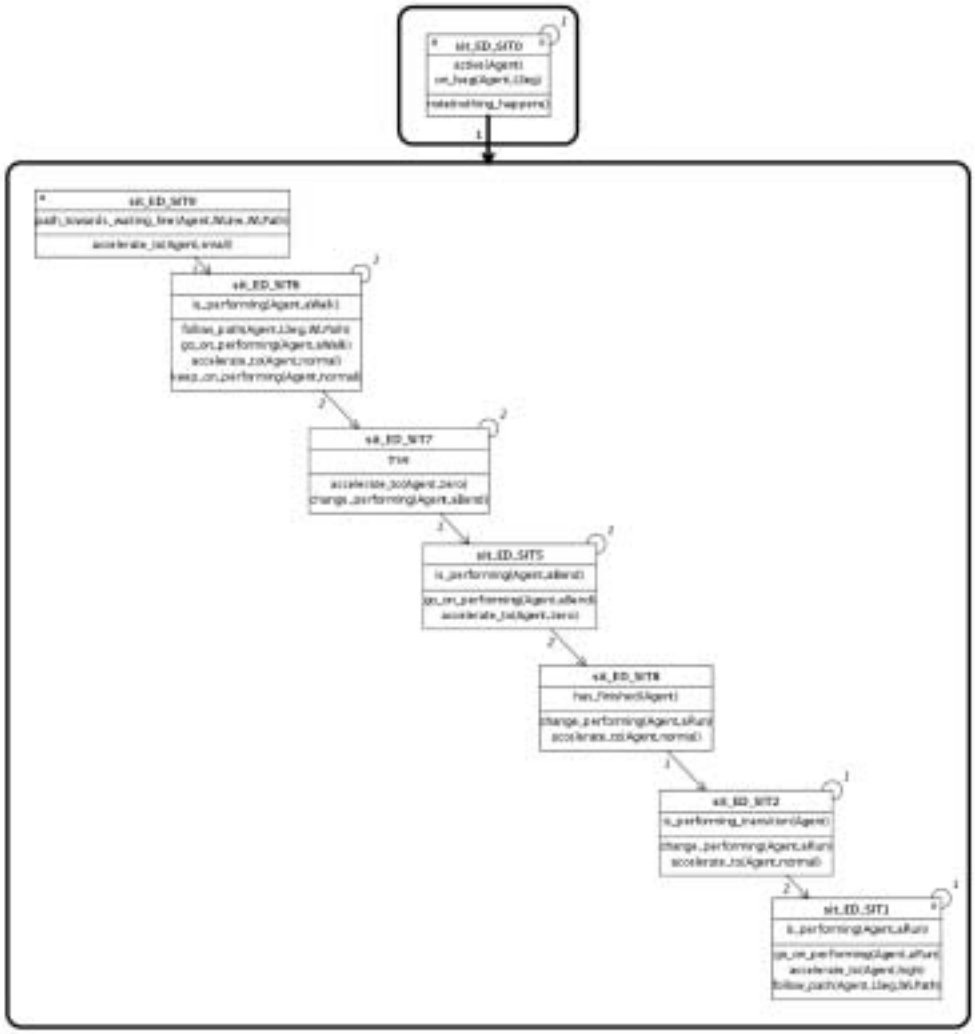


Figure 5.7: Situation Graph Tree (SGT) modeling the *thief sequence* for synthesis purposes.

another situation graph comprising seven situation schemes, which define deterministically the sequence actions to be performed by a virtual agent.

The first scheme (start-situation) determines the path of subway segments through which the human agent will perform its activity, *path_towards_waiting_line(Agent, WLine,WLPath)*. Context predicates are used here for illustrative pruposes, but contextual primitives are not required for activity modeling. The reaction predicate

accelerate_to(Agent,small) establishes that the speed value is set to the conceptual value of *small*. Therefore, the position of the agent will be updated by the human movement model according to this value.

The second situation scheme establishes that the agent goes on performing the *aWalk* action (*go_on_performing(Agent,aWalk)*) during the period of time determined by the conceptual value *normal*, *keep_on_performing(Agent,normal)*. As a result, the *aPosture* parameter increments its value and, as *aWalk* is a cyclic action, such a parameter is set to 0.0 when it reaches 1.0. Also, the orientation of the agent is updated to walk through the list of subway segments of *WLPath*, *follow_path (Agent,Lseg,WLPtah)*.

The next situation in temporal order imposes an action transition from the current action, i.e., *aWalk*, to the *aBend* action, *change_performing(Agent,aBend)*. Also, the predicate *accelerate_to(Agent,zero)* establishes that the agent diminishes its speed to reach zero.

The fourth situation shcheme is first instantiated when the action transition to the *aBend* action is finished. Therefore, this situation remains instantiated while the agent performs the *aBend* action (*go_on_performing(Agent,aBend)*) or, in other words, until the *aLabel* parameter of the agent state is set to *aFinish* by the human movement model.

The next two situations generate the transition from the *aBend* to the *aRun* action. First, once the agent has performed the *aBend* action, *has_finished(Agent)*, s/he starts the transition to the *aRun* action, which is implemented by executing the reaction predicate *change_performing(Agent,aRun)*. Also, the speed of the agent is increased to generate spatial displacement within the scene, *accelerate_to(Agent,normal)*. Subsequently, the fifth situation generates the transition postures until the first posture of the *aRun* manifold is reached, *is_ performing_transition(Agent)*.

The end situation is instantiated while the agent performs the *aRun* action. As a result, the *aPosture* parameter is increased at each time step, and set to 0.0 when it reaches 1.0, *go_on_performing(Agent,aRun)*. Also, the agent updates its speed parameter to increasingly reach a high value (*accelerate_to(Agent,high)*). Lastly, the orientation parameter of the agent is also modified in order to run through the list of subway segments established by *WLPath*, *follow_path(Agent,LSeg,WLPath)*.

Table 5.9 shows the sequence of synthetic agent states which results of applying a graph traversal to the SGT of Fig. 5.7. In Fig. 5.8, a set of human postures is shown in temporal order which reproduces the synthetic *thief sequence*. At the bottom of each human posture, its corresponding *aLabel* and *aPosture* parameters are displayed.

5.3.3 Human Behavior Synthesis

We present in this section an SGT which models a specific behavior in a subway station for synthesis purposes. That is, information about the spatial properties of the agent within the scene can be described with respect to static objects modeled

Frame	Agent State						
1235	has_status(**agent_1**,	6.60,	6.99,	14.93,	1.50,	aWalk,	0.0).
1236	has_status(**agent_1**,	6.46,	7.00,	14.93,	1.50,	aWalk,	0.05).
1237	has_status(**agent_1**,	6.49,	7.01,	14.93,	1.22,	aWalk2aBend,	-0.95).
1238	has_status(**agent_1**,	6.51,	7.02,	14.93,	1.00,	aWalk2aBend,	-0.85).
1239	has_status(**agent_1**,	6.53,	7.02,	14.93,	0.82,	aWalk2aBend,	-0.75).
⋮	⋮						
1247	has_status(**agent_1**,	6.60,	7.04,	14.93,	0.20,	aWalk2aBend,	-0.05).
1248	has_status(**agent_1**,	6.61,	7.04,	14.93,	0.16,	aBend,	0.0).
1249	has_status(**agent_1**,	6.61,	7.04,	14.93,	0.13,	aBend,	0.05).
⋮	⋮						
1267	has_status(**agent_1**,	6.62,	7.04,	14.93,	0.00,	aBend,	0.95).
1268	has_status(**agent_1**,	6.62,	7.04,	14.93,	0.00,	aFinish,	0.0).
1269	has_status(**agent_1**,	6.62,	7.04,	14.93,	1.09,	aFinish2aRun,	-0.95).
1270	has_status(**agent_1**,	6.64,	7.05,	29.56,	1.98,	aFinish2aRun,	-0.85).
⋮	⋮						
1278	has_status(**agent_1**,	6.79,	7.59,	106.17,	5.18,	aFinish2aRun,	-0.05).
1279	has_status(**agent_1**,	6.77,	7.69,	112.12,	5.33,	aRun,	0.0).
1280	has_status(**agent_1**,	6.72,	7.79,	117.50,	5.45,	aRun,	0.05).

Table 5.9: *Thief sequence* described in terms of the synthetic agent states generated by the SGT of Fig. 5.7. This sequence is defined as the concatenation of the *aWalk*, *aBend* and *aRun* actions. The agent state is defined as in (5.1).

at the conceptual subway model. Therefore, we allow to instantiate situations which characterizes semantically an agent activity by incorporating context information.

Considering the conceptual model of the subway station shown in Fig. 5.4, graph traversal applied to the SGT will generate a sequence of synthetic agent states, which synthesize the modeled behavior. Specifically, such a behavior is defined as:

- Given an initial state of a given agent, such an agent is driven to walk towards the *waiting_line* segment.
- At the *waiting_line* segment, the SGT checks for a free ticket machine (i.e., a *ticket_machine* segment in which no agent is present). If a free ticket machine is found, the agent is driven to walk towards the segment which corresponds to such a ticket machine. If no free ticket machine is found, the agent waits at the *waiting_line* segment until a ticket machine is found free.
- The agent stands at the *ticket_machine* segment during a predefined period.
- Once the agent leaves the *ticket_machine* segment, the SGT drives such an agent to walk towards an *entrance_segment* segment. There, the agent is driven to walk through its nearest *accessing_gate* segment, thus entering into the *inside_station* segment of the conceptual subway model.

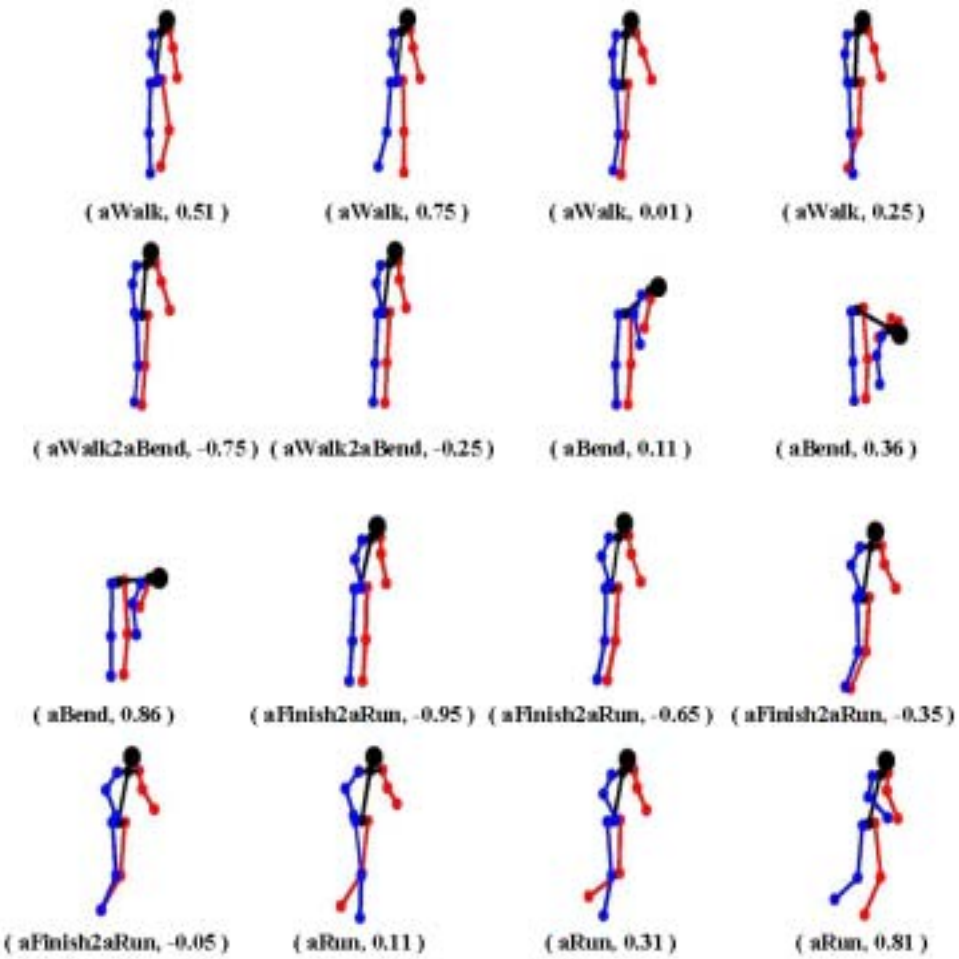


Figure 5.8: Automatic generation of a synthetic *thief sequence*. Only a small set of synthetic human postures is shown here in temporal order.

- At the *inside_station* segment, the agent performs the *aJump* action.

First, we describe how this behavior is modeled using SGTs. Subsequently, we detail the sequence of synthetic agent states for four different agents which will exhibit such a behavior.

These four synthetic sequences will be generated automatically by the SGT. Furthermore, different sequences of agent states are generated for each agent, due to the fact that such agents *interact* themselves at the same time within the subway station. That is to say, the spatial relationship between different actors is taken into account at each time step.

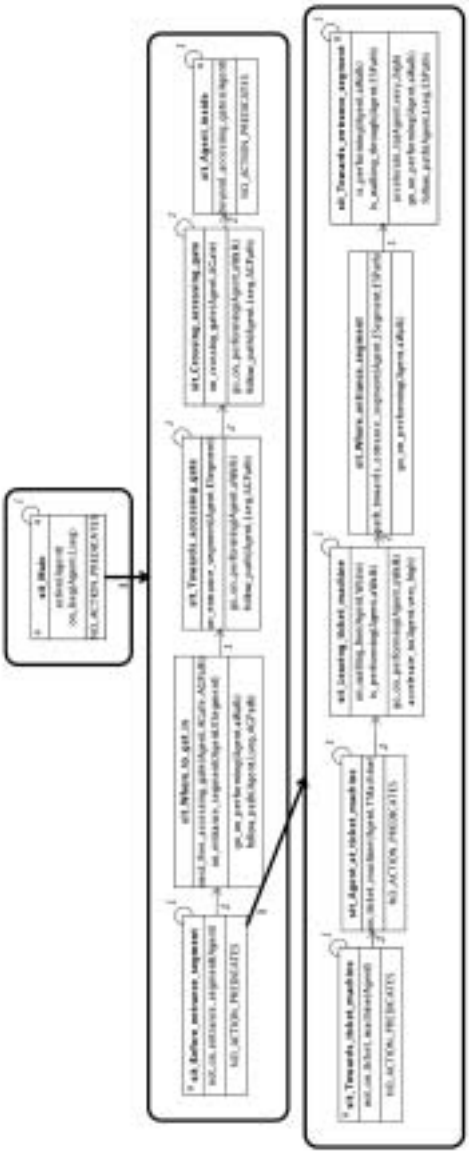


Figure 5.9: SGT for automatic generation of a synthetic behavior in a subway station.

Next, we detail the SGT which represents the particular behavior to be modeled. In Fig. 5.9, the root graph and the first two particularized situation graphs of the resulting SGT are shown.

The root graph comprises only one situation scheme, in which the logic predicate *active(Agent)* is instantiated when the agent state is available at the SDL. The predicate *on_lseg(Agent,Lseg)* determines the segment *Lseg* of the conceptual subway model in which the agent *Agent* is being placed.

The root scheme is particularized by another situation graph which comprise five situation schemes. The *sit_Before_entrance_segment* start-situation is instantiated when the agent is not placed at any *entrance_segment* segment. Once the agent has reached an *entrance_segment* segment, the *sit_Where_to_get_in* situation is instantiated (*on_entrance_segment(Agent,ESegment*), and checks for the nearest crossing gate, *next_free_crossing_gate(Agent,AGate,AGPath)*. As a result, two reaction predicates are instantiated. On the one hand, the *go_on_performing(Agent,aWalk)* reaction predicate increments the posture parameter of the agent in order to adopt the next posture described by the *aWalk p–action*, that means, to increment the arc length parameter of the *p–action* corresponding to *aWalk*. On the other hand, the *follow_path(Agent, Lseg, AGPath)* reaction predicate modifies the orientation of the agent in order to go from the current position of the agent at the *Lseg* segment towards its next subway segment in the segment list of *AGPath*, i.e., an *accessing_gate* segment. The next situation scheme, which is called *sit_Towards_accessing_gate*, is instantiated whether the agent is placed at any *entrance_segment* segment. Consequently, the orientation, position and *aPosture* parameters are modified to drive the agent to walk towards the *accessing_gate* segment *AGate*. When the agent reaches the *AGate* segment (*on_crossing_gate(Agent,AGate)*), the fourth situation scheme (*sit_Crossing_accessing_gate*) is instantiated. This situation modifies the agent state to cross such an accessing gate. Lastly, the end-situation *sit_Agent_inside* is instantiated when the agent is placed at the *inside_station* segment, *beyond_accessing_gates(Agent)*.

The start-situation *sit_Before_entrance_segment* is particularized by another situation graph which represents the modeled behavior to be synthesized before the agent reaches any *entrance_segment* segment, and comprises five situation schemes. In this particularized graph, the start-situation *sit_Towards_ticket_machine* is instantiated until the agent reaches a *ticket_machine* segment, *not_on_ticket_machine(Agent)*. When this occurs (*on_ticket_machine (Agent, TMachine)*), the following situation scheme (*sit_Agent_at_ticket_machine)* is instantiated. These two first situation schemes are particularized by additional situation graphs which will be detailed later.

Once the agent has left the *ticket_machine* segment, the situation scheme *sit_Leaving_ticket_machine* is instantiated whenever the agent walks (*is_performing (Agent, aWalk)*) through the *waiting_line* segment, *on_waiting_line(Agent,WLine)*. The reaction predicates modifies the velocity (*accelerate_to(Agent, very_high)*) and *aPosture* parameters (*go_on_performing(Agent, aWalk)*) to drive the synthetic agent to walk through the *waiting_line* segment. When the agent leaves the *waiting_line* segment, the situation scheme *sit_Where_entrance_segment* generates a list of subway segments which constitutes the path from the agent's position to the nearest *entrance_segment* subway segment *next_free_entrance_segment(Agent, ESegment, ESPath)*. Lastly, the situation scheme *sit_Towards_entrance_segment* is instantiated whenever the agent walks through the segments of the list *AGPath*. Consequently,

Figure 5.10: Automatic generation of a synthetic behavior: the situation scheme called *sit_Towards_ticket_machine*.

the state of the synthetic agent is modified at each time step in order to walk (*go_on_performing(Agent, aWalk)*) through the segments of *ESPath* until the agent reaches the segment *ESegment*. This is achieved by modifying the orientation of the agent (*follow_path(Agent,Lseg,ESPath)*) at each time step.

As commented before, the situation scheme *sit_Towards_ticket_machine* is particularized further by another situation graph which is shown in Fig. 5.10. This situation graph comprises three situation schemes which model the behavior of the agent before reaching a *ticket_machine* segment. The start-situation is called *sit_Where_waiting_line* which computes a list of subway segments. This list constitutes the path from the agent's position to the *waiting_line* segment, *path_towards_waiting_line(Agent, WLine, WLPath)*. The situation scheme *sit_Towards_waiting_line* checks whether the agent is walking through the list of segments of *WLPath* towards the segment

WLine. To synthesize this behavior, the agent state is modified to orientate (*follow_path(Agent,Lseg,WLPath)*) the agent to walk (*go_on_performing(Agent, aWalk)*) through the list of segments of *WLPath* by exhibiting a specific velocity, *accelerate_to(Agent,very_high*. When the agent reaches the segment *WLine* (*on_waiting_line (Agent,WLine)*), the situation scheme called *sit_At_ticket_machine* is instantiated. This situation models three different behaviors, each one represented as a particularized situation graph.

The first particularized graph comprises two situation graphs. The first one, called *sit_Free_ticket_machine*, checks for a *ticket_machine* segment in which no agent is present, *next_free_ticket_machine(Agent,TMachine,TMPath)*. If a free *ticket_machine* segment is found and the agent is walking (*is_performing(Agent,aWalk)*), the next situation scheme to be instantiated is called *sit_Towards_free_ticket_machine*. This situation checks whether the agent is placed at any segment of the list of segments *TMPath*. If so, the agent is driven to walk (*go_on_performing(Agent, aWalk)*) towards the segment *TMachine* by modifying the orientation of the synthesized agent state, *follow_path(Agent,Lseg,TMPath)*.

The second particularized graph represents the behavior when both *ticket_machine* segments are occupied by another agent. This is represented by one situation scheme, called *sit_No_free_ticket_machine*, which is instantiated when no free *ticket_machine* segments are found, *no_free_ticket_machines(Agent)*. This situation scheme is particularized by another situation graph which represents the activity of *waiting*, modeled using three situation schemes. The first situation is called *sit_Agent_will_wait*, which modifies the *aPosture* parameter of the agent state to begin the transition towards the *aStand* posture, *change_performing(Agent,aStand)*. As a transition has begun, this situation scheme is instantiated only once. That is, as an action transition is initiated, the next situation expected to be instantiated is *sit_Agent_is_going_to_wait*. This situation checks for an action transition (i.e., a negative value for the *aPosture* parameter, and the $\mathbf{A}_i\mathbf{2}\mathbf{A}_j$ syntax for the *aLabel* parameter) using the state predicate *is_performing_transition(Agent,aStand)*. This situation is instantiated until the transition towards the *aStand* posture is accomplished or, in other words, while the agent posture moves from the *aWalk p–action* to the *aStand* posture within the *UaSpace*. Once the agent exhibits the *aStand* posture (*is_standing(Agent)*), the end-situation scheme *sit_Agent_is_waiting* is instantiated until a free ticket machine is found.

Lastly, the third particularized graph of *sit_At_ticket_machine* is *sit_Stop_waiting*. In fact, this start-situation is instantiated when a *ticket_machine* segment is found free (*next_free_ticket_machine(Agent,TMachine,TMPath)*), and the agent exhibits a standing posture, *is_standing(Agent)*. In this case, we assume that the agent was waiting for a free *ticket_machine* segment. Therefore, the agent should begin to walk, *change_performing(Agent,aWalk)*. If so, the situation scheme *sit_Towards_new_free_ticket_machine* is instantiated as long as the transition towards the *aWalk p–action* is detected in the agent state, *is_performing_transition(Agent)*. Meanwhile, the orientation of the agent is modified in order to face the free *ticket_machine* segment, *follow_path(Agent,Lseg,TM)*. Also, the velocity parameter of the agent state is increased to move the agent within the subway scene, *accelerate_to(Agent,high)*. When the ac-

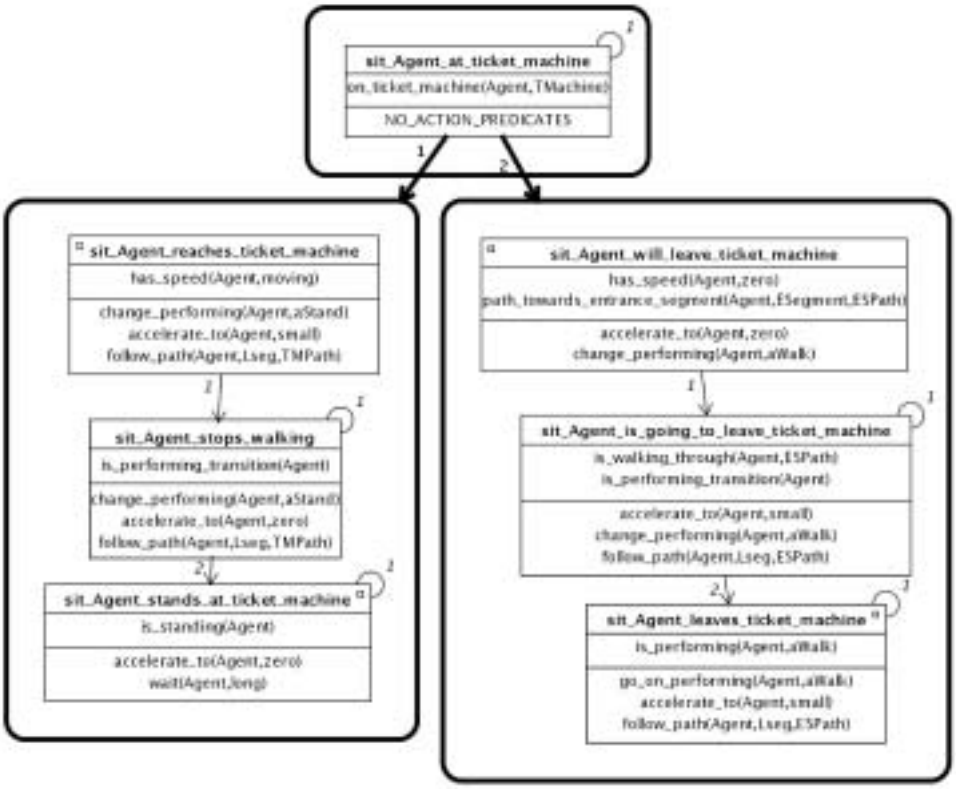


Figure 5.11: Automatic generation of a synthetic behavior: the situation scheme called *sit_Agent_at_ticket_machine*.

tion transition is finished, the agent exhibits the walking action, *is_performing(Agent, aWalk)*. Therefore, the end-situation *sit_Reaching_ticket_machine* is instantiated. As a result of the reaction predicates, the agent is driven to walk (*go_on_performing(Agent, aWalk)*) until the agent reaches the segment *TMachine*. This is achieved by modifying the orientation of the agent (*follow_path(Agent,Lseg,TMPath)*) at each time step.

Next, we detail the situation scheme *sit_Agent_at_ticket_machine* of Fig. 5.9. Such a situation is shown in Fig. 5.11, and is instantiated when the agent position is within a *ticket_machine* segment (*on_ticket_machine_segment(Agent,TMachine)*). Subsequently, two particularized situation graphs are implemented in order to discriminate whether the Agent reaches or leaves the *ticket_machine* segment.

The first particularized graph is composed of three situation schemes, which represents an agent reaching the segment *TMachine*. The start-situation *sit_Agent_reaches_ticket_machine* is first instantiated when the agent exhibits movement (*has_speed(Agent, moving)*). Consequently, the agent is required to change to the *aStand* posture

(*change_performing (Agent,aStand)*), at the same time as setting the velocity parameter value progressively to zero (*accelerate_to(Agent,zero)*). As an action transition is initiated, the next situation to be instantiated is *sit_Agent_stops_walking*, because such a transition is detected in the state of the agent (*is_performing_transition(Agent)*). Once the agent posture has finished its transition towards the *aStand* posture, the end-situation scheme *sit_Agent_stands_at_ticket_machine* is instantiated (*is_standing (Agent)*). Subsequently, the agent is set to wait motionless (*accelerate_to(Agent,zero)*) in the standing posture during the period of time defined by the conceptual term *long* (*wait(Agent,long*). When such a period of time is completed, this situation scheme is not instantiated any more. Consequently, the agent will be driven to leave the *ticket_machine* segment, as explained next.

The second particularized graph is also composed of three situation schemes, and represents an agent leaving the segment *TMachine*. The start-situation is called *sit_Agent_will_leave_ticket_machine*. If the agent exhibits no speed (*has_speed(Agent, zero)*), we assume that the agent is going to walk towards the accessing gate. Therefore, *path_towards_entrance_segment(Agent,ESegment,ESPath)* computes a list of subway segments which constitutes the path *ESPath* from the agent's position to the nearest *entrance_segment* subway segment. Furthermore, the reaction predicate *change_performing(Agent,aWalk)* initiates the action transition from the *aStand* posture to the *aWalk p–action*. Consequently, the situation scheme *sit_Agent_is_going_to_leave_ticket_machine* is instantiated at the next time step (*is_performing_transition(Agent*). Also, this situation scheme checks whether the agent walks through the list of segments of *ESPath* (*is_walking_through(Agent,ESPath)*). Reaction predicates modify the velocity (*accelerate_to(Agent,small)*), orientation (*follow_path(Agent,Lseg,ESPath*) and *aPosture* (*change_performing(Agent,aWalk)*) parameters of the agent. As a result, a synthetic sequence of states is generated, during which the agent begin to walk towards the *entrance_segment* segment *ESegment*. When the action transition is finished, the end-situation scheme *sit_Agent_leaves_ticket_machine* is instantiated *is_performing (Agent,aWalk)*). Consequently, the agent is driven to walk (*go_on_performing(Agent, aWalk)*) slowly (*accelerate_to(Agent,small)*) within the *ticket_machine* segment towards the *entrance_segment* segment (*follow_path(Agent,Lseg,ESPath)*).

To conclude the model of behavior, the agent will perform the *aJump* action at the *inside_station* segment, as detailed next: the particularized situation graph of the situation scheme *sit_Agent_inside* is shown in Fig. 5.12. Such a situation graph is composed of three situation schemes, which generates the synthetic sequence of the *aJump* action. The start-situation scheme *sit_Agent_will_jump* initiates the action transition from the *aWalk p–action* to the *aJump p–action* (*change_performing(Agent,aJump)*). Consequently, the situation scheme *sit_Agent_is_going_to_jump* is instantiated at the next time step (*is_performing_transition(Agent)*). This situation generates the sequence of agent states, which describes the action transition, and is instantiated until this transition is finished. In this case, the end-situation *sit_Agent_is_jumping* is instantiated, because the agent has begun to execute the *aJump* action (*is_performing (Agent,aJump)*). Consequently, the reaction predicate *go_on_performing(Agent, aJump)* modifies the *aPosture* parameter of the agent in order to generate the sequence of human postures which correspond to the *aJump p–action*.

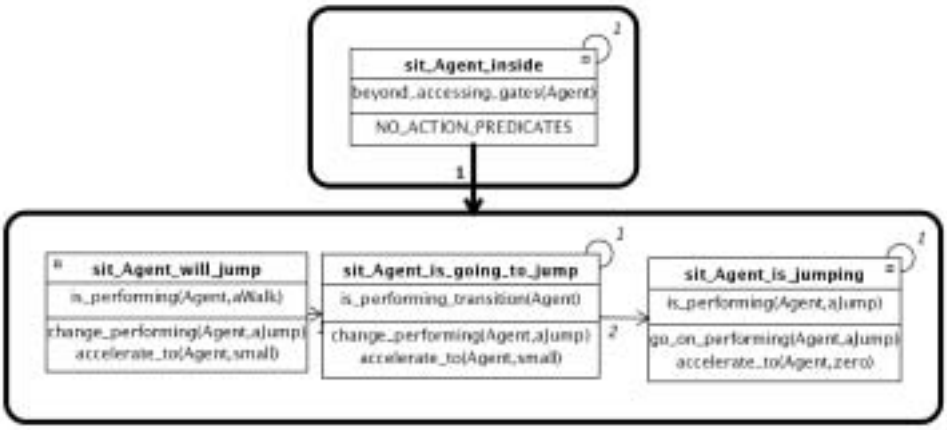


Figure 5.12: Automatic generation of a synthetic behavior: the situation scheme called *sit_Agent_inside*.

Tables 5.10, 5.11, 5.12, and 5.11 show the resulting sequences of states for four synthetic agents, which constitute different characterizations of the behavior modeled at the previously detailed SGT. Note that these four synthetic sequences are generated automatically by applying the traversal of such an SGT. Furthermore, different sequences of states are generated for different agents, due to the fact that such agents *interact* themselves at the same time within the subway station.

Frame	Agent State						
4	has_status(**agent_1**,	2.50,	2.00,	0.00,	1.09,	aWalk,	0.05).
5	has_status(**agent_1**,	2.52,	2.00,	1.04,	2.7,	aWalk,	0.1).
6	has_status(**agent_1**,	2.58,	2.00,	1.89,	4.03,	aWalk,	0.15).
⋮	⋮						
150	has_status(**agent_1**,	16.79,	19.02,	160.44,	6.04,	aWalk,	0.35).
151	has_status(**agent_1**,	16.68,	19.06,	160.44,	5.22,	aWalk2aStand,	-0.95).
152	has_status(**agent_1**,	16.58,	19.09,	160.44,	4.27,	aWalk2aStand,	-0.85).
⋮	⋮						
160	has_status(**agent_1**,	16.22,	19.22,	160.44,	0.86,	aWalk2aStand,	-0.05).
161	has_status(**agent_1**,	16.21,	19.23,	160.44,	0.71,	aStand,	0.0).
162	has_status(**agent_1**,	16.21,	19.23,	160.44,	0.58,	aStand,	0.0).
⋮	⋮						
261	has_status(**agent_1**,	16.14,	19.25,	160.44,	0.0,	aStand,	0.0).
262	has_status(**agent_1**,	16.14,	19.25,	160.44,	0.0,	aFinish,	0.0).
263	has_status(**agent_1**,	16.14,	19.25,	160.44,	0.0,	aFinish2aWalk,	-0.95).
264	has_status(**agent_1**,	16.14,	19.25,	194.6,	0.27,	aFinish2aWalk,	-0.85).
⋮	⋮						
272	has_status(**agent_1**,	16.15,	19.14,	318.09,	1.25,	aFinish2aWalk,	-0.05).
273	has_status(**agent_1**,	16.17,	19.12,	323.84,	1.3,	aWalk,	0.0).
274	has_status(**agent_1**,	16.19,	19.11,	328.57,	1.33,	aWalk,	0.05).
⋮	⋮						
428	has_status(**agent_1**,	42.01,	10.64,	339.99,	10.0,	aWalk,	0.7).
429	has_status(**agent_1**,	42.2,	10.57,	339.99,	8.46,	aWalk2aJump,	-0.95).
430	has_status(**agent_1**,	42.36,	10.51,	339.99,	7.2,	aWalk2aJump,	-0.85).
⋮	⋮						
438	has_status(**agent_1**,	43.05,	10.26,	339.99,	2.65,	aWalk2aJump,	-0.05).
439	has_status(**agent_1**,	43.1,	10.24,	339.99,	2.44,	aJump,	0.0).
440	has_status(**agent_1**,	43.15,	10.23,	339.99,	2.27,	aJump,	0.05).
⋮	⋮						
457	has_status(**agent_1**,	43.71,	10.02,	339.99,	0.86,	aJump,	0.9).
458	has_status(**agent_1**,	43.73,	10.01,	339.99,	0.15,	aJump,	0.95).
459	has_status(**agent_1**,	43.74,	10.0,	339.99,	0.0,	aFinish,	0.0).

Table 5.10: Synthetic agent states generated for **agent_1**.

Frame	Agent State						
32	has_status(**agent_2**,	3.0,	0.5,	0.00,	1.09,	aWalk,	0.05).
33	has_status(**agent_2**,	3.02,	0.5,	4.34,	2.7,	aWalk,	0.1).
34	has_status(**agent_2**,	3.08,	0.5,	7.92,	4.03,	aWalk,	0.15).
⋮	⋮						
237	has_status(**agent_2**,	27.5,	19.01,	11.07,	6.0,	aWalk,	0.3).
238	has_status(**agent_2**,	27.62,	19.03,	11.07,	5.18,	aWalk2aStand,	-0.95).
239	has_status(**agent_2**,	27.72,	19.05,	11.07,	4.24,	aWalk2aStand,	-0.85).
⋮	⋮						
247	has_status(**agent_2**,	28.09,	19.13,	11.07,	0.86,	aWalk2aStand,	-0.05).
248	has_status(**agent_2**,	28.11,	19.13,	11.07,	0.7,	aStand,	0.0).
249	has_status(**agent_2**,	28.12,	19.13,	11.07,	0.57,	aStand,	0.0).
⋮	⋮						
348	has_status(**agent_2**,	28.18,	19.14,	11.07,	0.0,	aStand,	0.0).
349	has_status(**agent_2**,	28.18,	19.14,	11.07,	0.0,	aFinish,	0.0).
350	has_status(**agent_2**,	28.18,	19.14,	11.07,	0.0,	aFinish2aWalk,	-0.95).
⋮	⋮						
359	has_status(**agent_2**,	28.11,	19.24,	162.0,	1.25,	aFinish2aWalk,	-0.05).
360	has_status(**agent_2**,	28.09,	19.25,	167.51,	1.3,	aWalk,	0.0).
361	has_status(**agent_2**,	28.07,	19.25,	172.05,	1.33,	aWalk,	0.05).
⋮	⋮						
611	has_status(**agent_2**,	42.15,	10.64,	341.98,	10.0,	aWalk,	0.55).
612	has_status(**agent_2**,	42.34,	10.58,	341.98,	8.46,	aWalk2aJump,	-0.95).
⋮	⋮						
621	has_status(**agent_2**,	43.2,	10.3,	341.98,	2.65,	aWalk2aJump,	-0.05).
622	has_status(**agent_2**,	43.25,	10.28,	341.98,	2.44,	aJump,	0.0).
⋮	⋮						
641	has_status(**agent_2**,	43.89,	10.07,	341.98,	0.15,	aJump,	0.95).
642	has_status(**agent_2**,	43.90,	10.06,	341.98,	0.0,	aFinish,	0.0).

Table 5.11: Synthetic agent states generated for **agent_2**.

Frame	Agent State						
122	has_status(**agent_3**,	3.0,	1.0,	0.00,	1.09,	aWalk,	0.05).
123	has_status(**agent_3**,	3.02,	1.0,	3.34,	2.7,	aWalk,	0.1).
⋮	⋮						
245	has_status(**agent_3**,	17.49,	17.02,	14.81,	10.0,	aWalk,	0.2).
246	has_status(**agent_3**,	17.68,	17.07,	14.81,	8.19,	aWalk2aStand,	-0.95).
⋮	⋮						
255	has_status(**agent_3**,	18.41,	17.26,	14.81,	1.35,	aWalk2aStand,	-0.05).
256	has_status(**agent_3**,	18.44,	17.27,	14.81,	1.11,	aStand,	0.0).
⋮	⋮						
286	has_status(**agent_3**,	18.56,	17.3,	14.81,	0.0,	aStand,	0.0).
287	has_status(**agent_3**,	18.56,	17.3,	14.81,	0.27,	aStand2aWalk,	-0.95).
⋮	⋮						
296	has_status(**agent_3**,	18.49,	17.8,	128.64,	5.05,	aStand2aWalk,	-0.05).
297	has_status(**agent_3**,	18.42,	17.8,	133.24,	5.23,	aWalk,	0.0).
⋮	⋮						
318	has_status(**agent_3**,	16.3,	19.03,	158.91,	5.99,	aWalk,	0.05).
319	has_status(**agent_3**,	16.18,	19.07,	158.91,	5.17,	aWalk2aStand,	-0.95).
⋮	⋮						
328	has_status(**agent_3**,	15.74,	19.24,	158.91,	0.86,	aWalk2aStand,	-0.05).
329	has_status(**agent_3**,	15.72,	19.25,	158.91,	0.7,	aStand,	0.0).
⋮	⋮						
430	has_status(**agent_3**,	15.65,	19.28,	158.91,	0.0,	aFinish,	0.0).
431	has_status(**agent_3**,	15.65,	19.28,	158.91,	0.0,	aFinish2aWalk,	-0.95).
⋮	⋮						
440	has_status(**agent_3**,	15.67,	19.17,	318.28,	1.25,	aFinish2aWalk,	-0.05).
441	has_status(**agent_3**,	15.69,	19.15,	324.09,	1.3,	aWalk,	0.0).
⋮	⋮						
605	has_status(**agent_3**,	42.11,	10.62,	340.6,	10.0,	aWalk,	0.2).
606	has_status(**agent_3**,	42.3,	10.55,	340.6,	8.46,	aWalk2aJump,	-0.95).
⋮	⋮						
615	has_status(**agent_3**,	43.16,	10.25,	340.6,	2.65,	aWalk2aJump,	-0.05).
616	has_status(**agent_3**,	43.21,	10.23,	340.6,	2.44,	aJump,	0.0).
⋮	⋮						
635	has_status(**agent_3**,	43.84,	10.01,	340.6,	0.15,	aJump,	0.95).
636	has_status(**agent_3**,	43.85,	10.0,	340.6,	0.0,	aFinish,	0.0).

Table 5.12: Synthetic agent states generated for **agent_3**.

Frame	Agent State						
202	has_status(**agent_4**,	2.5,	2.0,	0.00,	1.09,	aWalk,	0.05).
203	has_status(**agent_4**,	2.52,	2.0,	1.04,	2.7,	aWalk,	0.1).
⋮	⋮						
325	has_status(**agent_4**,	17.67,	17.02,	14.59,	10.0,	aWalk,	0.2).
326	has_status(**agent_4**,	17.86,	17.07,	14.59,	8.19,	aWalk2aStand,	-0.95).
⋮	⋮						
335	has_status(**agent_4**,	18.59,	17.26,	14.59,	1.35,	aWalk2aStand,	-0.05).
336	has_status(**agent_4**,	18.61,	17.27,	14.59,	1.11,	aStand,	0.0).
⋮	⋮						
407	has_status(**agent_4**,	18.73,	17.3,	14.59,	0.0,	aStand,	0.0).
408	has_status(**agent_4**,	18.73,	17.3,	14.59,	0.27,	aStand2aWalk,	-0.95).
⋮	⋮						
417	has_status(**agent_4**,	19.27,	17.42,	11.61,	5.05,	aStand2aWalk,	-0.05).
418	has_status(**agent_4**,	19.37,	17.44,	11.5,	5.22,	aWalk,	0.0).
⋮	⋮						
487	has_status(**agent_4**,	27.42,	19.0,	10.95,	6.0,	aWalk,	0.45).
488	has_status(**agent_4**,	27.53,	19.02,	10.95,	5.18,	aWalk2aStand,	-0.95).
⋮	⋮						
497	has_status(**agent_4**,	28.0,	19.11,	10.95,	0.86,	aWalk2aStand,	-0.05).
498	has_status(**agent_4**,	28.02,	19.12,	10.95,	0.7,	aStand,	0.0).
⋮	⋮						
599	has_status(**agent_4**,	28.09,	19.13,	10.95,	0.0,	aFinish,	0.0).
600	has_status(**agent_4**,	28.09,	19.13,	10.95,	0.0,	aFinish2aWalk,	-0.95).
⋮	⋮						
609	has_status(**agent_4**,	28.03,	19.23,	162.02,	1.25,	aFinish2aWalk,	-0.05).
610	has_status(**agent_4**,	28.0,	19.23,	167.54,	1.3,	aWalk,	0.0).
⋮	⋮						
857	has_status(**agent_4**,	42.03,	10.67,	341.63,	10.0,	aWalk,	0.2).
858	has_status(**agent_4**,	42.22,	10.61,	341.63,	8.46,	aWalk2aJump,	-0.95).
⋮	⋮						
867	has_status(**agent_4**,	43.08,	10.32,	341.63,	2.65,	aWalk2aJump,	-0.05).
868	has_status(**agent_4**,	43.13,	10.3,	341.63,	2.44,	aJump,	0.0).
⋮	⋮						
887	has_status(**agent_4**,	43.77,	10.09,	341.63,	0.15,	aJump,	0.95).
888	has_status(**agent_4**,	43.78,	10.08,	341.63,	0.0,	aFinish,	0.0).

Table 5.13: Synthetic agent states generated for **agent_4**.

Chapter 6

Concluding Remarks

6.1 Conclusions

This Thesis is focused on the problem of human motion modeling within the Human Sequence Evaluation (HSE) framework. An HSE system transforms image data into semantic descriptions used for reasoning. This abstraction procedure is also addressed in this contribution, where the final goal is to evaluate human behaviors.

Several topics should be addressed to develop an HSE system, and this complexity suggested to describe these systems by means of a modular scheme. Specifically, we adopted the architecture for Image Sequence Evaluation presented by Nagel in [102].

So we first defined the HSE architecture by establishing several interconnected levels, each level related to an specific knowledge domain. Generically speaking, the system architecture embeds two separated goals: first, to obtain a geometric description of the scene in terms of quantitative values (task related to the Signal, Image, Picture Domain, and Scene Domain levels); and second, to associate obtained geometric descriptions with pre-defined semantic primitives, used for reasoning about observed human behaviors (task related to the Conceptual, Language, and Behavioral levels).

Subsequently, the sources of knowledge required at each level were analyzed to determine the requirements of HSE systems, and to detect possible lacks of constraints. Consequently, we studied the transformation process from image data towards semantic concepts, thus establishing the characteristics of each abstraction level. As a result, a human motion taxonomy based on knowledge concepts was established.

- By considering a knowledge-based classification of human motion representations, we identify four levels of abstraction which are set to correspond to the terms embedded in the taxonomy, namely, movement, action, activity, and behavior.

- The taxonomy reflects the minimal abstraction steps required to derive conceptual descriptions from pixel values.

Based on this taxonomy, we surveyed several human motion analysis papers. Different movement and action representations were described according to the characteristics of the methodology used. Subsequently, we revisited different approaches for activity and behavior analysis. Specifically, we detailed how to represent the transition between different actions, and how to employ the knowledge about the context in order to assist human behavior modeling.

By reviewing related approaches, we stated that the basis of motion understanding is action recognition and description. Therefore, suitable models for human actions are required to be designed in order to recognize predefined movement patterns, and to generate activity and behavior descriptions based on observed actions.

- Despite the fact that some progress has been made in behavior understanding, conceptual interpretation still relies on a proper human action representation.
- Since human motion understanding highly relies on the performance of the human action recognition algorithms, procedures for the semantic interpretation of human behaviors are usually defined in highly constrained domains.

To this end, we have developed a novel human action model which is exploited in different applications. The action model is represented as a sequence of body postures, each posture defined by a novel human body model.

- The human body model embeds facilities for both human posture description and human action synthesis.
- Specifically, the 3-D human body model presented in this Thesis is composed of 12 limbs, and is described using 36 relative polar angles and the height of the hip. The polar space coordinate system has been shown suitable for limb movement description.

We then defined each action using different sequences of human postures, where each sequence was set to correspond to a performance of such an action.

- A human action space, called *aSpace*, was developed to model the principal modes of variation of the posture exhibited during several action performances. Each posture is expressed in terms of a combination of the eigenvectors of the *aSpace*, and a vector of weights.
- Dimensionality reduction is achieved, due to the fact that not all the body parts move when an action is being executed. The minimal number of eigenvectors required to represent a given action depends on the complexity of such an action.

- The *aSpace* is restricted to model only *plausible* human posture variations, because the training data has been obtained from a motion capture system.

Each performance is represented within the *aSpace* as a parametric manifold, so the mean performance can be calculated. Performance representations are not influenced by their duration, expressed in seconds or number of frames. Based on the mean performance, our novel human action model, called *p–action*, was introduced.

- The *p–action* is based on the most characteristic human body postures found during several action performances. These postures, called *key-frames*, are chosen automatically by means of a predefined distance function. In our experiments, the Mahalanobis distance has been proved to be suitable.
- By considering the key-frames, *p–actions* actually represent the *prototypical performance* for a given action. Moreover, as a *p–action* is parameterized, a mechanism of control over our action representation is attained.
- By considering a pose parameterization of the *p–action*, the temporal evolution of the posture depends on the mean speed at which human performances have been recorded. By considering an arc length parameterization, the temporal evolution of the posture is achieved to occur at any user-specified speed.
- Our strategy is not centered on modeling the complete set of feasible postures which an actor could adopt while performing an action, because any new subject can include posture configurations not presented during learning performances.

The benefits of the *aSpace* and *p–action* representation were analysed by addressing different HSE-related applications, such as human action recognition and synthesis, and performance analysis.

- In order to confront action comparison and activity synthesis, we built an unique feature space, called *UaSpace*, within which all learnt human actions are modeled as parametric manifolds. The modes of variation of the *UaSpace* are found repeated for different learnt actions.
- We adapted the key-frame set to perform human action recognition in the *UaSpace*. The distance between two projections in *UaSpace* is a measure of correlation between their corresponding human postures: the closer the points are, the more highly correlated the postures are. Three different classifiers were compared, and the key-frame-based one obtained the best results.

The *p–action* representation was used to synthesize virtual sequences which reproduce any action or activity. As the *p–action* is based on characteristic postures, i.e., the key-frame set, we guarantee that the viewer can recognize which action is being animated.

- Due to the interpolation nature of *p–actions*, synthetic sequences exhibit smooth and realistic movement. Furthermore, interpolation within the *UaSpace* allows activity synthesis. Consequently, transitions between actions, which were not learnt, are synthesized in a continuous and realistic manner.

A comparison framework was introduced in order to describe any action performance with respect to its corresponding action model. To this end, a synchronization procedure is implemented to compare the variation of the angles of specific joints during several performances.

- We addressed the analysis of human actions by comparing different performances of the same action. This analysis of human actions helps to determine those human body model parameters which best characterize an specific action style. Specifically, the *style* of human walking was analysed by establishing differences between a male and a female walkers.

To conclude this investigation, we embedded our human action model within the HSE framework. Specifically, we established an integration procedure addressed in two directions. First, we derived semantic primitives at the Conceptual and Behavior levels, based on the numerical results obtained at the Scene Domain Level. Also, the inversion of this abstraction process was established as the automatic creation of synthetic sequences at the Scene Domain Level, based on the semantic concepts instantiated at the Behavior Level.

- Semantic primitives are derived from the state of the agent, and refer to three different sources of knowledge: the spatial information of the human agent within the scene; the relationship of the agent with respect to its environment; and the action which the agent is performing. For integration, we used the *p–action* and the knowledge about the context to convert the numerical information embedded in the state of the agent into semantic primitives.

We utilized the *Situation Graph Tree* (SGT) formalism to model the knowledge required for human behavior analysis in an specific discourse domain. The basic component of SGTs is called *situation scheme* which comprises semantic primitives about a given agent for a given point in time. The set of all semantic primitives constitute the *a–priori terminology* of our specific discourse domain.

- SGTs operate with fuzzy truth values, which are essential to cope with the uncertainty arising in image evaluation systems. Furthermore, it is possible to explicitly reason about time. Therefore, SGTs provide a deterministic formalism which copes with human behavior modeling, once the human movement patterns involved are properly described in terms of semantic predicates.

We enhance the *terminology* used in [61] to include semantic primitives which *describe* the spatial information about the human agent within the scene, the properties of the actor with respect to its environment, and the action which the actor is performing.

- Based on the terminology, we showed the suitability of SGTs for HSE by modeling a human activity, called the *thief sequence*, and a particular human behavior within a subway station. In the latter case, additional information about the context was incorporated by means of a conceptual scene model.
- Both human activity and behavior models were used to describe observed developments, and to generate automatically synthetic sequences involving virtual actors. SGTs deterministically define which, and in which order, situation schemes should be instantiated to describe or to generate a particular behavior.
- As actors *interact* themselves within the scene, the spatial relationship between different actors is taken into account for behavior description and synthesis. For animation purposes, different synthetic sequences are generated for different virtual actors, and these sequences actually constitute particular characterizations of the behavior modeled at the SGT.

In order to enhance current capabilities of HSE, we require to place our conclusions into a more abstract framework which is independent of current restrictions of engineering-based approaches. To this end, philosophy provides generality and cogitation for identifying the functional limits of proposed methods. Consequently, we next describe the philosophical basis of this Thesis in order to establish the most suitable topics for future research.

6.2 Philosophical Foundations of HSE

Hermeneutics, according to Wilhelm Dilthey, is the art of interpretation of written documents [44]. The name comes from *Hermes*, known as the God of travellers, traders, and thieves, who interpreted the intentions of the Gods to the human beings. In the 19th century, the art of interpretation became significant in literature, philosophy, theology and jurisprudence. Philosophers as Friedrich Schleiermacher and W. Dilthey moved the focus of hermeneutical understanding from written texts to all human productions, such as verbal and nonverbal, historical and current.

For Dilthey, the goal was to respond to the 19th century challenge that all knowledge must follow the model of the physical sciences. He sought to secure for the humanist and cultural disciplines, such as literature and history (the **Geisteswissenschaften**), a status that was different from that of the physical sciences (the **Naturwissenschaften**). To accomplish this task, he formulated the principles of *understanding* in each discipline.

In fact, the core difference between the natural and cultural sciences relies on their respective objects of study (i.e., an object of the world, and another person, respectively), and in the way we understand such objects. Thus, the physical sciences explain nature, and the human studies understand expressions of life. He summarized this difference with his categories of *explanation* (**Erklären**) and *understanding* (**Verstehen**).

Explanation in natural sciences comprehends its object through causal connections: it *knows* its object from the outside. The object remains alien to the human scientist. In contrast, understanding *knows* its object, a human being or a human production, from the inside. That is, we can know the inner life of another person because we also are persons. This is not a knowledge of causal connections, but rather of a network of meanings, analogous to the network of meanings by which we understand ourselves.

Likewise, the Human Sequence Evaluation framework described throughout this Thesis reflects the categories defined by Dilthey. In fact, we consider that *explanation* provides a description of the observed movement by means of the state of the agent. Therefore, explanation is applied to any kind of objects of interest, such as cars or humans, because we characterize observed movement. Besides, we consider that *understanding* performs a conceptual abstraction of the resulting agent state, which derives semantic primitives for deep reasoning about human behaviors. That means, since the objects of interest are human beings, we assume that there are semantic meanings which can be inferred from patterns of movement.

The generation of quantitative descriptions about human movement actually *explains* the developments observed within the scene. That means, the Signal, Image, Picture Domain, and Scene Domain levels generate a geometric description of observed movement in terms of numerical values. Research on Pattern Analysis provides HSE with suitable tools to manipulate quantitatively the different representations used at these levels. As a result of explanation, we characterize *where* movement is being observed within the scene. Only recently, pattern analysis techniques have been used to describe *what* is happening. However, explanation is restricted to describe the causal connections of human movement patterns, i.e., the actions and activities.

Moreover, the knowledge about the context assists the interpretation of the quantitative parameters of the agent state, which allows to *understand* the explanation of observed developments. That means, the integration and manipulation of semantic primitives at the Conceptual, Language, and Behavioral levels provides interpretation and reasoning about observed human behaviors. Research on Artificial Intelligence supplies HSE with suitable techniques to manipulate networks of meanings [125]. Consequently, the final challenge is not only to describe what is happening within a scene, but to reason about *why* the observed movement is happening. To this end, human behavior models should incorporate automatic learning capabilities, and *intelligent agents* (i.e., agents with some knowledge about their internal states, plans, goals and beliefs) should be considered.

6.3 Future Work

The topics covered in this Thesis provides a number of areas of interest which may worth further investigation. Some of them have been already refereed throughout this document. We here group these possible lines of research according to the human motion taxonomy established for HSE. Besides of these lines of research, the algorithms

proposed here should be validated in different and larger human action databases, as well as in different contexts.

Among others, it may be interesting to consider the following lines for further research:

The Human Movement Model

Our representation of the human body is obtained from an optical motion capture system. However, to obtain accurately the 3-D coordinates of the body limbs within an uncontrolled environment is (still) unrealistic.

Instead, model-driven tracking procedures should be implemented in order to instantiate the parameters of the 3-D human body model presented in this Thesis, by using the information embedded in 2-D human body representations designed for visual surveillance purposes. Furthermore, physical restrictions (in terms of allowable degrees-of-freedom, for example) can be incorporated to the body model, thus reducing the set of angle values plausible for each joint.

However, in real-world applications, it is impossible for a single camera to visually detect a moving actor within an extended area. A solution to this problem is to use a network of cameras to cooperatively monitor all the detected actors, also called *multisensor visual surveillance*. Moreover, if each sensor incorporates an HSE system, a cooperative network of HSE systems can be implemented. Thus, the knowledge and representations obtained from different view-points can be combined to robustly instantiate a human movement model with both anatomical and accuracy characteristics.

The Human Action Model

The knowledge embedded in our human action model can be exploited to assist low-level tasks, such as movement segmentation. Due to the constrained nature of limb movement during action performances, tracking processes can be designed to predict the posture of the actor, once the action has been identified. For this purpose, motion information should be incorporated to the action representation.

As an action is defined as a sequence of movements, and each movement is represented using a human body model, one important issue remains open: is really compulsory to consider the 15 joints of the body model proposed in this Thesis to develop action representations? And the answer is *no*: since a reduction of dimensionality over the action representation has been attained, it is necessary to determine the minimal set of joints which actually characterize a given action. In fact, we can establish different subsets of the joints for each modeled action, depending on those joints which exhibit the most characteristic movements.

The selection of subsets of joints will optimize HSE procedures. For example, and according to the above statements, a subset of joints could be selected to address

human gesture evaluation. In this case, only the variation of the joints corresponding to the upper part of the body would be considered for action representation.

Based on the *aSpace* representation, different extensions can be performed, based on different criteria. For example, future research can be addressed to model the characteristics of individual performers. Therefore, we may plan to build a single *UaSpace* for each person, thus modeling the particular movements of specific actors. Thus, we can synthesize the same action for different people, where each synthesized action will reflect the particular characteristics of the actor, which can help to identify the performer. Therefore, biometric characteristics of human actions are worth to be exploited. Additionally, different action spaces can be built based on the action *style*, in order to perform ergonomic evaluation, to detect fatigue in athletic training processes, and even as a search query in digital libraries.

By only considering the style and the identity of the actor for each performance, we can build a quite huge 3-D human action library. Such a library should be organized in a proper way to provide an easy and fast access to the particular performance which will be synthesized. Therefore, content-based retrieving procedures should be developed, which can be designed based on the key-frame sets.

The Human Activity Model

The main challenge is to re-use motion capture data by assembling and combining learnt actions into an activity sequence. Actions involved in activity sequences should be linked together in such a way as to create realistic, constant animation. In fact, future research should obey the requirements of the computer games industry in recent years, because many games use motion capture technology.

Additionally, several assumptions need to be contemplated when a virtual character needs to interact with an object in its surroundings. In fact, there are different ways to interact with the environment, but it is physically impossible to have a learnt action for each possible way. Therefore, activity representations will be required to be capable of modifying previously learnt actions on-line in order to implement (real-time) interaction.

To this end, we plan to implement semantic primitives associated to specific modifications of action descriptions. These predicates may define the style of the action, the gender of the actor, and even the interaction with respect to predefined objects and another actors within the scene. Thus, we can build a suitable SGT which generates realistic activities exhibiting (conceptually) modified actions.

The Human Behavior Model

Knowledge about the context is the basis of integration, as described throughout this Thesis. Such knowledge can incorporate learning capabilities. Thus, behavior representations can be modified according to observed developments, in order to best evaluate what is happening within the scene. For this purpose, the terminology

should be automatically updated by identifying new movement patterns (which are found repeated), together with the (contextual) conditions in which these movement patterns are observed.

Also, *intelligent actors* will be considered, that is, human agents with some knowledge about their internal states. This will help to generate complex and rich behavior descriptions. Also, internal states, if known in advance, may help also to reduce the uncertainty inherent in integration processes. For this purpose, we need to determine how the context can derive not only the knowledge required to infer the internal state of an observed actor, but also its intentions, beliefs, and goals.

Another interesting domain is automatic behavior modeling using textual scripts. That is to say, the animator *writes* a natural language text which describes (roughly) a particular behavior in an specific scene. Subsequently, the input text is processed so that the situations are identified. These situations are temporally and contextually described using the input text, so they can be represented using SGTs. By applying traversal, the synthetic sequence will visualize a rough version of the behavior. Such sequence contain the sequence of prototypical human movements which will be refined later, according to the animator's preferences.

In our opinion, human behavior evaluation constitutes the most challenging and important area of future research.

Appendix A

Point Distribution Models

The Point Distribution Model (PDM) [38] is a shape description technique based on a vectorized representation of shapes. Specifically, PDM estimates a statistical model for non-rigid shape variation. By modeling this distribution, we can generate new samples, similar to those in the original training set, and we can also examine the plausibility of new shapes. This statistical modeling for shape variation, and its combination with several image processing techniques, has generated an important number of publications within the HSE domain in the last decade [14, 26, 64].

The construction of an appropriate PDM for a certain type of shape we wish to learn, requires both the selection of a good representation and of an appropriate density estimation method for the distribution of shapes within this representation.

For the representation we can use non-linear or linear models. As usual, if we use a non-linear model we can relax the hypotheses and obtain higher reliability. This precision has a high cost in the training stage, due to the undeterministic characteristic of non-linear models, and also in the application stage, due to the fact that non-linear algorithms are computationally expensive. This is justifiable for a large number of problems, so several non-linear models have been proposed [65, 131, 132]. However, a linear representation is still a common choice for their speed and straightforward interpretability. As a matter of fact, most non-linear representations are applied over a linear representation which previously performs the dimensionality reduction.

On the other hand, even when the training set generates complex distributions, a linear representation can be used and complexity charged to the statistical model [37]. The most successful representation so far is the one obtained through Principal Component Analysis (PCA). By projecting a shape in a previously learnt PCA space, a set of coefficients (the principal components) is obtained, which control the variation along the maximum variance directions. Consequently, we can naturally associate each principal component to a mode of variation of the shape.

Next, we describe the principal steps required to build a linear PDM. Consider a set of n shapes $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n\}$. Each shape, $\mathbf{s}_j$, is described by a vector of p

coordinate pairs:

$$\mathbf{s}_j = (x_1, y_1, ..., x_p, y_p)^T, \tag{A.1}$$

which represents the set of Cartesian coordinates of strategically chosen landmarks specifying that shape in the image. The set of n training vectors should be aligned (that is, translated, rotated and scaled) using a weighted least-squares algorithm, and the mean shape is calculated by finding the mean position of each landmark:

$$\bar{\mathbf{s}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{s}_i. \tag{A.2}$$

The modes of variation are represented by using PCA on the deviation of the shapes from the mean. So the covariance matrix is computed as:

$$\Sigma = \frac{1}{n}\tilde{\mathbf{S}}\tilde{\mathbf{S}}^T, \tag{A.3}$$

where $\tilde{\mathbf{S}} = \{\mathbf{s}_1 - \bar{\mathbf{s}}, \mathbf{s}_2 - \bar{\mathbf{s}}, ..., \mathbf{s}_n - \bar{\mathbf{s}}\}$.

Using eigenvector decomposition:

$$\lambda_i \mathbf{e}_i = \Sigma \mathbf{e}_i, \tag{A.4}$$

where each eigenvector, $\mathbf{e}_i$, corresponds to a mode of variation, and its corresponding eigenvalue λ_i is related to the variance explained by the eigenvector.

Commonly, when increasing the number of samples, the covariance matrix becomes too large for an easy computation of the eigenvectors. Furthermore, when the number of samples is much smaller than the size of each sample, this calculation is not efficient. In these cases, the *Singular Value Decomposition* algorithm [56] is used instead. The covariance matrix is calculated in a more efficient manner:

$$\hat{\Sigma} = \frac{1}{n}\tilde{\mathbf{S}}^T\tilde{\mathbf{S}}, \tag{A.5}$$

thus resulting in a much smaller covariance matrix. Once the eigenvalues $\hat{\lambda}_i$ and eigenvectors $\hat{\mathbf{e}}_i$ of $\hat{\Sigma}$ are calculated using Eq. (A.4), the eigenvalues λ_i and eigenvectors $\mathbf{e}_i$ of Σ are determined as (details are avoided):

$$\begin{aligned} \lambda_i &= \hat{\lambda}_i, \\ \mathbf{e}_i &= \hat{\lambda}_i^{-1/2}\tilde{\mathbf{S}}\hat{\mathbf{e}}_i. \end{aligned} \tag{A.6}$$

Subsequently, the smallest number m of eigenvalues which explains a large proportion of the total variance is chosen. This value is commonly determined by eigenvalue

thresholding. That is, if we need to guarantee that the first m eigenvectors actually model, for example, 95% of the overall variance of the samples, we should take m so that:

$$\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq 0.95, \tag{A.7}$$

where the denominator represents the the overall variance of the training samples.

However, when n is extremely large, not all the eigenvalues can be computed. So different alternatives can be found in the literature. In [16], a Bayesian treatment of PCA is presented, reformulated as the maximum likelihood solution of a latent variable model. The idea is to introduce a Gaussian latent variable. Thus, the n-dimensional samples can be described as a linear transformation of the m-dimensional latent variable, where the parameters of this relationship need to be instantiated. The resulting parameter values determine the best dimensionality m. However, this process is an iterative, highly time-consuming method. As an extension of this work, in [94] each possible dimensionality is scored by means of Bayesian model selection, and the maximum is selected as the optimum dimensionality.

Establishing the dimensionality m implies to determine the m most significant deformations of the shape. These deformations are found by choosing the eigenvectors corresponding to the m largest eigenvalues. Thus, training data noise can be discarded. Commonly, m is much smaller than n so a very compact model is built.

Consequently, each deformed shape $\hat{\mathbf{s}}$ is found as a combination of:

$$\hat{\mathbf{s}} = \bar{\mathbf{s}} + \mathbf{E}\mathbf{b}, \tag{A.8}$$

where $\mathbf{E} = (\mathbf{e}_1, ..., \mathbf{e}_m)$ corresponds to the selected eigenvectors, and $\mathbf{b} = (b_1, ..., b_m)^T$ is a vector of weights. Each weight is computed within suitable limits in order to generate new allowable shapes, similar to those in the training set. The limits are related to the variance of each variation mode, so each weight b_k is restricted to lie within:

$$-3\sqrt{\lambda_k} \leq b_k \leq 3\sqrt{\lambda_k}. \tag{A.9}$$

Thus, the model is restricted to plausible shapes, so all invalid shapes are excluded. Also, if using real-life training data, a realistic model of deformation is obtained.

Appendix B

Terminology Predicates for Situation Schemes

In this Appendix, we describe the conceptual predicates used for the Situation Graph Trees presented in chapter 5, i.e. the *thief sequence* activity and the *subway station* behavior. Basically, state predicates described here refer to human actions and interactions with other agents or with modeled static objects within the scene.

B.1 State Predicates Concerning Only the Actor

B.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.

B.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*).

B.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*.

B.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(Subway_seg) : the subway segment **Subway_seg** is not occupied by any agent.

agent_on_ticket_machine(Agent, TMachine) : there is an agent **Agent** on the subway segment **TMachine**, which is labeled as a *ticket_machine*.

agent_on_accessing_gates(Agent, AGate) : there is an agent **Agent** on the subway segment **AGate**, which is labeled as an *accessing_gate*.

B.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 subway segments constituing the path from the agent to **WLine**. The last subway segment of **WLPath** is labeled as a *waiting_line* segment.

path_towards_entrance_segment(Agent, ESegment, ESPath) : from the position of the agent **Agent**, **ESegment** is the next *entrance_segment* segment. Then, **ESPath** holds the list of lane segments constituing the path from the agent to **ESegment**. The last subway segment of **ESPath** is labeled as an *entrance_segment*.

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

not_on_ticket_machine(Agent) : the agent **Agent** does not reside on a subway segment which is labeled as a *ticket_machine*.

on_ticket_machine(Agent, TMachine) : the agent **Agent** is positioned on the subway segment **TMachine**, which is labeled as a *ticket_machine*.

standing_at_ticket_machine(Agent, TMachine) : the agent **Agent** is *standing* on the subway segment **TMachine** which is labeled as a *ticket_machine*.

next_free_ticket_machine(Agent, TMachine, TMPath) : from the position of the agent **Agent**, **TMachine** is the next subway segment which is not occupied and labeled as a *ticket_machine*. Then, **TMPath** holds the list of subway segments constituing the path from the agent to that subway segment. The last subway segment is labeled as a *ticket_machine* and the second last is labeled as a *waiting_line*.

no_free_ticket_machines(Agent) : all subway segments labeled as *ticket_machine* segments are being currently occupied by agents.

not_on_waiting_line(Agent) : the agent **Agent** does not reside on a subway segment which is labeled as a *waiting_line*.

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

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

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

next_free_accessing_gate(Agent, AGate, AGPath) : from the position of the agent **Agent**, **AGate** is the next subway segment which is not occupied and labeled as an *accessing_gate*. Then, **AGPath** holds the list of subway segments constituing the path from the agent to that subway segment. The last subway segment is labeled as a *accessing_gate* and the second last is labeled as an *entrance_segment*.

not_on_entrance_segment(Agent, AGate_seg) : the agent **Agent** is not positioned on the subway segment **AGate_seg**, which is labeled as an *entrance_segment*.

on_entrance_segment(Agent, AGate_seg) : the agent **Agent** resides on the subway segment **AGate_seg**, which is labeled as an *entrance_segment*.

not_on_crossing_gate(Agent, AGate) : the agent **Agent** is not positioned on the subway segment **AGate**, which is labeled as an *accessing_gate*.

on_crossing_gate(Agent, AGate) : the agent **Agent** resides on the subway segment **AGate**, which is labeled as an *accessing_gate*.

beyond_accessing_gates(Agent) : the agent **Agent** is presently positioned on a subway segment labeled as *subway_platform*.

on_subway_seg(Agent, Subway_seg) : the agent **Agent** resides on the subway segment **Subway_seg**.

B.4 Reaction Predicates

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

B.5 Defining the Conceptual Scene Model

B.5.1 (Factual) Predicates

point($<x>, <y>, <p>$) : the individual $<p>$ is a point in the subway 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 $<seg>$.

subway_segment($<seg>$) : the individual $<seg>$ constitutes a subway segment.

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

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

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

accessing_gate($<seg>$) : the individual $<seg>$ is labeled to be an *accessing_gate* segment.

subway_platform($<seg>$) : the individual $<seg>$ constitute a *subway_platform* segment.

B.5.2 (Factual) *Precomputed* Predicates

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

not_a_ticket_machine($<seg>$) : the subway segment $<seg>$ is not labeled to be a *ticket_machine* segment.

not_an_accessing_gate($<seg>$) : the subway segment $<seg>$ is not labeled to be an *accessing_gate* segment.

not_a_waiting_line($<seg>$) : the subway segment $<seg>$ is not labeled to be a *waiting_line* segment.

not_an_entrance_segment($<seg>$) : the subway segment $<seg>$ is not labeled to be a *entrance_segment* segment.

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

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

behind_all_subway_segments($<seg>$) : the subway segment $<seg>$ lies behind all subway segments, such as the *subway_platform* segment.

Appendix C

Publications

- Jordi Gonzàlez, Javier Varona, and F.Xavier Roca. **Action Recognition in Application Domains**. In Master Thesis, CVC Technical Report 013, Universitat Autònoma de Barcelona, September 1999.

- Javier Varona, Jordi Gonzàlez, F.Xavier Roca, and Juan J. Villanueva. **iTrack: Image-based Probabilistic Tracking of People**. In A. Sanfeliu, J.J. Villanueva, M. Vanrell, R. Alquézar, O. Eklundh, and Y. Aloimonos, editors, *15th International Conference on Pattern Recognition (ICPR'2000)*, volume 3, pages 1122-1125, Barcelona, Spain, September 2000.

- Javier Varona, Jordi Gonzàlez, F.Xavier Roca, and Juan J. Villanueva. **Automatic Selection of Keyframes for Activity Recognition**. In H.-H. Nagel, and F.J. Perales, editors, *1st International Workshop on Articulated Motion and Deformable Objects (AMDO'2000)*, pages 173-181, Palma de Mallorca, Spain, September 2000. Lecture Notes in Computer Science (LNCS) 1899, Springer-Verlag: Berlin, Heidelberg, New York/NY.

- Jordi Gonzàlez, Javier Varona, Juan J. Villanueva, and F.Xavier Roca. **Online Human Activity Recognition for Video Surveillance**. In J. Salvador Sánchez, and F. Pla, editors, *Proc. IX National Symposium on Pattern Recognition and Image Analysis (SNRFAI'2001)*, volume 2, pages 255-260, Benicàssim, Castelló, Spain, May 2001.

- Jordi Gonzàlez, Javier Varona, F.Xavier Roca, and Juan J. Villanueva. **Human Activity Learning and Recognition from Appearance**. In N. Callaos, Y.J. Chiu, J. Baina, and L.H. Sheng, editors, *World Multiconference on Systemics, Cybernetics and Informatics (SCI'2001)*, Volume XIII, pages 463-466, Orlando, Florida, USA, July 2001.

- Jordi Gonzàlez, Javier Varona, F.Xavier Roca and Juan J. Villanueva. **aSpaces: Action Spaces for Recognition and Synthesis of Human Actions**. In F.J. Perales, and E.R. Hancock, editors, *2nd International Workshop on Articulated Motion and Deformable Objects (AMDO'2002)*, pages 189-200, Palma de Mallorca, Spain, November 2002. Lecture Notes in Computer Science (LNCS) 2492, Springer-Verlag: Berlin, Heidelberg, New York/NY.

- Javier Varona, Jordi Gonzàlez, F.Xavier Roca and Juan J. Villanueva. **Appearance Tracking for Video Surveillance**. In F.J. Perales, A.J.C. Campilho, N. Pérez de la Blanca, and A. Sanfeliu, editors, *1st Iberian Conference on Pattern Recognition and Image Analysis (ibPRIA'2003)*, pages 1041-1048, Port d'Andratx, Mallorca, Spain, June 2003. Lecture Notes in Computer Science (LNCS) 2652, Springer-Verlag: Berlin, Heidelberg, New York/NY.

- Jordi Gonzàlez, Javier Varona, F.Xavier Roca and Juan J. Villanueva. **Automatic Keyframing of Human Actions for Computer Animation**. In F.J. Perales, A.J.C. Campilho, N. Pérez de la Blanca, and A. Sanfeliu, editors, *1st Iberian Conference on Pattern Recognition and Image Analysis (ibPRIA'2003)*, pages 287-296, Port d'Andratx, Mallorca, Spain, June 2003. Lecture Notes in Computer Science (LNCS) 2652, Springer-Verlag: Berlin, Heidelberg, New York/NY.

- Jordi Gonzàlez, Javier Varona, F.Xavier Roca and Juan J. Villanueva. **Human Sequence Evaluation: towards Knowledge-based Scene Interpretations**. In I. Aguiló, L. Valverde, and M.T. Escrig, editors, *6th Catalan Conference on Artificial Intelligence (CCIA'2003)*, pages 168-177, Palma de Mallorca, Spain, October 2003. Frontiers in Artificial Intelligence and Applications 100, IOS Press: Amsterdam, Berlin, Oxford.

- Jordi Gonzàlez, Javier Varona, F.Xavier Roca and Juan J. Villanueva. **A Human Action Comparison Framework for Motion Understanding**. In I. Aguiló, L. Valverde, and M.T. Escrig, editors, *6th Catalan Conference on Artificial Intelligence (CCIA'2003)*, pages 331-340, Palma de Mallorca, Spain, October 2003. Frontiers in Artificial Intelligence and Applications 100, IOS Press: Amsterdam, Berlin, Oxford.

- Jordi Gonzàlez, Javier Varona, F.Xavier Roca and Juan J. Villanueva. **Human Action Modeling based on Key-frames**. Submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bibliography

[1] J.K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[2] K. Akita. Image sequence analysis of real world human motion. *Pattern Recognition*, 17(1):73–83, 1984.

[3] A. Ali and J.K. Aggarwal. Segmentation and recognition of continuous human activity. In *IEEE Workshop on Detection and Recognition of Events in Video*, pages 28–35, Vancouver, Canada, 2001.

[4] M. Allmen and C.R. Dyer. Cyclic motion detection using spatiotemporal surfaces and curves. In *Proceedings of International Conference on Pattern Recognition*, pages 365–370, 1990.

[5] J. Aloimonos. Purposive and qualitative active vision. In *Proceedings of International Conference on Pattern Recognition (ICPR'90)*, pages 346–360, 1990.

[6] M. Arens. *SGTEditor v1.0 Reference Manual v1.1.* Institut für Algorithmen und Kognitive Systeme, Fakultät für Informatik der Universität Karlsruhe (TH), April 2003. http://cogvisys.iaks.uni-karlsruhe.de/Vid-Tex/sgt_editor/ SGTEditor-Manual_v1.1.pdf .

[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. See, too, [6].

[8] O. Arikan and D.A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, July 2002.

[9] D. Ayers and M. Shah. Monitoring human behavior from video taken in an office environment. *Image and Vision Computing*, 19(12):833–846, 2001.

[10] N.I. Badler, C.B. Phillips, and B.L. Webber. *Simulating Humans. Computer Graphics Animation and Control.* Oxford University Press, 1993.

[11] D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[12] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[13] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):42–77, February 1994.

[14] A. Baumberg and D. Hogg. An efficient method for contour tracking using active shape models. In *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 194–199, Austin, 1994.

[15] J. Ben-Aire, Z. Wang, P. Pandit, and S. Rajaram. Human activity recognition using multidimensional indexing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(8):1091–1104, 2002.

[16] C. M. Bishop. Bayesian PCA. In S. A. Solla, M. S. Kearns, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 382–388. MIT Press, 1999.

[17] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[18] M.J. Black and A.D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[19] M.J. Black and A.D. Jepson. A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *Proceedings European Conference Computer Vision (ECCV'98)*, pages 909–924, Freiburg, Germany, 1998.

[20] A.F. Bobick. Movement, activity and action: The role of knowledge in the perception of motion. In *Royal Society Workshop on Knowledge-based Vision in Man and Machine*, volume 352, pages 1257–1265, London, England, 1997.

[21] A.F. Bobick and J. Davis. An appearence-based representation of action. In *Proceedings of International Conference on Pattern Recognition (ICPR'96)*, volume 1, pages 307–312, 1996.

[22] A.F. Bobick and J. Davis. The representation and recognition of movement using temporal templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(3):257–267, march 2001.

[23] A.F. Bobick and A. Wilson. A state-based approach to the representation and recognition of gesture. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(12):1325–1337, 1997.

[24] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. Appearance-based active object recognition. *Image and Vision Computing*, 18:715–727, 2000.

[25] R. Boulic, R. Mas, and D. Thalmann. A robust approach for the center of mass position control with inverse kinetics. *Journal of Computers and Graphics*, 20(5):693–701, 1996.

[26] R. Bowden, T.A. Mitchell, and M. Sahardi. Non-linear statistical models for the 3d reconstruction of human pose and motion from monocular motion sequences. *Image and Vision Computing*, 18:729–737, 2000.

[27] M. Brand and V. Kettnaker. Discovery and segmentation of activities in video. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):844–851, 2000.

[28] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 994–999, San Juan, Puerto Rico, 1997.

[29] M. Braun. *Picturing time, work of Etienne-Jules Marey, 1830-1904*. University of Chicago Press, Chicago, 1995.

[30] C. Bregler. Learning and recognizing human dynamics in video sequences. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 568–574, San Juan, Puerto Rico, 1997.

[31] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pages 8–15, Santa Barbara, CA, 1998.

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

[33] C. Cédras and M. Shah. Motion-based recognition: A survey. *Image and Vision Computing*, 13(2):129–155, 1995.

[34] Z. Chen and H.J. Lee. Knowledge-guided visual perception of 3D human gait from a single image sequence. *IEEE Trans. On Systems, Man, and Cybernetics*, 22(2):336–342, 1992.

[35] 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.

[36] R.T. Collins, A.J. Lipton, and T. Kanade. Introduction to the special section on video surveillance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):745–746, 2000.

[37] T.F. Cootes and C.J. Taylor. A mixture model for representing shape variability. In Clark A.F., editor, *British Machine Vision Conference 1997 (BMVC'97)*, volume 1, pages 110–119, University of Essex, UK:BMVA, 1997.

[38] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):39–59, 1995.

[39] Comaniciu D, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[40] T.J. Darrell and A.P. Pentland. Space-time gestures. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pages 335–340, New York, 1993.

[41] J. Davis and A.F. Bobick. The representation and recognition of movement using temporal templates. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 928–934, San Juan, Puerto Rico, 1997.

[42] L. Davis, V. Philomin, and R. Duraiswami. Tracking humans from a moving platform. In *Proceedings of International Conference on Pattern Recognition (ICPR'00)*, volume 4, pages 171–178, Barcelona, Spain, 2000.

[43] J. Deutscher, A. Blake, and I. Reid. Articulated motion capture by annealed particle filtering. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR'00)*, volume 2, pages 126–133, Hilton Head Island, NC, June 2000.

[44] W. Dilthey. *Introduction to the Human Sciences. Selected Works, Volume I.* Princeton University Press, Princeton, NJ, USA, 1989.

[45] S.L. Dockstader, M.J. Berg, and A.M. Tekalp. Stochastic kinematic modeling and feature extraction for gait analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(8):962–976, 2003.

[46] A.A. Efros, A.C. Berg, G. Mori, and J. Malik. Recognizing action at distance. In *Proceedings of International Conference on Computer Vision (ICCV'03)*, pages 726–733, 2003.

[47] A. Elgammal, D. Harwood, and L. S. Davis. Nonparametric background model for background subtraction. In *Proceedings European Conference Computer Vision (ECCV'00)*, pages 751–767, Dublin, 2000.

[48] D.J. Fleet, M.J. Black, Y.Yaacob, and A.D. Jepson. Design and use of linear models for image motion analysis. *International Journal of Computer Vision*, 36(3):171–193, 2000.

[49] H. Fujiyoshi and A.J. Lipton. Real-time motion analysis by image skeletonization. In *IEEE Workshop on Applications of Computer Vision (WACV'98)*, pages 15–21, Princenton, NJ, 1998.

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

[51] D.M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[52] D.M. Gavrila and L.S. Davis. 3D model-based tracking of humans in action: A multi-view approach. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pages 73–80, 1996.

[53] M. Girard and A. Maciejewski. Computational models for the computer animation of legged figures. *Computer Graphics, Proceedings of ACM SIGGRAPH 85*, 19(3):263–270, 1985.

[54] M. Gleicher and N. Ferrier. Evaluating video-based motion capture. In *Proceedings of Computer Animation*, pages 75–80, Geneva, Switzerland, June 2002.

[55] M. Gleicher, H.J. Shin, L. Kovar, and A. Jepsen. Snap together motion: Assembling run-time animation. *ACM Transactions on Graphics*, 22(3):702–702, 2003.

[56] G.H. Golub and C. Reinsch. Singular value decomposition and least squares solution. *Numerische Mathematik*, 14:403–420, 1970.

[57] K. Gould and M.A. Shah. The trajectory primal sketch: a multiscale scheme for representing motion characteristics. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'89)*, pages 79–85, 1989.

[58] B. Guenter and R. Parent. Computing the arc length of parametric curves. *IEEE Computer Graphics and Applications*, 10(3):72–78, May 1990.

[59] Y. Guo, G. Xu, and S. Tsuji. Understanding human motion patterns. In *Proceedings of International Conference on Pattern Recognition (ICPR'94), Track B*, pages 325–329, Viena, 1994.

[60] M. Haag and H.-H. Nagel. Combination of edge element and optical flow estimates for 3d-model-based vehicle tracking in traffic image sequences. *International Journal of Computer Vision*, 35(3):295–319, 1999.

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

[62] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: Who? When? Where? What? A real time system for detecting and tracking people. In *Proceedings of Third International Conference on Automatic Face and Gesture Recognition*, pages 222–227, Nara, Japan, 1998.

[63] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000.

[64] T. Heap and D. Hogg. Extending the point distribution model using polar coordinates. *Image and Vision Computing*, 14:589–599, 1996.

[65] T. Heap and D. Hogg. Improving specificity in PDMs using a hierarchical approach. In Clark A.F., editor, *British Machine Vision Conference 1997 (BMVC'97)*, volume 1, pages 80–89, University of Essex, UK:BMVA, 1997.

[66] B. Heisele, U. Kressel, and W. Ritter. Tracking non-rigid, moving objects based on color cluster flow. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 697–703, San Juan, Puerto Rico, 1997.

[67] J.K. Hodgins, W.L. Wooten, D.C. Brogan, and J.F. O'Brien. Animating human athletics. In *Annual Conference Series, Proceedings of ACM SIGGRAPH 95*, pages 71–78, August 1995.

[68] J. Hoey and J.J. Little. Representation and recognition of complex human motion. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR'00)*, volume 1, pages 752–759, Hilton Head Island, NC, June 2000.

[69] D. Hogg. Model-based vision: A program to see a walking person. *Image and Vision Computing*, 1(1):5–20, February 1983.

[70] S.S. Intille and A.F. Bobick. Representation and visual recognition of complex, multi-agent actions using belief networks. Technical Report 454, M.I.T. Media Laboratory Perceptual Computing Section, 1998.

[71] S.S. Intille and A.F. Bobick. Recognized planned, multiperson action. *International Journal of Computer Vision*, 81(3):414–445, 2001.

[72] S.S. Intille, J.W. Davis, and A.F. Bobick. Real-time closed world tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 697–703, San Juan, Puerto Rico, 1997.

[73] M. Isard and A. Blake. Condensation: Conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[74] S. Iwasawa, K. Ebihara, J. Ohya, and S. Morishima. Real-time estimation of human body posture from monocular thermal images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 15–20, San Juan, Puerto Rico, 1997.

[75] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14(2):210–211, 1973.

[76] S.X. Ju, M.J. Black, and Y. Yacoob. Cardboard people: A parametrized model of articulated image motion. In *Proceedings of Second International Conference on Automatic Face and Gesture Recognition*, pages 38–44, Killington, Vermont, 1996.

[77] I.A. Kakadiaris and D. Metaxas. Model based estimation of 3d human motion with occlusion based on active multi-viewpoint selection. In *Proceedings on Computer Vision and Pattern Recognition*, pages 81–87, San Francisco, CA, 1996.

[78] A. Kale, N. Cuntoor, and R. Chellappa. A framework for activity-specific human recognition. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 3660–3663, Orlando, FL, May 2002.

[79] T. Kanade. Region segmentation: Signal vs. semantics. In *Proceedings Fourth International Joint Conference on Pattern Recognition (IJCPR'78)*, pages 95–105, Kyoto, Japan, November 1978.

[80] I.A. Karaulova, P.M. Hall, and A.D. Marshall. Tracking people in three dimensions using a hierarchical model of dynamics. *Image and Vision Computing*, 20:691–700, 2002.

[81] R. Koch. Dynamic 3-D scene analysis through synthesis feedback control. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(6):556–568, June 1993.

[82] K. Koffka. *Principle of Gestalt Pshycology.* Harcourt Brace, New York, 1935.

[83] 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.

[84] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, July 2002.

[85] M.K. Leung and Y-H Yang. First sight: A human body outline labeling system. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(4):359–377, 1995.

[86] Y. Li, S. Ma, and H. Lu. A multiscale morphological method for human posture recognition. In *Proceedings of Third International Conference on Automatic Face and Gesture Recognition*, pages 56–61, Nara, Japan, 1998.

[87] A.J. Lipton, H. Fujiyoshi, and R.S. Patil. Moving target classification and tracking from real-video. In *IEEE Workshop on Applications of Computer Vision (WACV'98)*, pages 8–14, Princenton,NJ, 1998.

[88] J.J. Little and J.E. Boyd. Recognizing people by their gait: The shape of motion. *VIDERE*, 1(2):2–32, 1998.

[89] C.K. Liu and Z. Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics*, 21(3):408–416, July 2002.

[90] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of International Conference on Computer Vision*, pages 416–425, Vancouver, Canada, July 2001.

[91] O. Masoud and N. Papanikolopoulos. A method for human action recognition. *Image and Vision Computing*, 21(8):729–743, 2003.

[92] S.J. McKenna, Y. Raja, and S. Gong. Tracking colour objects using adaptive mixture models. *Image and Vision Computing*, 17(3-4):225–231, 1999.

[93] A. Menache. *Understanding Motion Capture for Computer Animation and Video Games.* Academic Press, San Diego, CA, 2000.

[94] T.P. Minka. Automatic choice of dimensionality for PCA. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 598–604. MIT Press, 2001.

[95] A. Mitchie and P. Bouthemy. Computation and analysis of image motion: a synopsis of current problems and methods. *International Journal of Computer Vision*, 8(3):29–55, July 1996.

[96] T. Moeslund and E. Granum. A survey of computer vision based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, March 2001.

[97] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.

[98] R.J. Morris and D.C. Hogg. Statistical models of object interaction. *International Journal of Computer Vision*, 37(2):209–215, 2000.

[99] H. Murase and S.K. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[100] E. Muybridge. *The Human Figure in Motion.* Dover Publications, Inc., New York, 1955. First published 1887.

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

[102] H.-H. Nagel. Image sequence evaluation: 30 years and still going strong. In A. Sanfeliu, J.J. Villanueva, M. Vanrell, R. Alquézar, O. Eklundh, and Y. Aloimonos, editors, *Proceedings of International Conference on Pattern Recognition (ICPR'2000)*, volume 1, pages 149–158, Barcelona, Spain, 2000.

[103] H.-H. Nagel, M. Haag, V. Jeyakumar, and A. Mukerjee. Visualisation of conceptual descriptions derived from image sequences. In W. Förstner, J.M. Buhmann, A. Faber, and P. Faber, editors, *Reihe Informatik aktuell*, pages 364–371. Springler-Verlag: Berlin, Heidelberg, New York/NY, September 1999.

[104] A. Nakazawa, H. Kato, and S. Inokuchi. Human tracking using distributed video systems. In *Proceedings of International Conference on Pattern Recognition (ICPR'98)*, volume 1, pages 593–596, Brisbane, Australia, 1998.

[105] H.T. Nguyen, M. Worring, and R. van den Boomgaard. Occlusion robust adaptive template tracking. In *Proceedings of International Conference on Computer Vision (ICCV'01)*, volume 1, pages 678–683, 2001.

[106] S.A. Niyogi and E.H. Adelson. Analyzing and recognizing walking figures in xyt. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 469–474, 1994.

[107] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[108] R. Parent. *Computer Animation. Algorithms and Techniques.* Morgan Kaufmann Publishers, San Francisco, CA, 2002.

[109] F.J. Perales, A. Igelmo, J.M. Buades, P. Negre, and G.Bernat. Human motion analysis & synthesis using computer vision and graphics techniques. Some applications. In *IX Spanish Symposium on Pattern Recognition and Image Analysis*, volume 1, pages 271–277, Benicassim, Spain, 16-18 May 2001.

[110] F.J. Perales and J. Torres. A system for human motion matching between synthetic and real image based on a biometrical graphical model. In *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 83–88, Austin, 1994.

[111] R. Polana and R. Nelson. Low level recognition of human motion. In *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 77–82, Austin, 1994.

[112] R. Polana and R. Nelson. Recognizing activities. In *Proceedings of International Conference on Pattern Recognition (ICPR'94)*, pages 815–818, Viena, 1994.

[113] W. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, Cambridge, 1988.

[114] Y. Raja, S.J. McKenna, and S. Gong. Tracking and segmenting people in varying lighting conditions using colour. In *Proceedings of Third International Conference on Automatic Face and Gesture Recognition*, pages 228–233, Nara, Japan, 1998.

[115] K. Rangarajan, W. Allen, and M. Shah. Matching motion trajectories using scale space. *Pattern Recognition*, 26(4):595–609, 1993.

[116] C. Rao and M. Shah. View-invariant representation and learning of human action. In *IEEE Workshop on Detection and Recognition of Events in Video*, pages 55–63, Vancouver, Canada, 2001.

[117] P. Remagnino, T. Tan, and K. Baker. Agent oriented annotation in model based visual surveillance. In *Proceedings of International Conference on Computer Vision (ICCV'98)*, pages 857–862, Mumbai, India, 1998.

[118] Y. Ricquebourg and P. Bouthemy. Real-time tracking of moving persons by exploiting spatio-temporal image slices. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):797–808, 2000.

[119] K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image Understanding*, 59(1):94–115, 1994.

[120] R. Rosales and S. Sclaroff. 3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, volume 2, pages 117–123, June 1999.

[121] R. Rosales and S. Sclaroff. A framework for heading-guided recognition of human activity. *Computer Vision and Image Understanding*, 91:335–367, 2003.

[122] H.A. Rowley and J.M. Rehg. Analyzing articulated motion using expectation maximization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 935–941, San Juan, Puerto Rico, 1997.

[123] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 2003.

[124] C.L. Sabharwal. An intelligent approach to discrete sampling of parametric curves. In *Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing*, pages 397–401. ACM Press, 1993.

[125] G. Sagerer and H. Niemann. Semantic networks for understanding scenes. In M.D. Levine, editor, *Advances in Computer Vision and Machine Intelligence*. Plenum Press, New York, 1997.

[126] 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.

[127] J. Segen and S. Pingali. A camera-based system for tracking people in real time. In *Proceedings of 13th International Conference on Pattern Recognition (ICPR'96)*, volume 3, pages 63–67, Vienna, Austria, 1996.

[128] S.M. Seitz and C.R. Dyer. Cyclic motion analysis using the period trace. In M. Shah and R. Jain, editors, *Motion-based Recognition*, pages 61–85. Kluwer Academic, Dordretch, The Netherlands, 1997.

[129] H. Sidenbladh and M.J. Black. Learning the statistics of people in images and video. *International Journal of Computer Vision*, 54(1/2):181–207, 2003.

[130] H. Sidenbladh, M.J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Proceedings European Conference on Computer Vision (ECCV)*, volume 1, pages 784–800, Copenhagen, Denmark, 2002. Lecture Notes in Computer Science 2353, Springer-Verlag: Berlin, Heidelberg, New York/NY.

[131] P.D. Sozou, T.F. Cootes, C.J. Taylor, and E.C. Di-Mauro. A non-linear generalization of PDMs using polynomial regression. In Hancock E., editor, *British Machine Vision Conference 1994 (BMVC'94)*, volume 1, pages 397–406, University of York, UK:BMVA, 1994.

[132] P.D. Sozou, T.F. Cootes, C.J. Taylor, and E.C. Di-Mauro. Non-linear point distribution modeling using a multi-layer perceptron. In Pycock D., editor, *British Machine Vision Conference 1995 (BMVC'95)*, volume 1, pages 107–116, University of Birmingham, UK:BMVA, 1995.

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

[134] J. Sullivan, Andrew Blake, Michael Isard, and John MacCormick. Object localization by bayesian correlation. In *Proceedings International Conference on Computer Vision (ICCV'99)*, pages 1068–1075, Corfu, Greece, 1999.

[135] J. Sullivan and S. Carlsson. Recognizing and tracking human action. In *Proceedings of the seventh European Conference Computer Vision (ECCV'02)*, pages 629–644, Copenhagen, Denmark, 2002.

[136] J.K. Tsotsos. Motion understanding: Task-directed attention and representations that link perception with action. *International Journal of Computer Vision*, 45(3):265–280, 2001.

[137] X. Varona, J. Gonzàlez, F.X. Roca, and J.J. Villanueva. *iTrack*: Image-based probabilistic tracking of people. In *Proceedings of International Conference on Pattern Recognition (ICPR'2000)*, volume 2, pages 1122–1125, Barcelona, Spain, 2000.

[138] V.T. Vu, F. Brémond, and M. Thonnat. Human behavior visualisation abd simulation for automatic video understanding. In *Proc. 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG-2002)*, pages 458–492, Plzen-Bory, Czech Republic, 2002.

[139] S. Wachter and H.-H. Nagel. Tracking persons in monocular image sequences. *Computer Vision and Image Understanding*, 74(3):174–192, June 1999.

[140] J.J. Wang and S. Singh. Video analysis of human dynamics. *Real-time Imaging*, 9(5):321–346, 2003.

[141] L. Wang, W. Hu, and T. Tan. Recent developments in human motion analysis. *Pattern Recognition*, 36(3):585–601, 2003.

[142] D. Wiley and J. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6):39–45, 1997.

[143] C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[144] C.R. Wren, B.P. Clarkson, and A.P. Pentland. Understanding purposeful human motion. In *Proceedings of Fourth International Conference on Automatic Face and Gesture Recognition*, pages 378–383, Grenoble, France, March 2000.

[145] M. Yamada, K. Ebihara, and J. Ohya. A new robust real-time method for extracting human silhouettes from color images. In *Proceedings of Third International Conference on Automatic Face and Gesture Recognition*, pages 528–533, Nara, Japan, 1998.

[146] M. Yamamoto, Y. Ohta, T. Yamagiwa, and K. Yagishita. Human action tracking guided by key-frames. In *Proceedings of Fourth International Conference on Automatic Face and Gesture Recognition*, pages 354–361, Grenoble, France, March 2000.

[147] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pages 379–385, Champaign, IL, USA, June 1992.

[148] V.M. Zatsiorsky. *Kinematics of Human Motion*. Human Kinetics, Champaign, IL, USA, 1998.