Automatic Generation of Computer Animated Sequences Based on Human Behavior Modeling

Pau Baiget*, Joan Soto*, Xavier Roca*, Jordi Gonzàlez+

* *Computer Vision Center & Dept. de Ciències de la Computació,*
Edifici O, Campus UAB, 08193 Bellaterra, Spain
+ Institut de Robòtica i Informàtica Industrial (UPC – CSIC),
Llorens i Artigas 4-6, 08028, Barcelona, Spain

Abstract

This paper presents a complete framework to automatically generate synthetic image sequences by designing and simulating complex human behaviors in virtual environments. Given an initial state of a virtual agent, a simulation process generates posterior synthetic states by means of precomputed human motion and behavior models, taking into account the relationships of such an agent w.r.t its environment at each frame step. The resulting status sequence is further visualized into a virtual scene using a 3D graphic engine. Conceptual knowledge about human behavior patterns is represented using the *Situation Graph Tree* formalism and a rule-based inference system called *F-Limette*. Results obtained are very helpful for testing human interaction with real environments, such as a pedestrian crossing scenario, and for *virtual storytelling*, to automatically generate animated sequences.

Keywords: Behavioral Animation, Virtual Environments, Computer Graphics, Human Sequence Evaluation, Artificial Intelligence.

1 Introduction

Virtual Storytelling refers to the generation of synthetic video sequences involving virtual agents who exhibit complex behaviors and interact with each other. This discipline has been widely used in cinematographics helping with the creation of extra actors in animation films [5]; in virtual environments like road traffic simulation [13]; and in the game industry for developing autonomous agents that interact with the user[14].

This paper describes a framework for the automatic generation of synthetic image sequences using virtual human agents. Fig. 1 shows the overall scheme, divided into modules. The *Agent state generator* computes future agent states from the previous ones by simulating human behavior under the constraints of a precomputed motion model. Thus, the i-th agent state is fedback to the system in order to generate the $(i+1)$-th state. Finally, the sequence of states is sent to a 3D rendering engine based on OpenGL [12] which generates a synthetic image sequence.

To achieve this, some a–priori information is initially provided by the user: On the one hand, information about both the behavior of involved virtual agents and the characteristics of the virtual environment allows to contextualize the simulation. On the other hand, the initial states of these virtual agents allows the simulation to proceed. All this knowledge is represented using a set of conceptual predicates in *Fuzzy Metric Temporal Horn Logic* (FMTHL), being agent behavior modeled using the deterministic formalism Situation Graph Tree (SGT)[1].

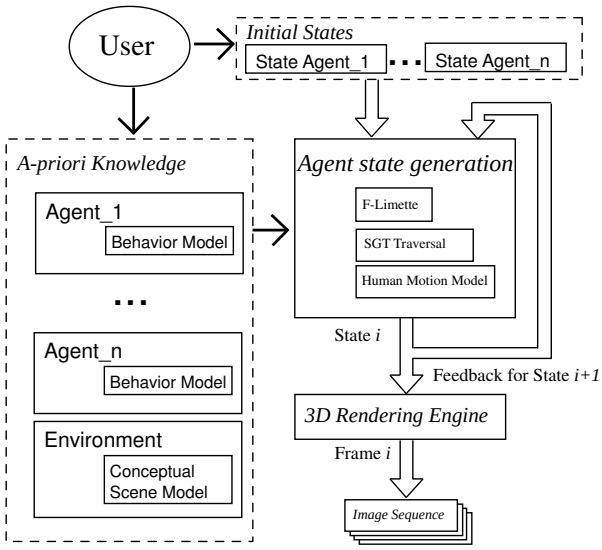


Figure 1: Overall scheme of the proposed framework. More details in the text.

Previous works have modeled human behavior by considering two different approaches: on the one hand, bottom–up approaches make use of machine learning techniques to represent human behaviors from a set of motion patterns. Thus, *behavioral learning* has begun to be exploited using machine learning techniques such as k–Nearest Neighbours [4] or reinforcement learning (RL) [11] , to cite few. On the other hand, top–down techniques predefine a set of behaviour patterns which represent the set of agent behaviors to be synthesized. Top–down approaches cannot learn, but they represent human behaviors in a consistent manner, and they do not depend on the specific data from the training set [2], [8], [9]. However, these models generate predefined trajectories for the whole sequence, without considering the relationships of the agent w.r.t. its environment *at each frame step*. We con-

sider here instead a procedure for the automatic animation of virtual agents which can change adaptively their behavior, depending on their environment at each particular frame step.

This paper is structured as follows: next section introduces the knowledge about human behavior and environment, which is provided beforehand to the system. Section 3 describes the process of automatic generation of behaviors for virtual agents. Section 4 shows the synthetic results obtained modeling pedestrian behaviors in a pedestrian crossing scenario. Finally, Section 5 concludes the paper and discusses future lines of research.

2 Use of A-priori Knowledge

Simulation of a virtual world requires to specify some requirements and restrictions about the agent capabilities and the properties of the virtual environment. Firstly, we describe the state which represents a virtual agent at each time step. Next, we introduce a human motion model which determines the postures to be adopted by the human agents in the environment. Finally, we discuss the representation of the virtual environment within which the agent interacts.

2.1 The State Vector of the Agent

The state vector of the virtual agent is determined by dynamical, positional and postural properties of the human body. In our case, this numerical knowledge comprises the following logic predicate, valid at time step t:

$$t \,!\, has\_status(Agent, x, y, or, vel, aLabel, p).$$

This state vector embeds the 2-D spatial position *pos* of the agent *Agent* in the floor plane, in addition to the velocity *vel* and the orientation *or*. The parameter *aLabel* refers to the action, i.e. *walking*, *standing* and *running*. Finally, the parameter p refers to the temporal evolution of the human body posture of a particular action, as explained next.

2.2 Human Motion Model

The first approach to movement generation has been realized using an abstract human motion model based on the *stick figure*, whose structure is shown in Fig. 2. The state vector already contains information about the action the agent is performing at each time step. Every *has_status* has a posture label which corresponds to a particular pose $p \in [0, 1]$. The posture sequence is represented with the so–called *p–action* representation as a model of human action, see [6] for details. Fig. 3 shows the stick figures corresponding to the running *p–action*.

2.3 Conceptual Scene Model

The state vector of the agent contains information about its position for each frame step. Therefore, some a-priori knowledge about the scene has been provided to reason about the interaction of the agents with components of the scene. In our case,

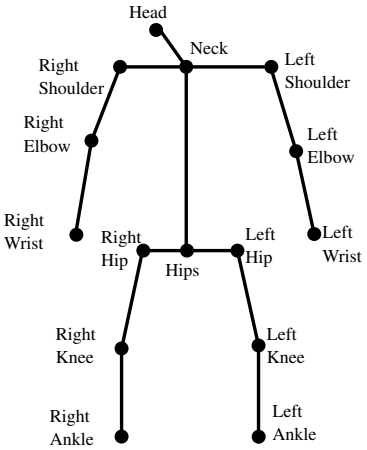


Figure 2: Generic human body model represented using a stick figure similar to [3], here composed of twelve limbs and fifteen joints.

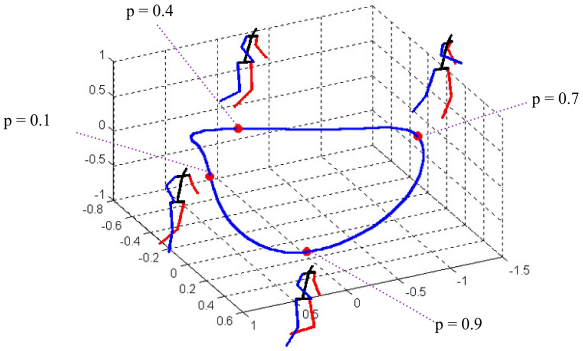


Figure 3: *p–actions* computed in the *aRun aSpace*, see [6] for details: by varying the parameter pose p, we actually move along the manifold, thus obtaining the temporal evolution of the human body posture during the prototypical performance of any learnt action.

a ground-plane representation is used as a virtual scene, see Fig. 4: it is divided into squares which represent the minimum portion of space an agent can occupy, called a *segment*. Thus, an active agent in the

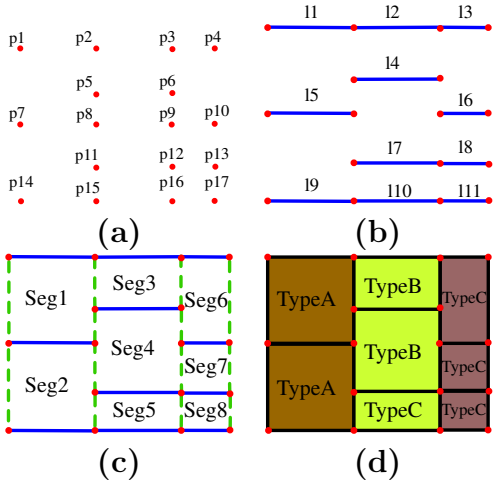


Figure 4: Design steps for the scene model. (a) Points (b) Lines (c) Segments (d) Segment Labels

scene is supposed to be in one of these segments. However, several agents can share a segment in a given time step. Each segment is labeled with a name that represents the part of the scene this segment belongs to. For instance, in a *pedestrian crossing* scenario segments can be labeled as *sidewalk*, *crosswalk*, *waiting line*, or *road*. This information allows the reasoning engine to infer whether an agent is crossing the street across the road or along the crosswalk, for example.

3 Human Behavior Modeling

In this section, we describe the automatic generation of synthetic status sequences based on a human behavior model. Let consider the automatic generation of a spiral motion for a particular agent within a

virtual scene. This first example implies, at the very least, to describe the human behavior model and the automatic path planning generation.

3.1 Fuzzy Metric Temporal Horn Logic

The knowledge involved in our behavior models is defined not by using quantitative data but with semantic concepts, which are represented using FMTHL and its inference engine *F-Limette*, both developed by Schaffer et al. [10]. FMTHL encapsulates the information in the form of logic predicates which have a temporal validity, specified with a frame interval and weighted with a value in the interval $[0, 1]$, thus specifying the degree of confidence for this predicate to be valid. Fig. 5 shows the discretization of the predicate *has_speed (Agent, Concept)* which establishes the correspondence between numerical speed values and fuzzy concepts like *slow*, *normal*, or *high*.

3.2 Situation Graph Trees

The conceptual knowledge about human behavior patterns is encoded in a set of rules in FMTHL and organized within a behavior modeling formalism called *Situation Graph Tree* (SGT). The basic component of SGTs is the *situation scheme*, which embeds the qualitative knowledge for a given agent at each frame step, see Fig. 6. These situation schemes are separated into two parts: the *state scheme* and the *reaction scheme*. On the one hand, the state scheme refers to logic predicates about the state of the agent, which should

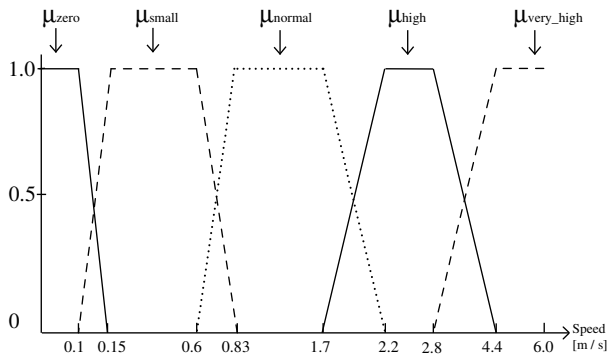


Figure 5: Discretization of continuous speed values into a set of intervals. the graph shows the fuzzy membership functions $\mu_{speedvalue}$ for the subset (*zero, small, normal, high, very_high*) of discrete conceptual speed values.

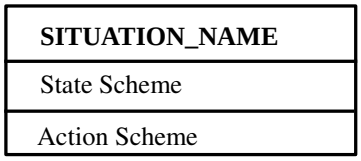


Figure 6: The situation scheme is the basic component of the SGTs. More details in the text.

be satisfied for *instantiating* the situation. On the other hand, the action scheme describes the changes (in terms of logic predicates, too) to be applied to the state vector of the agent when the state scheme is instantiated. See [1] for further details.

SGTs are used to recognize those situations which can be instantiated for a synthetic agent by applying the so-called *graph traversal* [7]. The goal is to determine the most specialized situation which can be instantiated by considering the state vector of the virtual agent at each frame

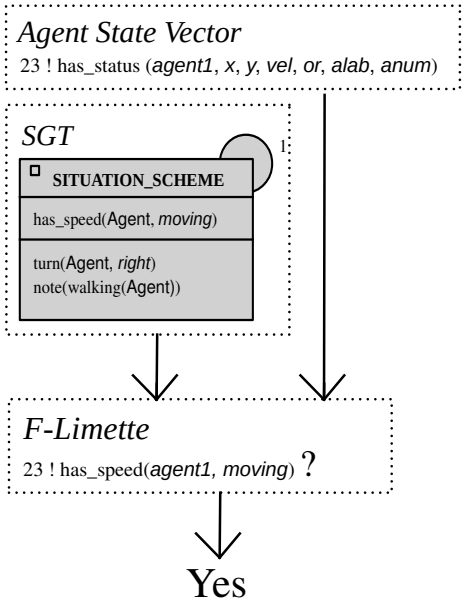


Figure 7: Instantiation of a situation scheme: the *FMTHL*-based engine called F-Limette evaluates the predicates of the SGT given the state vector.

step. This traversal is applied by considering the knowledge encoded in the form of prediction and specialization edges: on the one hand, given a situation, only its successors in temporal order will be evaluated at next time step. On the other hand, each situation can be described in a conceptually more detailed way, thus allowing abstraction and specificity. As a result, reaction predicates modify the state vector over time, see Fig. 7.

3.3 Automatic Path Planning

The reaction predicates determine particular movements for a synthetic agent. This is achieved by modifying its position, velocity and orientation, for example:

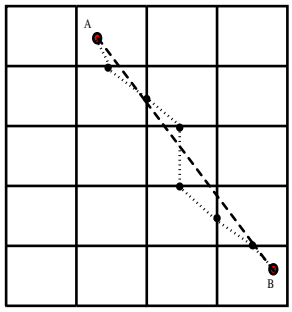


Figure 8: Automatic Path Planning: Once the list of segments is crossed by an straight line from A to B, the graph traversal generates the synthetic trajectory (dashed curve).

- *turn(Agent,* Value*)*: it causes a modification in the orientation of the agent for the next frame.
- *accelerate(Agent,* Value*)*: it modifies the velocity of the agent for the next time step.
- *wait(Agent, Duration)*: it causes the agent to remain stopped at the same position for a given quantity of time.
- *go_on_performing(Agent, Action)*: it makes the agent to continue performing the same action.

More complex reaction predicates take into account the semantics of the agent's position and the goal the agent is going to achieve , planned by the behavior. For example, the predicates *go_to_location* and *go_to_next_location* compute the minimum paths to go to a particular *location* segment. Since the scene model is divided into segments, a *path* is defined as the sequence of segments the agent should visit to reach the goal, see Fig. 8. In order to

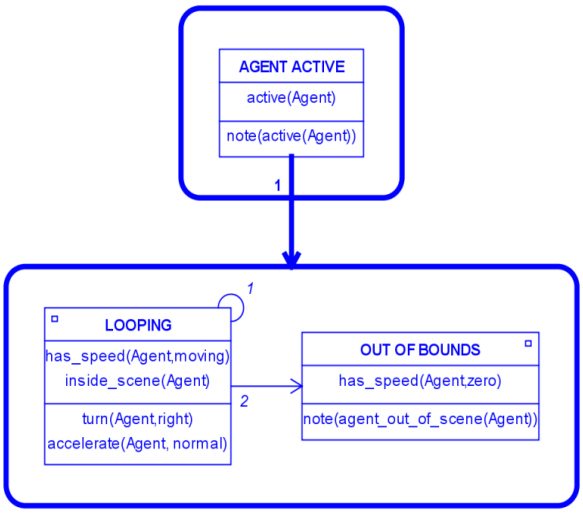


Figure 9: SGT for the *spiral* example.

generate smooth trajectories, transitions between two adjacent segments are computed by considering the middle point of their shared line. The curves of Fig. 8 represent the obtained trajectories.

So let consider the SGT depicted in Fig. 9. The traversal of the SGT will instantiate the *LOOPING* situation for a given frame step *t*. The reaction predicate *turn(Agent, right)* indicates that the agent will turn to the right for the next frame step *t+1*. This will change the status vector, see Fig. 10. Note that the qualitative data (*right*) of the predicate *turn* is converted to a numerical value in degrees (10). As a result, Fig. 11 shows the evolution of the agent trajectory performing a spiral motion. Fig. 12 shows the generated animation involving several agents.

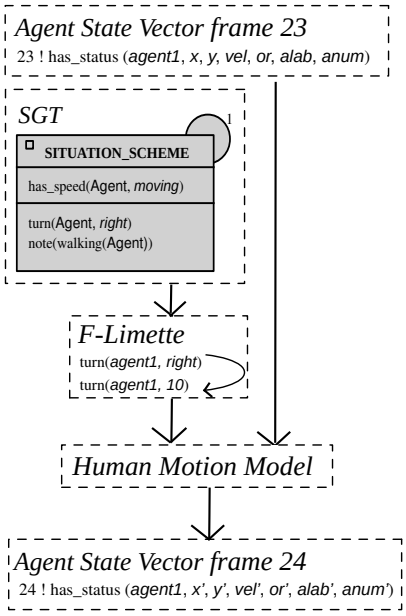


Figure 10: Scheme of the process since the reaction predicate is raised from the traversal until the new agent status is generated.

Frame Number	Agent State
...	
5	has_status(**agent_1**, 6.0, 6.09, 91.9, 2.73, aWalk, 0.1)
6	has_status(**agent_1**, 6.0, 6.09, 91.9, 2.73, aWalk, 0.1)
7	has_status(**agent_1**, 5.99, 6.2, 93.81, 3.45, aWalk, 0.2)
8	has_status(**agent_1**, 5.99, 6.27, 94.76, 3.81, aWalk, 0.25)
...	

Figure 11: Evolution of the state vector of the agent for the spiral trajectory..

4 Experimental Results

In this section we show the results obtained modeling the pedestrian behavior in a pedestrian crossing scenario, see Fig. 13. For this scenario, three different pedestrian behaviors have been considered:

1. *Behavior_1*: the pedestrian crosses the crosswalk without stopping in the waiting line.

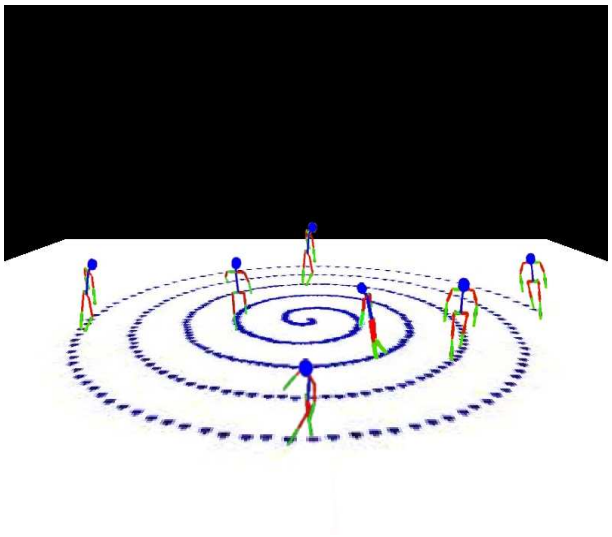


Figure 12: Animation of the trajectory generated with the *spiral* example.

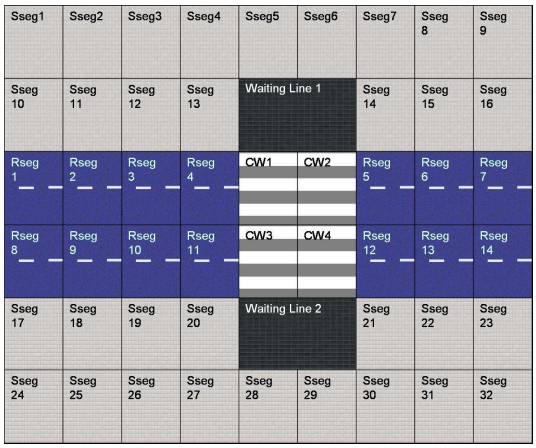


Figure 13: The *pedestrian crossing* scenario.

2. *Behavior_2*: the pedestrian crosses the crosswalk, stopping few seconds in the waiting line.

3. *Behavior_3*: the pedestrian crosses the road without approaching to the crosswalk.

Fig. 14 depicts the SGT designed to model *Behavior_1* behavior. Given an initial status, the most general situation is instantiated (*ED SIT 13*) and it is generalized depending on the agent's location. If it is in the road (*ED SIT 14*), the predicate *go_to_waiting_line* causes the agent to move to the nearest sideway segment. When located in a sideway segment, the agent walks to the nearest waiting line (*ED SIT 20*). Then, the situation *ED SIT 14* cannot be instantiated again and the posterior situations in sequence are inspected. Thus, the agent reaches the crosswalk (*ED SIT 16*) and crosses the road (*ED SIT 15*).

Fig. 15 shows an example of automatically generated video sequence, which involved four virtual agents. The agents were located at different positions and were given different behaviors, from the list mentioned above. Fig. 16 shows the results of a multitudinary simulation involving a hundred of virtual agents[1].

5 Conclusions and Future Work

This paper presented a complete framework to automatically generate synthetic image sequences by designing and simulating complex human behaviors in virtual environments. Given an initial state of a virtual agent, a simulation process generates posterior synthetic states by means of precomputed human motion and behavior models, taking into account the rela-

[1]More results can be found at the ISE Lab website: http://www.cvc.uab.es/ise

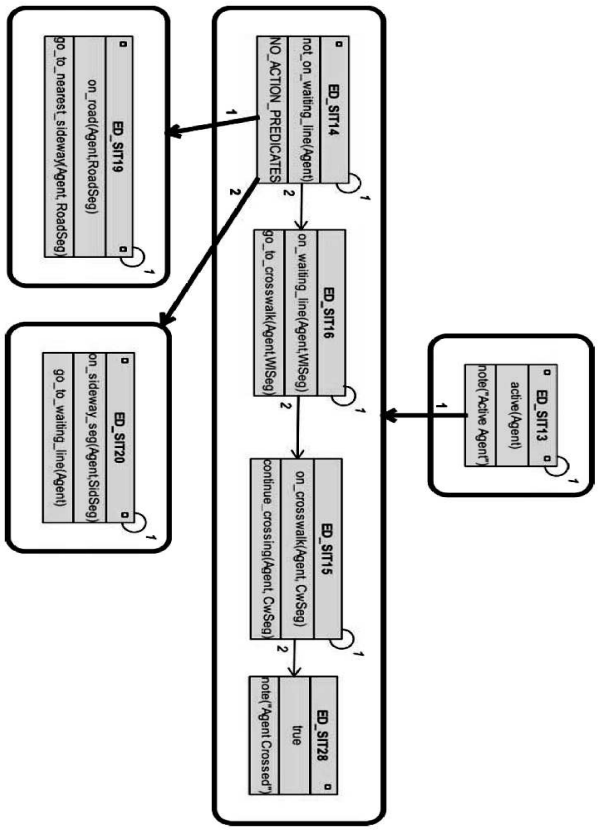
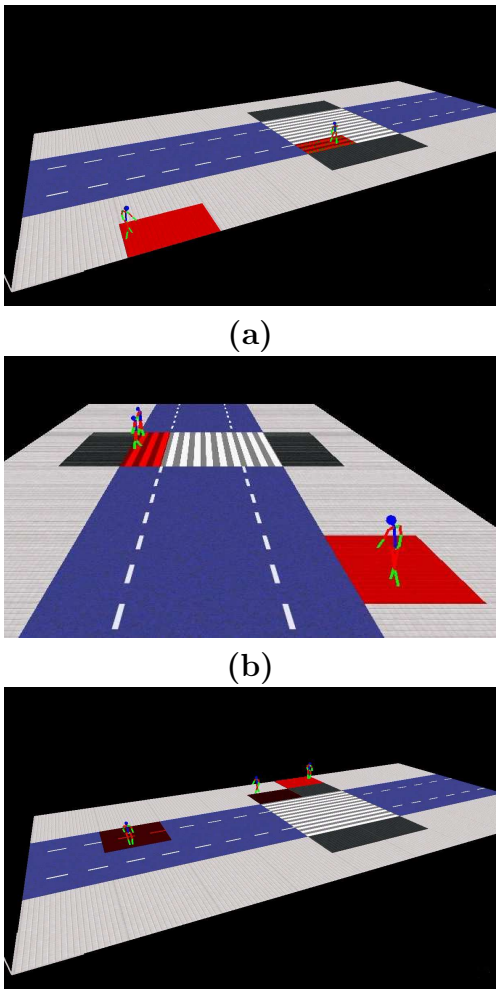


Figure 14: SGT for the *Behavior_1* model. Further explanation in the text.



(a)

(b)

(c)

Figure 15: Frames from an automatically generated image sequence. (a) A pedestrian crosses the crosswalk without stopping. (b) Two pedestrians cross the crosswalk at the same time. (c) A pedestrian crosses the road without approaching the crosswalk.

tionships of such an agent w.r.t its environment and w.r.t. other agents at each frame step. The resulting status sequence is further visualized into a virtual scene using a 3D graphic engine. Conceptual knowledge about human behavior patterns has been represented using the *Situation Graph Tree* formalism and a rule-based inference system called *F-Limette*.

The framework has been successfully tested in a pedestrian crossing scenario, and complex pedestrian behaviors have been simulated. Moreover, the SGT formalism is a highly suitable formalism which can be applied to different discourse domains. However, conceptual knowledge about the environment has to be provided to each

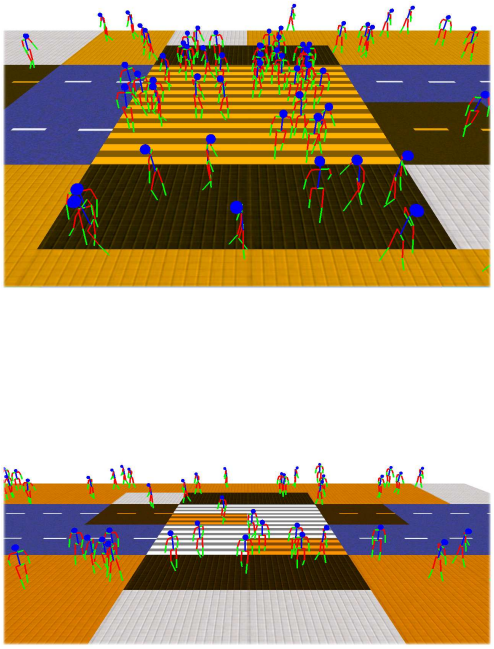


Figure 16: Example frames of a multitudinary crosswalk simulation, involving more than one hundred virtual agents at the same time.

new scenario. Although it is possible to model most of the possible human behaviors for a given scenario, there is a lack of flexibility to add more uncertainty to the simulation process, which is in the proper nature of human behavior. Future work will be focused to incorporate learning capabilities to the SGT formalism, in order to design more flexible and evolvable behavior models. In addition, more kinds of agents will be considered, e.g. vehicles and static objects, and their interaction with the environment will be also modeled. Finally, the 3D render engine will be provided with realistic body models.

Acknowledgments.

This work has been supported by EC grants IST-027110 for the HERMES project and IST-045547 for the VIDI-Video project, and by the Spanish MEC under projects TIN2006-14606 and DPI-2004-5414. Jordi Gonzàlez also acknowledges the support of a Juan de la Cierva Postdoctoral fellowship from the Spanish MEC.

References

[1] 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.

[2] T. A. Galyean B. M. Blumberg. Multi-level direction of autonomous creatures for real-time virtual environments. *Computer Graphics*, 29(Annual Conference Series):47–54, 1995.

[3] 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.

[4] T. Conde and D. Thalmann. Autonomous virtual agents learning a cognitive model and evolving. In *Intelligent Virtual Agents*, pages 88–98, 2005.

[5] N. Courty, F. Lamarche, S. Donikian, and E. Marchand. A cinematography system for virtual storytelling. In *Proceedings of the 2nd International Conference on Virtual Storytelling (ICVS 2003)*, volume 2897, pages 30–34, Toulouse, France, 2003.

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

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

[8] K. Perlin and A. Goldberg. Improv: A system for scripting interactive actors in virtual worlds. *Computer Graphics*, 30(Annual Conference Series):205–216, 1996.

[9] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[10] 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.

[11] W. Tambellini T. Conde and D. Thalmann. Behavioral animation of autonomous virtual agents helped by reinforcement. In *Lecture Notes in Computer Science*, volume 2792, pages 175–180, 2003.

[12] D. Shreiner M. Woo T. Davis, J. Neider. *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Addison-Wesley, 2005.

[13] W. Tang and T. Ruan Wan. Synthetic vision for road traffic simulation in a virtual environment. In *Intelligent Agents for Mobile and Virtual Media*, pages 176–185, London, UK, 2002. Springer-Verlag.

[14] N. Zagalo and V. Branco. Storytelling and interactivity in videogames, from myst to ico. *Deliverable 3.1.1 of INSCAPE Integrated Project (EU RTD IST-2004-004150)*, 2005.