REVIEW: Generation of augmented video sequences combining behavioral animation and multi–object tracking

Pau Baiget, Carles Fernández, Xavier Roca, Jordi Gonzàlez

Computer Vision Center (UAB)

email: pbaiget@cvc.uab.es

Abstract

In this paper we present a novel approach to on–line generate augmented video sequences involving interactions between virtual and real agents in real scenarios. On the one hand, real agent motion is estimated by means of a multi–object tracking algorithm, which determines real objects’ position over the scenario for each time step. On the other hand, virtual agents are provided with behavior models considering their interaction with the environment and with other agents. The resulting framework allows to generate video–sequences involving behavior–based virtual agents that react to real agent behavior and has applications in education, simulation, and in the game and
cinema industries. We show the performance of the proposed approach in an indoor and outdoor scenario simulating human and vehicle agents.

**Keywords:** augmented reality, behavioral animation, virtual human agents
Introduction

Augmented Reality (AR) is a growing area in virtual reality research [1, 2, 3]. AR is a combination of the real scene viewed by the user and a virtual scene generated by the computer that augments the original scene with additional information. Most augmented reality applications are primarily concerned with letting a user browse a 3D virtual world registered with the real world. However, the generation of augmented sequences through the animation of virtual characters in real worlds has raised as a promising challenge nowadays, with wide applications in the game industry [4] and education [5].

In this paper we present an efficient and general framework to on–line combine virtual (human and vehicle) agents with real agents in a real world image sequence. We combine a motion–based multi–object tracking algorithm [6] with behavioral autonomous agents in order to simulate interaction between real and virtual human agents. To enable such interaction, we provide the virtual agents with complex behaviors modeled using the Situation Graph Tree (SGT) formalism [7], where the knowledge is not defined in terms of quantitative data but using a conceptual methodology: described as a set of fuzzy spatio–temporal predicates in Fuzzy Metric Temporal Logic (FMTL) [8].

The capabilities of the SGT to represent and simulate human behavior have been demonstrated in previous works, where synthetic image sequences were automatically generated to describe stories (virtual storytelling) [9], and to be used as a tool for measuring the performance of tracking algorithms [10]. In the present work, agent interaction is extended
not only to other virtual agents but also to real agents. To this end, we propose a real–time framework to convert the real agent data into virtual agent data so that the actual virtual agents consider the real ones as if these were also virtual. Hence, virtual agents will react to real agents, taking their motion into account in order to select and perform their behavior.

In order to accomplish with these objectives, the three main tasks of this framework can be briefly summarized as: (a) modeling and animating of virtual agents and creating virtual environments (b) estimating real agent motion in real sequences; and (c) joining both virtual and real agents in an augmented reality image sequence.

Virtual agent behavior inside a virtual environment is defined using three different abstraction levels. Firstly, the Movement Level (ML) defines the physical activity of the virtual agent in short term, e.g. actions like walk, bend. Secondly, the Activity Level (AL) represents the final or intermediate goals that the agent should accomplish in order to satisfy the requirements imposed in the first place. Goals are represented in this paper by fuzzy–temporal predicates and are processed using the reasoning engine FMTL [8]. Finally, the Behavioral Level (BL) describes the capabilities of the agent to interact w.r.t. other agents, static objects and different locations in the scene, while trying to reach the goals specified by the Activity Level. The behavior is here represented using the Situation Graph Tree formalism [7].

The rest of the paper is structured as follows: Next we review previous works on augmented reality and behavior modeling. Then, the framework used to model virtual agents is described. The next section discusses about the generation of augmented sequences with
virtual agents, where behavior models are simulated in real environments to generate augmented sequences. Subsequently, we show the evaluation results of the proposed framework. Finally, last section concludes the paper and shows future lines of research.

Related Work

A complete description of the stages in creation of augmented reality involving virtual characters can be found in [11]. There, a list of problems related is stated, however most of them are related to computer graphic features, such as illumination or character realism, which are out of the scope in this work. On the other hand, interaction between real and virtual agents has been little considered previously [12, 13]. Balcisoy et al. [12] showed the application into a virtual storytelling system, restricting virtual agents to perform a given script. Gelenbe et al. [13] proposed an augmented reality system which combines computer vision with behavior–based object agents. Behavior is modeled using a hierarchy of three behavior modules, but without considering the particular features of human motion and behavior. Zhang et al. [14] presented a method to merge virtual objects into video sequences recorded with a freely moving camera. The method is consistent regarding illumination and shadows, but it does not tackle occlusions with real moving agents. The use of computer vision techniques in augmented reality has also been recently confronted by Douze et al. [15], where moving targets are tracked from image sequences and merged into other real or virtual environments. However, the method does not consider the animation of behavioral
virtual agents in the resulting sequence.

Regarding human behavior modeling, previous works have developed models 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 to be learnt. Thus, behavioral learning has been exploited using techniques such as k–Nearest Neighbours [16] or Reinforcement Learning (RL) [17, 18], 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 [19, 20, 21].

Since the real scenario on which the augmented sequence will be created conditions on agent behavior, this needs to be defined beforehand. Thus, in this work agent behavior is modeled based on top–down formalism called Situation Graph Tree (SGT).

**Modeling Virtual Agents**

To simulate virtual worlds we need to specify some requirements and restrictions about agent capabilities and properties of the virtual environment. Firstly, we define a human action model which determines the postures to be adopted by virtual agents in the environment. Next, we introduce the reasoning engine to manage facts and goals, i.e. conceptual information. Finally, we introduce a modeling tool used to structure semantic knowledge in
terms of behavior patterns.

**Movement Level**

In this level we define the physical actions that can be executed by a virtual agent. An agent $A$ exists inside a virtual environment during a period of time. This period of time is discretized into time steps, and the information of an agent at a given time step $t$ is defined by the *state vector* of the agent $s_t$:

$$s_t^A = (x, y, o, v, a, p)$$  \hspace{1cm} (1)

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

A physical action is defined as a discrete sequence of movements of the body parts. In this work we use a human model based on the stick figure, see Fig. 1. The sequence of movements is called *p–action* [22] and is represented by a parameter $p \in [0, 1]$ where $p = 0$ and $p = 1$ indicates the start and end of an action, respectively. Using p–actions, we can model both cyclic actions, e.g. *walk* or *run*, and non–cyclic ones, e.g. *wave* or *bend*.

In this work, a *trajectory* is defined as the sequence of state vectors for a particular virtual agent over time, during its existence within the virtual environment:

$$T^A = \{s_1^A, \ldots, s_n^A\}$$  \hspace{1cm} (2)
Activity Level

At this level we define the conceptual representation of the knowledge required to specify agent goals during its existence inside the virtual environment. The knowledge involved in the development of an agent is classified into two parts:

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

2. Objectives to be accomplished in short or middle term, e.g. going to a place, start a given action.

We represent this knowledge not using quantitative data but with temporal logic predicates. To this end, we use the Fuzzy Metric Temporal Logic (FMTL) formalism [8], which consists of a rule–based inference engine in which conventional logic formalisms are extended by a fuzzy and temporal component. In terms of notation, FMTL is similar to the well known reasoning engine PROLOG [23]. However, while the latter is based on resolution calculus and depth search, FMTL uses tableaux calculus, and provides several inference strategies (depth search, breadth search, beam search) which can be selected by the user. In addition, each FMTL predicate has a temporal validity expressed by a time interval $(t_1, t_2) \in \mathbb{Z}$. The following expression:

$$ t_1 : t_2 ! performing\_behavior(c_1, c_2, \ldots). $$

(3)
indicates that the predicate `performing_behavior` is valid for the time interval \((t_1, t_2)\). A recent application of FMTL can be found in [24].

Information related to agents is stored for each time step \(t\), by using two abstraction levels: On the one hand, we encapsulate the quantitative information of an agent, represented by the state vectors, into predicates in FMTL:

\[
t ! \text{has_status}(\text{Agent}, x, y, o, v, a, p).
\]

where \(t\) refers to the current time step, \(\text{Agent}\) is the agent identifier and the rest of values are inherited from \(s_t\).

On the other hand, we define predicates that allow to infer conceptual fuzzy values from particular quantitative values of the state vector. For example, the predicate `has_speed(\text{Agent}, Value)` holds for every particular time step but with the field `Value` is particularized with a fuzzy value assigned to the quantity \(v\) of the state vector, see Fig. 3. Other examples of predicates inferred from the current state vector are:

- `is_performing(\text{Agent}, \text{Action})` tells whether \(\text{Agent}\) is performing the action \(\text{Action}\), e.g. `walking`.

- `is_near_to(\text{Agent}, \text{Agent2})` tells whether \(\text{Agent}\) is near to \(\text{Agent2}\).

In addition, the definition of objectives is performed through predicates that require an adaptation of the agent trajectory to be valid in future time steps. For instance, given an agent with state vector \(s_t^A = (x_t, y_t, v_t, o_t, a_t, p_t)\), the predicate `go_to_location(\text{A}, \text{Location})`
computes the shortest trajectory \( \{s^A_{t+1}, \ldots, s^A_n\} \) to go to \emph{Location} and infers the next position \((x_{t+1}, y_{t+1})\) of such a trajectory, according to the current speed value \(v_t\).

**Behavior Level**

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

- Complex behaviors, defined as a sequence of actions, need to be specified in a higher level of abstraction.

- The existence of other agents in the virtual environment can affect the agent trajectory at each time step, and therefore some higher level mechanism must be provided to manage transition between goals.

We consider two types of interaction: (i) interaction between agents, in which every virtual agent knows about the existence of other agents and adapts its behavior w.r.t. other behaviors; and (ii) interaction with the environment, in which virtual agent acts depending not only on its numerical position but also on the meaning of the position. Therefore, the virtual environment is divided into semantic regions.

The conceptual knowledge about agent behavior is encoded in a set of rules in FMTL and organized within a behavior modeling formalism called \emph{Situation Graph Tree} (SGT). The basic component of SGTs is the \emph{situation scheme}, which embeds the qualitative knowledge for a given agent at each frame step, see Fig. 4. These situation schemes are separated into
two parts: the state scheme and the action scheme. On the one hand, the state scheme refers to FMTL predicates about the state of the agent, which should 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.

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 temporal successors are specified for each situation (prediction edges in Fig. 7). On the other hand, each situation can be described in a conceptually more detailed way, yielding a temporal sequence of more specific situations (specialization edge in Fig. 7), which can be seen as subbehaviors. See [7] for further details on the theoretical foundations of the SGT.

SGTs are used to recognize those situations which can be instantiated for a virtual agent by applying the so-called graph traversal [25]. 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 step. If the state scheme is satisfied for the current situation, that situation is instantiated, see Fig. 5. In this case, the action predicates are executed and modify the state vector for the next time step, see Fig. 6.

The action predicates determine particular movements and actions to be performed by a virtual agent. This is achieved by modifying its position, velocity, orientation, and action, for example:
• turn(Agent, Value) modifies the orientation of the agent for the next frame.

• accelerate(Agent, Value) modifies the velocity of the agent for the next time step.

• change_to_performing(Agent, Action): it makes the agent change from its current action to Action.

• follow(Agent, Agent2) makes agent Agent move towards the location of agent Agent2.

• wave_to(Agent, Agent2) makes agent Agent wave at agent Agent2.

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. The predicates go_to_location and go_to_next_location compute the minimum paths to go to a particular location segment.

Let us now consider the SGT depicted in Fig. 8.(a), which models the toy example walking in spiral behavior. Placing one virtual agent at a particular starting position, the most general situation will be instantiated. Moreover, the only possible specialization, LOOP-ING, will be instantiated since the agent is within the scenario. At time step t, the action predicates turn and accelerate modify the orientation and speed of the agent, thus generating a new state vector for time t + 1, which will be used as input in the next loop of the traversal, see Fig. 6. The traversal will finish when the agent falls out of the scenario boundary. Fig. 8.(b) shows the generated animation involving several agents.
Given the result of the traversal, a synthetic image sequence \( I'' \) is rendered by a 3D rendering engine, containing the silhouette of the virtual agent over the virtual environment.

**Augmented Sequences with Virtual Agents**

In previous sections we have shown how interactions between virtual agents can be managed using SGTs. Next objective is then to introduce virtual agents into a real environment in order to generate augmented image sequences. Thus, virtual agents will have to adapt their behavior depending on the behaviors of real agents.

Hence, the most straight way to adapt real agents into the described framework is to treat them as if they were virtual agents. This process implies the acquisition of real agent data and the conversion of that information into sequences of state vectors. Consequently, the process will be transparent to the rest of virtual agents.

**Converting real agents into virtual agents**

In the computer vision area, *tracking* is defined as establishing coherent relations among targets between frames of an image sequence; or as inferring the target state over time using all evidence up to date. Tracking is performed through a *scene*, which is the piece of the real world that a particular visual sensor (camera) can capture [6].

In this work we acquire real agent information by means of a real–time tracking algorithm based on efficient background substraction [26, 6]. Such a system extracts informa-
tion of moving targets in the image plane for each time step, consisting of the center of the
bounding ellipse and the estimated silhouette.

Let \( S^r \) be a real scenario and let \( C^r \) be a calibrated static camera\(^1\). In order to have a
consistent correspondence between real and virtual perspective view, a virtual scenario \( S^v \)
is modeled by using the real scene dimensions obtained with the calibration of \( C^r \), see Fig.
9. Finally, a virtual camera \( C^v \) is located in \( S^v \) at the same location of \( C^r \) in \( S^r \).

An image sequence \( I^r \) is recorded using \( C^r \) and processed by the tracker. At time step
\( t \), the tracker outputs a list of targets’ position coordinates over the image plane. We obtain
the ground plane representation of the agents’ positions \( O^r_t \) by applying a 2D homography
using the DLT algorithm [27], thus obtaining:

\[
O^r_t = \{\pi^1_t, \ldots, \pi^k_t\} \tag{4}
\]

where \( \pi^i_t = \{x^i_t, y^i_t\} \) contains the 2D ground–plane position for the target \( i \) at time step \( t \).

Next, position coordinates are converted into state vectors, i.e. \( \text{has\_status} \) predicates,
and incorporated to the list of state vectors for the current time step. Thus, the information
obtained by the tracker appears as if it was generated by the traversal of a SGT and will be
considered as a virtual agent by the actual ones.

\[
\pi^i_t \Rightarrow t ! \text{ has\_status}(\text{agent}_i, x^i_t, y^i_t, o^i_t, v^i_t, a^i_t, p^i_t) \tag{5}
\]

where the values \( o^i_t, v^i_t \) are computed using the previous agent location \((x^i_{t-1}, y^i_{t-1})\). The
action \( a^i_t \) is valued either \text{standing}, \text{walking}, or \text{running}, depending on \( v^i_t \), and the pose

\(^1\)The superindexes \( r \) and \( v \) denote \textit{real} and \textit{virtual}, respectively.
parameter $p_i^t$ is incremented frame by frame, looping in the range $[0, 1]$.

Finally, at time step $t + 1$, the graph traversal will consider the data from both real and virtual agents at time $t$. The procedure explained above is depicted in Fig. 10.

**Alternative Information for Behavior Analysis**

As explained above, our vision framework is essentially based on a multi–object tracking system that generates the state vectors of real agents for each time step. Nonetheless, in the case of human agents the framework can be extended with other computer vision approaches that would allow to generate additional information about human agent motion. Next, a set of possible extensions is listed:

- **Body Action Recognition** [28, 29]. Information about body action enriches knowledge about what are the complex actions performed by the agents, such as $are\_fighting(Agent1, Agent2)$ or $is\_waving(Agent)$.

- **Facial Expression Recognition** [30, 31]. Facial expression analysis provides a good estimation of the mood of the agent. Such an information can ease explanations of global agent behavior, e.g. the predicates $is\_angry(Agent)$, $is\_sad(Agent)$.

- **Head Pose Estimation** [32]. The estimation of head pose approximates the gaze of the agent at a given frame step, allowing the instantiation of high level predicates such as $is\_looking\_at(Agent, Location)$. 

15
Nevertheless, the above listed approaches constitute currently a hot topic in computer vision research. Therefore, a real–time solution for them is still missing depending on the scenario conditions. Furthermore, these approaches deal with some well known drawbacks in computer vision systems, such as illumination problems, reflections, or camouflage [6].

Moreover, some other inputs outside the computer vision research field may be used, such as touch and temperature sensors, although in this work we will restrict to show the performance using information from vision systems, as explained in the Experimental Results section.

To conclude, the sources of information listed above generate FMTL predicates, easing the incorporation of that information as conditions in the situation schemes of a SGT. This integration capability points out the extendibility of the proposed approach.

**Composing the augmented sequence**

In order to provide the resulting image sequence with consistence and realism inside the 3D environment, occlusions between virtual and real agents must be taken into account. Given a time step $t$, let $A_t = \{a_1, \ldots, a_n\}$ be the list of agents that are moving inside the scenario and let $P_t = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the list of their positions over the ground plane. The agents are sorted in terms of the distances between their current position and the position of the camera. Fig. 11 shows two agents $a_1, a_2$ at distances $d_1$ and $d_2$ from the camera, respectively. Since $d_2 > d_1$, the agents are sorted as $a_2, a_1$. Finally, their silhouettes
are drawn over the image background and hence agents occlude each other in a consistent manner.

**Experimental Results**

In this section we show the results obtained by simulating virtual agents in indoor and outdoor environments. First, the augmented sequence composition has been tested using the *Circle* sequence, where a real agent and a virtual agent walk in circles one in front of the other. The experiment demonstrates the capacity of SGTs to model virtual agent behavior that reacts to real agent position over the scenario. The virtual agent walks in circles maintaining a constant distance with the real agent.

Fig. 12.(a) shows the complete trajectories of both agents over the ground plane and the camera position. For this experiment, the behavior *walking in circle* is similar to the *spiral* example previously shown. Figs.12.(b-d) show example frames of the resulting image sequence. While (b) and (c) show good results, in (d) it can be seen how tracking errors due to camouflage between the arm and the floor affect the composition of the augmented image.

The second experiment has been designed in order to test the reliability of our approach to deal not only with tracking information, i.e. real agent positions, but also with body action recognition. To this end, a sequence has been recorded in a real urban scenario, involving a pedestrian crossing intersection, see Fig. 13.(a). For this scenario, four semantic labels have been considered, namely *road, crosswalk, sidewalk,* and *waiting line* (the area separating
sidewalk and crosswalk). These labels have been then assigned to different parts of the scenario and finally rendered into a virtual environment by using different textures, see Fig. 13. In the sequence, called Police sequence, a policeman is managing the traffic between vehicles and pedestrians in the pedestrian crossing. The policeman position is tracked over time and its action is recognized using the Motion History Images (MHI) approach [28]. As a result, the recognized actions are converted into FMTL predicates, see Fig. 14:

- **policeman_orders_stop(Policeman)** indicates that the policeman is giving right to pass to pedestrians, and vehicles must stop.

- **policeman_orders_pass(Policeman)** indicates that the policeman is giving right to pass to vehicles, and therefore pedestrians must stop.

The purpose of this experiment is to obtain augmented sequences involving virtual pedestrian and vehicle agents, whose behavior will be affected by the actions performed by the real policeman. To this end, two SGTs have been designed to model pedestrian and vehicle behavior, respectively:

- **Crossing the Crosswalk (CC)**, depicted in Fig. 15: A virtual human agent is located on the scenario. It first reaches a sidewalk (if previously in another region) and then moves to the crosswalk. If the policeman is giving grant to pass to vehicles, the agent stops at the waiting line and waits until the policeman grants the pass to pedestrians. After crossing the crosswalk, the agent leaves the scene.
• Driving on the Road (DR), depicted in Fig. 16: A virtual vehicle agent appears from one of the road limits. If the policeman is granting pass to pedestrians, the vehicle stops before crossing the crosswalk. When the policeman changes the action, the vehicle continues driving towards the end of the scene.

Subsequently, we have specialized previous behaviors in order to refine interaction between virtual agents: When being in a crosswalk, the virtual agent checks if there is a vehicle crossing at the same time. If so, the virtual agent raises the arm as a complaint (predicate complain_to(Agent,Agent2) where Agent2 is the vehicle in the crosswalk). Fig. 18.(a) shows a sample frame of the real sequence $I_r$. The result of tracking is shown in Fig. 18.(b). A virtual agent has been added to the sequence and it arrives to the crosswalk while one real vehicle is crossing. The agent detects a danger of runover situation and then behaves as explained above, see Fig. 18.(c). Since the system is working in ground–plane coordinates, we have independence from the point of view, so a multi–camera system can reproduce the same result, as shown in Fig. 18.(d).

Conclusions and Future Work

In this paper we have presented a novel approach to generate augmented video sequences involving real and virtual agents. Virtual agents are capable of developing complex behaviors and reacting to real agent behavior in real environments. On the one hand, real agent motion is estimated in real–time by means of a multi–object tracking algorithm. On the
other hand, virtual agents are provided with behavior models considering their interaction with the environment and with other real or virtual agents. We have tested our approach in indoor and outdoor scenarios by simulating different behaviors and interactions between real and virtual agents.

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

Future work will focus on the different topics discussed in this paper. First, we need to provide our framework with more realism in terms of illumination and shadows, as stated in [11]. Also, silhouette estimation in tracking algorithms remains as an open problem when dealing with long occlusions or camouflage, and research towards this end will improve the composition of the augmented sequences. Finally, the extension explained in the previous
paragraph will be implemented in order to provide the real agents with a feedback of virtual agents behavior.

References


Figure 1: (a) Generic human body model represented using a stick figure similar to [33], here composed of twelve limbs and fifteen joints. (b) Different human models used performing dancing and running actions.
Figure 2: \textit{p--actions} computed in the \textit{aRun aSpace}, see [34] 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.
Figure 3: Discretization of continuous speed values into a set of intervals. The graph shows the fuzzy membership functions $\mu_{\text{speed value}}$ for the subset (zero, small, normal, high, very high) of discrete conceptual speed values.

Figure 4: The situation scheme is the basic component of the SGTs. More details in the text.
Figure 5: Instantiation of a situation scheme: the FMTL-based engine called F-Limette evaluates the predicates of the SGT given the state vector.

Figure 6: Scheme of the process since the reaction predicate is raised from the traversal until the new state vector is generated.
Figure 7: Example of SGT representing a pedestrian behavior model. Thin arrows represent the temporal evolution of situations and the numbers close to them represent the priority order of arrows with the same source. Thick arrows represent the specialization of one situation into a set of more specific situations.
Figure 8: Spiral example to describe the SGT traversal. (a) SGT model of the *walking in spiral* behavior. (b) Several virtual agents walking in spiral while performing different actions.

Figure 9: Overall scheme of the virtual representation of a real scenario. The virtual scenario is modeled using the measurements from the camera calibration.
Figure 10: Generation of an augmented reality image sequence $I^{ru}$. More details in the text.
Figure 11: Ordering agents by the distance between their position and the camera. More details in the text.
Figure 12: Results of processing the Circle sequence. (a) (b–d) Example frames of the obtained augmented sequence. More details in text.
Figure 13: Creation of the virtual environment. a) Real scenario $S^r$ b) Homography applied to the image to obtain a ground–plane image c) Top view of the resulting virtual scenario $S^v$.

Figure 14: The actions performed by the policeman in the Police sequence allow to instantiate two predicates, giving the right to pass to either pedestrians or vehicles.
Figure 15: Pedestrian behavior for the Police experiment. The predicates \textit{police\_orders\_stop} and \textit{police\_orders\_pass} are the police action recognized for each time step.
Figure 16: Vehicle behavior for thePolice experiment. The predicates `police_orders_stop` and `police_orders_pass` are the police action recognized for each time step.
Figure 17: Result frames of the augmented *Police* sequence.
Figure 18: Result frames of the steps shown in Fig. 10. (a) Original frame from $I^r$ (b) Graphical output of the tracking algorithm (c) Augmented frame with one virtual agent (d) Same frame from another point of view. More details in the text.
Figure 19: Scheme of the extension of our approach in order to provide a feedback to the real agents. More details in the text.