Learnable Behavioural Model for Autonomous Virtual Agents: Low-Level Learning

Toni Conde
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Virtual Reality Lab (VRLab)
CH-1015 Lausanne, Switzerland
toni.conde@epfl.ch

Daniel Thalmann
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Virtual Reality Lab (VRLab)
CH-1015 Lausanne, Switzerland
daniel.thalmann@epfl.ch

ABSTRACT
In this paper, we propose a new integration approach for simulation and behaviour in the learning context that is able to coherently manage the shared virtual environment for the simulation of autonomous virtual agents. Our low-level learning technique has proved fast, simple and robust. It is also able to automatically learn behavioural models for difficult tasks. Thus, we believe it will be more useful to the computer graphics community than a technique based on the classical Q-learning approach. The results are illustrated in two case studies that require effective coordination of behaviours.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning; I.3.6 [**Computer Graphics**]: Methodology and Techniques – *Interaction techniques*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism - *Animation*.

General Terms
Algorithms, Human Factors, Performance.

Keywords
Computer animation, character animation, reinforcement learning, behavioural modelling, AI-based animation, machine learning.

1. INTRODUCTION
Autonomous Virtual Agents (AVAs) play an important role in computer graphics. Animating them is one of the fundamental challenges.

In this paper we are presenting research in the domain of behavioural animation using a low-level learning technique. This is accomplished with a behavioural model defining how an AVA reacts to stimuli from its environment.

Human-like behaviour is achieved through the use of a complex "cognitive map" and the application of a hierarchy of behavioural strategies [1]. The overall cognitive mapping process involves acquisition, coding, storage, recall and decoding of environmental information [2]. In fact, an individual "cognitive map" will often contain numerous inaccuracies or distortions [3]. Many of these anomalies result from the fact that humans use a predominantly visual perception system and are not capable of processing everything they see due to the tremendous amount of incoming information. Other errors result from the way the information is processed and stored within the "cognitive map" structure itself [4].

Previously, [5] have proposed new methodologies to carry out the mapping of all information coming from virtual sensors of vision, audition and touch as well as from the Virtual Environment (VE) in the form of a "cognitive map". The approach enables the partial re-mapping of cognitive and semantic information at a behavioural level. For example, when spatial attention is primed with tactile stimulation, the location of the attention spotlight is only partially remapped in visual coordinates. This framework will generate the multi-sensory information necessary for our *behavioural learning*.

In summary, this paper presents an original contribution with a novel approach allowing an AVA to independently learn a behavioural model.

This paper is organized as follows: Section 2 – State of the Art; Section 3 – Methodology; Section 4 - Integration; Section 5 – Experimental Results with two Case Studies; Section 6 – Discussion.

2. STATE OF THE ART
A great deal of research has been performed on the control of animated autonomous characters by [6], [7], [8] and [9]. The various techniques have produced impressive results, but they are limited. They have no learning ability and are therefore limited to explicit pre-specified behaviours.

On-line *behavioural learning* has begun to be explored in computer graphics [10] and [11]. A notable example is in [12], where a virtual dog can be interactively taught by the user to exhibit a desired behaviour. This technique is based on *Reinforcement Learning* (RL) and it has been shown to work well in [13]. Also, since all these learning techniques are designed to be used on-line, they have, for the sake of interactive speed, a limited learning capacity.

In RL, the machine learning of an input-output mapping is performed through continued interaction with an environment in order to maximize a scalar index of performance. This performance index is called a fitness function. Some of the earliest research in computer graphics involving RL sought to make AVAs automatically learn how to walk, swim, or jump [14] and [15]. However, whilst interesting and useful, this type of learning does not provide AVAs with decision-making abilities. On-line *behavioural* (reactionary) *learning* in computer graphics is a new field of exploration.

The goal of RL is to automatically learn an optimal policy. Optimal implies that the policy always maps the current state to the best possible action according to the fitness function. Since it is impractical to store a large Q-factor table explicitly, it must be approximated. We have performed several experiments using *Q-learning* for AVA learning, where the only information available to an AVA was a fitness function. Previously, [13] have demonstrated the difficulty of this approach. Firstly, to obtain stable results, the Q-factor table must be approximated to a high degree of accuracy. Secondly, *Q-learning* is harder to perform if there are no terminal states. Also, as discussed in [16], *Q-learning* can be challenging to successfully use, especially since it requires the animator to visit every state-action many times.

The technique presented in this paper is built on the success of traditional behavioural modelling with the goal of alleviating two of its major weaknesses: performance and time-consuming construction. In our opinion, the difficulties associated with *Q-learning* make it an undesirable approach for our application. For this reason, we have developed an alternative technique for AVA low-level learning. The objective is to allow an AVA to explore its unknown VE and to build structures, in the form of cognitive models or maps, based on this exploration. Once its representation has been constructed, the AVA could then easily communicate its knowledge to, for example, other naive AVAs.

3. METHODOLOGY

In this section we introduce our low-level learning methodology. This means that an AVA can automatically learn a behavioural model. We have developed a new technique to perform AVA learning, using a tree search with *Inverse Reinforcement Learning* (IRL).

3.1 Planning-based Reinforcement Learning (RL) with Subagents

Given the shortcomings of *Q-learning* discussed above, we have developed an alternative approach to AVA low-level learning by computing discrete examples of a policy.

A planning-based RL technique will only learn a sub-optimal policy, the quality of which depends on the search depth limit used [17] and [18]. To ensure that on the whole an AVA's behaviour is optimal, we utilize the *Q-decomposition* approach proposed by [19], where only the sum of the subagent *Q-values* for a particular state determines the optimal action. This requires each subagent to indicate, from its perspective, a value for every action. Subagent *j* reports its action values $Q_j(s, a)$ for the current state *s* to the *arbitror*. Then, the *arbitror* chooses an action maximizing the sum of the *Q-values* (see Figure 1). In our proposed methodology we use *pseudo values* – instead of *Q-values* – for vision (Pv_{vision}), avoidance ($Pv_{avoidance}$) and navigation ($Pv_{navigation}$) to trade-off different features of virtual sensors.

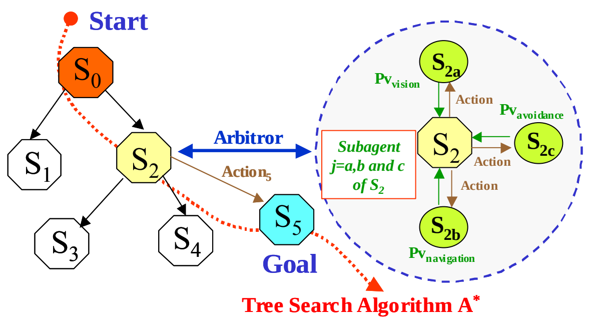


Figure 1: (left) Planning is performed with a tree search algorithm A* of all proposed actions throughout the time limit. (right) The *arbitror* chooses an action maximizing the sum of the *pseudo values* ($Action_5$).

3.2 Apprenticeship Learning via Inverse Reinforcement Learning (IRL)

It may be impossible to use straightforward RL since an AVA may only have an approximate idea of the reward function which, when optimised, should generate a "desirable" behaviour. We believe that the difficulty of manually specifying a reward function represents a barrier to the broader applicability of RL and the optimal control of the algorithms.

As the entire field of RL is founded on the supposition that the reward function, rather than the policy or the value function, constitutes the most succinct, robust and transferable definition of the task, it seems natural to consider an apprenticeship learning approach via IRL [20] to learn the reward function.

In the learning process, one source of information consists of observing the behaviour of other "expert" AVAs as in *imitation* and *apprenticeship learning*. In this context, it is commonly assumed that the purpose of observation is to learn a policy which is represented by the mapping of states to actions.

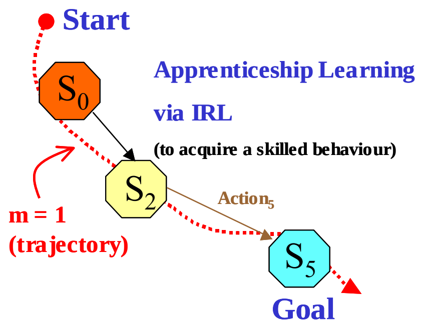


Figure 2: *Apprenticeship learning* via IRL is performed after *m* trajectories generated by the expert.

In our methodology, IRL is used in *apprenticeship learning* to acquire a skilled behaviour and to ascertain the optimisation of a reward function through a natural system. Yet, in using IRL, it is difficult to extract an observed reward function [21]. This is a fundamental challenge in theoretical biology, econometrics and other fields.

Using this method, an AVA can automatically learn a behavioural model. For the animator, this alleviates the workload of programming an explicit behavioural model (see Figure 2).

4. INTEGRATION

The integration of our low-level learning technique combined with different machine-learning techniques – but with several greatly improved – will be more useful to the computer graphics community than techniques based on a purely machine-learning approach (see Figure 3).

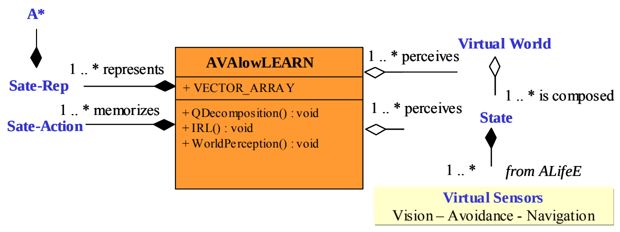


Figure 3: Comprehensive UML design of AVA low-level Learning (AVAlowLEARN). (left) State and action space. (right) Multi-sensory information necessary for our *behavioural learning*.

Firstly, we used a tree search algorithm A* [22] to observe the trajectories (state sequences) generated by the user (expert) and which take actions according to π_E (π_{expert}).

Secondly, we integrated the same *Q-decomposition* approach as [19], which uses all *pseudo value* function components (vision, avoidance and navigation) to choose the actions. *Q-decomposition* allows for multiple value functions, with each of these residing in a subagent, but it still requires each subagent to report its value estimates to an *arbitror* and to receive in turn the action chosen by the *arbitror*.

Thirdly, we took the *apprenticeship learning* via IRL with the projection algorithm proposed by [20] and adapted it to our behavioural animation.

A policy π is a mapping from states to probability distributions over actions. Given that the reward R is expressible as a linear combination of the features ϕ, the expectation values of the features for a given policy π completely determine the expected sum of discounted rewards for acting according to that policy (see Figure 4).

Given a MDP\R (denotes a Markov Decision Process without a reward function), a feature mapping ϕ and the expert's feature expectations μ_E, the problem is to find a policy whose performance is close to that of the expert's, on the unknown reward function $R^* = \omega^* . \phi$.

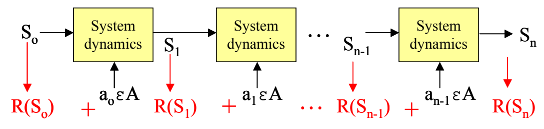


Figure 4: Reinforcement Learning (RL) description with a finite-state Markov Decision Process (MDP).

The *apprenticeship learning* projection algorithm proposed by [20] for finding a policy is as follows (pseudo-code):

```
1. Randomly we pick some policy $\pi^{(0)}$, $\mu^{(0)} = \mu(\pi^{(0)})$, and set i = 1.
   $\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t . \phi(s_t) \mid \pi] \varepsilon \Re^k$
   ($\mu(\pi)$ is the expected discounted accumulated features value vector)
2. Set $\bar{\mu}^{(i-1)} =$
   $\bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})} (\mu^{(i-1)} - \bar{\mu}^{(i-2)})$
   (This computes the orthogonal projection of $\mu_E$ onto the line through $\mu^{(i-2)}$ and $\mu^{(i-1)}$).
   Set $\omega^{(i)} = \mu_E - \bar{\mu}^{(i-1)}$
   Set $t^{(i)} = \|\mu_E - \bar{\mu}^{(i-1)}\|_2$
3. If $t^{(i)} \leq \varepsilon$, then terminate.
4. Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (\omega^{(i)})^T \phi$.
5. Estimate $\mu^{(i)} = \mu(\pi^{(i)})$.
6. Set i = i + 1, and go back to step 2.
Upon termination, the algorithm returns $\{\pi^{(i)}: i = 0 \ldots n\}$.
```

The performance of this algorithm only depends on (approximately) matching the feature expectations, not on recovering the true underlying reward function.

5. EXPERIMENTAL RESULTS WITH TWO CASE STUDIES

To test whether our low-level learning model was qualitatively able to learn the demonstrated AVA's behaviour, we created two environments requiring effective coordination of behaviours: a virtual apartment and a more complex situation with a car driving simulation inside a virtual city.

5.1 Case Study No. 1: an AVA learning to avoid obstacles and to navigate inside a virtual apartment

We chose to simulate an AVA in an apartment where he can "live" autonomously by perceiving his VE and generating a few behaviours at specific locations.

We can test our *behavioural model* applied to an AVA by implementing the model in C++ and Python [23] thanks to the Python module of the real time framework [24] for advanced virtual human simulations.

As depicted in Figure 5, an interface has been designed to monitor and change, if necessary, the evolution of the AVA's states and actions in real-time inside the virtual apartment. The path planning module [25] is applied for obstacle avoidance when the AVA walks to a specific location (see Figure 6). It is also utilized to the decisions the AVA makes. Its 3D viewer shows, in real-time, what the AVA decides to do at each moment and can play keyframes for the learned actions using the walk engine [26].

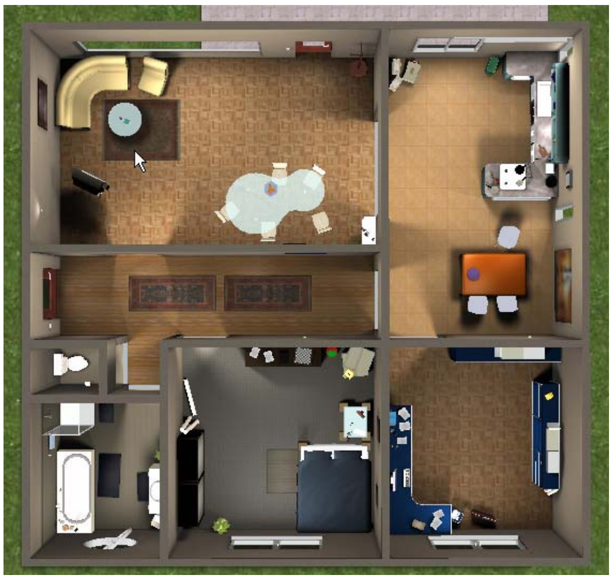


Figure 5: Top view of the virtual apartment.

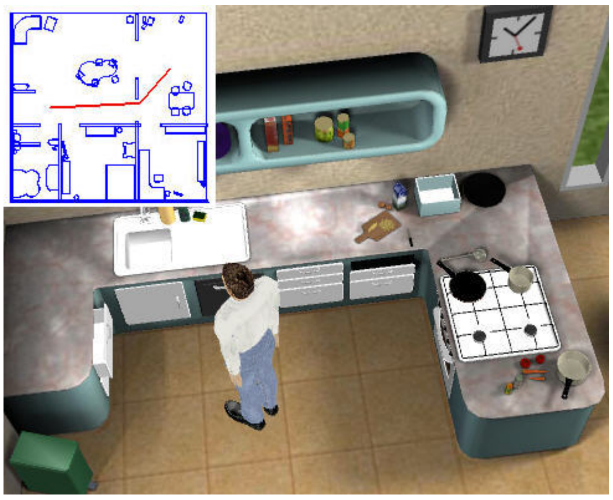


Figure 6: (upper left corner) Top view of the virtual apartment with path planning after low-level learning process. The AVA is now inside the kitchen.

Figure 7 shows the inputs and outputs of the behavioural model for the navigation inside a virtual apartment. The input for Vision (r, φ) is the spherical coordinates of the AVA's head. Our Artificial Life Environment (ALifeE) [5] gives multi-sensory information to the AVA's sensors of vision, audition and touch (see Figure 3).

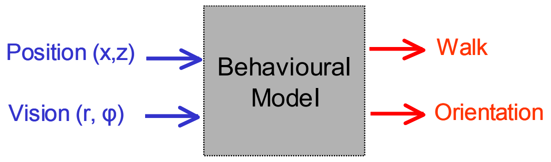
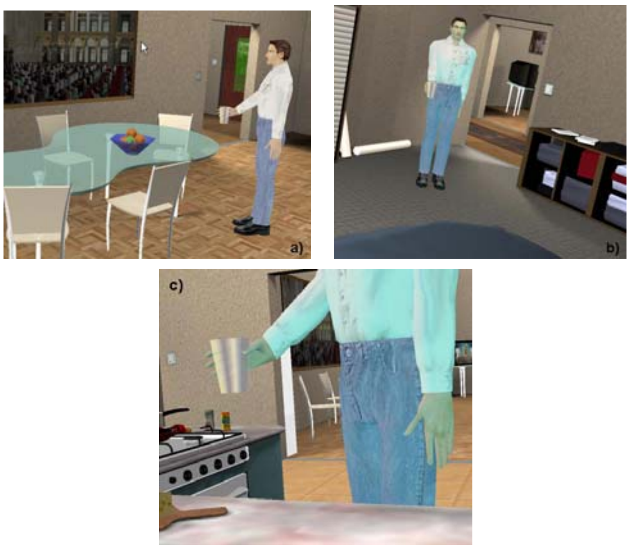


Figure 7: Behavioural model for the navigation inside a virtual apartment with inputs and outputs. The MDP has two different actions: walk and orientation.

In the first scenario (see Figures 8 a), b) and c)), an AVA learns the behaviour of taking a *CD box* from the kitchen table. Then, the AVA walks around inside the different rooms whilst taking care to observe and avoid virtual objects and also navigates with a path planning between doors. Finally, the AVA puts the *CD box* on the desk in the office.

Figures 8 a), b) and c): Snapshots of the AVA learning (through seeing, avoiding and navigating) to take a *CD box* from the kitchen table and to put in on the office desk.

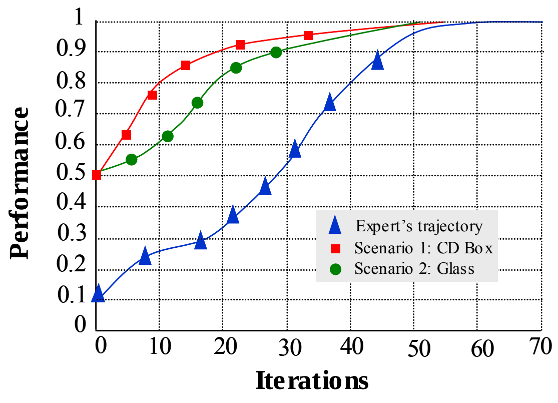
In the second scenario (see Figures 9 a), b) and c)), an AVA learns to take a *glass* from the living room table or from the bedroom shelf and to bring it to the kitchen.



Figures 9 a), b) and c): Snapshots of the AVA learning (through seeing, avoiding and navigating) to take a *glass* from the living room table and to bring it to the kitchen.

The two scenarios were reported in Table 1 to illustrate the performance of our low-level learning methodology. Using our methodology, only a few sample trajectories were needed to attain a performance mimicking that of the expert. Thus, by learning a compact representation of the reward function, our methodology outperformed.

Table 1. Plot of performances obtained with our low-level learning methodology. All animations were rendered in real time using OpenGL on a 3.0 GHz PC with an nVIDIA GeForce FX Go5350 video card.



5.2 Case Study No. 2: an AVA learning to drive a car inside a virtual city

We implemented our approach to learn the behavioural model applied to a car driving simulation inside a virtual city. An AVA is the driver and has dual control over the acceleration and the wheel of the car (see Figure 10). The controls are real-value (e.g., the action space is continuous) and the car can move to any location or take any orientation.

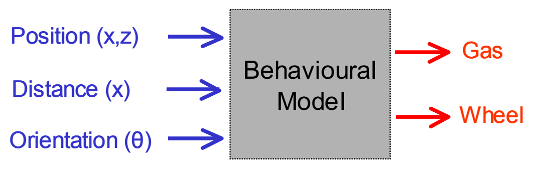


Figure 10: Behavioural model for the car driving simulation inside a virtual city with inputs and outputs. The MDP has two different actions: gas and wheel.

Figure 10 shows the inputs and outputs of the behavioural model for the car driving simulation inside a virtual city. The input for Distance (x) is the distance between the driver and the traffic signal.

The continuous action space was quantified to achieve real-time performance. The simulation ran at 15 [frames/s] and the expert's features were estimated from a single trajectory of 2'700 samples, corresponding to 3 minutes of driving time.

Our approach communicates the semantic information required for the "right way" behaviour of our low-level learning method in two ways:

- adding semantics to dynamic objects, and
- adding tools to sound, pedestrians and semantic objects (e.g., road signals).

Adding Semantics to Dynamic Objects

The appropriate description of tools for autonomous cars and pedestrians is available inside our simulation application. We use scripts to configure the VE application and to determine the behaviour of the autonomous cars in each different situation. The behaviour of the autonomous cars is controlled with Python scripts [23]. To simulate the life of a virtual city in a realistic manner, the VE also integrates some semantic notions about specific areas.

Adding Tools to Sound, Pedestrians and Semantic Objects

We have defined three types of plug-ins:

1. The plug-in for sound permits the definition of the properties of the sound environment in 3Ds max® (3D render software). To achieve a realistic illusion, it is important to generate a 3D spatialization in sound. The geometry of each object is transformed into a list of points outlining the sound obstacle. We define four types of sound primitives: environment, obstacle, listener and sound source. Inside the VE, we can have several sound sources with the following essential parameters: position, orientation (e.g., the direction of propagation) and ray configuration of the two cones of propagation. A sound source can be attenuated, absorbed or reflected depending on when the sound strikes an obstacle. The

obstacles are characterized by a two dimensional contour.

2. The plug-in for pedestrians covers a crowd composed of two types of people: those who walk in the virtual city with a precise goal and those who stroll.
3. The plug-in for semantic objects, inspired by the smart objects concept [27], gives significance to the road signals and traffic lights so that the sensory module of vision [5] can recognize them.

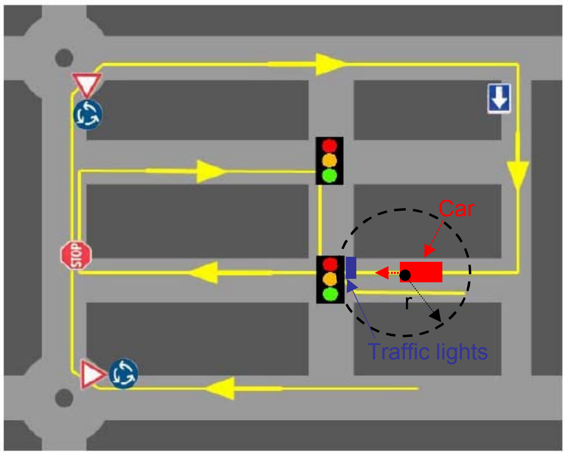


Figure 11: AVA learning to drive a car inside a virtual city with visual pseudo-perception.

Using our plug-ins along with 3Ds max®, designers have the information necessary to construct all types of VEs. We are now able to create the semantic information for our simulation but it will have a static character. However, we wish to obtain dynamic semantic information and we will use pseudo-perception (see Figure 11) or virtual perception [5] to achieve this goal.

In our car driving simulation, ϕ is the vector of the features indicating the different driving desiderata that need to be accounted for, such as a collision with another car, a pedestrian or a virtual object, or driving on the sidewalk, for example. The unknown vector ω^* specifies the relative weight of these various desiderata.

Blindly following the expert's trajectory would not work, because the pattern of traffic encountered is different each time.

We applied *apprenticeship learning* to try to learn behaviours to avoid pedestrians and to take traffic lights into account (e.g., stop at a red light, decelerate at an orange light and keep driving at a green light).

The final result was a good driving behaviour, since the driver could plan far enough ahead to manoeuvre the car adequately around the virtual city, avoiding pedestrians and reacting correctly to traffic lights (see Figures 12 a) and b)).

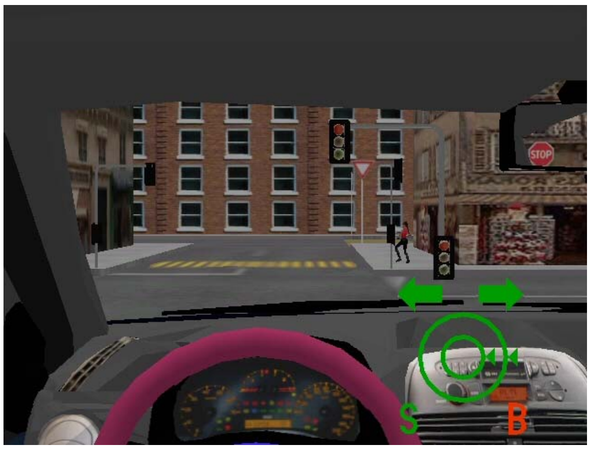


Figure 12 a): The car "sees" the traffic lights and the pedestrian inside a circular zone. The panel informs the driver that he/she must brake (B – in red at bottom right corner).

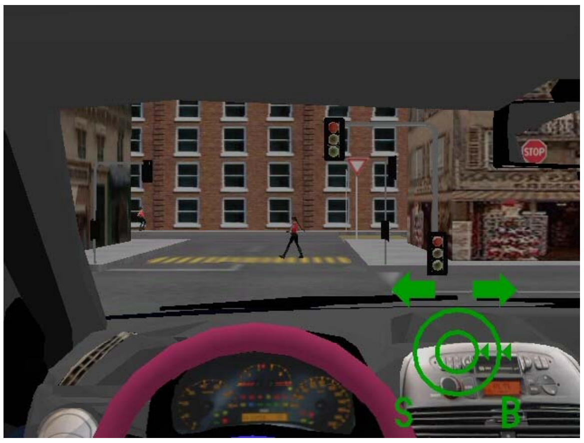


Figure 12 b): A pedestrian wishes to cross the road according to the rules defined.

We achieved our best results by performing the low-level learning method for 40 iterations and a policy was selected by inspection (see Table 2).

Table 2. Results obtained with our low-level learning methodology and one driving style (as estimated from 3 minutes of driver demonstration). All animations were rendered in real time using OpenGL on a 3.0 GHz PC with an nVIDIA GeForce FX Go5350 video card.

	Gas	Wheel
μ_e	0.060	0.290
$\mu(\pi)$	0.059	0.266
ω	0.107	0.059

Our method was qualitatively able to mimic the demonstrated driving style. Table 2 shows the feature expectations of the expert (driver) μ_e, the feature expectations learned $\mu(\pi)$ and the weights ω used to generate the policy.

Although our technique performed well in our two case studies, there is no guarantee that it will work for every imaginable AVA and simulated VE.

6. DISCUSSION

Our low-level learning technique has proved to be fast, simple and robust. It has also succeeded in automatically learning behavioural models for difficult tasks. Thus, we believe it will be more useful to the computer graphics community than a technique based on the classical *Q-learning* approach. Unlike explicit *Q-learning*, our technique is not guaranteed to find an optimal policy. But *Q-learning* is usually impractical because of a large Q-factor table

Our AVA low-level learning technique will only learn a sub-optimal policy, whose quality depends on the selected search depth limit. However, optimality is probably not necessary to achieve the appearance of intelligent behaviour in an AVA. Indeed, good behaviour usually looks just as, if not more, realistic than perfect behaviour

With our method, an AVA can automatically learn a behavioural model. For the animator, this relieves the workload of designing an explicit model. Moreover, it allows to model tasks for which it would be difficult or virtually impossible, to develop an explicit model.

Our work needs to be distinguished from that of [28]. In fact, when they performed off-line AVA learning, it could be difficult to design a fitness function that resulted exactly in the desired behavioural model. Our contribution in this paper is important because we present a solution for this previously unsolved problem and also we perform on-line AVA low-level learning.

In the robotics field, various methods like *teaching by imitation*, *imitation learning* or *learning by observation* are effective. However, since they are not automatic, they cannot be used for on-line adaptation. The technique presented by [29] for teaching a robot simple navigational behaviours is too limited for AVA behavioural learning. By comparison, our mimicking method is less general, but is fully automatic, learns quickly and can easily encapsulate very complex behaviours. One of its most original aspects is indeed to detect automatically novel behaviours.

However, there are some weaknesses in our approach. When performing AVA low-level learning, it can be difficult to find a policy that exactly corresponds to the desired behavioural model. Also, the reward function should be expressible as a linear combination of known features.

The methodology behind automatic learning behaviour is difficult but interesting. This could take interactive computer graphics, especially in the entertainment market, to a completely new level. It could also be interesting if an animator could interactively train an AVA for *behavioural learning*, rather than using a reward value.

The approach presented here is part of a more complex model that is the object of our research. The goal is to realize an Artificial Life environment for an AVA including different interfaces and sensorial modalities coupled with various evolving learning methodologies.

7. ACKNOWLEDGMENTS

This research has been partially funded by the Swiss National Science Foundation.

8. REFERENCES

[1] P. Larkin. Achieving human style navigation for synthetic characters. A survey. In *Proceedings of Neural Networks and Computational Intelligence*, pages 30-37, 2003.

[2] R. Downs and D. Stea. Cognitive maps and spatial behavior. *Image and Environment*. R. Downs and D. Stea, Eds., pages 8-26, Chicago: Adline Publishing, 1973.

[3] D. Griffin. Topographical Orientation. *Image and Environment*. R. Downs and D. Stea, Eds., pages 296-299, Chicago: Adline Publishing, 1973.

[4] N. Kamwisher and P. Downing. Separating the wheat from the chaff. *Science*, vol. 282, pages 57-58, 1998.

[5] T. Conde and D. Thalmann. An Artificial Life Environment for Autonomous Virtual Agents with multi-sensorial and multi-perceptive features. *Computer Animation and Virtual Worlds*, **15**(3-4), pages 311-318, John Wiley, 2004.

[6] C. Reynolds. Flocks, herds, and schools: A distributed behavioural model. In *Proceedings of ACM SIGGRAPH*, pages 25-34, 1987.

[7] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behaviour. In *Proceedings of ACM SIGGRAPH*, pages 43-50, 1994.

[8] B. Blumberg and T. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of ACM SIGGRAPH*, pages 47-54, 1996.

[9] K. Perlin and A. Golberg. A improv: a system for scripting interactive actors in virtual worlds. In *Proceedings of ACM SIGGRAPH*, pages 205-216, 1996.

[10] R. Burke, D. Isla, M. Downie, Y. Ivanov and B. Blumberg. Creature smarts: The art and architecture of a virtual brain. In *Proceedings of the Computer Game Developers Conference*, pages 147-166, 2001.

[11] B. Tomlinson and B. Blumberg. Alphawolf: Social learning, emotion and development in autonomous virtual agents. In *Proceedings of First GSFC/JPL Workshop on Radical Agent Concepts*, pages 35-45, 2002.

[12] B. Blumberg, M. Downie, Y. Ivanov, M. Berlin, M. Johnson and B. Tomlinson. Integrated learning for interactive synthetic characters. In *Proceedings of ACM SIGGRAPH*, pages 417-426, 2002.

[13] T. Conde, W. Tambellini and D. Thalmann. Behavioral Animation of Autonomous Virtual Agents helped by Reinforcement Learning. *Lecture Notes in Computer Science*, vol. 272, pages 175-180, Springer-Verlag: Berlin, 2003.

[14] K. Sims. Evolving virtual creatures. In *Proceedings of ACM SIGGRAPH*, pages 15-22, 1994.

[15] R. Grzeszczuk and D. Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of ACM SIGGRAPH*, pages 63-70, 1995.

[16] R. S. Sutton and G. Barto. *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[17] T. Mitchell. *Machine Learning,* McGraw Hill, 1997.

[18] R. Pfeiffer and C. Scheier. *Understanding Intelligence*, MIT Press, 1999.

[19] S. Russell and A. L. Zimdars. Q-Decomposition for Reinforcement Learning Agents. In *Proceedings of Conference on Machine Learning ICML*, pages 656-663, 2003.

[20] P. Abbeel and A.Y. Ng. Apprenticeship Learning via Inverse Reinforcement Learning. In *Proceedings of Conference on Machine Learning ICML*, vol. 69, ACM Press, New York, 2004.

[21] A. Y. Ng and S. Russell. Algorithms for Inverse Reinforcement Learning. In *Proceedings of Conference on Machine Learning ICML*, pages 663-670, 2000.

[22] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, Prentice-Hall, 1995.

[23] M. Lutz. *Programming Python*, O'Reilly Ed., Sebastopol, CA, 1996.

[24] M. Ponder, G. Papagiannakis, T. Molet, N. Magnenat-Thalmann and D. Thalmann. VHD++ Development Framework: Towards Extendible, Component Based VR/AR Simulation Engine Featuring Advanced Virtual Character Technologies. *Computer Graphics International (CGI)*, pages 96-104, 2003.

[25] M. Kallmann, H. Bieri and D. Thalmann. Fully Dynamic Constrained Delaunay Triangulations. *Geometric Modelling for Scientific Visualization*, **4**:74-123, Heidelberg, Germany, 2003.

[26] R. Boulic, N. Magnenat-Thalmann and D. Thalmann. A Glogal Human Walking Model with Real-Time kinematics Personification. *Visual Computer*, **6**:344-358, 1990.

[27] L. M. G. Gonçalvez, M. Kallmann and D. Thalmann. Defining Behaviors for Autonomous Agents based on Local Perception and Smart Objects. *Computer & Graphics,* **26**(6):887-897, 2002.

[28] J. Dinerstein, P.K. Egbert and H. de Garis. Fast and learnable behavioural and cognitive modelling for virtual character animation. *Computer Animation and Virtual Worlds*, **15**:95-108, 2004.

[29] M. Kasper, G. Fricke, K. Steuernagi and E. von Puttkamer. A behavior-based mobile robot architecture for learning from demonstration. *Robotics and Autonomous Systems*, vol. 34, pages 153-164, 2001.