COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2013; **24**:477–496
Published online 18 June 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1524

RESEARCH ARTICLE

CHAMELEON: online learning for believable behaviors based on humans imitation in computer games

F. Tence[2], L. Gaubert[1], J. Soler[1,2], P. De Loor[1] and C. Buche[1]*

[1] UEB/ENIB/LAB-STICC, CERV, 25 rue Claude Chappe, 29280 Plouzané, France
[2] VIRTUALYS, CERV, 25 rue Claude Chappe, 29280 Plouzané, France

ABSTRACT

In some video games, humans and computer programs can play together, each one controlling a virtual humanoid. These computer programs usually aim at replacing missing human players; however, they partially miss their goal, as they can be easily spotted by players as being artificial. Our objective is to find a method to create programs whose behaviors cannot be told apart from players when observed playing the game. We call this kind of behavior a *believable behavior*. To achieve this goal, we choose models using Markov chains to generate the behaviors by imitation. Such models use probability distributions to find which decision to choose depending on the perceptions of the virtual humanoid. Then, actions are chosen depending on the perceptions and the decision. We propose a new model, called CHAMELEON, to enhance expressiveness and the associated imitation learning algorithm. We first organize the sensors and motors by semantic refinement and add a focus mechanism in order to improve the believability. Then, we integrate an algorithm to learn the topology of the environment that tries to best represent the use of the environment by the players. Finally, we propose an algorithm to learn parameters of the decision model. Copyright © 2013 John Wiley & Sons, Ltd.

KEYWORDS
believable agent; decision making; video games
Supporting information may be found in the online version of this article.

***Correspondence**
C. Buche, UEB/ENIB/LAB-STICC, CERV, 25 rue Claude Chappe, 29280 Plouzané, France.
E-mail: buche@enib.fr

1. INTRODUCTION

To make a human user feel like he or she is inside the *simulation*, two criteria have been defined in academic research: immersion and presence. According to Slater *et al.*, *immersion* is an objective criterion that depends on the hardware and software [1]. It includes criteria based on virtual sensory information's types, variety, richness, direction, and extent to which they override real ones. For example, force feedback and motion sensing controllers, and surround sound and high-dynamic range rendering can improve the immersion. *Presence*, also known as telepresence [2], is a more subjective criterion. It is defined as the psychological sense of "being there" in the environment.

As stated in [1], *presence* partly depends on the match between sensory data and internal representation. This match expresses the fact that we try to use world models to better understand what we perceive and to be able to anticipate [3]. This idea is close to what is called *believability* in the arts. Indeed, we can believe in fictional objects, places, characters, and story only if they mostly fit in our models.

As there are many ways to enhance believability, we choose to focus on believable virtual characters, also known as *believable agents*. Characters often play a major role in the believability of book and movie stories. However, unlike book and movie characters, agents in simulation should be able to cope with a wide range of possible situations without anyone telling them what to do. Instead of defining manually these behaviors, we propose that our entities will be *able to learn*. This learning will be unsupervised and online: the entity will learn while it acts. This will both remove the burden of parameterizing the models controlling the characters and increase believability by having real-time evolution of the behavior.

To be able to achieve the best believability, we want the agents to be like human-controlled virtual characters. Indeed, there are no better example of what a believable behavior is than a human behavior itself. It is this kind of learning, by example [4] or *by imitation* [5,6], we want to use to model believable and autonomous characters.

The way the characters act and learn depends heavily on the kind of virtual environment they are in. We share issues with the *video games* industry because the game designers want the players to be immersed in the simulation too. They are trying to be as close as possible to reality, making rich and complex environments. Researchers can avoid some technical difficulties (rendering, physics, networking, etc.) by using such games. They can then focus on the making of the entities they want to study [7,8]. Furthermore, video games being made for human beings offer a real challenge for the entities to be believable. In addition, video games offer experts of these environments, human players, give pertinent criticisms on the entities' behaviors [9].

This paper proposes a decision-making model for believable agents by using real-time learning by imitation in video games. In this paper, we first define what believable agents are, give an overview of the kind of models that drive entities' behaviors, and focus on models based on an input–output hidden Markov approach (Section 2). Then, we propose a new model, Chameleon, in order to make the virtual character more believable and able of learning by imitation almost all the parameters of the model (Section 3). Results in a video game are shown in Section 4. To conclude, we give explanations about how we want to evaluate the believability of our model (Section 5).

2. BELIEVABLE BEHAVIORS IN VIDEO GAMES

2.1. Believability

2.1.1. Definition of Believability.

The notion of believability is highly complex and subjective. To define and understand this concept, we must look at its meaning in the arts where it is a factor of *suspension of disbelief* [10]. According to Thomas and Johnston, two core animators of Disney, believable characters' goal is to provide the "illusion of life" [11]. Reidl's definition is more precise: "Character believability refers to the numerous elements that allow a character to achieve the 'illusion of life', including but not limited to personality, emotion, intentionality, and physiology and physiological movement" [12, page 2]. Loyall tries to be more objective by saying that such a character "provides a convincing portrayal of the personality they [the spectators] expect or come to expect" [13, page 1]. This definition is quite close to one factor of the presence, the match between players' world model and sensory data.

If we want to apply the believability definition for video games, things become even more complex. Unlike classic arts, players can be embodied in a game by the mean of virtual bodies and can interact. The believability question now is, does a believable character have to give the illusion of life or have to give the illusion that it is controlled by a player? [14]. There can be very important differences as even if the video game depicts the real world; all is virtual, and players know that their acts have no real consequence.

In this research work, we consider only *believable* as *giving the illusion of being controlled by a player*. Now that we have defined *believability*, we then have to find how to improve it and measure the improvement.

2.1.2. Believability Criteria.

As believability is a broad concept, we need to find more precise criteria to break this concept down. According to the literature, we listed criteria that were reported to have an impact on believability. The requirements for believability are listed in Table 1.

First, the agent must react to the environment and the other players in a coherent way. This reaction must simulate a reaction time similar to a human reaction time. The agent must also avoid repetitiveness, both in the actions and in the behavior. It is also necessary that the intention of the agent can be easily understood by the human players. Contrary to what is done in most video games, the perception abilities of the agent must be similar to those of a player. The agent should be able to handle the flow of time, remembering information from the past and thinking ahead, making plans. Finally, the agent has to be able to evolve, changing its behavior for a more efficient and believable one. This evolution must be fast enough for the players to notice it, making them feel they play against an evolved being.

Table 1. List of the requirements for a character to be believable.

Believability requirement	Summary of the requirement	References
[B1: Reaction]	React to the players and changes in the environment	[14,15]
[B2: Reaction time]	Simulate a human-like reaction time	[14,16]
[B3: Variability]	Have some variability in the actions	[13, page 18; 14; 16]
[B4: Unpredictability]	Surprise the players with unpredictable behavior	[17,18]
[B5: Understandable]	Have an understandable behavior	[18,19]
[B6: Perception]	Have human-like perception	[20; 21, page 34]
[B7: Planning]	Plan actions in the future to avoid mistakes	[14]
[B8: Memory]	Memorize information	[13, page 22]
[B9: Evolution]	Evolve to avoid repeating mistakes	[5,22]
[B10: Fast evolution]	Evolve fast enough for the players to see it	

Table 2. Requirements for the models to make agents express believable behaviors.

Model requirements	Summary of the requirement
[M1: Variability]	Model variations in the actions and behaviors [B3: Variability] and [B4: Unpredictability]
[M2: Learning]	Is compatible with learning algorithms [B9: Evolution]
[M3: White box]	Can be modified and parametrized manually [18] to make the agent overdo [B5: Understandable]
[M4: Exaggeration]	Generate exaggerated behaviors so that players can easily understand the agent's objectives [B5: Understandable]
[M5: Planning]	Elaborate plans to avoid doing easily avoidable mistakes [B7: Planning]
[M6: Reaction time]	Simulate reaction time [B2: Reaction time]
[M7: Memory]	Model memory [B8: Memory]

Table 3. Summary of the characteristics of models for the control of believable agents.

	Connectionist	State transition	Production	Probabilistic
[M1: Variability]	✗	✗	✗	✓
[M2: Learning]	✓	✗	∼	✓
[M3: White box]	✗	✓	✓	✓
[M4: Exaggeration]	✗	✓	∼	✓
[M5: Planning]	✗	✗	✓	∼
[M6: Reaction time]	✓	∼	✓	✓
[M7: Memory]	∼	∼	✓	✓

2.2. Behavior Models for Believable Agents

2.2.1. Models Criteria.

As there are very few evaluations of the believability of behavior models, we will try to find the models that fulfill most of the requirements for believability. Because of the context of this study, we will only look at models for embodied agents in the following study. Those models can handle interactions with virtual environments and avatars. Therefore, they all fulfill the requirement [B1: Reaction]. According to the criteria we listed in Section 2.1.2, we list requirements for the model itself in the Table 2.

2.2.2. Behavior Model in Literature.

In order to fulfill these requirements, we studied the existing behavior models developed in both the research and the industry. We grouped behavior models into four types: connectionist models, state transition systems, production systems, and probabilistic models. Connectionist models are very good at learning but usually have problems in handling memory and planning. State transition systems can be easily understood and modified but may not be very well adapted to learning and planning. Production systems are quite good for learning, planning, and memory but make the agent act in a predictable manner. Finally, probabilistic models are good at variability, learning, and memory but may show problems with planning (Table 3).

As one of the requirements is that the model is able to evolve, we had to find adapted learning algorithms. In order to achieve behavior believability, the best method we found is imitation learning: the agent learns its behavior by using observations of one or several players. According to our definition of believability, it is the best way for the agent to look like players. Indeed, the goal of imitation learning is to make the agents act as human players. This learning method is also much faster than trial and error.

2.3. Model Choice

With previously presented studies in mind, we found out that probabilistic models based on an input–output hidden Markov model (IOHMM)[†] answer most of the requirements. In this kind of model, a hidden state is chosen according to inputs and the previous hidden state, and outputs are chosen according to inputs and the current hidden state. In most models, hidden states are decisions, inputs are stimuli, and outputs are actions. Several learning algorithms have been developed. The best combination is a Laplace's rule for the learning of the action distributions

[†]An IOHMM is a model used to learn to map input sequences to output sequences (unlike standard hidden Markov models, which learn the output sequence distribution).

Table 4. List of the noticed limitations with Le Hy's model.

Problems	Summary of the problem
[P1: Navigation]	The agent has problem navigating in environments.
[P2: Sensors]	The sensors are not well organized.
[P3: Expressiveness]	The FEC does not provide enough expressiveness for the agent to be believable.
[P4: Scaling]	The IP have problem handling many sensors.
[P5: Readability]	The IP is not easy to read for novices.
[P6: Learning]	The learning algorithm uses strong hypothesis, which may hinder its capabilities.

FEC, fusion by enhanced coherence; IP, inverse programming.

and an expectation–maximization (EM)[‡] algorithm for the learning of the Markovian distributions. An example is presented in the study by Le Hy *et al.* [23]. This particular work will be used to evaluate our proposal.

2.4. Evaluation and Limits of Le Hy's Model

Now that we have described the details of both the model and the learning algorithm, we will then try to assess the believability of the behavior generated by the model. The only way to evaluate the model is to implement it and to observe the behavior it produces. In this section, we point out the weakness of the model in terms of behavior and in the implication on the implementation.

As the architecture of the model reminds of a finite-state machine, it is easy to understand and to adjust the parameters. The model is modular, allowing the programmer to add or remove sensors, decisions, and actions without modifying the code too much.

However, the behaviors produced by the model in the game can easily be spotted as artificial by casual and regular players. In [24], we point out several problems of such models (Table 4).

[P1: Navigation]. The paths the agent uses to go from one point of the environment to another do not look like the ones a player would take. This problem does not comes from the model itself but from the representation it uses for the environment. Indeed, the agent uses navigation points placed by the designers of the environment that may not represent well how players prefer to use the environment.

[P2: Sensors]. Even if it is easy to add sensors to the model, increasing the number of perceptions makes the model more and more complex. We found that sensors were useful for the choice of the decisions but not for the choice of actions and vice versa. This is confirmed by the fact Le Hy uses in his implementation of different values for the random variables S_i depending if they are used for the choice of decision [25, pages 60–70, in French] or actions [25, pages 36–54, in French]. This should be clearly defined in the model.

[P3: Expressiveness]. The fusion by enhanced coherence (FEC) does not provide enough expressiveness: each

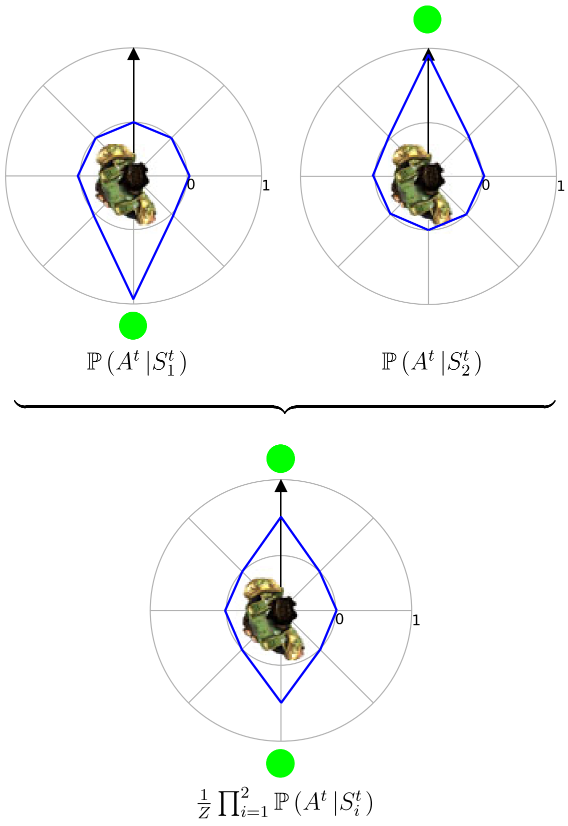


Figure 1. An illustration of the problem with FEC: (top) each distribution (blue line) gives a believable direction to go for a single attractor (green dot); and (bottom) the FEC does not give a believable action because the agent may constantly switch between the two attractors, oscillating constantly. FEC, fusion by enhanced coherence.

[‡]An EM algorithm is an iterative method to compute the maximum likelihood estimate of hidden data. The EM iteration alternates between performing an expectation (E) step and a maximization (M) step. In the E step, the missing data are estimated given the observed data and current estimate of the model parameters. In the M step, the likelihood function is maximized using the previous estimates (E step) of the missing data. These parameter estimates are then used to determine the distribution of the variables in the next E step.

Table 5. List of the noticed limitations with IOHMM models.

Problems	Summary of the problem
[P1: Navigation]	The agent has problem navigating in environments.
[P2: Sensors]	The sensors are not well organized.
[P3: Expressiveness]	The IOHMM models do not provide enough expressiveness for the agent to be believable.
[P4: Scaling]	The IOHMM models have problem handling many sensors.
[P5: Readability]	The inverse programming used in IOHMM models is not easy to read for novices.
[P6: Learning]	The learning algorithm uses strong hypothesis, which may hinder its capabilities.

IOHMM, input–output hidden Markov model.

sensory datum is considered to have the same importance. For example, if there are two attractors, one straight ahead and the other behind, the agent may constantly switch between the two (Figure 1). A real player will choose to focus on the attractor ahead, supposing the attractors are of the same strength.

[P4: Scaling]. For the inverse programming (IP) (where $P(D^t|D^{t-1}S^t)$ is computed using $P\left(S_i^t|D^t\right)$) to be valid, all the sensors S_i^t must be conditionally independent given the decision D^t. This hypothesis is very strong but can work for few sensors. However, the more the sensors, the higher the chances the hypothesis is wrong, which can make the model produce unbelievable behaviors.

[P5: Readability]. The IP technique makes the parameters not very understandable [M3: White box]. It is not natural to ask "What should I see if I take this decision?" which corresponds to the parameter $\mathbb{P}\left(S_i^t\,|D^t\right)$. A much more natural way is to express it as $\mathbb{P}\left(D^t\,|S_i^t\right)$, which corresponds to "What should I decide to do if I see that?"

[P6: Learning]. In order to make the behavior more complex and believable, one must add sensors, actions, and decisions. However, the number of parameters rapidly becomes intractable for a programmer to specify them manually. The learning algorithms Le Hy developed are very simple, and the parameters could be learned much more precisely given the observations of a teacher's avatar [B9: Evolution]. Many hypotheses are used to simplify the computation that may be harmful for the learning. Also, all the distributions may be learned with one algorithm instead of splitting the learning in two different algorithms: the interfacing between the algorithms would not be needed any more, and the convergence could be proven more easily.

Le Hy's approach allows an easy implementation and a great flexibility. Sensors and actions can be added without rethinking all the architecture, by adding only a class and few parameters. The model is generic and can be adapted to different virtual environments.

The behaviors generated by the model are, however, too simple to sustain the illusion of believability very long. This is due to the inner mechanisms of the model, the FEC and the IP. The parameters of the model could also be easily read for non-programmers to be able to modify the behavior. The model could be refined to express more complex behaviors without adding too many parameters. Finally, the learning could be done by one algorithm, making the learning more efficient.

The idea of CHAMELEON is to solve parts of problems.

In [24], we point out several problems of such models (Table 5).

3. CHAMELEON

This section describes four enhancements of classical IOHMM models to achieve more believable behaviors. In Section 3.1, we split the sensors into two types representing two granularities of information: high-level stimuli and low-level stimuli to clarify [P2: Sensors]. We also define two kinds of actions, each one associated to a kind of stimuli. Then, we propose an attention selection mechanism, where the agent selects one high-level stimulus and one low-level stimulus that answer to the problems [P4: Scaling] and [P5: Readability]. This mechanism makes the model more flexible and allows the model to express more complex behaviors, solving [P3: Expressiveness]. In Section 3.2, we propose to use a modification of the growing neural gas (GNG)[§] algorithm to learn by imitation information the layout of the environment, giving an answer to [P1: Navigation]. Finally, in Section 3.3, we propose a revamped imitation learning algorithm to learn almost all the model parameters, resolving the problem [P6: Learning].

[§]The GNG algorithm is an unsupervised incremental clustering algorithm. Given some input distributions in $\mathbb{R}^n$, GNG incrementally creates a network of nodes, where each node in the graph has a position in $\mathbb{R}^n$.

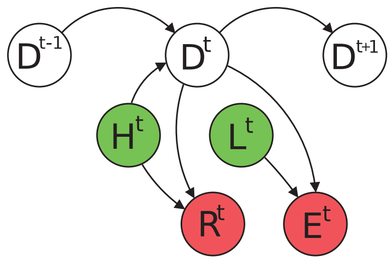


Figure 2. Partial representation of the model depicting the relation between the decision D^t, the stimuli H^t and L^t, and the actions R^t and E^t.

3.1. Improvements on Input–Output Hidden Markov Models

3.1.1. Semantic Refinement.

Principle. The problem [P2: Sensors] is related to the independence between the stimuli and the other random variables in the model. Decisions and actions can be obviously independent from some stimuli. Therefore, it is possible to alleviate the work of the learning algorithm by specifying from the beginning the independences.

The goal of the semantic refinement is to reduce the number of parameters by defining *a priori* some independence between random variables. By reducing the number of parameters and giving the model some knowledge, the learning should be faster without reducing the expressiveness of the model.

The principle of the proposition (Figure 2) is the following: instead of considering all the stimuli for each choice (decision and actions), each choice uses stimuli with different levels of granularity. Global and spatially inaccurate stimuli, called high-level stimuli (random variables H_i^t), are used for the choice of the decisions and also for actions that only involve the agent. High-level stimuli can be used to represent internal information (e.g., hunger, level of hormones, and pain) or global information about the surroundings (e.g., temperature, presence of food, and lighting). Spatially accurate stimuli, called low-level stimuli (random variables L_j^t), are used to perform actions that aim at an interaction with the environment. We also define two kinds of action, reflexive (R_i^t) and external actions (E_i^t), each one being a group of dependent actions. Low-level stimuli can be used, for instance, to represent the exact location of surroundings objects, their speed, and direction.

Example. We define the random variables and their values in Table 6 and give a graphical representation in Figure 3. In order to compare our model with Le Hy's, we define the equivalent model by using Le Hy's proposition (Figure 4). The independence between some stimuli, decisions, and actions reduces the number of the parameters of the model. A more detailed analysis is done in Section 4.1.1.

Comparison with Le Hy's Work. Our model can be seen as a generalization of Le Hy's model. By stating that $H^t = S^t$, the random variables L^t and E^t being unused, our model is equivalent to Le Hy's.

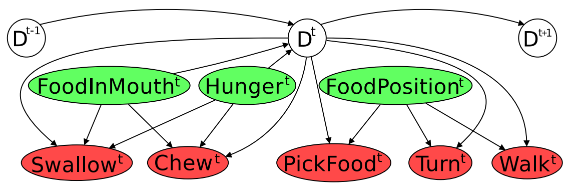


Figure 3. Example of an application of the model. *FoodInMouth* and *Hunger* are high-level stimuli, and *FoodPosition* is a low-level stimulus. *Chew* and *Swallow* are two reflexive actions. *Walk*, *Turn*, and *PickFood* are external actions.

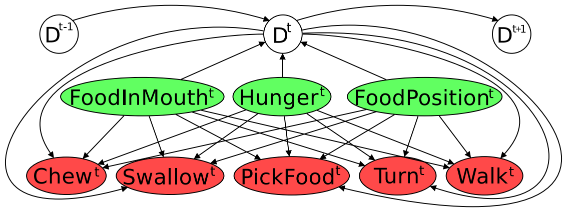


Figure 4. Example of a model following Le Hy's specifications and aiming at expressing the same behaviors as the example in Figure 3.

Table 6. Example of a model following the CHAMELEON proposition.

	Variable	Definition	Values
High-level stimuli (H)	H_1	*FoodInMouth*	(*No*, *Solid*, *Chewed*)
	H_2	*Hunger*	(*Low*, *Medium*, *High*)
Low-level stimuli (L)	L_1	*FoodPosition*	(*Close*, *Far*) × (*Right*, *Left*)
Reflexive actions (R)	R_1	*Chew*	(*Yes*, *No*)
	R_2	*Swallow*	(*Yes*, *No*)
External actions (E)	E_1	*PickFood*	(*Yes*, *No*)
	E_2	*Walk*	(*Foward*, *Backward*)
	E_3	*Turn*	(*Right*, *Left*)
Decisions (D)	D	*Decision*	(*FindFood*, *Eat*)

3.1.2. Attention Selection Mechanism.

Principle. Here we present another proposal in order to reduce the number of parameters: the attention selection mechanism. This principle is based on the imitation of a natural phenomenon: when one has to handle a high quantity of information, a solution is to select the most relevant part of this information and to act in accordance with this partial information. But this selection mechanism is complex and is not systematic: it depends on one's "internal state" such as current aims and constraints, which are subject to evolution along time. Therefore, we propose to use a mechanism where, at time t, the agent focuses on one high-level sensor (H_g^t) and one low- level sensor (L_j^t). In order to model this mechanism, we introduce two random variables G and J, which give respectively the index of the high-level sensor and the index of the low-level sensor the agent focuses on. The random variable G depends on the high-level sensors, so its distribution has the shape $P(G^t|H^t)$. On the other hand, the random variable J depends on the low-level sensors and on the current decision, so its distribution has the following shape: $P(J^t|D^t L^t)$. As a result, the agent will choose its actions according to some distributions that are defined as weighted sum of the single distributions: the ones the agent would refer to in the situation where it would face only one stimulus. The first gain in this strategy is quite obvious: one has to learn and store only one distribution by a sensor (e.g., one distribution by point of interest, instead of a distribution that would handle any number and kind of points of interest). Moreover, the conditional distributions of the random variables G and J may as well be too complex, so we propose to express these distributions as follows:

$$\mathbb{P}\left(G^t = i \,|H^t\right) = \frac{\theta_i\left(H_i^t\right)}{\sum_n \theta_n\left(H_n^t\right)} \tag{1}$$

$$\mathbb{P}\left(J^t = j \,|D^t, L^t, K^t\right) = \frac{\lambda\left(D^t, L_j^t\right)}{\sum_m \lambda\left(D^t, L_m^t\right)} \tag{2}$$

The higher the values of θ and λ, the more likely the agent will focus on the associated sensor. This reduces greatly the number of parameters still giving the agent a mechanism to focus only on one sensor. The model uses a very simple algorithm (Figure 5).

Example. A concrete example is given in Figure 6.

```
while agent in world do
  h ← agent's high-level stimuli
  l ← agent's low-level stimuli
  Pick $i$ using (1)
  Pick $d$ using $\mathbb{P}(D^t = d\,|D^{t-1} = d^{t-1}, H_i^t = h_i)$
  for all $u \in [\![1, N_R]\!]$ do
    Pick $r$ using $\mathbb{P}(R_u^t = r_u\,|D^t = d^t, H_i = h_i)$
  end for
  Pick $j$ using (2)
  for all $f \in [\![1, N_E]\!]$ do
    Pick $e$ using $\mathbb{P}(E_f^t = e_f\,|D^t = d^t, L_j = l_j)$
  end for
  Make avatar do $(r_1, ..., r_{N_R}, e_1, ..., e_{N_E})$
  $d^{t-1} \leftarrow d$
end while
```

Figure 5. Algorithm of the model. It is possible to choose the way the value are picked: randomly following the distribution, using the maximum value in the distribution, and so on.

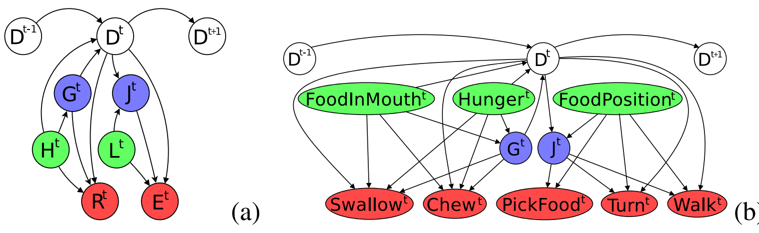


Figure 6. (a) Summary of the relation between the random variable of the model: in green, the inputs (sensory data); in red, the outputs (actions); and in blue, the attention variables. (b) Whole model applied to the example defined in Section 3.1.1.

Comparison with Le Hy's Work. Before we turn to the next section, we present a relevant comparison with a model developed by Le Hy. The first mechanism used by Le Hy is the IP, where $P(D^t|D^{t-1}S^t)$ is computed using $P\left(S_i^t|D^t\right)$, assuming that all sensors are independent knowing the decision. This hypothesis may be clearly wrong depending on the chosen sensors; moreover, the more sensors used, the higher the chances of the hypothesis to be wrong. The second mechanism is the FEC. This technique suffers some simple problems: it makes use of probability distributions but handle them in total opposition with their natural properties (by mean of products of distributions functions that share a common lower positive bound). The main consequence of this technique is, in the end, to consider something similar to a mean of probability distributions, which is easily and rigorously achieved with the sum of random variable over a random index (this is in scope of the mechanism we propose). From this point of view, it is noticeable that the mechanism we propose gives, but not only, a clear and rigorous formal basis to these kind of ad hoc computations.

3.2. Learning the Topology: Stable Growing Neural Gas

To achieve the best believability, we want those nodes to be learned by imitation of a human player instead of being placed *a priori* by a designer. This work has been done in [26] where nodes and a potential field are learned from humans playing a video game. The agent is then using this representation to move in the game environment, following the field defined at each node. To learn the position of the nodes, Thurau *et al.* used an algorithm called GNG.

The GNG [27] is a graph model that is capable of incremental learning. Each node has a position in the environment and has a cumulated error that measures how well the node represents its surroundings. Each edge links two

```
nodes ← {}
edges ← {}
while teacher plays do
  (x,y,z) ← teacher's position
  if |nodes| = 0 or 1 then
    nodes ← nodes ∪ {(x,y,z,error=0)}
  end if
  if |nodes| = 2 then
    edges ← {(nodes,age=0)}
  end if
  $n_1$ ← closest((x,y,z),nodes)
  $n_2$ ← secondClosest((x,y,z),nodes)
  edge ← edges ∪ {{$n_1$,$n_2$},age=0)}

  $n_1$.error+=||(x,y,z)-$n_1$||
  Attract $n_1$ toward (x,y,z)
  ∀ edge ∈ edgesFrom($n_1$), edge.age++
  Delete edges older than $\overline{Age}$
  Attract neighbours($n_1$) toward (x,y,z)
  ∀ node ∈ nodes, node.error-=$Err$

  if $n_1$.error > $\overline{Err}$ then
    maxErrNei ← maxErrorNeighbour($n_1$)
    newNode ← between($n_1$,maxErrNei)
    $n_1$.error/=2
    maxErrNei.error/=2
    newError ← $n_1$.error+maxErrNei.error
    nodes ← nodes ∪ {(newNode,newError)}
  end if
end while
```

Figure 7. Algorithm used to learn the topology of the environment by the stable growing neural gas.

nodes and has an age that gives the time it was last activated. This algorithm needs to be omniscient, because the position of the imitated player, the demonstrator, is to be known at any time.

The principle of the GNG is to modify its graph, adding or removing nodes and edges and changing the nodes' position for each input of the demonstrator's position. For each input, the closest and the second closest nodes are picked. An edge is created between those nodes, and the closest node's error is increased. Then, the closest nodes and its neighbors are attracted towards the input. All the closest node's edges' age is increased by 1, and too old edges are deleted. With each ζ input, a node is inserted between the node with the maximum error and its neighbors having the maximum error. At the end of an iteration, each node's error is decreased by a small amount. The graph stretch and grow to cover the whole space where the player has been monitored to go.

The model has been modified (Figure 7) to be able to learn continuously on a player without growing indefinitely but being able to grow if the teacher (in our case, the player) begins to use a new part of the environment. We called this new version *stable GNG* (SGNG). Instead of inserting a new node each ζ input, a node is inserted when a node's error is superior to a parameter $\overline{Err}$. As each node's error is reduced by a small amount Err for each input, the SGNG algorithm does not need a stopping criterion. Indeed, if there are many nodes, which represent well the environment, the error added for the input will be small, and for a set of inputs, the total added error will be distributed among several nodes. The decreasing of error will avoid new nodes to be added to the GNG, resulting in a stable state. However, if the player who serves as an example, the demonstrator, goes to a place in the environment he or she has never gone before, the added error will be enough to counter the decay of the error, resulting in new nodes to be created.

The nodes learned by this model can be used directly by the model as low-level stimuli. Indeed, they represent precise information, which will be particularly important for the choice of motion actions. However, the information given by the edges cannot be used, as it denotes only proximity between nodes and not a path between them. Two nodes may be linked by an edge because they are close with an obstacle between them.

3.3. Learning the Parameters of the Model with Expectation–Maximization Algorithm

In our model, probabilities computed by the backward procedure give the chances of taking a certain decision at t, knowing what the demonstrator did at $t+1,\ldots,T$. This information should not be lost (for instance, if the demonstrator is looking for something specific, the learning algorithm cannot know what it is until it is picked up). As a consequence, it seems wiser to learn on a whole sequence of observations (from time 0 to time T) instead of using an incremental version.

Expectation–Maximization Principle. In our case, the algorithm gathers the values of the stimuli L_j^t and H_i^t and the actions R^t and E^t at each time step. The values of the hidden states I^t, J^t, and D^t are not known; thus, the data are incomplete. We will apply an EM algorithm [28] to be able to learn the model parameters.

The following notations are adopted:

- T is the length of the sequence of observation used for the learning;
- $\mathcal{A} = \mathcal{A}^{1,\ldots,T}$ the total sequence of observed actions;
- $\mathcal{S} = \mathcal{S}^{1,\ldots,T}$ the total sequence of observed stimuli;
- $\mathcal{Q} = \mathcal{Q}^{1,\ldots,T}$ the total sequence of hidden states; and
- Φ is the set of model parameters.

Our goal is to find the best model parameters, Φ^*, such that the likelihood, $P(\mathcal{A}|\mathcal{S},\Phi^*)$, is maximal. This means that we want to find the parameters of the model that are most likely to generate an observed sequence given the sensory information: if the model were in place of the observed player, it would most likely generate approximatively the same actions. As we do not have any information about the hidden variables ($\mathcal{Q}$), we choose to use an EM to find a local maximum. The idea is to find iteratively a Φ_{n+1} such that $P(\mathcal{A}|\mathcal{S},\Phi_{n+1}) \geq P(\mathcal{A}|\mathcal{S},\Phi_n)$.

Finding a Sequence of Observations. During the introduction of the algorithm, we defined the sequences of observation $\mathcal{A}$ and $\mathcal{S}$ of length T. They are the sequences of what the teacher does and perceives. These sequences allow the model to estimate the probability of the hidden states. In our games, avatars "die" disappearing from the environment and "resurrect" reappearing in a random place. Therefore, a sequence is the actions and stimuli from the "resurrection" to the "death" of the teacher's avatar. Such sequences usually last from 10 s to 5 min, which is not too long for the algorithm. A last problem concerning the sequences had to be solved: the teacher has a reaction time. Among all the solutions we tried, the most simple worked the best. Instead of associating the actions at t to the stimuli at t, we use the actions at $t + t_{\text{reac}}$. The actual value of t_{reac} is discussed in Section 4.3.1.

Parameters Initialization. We choose to stick to the simplest method for the moment: the random initialization. Each parameter is initialized to a random value, then they are normalized for the sum of probability distributions to be equal to 1. It allows the algorithm to cover many possible solutions at the cost of convergence time.

Stopping Criterion. The algorithm needs a stopping criterion, based on the quality of the current set of parameters Φ_n. As the algorithm converges towards a local maximum, we do not know *a priori* the value of the likelihood function at this maximum. When the value of $Q(\Phi_{n+1}|\Phi_n)$ converges, it increases, so that the increase is smaller and smaller at each iteration of the algorithm. We based the stopping criterion on this increase: if $Q(\Phi_{n+1}|\Phi_n) - Q(\Phi_n|\Phi_{n-1}) < w$, the algorithm is stopped, and Φ_{n+1} is considered to be the solution.

Expectation–Maximization Online. Like all the EM, our algorithm is offline, which is contrary to our objective. Indeed, the algorithm gives a set of parameters that only satisfies the observed sequence. However, because we have short learning sequences, we can learn from them one after another, merging the final resulting set of parameters to a global one, Φ_g.

Results. Our learning algorithm allows to learn almost all the parameters by imitation. The attention functions θ_i and λ_j are not yet learned. Our algorithm is much slower than Le Hy's, but by avoiding simplistic hypothesis, should give much more accurate results (Section 4.3.2).

4. RESULTS

In this section, we present how we adapted our model to the video game *UT2004* and the result for each of the four proposed modifications. The semantic refinement of the model reduces the number of parameters for the model, making the learning faster and the model clearer (Section 4.1.1). The attention selection mechanism allows the agent to express behavior that cannot be expressed by most IOHMM models, as Le Hy's model (Section 4.1.2). The SGNG makes the agent able to adapt rapidly to unknown environments by observing multiple teachers (Section 4.2). Finally, the imitation learning algorithm allows the agent to evolve rapidly towards a more believable behavior (Section 4.3).

4.1. Improvements on Classical Input–Output Hidden Markov Models

4.1.1. Semantic Refinement.

The definition of high-level and low-level stimuli allows the model to reduce the parameters by

$$\left(\prod_{j=1}^{N_L} L_j\right)|D|^2 + \left(\prod_{i=1}^{N_H}|H_i|\sum_{f=1}^{N_E}|E_f| + \prod_{j=1}^{N_L}|L_j|\sum_{u=1}^{N_R}|R_u|\right)|D| \quad (3)$$

So the more complex the model, the more favorable is the semantic refinement.

In order to study the consequences of the semantic refinement on a concrete example, we will use our application. The definition of each random variable and the number of values they can take are given in Table 7.

With this example, we can now study the number of values needed for the definition of Le Hy's model and CHAMELEON. In order to focus only on the influence of

Table 7. Definition of each random variable used in the model applied to the game UT2004.

	Variable	Definition	Number of values
High-level stimuli (*H*)	H_1	*Life*	3
	H_2	*NumberOfEnemies*	4
	H_3	*CurrentWeapon*	14
	H_4	*CurrentWeaponAmmunition*	4
	H_5	*TakeDamage*	2
	H_6	*WeaponsInInventory*	70
Low-level stimuli (*L*)	L_1	*Player*	405
	L_2	*Weapon*	45
	L_3	*Health*	45
	L_4	*NavigationPoint*	72
	L_5	*RayImpact*	5
Reflexive actions (*R*)	R_1	*ChangeWeapon*	12
External actions (*E*)	E_1	*Fire*	3
	E_2	*Jump*	3
	E_3	*Pitch*	3
	E_4	*Yaw*	5
	E_5	*Walk*	3
	E_6	*Lateral*	3
Decisions (*D*)	*D*	*Decision*	around 10

the semantic refinement of the stimuli, we will not consider for now the mechanisms to decrease the complexity of the models: FEC and IP for Le Hy's model and attention selection for CHAMELEON.

In our application, the gain is worthwhile, the decrease of the number of parameters being between 10% and 92% compared with a model without semantic refinement is show in Figure 8 and Table 8. In the game UT2004, the number of states, which approximatively corresponds to decisions in our model, is around 10. For 10 decisions, the reduction of the number of parameters is around 85%.

The semantic refinement of the model, clearly defining high-level and low-level stimuli and the associated actions, allows an important reduction of the number of values needed for the definition of the probability distributions. This reduction will make the learning faster because less knowledge is to be learned, the independence between variables being already specified. This should allow the agent to adapt even faster answering to the requirement [B10: Fast evolution].

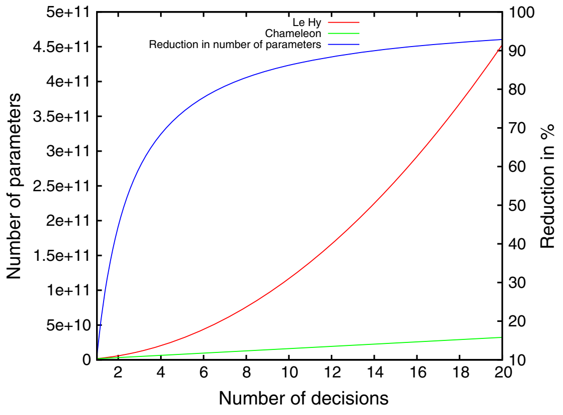


Figure 8. Number of parameters for our application of the model in UT2004. The IP and FEC for Le Hy's model and attention mechanism for CHAMELEON are not taken into account. The reduction is given as compared with Le Hy's model. IP, inverse programming; FEC, fusion by enhanced coherence.

Table 8. For each hypothesis, the number of values for the definition of the probability distributions with 10 decisions.

	Number of parameters
Full dependence between random variables	3×10^{18}
Independence of actions	8.6×10^{15}
Independence of actions and semantic refinement	6×10^{10}

The semantic refinement allows a noticeable reduction of the number of parameters. The independence of the actions is introduced in Le Hy's work.

4.1.2. Attention Selection Mechanism.

Number of Parameters. The difference in the number of parameters for the previous example is given in Table 9 and in Figure 9. All the modifications we proposed decrease the number of parameters by 36% to 40% compared with that by Le Hy's model. However, the results for this example cannot be generalized for all the problems. Indeed, the semantic refinement allows an important decrease in the number of parameters, whereas the attention selection makes the number of parameters increase. There may be some applications of our model where more parameters are needed than Le Hy's.

Expressiveness. Le Hy's FEC cannot express several simple behaviors. In our example (Figure 10), if the attractors are navigation points and the actions are forward or backward, the agent will randomly go back and forth, barely moving because the "expected value" is to stay still. There is also another problem: if a random distribution is

Table 9. For each hypothesis, the number of values for the definition of the probability distributions with 10 decisions.

	No. of parameters
Full dependence between random variables	3×10^{18}
Independence of actions	8.6×10^{15}
Independence of actions and semantic refinement	6×10^{10}
Le Hy: independence of actions and IP and FEC	2.2×10^{5}
CHAMELEON: independence of actions and semantic refinement and attention	1.4×10^{5}

Our model totals 36% less parameters than Le Hy's model. The IP and FEC are introduced in Le Hy's thesis and the attention selection mechanism.
IP, inverse programming; FEC, fusion by enhanced coherence.

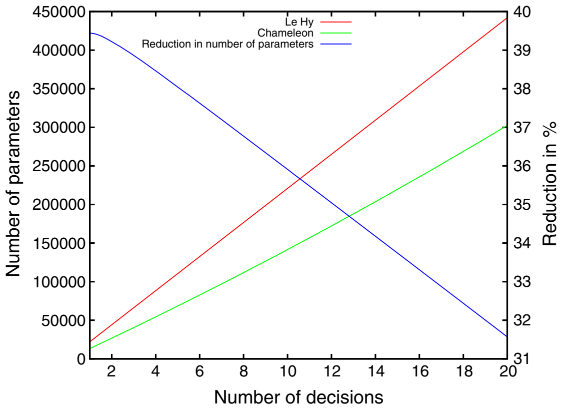


Figure 9. Number of parameters for our application of the model in UT2004. The reduction is given compared with that in Le Hy's model.

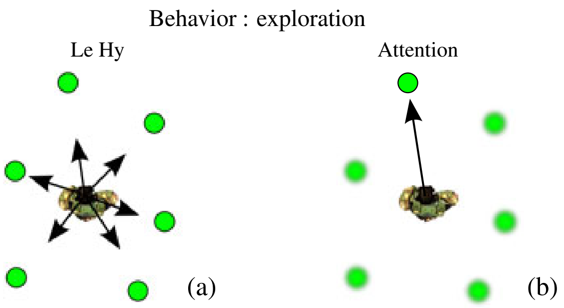


Figure 10. (a) An illustration of the problem with FEC: it does not give a believable action because the agent may constantly switch between the two attractors, oscillating constantly. (b) The attention selection mechanism produces a better distribution in term of believability: the attention gives a believable action because the agent focus on the attractor ahead instead of switching between the two attractors like the FEC would do. FEC, fusion by enhanced coherence.

0 for an action, the product is also 0. For our example, one could set the value for going backward when seeing an attractor in front to 0 and vice versa. In this case, the FEC cannot be applied because the product would be 0 for all the probabilities.

4.2. Learning the Environment: Stable Growing Neural Gas

Results 2D/3D. We trained two SGNG on two different maps. The first one is a simple map, called Training Day; it is small and flat, which is interesting to visualize the data in two dimensions. The second one, called Mixer, is much bigger and complex with stairs, elevators, and slopes, which is interesting to see if the SGNG behaves well in three dimensions. The results are given in Figure 11.

Similarity with Manually Placing the Navigation Points. To study the quality of the learned topology, we first chose to compare the SGNG's nodes with the navigation point placed manually by the map creators (Figure 12). Of course, we do not want the SGNG to fit exactly those points, but it gives a first evaluation of the learned representation. The two representations look alike, which indicates that the model is very effective in learning the shape of the map. However, there are zones where the SGNG's nodes are more concentrated than the navigation points and other zones where they are less concentrated. We cannot tell now if it is a good behavior or not, as we should evaluate an agent by using this representation to see if it navigates well. Even in the less-concentrated zones, the nodes are always close enough to be seen from one to another, so it should not be a problem.

Time Evolution. To study the time evolution of the SGNG's characteristics, we introduce a distance measure: the sum of the distance between each navigation point and its closest node. We also study the evolution of the number of nodes because we do not want the SGNG to grow indefinitely. Figure 13(a) shows this two measures for the simple and the complex maps. For the simple map, the SGNG reached its maximum number of node and minimum error in approximatively 5 min of real-time simulation. For the complex map, it takes more time, about 25 min, but results at 12 min are quite good. Those results show that it is possible to have an agent learn during the play.

Learning with Multiple Professors. The SGNG can handle inputs from multiple professors. Figure 13(b) shows the distance and number of node for an SGNG trained on one professor and for an SGNG trained on four professors. The learning with four professors is, as expected, faster: about 3 min for the distance to stabilize instead of 5 min for one professor. It is interesting to note that the learning is not four times faster, but the gain is still important. Learning with multiple professors seems to give an SGNG with less variation during the learning. The gain has, however, a small drawback: the number of nodes is a little superior for multiple professors. It may be because professors are scattered in the environment instead of a unique professor following a path.

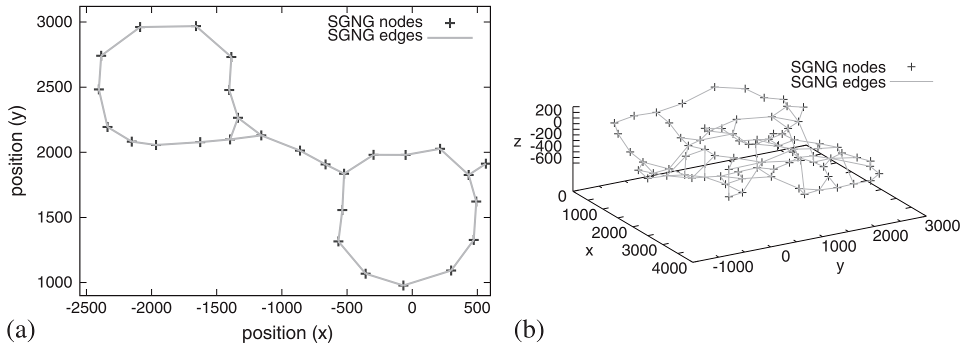


Figure 11. Result of a Stable Growing Neural Gas (SGNG) learned (a) from a player for a simple map and (b) from a player for a complex map.

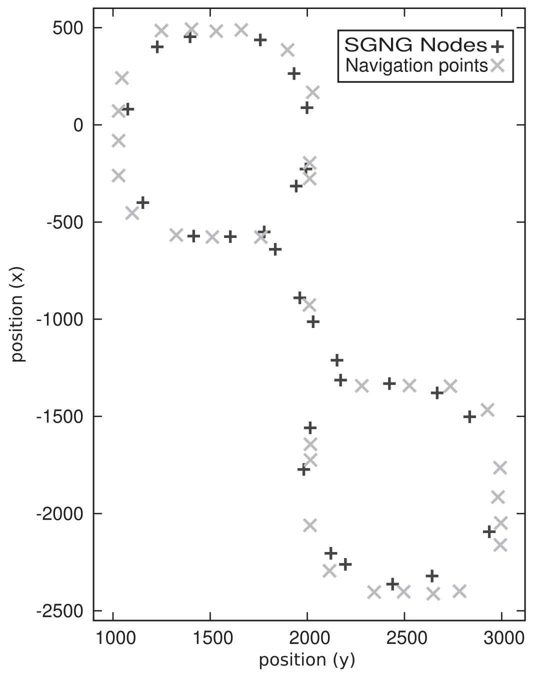


Figure 12. Comparison of nodes learned by the stable growing neural gas (GNG) with the navigation points placed manually by the game developers.

Comparison: Growing Neural Gas/Stable Growing Neural Gas. Figure 14 shows a comparison between the GNG and the SGNG, using three measures during the learning for the simple environment. *Sensitivity* measures how much the SGNG successfully represents the part of the environment the teacher used, which can be seen as true positives. The higher the value, the better the SGNG is. *Specificity* measures how much the SGNG did not represent the part of the environment the teacher did not use, which can be seen as true negatives. The higher the value, the better the SGNG is. We also study the number of nodes the SGNG has because we do not want the SGNG to have either too many or too few nodes.

4.3. Learning the Parameters of the Model with Expectation–Maximization Algorithm

In this section, we first study the influence of the data given to the EM: the observation sequences and the model parameters (Section 4.3.1). Then, we study the convergence of a learning on one sequence and multiple sequences (Section 4.3.2). Finally, we try to evaluate the resulting behaviors with different objective and subjective measures (Section 4.3.3).

4.3.1. Impact of the Expectation–Maximization and Parameters on the Results.

Impact of the Teacher's Reaction Time. When the teacher is observed by the learning algorithm, a snapshot of the values of the stimuli and the actions is taken at each time step t. However, the teacher's actions at time t may not reflect a reaction to the stimuli at time t. We must then take into account the reaction time of the teacher, allowing our model to find the relation between stimuli and actions, which are actually related.

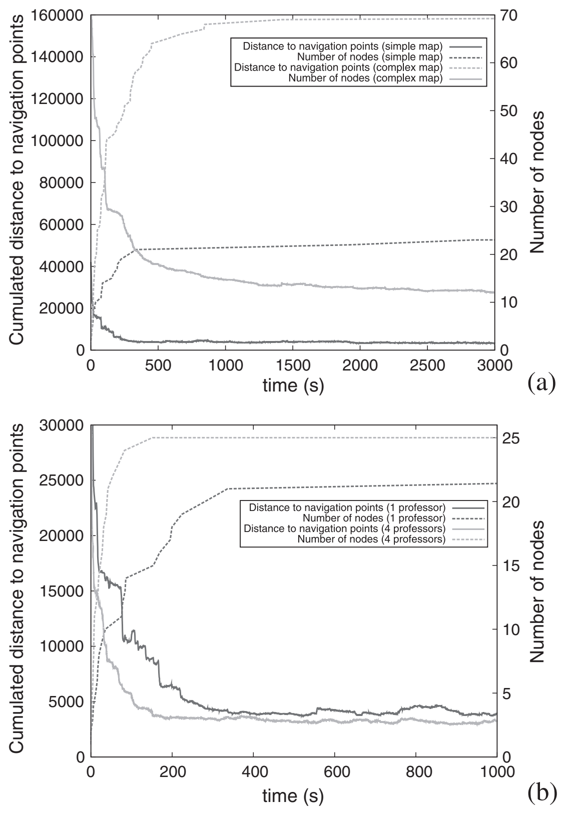


Figure 13. Time evolution of the cumulated distance to navigation points (a) defined manually and (b) the SGNG's nodes and the SGNG's number of nodes.

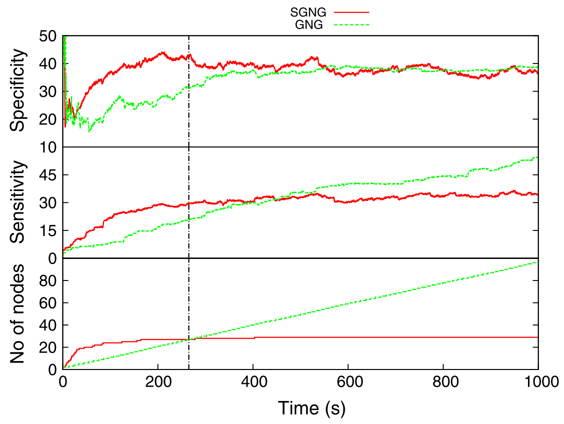


Figure 14. Comparison: growing neural gas (GNG)/stable growing neural gas (SGNG).

The reaction time may not be the same between individuals and may also vary for one individual over the time. However, we find that the simplest solution, using a constant reaction time for all the teachers, give the best results. Other methods were tried, such as associating the more likely action to the stimuli according to the current model parameters or aligning variations in both stimuli and actions, but they did not make the learning algorithm converge towards parameters more likely to generate the observations. The choice of the reaction time for a game has to be done once and for all using a graph like Figure 15.

According to Figure 15, the reaction time of a player is very variable. The reaction time of 300 ms gives the maximum likelihood, but the difference is not very important with the likelihood found for reaction times between 400 and 800 ms. For now, we will use a reaction time of 300 ms in the following experiments, but a variable time of reaction could improve the results.

In order to show that the results of the learning are really impacted by the reaction time, we studied the variations of the log-likelihood depending on the reaction time for a UT2004 agent teacher (all the other experiments are done with human teachers). Figure 15 shows that the best log-likelihood are achieved for reaction times between 0 and 200 ms. These results are coherent with the fact that UT2004 agents do not model the reaction time. It also shows that the players' behaviors are much more complicated to learn, their reaction time being much more variable.

Impact of the Number of Decisions. The mechanism of decisions allows the agent to produce logical sequences of actions, to make actions according to its needs, and to simulate a short-time memory. As expected, Figure 16 shows that the more the decisions, the better the model can express the observed behaviors. Indeed, the model is more complex and can make the agent behave in a more subtle way. However, as the number of parameters increase with the number of decisions, the time needed for the algorithm to converge is longer. Too many decisions should be avoided to make the learning fast and fulfill the requirement [B10: Fast evolution]. According to the results, increasing the number of decisions over 10 does not improve the results in a significant way. We remind the reader that agents originally coded in the UT2004 game use approximatively 10 main states.

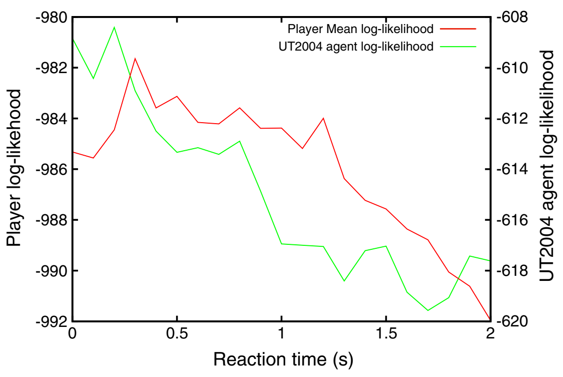


Figure 15. Mean log-likelihood of the final result after learning on 105 different sequences of observations of the behavior of a player and 100 different sequences of observations for a UT2004 agent. The reaction time varies from 0 to 2 s, and the model has 10 decisions.

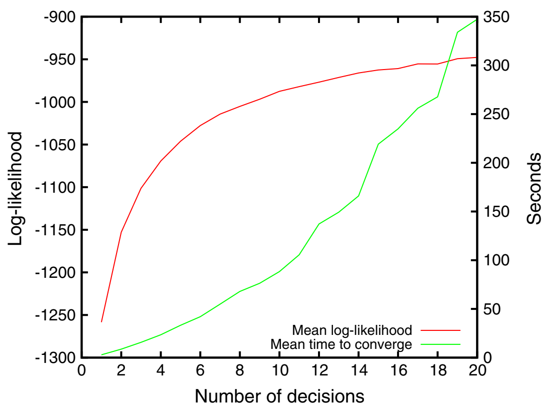


Figure 16. Mean log-likelihood of the final result and time to converge for 105 different sequences of observations. The number of decisions varies from 1 to 20 with a fixed reaction time of 300 ms.

4.3.2. Characteristics of the Expectation–Maximization.

According to the two previous studies, we will use 10 decisions and a reaction time of 300 ms for the following experiments. The function Q is the function that is optimized by the EM algorithm. The higher it is, the higher is the log-likelihood. Figure 17 shows the value of the total log-likelihood for each iteration of the EM. The first iterations make the value increase sharply; after 10 iterations, the increase becomes very slow, stabilizing between -500 and -300. As the value is a log, this likelihood is very small. We can also see that some learnings finish faster, in about 30 iterations, whereas others take around 100 iterations.

4.3.3. Resulting Behaviors.

The final goal of the four propositions (semantic refinement, attention selection, learning of the environment, and learning of the behavior) is to make the agent produce believable behaviors. In the following experiments, the model learned on one unique teacher using 10 decisions. It also considered that the teacher had a constant reaction time of 300 ms. The results given are after learning on the teacher during 40 min. The teacher played against a UT2004 agent in the environment named in the game *Training Day*.

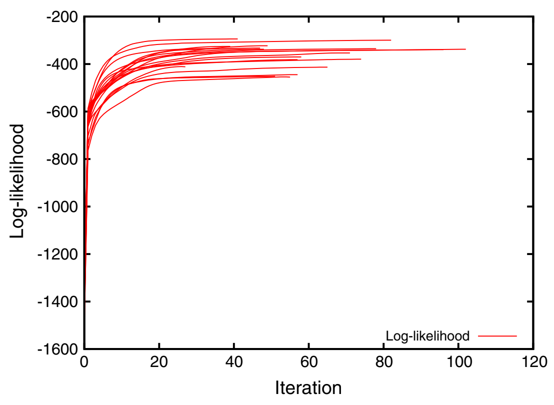


Figure 17. Time evolution of the total log-likelihood for 20 learnings on sequences of the same length. Each learning starts the same initial distributions.

Study of the Distributions. Before analyzing the whole behavior, we can study the distributions of actions to see if they really look like the ones in [25, page 47, in French]. For the same decision, the distributions shown in Figure 18 confirm that the agent does not react in the same way for different positions of the same object. In the example, if a weapon is at the left, close, and of the same height as the agent (left figure), the agent will go forward, move laterally left, and not turn. If the weapon is at the right and of the same height (right figure), the agent will probably not turn, go forward, and move laterally right. In the two cases, the agent will look in the direction of the horizon, which is the direction of the weapon in terms of pitch. In the two cases, the agent will surely pick the weapon.

For the same stimulus but different decisions, the agent may act differently. Figure 19 shows the distributions for an enemy player on the right of the agent, moving to the right, of the same height, and at an average distance for two different decisions. In the first decision (left figure), the agent moves forward and turns right in order to aim at the enemy and reduce the distance to the player. In the second decision, the agent turns also right but may move forward or backward and move laterally to the left in order to aim at the enemy but keep the same distance to the player. In the two cases, the agent looks at the direction of the horizon, as it is the direction of the enemy.

The study of the movement distributions shows that some knowledge is assimilated by the model parameters. The role of the decision becomes obvious with different tactics being used according to the state of the agent. However, some distributions do not represent a believable behavior at all. The reasons can be many: problem with reaction time, bad attention values, and so on.

Signatures. In a previous study [29], we presented a method to spot differences in the behavior of players and agents. Although it is not proved to be an indicator of believability, it may be used to spot non-believability. The idea is to monitor the movements of avatars and to extract some statistics. These statistics are the signatures of the behaviors; similarities can be spotted between agents and between players. The most important information is the differences spotted between players and agents, which often reveal problems in the behavior of the agent. It is also possible to represent the distance between the signatures, visualizing the similarities in a more natural way. This distance allows the measure to take into account that turning 80° right and 90° right is almost the same but 90° right and 90° left is very different. So to visualize the distance, we represent the signatures on a plane by using the Multidimensional Scaling (MDS) method. Both the Earth Mover's Distance (EMD) and MDS are detailed in [29]. The idea is that close (Euclidean distance) representations of the signatures in the MDS are close in the EMD distance and thus are similar, the contrary being also true. As it is only a question of relative distance to the other, the graphs do not have any tics or values on the axis. In order to have a more complete

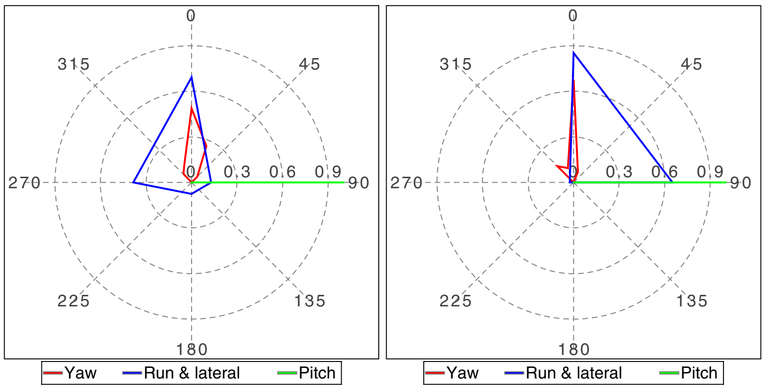


Figure 18. Two distributions of movement actions for the same decision and same kind of stimuli but different values. The left graph is for a stimulus representing a weapon on the left of the agent and very close. The right graph is for a stimulus representing also a weapon but is on the right of the agent and is also very close. The weapon is about the same height as the agent.

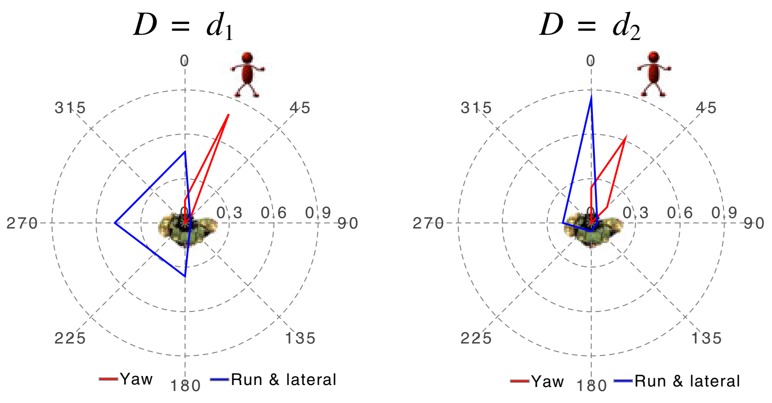


Figure 19. Two distributions of movement actions for the same stimulus but different decisions. The two graphs are for stimulus representing an enemy player, right of the agent, moving to the right of the agent, and at an average distance.

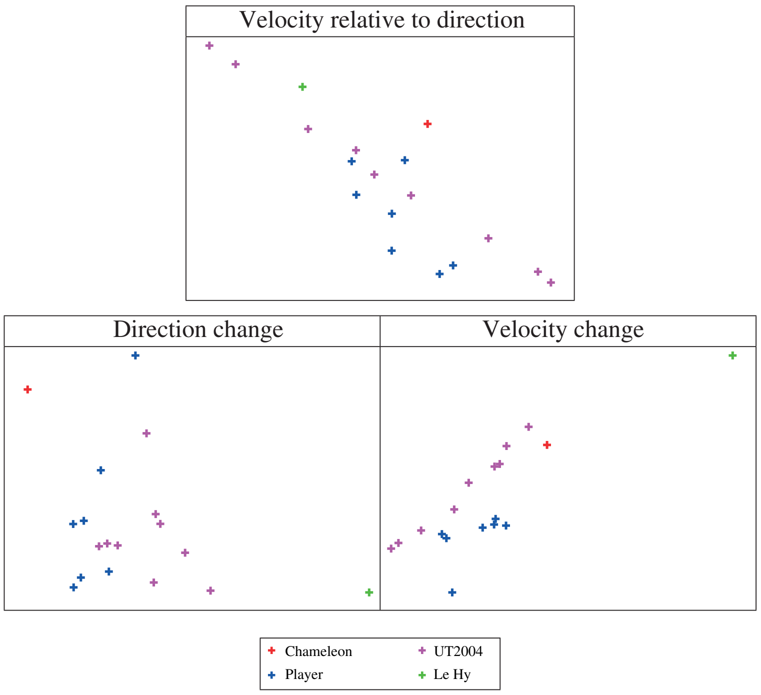


Figure 20. EMD between the signatures represented using the MDS for one CHAMELEON, one Le Hy agent, seven different players, and nine UT2004 agent, each one with a different skill level. The correlation factors for the MDS for all the representations are very high (>0.98).

study of the signatures, we represent the signatures for the CHAMELEON and the Le Hy agent as well as seven different players and nine UT2004 agents, each with a different skill level.

For the first graph (Figure 20, top), the UT2004 agents are widespread. The medium-skilled ones are close to the players. Le Hy agent is pretty far from the players and CHAMELEON is not very far but not in the "cloud" of the players. That means it may be taken for a player but not an average one. In the second graph (Figure 20, bottom left), the players are quite widespread, and the UT2004 agents are still quite close. Le Hy agent is very far from the players and CHAMELEON is a bit far also. Finally, for the last graph (Figure 20, bottom right), Le Hy agent is very far from the players, CHAMELEON is quite far, and the agents are not very far but can be clearly separated from the players. To conclude, it appears that some UT2004 agents are the closest to the players, then comes CHAMELEON and then Le Hy's agent. The fact that the agents from the game are this close is because character designers used much time improving the way their agents move. However, it may not apply to the behaviors.

It seems that CHAMELEON managed to perform better than a particular IOHMM (Le Hy's agent). Because the discretization of the actions and the interface between the game and the model can be greatly improved, UT2004 agents perform better than CHAMELEON. As all the details of the agent's behavior are important, this should be improved on CHAMELEON in order to be able to fool the players into thinking CHAMELEON is another player.

5. CONCLUSION

5.1. Bottleneck

This paper aims at designing a behavior model for the control of believable characters in video games. The character is controlled by a computer program we call an agent. We define a *believable agent* as a computer program able to control a virtual body in a virtual environment so that other human users in the environment think the virtual body is controlled by another human user. As this definition is pretty vague, we define 10 requirements for a character to be believable, on the basis of previous experiments and work: reaction, reaction time, variability, unpredictability, understandability, human-like perceptions, planning abilities, memory evolution, and fast evolution.

5.2. Contribution

In order to fulfill these requirements, we studied the existing behavior models developed both in the research and the industry. We grouped them into four types: connectionist models, state transition systems, production systems, and probabilistic models, each one having its strengths and weaknesses. As one of the requirement is that the model is able to evolve, we had to find learning algorithms for the behavior model. We found out that imitation is the best way to believability. With these studies in mind, we found out that the behavior model developed using IOHMM, as Le Hy's model, answers to most of the requirements but has still some limitations.

In this paper, we proposed four modifications or replacements to classical IOHMM models. We first tried to reduce the number of parameters in the model with a semantic refinement. Then, we replaced the two mechanisms to break the complexity of the probability distributions by an attention selection mechanism. This avoids the agent from switching constantly between stimuli. We added to the model the ability to learn by imitation the layout of environments with a model named growing neural network. Finally, we totally revamped the learning algorithm with an EM method.

The proposition makes the model able to learn how to act in the environment rapidly. Stimulus–action associations are made for the agent to look like a human player. However, the learning also learned wrong associations, which destroys the illusion of believability. According to our studies, our model performs better than Le Hy's but work is still to be done on the model to achieve the final goal.

5.3. CHAMELEON: A Novel Contribution for a Character to be Believable

As a conclusion, the propositions allowed the agents controlled by our model to fulfill the requirements listed in the beginning. The model and the learning algorithm make the agent react to stimuli often in a human-like fashion [B1: Reaction] and can simulate delay in the reaction [B2: Reaction time]. The probabilities allow the agent to fulfill [B3: Variability] and [B4: Unpredictability]. However, learned distributions can make the movements choppy, so it can be necessary to modify the way the model picks decisions. The agent can adapt to new environments and rules with the GNG and the EM [B9: Evolution] with results converging very rapidly [B10: Fast evolution]. The attention selection mechanism makes the behavior of the agent reflect more human-like perceptions [B6: Perception] and more understandable for players [B5: Understandable]. Finally, the agent can "remember," as its choice of decision is based on the previous one [B8: Memory]. This information is, however, very limited. Indeed, it often does not give enough information for the agent to react to stimuli that are not visible any more.

5.4. Future Work

The next step is to evaluate our work. As believability is subjective, evaluation is a critical and complex step. Even if it was not intended to, Turing's test is still considered as a reference for believability evaluation [30]. In its standard interpretation, a judge must chat with a human and a machine by using only text. If, after a certain amount of time, the judge cannot tell which one is artificial, the machine is said to be intelligent. This test's goal was to assess intelligence, but it has been much criticized [31,32]. This critique, however, does not apply to believability, and it is even a very good basis for assessing believability as we defined earlier.

There are many parameters for believability evaluation methods [8,14,33]. The first one is to cast or not to cast doubt on the nature of the presented character(s). This choice is often linked with mixing agents and humans so that the judges assess real humans' believability too. This can be useful to avoid bias induced by prejudices and to have a reference: humans usually do not score a perfect believability. Another parameter is the number of questions and answers. Turing's test features only one question and a yes/no answer, whereas other tests feature many questions and scales to answer. The former choice may be too restrictive, whereas the latter may result in too much undecided answer to "beat" the test. Another problem is the computation of the overall believability score, which, in case of multiple questions, may give experimenters too much influence on the results. To add more objectivity, it is possible to have relative scoring instead of absolute: the score given to an example can answer to "Is example A better than example B?" It is also necessary to decide if judges are players or only spectators. Whereas players can actively test evaluated characters, spectators are more focused on them and can notice much more details in the behaviors. Finally, the choice of the judges is really important. Cultural origins [8] and level of experience [34] may have a noticeable impact on believability scores.

Moreover, we plan to design specific experiments in order to evaluate online both the learning process and the believability of our model (and of similar ones). This work will be done in collaboration with psychology specialists. The principle is quite simple but should be particularly relevant in our context: we aim to design testing servers embedded with a learning artificial character, controlled by our CHAMELEON algorithm. Any human player who participates to the test and connects to the server has to play according to specific rules or scenarios designed to evaluate any part of the learning process and of the believability of our model. Once the player ends his session, he must answer a few questions, also precisely designed, that will evaluate how believable the other player is (the artificial one). The specific part of our experiment is that the artificial player will learn only during the tests, when human players/testers will participate. Thus, we will be able to plot the evolution of the believability of our model as more and more players will have participated, even many times each; we evaluate both the efficiency of the learning algorithm and the efficiency of the model itself to produce believable behaviors. We believe that this could produce a very versatile, informative, and autonomous evaluation workbench.

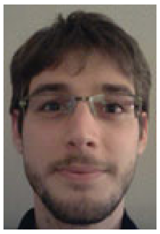
REFERENCES

1. Slater M, Usoh M, Steed A. Taking steps: the influence of a walking technique on presence in virtual reality. *ACM Transactions on Computer-Human Interaction (TOCHI)* 1995; **2**(3): 201–219.
2. Steuer J. Defining virtual reality: dimensions determining telepresence. *Journal of Communication* 1992; **42**(4): 73–93.
3. Held R, Durlach N. Telepresence, time delay and adaptation. *Pictorial Communication in Virtual and Real Environments,* chapter 14, 1991: 232–246.
4. Del Bimbo A, Vicario E. Specification by example of virtual agents behavior. *IEEE Transactions on Visualization and Computer Graphics* 1995; **1**(4): 350–360.
5. Gorman B, Humphrys M. Imitative learning of combat behaviours in first-person computer games, In *Proceedings of CGAMES 2007, the 11th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games*, La Rochelle, France, 2007.
6. Bauckhage C, Gorman B, Thurau C, Humphrys M. Learning human behavior from analyzing activities in virtual environments. *MMI-Interaktiv* 2007; **12**: 3–17.
7. Cavazza M, Charles F, Mead S. Interactive storytelling: from AI experiment to new media. In *ICEC '03: Proceedings of the Second International Conference on Entertainment Computing*. Carnegie Mellon University, Pittsburgh, PA, USA, 2003; 1–8.
8. Mac Namee B. Proactive persistent agents: using situational intelligence to create support characters in character-centric computer games, *Ph.D. Thesis*, Trinity College Dublin, 2004.
9. Silverman BG, Bharathy G, O'Brien K, Cornwell J. Human behavior models for agents in simulators and games: part ii: gamebot engineering with PMFserv. *Presence: Teleoperators and Virtual Environments* 2006; **15**(2): 163–185.
10. Bates J. The nature of characters in interactive worlds and the Oz project. *Technical Report CMU-CS-92-200*, School of Computer Science, Carnegie Mellon University, 1992.
11. Thomas F, Johnston O. *Disney Animation: The Illusion of Life*. Abbeville Press, New York, USA, 1981.
12. Riedl MO, Young RM. An objective character believability evaluation procedure for multi-agent story generation systems. In *Intelligent Virtual Agents*, Vol. 3661. Springer, Berlin, Germany, 2005; 278–291.
13. Loyall AB. Believable agents: building interactive personalities, *Ph.D. Thesis*, Carnegie Mellon University, 1997.
14. Livingstone D. Turing's test and believable AI in games. *Computers in Entertainment* 2006; **4**(1): 6.
15. Wetzel B. Step one: document the problem. In *Challenges in Game Artificial Intelligence: Papers from the 2004 AAAI Workshop*. AAAI Press, Menlo Park, CA, 2004; 11–15.
16. Laird JE, Duchi JC. Creating human-like synthetic characters with multiple skill levels: a case study using the Soar Quakebot, In *Simulating Human Agents, Papers from the 2000 AAAI Fall Symposium*, Massachusetts, USA, 2000; 75–79.
17. Bryant BD, Miikkulainen R. Evolving stochastic controller networks for intelligent game agents, In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, Vancouver, Canada, 2006; 3752–3759.
18. Isla D. Handling complexity in the Halo 2 AI, In *Game Developers Conference*, San Francisco, USA, 2005; 12.
19. Pinchbeck D. Trigens can't swim: intelligence and intentionality in first person game worlds. In *Proceedings of the Philosophy of Computer*

Games 2008. University of Potsdam, Potsdam, 2008; 242–260.

20. Cass S. Mind games. *IEEE Spectrum* 2002; **39**(12): 40–44.
21. Mac Namee B. Proactive persistent agents: using situational intelligence to create support characters in character-centric computer games, *Ph.D. Thesis*, University of Dublin, 2004.
22. Thurau C, Paczian T, Bauckhage C. Is Bayesian imitation learning the route to believable gamebots? In *GAMEON-NA'2005*, Montreal, Canada, 2005; 3–9.
23. Le Hy R, Arrigoni A, Bessiere P, Lebeltel O. Teaching Bayesian behaviours to video game characters. *Robotics and Autonomous Systems* 2004; **47**: 177–185.
24. Tence F. Probabilistic behaviour model and imitation learning algorithm for believable characters in video games, *Ph.D. Thesis*, UBO, Brest, France, 2011.
25. Le Hy R. Programmation et apprentissage bayésien de comportements pour des personnages synthétiques, application aux personnages de jeux vidéos, *Ph.D. Thesis*, Institut National Polytechnique de Grenoble, 2007.
26. Thurau C, Bauckhage C, Sagerer G. Learning human-like movement behavior for computer games, In *Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior (SAB'04)*, Los Angeles, USA, 2004; 315–323.
27. Fritzke B. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*. MIT Press, Massachusetts, USA, 1995; 625–632.
28. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 1977; **39**(1): 1–38.
29. Tence F, Buche C. Automatable evaluation method oriented toward behaviour believability for video games, In *International Conference on Intelligent Games and Simulation (GAMEON'08)*, Valencia, Spain, 2008; 39–43.
30. Turing AM. Computing machinery and intelligence. *Mind* 1950; **59**(236): 433–460.
31. Hayes P, Ford K. Turing test considered harmful. In *International Joint Conference on Artificial Intelligence*, Vol. 14. Lawrence Erlbaum Associates LTD, San Francisco, USA, 1995; 972–977.
32. Searle JR. Minds, brains, and programs. *The Behavioral and Brain Sciences* 1980; **3**: 353–382.
33. Gorman B, Thurau C, Bauckhage C, Humphrys M. Believability testing and Bayesian imitation in interactive computer games. In *From Animals to Animats 9*, Vol. 4095. Springer, Berlin, Germany, 2006; 655–666.
34. Bossard C, Benard R, De Loor P, Kermarrec G, Tisseau J. An exploratory evaluation of virtual football player's believability. In *Proceedings of the 11th Virtual Reality International Conference (VRIC'09)*, Richir S, Shirai A (eds). IEEE, Laval, France, April 2009; 171–172.

AUTHORS' BIOGRAPHIES



Fabien Tencé received his PhD in computer science in 2011 concerning "Probabilistic Behaviour Model and Imitation Learning Algorithm for Believable Characters in Video Games". Now, he is working at Virtualys.

Laurent Gaubert is an associate professor in mathematics at the Lab-STICC (French Laboratory UMR-CNRS 3192), member of the team IHSEV team at the European Center for Virtual Reality (CERV). In his doctoral thesis, he studied the links between different scales of a variety of coupled dynamical systems: emerging properties, synchronization, biological data individuation... (2007). He is also interested in artificial intelligence algorithms, their automatic learning and their believability properties.

Julien Soler works for Virtualys, a software development company specialized in virtual reality, 3D interactions and web based technologies. He is currently a PhD student at the Lab-STICC and works on the CHAMELEON Project.

Pierre De Loor is a professor at the Lab-STICC. He is the head of the IHSEV team since 2012 and the Head of the CERV. He wrote his doctoral thesis in Automatic, Signal and Software Engineering (1996) and his Accreditation to Direct Research (HDR) in Computer Science (2006). He is interested on the link between artificial intelligence, cognitive science and virtual reality. He is also interested on phenomenology, epistemology and links between art and science.

Cédric Buche is an associate professor in computer science and works at CERV. His research concerns the simulation of adaptive behaviors. He wrote an Accreditation to Direct Research (HDR) concerning the simulation of adaptive behaviors (2011). He is the leader of the CHAMELEON project.