Bayesian Imitation Learning in Game Characters

Christian Thurau, Tobias Paczian and Gerhard Sagerer
Applied Computer Science, Bielefeld University, Germany
Christian Bauckhage
Centre for Vision Research, York University, Canada

Abstract: Imitation learning is a powerful mechanism applied by primates and humans. It allows for a straightforward acquisition of behaviors that, through observation, are known to solve everyday tasks. Recently, a Bayesian formulation has been proposed that provides a mathematical model of imitation learning. In this paper, we apply this framework to the problem of programming believable computer games characters. We will present experiments in imitation learning from the network traffic of multi-player online games. Our results underline that this indeed produces agents that behave more human-like than characters controlled by common game AI techniques.

Keywords: Imitation learning, bayesian learning, behavior modeling, game AI

Reference to this paper should be made as follows: Thurau, C., Paczian, T., Bauckhage, C. and Sagerer, G. (2005) 'Bayesian Imitation Learning in Game Characters'.

Biographical notes: Christian Thurau received the diploma in computer science from Bielefeld University in 2003. He joined the research group for Applied Computer Science at Bielefeld University in 2003, as a Ph.D. student in the graduate program Strategies and Optimisation of Behavior.
Tobias Paczian is currently working towards his Diploma in computer science at Bielefeld University.
Christian Bauckhage received the diploma and Ph.D. degree in computer science from Bielefeld University in 1998 and 2002, respectively. Currently, Christian has joined the Centre for Vision Research at York University, Toronto where he is a research associate in the Active and Attentive Vision Lab.
Gerhard Sagerer is Professor for computer science at Bielefeld University, and head of the research group for Applied Computer Science. His fields of research are image and speech understanding including artificial intelligence techniques and the application of pattern understanding methods to natural science domains.

1 INTRODUCTION

1.1 Imitation Learning in Infants

Imitation learning or learning through imitation is one of the most important mechanisms for the acquisition of new behaviors in humans. Since it avoids time consuming trial and error steps, it provides us with the capability of rapid learning. It therefore comes with little surprise to see that imitation learning lately has gained a lot of interest in the robotics community.

The motivation is obvious: What helps us in mastering our world could also be a benefit for robots. However, although imitation learning proved useful in many robotic applications, the complexity of the tasks that have been considered so far is still rather low. Although the imitation of pressing a button or swinging a tennis racket (see, for example, the work of (Schaal 1999) or (Breazeal & Scassellati 2002)) is indeed a great achievement from a robotics point of view, it does not even come close to what humans are capable of.

Recently, (Rao et al. 2004) proposed “A Bayesian Model of Imitation in Infants and Robots”. They introduce a psychologically inspired probabilistic framework for imitation in infants. The model consists of a formalization of four stages of imitative abilities that have been discovered by psychologists (Rao & Meltzoff 2003). Since a basic understanding of these stages is important for this paper, we will summarize them briefly:

Body babbling: In their early stages of acquiring basic movements, so called movement primitives, infants learn how their actions achieve goal states. These motor primitives can be used as a basis for imitative...
Imitation of body movements: This describes the ability to map between observed actions and one’s own body. Infants are able to execute observed facial expressions without having seen them before.

Imitation of actions on objects: This stage involves imitation of the manipulation of objects that are external to the infants body parts. Examples would include toy or tool handling.

Imitation based on inferring intentions of others:
Finally, imitation learning is not restricted to the imitation of actions but can be applied to goals. This implies inference of intentions and the ability to accomplish them.

Although section 3.2 will give an outlook on the usage of “imitation based on inferring intentions of others” and “imitation of body movements”, in the following we will focus on the stages of “body babbling” and “imitation of actions on objects”. We will describe how we applied and modified the approach by Rao et al. in order to create an artificial agent who imitates human behavior in a virtual environment.

The virtual environment is provided by the commercial computer game QUAKE II®. Assuming a machine learning perspective on behavior programming, computer games have proven to be an excellent platform when it comes to availability of observation data of human behavior. In contrast to robotics scenarios computer games provide noise free measurements of highly complex worlds. For general discussions of the benefits of computer games in AI and ML research see for instance (Bauckhage & Thurau 2004), (Funge 2004) and (Laird 2001).

The work reported in this paper aims at letting an artificial game player learn its behavior solely from observations of human gameplay. QUAKE II® (see Figure 1) represents the popular genre of First-Person-Shooter games. These games provide a demanding yet easy to learn gameplay. Human players control avatars that walk around a (more or less) realistic 3D world. The goal is to earn points by shooting enemy players. Subgoals arise from the game context and are influenced by various important items which can be picked up all around the 3D-world (e.g. armor or health pickups to refill lost energy).

Within the limits of the game physics, all kinds of movements and strategies are allowed. To no surprise, creating truly human-like game characters is a still an open problem – despite all efforts of the multi-billion dollar computer gaming industry in achieving this goal.

In the next section we will outline our imitation learning approach to behavior programming which forms the basis for the experiments we will present afterwards. Finally, the paper will close with a conclusion and an outlook to future directions of research regarding imitation learning in games.

Figure 1: QUAKE II® character in its environment

2 IMITATING HUMANS IN GAMES

2.1 Bayesian Imitative Learning
In (Thurau et al. 2004) we modeled behaviors by a probabilistic inverse model \( P(a_t | s_t) \), where \( s_t \) denotes the game world state at time \( t \) and \( a_t \) denotes the most appropriate action. We used discrete sets of states and actions where conditional probabilities were estimated from recorded matches of human players. Although the resulting, synthesized movements were not explicitly goal oriented, recurring movement sequences could be observed which yielded implicitly goal oriented behaviors. Still, the movements showed a lack of strategical smartness and appeared reasonable only on certain occasions. However, as we did not include knowledge about goal or subgoal states, a real goal-directed behavior could not have been expected.

The psychologically inspired Bayesian approach by Rao et al. (2004) includes dependencies on global goals and subgoals. Consequently it should avoid some of the shortcomings of our initial attempts on probabilistic behavior modeling. The probability for the execution of an action \( A_i \) at timestep \( t \) is given as follows:

\[
P(a_t = A_i | s_t = S_i, s_{t+1} = S_j, s_g = S_h) = \frac{1}{C} P(s_{t+1} = S_j | s_t = S_i, a_t = A_i) P(a_t = A_i | s_t = S_i, s_g = S_h)
\]

where \( S_i \) is the value of the random variable \( s_t \) characterizing the current state, \( S_j \) is the value of the next desired state (or subgoal) \( s_{t+1} \), and \( S_h \) denotes the value of the goal state \( s_g \). The constant \( C \) is given by

\[
C = P(s_{t+1} = S_j | s_t = S_i, s_g = S_h).
\]

It is used as a normalization constant which results from marginalizing over all actions:

\[
C = \sum_m P(s_{t+1} = S_j | s_t = S_i, a_t = A_m) P(a_t = A_m | s_t = S_i, s_g = S_h)
\]
In order to retrieve probabilities for the selection of an action \( a_t \), the next desired subgoal state \( s_{t+1} \) and the global goal state \( s_g \) need to be known. However, in our case there is only one true goal state – winning the game. If state successors are interpreted as subgoals, the model is not being altered but one can eliminate the need for discovering more true goal states. (Indeed, we expect expert human players to act rather deterministic. Most states seem to allow for only one good choice of actions in order to play successful. That choice however is highly player and playing style dependent. Nevertheless, future work might as well include the usage of various, more explicit long term goal states, whereas recent approaches (e.g. (Wright 2004)) might be altered but one can eliminate the need for discovering more prototypes available is a rare but great opportunity in machine learning research. Having an almost unlimited amount of training samples available is a rare but great opportunity in machine learning research. However, it necessitates automatic processing and a careful, manual selection of specific state sequences does not seem feasible (the latter should not be an discouragement; after all, we aim at an fully automatic approach). The next chapter describes our approach in recreating these observed state sequences.

2.2 State Sequences

State vectors \( s_t \) should contain a suitable representation of the game's world state at timestep \( t \). For instance, in using imitation learning to solve classical maze problems (i.e. finding goal directed paths in a maze world) the position vector \( p_t = [x_t, y_t, z_t]^T \) of the observed player contains all necessary information.

Dependent on the current state of the player, the next state can be estimated from analyzing observed sequences of human players and determining how they acted in the game world.

In order to retrieve a discrete approximations of the state space, we apply Neural Gas clustering as introduced by (Martinetz et al. 1993) to recorded state vectors. Neural Gas clustering shows very good results when it comes to the approximation of topological structures (Martinetz & Schulten 1991). Since we can expect certain topologies within state space (e.g. the positions a player held during a match are a good representation of the topological structure of 3D game worlds), choosing this approach appears reasonable.

Clustering yields a set of prototypical state vectors \( \{s_1, s_2, \ldots, s_n\} \), where good choices for the number \( n \) of prototypes depend on the state space under consideration (see also section 2.4 for chosen cluster sizes in the carried out experiments).

To provide further contextual structure, a graph is constructed based on observed state sequences. Directed edges between state prototypes indicate the possibility for a state transition. They are drawn based on state transitions observed in the training data. Given this transition graph, the estimation of successors to a state at timestep \( t \) only considers nodes that are connected to the current state. The selection of the successor state is based on transition probabilities. The necessary state transition probabilities are recovered from the observed traversing frequency between connected states.

The transition graph and its transition probabilities are inspired by edge reinforced random walks known from statistical mathematics (see also Figure 2). Therefore, we use the term weights when referring to state transition probabilities. We end up with a directed, weighted graph \( G = (V, E, w) \), where \( V \) and \( E \) are the sets of vertices and edges of \( G \), and \( w : E \rightarrow \mathbb{R}_{>0} \) is its weight function. Each vertex in \( V \) corresponds to a clustered state prototype \( s_i \in \{s_1, \ldots, s_n\} \). Edges in \( E \) represent observed state sequences, where

\[
E(vu) = \begin{cases} 
1 & : s_i = u, s_{i+1} = v \\
0 & : \text{otherwise}
\end{cases}
\]

The weights for an edge \( vu \) connecting vertices \( u \) and \( v \) is determined by the number of times \( c_t(vu) \), a human player could been observed traversing the edge, e.g. how often did the human player move from one world position to a neighboring position. Consequently, \( w(vu) = c_t(vu) \).

Conditional probabilities for successor states of a state \( v_t \) are determined as follows:

\[
P(v_{t+1} = u|v_t) = \begin{cases} 
w(vu) & : \text{for } E(vu) = 1 \\
0 & : \text{otherwise}
\end{cases}
\]
2.3 Movement Primitives

The created structure implies certain advantages in comparison to non-interconnected states prototypes. For instance, limiting the choice for a successor of the current state to connected state-nodes only, considerably lowers the computation time while still allows for a reconstruction of all observed state sequences.

However, this does not mean that in principal not all states can be entered from the current state. This decision is solely dependent on the game engine and is out of reach for the artificial agent – to make the game fair, the game world alone is responsible for deciding about player states (usually the game world tends to be rather deterministic, though, with teleporters producing an instant position change being an exception). But it does mean, that there are states which might be out of reach for the game agent, just because no human player was observed performing the state transition.

Given the transition graph, we are able to select the next desired state based solely on the current state. This can be done via a roulette wheel or maximum a-posteriori selection over the conditional probabilities \( P(s_{t+1} = u|s_t) \). This leaves us with the problem of creating an adequate discrete set of motor or movement primitives and with generating a forward model \( P(S_{t+1}|s_t, a_t) \) based on state dependent primitive execution.

2.4 Action Sequences

A human player controls the game character by means of input devices such as mouse and keyboard. Naturally, this implies a certain limitation for the sequencing of actions; for instance, it is impossible for a human player to turn instantly. Due to the input modalities, the previous action effectively restricts possible successor actions. As differences between actions that are generated using mouse and keyboard are rather small, the motion of a human controlled player usually appears very smooth.

To recreate similar motions and to reflect these natural limitations, we introduced conditional probabilities \( P(a_t|a_{t-1}) \) for the execution of movement primitives at time \( t \) which can be learned from the observations of a human player and can be incorporated into the approach by weighting primitive selection probabilities (Thurau et al. 2004).

Since the action vector that was executed last can be seen as part of the world state vector, the prior probability (see also Equation (1)) for \( a_t \)

\[
P(a_t = A_t|s_t = S_t, s_g = S_k)
\]

can be modified by introducing \( a_{t-1} \) and assuming independence of \( a_{t-1} \) and \( s_t, s_g \) (after all, physical limitations
should not affect environmental conditions). This results in a slightly changed prior probability:

\[ P(a_t = A_i | s_t = S_i, s_g = S_k, a_{t-1} = A_j) = \frac{P(a_t = A_i | s_t = S_i, s_g = S_k)P(a_t = A_i | a_{t-1} = A_j)}{P(a_t = A_i)} \]  

The normalization constant (taken from Equation (1))

\[
C = P(s_{t+1} = S_j | s_t = S_i, s_g = S_k) = \sum_m P(s_{t+1} = S_j | s_t = S_i, a_t = A_m) P(a_t = A_m | s_t = S_i, s_g = S_k)
\]

has to be adapted to the new variables \(a_{t-1}\):

\[
C = P(s_{t+1} = S_j | s_t = S_i, s_g = S_k) = \sum_m P(s_{t+1} = S_j | s_t = S_i, a_t = A_m) P(a_t = A_m | s_t = S_i, a_{t-1} = A_j)
\]

Finally, the conditional probability \(P(a_t = A_i | s_t = S_i, s_{t+1} = S_j, s_g = S_k, a_{t-1} = A_j)\) for executing the movement primitive \(A_i\) can be written as follows:

\[
P(a_t = A_i | s_t = S_i, s_{t+1} = S_j, s_g = S_k, a_{t-1} = A_j) = \frac{1}{C} P(s_{t+1} = S_j | s_t = S_i, a_t = A_i) P(a_t = A_i | a_{t-1} = A_j)
\]  

Equation (4) can now be used for calculating movement primitive probabilities, the required conditional probabilities can be computed right away from observation data.

To illustrate the basic concepts and the usability of the approach for the game domain, we carried out several smaller experiments. In each experiment, recordings of human players were taken as training data. The experimental goal was that all observable behaviors should be recreated or synthesized as believable as possible.

\section*{3 EXPERIMENTS}

\subsection*{3.1 Maze Problem}

The first experiment was mostly intended for testing basic functionality of the approach. A player was recorded several times while performing goal directed movements, every recorded move should end at the same map position. State vectors consisted of the observed player positions \(x, y, z\).

From the recorded matches state vector prototypes and transition probabilities were extracted. Building up a graph model using the player positions made sure, that Recreation of the movements was done, by connecting the agent to a game server, and selecting subgoal state prototypes based on the graph model. Knowledge about subgoal state prototypes completed the information for equation (4), thus movement primitive probabilities could be computed.

However, since the agent moves on a discrete set of prototypical positions, the number of clustered position prototypes plays an important role (smaller game worlds require 50-100 prototypes, to allow for collision free navigation).

The approach worked as expected and the movements could be recreated by the artificial player.

\subsection*{3.2 Extended Maze Problem}

The second experiment consisted of an “extended“ maze problem (similar to experiments described in (Rao et al. 2004)). A schematic description of this experiment can be seen in Figure 4. A human player was recorded several times; in each recording he first picked up goal-item 1 (a better weapon) and then continued to goal-item 2 (an armor item).

We extended the used state space, to also contain information about the player’s inventory, in addition to the \(x, y, z\) coordinates. Consequently we extended the state vector by values for holding the new weapon, and the internal armor value.

When the state graph is computed under these preconditions, we expect to observe certain paths which correspond to the order of the item pickups. In order to reach a state with higher armor values, the agent first has to go to the armor item. Since the state prototype with an increase in the armor value is on his “way” (paths in state space do not necessarily correspond to paths in 3D), he should consequently go to that item position on the map.

Here it becomes obvious why we decided for a rather abstract model. The actions of the player can have all sorts of influence to the state space but most actions do not necessarily alter 3D world positions. E.g. earning a point could be considered a change to state space which occurs after having an enemy player right in front and pressing
the fire button.

Again, the number of state prototypes is crucial. We found a number 150-250 prototypes sufficient for our experiments (using a 5 dimensional state space and a mid-sized game-map). However, the number of state prototypes is map and task dependent.

In detail we carried out a set of 20 test runs, in which the agent should reproduce state dependent item pickups. The training set consisted of approx. 3000 sample vectors, showing a human player picking up first the rail gun item, then the armor item and finally ending at a fixed map position. We used 250 unique state prototypes and a number of 175 movement/action primitives. In 17 test runs, the artificial player managed to reproduce the observed behavior without any problems. In the 3 failed test runs, the agent tended to get stuck or to run into a wall. We thus had to manually correct the agent to make him reach the goals. Occasionally some movements were not executed as precisely as needed, therefore the agent fell down of ledges or did some other irritating movement. This however only lead to problems when the agent entered completely unknown states not encountered in the training sample set. Still, with the use of more training samples and the availability of more information on how to handle these unknown states, we can expect even better results.

All in all, the observed behaviors were imitated convincingly. The artificial player managed to reach the goals in the predefined order. Besides, the movements appeared smooth and showed characteristics mostly seen in human players, e.g. the artificial player “slided” around corners, preserving observed player habits.

However, it is very hard to judge whether the artificial agent appears truly human-like or not (or to what degree its appearance is human-like). In contrast to similar projects (e.g. the Drivatar – Driving Avatar racing game AI¹) FPS games are arguably more complex, after all a number of different behaviors (movement, shooting, tactical reasoning . . . ) have to be integrated. Therefore, and although first results are promising, a fully functional imitating artificial player remains a challenging task.

At this stage, complete matches against the presented artificial player are not possible. Accordingly, we can decide about the quality of the agent solely based on non interactive observations. Since interactivity is one of the key aspects of believable game agents, for the time being, we avoided believability measures. Still, from comparing recordings of the agent’s movements to the human player’s movement, we can clearly see many similarities in the appearance and reproduction of actions (after all the agent is solely based on observation data recorded from a human player, with as little expert knowledge as possible introduced into the imitation learning model).

4 CONCLUSION

Imitation learning is a powerful, yet seldom applied mechanism when it comes to behavior acquisition in robots or virtual agents. We applied and extended a recently proposed Bayesian model of imitation learning to the task of behavior acquisition in artificial game characters.

By means of the example of the commercially available computer game QUAKE II®, we could show, that human behaviors can be imitated by artificial players using the psychologically motivated Bayesian framework. We established a state and goal/subgoal dependent recreation of behaviors. Synthesis of useful state sequences was done using a state graph that was inspired by aspects of edge reinforced random walks. Human-like behaviors of artificial game agents resulted from sequenining basic movement primitives dependent on the world state and subgoal states. In addition, we could create smooth, i.e. realistic, motions by incorporating temporal context via an action-history dependent movement primitive selection.

Initial experiments showed very promising results in imitating goal directed strategic movement paths.

Although we did not yet realize a fully functional artificial agent, we already dare to conclude that doing behavior programming the imitation learning way makes game agent behavior acquisition less demanding and ultimately leads to more human-like artificial agents. Not only were the recreated behaviors a correct imitation of reaching certain goals, they also appeared plausible and natural.

5 FUTURE WORK

Since the used state spaces do not contain all aspects of the game world, higher dimensional state spaces will have to be considered. In order to handle higher dimensional state spaces, dimensionally reduction methods, as for instance Locally Linear Embedding (Roweis & Saul 2000), will have to be applied.

Moreover, imitation learning could and should be done in an online manner. This would imply a mapping from observed actions onto the game agent’s own “body” – since not all state vector features of another player are known, this is not an easy task (“imitation of body movements”). Finally, the anticipation of another player’s movement and goals could be realized in a similar way – the ideas of inferring intentions of others are a crucial part in imitation learning (“imitation based on inferring intentions of other”).

6 ACKNOWLEDGEMENTS

This work was supported by the German Research Foundation (DFG) within the graduate program “Strategies & Optimization of Behavior”.

¹Some information about this project can be found at http://research.microsoft.com/mlp/Forza/default.htm. Unfortunately detailed informations are not yet available.
REFERENCES


