Is Bayesian Imitation Learning the Route to Believable Gamebots?

Christian Thurau, Tobias Paczian
Applied Computer Science
Bielefeld University
P.O. Box 100131, 33501 Bielefeld, Germany
{cthurau,tpaczian}@techfak.uni-bielefeld.de

Christian Bauckhage
Centre for Vision Research
York University
4700 Keele St, Toronto, ON, M3J 1P3, Canada
bauckhag@cs.yorku.ca

ABSTRACT

As it strives to imitate observably successful actions, imitation learning allows for a quick acquisition of proven behaviors. Recent work from psychology and robotics suggests that Bayesian probability theory provides a mathematical framework for imitation learning. In this paper, we investigate the use of Bayesian imitation learning in realizing more life-like computer game characters. Following our general strategy of analyzing the network traffic of multi-player online games, we will present experiments in automatic imitation of behaviors contained in human generated data. Our results show that the Bayesian framework indeed leads to game agent behavior that appears very much human-like.

Introduction

Statistical machine learning is gaining more and more attention as a means of enhancing game AI. While the majority of contributions considers supervised learning or reinforcement learning, the paradigm of *imitation learning* has drawn little attention so far. However, learning through imitation is a powerful mechanism for the acquisition of behaviors. Since it avoids tedious and futile trial and error strategies and does not require labeled training data, imitation might be a short cut on the route to more engaging artificial game agents.

In a recent contribution, Rao, Shon & Meltzoff (2004) introduced a Bayesian model of imitation learning for applications in robotics. Based on experiments in developmental psychology, their probabilistic framework models four stages of imitative abilities that were observed in infant behavior. Since a basic understanding of these stages helps grasping the approach discussed below, we shall summarize them briefly: By means of postnatal *body babbling* infants acquire a model of their body. They learn which muscular actions lead to what kind of limb configurations and thus acquire a vocabulary of useful *motor primitives*. This enables the *imitation of body movements* where infants map observed actions onto their own body. At the age of several weeks, for instance, they can mimic facial expressions they have never seen before. In a third stage, infants start *imitating actions on physical objects* such as toys which are external to their body. By the time they are 1.5 years old, infants are experienced in interacting with other humans. Consequently, they can acquire models of agents with intentions. Forward models alow them to *infer the goals of an agent* even if they only observe unsuccessful demonstrations; inverse models are used to select motor commands that will achieve undemonstrated but inferred goals.

In this paper, we will describe how similar mechanisms embedded in a Bayesian framework can produce more life-like computer game agents (henceforth called *gamebots*). Using the game QUAKE II® as a platform for our research, we investigate behavior acquisition from imitating how human players steer their avatars through 3D game worlds (often called *maps*). Since QUAKE II® represents the arguably most popular genre of First-Person-Shooter games, a player's overall goal is to score as many points as possible by shooting enemy players. Situation dependent behaviors and subgoals arise from the current game context. Factors that influence context dependent behavior are the internal state of a player's character, the behavior of opponents, as well as various items placed all over the map. Low avatar energy, for instance, might be compensated by picking up armor or health packages.

Within the limits imposed by the game physics, games like QUAKE II® allow for all kinds of movements, strategies and cunning. As this kind of *intelligent* gameplay requires some sort of cognitive capabilities, it comes with little surprise that gamebots which would behave truly human-like are still not available. In the next section, we will discuss this problem in more depth and shall roughly survey related work. Afterwards, we will discuss our approach to Baysian imitation learning of human-like behavior. As we shall see, our contribution is fourfold: (i) we apply concepts from the theory of edge reinforced random walks to impose an adequate topology on the space of internal states of game agents; (ii) we make use of clustering methods from statistical machine learning to acquire a vocabulary of action primitives for game agents; (iii) we extend the model of Rao et al. (2004) in order to guarantee the temporal coherency of the actions selected for game characters; (iv) we integrate these three techniques within a Bayesian learning framework. In the fourth section, we will present and discuss first experimental results which underline that this framework indeed provides an auspicious avenue to believable gamebots. Finally, a conclusion will close this contribution.

In Proc. GAME-ON North America, pages 3–9, Montreal, Canada, August 2005

Related Work

It is a widely accepted notion these days that life-like behaving gamebots are the next big step towards ever more exciting game experience. This holds even in the age of online gaming where human players engage each other in virtual battle over the Internet. The nature of popular genres such as massively multiplayer online role playing games (MMORPGs) simply calls for non-player characters to set forth the action.

However, when it comes to gamebot control, the gaming industry still mostly relies on seasoned deliberative AI techniques like finite state machines or the A* algorithm. While A*-search was introduced almost four decades ago (Hart, Nilsson & Raphael 1968), output generating finite state machines date back even further (Mealy 1955, Moore 1956). Of course, the problem with these techniques is not their maturity but rather their lack of life-likeness. Once human players get used to a game, common gamebots are perceived to miss the element of surprise human opponents would provide; they appear to behave 'dumb' (Cass 2002).

The picture could be different, if gamebots were to learn from experienced human players. Work in this direction was first reported by Sklar, Blair, Funes & Pollack (1999) who collected the key-strokes of people playing *Tron* in order to train neural network based game agents. Neural networks also proved to perform satisfiable in First-Person-Shooter games. Trained on the data contained in the network traffic of multiplayer games, neural architectures learned different aspects of engaging gameplay. They were shown to reproduce convincing *reactive* behavior (Bauckhage, Thurau & Sagerer 2003, Thurau, Bauckhage & Sagerer 2003) as well as *tactical* decisions such as context aware weapon selection (Bauckhage & Thurau 2004).

Other machine learning techniques have been applied as well: Spronck, Sprinkhuizen-Kuyper & Postma (2003) incorporate reinforcement learning into rule selection for agent behavior in a role playing game and Le Hy, Arrigioni, Bessière & Lebeltel (2004) describe action selection for a commercial game using Baysian networks which were trained by means of human generated input.

Next, we will present a different approach which underlines that –given suitably preprocessed data– even simple Bayesian statistics provides a powerful tool for game AI programming.

Bayesian Imitation Learning

In earlier work (Thurau, Bauckhage & Sagerer 2004), we realized movement behaviors for QUAKE II® gamebots using a straightforward, probabilistic model $P(a_t|s_t)$. The values of the random variable s_t are given by vectors that encode the state of an agent and its surroundings at time t. The values of the random variable a_t denote the most appropriate (re)action. The discrete sets of possible state vectors $S = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_M\}$ and action vectors $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_P\}$ result from clustering the data contained in recordings of matches played by human players. Although goal orientation was not explicitly modeled in this approach, it yielded movement sequences that human players identified as goal driven behavior. Still, the movements lacked strategic sophistication and were convincing only in certain constrained situations.

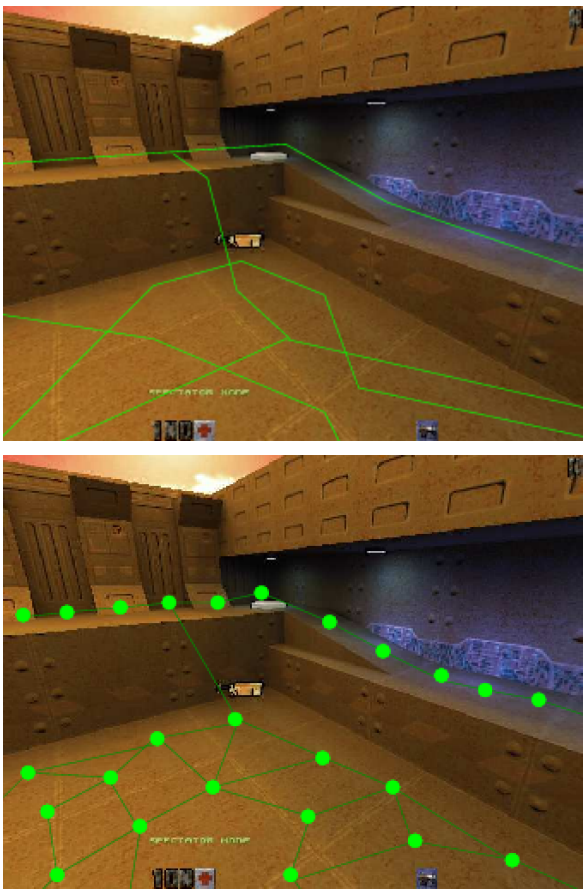

Figure 1: If applied to data representing the paths a player did run during a match, Neural Gas clustering will result in a structured set of waypoints which captures the topology of the corresponding map.

The Bayesian model of imitation learning proposed by Rao et al. (2004) also accounts for goals and subgoals. Consequently it can be expected to avoid the shortcomings of our initial attempts on probabilistic behavior modeling. Within this Bayesian framework, the probability for the execution of an action $\mathbf{a}_i$ at time step t depends on the current state s_t, the next desired state (or subgoal) s_{t+1} and the overall goal state s_g. Using Bayes' theorem, it amounts to

$$P(a_t = \mathbf{a}_i | s_t = \mathbf{s}_i, s_{t+1} = \mathbf{s}_j, s_g = \mathbf{s}_k) = \frac{1}{C} P(s_{t+1} = \mathbf{s}_j | s_t = \mathbf{s}_i, a_t = \mathbf{a}_i)\, P(a_t = \mathbf{a}_i | s_t = \mathbf{s}_i, s_g = \mathbf{s}_k)$$

where the normalization constant C results from marginalizing over all possible actions:

$$C = P(s_{t+1} = \mathbf{s}_j | s_t = \mathbf{s}_i, s_g = \mathbf{s}_k) = \sum_m P(s_{t+1} = \mathbf{s}_j | s_t = \mathbf{s}_i, a_t = \mathbf{a}_m) P(a_t = \mathbf{a}_m | s_t = \mathbf{s}_i, s_g = \mathbf{s}_k).$$

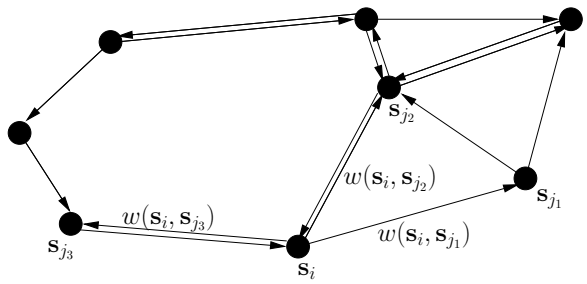

Figure 2: Didactic example of a state transition graph.

In order to obtain reasonable probabilities for the selection of an action a_t, the desired subgoal s_{t+1} and the global goal s_g need to be known. Note, however, that in the game domain winning the game is the only true global goal. Also note that interpreting successive states as subgoals does not alter the model but eliminates the need of mining for subgoals in the training data. Indeed, experienced human players act reasonable and with implicit subgoals in mind. Therefore, reasonable goal states will emerge from clustering the network traffic of recorded matches and automatically provide a game agent with suitable choices[1].

A Structured State Space

State vectors $\mathbf{s}_i$ should contain a problem specific description of a gamebot's internal state and the surrounding gameworld. For instance, if imitation learning is applied to simple classical maze problems (i.e. finding goal directed paths in a maze world), recording a player's position vectors $\mathbf{p}_t = [x_t, y_t, z_t]^T, t = t_1, \ldots, t_{\text{end}}$ provides all necessary information.

In order to derive a discrete approximation of the state space, we cluster the state vectors recorded from the network traffic using Neural Gas clustering, a technique introduced by Martinetz, Berkovich & Schulten (1993). Since we may expect to find certain topologies within the state space (see Fig. 1), Neural Gas clustering is well suited for our problem because it is known to yield superior results in recovering topological structures of a dataset.

Knowledge of state space structure may facilitate computing some of the probabilities required in the Bayesian framework. In order to provide the resulting discrete state space with a useful structure, we therefore compute a state transition graph. In this graph, a directed edge between two state prototypes indicates a possible state transition. Moreover, edges are labeled with transition counts (see Fig. 2). As the idea for the transition graph was inspired by the theory of *edge reinforced random walks* known from statistics (Diaconis 1988), we use the term weights when referring to state transition counts. Formally, the transition graph is thus given as a triple $G = (V, E, w)$ where $V = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_M\}$ is a set of vertices, $E \subseteq V \times V$ is a set of directed edges and $w : E \rightarrow \mathbb{R}^+$ is a function assigning weights to the edges. Edges are drawn based on state transitions observed in the training data; i.e. we have

$$(\mathbf{s}_i, \mathbf{s}_j) \in E \iff \exists s_t, s_{t+1} : s_t = \mathbf{s}_i \wedge s_{t+1} = \mathbf{s}_j \qquad (1)$$

Transition counts are recovered from traversal frequencies in the training data; the weight w of an edge $(\mathbf{s}_i, \mathbf{s}_j)$ corresponds to the number of times a human players was observed to move from state space position $\mathbf{s}_i$ to position $\mathbf{s}_j$.

Given this weighting scheme, suitable upcoming (sub)goals for Bayesian behavior synthesis can be determined either from a roulette wheel or a maximum a-posteriori selection over the state transition probabilities

$$P(s_{t+1} = \mathbf{s}_j | s_t = \mathbf{s}_i) = \frac{w(\mathbf{s}_i, \mathbf{s}_j)}{\sum_k w(\mathbf{s}_i, \mathbf{s}_k)} \qquad (2)$$

Imposing a discrete state space structure like this provides certain advantages over an unstructured collection of prototypes. For instance, limiting the choice for a successor to those states that are connected to the current state considerably lowers the computation time but still allows for a reconstruction of all observed state sequences[2]. What is left is determining an adequate discrete set of motor or movement primitives $\mathbf{a}_t$ to generate an action dependent forward model $P(s_{t+1} | s_t, a_t)$.

Finding Movement Primitives

From the point of view of a gamebot, the only way to traverse the state transition graph lies in choosing appropriate actions. For instance, if the next desired state would correspond to a position direct in front of the gamebot, it simply had to move forward in order to reach that state.

The set of available actions is given by *movement primitives*, prototypical actions, which are extracted from recorded matches. For the experiments, presented below, a movement primitive is a 5 dimensional vector

$$\mathbf{a} = \begin{bmatrix} \text{yaw angle} \\ \text{pith angle} \\ \text{forward velocity} \\ \text{sideward velocity} \\ \text{upward velocity} \end{bmatrix}$$

Prototypic movement primitives result form applying k-means clustering to the given movements of human players. We found a number of $k \in [150, \ldots, 200]$ movement primitives sufficient for synthesizing smooth looking motions.

In the context of Bayesian behavior learning, we are interested in the effects of movement primitives on state transitions. This will provide a forward model for choosing movements in order to reach specific subgoal states. However,

[1] The resulting (sub)goals are of course highly player and playing style dependent.

[2] Obviously, some state transitions might become impossible for the game agent just because no human player was observed performing the state traversal. However, this is what human-like behavior is all about.

not all actions are reasonable in any game context. Sometimes, an otherwise frequently used movement could lead into a wall or down a ledge. The forward model for action selection thus will have to depend on the current state.

Following the proposal of Rao et al. (2004), we apply the mechanism of *body-babbling* to estimate a forward model $P(s_{t+1}|a_t, s_t)$. Again, we make use of the widely available demo data.

Synthesizing Action Sequences

Human players control their avatars using mouse and keyboard. This implicitly limits their choice of actions. In QUAKE II®, for example, instant turns are impossible for a human player. As the input modalities thus constrain the sequencing of actions and as differences between movement primitives which are generated using mouse and keyboard are rather small, motions of human controlled avatars usually appear to be smooth.

In order to recreate such natural motions, we introduce conditional probabilities $P(a_t|a_{t-1})$ for the execution of movement primitives at time t. Again, these can be learned from observing human players. Since the action vector that was executed last can be seen as part of the world state vector, we extend the model of Rao et al. (2004) by introducing a_{t-1} into the equations. Assuming independence of a_{t-1}, s_t and s_g (after all, physical limitations should not affect environmental conditions), this results in the following model:

$$P(a_t = \mathbf{a}_i | s_t = \mathbf{s}_i, s_g = \mathbf{s}_k, a_{t-1} = \mathbf{a}_j) = \frac{1}{C} \frac{P(a_t = \mathbf{a}_i | s_t = \mathbf{s}_i, s_g = \mathbf{s}_k) P(a_t = \mathbf{a}_i | a_{t-1} = \mathbf{a}_j)}{P(a_t = \mathbf{a}_i)}$$

where the normalization constant C has to be adapted to the new variables a_{t-1}:

$$\begin{aligned} C =& P(s_{t+1} = \mathbf{s}_j | s_t = \mathbf{s}_i, s_g = \mathbf{s}_k) \\ =& \sum_m P(s_{t+1} = \mathbf{s}_j | s_t = \mathbf{s}_i, a_t = \mathbf{a}_m) \\ & P(a_t = \mathbf{a}_m | s_t = \mathbf{s}_i, s_g = \mathbf{s}_k, a_{t-1} = \mathbf{a}_j) \\ =& \sum_m P(s_{t+1} = \mathbf{s}_j | s_t = \mathbf{s}_i, a_t = \mathbf{a}_m) \\ & \frac{P(a_t = \mathbf{a}_m | s_t = \mathbf{s}_i, s_g = \mathbf{s}_k) P(a_t = \mathbf{a}_m | a_{t-1} = \mathbf{a}_j)}{P(a_t = \mathbf{a}_i)} \end{aligned}$$

Finally, given a current game state and a desired goal state, the conditional probability for the execution of a movement primitive $\mathbf{a}_i$ can be written as follows:

$$\begin{aligned} P(a_t =& \mathbf{a}_i | s_t = \mathbf{s}_i, s_{t+1} = \mathbf{s}_j, s_g = \mathbf{s}_k, a_{t-1} = \mathbf{a}_k) \\ =& \frac{1}{C} P(s_{t+1} = \mathbf{s}_j | s_t = \mathbf{s}_i, a_t = \mathbf{a}_i) \\ & \frac{P(a_t = \mathbf{a}_i | s_t = \mathbf{s}_i, s_g = \mathbf{s}_k) P(a_t = \mathbf{a}_i | a_{t-1} = \mathbf{a}_j)}{P(a_t = \mathbf{a}_i)} \end{aligned}$$

Next, we will present experiments which indicate that this Bayesian scheme for goal oriented action selection indeed results in gamebot behavior that appears to be human-like.

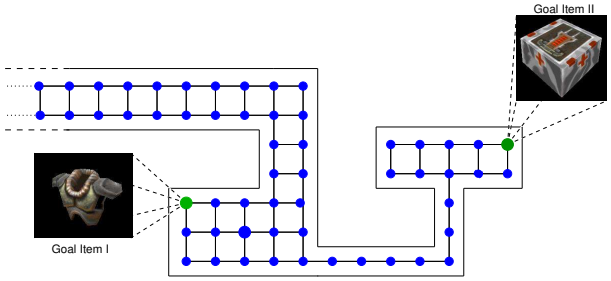

Figure 3: Schematic representation of an extended simple maze task. Two points in the waypoint map that results from clustering recorded player movements correspond to the locations of desirable items on the map. In the our experimental setting, a subgoal of the game agent is to increase its amour value. It therefore has to devise a path trough the abstract higher dimensional state space that accords with a path through the 3D waypoint map leading to the armor item.

Experiments

In order to test the Bayesian framework and its usefulness for the game domain, we carried out a series of experiments in motion planning. The basis of all experiments was of course formed by recordings of data generated by human players. The experimental goal was to reproduced all observable movement behaviors.

Maze Problem

The first experiment basically tested the functionality of the approach. Several examples of a goal directed movement sequence were recorded; in each sequence the player's motion ended at the same map position. Due to the simplicity of the task, the state vectors we considered here only contained observed player positions x, y, z.

After connecting the agent to a game server, it was supposed to reproduce human-like movements by selecting subgoals based on the probabilities encoded in a graph model learned from the training data.

Since the agent traverses a discrete lattice of prototypical positions, the number of clustered state prototypes plays a pivotal role. Our experiments revealed that smaller maps require between 50 and 100 prototypes to allow for collision free navigation. Given this, the Baysian approach performed as expected and believable human-like movements were recreated.

Extended Maze Problem

Our second experiment considered an extended maze problem; Fig. 3 sketches its setting. The training data we considered here displayed several instances of a human player first picking up goal-item 1 (an armor) and then continuing to goal-item 2 (a better weapon).

The state space dimensionality was extended to account for the player's inventory. The additional dimensions encode

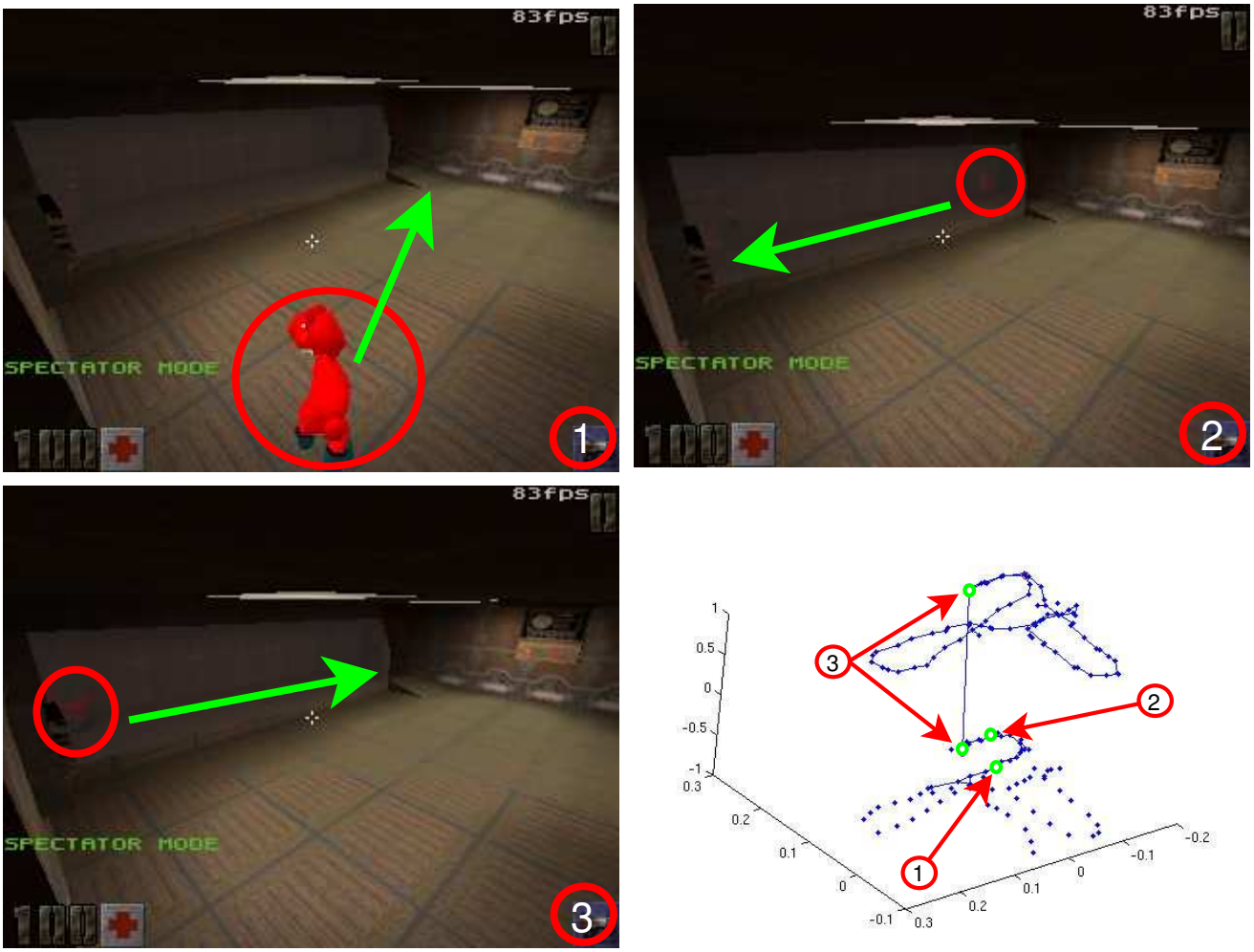

Figure 4: Screenshots showing an artificial player solving an item pickup task and visualization of the corresponding transitions in state space. The screenshots display the agent's actions in the game world; the plot in the lower right corner shows what is going on in a subspace of the state space considered in this experiment: the X and Y axes denote the gamebot's (x,y) positions, while the Z axis represents inventory item information. Movements in the 3D gameworld and resulting changes of the agent's internal state correspond to movements between nodes of the state graph. First, in (1), the agent is seen moving along the graph closer to a node whose (x,y) coordinates coincide with a goal item. In (2), the agent is strafing around the corner of a transparent wall. Finally, as seen in screenshot (3), it reaches the item and continues its way to the next item. The item pickup in (3) considerably increases the inventory value for this item and thus results in a 'jump' along the Z axis of the subspace shown here. Although, in visualizations like this, such state space discontinuities appear random to the human eye, they are not. In fact, the screenshots in this figure show but a part of a longer sequence of actions. At the beginning of this sequence, the agent determined its next suitable subgoal represented by a state s_{t+1} and the item pickup in this figure is actually a planned action to reach this graph node s_{t+1}.

information about the internal armor value and the weapon currently hold.

Given state graphs computed under these conditions, we expect to observe state sequences which reflect the order of the item pickups. In order to reach a state with higher armor values, the agent has to obtain the armor item. Since in the demo data the armor pickup precedes the weapon pickup, states of higher armor value must lie on the way through state space that leads to the overall goal state. On the map, the agent therefor has to visit the location of the armor prior to continuing to the final destination (note that state space paths must not be confused with paths in the 3D gameworld). Figure 4 shows an example of a trained gamebot trying to accomplish this task and a three dimensional projection of the corresponding state graph transitions.

Again, the number of state prototypes is crucial. Using a 5 dimensional state space and a mid-sized game-map, we found a number of 150-200 prototypes sufficient for our purpose. All in all, the observed behaviors were imitated very convincingly (see the examples of state space trajectories in

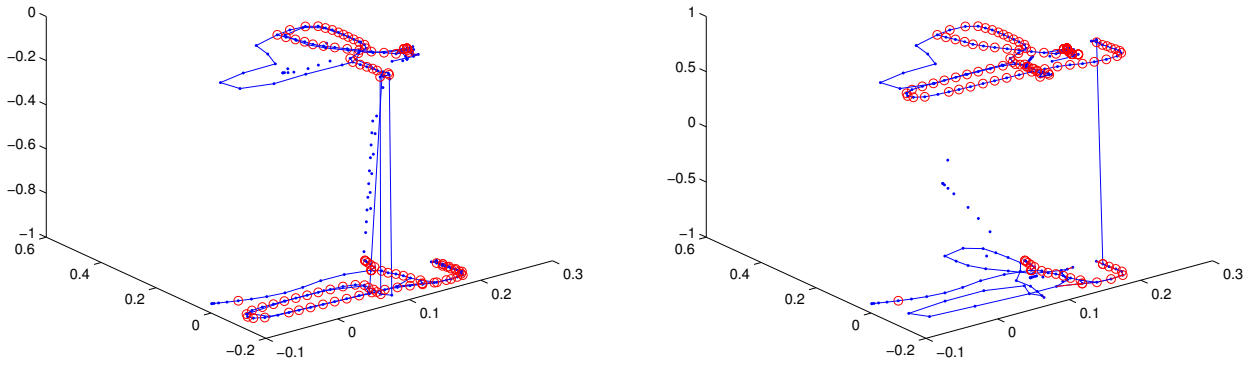

Figure 5: Examples of synthesized movements (plotted in state space coordinates). Again, the X and Y axes correspond to *x* and *y* positions of the agent whereas the Z axis denotes inventory armor values (left figure) and inventory rail gun values (right figure). The blue dots correspond to sequences of state vectors generated by human player while 'solving' the task of item cycling considered in our second experiment. Red circles indicate state prototypes reached by the artificial player.

Fig. 5). The agent managed to reach both goals in the predefined order. Additionally, the movements appeared to be smooth and showed characteristics mostly seen in human players. For instance, the agent "slides" around corners thus preserving observed player habits.

Discussion

Generating action primitives by means of training data clustering and estimating the consequences of their execution in terms of state space transitions can effectively reproduce various sorts of movement behavior observed in humans. This even holds for more complex movements such as swimming or the famous rocket-jump[3] (like simple movements, these activities are as good as any other action and might be necessary to achieve certain goals). The convincing reproduction of single complex movements already leads to a very lifelike appearance, because such movements naturally appear as if generated by human players. However, in conjunction with probabilistic action selection based on goal states this impression is further strengthened: longer sequences of individual human-like movements resulting from the Bayesian approach convey a strong impression of intelligently planned and purposeful behavior.

Of course, the number of state prototypes necessary to produce smooth, humanlike movements is map- and task-dependent. While for the simple maze problem a set of 50 to 100 state vectors was sufficient to create convincing behavior (trials with a lesser number failed to do so; adding more prototypes did not further improve the appearance of the movements), solving the extended maze problem required 150 to 200 state vectors. This raises the question, if the presented approach will scale to more complex behaviors and more demanding contexts? Up to a certain level of complexity it is in fact reasonable to assume it will: even if the dimension of state space increases, the state transition graph ensures that action selection will be based upon *local evidence*. The graph structure ensures that, even if there is a growing number of states, not all of them will have to be considered in the necessary computations. As this locality constraint is thus desirable, future work should further explore techniques for manifold learning and state space structuring. For a full blown state space dimensionality (which is inevitable for a full featured artificial game agent), however, a single state transition graph graph might not be sufficient any more. Especially the interplay of strategic, tactical and reactive behavior may not be sufficiently captured by means of a monolithic graph. A separation into several graphs each targeting the emulation of different behaviors may overcome this problem. However, while the approach presented in this paper is solely based on automatic data analysis and learning, a technique combining several state spaces will surely require expert knowledge to be introduced into the model selection mechanism. Again, this remains as a topic for further research.

Now, if state space manifold identification appears to be the predominant factor for a satisfying performance of our framework, then why, after all, consider Bayesian imitation learning? The answer is clear: without identifying a vocabulary of action primitives and recovering probabilities for their use in different contexts, traversing the state graph would reduce to a deterministic process similar to the ones produced by finite state machine. State transition based on Bayesian probability theory introduce flexibility and surprise. Compared to the usual Baysian techniques found in game AI programming, however, imitation learning as presented here has two noticeable advantages: first, incorporating (sub)goals

[3]An expert player's move, where a rocket is fired on the ground while the player jumps. It results in a much higher altitudes than reachable by ordinary jump movements.

into the process of action selection produces unpredictable short term but nevertheless goal driven long term behavior. Second, where common Baysian approaches to game AI rely on preprogrammed sets of behaviors whose a-priori probabilities are predetermined by programmers, clustering of network data provides a natural set of action primitives. Given these, *body babbling*, i.e. a training phase of random invocation of these actions in different states to learn about their effects, provides suitable estimates for the priors in life-like behavior generation.

Conclusion

Imitation learning is a powerful yet so far underexploited mechanism for behavior acquisition for game agents. In this paper, we applied a Bayesian formulation of imitation learning to game AI programming.

By means of the example of the game QUAKE II®, we could show that artificial players can convincingly imitate human movement behavior. Our results are based on a state and (sub)goal dependent model. Useful state sequences are synthesized using a transition graph that structures a codebook of state vectors extracted from the network traffic of the game. Incorporating temporal context into the probabilistic action selection mechanism leads to especially smooth and realistic movements.

Given these results, it seems that imitation provides an auspicious approach to game AI programming. Currently, we are extending the technique presented in this paper to more complex behaviors. Although the focus of the presented paper is on the reproduction of strategical, i.e. state dependent, item pickups, first experiments applying the presented approach to more tactical (movements relative to an enemy player) and reactive (close combat and shooting) behaviors are work in progress. As first results are encouraging, Bayesian imitation learning in conjunction with learned movement primitives may indeed be the route to believable, life-like gamebots.

Acknowledgments

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

REFERENCES

Bauckhage, C. & Thurau, C. (2004), Towards a Fair ’n Square Aimbot – Using Mixtures of Experts to Learn Context Aware Weapon Handling, *in* ‘Proc. GAME-ON’, pp. 20–24.

Bauckhage, C., Thurau, C. & Sagerer, G. (2003), Learning Human-like Opponent Behavior for Interactive Computer Games, *in* B. Michaelis & G. Krell, eds, ‘Pattern Recognition’, Vol. 2781 of *LNCS*, Springer-Verlag, pp. 148–155.

Cass, S. (2002), ‘Mind games’, *IEEE Spectrum* pp. 40–44.

Diaconis, P. (1988), Recent Progress on de Finetti’s Notions of Exchangeability, *in* J. Bernardo, M. DeGroot, D. Lindley & A. Smith, eds, ‘Bayesian Statistics’, Vol. 3, Oxford Univ. Press, pp. 111–125.

Hart, P., Nilsson, N. & Raphael, B. (1968), ‘A Formal Basis for the Heuristic Determination of Minimum Cost Paths’, *IEEE Trans. on Systems Science and Cybernetics* **4**(2), 100–107.

Le Hy, R., Arrigioni, A., Bessière, P. & Lebeltel, O. (2004), ‘Teaching bayesian behaviours to video game characters’, *Robotics and Autonomous Systems* **47**(2–3), 177–185.

Martinetz, T., Berkovich, S. & Schulten, K. (1993), ‘Neural Gas Network for Vector Quantization and its Application to Time-Series Prediction’, *IEEE Trans. on Neural Networks* **4**(4), 558–569.

Mealy, G. (1955), ‘A Method for Synthesizing Sequential Circuits’, *Bell System Technology J.* **34**, 1045–1079.

Moore, E. (1956), Gedanken-experiments on sequential machines, *in* ‘Automata Studies’, number 34 *in* ‘Annals of Mathematical Studies’, Princeton University Press, pp. 129–153.

Rao, R., Shon, A. & Meltzoff, A. (2004), A Bayesian Model of Imitation in Infants and Robots, *in* K. Dautenhahn & C. Nehaniv, eds, ‘Imitation and Social Learning in Robots, Humans, and Animals: Behavioural, Social and Communicative Dimensions’, Cambridge University Press,.

Sklar, E., Blair, A., Funes, P. & Pollack, J. (1999), Training intelligent agents using human internet data, *in* ‘Proc. Asia-Pacific Conf. on Intelligent Agent Technology’, pp. 354–363.

Spronck, P., Sprinkhuizen-Kuyper, I. & Postma, E. (2003), Online Adaptation of Game Opponent AI in Simulation and in Practice, *in* ‘Proc. GAME-ON’, pp. 93–100.

Thurau, C., Bauckhage, C. & Sagerer, G. (2003), Combining Self Organizing Maps and Multilayer Perceptrons to Learn Bot-Behavior for a Commercial Computer Game, *in* ‘Proc. GAME-ON’, pp. 119–123.

Thurau, C., Bauckhage, C. & Sagerer, G. (2004), Synthesizing Movements for Computer Game Characters, *in* C. Rasmussen, H. Bülthoff, M. Giese & B. Schölkopf, eds, ‘Pattern Recognition’, Vol. 3175 of *LNCS*, Springer-Verlag, pp. 179–186.