Learning human-like Movement Behavior for Computer Games

Christian Thurau
Applied Computer Science
Bielefeld University
P.O.Box 100131, 33502 Bielefeld, Germany
cthurau@techfak.uni-bielefeld.de

Christian Bauckhage
Applied Computer Science
Bielefeld University
P.O.Box 100131, 33502 Bielefeld, Germany
cbauckha@techfak.uni-bielefeld.de

Gerhard Sagerer Applied Computer Science
Bielefeld University
P.O.Box 100131, 33502 Bielefeld, Germany
sagerer@techfak.uni-bielefeld.de

Abstract

Modern Computer Game AI still relies on rule-based approaches, so far failing to develop a convincing, human-like opponent. Towards the development of more human-like computer game agents, we propose an approach for learning strategies by observation of human players, principally viewing the design of a computer game agent as a problem of pattern recognition. First we introduce a Neural Gas based grid learning of an internal representation of the 3D game-world. Then we discuss the use of a learning potential fields approach, establishing a mapping from world state vectors to corresponding potential field forces for agent guidance. The training data being used is acquired by observation of human players acting in the 3D game environment of a commercial computer game. Finally, some experiments are presented.

1. Introduction

Over the last years computer games changed their appearance from simple card or reaction games to complex virtual world simulations. Although the graphical advances are impressive, modern games most often ruin realistic world impressions by poor implementation of their virtual inhabitants, artificial agents (game term:*bots*). Up to now, actions and behaviors of game agents are scripted or rely on (at times very complex) finite state machines (Cass, 2002). It is at least questionable, if finite state machines will lead to convincing, human like, game agents.

Laird outlined, that computer games are an excellent experimental environment, when it comes to the investigation of human level AI (Laird and v. Lent, 2000), and Isla (Isla and Blumberg, 2002) suggested some of the possible research directions in Game AI. Nevertheless, up to now there has been little effort in applying techniques from pattern recognition or machine learning. While learning movement primitives for humanoid robots (Schaal et al., 2004) or implementations of autonomous behavior for simulated worlds (Terzopoulus et al., 1994)(Tomlinson and Blumberg, 2002) are quite common, so far no one seems to use computer games as an application for learning human behavior in virtual worlds. A closer look at these games features will reveal some of the opportunities.

On the one hand there are the gaming worlds, which are inspired by the real world. That does not only take into account the graphical representation or game physics, but also the tasks a human player has to fulfill, which are a lot less abstract than for example in a game of chess. Especially the genre of MMORPG's (Massively Multiplayer Online Role Playing Games) focuses on a diversity of aspects, the social interaction with other human players taking the most important role, since these games want to be understood as a world simulation, giving the player the opportunity to be whatever he wanted to. Although these games can be seen as a simulation, they are not from a scientist point of view:"..[computer games are] real products and real environments on their own that millions of humans vigorously interact with and become immersed in.."(Laird and v. Lent, 2000).

On the other hand, artificial agents in a computer game can "embody" a human player. They can be restricted to a human's world perception and reaction spectrum, which makes the adaption of training samples for classification approaches less of a problem (in many games a human player and a game agent share the same interface to the game world via the network protocol). Further, just like in a simulation, there is no need for low-level operations, such as object recognition or basic motor-control. Most importantly, although it is a game, human players show a lot of different aspects of human act-

Figure 1: A game bot's (foreground character) possible perception (left picture) and reaction (right picture) spectrum.

ing and thinking, providing interesting challenges for learning approaches.

Consider for instance the well known genre of first persons shooters (FPS) games on which we will concentrate in our experiments. They provide a complex, feature rich 3D game world, in which humans compete each other over the Internet. Every player taking part has control over a single character and therefor has only a limited local view onto the game world, Figure 1 shows a game-bot and his possible perception and reaction spectrum. The main goal is to survive as long as you can, by picking up different types of items, and shoot as much enemy players as you can, by using different types of weapons. The game physics stretch what can be commonly understood as realistic physics, nevertheless the players are bound to it (we are currently focusing on more comic like FPS games, instead of "realistic" FPS games, since they do not only add realism to the physics but also to the violence). Although these games share a violent background, players are more interested in the sport aspects of the game, team based modifications are highly popular. Which at first seems like a mindless shooter game, turns out to get very complex, when the aspect of a sport competition comes into play, no wonder there are numerous Internet sites which deal with all kinds of different tactics on how to play the game in a successful manner, simple rules do not necessarily lead to a simple game.

What was said for computer games in general, holds for FPS games as well: Usually, finite state machines are used for agent control. These agents offer a challenge rather by perfect aiming than by a superior game play, and can be easily distinguished from a human player.

The FPS games fast pace does not leave much time for thinking. In order to play the game successfully, the human player has to develop a set of *reactive behaviors* gathered from his own gaming experiences. Thus a direct mapping of the k last world states $\vec{s}_t, \vec{s}_{t-1}, \ldots, \vec{s}_{t-k}$ onto a bot's reaction $\vec{a}_t$ seems to be possible, it is exactly what a human player is doing most of the time. These games popularity offers the opportunity to gain access to large training data sets. Not only by observation of an ongoing match, but also by downloading so called *demo files*. These demos are basically records of the network code of an ongoing match and can fully reproduce a single match from the perspective of one player. A lot of sites on the Internet give access to thousands of those demo files, each of which containing hundreds of training samples. The basic idea is to learn human-like behavior through analysis of these freely available training sets.

Although the overall problem of learning human-like behavior for an FPS game agent can be reduced to building up a set of reactive behaviors, it can be understood as a common task dependent problem solving. Doing so, we can divide the player's actions into global strategies and more localized tactics and basic reactive behaviors. Global strategies are, in our domain, mostly targeted at controlling the 3D gaming world, by gaining access to important items and holding off opposing players from certain map areas. Where tactics usually include enemy movement pattern anticipation and appropriate weapon selection and usage. Behaviors are very similar to the described reactive behaviors and build up movement/action primitives. Dividing human reactions into the described categories simplifies the overall problem and allows for an in depth analysis of each separate layer of a human's thinking reaction process. Of course, some kind of merging function, selecting a layers activation level has to be developed.

We already showed, that purely reactive behaviors can be learned by observation of a human player (Thurau et al., 2003) by means of a combination of Self-Organizing-Maps and Multi-Layer-Perceptrons. In this paper we concentrate on methods for learning a *game strategy*. Although there are other important behaviors to learn, this one is crucial, since it often makes up for the impression on how smart a human or a game bot is. A good aim would never add to a human like artificial agent's impression (the opposite is the case, human players are often thought of using an "aiming-bot", when there aim is too good), where a smart movement in the gaming world, a plan behind certain movements, would certainly create a convincing, realistic impression.

Strategic decisions in a FPS game are mostly based on de-

cision processes on picking up certain items and guarding areas. Since *Artificial Potential Fields* (Arkin, 1998) provide an computationally inexpensive and well examined approach for navigational, goal oriented movement control, and allow for an overlay of concurrent goals, we chose them for pathfinding. Because of the strong coupling between a player's decision at timestep t and the world state $\vec{s}_t$, we want to learn the mapping between world states and corresponding potential field forces, section 2.2 deals with this subject. To overcome some common potential field problems, we are using an *Avoiding The Past Pheromone Trail*, which is described in section 2.3. In section 3., we present some first experiments and their evaluation and finally end with a small conclusion and an outlook on our future work in section 5. First, however, the next section describes our approach for learning an internal grid based representation of the 3D gaming-world, using a *Neural Gas* algorithm.

2. Learning plans

2.1 Neural Gas for Waypoint Learning

For a representation of the virtual world we are learning a grid based *waypoint map* using a Neural Gas (Martinez et al., 1993) algorithm, an improved k-means algorithm showing good performance when it comes to topology learning (Martinez and Schulten, 1991). The training data used for Neural Gas learning consists of all positions $\vec{p} = \{x, y, z\}$ a human player held during various plan executions, thus staying very close to the actual human movement paths by only taking visited places into account.

Applying a Neural Gas algorithm to the player positions results in a set of position prototypes, places a player visited frequently. The number of prototypes dependes obviously on the 3D map size. In our experiments a number of 1100 prototypes showed good results where 19000 player positions were used for training. In a second step neighbouring prototypes are interconnected. The edges connecting the nodes are not computed using the classical hebbian topology learning (Martinez and Schulten, 1991). Drawing edges based on concurrent activity would certainly lead to problems on one-way paths. Therefor we only connected two nodes, if there was an observabel player movemement from one node to the other. This finally results in a topological map representation or waypoint map. The edges are stored in a $n \times n$ matrix $C_{i,j} \in N^+$ where n denotes the number of units/neurons. If $C_{i,j} = 0$ unit i is not connected to unit j, if $C_{i,j} = 1$ unit i is connected to unit j, Figure 2 shows a resulting waypoint map. The node distances can be computed in advance and are stored in a $n \times n$ matrix D, where $D_{i,j} \in \mathcal{R}^+$ denotes the distance from node i to node j.

Using a nearest neighbour criterion, every position on the 3D map can be assigned to a single node. A movement from one node to another is only possible if a connection between the two exists, and can be easily translated into "real" movements in the 3D map.

When considering potential fields which will guide an agent through the 3D world, the waypoint representation has several advantages over directly moving in 3D. Potential fields placed on certain nodes won't influence the agent's behavior through walls or one-way paths. Besides, we can be sure that when our agent is moving along the waypoint nodes, he won't fall of ledges or run into walls.

2.2 Learning Potential Fields

A world state vector is denoted as $\vec{s} = \{s_1, s_2, \cdots, s_n\}$, where $s_1, s_2, \cdots, s_3$ describes certain world aspects or internal states of an agent. In our experiments we used the player's inventory (which weapons or items did he already picked up) as well as his current armor and health value as state vector variables.

Similar encountered situations will most always lead to similar strategic actions or movement patterns. Therefor we are grouping similar observed player states using a Neural Gas algorithm, resulting in a number of state vector prototypes, representing more common game situations. In a second step each observed player position is assigned to one state prototype, effectively forming a number of situation dependent movement behaviors. Each movement behavior begins when entering a new state space cluster, and ends when a state change occurs. Most state changes -in the state space we consider- occur when items are picked up. Therefor almost all movement patterns end with an item pickup, implicitly defining item positions as the goals to attain.

The state space clustering finally leads to a number of movement patterns which were executed, when the human player encountered a certain situation. A single movement behavior can be denoted as:

$$m_i = [m_{i,1}, m_{i,2}, m_{i,3}, \cdots, m_{i,k}]$$

where m_i denotes a complete movement pattern and $m_{i,k} \in N_1^n$ denote a single node, where n denotes the length off the overall movement pattern m_i.

To rebuild the human movement behavior, we decided to use potential fields for agent guidance. Potential fields originating in a node in the waypoint map produce certain potential on all nodes, which decrease with the distance to the potential field source. Since any node is a possible potential field source, the potential f_j at node j is denoted as:

$$f_j = \sum_{i=1}^{n} \frac{F_i}{d_{i,j}^2} \tag{1}$$

where $d_{i,j}^2$ denotes the distance from node i to node j and F_i denotes the potential field strength at node i.

For every given state space cluster we have to find a potential field force configuration, which will lead to the observed movement patterns. Since the movement in a potential field

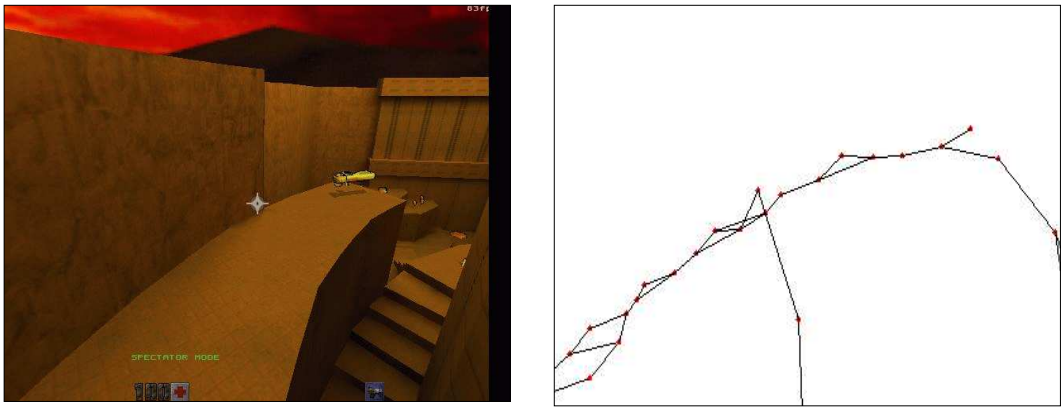

Figure 2: A screenshot of a small part of the map and the corresponding waypoint map

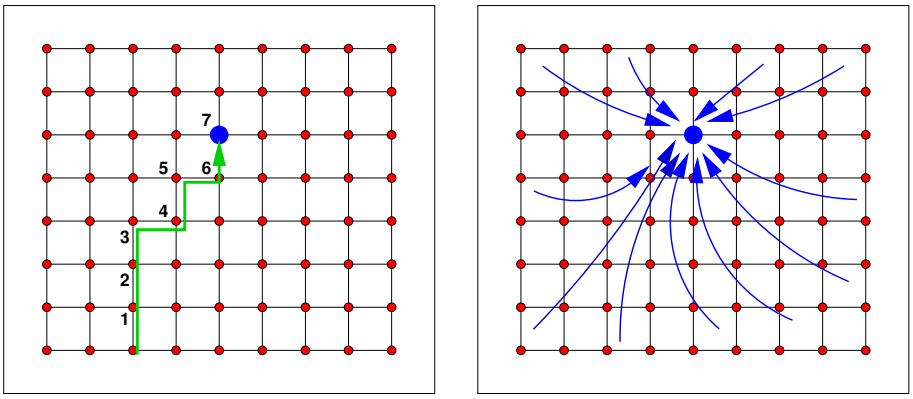

Figure 3: The left figure shows the observed movement path and the assigned discrete potentials, the right the learned potential field forces

does always follow the steepest gradient, there has to be an increase in the potentials f_i found on every node along a specific movement pattern. Consequently we construct a set of possible potentials for every node along a certain movement pattern, by assigning ascending discrete values to the node sequence m_i.

$$m_i = [(m_{i,1}, f_1 = 1), (m_{i,2}, f_2 = 2), \cdots, (m_{i,k} = k, f_k = k)]$$

where f_n denotes the assigned potential for a node n. This is done with all observed movement paths belonging to a certain state space cluster, giving more weight to more common plans by summing up all observed potential distributions. As a side effect arbitrary movements which do not approach the final goal node $m_{i,k}$ are always assigned lower potentials than their successor nodes, effectively favoring goal oriented nodes and still producing goal oriented potential distribution as Figure 4 illustrates.

In the next step a potential field strength F_i for every node i which leads to the constructed potential distribution, needs to be computed, the following error function has to be minimized:

$$E = \frac{1}{2} \sum_{i=1}^{n} (f_{i,desired} - f_i)^2 \tag{2}$$

where $f_{i,desired}$ denotes the sum over all assigned potentials of a node belonging to a state prototype. A gradient descent leads to the update rule for the potential field force F_i of a node i:

$$F_i + 1 = F_i - \alpha \sum_{k=1}^{n} \left(f_{i,desired} - \sum_{l=1}^{n} \frac{F_l}{{d_{k,l}}^2} \right) \frac{1}{{d_{k,i}}^2} \tag{3}$$

where α denotes a learning rate which is decreased during the training process.

Applying the learning rule, we end up with a maximum of active potential fields which is equal to the number of passed nodes assigned to a state prototype. Figure 3 illustrates the relation between the observed movement path and resulting potential field forces, given a simple goal oriented movement pattern.

On the first sight this methods seems a bit ineffective, a shortest path algorithm like Dijkstra might provide the same

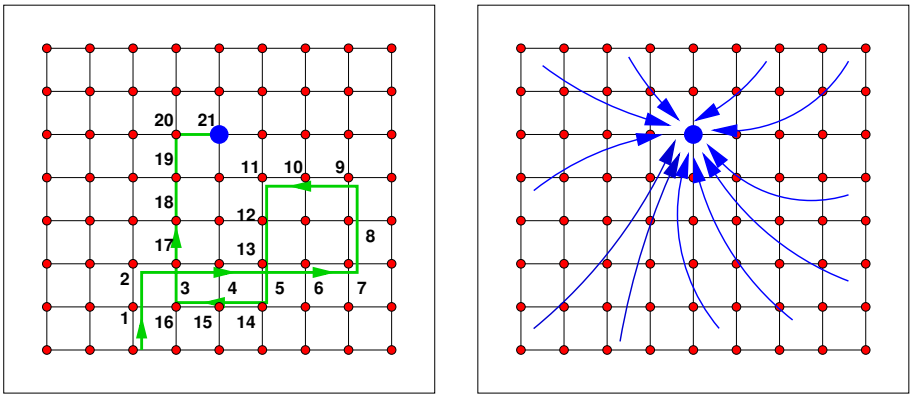

Figure 4: An observed movement pattern with some arbitrary, non goal oriented movement, does still produce goal oriented potential fields.

results. But the problem we are facing is different from finding the shortest path to a specific map location. Learning a strategic movement behavior implies, that we are not searching for the shortest path, but for the strategically most appealing. Which could also mean to guide the agent on a longer path, if it provides strategic advantages (a better path could be in darker areas, avoiding open terrain and so on), Figure 5 illustrates how a longer way would be favored over the shorter one.

If the number of training samples is limited, the potential field learning effectively results in a shortest path way finding to the most attractive area, when approaching from a direction which doesn't provide any local guidance. With more training samples available more information on how to approach certain important nodes is provided, resulting in "smarter" movements and a better local guidance.

While the overall approach is similar to learning a value function known from dynamic programming or reinforcemeant learning (Sutton and Barto, 1998), here it is merely a representation of specific movement patterns. A further improvement of certain strategies by means of reinforcement learning, though desirable, is not intended at the moment. Besides, in its current form our approach is efficient and computationally inexpensive, thus well suited for fast paced real time computer games.

However, the usage of potential fields comes along with some problems. On the one hand we might encounter local maxima, stucking the agent at unimportant nodes, on the other hand we have a relatively poor guidance when the distance to originating potential field nodes increases. The next section deals with both problems.

2.3 *Dealing with Potential Field flaws*

Avoiding the past potential fields (T.Balch and R.Arkin, 1993) serve as a local spatial memory, to avoid already visited nodes. At fixed time intervals a repulsing potential force at the current node is deposited. To reduce interference with goal based potential fields, the avoid-past potential field forces are reduced over time, thus effectively resulting in a pheromone trail, known from ant-algorithms. In addition the area of effect is reduced compared to the goal based potential field forces.

These pheromone trails have two purposes. First, local maxima can be avoided by pushing the agent to nearby nodes, hopefully back to the field of influence of goal orientated potential fields. Second, areas which provide poor guidance because of the distance to goal oriented potential fields can be escaped easily, by reinforcing a chosen direction.

The overall potential distribution on waypoint map node m_j, at timestep t is now computed as follows:

$$f_j(t) = \sum_{i=1}^{n} \frac{\mathcal{F}_i}{{d_{i,j}}^2} - \sum_{i=1}^{n} \frac{\mathcal{P}_i(t)}{{d_{i,j}}^3} \tag{4}$$

where $P_i(t)$ denotes the Pheromone concentration at node i, at timestep t, and with $P_i(t+1) = P_i(t) - \varrho P_{evap}$, where ϱ is a diffusion parameter and P_{evap} is the fixed evaporation parameter for an occupied node.

Both parameters P_{evap} and ϱ have to be carefully chosen, to provide the expected effects and to not interfere too much with the goal oriented potential field forces. If chosen properly to fit the premise of very local influence on the goal oriented potential fields, a recomputation of the goal oriented potential field forces is not necessary. However, our experiments were not influenced by moderate changes to the parameters, after all the field of influence of a pheromone is more localized than a potential field.

If goal oriented potential fields are not available, the pheromone trail generates an "explore map" behavior, which, though not intended, adds to a more human-like agent impression. A proper guidance in any given world state is to favor over time consuming non goal oriented movements, evolved from the pheromone trail, just relying on the pheromone trail would not produce convincing results.

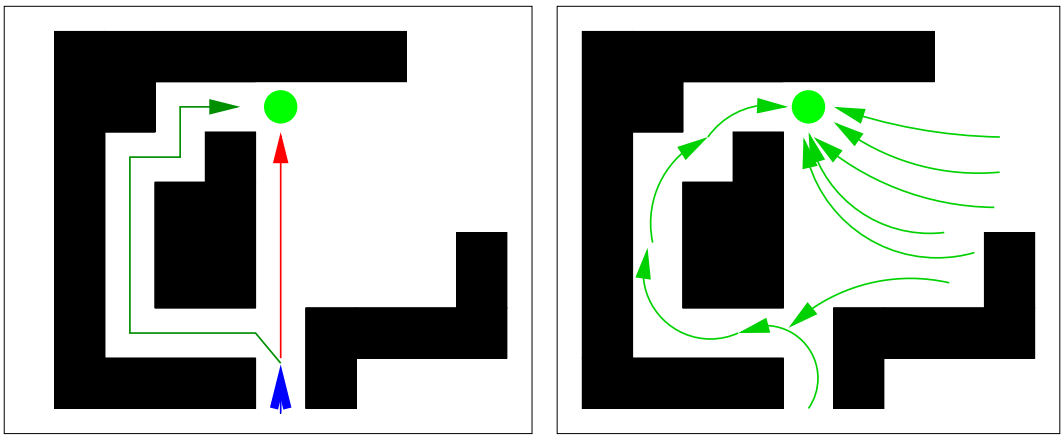

Figure 5: The left figure shows an observed movement path, where the green arrows denotes the actual observation, the red one the shortest path to the goal. The right figure shows the learned potential field forces, which would guide the agent along the longer path, when approaching the goal from the initially observed direction.

3. Experiments

3.1 Simple Tasks

The experimental environment consists of the commercial computer game Quake II and a Quake II Matlab client. To focus on the presented approach human players were presented a certain task, which had to be executed and was afterwards used for training.

The agent control mechanism is quite simple. With the knowledge about the potential field forces, the potentials for all nodes, connected to the currently occupied node, can be computed very fast. From the list of possible successor nodes the one with the highest potential is selected. Environmental influences or item pickups change the agents state vector $\vec{s}$ and result in a change to the overall potential field forces, thereby leading the way to other nodes.

The training data used consisted of the observation of 20 different tasks on one map, mainly consisting of item pickup sequences. For each task a separate bot was trained. To estimate the game-bot performance, we evaluated four different starting locations (game term: "spawn point") from which it had to find its way according to the human behavior in the training samples. Performance can be measured over the number of goals the bot reached, and the time it took him to do so. Table 1 shows some of the results. Generally our approach showed very good results, failing to reach the goals in the predefined order only once. In that particular failed experiment, the game-bot got stuck at some part of the map and didn't manage to reproduce the desired behavior in less than two minutes, however, he might have reached the goals later on, since the pheromone trail would have driven him towards an (even weak) attractor.

Visual/graphical comparison of the human and artificial player's movement paths gives further insights in how well the potential fields are able to learn human movement patterns. Figure 7 shows the observed game agent's movement path in comparison to the human's movement path. While Figure 6 illustrates the agent's behavior when starting from a different spawn point, then the one observed in the training samples.

Task	SP 1	SP 2	SP 3	SP 4
Set 1	13.89	17.60	16.51	8.31
Set 2	33.30	25.71	24.50	31.42
Set 3	8.42	9.00	11.43	3.02
Set 4	22.49	19.30	23.62	14.89
Set 5	11.64	9.21	13.61	14.01
Set 6	16.82	16.52	22.70	16.42
Set 7	33.30	35.03	36.40	31.54
Set 8	28.01	–	–	–
Set 9	41.50	41.24	41.80	46.33
Set 10	20.02	14.71	24.82	15.71
Set 15	39.19	46.48	41.20	36.21
Set 20	40.2	55.10	32.41	50.91

Table 1: The bot had to complete every learned task when starting from four different locations SP 1-4. To give an indication about the agent's performance, we measured the time in sec. it took him to reach all goals in the predefined order, thereby reproducing the human movement behavior.

3.2 More Complex Behaviors

Since strategic movements do not depend on the execution of a single, isolated movement pattern, but on a combination of situative adequate movement paths, we decided to test the described approach with more complex behavior learning. Hereby we instructed a human player to play the game Quake II, as if an enemy was present (again we simplified the situa-

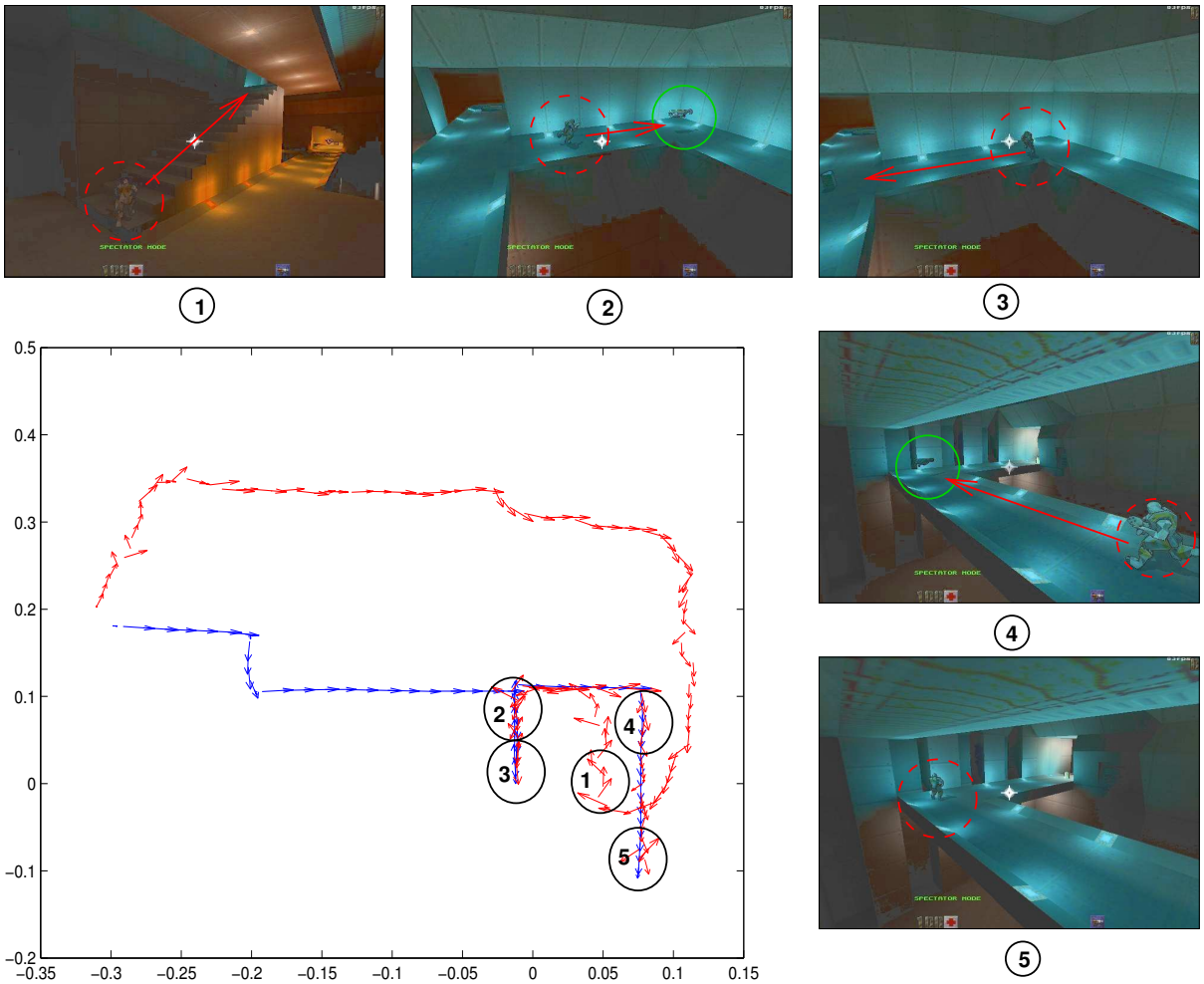

Figure 6: The bot started from a map location, other than the one used during the learning process. The red arrows show all bot positions and its velocity, while the blue arrows show the positions and velocity of the observed human player's movement path. Picture 1 shows, where the bot moves up a staircase and comes close to the observed movement pattern. In Picture 2 he is targeting the first item and picks it up at 3, reproducing the observed human behavior. A state change occurs, switching the potential field forces. The bot moves on to the next goal in 4 and finally reaches it at 5.

tion in order to focus on the strategical aspects). This resulted in a set of training samples, where the player collected various items in different orders, dependent on the initial starting position, and after that securing important areas by cycling different types of armor (a quite common behavior shown by many players).

This experiment setup, too, yielded good results, indicating that the approach is also working reasonably well for a larger scale. The artificial agent moved around the 3D gameworld in a convincing manner and managed to reproduce the observed behavior. He picked up several different types of items, before he mostly stayed at one area, usually located around an important armor item, and only moved away when driven by its pheromone trail or when there were sudden changes in its state vector (for example a decrease in its armor value due to an attack by another player). Usually the agent reached a state where he owned all weapons and filled up his armor value to the maximum.

In comparison to the first experimental setup, no fixed movement behavior sequence was reproduced. Instead, seemingly intelligent behavior emerged, by evolving a mixture of appropiate observed movement patterns.

4. Evaluation

As Table 1 indicates, the movement patterns were successfully learned for almost all tasks, and the agent managed to reproduce the observed behavior. Also local maxima or low potentials on certain nodes could be avoided using the described avoiding the past pheromone trail. Only task number 8 showed problems. The bot got stuck in three out of four different runs. The main problem in that experiment was that the first goal was located very close to the last goal, thus the agent accidentally picked up the last item at the beginning of

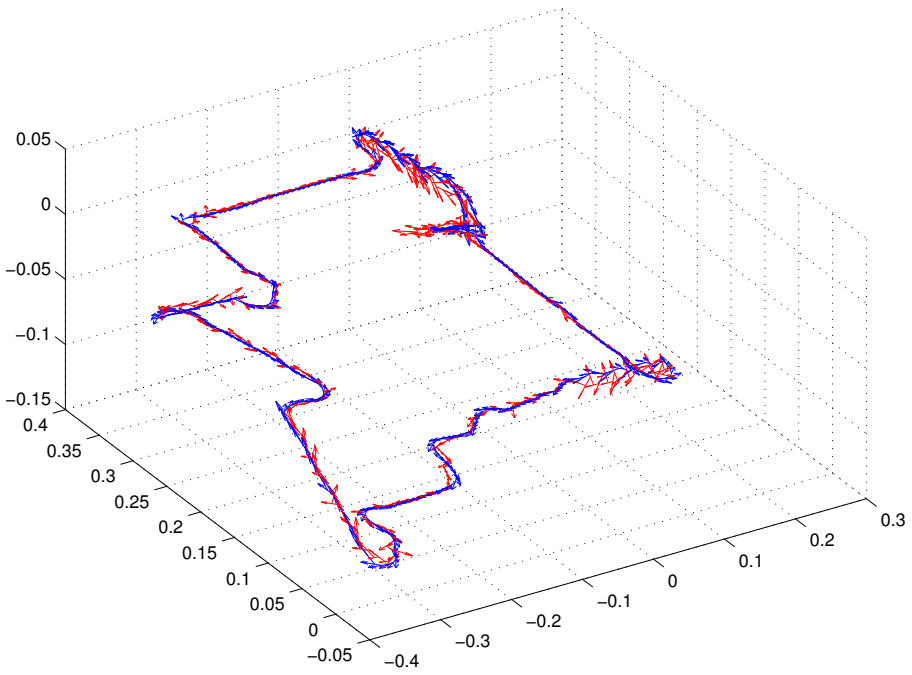

Figure 7: Visualization of a human and a game agents movement path in x, y, z. The blue arrows indicate the game bot's current positions and its velocity, the red arrows show the observed human player's position and velocity, which was used for learning the potential field forces. The bot started from the same positions as the human player and managed to fully imitate the human movement behavior.

its run, which somehow trapped it in a internal state which was associated with only very weak potential field forces and therefor poor guidance throughout the map. However, the agents pheromone trail might have led him to pick up the other items sooner or later, but we skipped the experiments after a waiting time of two minutes.

The usage of more training samples generally avoids that problem, since all states should have a sufficient number of potential field forces associated with it. So far we couldn't watch the problem of entering an almost undefined state, when using larger sets of trainings samples for learning more complex behaviors. The second experiment indicates that common complex strategic behaviors can indeed be learned using our described approach. The bot behaved in a very convincing manner, showing combinations of all observed movement patterns, while still following a more global strategy, presented in all training demos. Sudden state changes lead to understandable behavior switching and added to the overall good impression.

5. Conclusion

In order to learn a functional mapping of a computer game agents internal state on to strategic decisions, we investigated human movements in a virtual environment, the computer game Quake II.

To find a suitable way of representing the 3D game world, we learned a grid based topology of the 3D game world from the observation of a human player's movements, applying a Neural Gas algorithm. Then we learned potential field forces for agent guidance, to create human-like situation dependent movement patterns. At first we clustered the player's state space into several state prototypes, to which observed corresponding movement patterns were assigned. In a second step potential field forces, leading to the observed movement patterns, were computed using a simple error minimization. To deal with local maxima and occasionally poor potential field guidance, we were using an avoiding the past pheromone trail, evaporated from the agent's position.

And indeed, our experiments show very good imitation of the observed human behavior. Besides reaching the only implicitly encoded aims, the approach paid attention to strategic decisions of the human player on how to reach certain goals - without any knowledge of the terrain structure. Thereby staying close to the observed movement patterns.

The second experiment finally showed that larger numbers of strategic movement patterns can be decomposed and learned using our approach, covering a greater spectrum of situation dependent strategic movements and leading to more realistic sequences of movement patterns.

6. Future Work

Since not all items are available all the time, we have to investigate mechanisms of learning a time dependent potential field.

So far, except for the agents own pheromone trail, the potential fields were centered on fixed map positions. While

this provides a goal oriented movement, localized potential fields attached to moving objects, other players or projectiles could certainly lead to some interesting behavior, although this might be more suited for learning tactics, since the overall strategy is mostly determined through fixed item/ area locations.

In order to use real demo files downloaded from the Internet to model a specific player's style of play, we need to find a way to extract the strategy when there are concurring tactical and reactive behaviors. A simple smoothing of the overall movement path might reveal the long term goals.

Despite the good results, we can not be sure if the world state variables, expressing the state - potential field dependency are indeed the "really" important state variables. Datamining methods applied to real demos could provide useful insights on the relevance of certain state variables and their further influence on the learned potential field forces.

References

Arkin, R. C. (1998). *Behavior-Based Robotics*. MIT Press.

Cass, S. (2002). Mind games. *IEEE Spectrum*, pages 40–44.

Isla, D. and Blumberg, B. (2002). New challenges for character-based ai for games. In *Proceedings of the AAAI Spring Symposium on AI and Interactive Entertainment*.

Laird, J. E. and v. Lent, M. (2000). Interactice Computer Games: Human-Level AI's Killer Application. In *Proc. AAAI*, pages 1171–1178.

Martinez, T., Berkovich, S., and Schulten, K. (1993). Neural gas network for vector quantization and its application to time-series prediction. In *IEEE Transactions on Neural Networks*, pages 558–569.

Martinez, T. and Schulten, K. (1991). A neural gas network learns topologies. In *Artificial Neural Networks*. Elseviers Science Publishers B.V.

Schaal, S., Peters, J., Nakanishi, J., and Ijspeert, A. (2004). Learning movement primitives. In *International Symposium on Robotics Research (ISRR 2003)*. Springer.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press.

T.Balch and R.Arkin (1993). Avoiding the past: A simple but effective strategy for reactive navigation. In *IEEE International Conference on Robotics and Automation*, pages 678–685.

Terzopoulus, D., Tu, X., and Grzeszczuk, R. (1994). Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. In *Artificial Life*, pages 327–351.

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

Tomlinson, B. and Blumberg, B. (2002). Alphawolf: Social learning, emotion and development in autonomous virtual agents. First GSFC/JPL Workshop on Radical Agent Concepts.