IMITATIVE LEARNING OF COMBAT BEHAVIOURS IN FIRST-PERSON COMPUTER GAMES

Bernard Gorman
Dublin City University
Glasnevin, Dublin 9
Rep. of Ireland
+353 1 4902714
bernard.gorman@computing.dcu.ie

Mark Humphrys
Dublin City University
Glasnevin, Dublin 9
Rep. of Ireland
+353 1 700 8059
mark.humphrys@computing.dcu.ie

KEYWORDS
Imitation, machine learning, artificial intelligence, combat, game, bot, first-person, agent, neural network, virtual world, Quake, QASE.

ABSTRACT

Modern, commercial computer games rely primarily on AI techniques that were developed several decades ago, and until recently there has been little impetus to change this. Despite the fact that the computer-controlled agents in such games often possess abilities far in advance of the limits imposed on human participants, competent players are capable of easily beating their artificial opponents - suggesting that approaches based on the analysis and *imitation* of human play may produce superior agents, both in terms of performance and believability. The very fact that games provide the ability to quickly and easily generate vast quantities of raw, objective human behavioural data presents many fascinating opportunities to the AI community; opportunities which, with few exceptions, have not yet been suitably explored. Through our work in the field of *imitation learning*, we therefore investigate how best to utilise the low-level data accrued from recorded game sessions in the creation of intelligent, convincingly human game agents.

In previous contributions, we have described models capable of imitating goal-oriented strategic navigation (Gorman & Humphrys 2005) and of reproducing characteristically human movement in *first-person shooter* games (Gorman, Thurau, Bauckhage & Humphrys 2006a); we have also outlined a comprehensive approach to the evaluation of agent believability (Gorman et al 2006b). Here, we present an approach to the imitation of combat behaviours in such environments. We first describe the extraction and processing of relevant feature vectors from the game session using our custom-built QASE API (Gorman, Fredriksson & Humphrys 2005). We then outline a neural-network based model designed to learn the aiming and context-sensitive weapon handling exhibited by the human players. Finally, we describe an experiment to demonstrate the efficacy of this approach; some observations and future directions for our work close this contribution.

INTRODUCTION

Imitation learning, as the name suggests, refers to the acquisition of skills or behaviors through examination of a demonstrator's execution of a given task. Imitative techniques have been adopted by many researchers in the field of robotics as a means of "bootstrapping" their machines' intelligence, providing them with a near-optimal level of competence after a comparatively short training period (Hayes & Demiris 1994, Schaal 1999, Fod et al 2002). Despite the interest exhibited by the robotics community, however, very few attempts have been made to apply these principles to interactive computer games; indeed, even the most modern games still predominantly rely on symbolic AI techniques that have evolved little since the 1950s (Laird 2000, Fairclough et all 2001). Given that many present-day games allow the recording of entire sessions, and that - rather than limb movement data, as is common in robotic imitation - these recordings encode the frame-by-frame behaviour of the player under complex, rapidly-changing conditions and in competition with opponents of comparative skill, it becomes clear that computer games are an ideal platform for research in imitation learning. In addition, since many modern games are based an abstraction of the real world - as opposed to a board or puzzle game with no relevance to reality - the possibility exists that techniques developed to imitate in-game human behaviour may be adaptable for application in the real world.

For our own work in this field, we opted to investigate the *first-person shooter* genre, in which players explore a three-dimensional environment and engage in armed combat against other human or AI players. This particular genre was chosen in preference to others due to the fact that it provides a comparatively *direct* mapping of human decisions onto agent actions, imposing a minimal degree of abstraction between the player and his/her virtual avatar. We chose ID

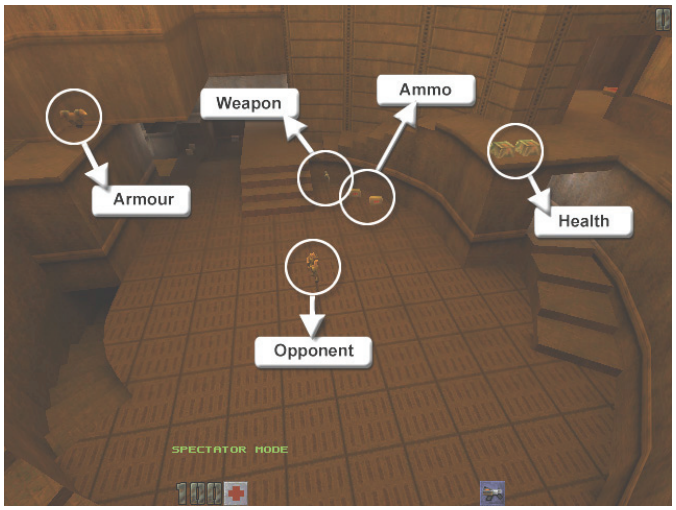

Figure 1 - Typical Quake 2 environment

Software's **Quake 2** as our testbed, given its relative prominence in the literature and the comparatively substantial resources available to developers (Laird 2001). **Figure 1** above shows a typical Quake 2 environment with features labelled.

MOTIVATION AND RELATED WORK

In Quake 2, as in all first-person shooter games, players engage in direct combat with a wide variety of weaponry. While the specific weapons vary from game to game, a number of common characteristics prevail; certain guns fire slow-moving but powerful projectiles, others fire quick but weak bursts. Some are ideally suited to ranged combat, whereas others are more useful in close quarters. Depending on the situation, the player may need to aim *ahead* of the opponent in order to give the projectile sufficient time to meet its target. Experienced human players will - with little or no conscious effort - collect and utilise these weapons in such a way as to maximise their effectiveness, instantly weighing a vast array of variables (proximity to opponent, remaining ammunition, etc) before deciding upon their course of action. In comparison to this lucid play style, the performance of traditional rule-based artificial agents leaves much to be desired. Designed top-down, they exhibit at best a crude approximation of the human player's context-sensitive weapon handling; often, they will simply retain whichever gun they happened to last pick up. To compound this problem, game developers generally compensate for their agents' shallow intelligence by endowing them with superhuman capabilities far beyond the constraints imposed by the human player's mouse-and-keyboard interface, enabling them to read the position of their human opponent from the gamestate and strike with pinpoint accuracy. The fact that such bots are consistently unpopular with gamesplayers demonstrates that sacrificing believability to artificially increase the competitiveness of agents is an inherently misguided approach; even when losing to a superior human opponent, gamesplayers invariably enjoy the experience more than fighting an unrealistic bot.

Learning the Inaccuracy

With this in mind, we look to imitation learning to provide an alternative. (Bauckhage & Thurau 2004) presents a discrimination of the differing behaviours typically present in game sessions - short term *reactive*, mid-term *tactical* and long-term *strategic* - based on Hollnagel's model of human behaviour as a function of available time and information (Hollnagel 1993). Our previous contributions have primarily concentrated on the strategic and tactical layers; this work is situated on the boundary of the reactive (close-quarter combat with limited available reaction time) and tactical strata (long-rage combat with greater scope for planning). See **Figure 2**.

(Bauckhage et al 2003) present an initial approach using a mixture of 3 experts to aim and switch between three predefined weapons - one short range, one medium, one long - in a simple 2D setting. Counter-intuitively, their experiments revealed that the weapon competencies were not allotted bijectively to the networks, but rather produced a holistic representation distributed across all three experts; they speculated that the handling of different weapons does not differ as widely in the data space as the visual in-game result suggests. In addition, they report that - because the underlying function is quite straightforward, as shown by the traditional bots discussed above - the network learned to aim *too* well, resulting in precisely the kind of pinpoint accuracy we wish to avoid.

It is from these findings that we draw the intuition underpinning this paper; rather than attempting to learn the player's *aiming* (since we could, if we so desired, write a highly accurate aiming algorithm in perhaps five lines of code) we instead learn what distinguishes a human player from a traditional game bot - his *inaccuracy*. As sometimes occurs in our work, this places us at odds with most of our counterparts in the field of robotics, who use imitation as a means of quickly attaining optimal performance; since we are primarily interested in producing realistically human behaviour, we often need to deliberately strive for the competent yet suboptimal. (Dillman et al 1995), for instance, discuss what they regard as bothersome human-produced "noise" in imitation learning - *incorrect, unmotivated or unnecessary* actions - which are, from our perspective, precisely the idiosyncratic, suboptimal, quintessentially human behavioural traits we are attempting to capture.

The inaccuracy of the player's aim derives from two distinct components:

- **unintentional inaccuracy**, due simply to human error
- **intentional inaccuracy**, the player's practice of accounting for his opponent's motion while aiming

Using this as our starting point, we attempt to learn context-sensitive weapon switching and aiming across the entire range of available weapons, and in all three dimensions.

METHODOLOGY

Data Extraction

Quake 2's multi-player mode operates on a standard client-server model. One player starts a server and other combatants connect to it, entering whatever environment (known as a *map*) the instigating player has selected. The server transmits an update frame to all connected clients ten times per second; each client merges the update into its existing gamestate record, and then responds by sending its desired movement, aiming and action back to the server. A

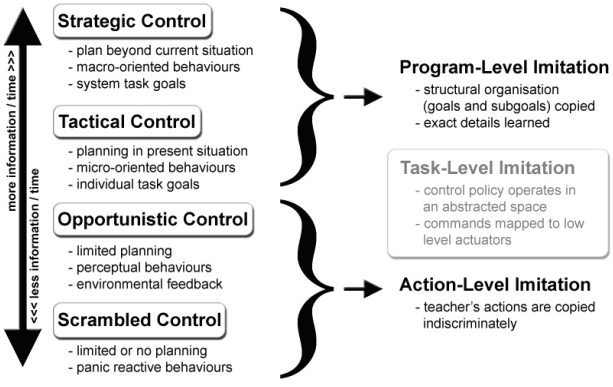

Figure 2 - Thurau's adaptation of Hollnagel's COCOM

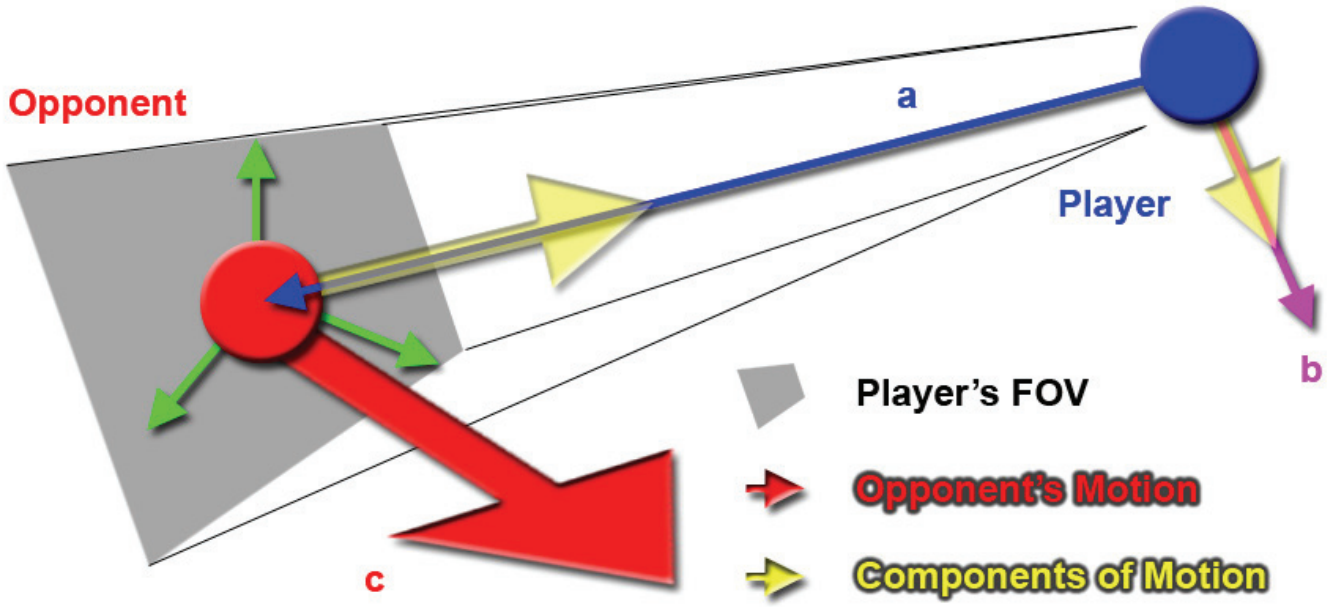

Figure 3 - Reconstruction of the player's perception from low-level data.

facility for recording these network packets to so-called *demos* or *DM2* files is provided. In order to extract the necessary feature vectors from the recorded sessions, we employ our own QASE API (Gorman & Humphrys 2005).

To facilitate the learning process, we must first reconstruct the player's perception of his opponent's motion using the available low-level data. For each frame, we read the player's current position and the position of the nearest opponent. From this, we derive the opponent's current directional vector, and the vector running from the player to the enemy (i.e. the player's view vector). We can now express the opponent's velocity relative to the player's field of vision, by computing the projection of his directional vector onto the vectors parallel and perpendicular to the player's view angle, as shown in the diagram below.

$$v_{par} = comp_{\vec{a}}\vec{c} = \frac{\vec{a}\cdot\vec{c}}{|\vec{a}|} \qquad 1.$$

$$v_{perp} = comp_{\vec{b}}\vec{c} = \frac{\vec{b}\cdot\vec{c}}{|\vec{b}|} \qquad 2.$$

where $\vec{a}$ is the player's viewing vector, $\vec{b}$ is perpendicular to the viewing vector, and $\vec{c}$ is the direction of the opponent's motion. We also record the opponent's velocity in the vertical plane as the change in his position on the Z-axis between successive frames:

$$v_{vert} = z_t - z_{t-1} \qquad 3.$$

However, we must also consider the impact that *elevation* has on the player's perception of his opponent' motion. Consider the scenario where the opponent is directly above or below the player. In this case, any movement along the Z-axis will have no bearing upon the player's aim; if, on the other hand, the opponent is on the same horizontal plane as the player and begins to ascend or descend, the player will need to compensate by adjusting his pitch. Similarly, if the opponent is on the same horizontal plane as the player and is moving towards him, the parallel component of the opponent's movement will have little influence on the player's resulting aim; if, however, the opponent is above or below the player and moving along the parallel vector, then he will need to adjust his pitch accordingly as the enemy approaches.

To account for this, we weight both the parallel and vertical velocities according to the angular elevation of the opponent relative to the player; that is

$$v_{par} = \sin(\phi)\left(\frac{\vec{a}\cdot\vec{c}}{|\vec{a}|}\right) \qquad 4.$$

$$v_{vert} = \cos(\phi)(z_t - z_{t-1}) \qquad 5.$$

As an aside, it is worth mentioning that this is good example of the benefits of using computer games in imitation (and general) research. In robotic imitation, data is commonly derived either through video analysis or by using expensive motion-sensing equipment; here, by contrast, we can build a noise-free reconstruction of the player's visual perception of the virtual environment, without having to perform any image processing of the game session's graphical representation. Rather than using our access to the gamestate data to *cheat*, as is the case with the traditional agents discussed earlier, we instead use it to build a model which provides our agent with the same information as was obtained visually by the human player during the session.

In addition to the above, we must also read the following data on each timestep:

- the player's current weapon
- a "bitmask" indicating which weapons he currently possesses and has ammunition for

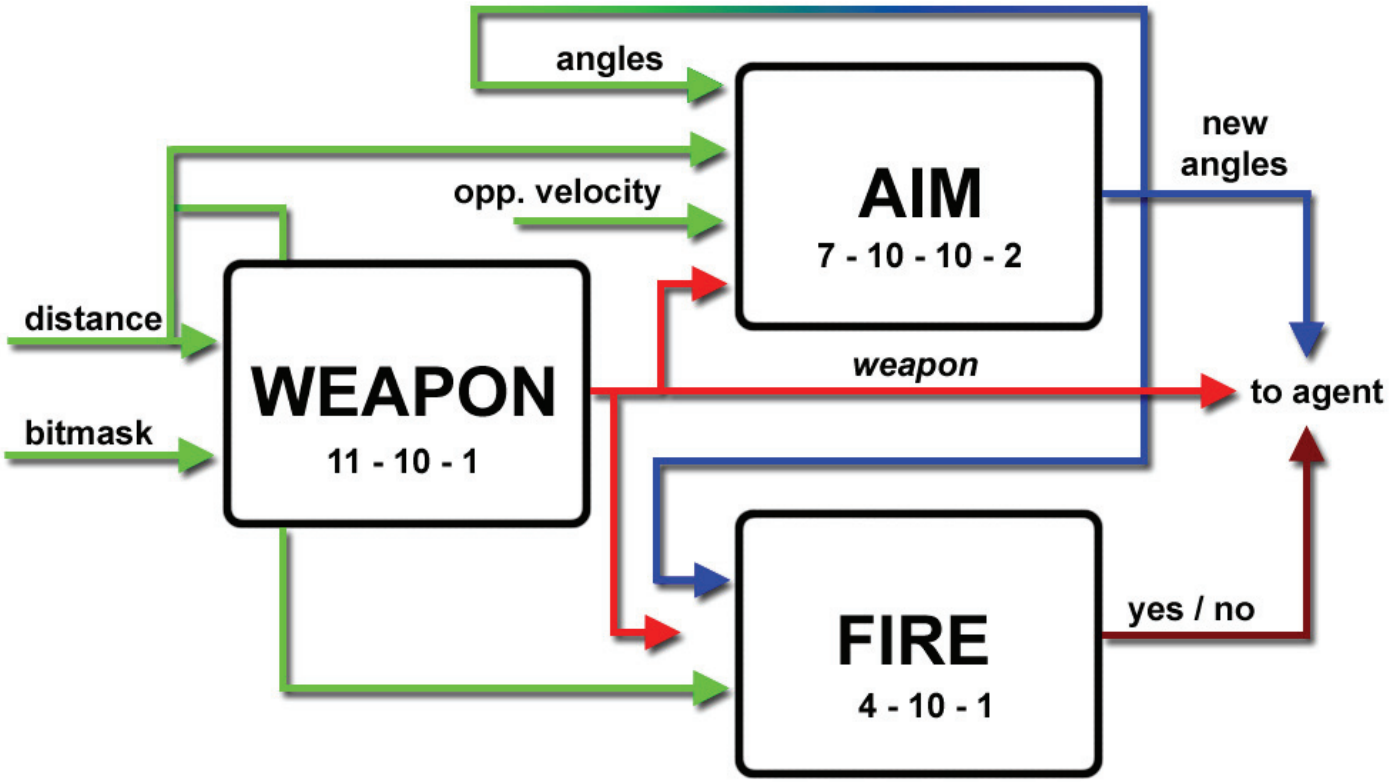

Figure 4 - A series of interconnected neural networks is employed to learn and reproduce the observed combat behaviours

- the player's distance from the opponent
- whether or not the player is firing his weapon

Together, these will form the training set for the neural network architecture.

Network Architecture

The neural network architecture itself consists of three interconnected networks, each of which is trained to perform a specific function; the model's structure is outlined in **Figure 4** above. Each network is trained as follows:

WEAPON
The Weapon network is responsible for selecting the most appropriate gun, given the available options. Its training data consists of the distance between the player and opponent on each timestep, together with a bitmask indicating weapon availability. The target output is the active weapon on each subsequent timestep.

FIRE
The Fire network is, obviously, responsible for determining whether or not the agent should fire its gun. Its training data consists of the current weapon, the distance to the opponent, and the angular difference between the vector from player to opponent and the player's current aiming vector. Its target outputs are 1 if the player fired his gun, and 0 if not.

AIM
Arguably the most important of the three networks is the Aim network. It is responsible for adjusting the aim of the agent in accordance with the opponent's manoeuvres. Its training data consists of the distance from player to opponent, the parallel, perpendicular and vertical velocities of the enemy, the current weapon, and the previous angular inaccuracy for some temporal context. In keeping with the approach discussed earlier, its target on each timestep is the angular difference between the vector running from the player to the opponent and the player's current aim vector (i.e. the inaccuracy of the player's aim).

When trained and deployed, the relevant feature vectors are obtained from the agent's gamestate on each timestep. The opponent's proximity and the weapon bitmask are presented to the Weapon network, producing the index of the gun to be used on the next timestep. This is in turn supplied both to the Aim network, producing the pitch and yaw accuracy of the agent's aim on the next frame, and to the agent. The Aim network supplies its angular output to the agent and also passes it to the Fire network, where together with the new weapon index and the distance between the player and enemy it determines whether or not the agent should commence firing its weapon.

EXPERIMENTS

In testing our model, we wished to use data derived from a free-flowing game session, rather than it being generated in a more controlled scenario. To that end, we first built a custom-purpose Quake 2 environment, designed specifically to maximise the variation of distance, angle and weaponry experienced by a player in the course of a game session (see **Figure 5**). We then had two players of comparable experience in first-person shooter games play on this map for just over 60 minutes, resulting in a wealth of data with which to train the neural network architecture. For this purpose, we used the victor's data as the training data, and

the runner-up's as an evaluation set. In the case of the Weapon and Aiming networks, we limited the dataset to those timesteps on which the agent is either actively firing, or is waiting for his gun to "cool down" (that is, the period immediately after a weapon discharge during which another round cannot be fired). In this way, we were able to view the player's aiming behaviours across each period of engagement with his opponent, including his adjustments between shots. This approach was obviously not appropriate for the Fire network; in its case, we extracted every available sample, whether the player was firing or not.

We then used this data to train each network using a range of topologies, employing the LM algorithm across 3000 epochs. Ultimately, we found that the topologies shown in **Figure 4** worked best; some insight may be gained by noting that the tests considered 10 distinct weapons. Very little improvement in performance was noted by increasing the number of neurons or layers in the case of the Fire and Weapon networks; we therefore opted for a 10-neuron single hidden layer configuration. We also noted that the test data error was in some cases not as close to the training set error as expected; this was not mirrored in the in-game trials, where the networks proved quite capable of generalising to unseen circumstances. On reflection, however, this seems reasonable; while the general rules of weapon handling will apply to all competent players, the individual quirks of two distinct human gamers will always lead to divergences of this kind. The effect was likely amplified by the fact that the victorious human player, from whom the training data was drawn, had quickly come to dominate the match, leaving his opponent with scarce opportunity to retaliate. An indication of this can be seen in the table below - the runner-up was observed to fire far fewer times than the victor.

RESULTS

The agent was observed to switch between weapons dependent upon its proximity to the opponent, mirroring not only the concepts outlined above, but also the human exemplar's specific weapon preferences. For instance, in close quarters combat, one might justifiably choose among the Blaster, Hyperblaster, Shotgun or Chaingun; the agent exhibited a preference for the Chaingun, switching to one of the others when it ran out of ammunition. When the opponent moved to medium range, the agent opted to use the Hyperblaster or Chaingun; at medium-to-long ranges, the Railgun or Rocket launcher was preferred.

Beyond this, a number of interesting phenomena were observed. When in close-to-medium range combat, where the player perceives the opponent as moving very fast, the agent exhibited only limited "leading" behaviour (that is, aiming ahead of its opponent to account for projectile speed). We believe this to be due partly to inadequate reaction time due to the opponent's proximity, and partly to the constraints imposed by the mouse interface. When the opponent moved further away, however, the leading behaviour became more pronounced and deliberate. Thus it appears that the weapon-handling behaviour encompasses both *reactive* and some elements of *tactical* behaviours from the Hollnagel model discussed earlier; the practice of leading the opponent becomes more prominent as the amount of time available to the player increases. We expected to see this effect to some degree at the outset of the experiment.

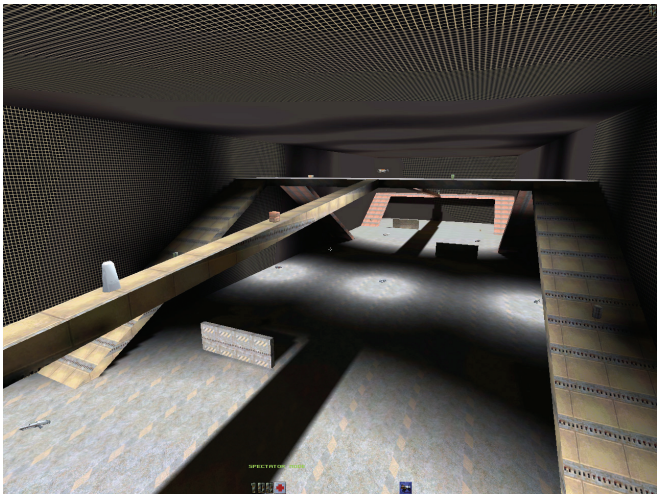

Figure 5 - The custom Quake 2 map used in our experiments

It was occasionally observed, during some trials, that the Weapon network tended to switch between multiple guns in quick succession rather than simply picking one which was range-appropriate; we believe this was due to the human exemplar's tendency to rapidly cycle through his inventory with the mouse wheel until he reached his desired gun. We are currently investigating the possibility of using a probabilistic approach to the imitation of weapon selection, given that there is a relatively discrete logical partitioning of guns with varying range. It may, however, ultimately be more beneficial to attempt to capture this rapid weapon-shifting behaviour more accurately, since it is a very recognisable human action.

Network	*Training Set #*	*Test Set #*
Aim	10184	6008
Weapon	10184	6008
Fire	37981	35678

Table 1 - Sizes of training and test datasets

Network	*Topology*	MSE_{Train}	MSE_{Test}
Aim	7-5-2	0.0065	0.0102
	7-10-2	0.0056	0.0089
	7-10-10-2	0.0038	0.0054
Weapon	11-5-1	0.0287	0.0483
	11-10-1	0.0272	0.0416
	11-10-10-1	0.0261	0.0455
Fire	4-5-1	0.1333	0.1910
	4-10-1	0.1292	0.1881
	4-10-10-1	0.1254	0.1844

Table 2 - Training phase results

Figure 6 - An example of "leading" the opponent. The agent aims ahead of the enemy, such that the Hyperblaster's projectiles coincide with his arrival. The agent tracks the opponent up the ramp in this manner, ultimately defeating him and scoring a point.

One of the most interesting (and unanticipated) observations was that, in close combat with the opponent rapidly moving left and right around the agent, it seemed to be slightly more capable of tracking the enemy as it moved anti-clockwise than as it moved clockwise. At first, we naturally assumed that the network had simply learned to reproduce one motion more accurately than the other; however, the same phenomenon was observed when the network was re-trained with differing topologies, and even when the evaluation dataset from the second human player was employed as the primary training set. Having reviewed the recordings of both the original game session and the agent's reproduction, we suspect that this is due to the fact that both players were *right-handed;* their range of motion while sweeping the mouse inward - that is, the movement required to track an opponent anti-clockwise in close proximity - is therefore slightly greater, due to it being a more ergonomically comfortable motion than an outward sweep. While reproducing this was not an intended consequence of our imitative model, it does serve to aptly illustrate the benefits of imitation learning in producing humanlike behaviour, by capturing the physical constraints of the player's mouse-and-keyboard input.

CONCLUSION

In this paper, we discussed the importance of humanlike, context-sensitive weapon handling, and the lack of same among traditional rule-based AI bots. We outlined some previous work in this area, and explained the intuition behind our approach of "learning the inaccuracy". We demonstrated how the player's visual perception of the game at each timestep may be reconstructed from low-level data, without the need to rely on time-consuming image-processing techniques. We then described the extraction of the gamestate vectors necessary to learn the human exemplar's weapon handling behaviour, and presented a neural network architecture designed to avail of them. A description of some experiments using this system, and some observations thereupon, closed this contribution.

FUTURE WORK

We intend to continue investigating and developing the model outlined in this paper, with the aim of gaining further insight into the observations made in the Experiments section above. We are also investigating approaches to imitating the human player's opponent-relative tactical movement while engaged in combat; that is, how he moves in relation to his opponent in order to minimise his chances of being hit while maximising his opportunities to strike. Together with the weapon-handling system detailed here, and the strategic navigation systems outlined in our previous work, we hope to ultimately produce a full imitation agent and test it in direct competition against human opponents.

REFERENCES

Bauckhage C, Thurau C. & Sagerer G 2003 "Learning Humanlike Opponent Behaviour for Interactive Computer Games", in *Pattern Recognition*, Vol 2781

Bauckhage C. and Thurau C. 2004 "Towards a Fair 'n Square Aimbot - Using Mixtures of Experts to Learn Context Aware Weapon Handling", In *Proc. GAME-ON*, pages 20-24

Dillmann, R., Kaiser, M. and Ude, A. 1995 "Acquisition of elementary robot skills from human demonstration" in *Proc International Symposium on Intelligent Robotic Systems (SIRS'95)* (pp. 1-38)

Fairclough, C., Fagan, M., MacNamee, B and Cunningham, P. 2001 "Research Directions for AI in Computer Games",. *Technical report, TCD*

Fod, A., Mataric, M., Jenkins, O. 2002 "Automated Derivation of Primitives for Movement Classification",. *Autonomous Robots 12(1)* 39–54

Gorman, B., Fredriksson, M. and Humphrys, M 2005 "QASE - An Integrated API for Imitation and General AI Research in Commercial Computer Games", In *Proc. 7th International Conference on Computer Games*, pgs201-207

Gorman, B. and Humphrys, M 2005 "Towards Integrated Imitation of Strategic Planning and Motion Modelling in Interactive Computer Games", In *Proc. 3rd ACM International Conference in Computer Game Design and Technology (GDTW 05)*, pages 92-99

Gorman B, Thurau C, Bauckhage C, and Humphrys M: 2006a "Bayesian Imitation of Human Behavior in Interactive Computer Games", In *Proc. of the Int. Conf. on Pattern Recognition (ICPR'06)*, volume 1, pages 1244-1247. IEEE, 2006.

Gorman B, Thurau C, Bauckhage C, and Humphrys M 2006b "Believability Testing and Bayesian Imitation in Interactive Computer Games", In *Proc. 9th Int. Conf. on the Simulation of Adaptive Behavior (SAB'06)*, pages 655-666, volume LNAI. Springer, 2006

Hayes G, Demiris J. 1994 "A Robot Controller Using Learning by Imitation" In: *Proc. of the 2nd Int. Symposium on Intelligent Robotic Systems* 198–204

Hollnagel, E. 1993 "Human reliability analysis: Context and control", *L:AP*

Jenkins, OC and Mataric, MJ "Deriving Action and Behavior Primitives from Human Motion Data". in *Proc. IEEE/RSJ IROS-2002*, pages 2551-2556

Laird, J. E. and v. Lent, M. 2000 "Interactive Computer Games: Human-Level AI's Killer Application", *AAAI*, pages 1171-1178.

Laird, J.E. 2001 "Using a Computer Game to develop advanced AI", *IEEE Computer*, pages 70 -75, July 2001.

Schaal, S. 1999 "Is Imitation Learning the Route to Humanoid Robots?" *Trends in Cognitive Sciences* 3(6) 233–242

Thurau, C., C. Bauckhage & G. Sagerer 2004b: "Synthesising Movement for Computer Games", in *Pattern Recognition*, Vol. 3175 of LCNS Springer