Towards Integrated Imitation of Strategic Planning and Motion Modelling in Interactive 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

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, in terms of both performance and believability.

In this paper, we describe our work in imitating the observed *goal-oriented behaviours* of a human player, based on concepts from data analysis and reinforcement learning. Since even the most intelligent artificial agent will be quickly identified as such if it is observed to move in a robotic manner, we also seek to incorporate mechanisms that will result in *believably humanlike motion*. We then present some illustrative examples, demonstrating the effectiveness of our model. Finally, we discuss future work in this field.

Keywords
Imitation learning, pattern recognition, artificial intelligence, clustering, statistical analysis, reinforcement learning.

1. INTRODUCTION

1.1 Imitation Learning and Games
Imitation learning is a field of pattern recognition wherein agents learn to perform complex procedures by first examining a demonstration of the task. Imitative techniques have been adopted by many researchers in the field of robotics, as a means of "bootstrapping" their machines' intelligence [6]. Despite the interest exhibited by the robotics community, however, very few attempts have been made to apply these principles to interactive computer games, with the notable exception of Thurau et al [7][8]. Given that modern games allow the recording of entire sessions, that vast online libraries of samples are in many cases already available, and that - rather than limb movement data or similar - these recordings encode the frame-by-frame behaviour of the player under rapidly-changing conditions and in competition with opponents of comparative skill, it becomes clear that games are an ideal platform for research in imitation learning.

From the industry's point of view, imitation learning would seem to offer more potential for future development than the outdated, symbolic systems upon which computer games have traditionally relied [2][3][4][5]. From the academic standpoint, commercial games provide a far greater range of challenges and opportunities than those games (chess and its ilk) which have typically been used in research The goal of imitation learning in computer games, then, is to construct models which explain and accurately reproduce the recorded behaviours, thereby 'reverse-engineering' the player's decision-making process from its observed results.

For our experiments, we opted to investigate the *first-person shooter* genre, due to the fact that it provides a relatively *direct* mapping of human decisions onto agent actions as compared with other genres. We ultimately chose iD Software's *Quake 2* as our test environment; it was prominent in the literature, and thanks to Laird [19] had become the de facto standard for research of this nature. **Figure 1** below shows a typical Quake 2 environment, with various features labelled. In order to extract the required data from Quake 2's recorded *DM2 demo* files (basically an edited copy of the network traffic received during the session) and to realise the in-game agents, we employ our own custom API.

2. METHODOLOGY

2.1 Overview
Our approach focuses on the tendency of competent players to cycle the environment in a strategic manner, collecting items to strengthen his character while denying them to other players; thus, we define the player's *goals* to be the *items* scattered at

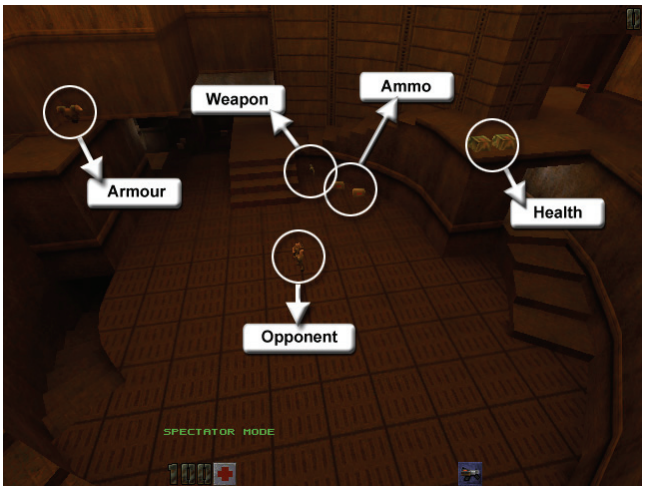

Figure 1 - Typical Quake 2 environment

fixed points around each level. Given a particular situation, the player attempts to collect weapons, ammunition or armor which will better his condition. By learning the mappings between the player's status and his subsequent item pickups, the bot can intelligently react to its current situation and will be capable of determining its *objectives*, allowing the agent to adapt observed strategies to situations which the player may not have faced.

2.2 Behavior Model

One of the first questions which arises when considering the problem of imitation learning is, quite simply, "what behaviours does the demonstration encode?" To this end, Thurau et al [7] propose a model of in-game behaviour based closely on Hollnagel's Contextual Control Model [15], shown in **Figure 2**.

Strategic behaviours refer to actions the player takes with long-term goals in mind; these include maximising the number of weapons or items he possesses, controlling certain areas of the map, and so forth. ***Tactical*** behaviours are mostly concerned with localised tasks such as evading or engaging opponents. ***Reactive*** behaviours involve little or no planning; the player simply reacts to stimuli in his immediate surroundings. ***Motion modelling*** refers to the imitation of the player's movement; in theory, this should produce *humanlike* motion along the bot's path, and should also prevent the agent from performing actions which are impossible for the human player's mouse-and-keyboard interface (instantaneous 180˚ turning, perfect aim, etc).

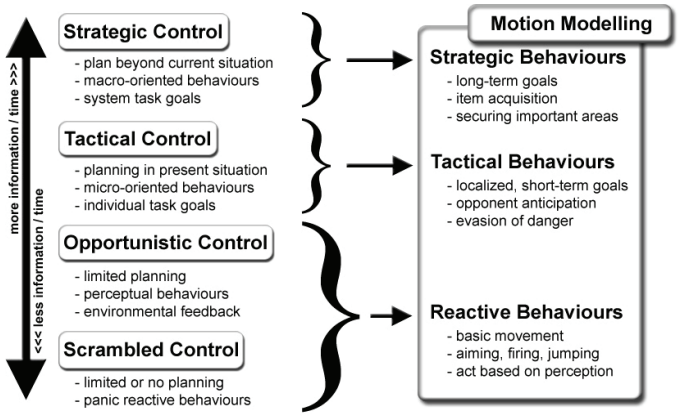

Figure 2 - Thurau's adaptation of Hollnagel's COCOM

In a number of contributions [13][14], the ability of agents to exhibit long-term strategic planning consistently emerges as a key factor in determining its "believability". In the context of Quake and similar games, equivalent importance must be given to the imitation of humanlike *motion*; an agent which does not exhibit the idiosyncrasies of a human player will easily be identified as a fake, regardless of how impressive its planning abilities are. Therefore, we concentrate on developing and integrating the ***strategic*** and ***motion modelling*** levels of the hierarchy.

2.3 Goal-Oriented Strategic Behaviour

In order to learn long-term *strategic* behaviours from human demonstration, we developed a model designed to emulate the notion of *program level imitation* discussed by Byrne and Russon [12]; in other words, to identify the demonstrator's *intent*, rather than simply reproducing his precise *actions*. Thurau et al [7] present an approach to such behaviours based on artificial potential fields; here we consider the application of reinforcement learning and fuzzy clustering techniques.

2.3.1 Topology Learning

As mentioned earlier, in the context of Quake, strategic planning is mostly concerned with the efficient collection and monopolisation of *items* and the control of certain important areas of the map. With this in mind, we first read the set of all player locations $\vec{l} = \{x, y, z\}$ from the DM2 recording, and the points are clustered to produce a reduced set of typical positions, called *nodes*. We developed a custom modification of Elkan's *fast k-means* [1] designed to treat the positions at which items were collected as *immovable "anchor" centroids*, thereby deriving a goal-oriented clustering of the dataset. By examining the sequence of player positions, we also construct an *n x n* matrix of edges *E*, where *n* is the number of clusters, and $E_{ij} = 1$ if the player was observed to move from node *i* to node *j* and 0 otherwise.

2.3.2 Deriving Movement Paths

Because the environment described above may be seen as a Markov Decision Process, with the nodes corresponding to states and the edges to transitions, we chose to investigate approaches to goal-oriented movement based on concepts from *reinforcement learning*, in particular the *value iteration algorithm*.

To do so, we first read the player's *inventory* from the demo at each timestep. In our experiments, we construct an inventory state vector of 18 elements, specifying the player's *health* and *armour* values together with the *weapons* he has collected and the amount of *ammo* he has for each. The set of unique state vectors is then obtained; these *state prototypes* represent the varying situations faced by the player during the game session.

We can now construct a set of *paths* which the player followed while in each inventory state. These paths consist of a series of transitions between clusters:

$$t_i = [c_{i,1}, c_{i,2}, ..., c_{i,k}]$$

where t_i is a transition sequence (path), and $c_{i,j}$ is a single node along that sequence. Each path begins at the point where the player enters a given state, and ends where he exits that state - in other words, when an item is collected that causes the player's inventory to shift towards a different prototype. **Figure 3** illustrates a typical path followed in one such prototype.

2.3.3 Assigning Rewards

Having obtained the different paths pursued by the player in each inventory state, we turn to reinforcement learning to reproduce his behaviour. In this scenario, the MDP's actions are considered to be the *choice to move to a given node from the current position*. Thus, the transition probabilities are

$$P(s' = j \mid s = i, a = j) = E_{ij}$$

where *s* is the current node, *s'* is the next node, *a* is the executed action, and *E* is the edge matrix.

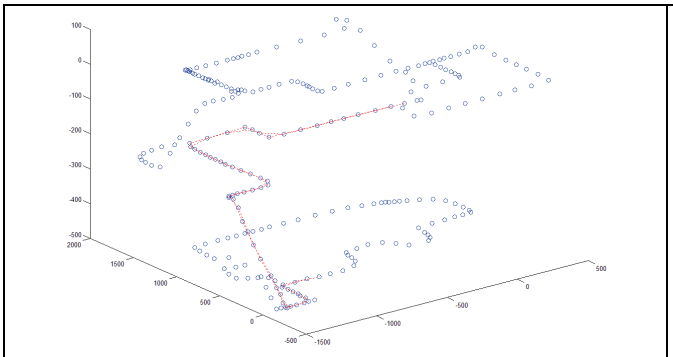

Figure 3 - An example of a path followed by the player while in a particular inventory state.

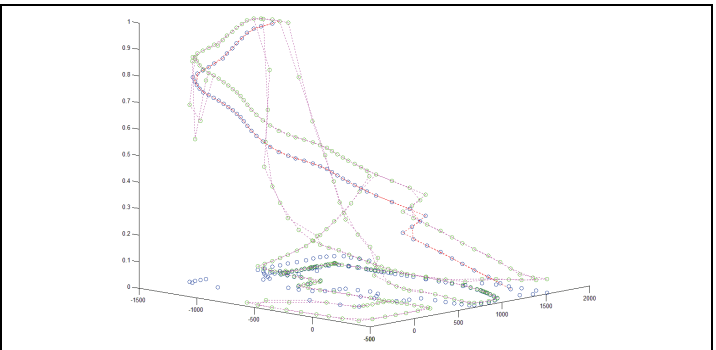

Figure 4 - Rewards (blue/red) and utilities (green/magenta) assigned to this path, as represented by the y-axis.

To guide the agent along the same routes taken by the player, we assign an increasing reward to consecutive nodes in every path taken under each prototype, such that

$$R(p_i, c_{i,j}) = j$$

where p_i is a prototype, and $c_{i,j}$ is the j^{th} cluster in the associated movement sequence. Each successive node along the path's length receives a reward greater than the last, until the final cluster (at which an inventory state change occurred) is assigned the highest reward. If a path loops back or crosses over itself en route to the goal, then the higher values will overwrite the previous rewards, ensuring that the agent will be guided towards the terminal node while ignoring any non-goal-oriented diversions. Thus, as mentioned above, the agent will emulate the player's *program-level* behaviour, instead of simply duplicating his exact actions.

2.3.4 Learning Utility Values

With the transition probabilities and rewards in place, we can now run the *value iteration algorithm* in order to compute the utility values for each node in the topological map under each inventory state prototype. The value iteration algorithm iteratively propagates rewards outwards from terminal nodes to all others, discounting them by distance from the reward signal; once complete, these utility values will represent the "usefulness" of being at that node while moving to the goal.

In our case, it is important that *every* node in the map should possess a utility value under *every* state prototype by the end of the learning process, thereby ensuring that the agent will always receive strong guidance towards its goal. We therefore adopt the *game value iteration* approach outlined by Hartley et al [17]; the algorithm is applied until all nodes have been affected by a reward at least once - that is, until every node has a non-zero utility value.

2.3.5 Multiple Weighted Objectives

Faced with a situation where several different items are of strategic benefit, a human player will intuitively *weigh* their respective importance before deciding on his next move. To model this, we adopt a *fuzzy clustering* approach. On each update, the agent's current inventory is expressed as a membership distribution across all prototype inventory states, based on its relative similarity to each. This is computed as follows:

$$m_p(s) = \frac{d(\vec{s}, \vec{p})^{-1}}{\sum_{i=1}^{P} d(\vec{s}, \vec{i})^{-1}}$$

where s is the current inventory state, p is a prototype inventory state, P is the number of prototypes, d^{-1} is an inverse-distance or proximity function, and $m_p(s)$ is the degree to which state vector s is a member of prototype p, relative to all other prototypes. The utility configurations associated with each prototype are then weighted according to the membership distribution, and the adjusted configurations *superimposed*; we also apply an *online discount* to prevent the possibility of backtracking. The formula used to compute the final utilities is thus:

$$U(c) = \gamma^{e(c)} \sum_{p=1}^{P} V_p(c) m_p(s)$$

$$c_{t+1} = \max_y U(y), y \in \{x \mid E_{c_t x} = 1\}$$

where $U(c)$ is the final utility of node c, γ is the online discount, $e(c)$ is the number of times the player has entered cluster c since the last state transition, $V_p(c)$ is the original value of node c in state prototype p, and E is the edge matrix. **Figure 4** shows the results of the value iteration process on a typical path.

2.3.6 Object Transience

Another important element of planning behaviour is the human's understanding of *object transience*. A human player intuitively tracks which items he has collected from which areas of the map, can easily estimate when they are scheduled to reappear, and adjusts his strategy accordingly. To capture this, we introduce an *activation* variable in the computation of the membership values; inactive items are nullified, and the membership values are redistributed among items which are still active:

$$m_p(s) = \frac{a(o_p) d(\vec{s}, \vec{p})^{-1}}{\sum_{i=1}^{P} a(o_i) d(\vec{s}, \vec{i})^{-1}}$$

where a, the *activation* of an item, is 1 if the object o at the terminal node of the path associated with prototype state p is present, and 0 otherwise. See the Experiments section for an illustration of the navigation system in operation.

2.4 Motion Modelling

As discussed earlier, the illusion created by even the most intricate planning behaviours will be shattered if the agent is observed to *move* in an unconvincing manner; it therefore becomes imperative to implement a mechanism capable of imitating the human player's *motion*. To do so, we draw on the theory of *action primitives*, as outlined by various researchers in the field of robotic imitation and behavioural science [9][10]. Action primitives are a basis set of modularised motor commands which are sufficient, through combination operators, for generating entire movement repertoires [11]. A means of deriving such primitives from demonstration in Quake was first proposed by Thurau et al [8]; we adopt the same approach, while modifying it to suit our needs.

We first read the sequence of *actions* executed by the player at each interval during the game session. In our current model, this results in a set of four-dimensional vectors as shown below:

v = [yaw, pitch, fire, jump]

That is, the angular orientation of the player in each plane, whether he is firing his weapon, and whether he is jumping. We then *cluster* the set, thereby aggregating similar actions executed at different times or locations, and reducing the data to a smaller set of *prototypical actions*, otherwise known as *primitives*.

In order to reconstruct the human player's motion, the agent must also learn how to *sequence* these primitives. To this end, we employ a series of *probability functions* similar to Thurau. In that case, however, the primitives were used in isolation; they were not underpinned by a dedicated navigation system. We must therefore adapt the approach, such that the *strategic planning* system outlined earlier is responsible for navigation around the environment, and the *motion modelling* system is *layered* on top.

Since the player's action depends on his current location in the environment, the probability of executing a particular action primitive v_i when the agent is within the Voronoi region of node w_k in the topological map was originally written as

$$P(v_i) = P(v_i \mid w_k)$$

However, in our new model, this could lead to situations where an incongruous primitive is chosen if the agent is currently at a node with more than one possible successor; for instance, if the navigation system chooses to move left, but the motion-modelling system chooses a primitive which the player executed while moving to the right. To constrain the motion model such that it adheres to the paths chosen by the navigation system, we choose primitives not based on the current *node w*, but rather on the *edge e* along which the agent is travelling; that is,

$$P(v_i) = P(v_i \mid e_k)$$

The conditional probabilities can be computed by re-examining the recorded data. Each action in the training set can be associated with an edge in the topological map, and it can be associated with an action primitive. By counting the number of times a particular action primitive is observed to be executed on a given edge, an *m x n* matrix is formed where *m* is the number of edges and *n* is the number of action primitives; the probabilities are then deduced by simply normalising the rows of the matrix. An entry at position *(k, i)* thus indicates the probability $P(v_i \mid e_k)$ as above.

Of course, not every action primitive can be executed as the successor of every other; there is a constrained ordering of primitives which defines the behaviours exhibited by the player. As such, it is necessary to define a condition which expresses the probability of performing a particular action given the previously-executed action:

$$P(v_i) = P(v_i \mid v_l)$$

where v_l is the action which was executed on the last time step. These probabilities are computed in a similar manner to the above; an *n x n* transition matrix is constructed, and populated by observing the sequence of actions from the demo. The overall probability of executing action primitive *i* is therefore given as

$$P(v_i) = \frac{P(v_i \mid v_l, e_k)}{\sum_{u=1}^{n} P(v_u \mid v_l, e_k)} = \frac{P(v_i \mid v_l) P(v_i \mid e_k)}{\sum_{u=1}^{n} P(v_u \mid v_l) P(v_u \mid e_k)}$$

Far from being a simple aesthetic effect, motion modelling also has an important *functional* element. If the human player reached a particular point in the level by performing some action other than simply running (e.g. jumping), then the navigation system alone will not be able to reach it. In other words, the goal-oriented navigation system defines *where* the agent needs to go, while the action primitives provide the *means* to get there. A good illustration of this is outlined in **Figure 6** below.

3. EXPERIMENTS

Having developed the behaviour model described above, we proceeded to carry out a number of experiments to verify its effectiveness. Experiments were designed to test both the *goal-oriented navigation* and *motion modeling* systems, as well as examining their interaction with one another.

3.1 Goal-Oriented Navigation

3.1.1 Pickup Sequences

To evaluate the agents' ability to learn the human's strategic behaviour, we first conducted 150 separate tests on 30 sequences of item pickups spanning four different game levels, some of which incorporated environmental features such as interaction with moving elevator platforms. For each set of five tests, the first attempt placed the agent at the same starting position as the player; the four subsequent tests placed the bot at a random position along the sequence. When starting at the same location as the player, the agent was expected to reproduce the exact sequence of pickups. When starting at each of the random positions, it was expected to *attempt* to reproduce the sequence if possible; however, if its starting location caused it to collect an object while en route to the first item in the sequence, the agent was expected to deal with this unencountered situation appropriately. Because these samples represented isolated pickup sequences rather than continuous gameplay, the agent did not take item respawns into account - this was tested separately on several more extensive samples, as described later. Also, to account for the random positioning of the agent, we made each internodal link in the topological map bi-directional. For each test, the number of seconds taken for the bot to complete its task was recorded, and can be compared with the human player's time. These results are summarized below.

	Dataset	Player	Bot (Same SP)	Random 1	Random 2	Random 3	Random 4
Map 1: q2dm1	Sample 1	24.8	14.2	25.4	36.7	23.6	20.1
	Sample 6	12.0	11.8	16.1	15.3	21.4	18.1
	Sample 7	15.8	10.8	11.1	10.8	12.6	15.9
	Sample 9	79.2	51.0	76.9	72.1	61.3	76.9
	Sample 10	60.8	47.1	46.8	71.7	50.9	72.5
Maps 2-3: q2dm5,7	Sample 11	32.8	43.1	56.0	68.7	58.2	65.7
	Sample 12	35.6	45.3	44.7	50.4	70.8	45.5
	Sample 13	28.4	37.3	42.9	48.7	56.7	44.5
	Sample 18	54.4	20.6	31.2	22.7	20.4	37.2
	Sample 19	20.8	12.7	43.3	12.9	43.5	20.7
	Sample 20	50.4	29.3	76.9	31.0	33.4	31.2
Map 4: q2dm8	Sample 21	20.0	17.3	15.8	13.5	11.2	12.2
	Sample 22	38.4	51.3	43.0	40.1	59.8	47.4
	Sample 26	20.2	24.6	22.8	22.7	30.2	24.0
	Sample 27	28.8	37.9	55.8	35.2	35.6	52.1
	Sample 30	24.8	29.6	28.3	30.0	34.0	35.1

As can be seen, the agent successfully completed each task, although with greatly varying times. When the agent starts at the same point as the human, the times are generally quite similar; in some cases, the agent actually manages to complete the collection sequence faster than the human, since it will (as noted above) ignore any non-goal-oriented movements present in the gameplay sample. The larger deviations observed in the random tests are due to the fact that the agent often started at around the midway point of the sequence, requiring it to collect all subsequent items before doubling-back to pick up the remainder. Some minor issues were noted - very occasional repetition of paths, rough motion between waypoints, both caused mostly by the artificial addition of bidirectional edges - but overall the bot showed impressive performance and adaptability to unseen circumstances.

3.1.2 Continuous Gameplay

While useful for a numerical comparison, however, these isolated pickup sequences do not fully test the agent's planning abilities. A more instructive examination of its performance can be made by supplying more extensive gameplay samples, wherein the player is observed to cycle continuously around the map in accordance with his changing state. We therefore trained the agent on four extended demonstrations across two different levels, where the human player had more scope to develop the kind of elaborate strategy typically seen in a live game session. Here, the *direction* in which the player traverses the map is often an important element of his strategy, allowing him to collect weapons before their relevant ammunition, or to build up armor reserves before entering the open areas of the level; consequently, edges in the topological map are taken as being unidirectional, except in cases where the player was explicitly observed to move along them in both directions. The objective-weighting and item transience mechanisms resulted in the emergence of a noticeably more complex, more involved long-term planning behaviour. Rather than simply cycling the map from one pickup to the next, the agents were instead observed to react dynamically to changes in an object's activation, to concentrate on areas of the map which were favoured by the human player over the course of his demonstration, and to give added significance to regions which were more heavily populated with beneficial items. For instance, in several samples the player was observed to first obtain a full complement of weaponry, after which he would patrol specific areas of the map which contained many of the most useful items; the agent took this into account by concentrating on those same areas throughout its test run.

The only issue noted during these tests was an occasional 'indecisiveness' at points where multiple paths intersected - that is, the agent would sometimes move circuitously among several neighbouring nodes, due to a small spike in the overlapping utility values. In such cases, however, the impasse was quickly resolved by the online discount, resulting in minimal disruption to the agent's movement.

Illustrative examples of the weighting and activation functions are shown in **Figure 5** below. It is important to note, however, that these microcosmic examples are not as important on the wider scale as they may first appear. While it is fortunate that some items happened to respawn while the bot was in the vicinity, thus providing some excellent demonstrations of the mechanisms involved, the very fact that the agent was able to resolve these reappearances in the short-term negates their long-term planning importance. Additionally, although it is tempting to view the agent's behaviour as a response to immediate visual stimuli, it bears re-stating that the agent is internally tracking the times at which the inactive items are scheduled to respawn - it therefore perceives these reappearances no differently than it does the reactivation of items in other areas of the map. The more significant implications of this mechanism occur when the agent realizes that items at medium or greater distances from its current

position have respawned, leading it to adjust its long-term strategic goals accordingly. A full gameplay example is shown and briefly discussed in **Figure 7**, in the appendix.

3.2 Motion Modelling

In order to test the functional and aesthetic aspects of the motion-modelling system, we ensured that the recorded demonstrations adhered to certain criteria:

1. some items were collected specifically because they required the human player to interact with the environment - that is, in order to collect the same item, the agent would need to learn and perform similar actions.
2. the human player exhibited certain behavioural traits in the course of the recording; the agent was expected to adopt similar mannerisms in its play.

In each case, the effectiveness of the motion-modelling system in capturing the distinct style of the human's play was observed. Because the collection of items mentioned in point 1 above requires the motion-modelling system to *augment* the standard navigation system - for instance, by jumping across a chasm - the experiments also test these systems' ability to interact with one another. Some examples of the resulting agent are illustrated in **Figure 6** below, which notes both the humanlike visual appearance of the bot's movement, and its ability to perform the actions necessary to negotiate such environmental challenges.

4. CONCLUSION

In this paper, we first described an approach to generating an explicitly goal-oriented topological map from observation data. We then proposed a modified value iteration approach as a means of learning the paths traversed by the player, in order to find a mapping from the player's current state to his subsequent strategy. We further outlined a mechanism based on *fuzzy clustering* which allows the agent to pursue multiple weighted goals in parallel. A system which imitates the human player's comprehension of object transience and his ability to dynamically re-evaluate his current strategy was also presented. We then discussed an approach to the extraction and sequencing of *action primitives*, which allow the agent to manifest humanlike idiosyncrasies as it pursues its strategic goals. Finally, a series of experiments designed to demonstrate the model's effectiveness were described.

Agents were observed to very accurately convey the impression of being a human rather than artificial player. In the short term, the imitation of human *motion* - particularly such nuances as visually examining an upcoming obstacle, which a computer-controlled character would never need to do - gave strong indicators that this was more than a simple AI player. In the longer-term, the emergence of a cohesive, consistent *strategy* further reinforced that initial view. Preliminary, informal questioning of independent observers confirmed these findings; we intend to fully verify this by conducting a detailed believability study in the near future.

5. FUTURE WORK

One of the shortcomings of the current system is its inherently *disjoint* nature - that is, the motion modeling system is layered on top of the strategic-planning system, with no communication and only minimal relation between them. The planning module picks the next edge to traverse, and the motion module picks the actions on that basis, with no regard for the agent's condition or objectives. While this does result in humanlike motion along demonstrably strategic paths, it is far from ideal. We are therefore investigating, in conjunction with Christian Thurau, an alternative system based on Christian's adaptation of Rao et al's Bayesian model of action sequencing [18]. This formulates the probability of each action as a function of not only the previous action, but also the current state and goal; it is therefore a natural fit for the strategic system described here. Not only will this provide us with a *forward* model to determine the next action to be executed, but it will also act as an *inverse* model, allowing us to determine the probability that the agent is pursuing each goal by simple observation of its action sequence.

Figure 5 - Two sequences showing the objective-weighting and item transience mechanisms. Top, the agent (white circle) returns to a previously-visited point before some ammo items (red circles) have respawned (1.1), and since they are inactive it initially passes by (1.2); however, their sudden re-emergence (1.2) causes the utilities to reactivate, and the agent is drawn to collect them (1.3) before continuing (1.4). In a different session, the agent is moving along a path (2.1) as a row of armour shards respawns behind it (2.2). It reacts at once, turning around to collect the items (2.3), before resuming its course (2.4).

Figure 6 - Examples of the *aesthetic* (top) and *functional* (middle) aspects of the motion-modelling system; the bottom sequence demonstrates a combination of both. The top sequence shows the agent *leaning into* and *strafing around* a corner, as a human player does. In the middle, the agent's next goal is an item on top of the box. As it approaches, it looks downwards, jumps, and fires a rocket to propel itself upwards. This so-called *rocket jump* is considered an advanced move and is commonly employed by players to reach inaccessible areas. Bottom, the agent interacts with a lift by standing still as it ascends (functional) and then jumping off at the top, an unnecessary action which is nonetheless common among human players (aesthetic).

6. ACKNOWLEDGMENTS

This work is kindly sponsored by the Irish Research Council for Science, Engineering and Technology's EMBARK initiative.

7. REFERENCES

[1] Elkan, C. "Using the Triangle Inequality to Accelerate k-Means", Proceedings of 20th ICML 2003

[2] Laird, J. E. and v. Lent, M. (2000). Interactive Computer Games: Human-Level AI's Killer App AAAI, 1171-1178

[3] C. Fairclough, M. Fagan, B. MacNamee, and P. Cunningham. Research Directions for AI in Computer Games. TCD, 2001.

[4] A. Naraeyek. Computer Games - Boon or Bane for AI Research. Künstliche Intelligenz, pages 43-44, February 2004

[5] D Charles, S McGlinchey, "The Past, Present and Future of Artificial Neural Networks in Games", CGAIDE 2004: p167

[6] S. Schaal. Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences, 3(6):233-242, 1999

[7] Thurau, C., C. Bauckhage & G. Sagerer 2004a: Learning Humanlike Movement Behaviour for Computer Games, in Proc. 8th SAB Conf.

[8] Thurau, C., C. Bauckhage & G. Sagerer 2004b: Synthesising Movement for Computer Games, in Pattern Recognition, Vol. 3175 of LCNS

[9] OC Jenkins and MJ Mataric, "Automated Derivation of Behavior Vocabularies for Autonomous Humanoid Motion", Proceedings of the Second AAMAS (2003)

[10] Thoroughman KA, Shadmehr R: Learning of action through adaptive combination of motor primitives. Nature 407: 742-747

[11] A. Fod and M. Mataric and O. Jenkins: Automated Derivation of Primitives for Movement Classification, Autonomous Robots,12 (1), pp. 39--54, 2002

[12] R.W. Byrne, A.E. Russon, "Learning by Imitation: A Hierarchical Approach", Behavioral & Brain Sciences 21

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

[14] Wallace, SA, Laird, JE: Behavior Bounding: Toward Effective Comparisons of Agents & Humans IJCAI 2003: 727-732

[15] Hollnagel, E. (1993) Human reliability analysis: Context and control. London: Academic Press

[16] D Livingstone & S McGlinchey, "What Believability Testing Can Tell Us", Proceedings of CGAIDE 2004: p273

[17] T Hartley, Q Mehdi, N Gough, "Applying Markov Decision Processes to 2D Real-Time Games", CGAIDE 2004: p55-59

[18] RPN Rao, AP Shon, AN Meltzoff, "A Bayesian Model of Imitation in Infants and Robots", Cambridge University Press 2004

[19] J.E. Laird, J.C. Duchi, Creating human-like synthetic characters with multiple skill-levels: a case study using the Soar quakebot, in: Proceedings of the AAAI Fall Symposium Technical Report, 2000.

8. APPENDIX

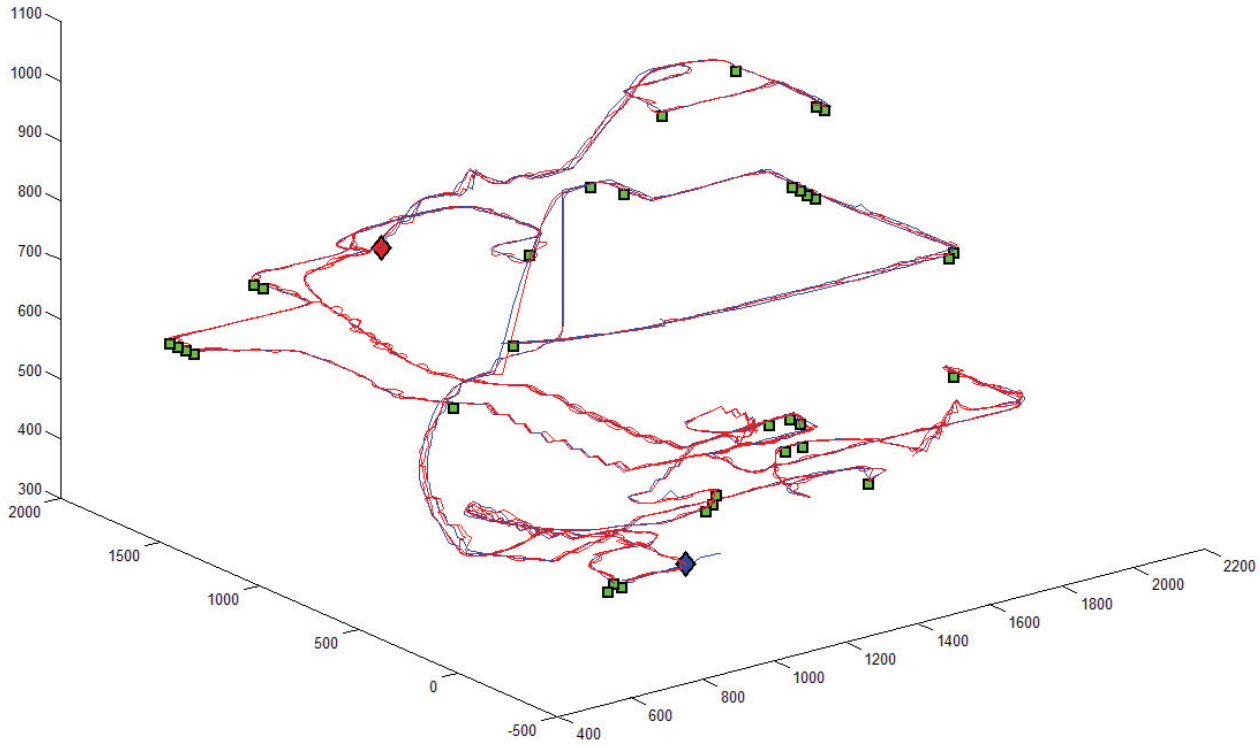

Figure 7 - One of the continuous gameplay samples; the agent (red) successfully reproduced the observed behaviour (blue) - including interaction with an elevating lift platform, represented by the vertical line in the centre of the visualisation - having started at a different point (red diamond) than the demonstrator (blue diamond). Because of this, it was not possible to simply reproduce the exact sequence of pickups - instead, the agent was forced to rely on its objective-weighting mechanism.