meIRL-BC: Predicting Player Positions in Video Games

Inge Becht
University of Amsterdam
Intelligent Systems Laboratory Amsterdam
The Netherlands
inge.becht91@gmail.com

Sander Bakkes
University of Amsterdam
Intelligent Systems Laboratory Amsterdam
The Netherlands
s.c.j.bakkes@uva.nl

ABSTRACT

In this paper we demonstrate how behaviour-classification models can improve player position prediction for video game AI. To this end, we propose a novel method named *meIRL-BC*, which (1) uses maximum-entropy Inverse Reinforcement Learning for the creation of position prediction models [15], and (2) predicts player positions based on estimates of their most likely behavioural roles in the game (e.g., attacking, defending, ambushing, etc.). Experiments that test meIRL-BC in an actual Capture the Flag video game yielded the following three results. First, depending on the behavioural role to be classified, the prediction accuracy of meIRL-BC approximates, slightly improves, or substantially improves upon that of meIRL. Second, the accuracy of the trained behavioural classifier is presently insufficient for classifying actual game-player behaviour. Third, surprisingly, despite use of an ineffective classifier, both meIRL-BC and meIRL yielded comparable overall performance in actual gameplay. What is more, when in actual gameplay the percentage of correctly classified instances was above approximately 40%, meIRL-BC substantially outperformed meIRL is terms of win/loss/draw ratio, the number of bots that were successfully killed, and the absolute number of flags that were captured. Also, of all AIs in competition with the standard AISandbox AI, meIRL-BC was able to capture opponent flags more rapidly, thereby generally achieving a higher game score than meIRL. From these results, we can draw the conclusion that when the meIRL-BC method is correctly applied in video-game AI, it indeed outperforms meIRL in actual gameplay.

Keywords

Predicting Player Positions, Tactical Video Games, Maximum-entropy Inverse Reinforcement Learning

1. INTRODUCTION

A common challenge for game AI of modern video games, is to accurately model an (opponent) player on the basis of imperfect information. That is, traditionally game AI were equipped with full information on the game environment – and the whereabouts of the (opponent) game players – in order to increase its challenge level without much effort [4]. This practice was shown to generally not offer an entertaining experience to the player [14], as the challenge level of the AI was not based on skillful tactics, but rather on cheating tactics; by having complete knowledge of e.g., the game map and all player positions, an unfair advantage is created towards the human player. As a result, academic game AI is increasingly focussing on non-cheating game AI that is equipped solely with realistically available, albeit imperfect information.

For many game genres, predicting player positions is an important capability of the game AI; one which heavily impacts the AI's effectiveness. However, accurate position prediction in a realistic setting, based solely on the aggregated observations of individual game-playing agents, remains an open issue [1]. Although some approaches have investigated the automatic generation of position prediction models for video games [11, 4], generally no discrimination is made between distinct behaviours that occur in a game. Indeed, behaviour and position can be considered closely correlated in many video games. For instance, in a capture the flag game mode, players that are consistently located nearby their team's flag likely have adopted a defensive role, while players that are consistently located nearby the enemy's flag likely have adopted an offensive role. One may therefore argue that when such distinct behaviours are not taken into account when modelling player behaviour, the resulting accuracy of position predictions will be limited [4].

As such, the contribution of this paper is enhancing an established method for position prediction in video games (i.e., maximum entropy Inverse Reinforcement Learning, or: meIRL) [11] with a capability for Behavioural Classification (meIRL-BC). The enhancement is achieved by creating a position-prediction model for each distinct behavioural role that bots may exhibit in a video game. We will validate the enhanced method in an actual capture the flag (CTF) game environment, namely AISandbox.

2. RELATED WORK

Position prediction of video-game players

Research into position prediction of video-game players is still in its relative infancy. At present, some contributors to the field are Hladky and Bulitko [4], Weber *et al.* [14], and Laird [6].

Hladky and Bulitko [4] compared two different prediction methods with regard to (1) the accuracy of predictions, and (2) the human-likeness of predictions. The two compared methods are hidden semi-Markov models and particle filters. Experiments in a CTF game revealed that the hidden semi-Markov models were the most accurate and made the most human-like predictions (i.e., the errors were most human-like). The research however focuses strictly on comparative measurements, and does not investigate implementing the methods in actual game-playing bots.

Weber *et al.* [14] employed particle filters too, but rather than investigating the human-likeness of position predictions, their work investigated whether employing such a filter can enhance the effectiveness of game AI. Experiments in which a learned particle model is employed by a Star Craft game-playing AI showed a 10 per cent improvement in game-playing performance. However, they make simplifying assumptions for their particle filters, i.e., incorporating constant trajectory and linear decay functions for particles, that may not hold in general. A possible improvement in this regard, would be to utilise movement models that incorporate qualitative spatial reasoning [3].

Finally, Laird [6] investigated how reasoning over player observations can allow a bot to anticipate a player's actions. Specifically, Laird implemented a number of anticipation strategies into a Quake bot, to be executed only at an estimated high probability of accurately anticipating the player behaviour. That is, when an anticipating strategy is triggered, the bots attempt to predict the behaviour of the observed player, by reasoning how itself would have behaved in the observed player's situation. Though it is stated that these additions of anticipating strategies are beneficial for the performance of the AI, no experimental results are given that support this claim. Still, in terms of enhancing prediction performance, the intuition of building upon anticipatory models may still be regarded an important contribution.

IRL in video games

The above mentioned position predictors employ a direct approach to modelling the opponent's behaviour. In contrast, the inverse reinforcement learning (IRL) [9] approach attempts not to learn a model of the opponent's behaviours, but assumes that the opponent is making rational decisions on the basis of an underlying Markov decision process (MDP) [10], and tries to learn this MDP from demonstrations of the opponent's behaviour. The opponent's policy is then found with MDP planning methods.

In Tastan *et al.* [11], predictive models were established using meIRL. Here, inverse reinforcement learning is used to learn a value map of the environment and to generate a maximum entropy distribution over the opponent's likely paths. In the work, meIRL – of which the theoretical foundation is discussed in Section 3 – is applied for the goal of intercepting opponents in the video game Quake. The work builds upon a particle filter that integrates meIRL position prediction models (*cf.* Ziebart *et al.* [15]), and is compared with baseline Brownian prediction models that randomly spread particles over the map. The predictive models that use meIRL had a significantly smaller tracking error in predicting the position of opponent players. Also, when employed in actual game AI, the meIRL method was more likely to successfully intercept the opponents bots.

Lee and Popovic [7] utilise IRL for transfer learning of distinct player behaviour. Their approach attempts to learn the optimal policy belonging to observed player behaviour, assuming the Markov model state-space is deterministic and discrete. A similar approach is taken by Tastan and Sukthankar [12], who utilise IRL for learning when game bots should attack, explore and target opponents in a human-like manner. This is implemented by deriving policies from human-expert playing demonstrations. Experimental results with this approach were promising, as the bots were shown to act more human-like (according to a jury of human participants). A remaining obstacle of the suggested approach, however, is that multiple policies can be learned from a single demonstration by using IRL, making the selection of an appropriate policy not a straightforward choice.

Our own research is closely related to more recent work of Tastan *et al.* [11], which learns position-prediction models effectively by applying meIRL. We believe that this approach can be further enhanced by learning unique prediction models per distinguished behaviour in the game (as opposed to learning a single model over all observed players in the game). Our intuition is that our approach provides enhanced expressive power on the whereabouts of individual game players. A formalisation of the approach is discussed next.

3. THEORETICAL FOUNDATION

To provide context for the reader, we provide a short introduction on (1) inverse reinforcement learning (IRL), and (2) how it can be constrained to find the maximum entropy solution (meIRL). For a more detailed discussion on these topics, and its relationship to reinforcement learning, we refer the reader to Ziebart *et al.* [15].

To utilise IRL for generating prediction models, training trajectories are required that are an adequate representation of the model that needs to be learned, expressed in movement actions a and state positions s. When learning the model from the training trajectories, a value map is created that for each position contains the probability that it will be visited by the expert agent. The creation of the model by IRL can be written as the following Bellman equation [11].

$$V(s) = \max_{a_s}\{R(s, a_s) + \sum_{s'} P(s'|s, a_s)V(s')\} \tag{1}$$

Here, $V(s)$ is the value of each state when applying a rational (optimal) policy, and $P(s'|s, a_s)$ is the probability of ending up in state s' when applying action a in state s. Video-game environments can be abstracted to be deterministic in nature, and thus only have value 1 or 0 for the summation over s' in case $a \in A(s)$, where $A(s)$ is the set of actions that can be taken at state s. $R(s, a_s)$ is the reward function, which is unknown in case of IRL. The reward function is the product of a weight vector and a vector of features in which each state is expressed, to create a position invariant expression of positions on the map, of which the relative importance is learned by adjusting the weights accordingly. The weights converge by applying the forward-backward pass algorithm, as described in [15, 11].

Because in practice IRL is an under-constrained problem, there are multiple possible optimal rewards that fit the demonstrated expert behaviour [12]. For tasks like ours this typicality is undesired. Maximum entropy IRL (meIRL) addresses this by computing the maximum entropy solution to the IRL problem, using the following equation:

$$V(s) = \sum_{a_s} P(a_s|s)\{R(s,a_s) + \sum_{s'} P(s'|s,a_s)V(s')\} \quad (2)$$

The maximum entropy solution assumes that the natural distribution of trajectories is the distribution that maximizes the entropy, i.e., the diversity is maximized. This is a desirable property in our problem domain, because behaviour observed in typical modern video game environments is inherently non-deterministic.

4. APPROACH

Here, we discuss meIRL-BC; our approach to predicting player positions in video games. Its contribution is enhancing an established method for position prediction in video games (i.e., maximum entropy Inverse Reinforcement Learning, or: meIRL) [11] with a capability for Behavioural Classification (meIRL-BC). The enhancement is achieved by creating a position prediction model for each distinct behavioural role that bots may exhibit in a video game.

Figure 1 schematically illustrates our approach. The figure depicts two offline components that are activated before the start of a game: (A) the position prediction modeller (4.3) that creates a position prediction model for each behaviour (4.2), and (B) the construction of the behaviour classifier (4.4). The classifier is applied each game loop for each opponent in case it is witnessed by one of the meIRL-BC bots. In case the opponent is not witnessed, a final component is activated; the position propagator (4.5).

4.1 Game environment

Though our approach to predictive modelling is not limited to a specific video game, its implementation is domain specific; the different behaviours that need to be classified and the features used for creating the motion model will vary for each video game. In this paper the typical Capture The Flag (CTF) game mode is used for implementing meIRL-BC. CTF is a tactical game mode typically found in first-person shooter games, in which two teams attempt to capture each others flag, while attempting to prevent the other team from succeeding by firing at the opponent team. When delivering a captured flag to a certain position on the map, a point is scored. The team with the most points after a certain amount of time wins the game.

Our experiments will take place in AISandbox [2], a CTF game environment and prototyping toolkit that is specifically designed for creating new multi-agent AI systems. Our approach will be implemented in Terminator [13], an existing game AI that has been specifically designed for CTF games. This game AI has already successfully competed in the 2013 Capture the Flag AI competition.

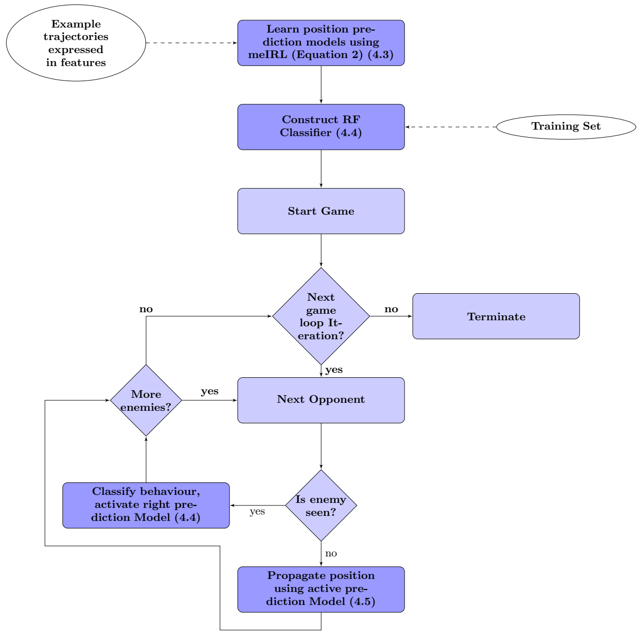

Example trajectories expressed in features
Learn position prediction models using meIRL (Equation 2) (4.3)
Construct RF Classifier (4.4)
Training Set
Start Game
Next game loop Iteration?
no
no
Terminate
yes
More enemies?
yes
Next Opponent
Classify behaviour, activate right prediction Model (4.4)
yes
Is enemy seen?
no
Propagate position using active prediction Model (4.5)

Figure 1: A schematic representation of all components of meIRL-BC. The ellipses indicate input data, the rectangles indicate actions, and the diamonds indicate if-statements. The *dark blue* elements are specifically constructed for meIRL-BC – the numbers indicate the subsection in which they are discussed.

4.2 Behaviours

Naturally, for creating a motion model for each distinct behaviour that can be observed in a game (i.e., behavioural roles that players may take upon themselves), one first needs to decide on which behaviours need to be modelled. We have chosen to model behaviours that are generic to a CTF game mode, thereby acknowledging that behaviours that our own bots exhibit, are also likely to be exhibited by the opponent bots (*cf.* [6])[1]. As a result, we have decided to model the following six bot behaviours.

Attack flag. Moving to an opponent's flag to capture it from its spawning location. This generally is a single trajectory from the bot spawn location to the spawn location of the opponent's flag.

Defend flag. Moving to ones own flag for defending purposes, killing enemies that come nearby. This behaviour is generally centred around a few defensible locations in the map.

Carry flag. When the opponent flag is captured, a bot carries it to its score position. Players exhibiting this behaviour generally attempt to rapidly traverse a safe path to the score position.

Assist flag carrier. Here a player assists the flag carrier with safely traversing to the score position. Generally a path similar to that of the flag carrier is taken.

Ambush. Players may place ambushes that generally are positioned on places that contain points of interest, e.g., defensible points nearby the flag score location.

[1]An alternative albeit less generally applicable approach would be to model a team's overarching play style in a game. This has been investigated previously by Matsumoto and Thawonmas [8].

Stalk. In this behaviour, when a player knows the position of an opponent while this opponent is not aware of the player, the player stalks the opponent until it is in shooting range.

4.3 Position prediction models

For creating position prediction models, meIRL (Equation 2) requires training trajectories in which the behaviours described above are exhibited in actual gameplay (as illustrated in Figure 1). In our approach, these training trajectories are derived from observing the TERMINATOR bot operating in a self-play CTF match; by logging every bot position for each iteration of the game loop, and hand-labelling the behaviour that was exhibited. Subsequently, the observed player positions are discretized into a grid, so that the number of possible states is finite, as is required by the meIRL method. In our implementation, the grid is equal in size as the employed game map (i.e., 88 × 50 tiles).

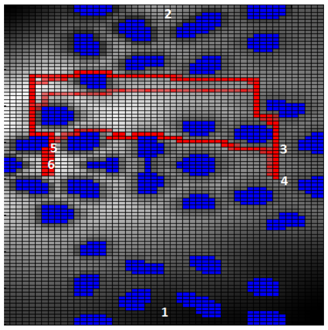
The features that are used for training the reward function in meIRL, are based on previous research by Tastan *et al.* [11]. In this work, the distance to specific points on the map – that are deemed important for that specific game environment – are used as features. We adopt the same approach, but tailored the features to the CTF game. The resulting seven features are: (1) shortest distance to opponent bot spawn, (2 shortest distance to friendly bot spawn, (3) shortest distance to opponent score position, (4) shortest distance to friendly score position, (5) shortest distance to friendly flag spawn location, (6) shortest distance to opponent flag spawn location, (7) visibility of game environment, as expressed by how many grids of the game world are visible to the game AI.

While gathering training trajectories (according to the procedure described above), features values for all seven features are logged. Together with the raw positional observations and collected labels, this provides input for creating the actual position prediction models via Equation 2. In our approach, adequate weights for the features are determined by the forward-backward pass algorithm over 500 iterations (*cf.* Tastan *et al.* [11]).

To ensure convergence in our relatively large state-space, without decreasing effective resolution, we train multiple position prediction models for a single behaviour by using chunks of the complete trajectories, and subsequently combine the chunks by taking the means over the combined models. Specifically, the trajectories were divided into smaller chunks until convergence of the meIRL algorithm took place, making the chunk sizes range anywhere between five and ten subsequent positions. Subsequently, the prediction models that were generated using these smaller trajectories were combined into the final prediction model by calculating the mean of each position on the grid, by summating all probabilities of a single position, and dividing the summation by the number of chunks that were employed. Figure 2 illustrates two position prediction models that were learned by the meIRL method for the AISANDBOX CTF game.

4.4 Behaviour classification

Recall that in this research we posit that distinguishing between distinct behaviours in a game can enhance the accuracy of position predictions. To use behaviour-based prediction models, instead of straightforwardly applying a single model throughout a game, a decision needs to be made on which learned behavioural model best describes observed opponent behaviour (i.e., behavioural classification). In our implementation of meIRL-BC, this classification is made each time an opponent is witnessed by the game AI that uses meIRL-BC. The procedure is detailed in Algorithm 1; it incorporates a position propagation subroutine (Algorithm 2) as discussed in Subsection 4.5.



(a) Prediction model for the behaviour Carry flag

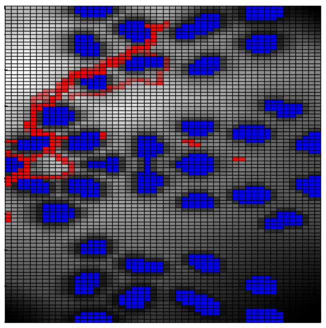

(b) Prediction model for the behaviour Defend flag

Figure 2: Two learned position prediction models. Training trajectories are indicated in red, the map obstacles in blue, and probable player positions for the trajectories in white shades (improbable positions in dark shades).

ALGORITHM 1. Position Prediction - Main algorithm

```
for Every iteration of the game loop do
    for All enemy agents do
        if Agent is seen then
            Classify behaviour;
            Activate relevant position prediction model for this agent
        else
            if Agent is alive then
                Execute PropagateStep() using the active position
                prediction motion model
            else
                Clear propagation steps
            end
        end
    end
    Use prediction data
end
```

To learn such a behavioural classifier, as with the previous components, requires a set of training instances. Given that the task of behavioural classification is distinct from position prediction, the set of adequate features is different too. Following from our domain knowledge of the game environment, we have decided on the following five features: (1) the agent's position expressed in point-of-interest distances (analogous to those utilised in the positions prediction models), (2) the orientation of the agent, (3) the game state - being either: both flags are at their spawn point, both flags not at spawn point, friendly flag not at spawn point, opponent flag not at spawn point, (4) the distance to the closest opponent, (5) is the bot presently observing an opponent bot (boolean), and (6) environment visibility. As such, we collect feature data by observing a self-play match in which the TERMINATOR AI is exhibiting all six behaviours described in Subsection 4.2; thereby generating feature data from over 6000 instances.

The adopted classification method in our approach is ran-

Algorithm 2. Position Prediction - PropagateStep routine

```
Data: $T \leftarrow$ Set of tuples, containing trajectories ($t$) and their probabilities;
$O \leftarrow$ Preferred Directions by agents;
$s \leftarrow$ Propagation Step;
$MM \leftarrow$ Active motion model: probability map;
Result: $T_{new} \leftarrow$ new Trajectories
1  for $(t, probability)$ in $T$ do
2      $position \leftarrow$ getLastPosition($trajectory$);
3      $possiblePositions \leftarrow$ possiblePositions($position$);
4      $total \leftarrow 0$ ;
5      for $possiblePos$ in $possiblePositions$ do
6          $total \leftarrow total + MM[possiblePos]$
7      end
8      for $possiblePos$ in $possiblePositions$ do
9          $p \leftarrow$ calcProb( $position, total, MM$);
10         if $p \in O$ then
11             $probability_{new} \leftarrow probability \times (p + (\frac{1}{s} * p))$
12         else
13             $probabilty_{new} \leftarrow probabilty \times p$
14         end
15         $t_{new} \leftarrow$ Add $possiblePos$ to $t$;
16         Add $(t_{new}, probability_{new})$ to $T_{new}$ ;
17         Sort $T_{new}$ on highest probability;
18     end
19 end
20 Discard $T_{new}[i]$ if i > trajectoryMaximum
```

Algorithm 3. Position Prediction - CalcProb routine

```
Data: $position \leftarrow$ Position for which to find probability;
$total \leftarrow$ Total probability of surrounding tiles;
$MM \leftarrow$ Active motion model
Result: $prob \leftarrow$ Tile traverse probability
1  $prob \leftarrow \frac{MM[position]}{total}$;
```

Algorithm 4. Terminator trajectory labelling

```
Data: $nrVisited \leftarrow$ Matrix consisting of every tile on board
1  for every iteration of the game loop do
2      for all enemy agents do
3          if agent is alive and just seen then
4              $(x, y) \leftarrow$ position of enemy;
5              $(prevx, prevy) \leftarrow$ position of enemy at previous sighting;
6              $path \leftarrow$ shortestPath($x$,$y$, $prevx$, $prevy$);
7              for $(xpos, ypos)$ in path do
8                  $nrVisited[xpos][ypos] + +$
9              end
10         else
11             continue;
12         end
13     end
14     Use new iterated values for determining the best course of actions.
15 end
```

dom forest decision-tree classification [5]. The classifier operates by creating multiple decision trees which all cast a vote when an instance requires a classification. The advantage of this method, is that it does not solely output a classification, but returns a probability distribution of the classification.

4.5 Position propagation

Finally, a method is required that propagates position predictions of unobserved game players to future game iterations. Analogous to previous work into particle filters [14, 11, 4], in our discrete-spaced game environment we probabilistically propagate candidate positions in the available directions of player movement. The precise algorithm adopted for position propagation is given in Algorithm 2; it incorporates a subroutine for determining position probabilities (Algorithm 3). Note that the algorithm propagates positions of a single game player.

Each time an opponent game player is not observed, the n most likely trajectories are propagated a single step, by looking at the last candidate position (Algorithm 2, Line 2), in the four possible movement directions (i.e., up, down, left, right), or fewer in case of obstructions (Line 3). In our implementation, we propagate a maximum of twenty likely trajectories. At the first propagation step the content of the set of trajectories, will be a single tuple with the position of the opponent where it was last spotted, with a probability of one. When a propagation step is applied to this single position, at most four new trajectories are made. For these new trajectories, the new probability is the product of the previous probability and the probability for entering this new state (Line 13).

The motion model gives for each position the absolute probability that it is will be visited. Directly utilising these absolute prediction values is undesirable for predicting the trajectories, as we are interested in the probability of any of these new positions being chosen by the opponent player, relative to its currently estimated position. To this end, we divide each probability of the possible positions by the summation of the probabilities of all subsequent states (Line 6). Note that in the algorithm a small bias is introduced for grid tiles that an opponent player was facing when observed, as this is a likely movement direction for that player (Line 11).

5. EXPERIMENTS

Experimental setup

In our experiments, we enhance the Terminator AI with our meIRL-BC method. This enhancement is set-up as follows. The Terminator AI utilises a straightforward histogram approach to position prediction (Algorithm 4), in which when an opponent player is observed at a certain grid position, it increases the probability of observing the opponent on this position as well as all tiles on the shortest path from this position to that of the previous sighting of this particular opponent. These histograms are subsequently utilised for calculating safe routes and suitable ambush positions. Naturally, however, when opponents are not observed for several iterations, the histogram approach yields highly inaccurate position predictions, as the shortest routes between observations are less and less likely to reflect actual player trajectories.

Our enhancement is straightforward in that instead of heuristically determining likely trajectories, the learned position prediction models and position propagation method are now utilised (*cf.* Section 4), thereby extending the original histogram approach. The enhancement is described in Algorithm 5. Here the *path* (Line 4) that is used to update the histogram in each iteration now is determined by the most probable trajectory at that moment, as calculated by the propagateStep routine described in Algorithm 2. The update of the trajectories and their probabilities (Line 11) is as described in Algorithm 1.

Performance evaluation

As meIRL-BC consists of several components, to validate its performance we perform the following three evaluations: (1) the accuracy of the position predictions, (2) the accuracy of the behavioural classifier, and finally (3) the effectiveness of meIRL-BC in actual gameplay.

All experiments take place in the form of actual gameplay matches. For all Terminator AI's (i.e., the original Terminator, Terminator enhanced with meIRL – which only uses a single prediction model for all opponents, and Terminator enhanced with meIRL-BC) we perform a competition. In each competition the AI is pitted against the standard

Algorithm 5. Terminator meIRL-BC trajectory labelling

```
Data: $nrVisited$ ← Matrix consisting of every tile on board
for every iteration of the game loop do
    for all enemy agents do
        if agent is alive then
            $path$ ← most likely trajectory for the agent;
            $(posx, posy)$ ← last position of $path$;
            $nrVisited[xpos][ypos]++$;
        else
            continue;
        end
    end
    Update trajectories and probabilities;
    Use new iterated values for determining the best course of actions;
end
```

AISandbox AI (10 matches), and the two other Terminator AI's (2×40 matches). Matches against the standard AISandbox AI are included to assess how well meIRL-BC performs when pitted against an AI that does not exhibit the exact same behaviours as observed in the training phase. For performance evaluation, in each iteration of a game match, observations are gathered on the position and behaviour of each player, and the predicted class and position of the opponent players. All matches take place under counter-balanced starting conditions on the same map on which the prediction models were learned.

Results

Position prediction accuracy

Figure 3 illustrates the accuracy of position predictions, as expressed in the obtained prediction error (i.e., the Euclidean distance between the predicted position and the actual position); a lower prediction error is better. Figure 3a illustrates the prediction error of Terminator enhanced with the original meIRL. Figure 3b-f illustrate the prediction error of Terminator enhanced with meIRL-BC for specific classified behaviours. Note that in the case when meIRL-BC position prediction models were employed, the learned trajectories were only utilised when the player behaviour was classified, so that the resulting prediction error is solely dependent on the performance of the model, and not on the behavioural classification.

The obtained results reveal that over twenty propagation steps, the mean prediction error for meIRL-BC: Attack flag and meIRL-BC: Defend flag is substantially lower (better) than that of meIRL. In addition, the results reveal that over twenty propagation steps, the mean prediction error for meIRL-BC: Ambush and meIRL-BC: Assist flag carrier is slightly lower (better) than that of meIRL, and the prediction error for meIRL-BC: Carry flag approximates that of meIRL. Moreover, while the prediction error of meIRL does not seem to converge, the prediction error of meIRL-BC: Attack flag and meIRL-BC: Defend flag appears to converge at a relatively low prediction error (i.e., 10). Note that the behaviour that is observed most often is a player defending the flag, and that the least observed behaviour is a player stalking another bot (observed only once – as such no mean prediction error figure is plotted). To show statistical significant of these results, we applied a sample-paired t-test on the mean propagation error of meIRL in combination with the mean propagation of each meIRL-BC behaviour. Using a significance value of 5% we can reject the hypothesis that the means for both data sets is the same in the case of the behaviours meIRL-BC: Attack, meIRL-BC: Defend and meIRL-BC: Deliver flag.

Classification accuracy

The accuracy of the learned random forest classifier, using 10-fold cross validation applied to the set of training instances, amounted to 72%. However, in actual gameplay in the 40 matches against Terminator enhanced with meIRL, a classification accuracy of only 27.5% was obtained, indicating that the trained classifier was heavily over-fitted to the set of training instances. The ineffectiveness of the learned classifier is revealed as well in the confusion matrix (Table 1). Each row of the matrix represents the actual behaviour that was observed in the gameplay instance, while each column of the matrix represents the behaviour which was predicted.

Effectiveness in actual gameplay

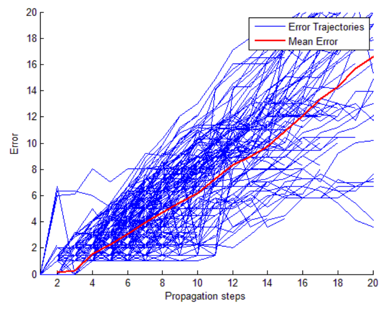
Surprisingly, given the inaccurate behavioural classifier that is employed in meIRL-BC, both meIRL-BC and meIRL are comparable in terms of actual gameplay performance. This is revealed in Table 2, which gives the absolute performance measured in the three performed competitions. Both in terms of win/loss/draw ratio, as well in terms of the total flags captured/lost ratio, meIRL-BC and meIRL do not substantially differ. Also note that the original Terminator AI consistently outperforms both meIRL approaches – presumably due to lower-level AI of the Terminator AI not yet effectively utilising the learned position-prediction models.

What is more, Figure 4 reveals that when in actual gameplay the percentage of correctly classified instances is above approximately 40%, meIRL-BC substantially outperforms meIRL is terms of win/loss/draw ratio (Figure 4a), the number of bots that were successfully killed (Figure 4b), and the absolute number of points that were scored (i.e. flags that were captured) (Figure 4c). These results indicate that when meIRL-BC is correctly applied, it indeed outperforms meIRL in actual gameplay. Finally, Figure 5 illustrated the average point score plotted against the score time for the three Terminator AIs in competition with the standard AISandbox AI. It reveals that meIRL-BC captures flags more rapidly than meIRL. Thereby, it generally achieves a higher game score than meIRL.

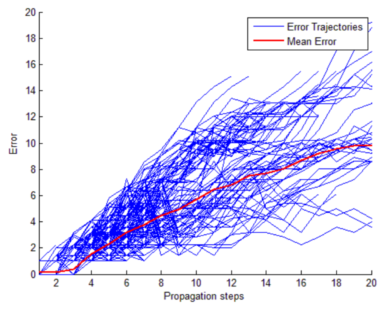
6. DISCUSSION

In experiments that test meIRL-BC we observed that, dependent on the behavioural role that was to be classified, meIRL-BC approximated the prediction accuracy of meIRL (1x), achieved a slightly improved prediction accuracy (2x), or achieved a substantially improved prediction accuracy (2x). However, in some cases, outliers were observed with a relatively high prediction error. We believe that such outliers cannot be avoided due to the inherent randomness that is typical to video games. For instance, in the investigated CTF game, (un)lucky circumstances may suddenly alter the trajectory of a player (e.g., it suddenly is being chased by an opponent player). Naturally, the prediction models may be further refined by incorporating features that can reduce the number of prediction outliers.

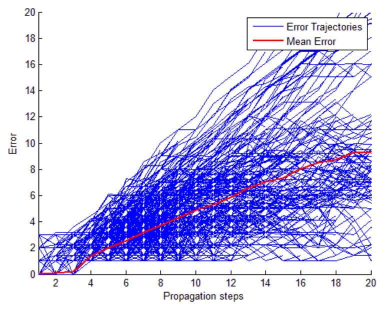
The aspect of generalisation was briefly investigated by pitting meIRL-BC against a game AI whose behaviour it had never observed in the training phase (the standard AISandbox AI). The obtained results given in Table 2 revealed that



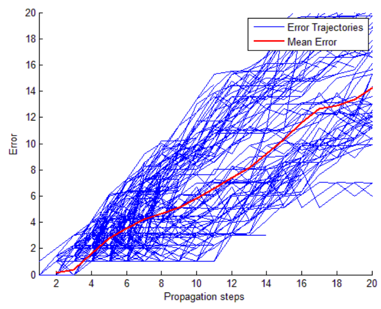
(a) meIRL



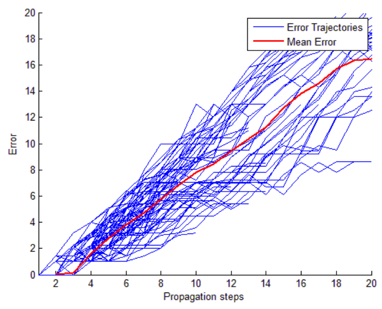
(b) meIRL-BC: Attack flag



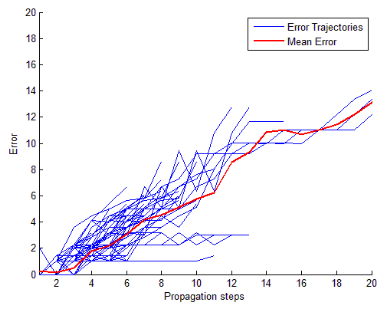
(c) meIRL-BC: Defend flag



(d) meIRL-BC: Ambush



(e) meIRL-BC: Carry flag



(f) meIRL-BC: Assist flag carrier

Figure 3: Prediction error of meIRL and the meIRL-BC enhancement (a lower prediction error is better).

Table 1: Confusion matrix for all classifications performed in forty games of meIRL-BC versus meIRL.

	Attack flag	Defend flag	Ambush	Assist flag carrier	Carry flag	Stalk
Attack flag	1781	689	1776	888	508	22
Defend flag	8617	7528	6208	6750	4001	120
Ambush	1905	216	2069	576	231	8
Carry flag	495	329	568	378	1507	40
Assist flag carrier	302	432	276	472	386	3
Stalk	112	102	72	56	51	1

meIRL-BC consistently outperformed the standard AISandbox AI. While this comparison is skewed with respect to the AI's being distinct in terms of lower-level AI, the obtained results give rise to the idea that the modelled behaviours (attacking, defending, carrying flag, etc.) are general enough to adequately model behaviour of different, previously unobserved opponent AIs.

7. CONCLUSION AND FUTURE WORK

In this paper we demonstrated how behaviour-classification models can improve player position prediction for video game AI. To this end, we proposed a novel method named *meIRL-BC*, which (1) uses maximum-entropy Inverse Reinforcement Learning for the creation of position prediction models [15], and (2) predicts player positions based on estimates of their most likely behavioural roles in the game (e.g., attacking, defending, ambushing, etc.).

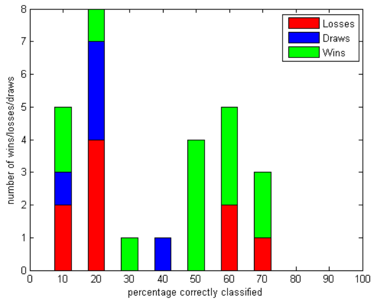
Experiments that test meIRL-BC in an actual Capture the Flag video game yielded the following three results. First, depending on the behavioural role to be classified, the prediction accuracy of meIRL-BC approximates, slightly improves, or substantially improves upon that of meIRL. Second, the accuracy of the trained behavioural classifier is presently insufficient for classifying actual game-player behaviour. Third, surprisingly, despite use of an ineffective classifier, both meIRL-BC and meIRL yielded comparable overall performance in actual gameplay. What is more, when in actual gameplay the percentage of correctly classified instances was above approximately 40%, meIRL-BC substantially outperformed meIRL is terms of win/loss/draw ratio, the number of bots that were successfully killed, and the absolute number of flags that were captured. Also, of all AIs in competition with the standard AISandbox AI, meIRL-BC was able to capture opponent flags more rapidly, thereby generally achieving a higher game score than meIRL.

From these results, we can draw the conclusion that when the meIRL-BC method is correctly applied in video-game AI, it indeed outperforms meIRL in actual gameplay. For future work, we will investigate how to improve behavioural classification by considering the sequential nature of player behaviour (i.e., behaviour is generally consistent over several time-steps).

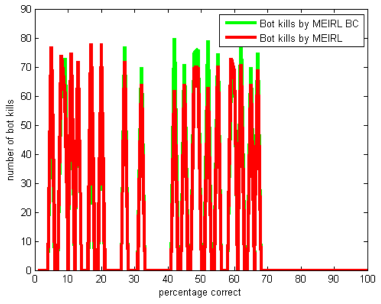
Acknowledgement. We gratefully acknowledge Diederik Roijers' contribution to the paper's theoretical foundation.

Table 2: Overall gameplay performance as measured in three competitions.

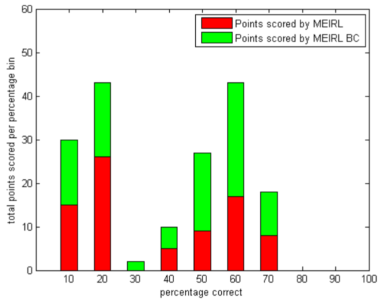
Terminator vs. ...	Win	Draw	Loss	Total matches	Total flags captured	Total flags lost
Standard AI	10	0	0	10	97	2
Terminator meIRL	26	7	7	40	185	113
Terminator meIRL-BC	27	10	3	40	153	95
Terminator meIRL vs. ...	**Win**	**Draw**	**Loss**	**Total matches**	**Total flags captured**	**Total flags lost**
Standard AI	9	1	0	10	46	3
Terminator	7	7	26	40	113	185
Terminator meIRL-BC	12	12	16	40	98	113
Terminator meIRL-BC vs. ...	**Win**	**Draw**	**Loss**	**Total matches**	**Total flags captured**	**Total flags lost**
Standard AI	10	0	0	10	58	1
Terminator	3	10	27	40	95	153
Terminator meIRL	16	12	12	40	113	98



(a) Wins/losses/draws of meIRL-BC



(b) Bot kills by meIRL-BC



(c) Point (flag) scores of meIRL-BC

Figure 4: Game wins, bot kills, and point scores of meIRL-BC as plotted against the classification error.

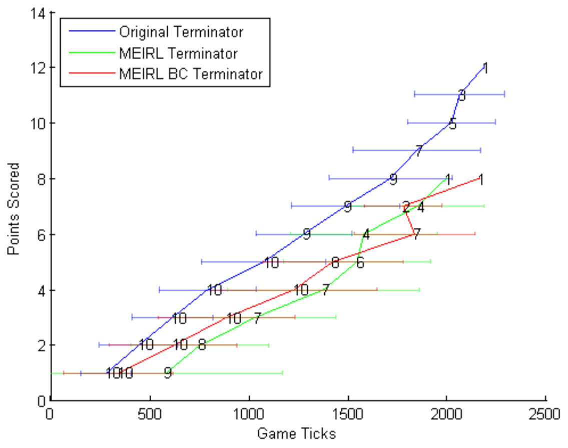


Figure 5: Average point score plotted against the score time (in game ticks) for the three Terminator AIs. The data labels indicate the number of matches in which the concerning number of points were scored.

8. REFERENCES

[1] A. Champandard. Open challenges in first-person shooter (FPS) AI technology, April 2011.

[2] A. Champandard, P. Champandard-Pail, P. Dunstan, R. Kogelnig, K. Lord, and M. F. Brandstetter. The AI Sandbox, 2013.

[3] K. D. Forbus, J. V. Mahoney, and K. Dill. How qualitative spatial reasoning can improve strategy game ais. *Intelligent Systems, IEEE*, 17(4):25–30, 2002.

[4] S. Hladky and V. Bulitko. An evaluation of models for predicting opponent positions in first-person shooter video games. In *IEEE CIG 2008*, pages 39–46, 2008.

[5] T. K. Ho. Random decision forests. In *Proc. of the 3d Int. Conf. on Document Analysis and Recognition*, pages 278–282. IEEE, 1995.

[6] J. E. Laird and M. Van Lent. It knows what you're going to do: Adding anticipation to a quakebot. In *Proc. of the 5th Int. Conf. on Autonomous Agents*, pages 385–392, 2001.

[7] S. J. Lee and Z. Popović. Learning behavior styles with inverse reinforcement learning. *ACM Transactions on Graphics (TOG)*, 29(4):122, 2010.

[8] Y. Matsumoto and R. Thawonmas. MMOG player classification using hidden markov models. In *Entertainment Computing - ICEC 2004*, pages 429–434. Springer, 2004.

[9] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proc. of the 7th Int. Conf. on Machine Learning*, pages 663–670, 2000.

[10] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[11] B. Tastan, Y. Chang, and G. Sukthankar. Learning to intercept opponents in first person shooter games. In *IEEE CIG 2012*, pages 100–107, 2012.

[12] B. Tastan and G. Sukthankar. Learning policies for first person shooter games using inverse reinforcement learning. *Proc. of the AIIDE 2011*, pages 85–90, 2011.

[13] L. Verhaard. Terminator bot, 2013.

[14] B. G. Weber, M. Mateas, and A. Jhala. A particle model for state estimation in real-time strategy games. In *Proc. of the AIIDE 2011*, page 103–108. AAAI Press, 2011.

[15] B. D. Ziebart, A. Maas, J. A. D. Bagnell, and A. Dey. Maximum entropy inverse reinforcement learning. In *Proc. of AAAI 2008*, July 2008.