2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Learning a Super Mario Controller from Examples of Human Play

Geoffrey Lee
School of Computer Science and Information Technology
RMIT University, Australia
geoff.lee@rmit.edu.au

Min Luo
School of Computer Science and Information Technology
RMIT University, Australia
s3195794@student.rmit.edu.au

Fabio Zambetta
School of Computer Science and Information Technology
RMIT University, Australia
fabio.zambetta@rmit.edu.au

Xiaodong Li
School of Computer Science and Information Technology
RMIT University, Australia
xiaodong.li@rmit.edu.au

***Abstract*—Imitating human-like behaviour in action games is a challenging but intriguing task in Artificial Intelligence research, with various strategies being employed to solve the human-like imitation problem. In this research we consider learning human-like behaviour via Markov decision processes without being explicitly given a reward function, and learning to perform the task by observing expert's demonstration. Individual players often have characteristic styles when playing the game, and this method attempts to find the behaviours which make them unique. During play sessions of Super Mario we calculate player's behaviour policies and reward functions by applying inverse reinforcement learning to the player's actions in game. We conduct an online questionnaire which displays two video clips, where one is played by a human expert and the other is played by the designed controller based on the player's policy. We demonstrate that by using apprenticeship learning via Inverse Reinforcement Learning, we are able to get an optimal policy which yields performance close to that of an human expert playing the game, at least under specific conditions.**

I. Introduction

The game industry has been rapidly expanding for the past few decades and it is the fastest-growing component of the international media sector. It has been devoting considerable resources to design highly sophisticated graphical content and challenging and believable Artificial Intelligence (AI). Various artificial intelligence methods have been employed in modern video games to engage players longer, game agents built with human-like behaviour and cooperation, which raise the players' emotional involvement and increase immersion in the game simulation.

To better develop human-like behaviour in game agents, an AI technique called imitation learning has been developed which allows for the AI to learn from observation. It was originally applied with success to robot manufacturing processes [1]. Preliminary work on imitation learning has been focused on the task of motion planning for artificial opponent in first-person shooter games [2], but modelling game AI through imitation learning is seen to have great potential for more games than just first-person-shooters.

A particularly fitting challenge for imitation learning has been posed by the Super Mario Turing Test AI competition [3] [4] whose goal is to develop an artificial controller that plays Super Mario in a human-like fashion. With this challenge in mind, this paper presents work on realising a controller for the game Super Mario by applying Apprenticeship Learning via Inverse Reinforcement Learning (IRL) [5], a high-performance method of imitation learning.

Imitating player behaviour has several key benefits: We can create games with more intelligent and believable NPCs[1], and opponents that do not react to players in a pre-determined fashion, regardless of context [6]. Game play can be dynamically altered to adapt to different players by their features of play (their playing "style" as well as their "skill") to sustain their engagement with the game longer [7], and learned AI agents are able to help game companies test the strength of game AI and discover defects or limitations before its release to the market [8].

In order to implement and test the application of IRL to a Super Mario controller, this paper investigates how we can design an appropriate knowledge representation for the Super Mario testbed environment. We investigate the MDP framework and inverse reinforcement learning algorithms that allows us to learn a controller via imitation learning. We also address objective methods to evaluate our controller's performance, and how to best evaluate if our controller's behaviour is human-like.

Our contributions include providing a solution to represent knowledge in Super Mario game within an MDP framework, on which we applied apprenticeship learning via IRL. We show two experiments using self-reporting that forms the basis of a "Super Mario Turing Test", and we provide an experimental analysis of why AL/IRL holds promise for providing human-like controllers and eventually passing this modified Turing test.

The rest of this paper is organized as follows: Section II describes the basic techniques and algorithms which serve as foundations, and brief current state of the art in human-behaviour modelling. Section III is our proposed framework used to generate our experimental results. In Section IV we discuss experimental results, analysing the convergence results for apprenticeship learning and present and discussing quantitative results of questionnaires. Finally, Section V summarises our results and lays out opportunities for future work.

II. Background and Related Work

A. Mario, AI and Turing Tests

Super Mario has been used to study human-like behaviour since 1990's. In 1992, John and Vera used GOMS (Goals, Operations, Methods and Selection rules) to predict the behaviour of an expert in Super Mario Bros [9]. By using the only information in the booklet and some hand coded heuristics,

[1] Non-Player Characters

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

they carried out their experiment at two levels, the functional-level operators and keystroke-level operators, and proved that GOMS is capable of predicting the expert's behaviour and the expert's behaviour is objective oriented.

From 2009 to 2012 a yearly AI competition was held in which teams developed AI controllers to play Infinite Mario, with the intention to win as many levels as possible[2]. The various entries to this competition utilised all manner of machine learning and evolutionary programming techniques. The clear victor in the competition was the A* algorithm [10], which due to Mario's deterministic nature, allowed for perfect prediction and timing. The algorithm successfully navigated all challenges in increasingly difficult levels. One defining point was that the algorithm, by performing so well, played in a manner that performs impossible feats, appearing completely unnatural to the human eye. Due to the dominance of the A* algorithms over the competition, an advanced test track was implemented with the goal to create a controller that exhibited human-like behaviour.

In the Turing test track of Mario AI Championship[3], the most common methods used in the competition were hand coded rules, supervised learning and indirect representation (using fitness values as a guide to evolution) [4]. Supervised learning, neuroevolution via Artificial Neural Networks (ANN), directed scripting and behaviour trees all featured in the competitors algorithms. The most successfully 'humanlike' controller employed neuroevolution, which differs from our approach as it applies evolutionary algorithms to adjust the weights of an ANN.

Based on John and Vera's work with GOMS, Mohan and Laird proposed a hierarchical RL framework that allows learning playing Infinite Mario [11]. They used object-oriented behaviours to achieve object related goals and by selecting primitive actions corresponding to the object. The limitation of their learning approach is that the agent is not able to find a good policy. In [11] they assigned rewards for each positive interaction with the world, which while proper for traditional reinforcement learning, is different from what we do in this project. The agent learns to play the game to maximize it's score, rather than learning to play like a human. We instead approach the problem by defining that the environment is filled with unknown rewards, which can be derived (as a linear combination function) by learning from observing an expert at work.

Our approach differs significantly in that we do not deal with evolutionary algorithms and genomes. Our approach, by applying IRL to the problem, allows us to utilise reinforcement learning to maximize the reward over time, where the reward is behaving in a humanlike fashion. The main difference being that the algorithm will take a less optimal solution if it predicts larger gains in the future. To the best of our knowledge, our approach is the first attempt to use apprenticeship learning via IRL to the Turing Test challenge.

B. Reinforcement Learning

Reinforcement Learning (RL) is an area of machine learning inspired by behaviourist psychology and has been studied in areas such as genetic algorithms, statistics, game theory, control theory and checkers [12]. An RL agent learns by selecting actions that yield the greatest accumulated numerical reward over time rather than seeking an immediate reward.

In machine learning, the environment is typically represented as a Markov Decision Process (MDP) and decisions made by the agent are called a policy π (a probability distribution for selecting actions at each state). The goal of the agent is to find the optimal policy $\pi*$, a policy that maximizes accumulated rewards over time [12].

RL proceeds by trying actions in a particular state and updating an evaluation function which assigns expected reward values to the available actions based on the received, possibly delayed, reward. Dynamic Programming (DP) is the selected RL algorithm for this paper, as was used in the original implementation of the IRL problem. More recent versions may use alternative methods. See [12] for more details on the DP algorithm.

C. Apprenticeship Learning via Inverse Reinforcement Learning

When teaching a young child to play a game, it is much easier and more practical to demonstrate it to the child rather than listing all the game rules. Then the child will learn by trying to mimic the demonstrator's performance and grasp the task gradually. Learning from an expert by watching, imitating, or from demonstration is called *apprenticeship learning* (AL) [5], which employs IRL to solve such a problem.

Inverse Reinforcement Learning (IRL) problems work in the opposite way to reinforcement learning - when there is no explicitly given reward function, we can use an IRL algorithm to derive a reward function by observing an expert's behaviour throughout the MDP environment. Rewards are mapped to features within the states to reflect the importance of those features to the expert. This analysis of expert behaviour yields a policy that attempts to perform in a manner close to the expert. MDPs without reward functions are defined as MDP\R [5].

For this project, we invited three experts to demonstrate gameplay tasks to the Apprenticeship Learning algorithm. To observe the performance requires logging the experts' actions in related MDP states (further details in Section III-A). While Mario has some solid goals to associate with rewards (coins, points, goombas stomped) it is not obvious how an expert player would prioritize each goal. In this scenario there is no way to get a reward function directly, so it becomes a MDP\R problem to learn the expert's internal reward function.

The method Abbeel et al. [5] used to derive a reward function in MDP\R was by thinking of an expert as an agent trying to maximize a hidden reward function, which can be expressed as a linear combination of known features from the environment:

$$R^*(s) = w^* \cdot \phi(s) \tag{1}$$

where ϕ is a vector matrix of features, w^* is a vector which specified the relative weights between these features corresponding to the reward function.

[2]http://www.marioai.org/LearningTrack

[3]http://www.marioai.org/turing-test-track

The apprenticeship learning algorithm is described in detail by Abbeel and Ng [5]. The algorithm solves a linear programming problem by bringing the policy $\tilde{\pi}$ of a RL algorithm close to π_E, the optimal policy performed by expert.

We assume that π_E is the optimal policy performed by expert based on an unknown reward function. By analysing the prevalence of each feature in the expert's environment every timestep, we can calculate the feature expectations of the expert $\mu(\pi_E)$. The AL algorithm iteratively calculates feature weightings w via a max-margin calculation to create reward functions, which are subsequently used by a RL algorithm to learn the optimum policies. The feature expectations of these policies $\mu(\tilde{\pi})$ are compared to the expert's feature expectations, and the process is repeated until polices are generated that are comparable within an acceptable error rate (ϵ), where $\epsilon > 0$.

$$\|\mu(\pi_E) - \mu(\tilde{\pi})\| < \epsilon \tag{2}$$

A policy with a matching feature expectation performs actions in a way that produces the same state transitions as that of the expert. The result is not only a policy that mimics the expert, but also one where the underlying reward structure can be extrapolated to states previously unexplored by the expert, allowing for behaviour that performs as the expert would in unfamiliar situations.

III. IRL for Super Mario

In this section, we describe how we applied AL/IRL to our project. We explain the construction of the MDP and features, based on the human player style and Mario's interaction the environment, and describe how we applied the learned policies to the Super Mario game.

The process of designing a controller in our proposed framework is comprised of 3 distinct steps. First, we record and export the expert trajectories from the game. Then we set up the environment and run the AL and IRL algorithms. The optimal policy for Super Mario is obtained at the end of this step. Finally the obtained policy is applied directly to the modified version of Super Mario.

A. MDP Representation and Features Definition

In the Turing Test Track, the standard Mario AI Benchmark used a grid to represent the current state of the environment. In line with previous attempts we have used a 5×5 grid to describe the world around Mario (see Figure 1).

To limit the scope of our preliminary investigation, some constraints were put on the game. Mario's size was fixed to "big" without any possibility of shooting. There are two types of enemies adopted in the levels, Goombas and Green Koopas. At this stage of the project, we chose overground without gaps as our basic level type. Additionally, the usage of a 5×5 grid representation is actually described as a partially observable Markov Decision Process (POMDP), as we only view small window of the world at once. This will affect the transition probabilities from state to state as outside influences will affect the state in ways the MDP cannot predict. The ability of the algorithm to adapt to new situations is part of the advantage

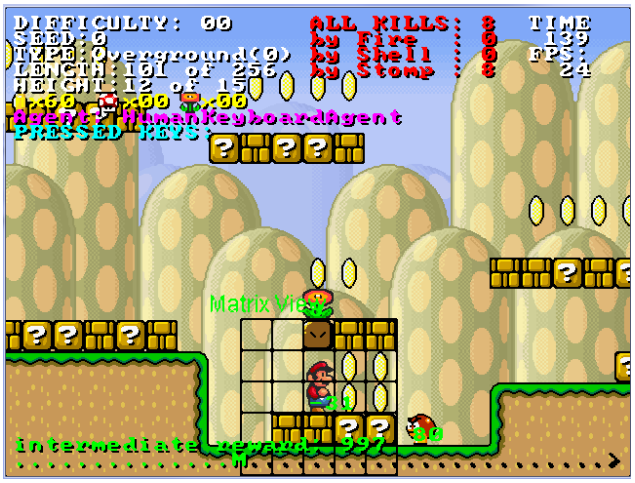
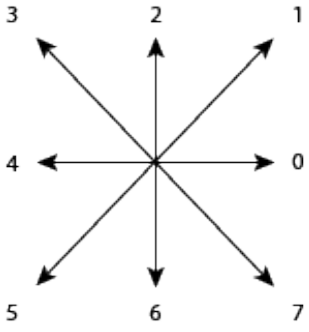


Fig. 1. Using a 5×5 grid to represent the MDP state, the origin is located in the upper left corner and Mario is in the middle of the grid and occupies two cells (position [1,2] and [2,2]). Assuming there are 4 possible items: coin, enemy, platform and brick appearing in the grid, we would have 4^{23} states overall.

IRL provides, as it learns behaviours based on values of the features observed, allowing it to act like the learnt behaviour even in environments the player has not been before.

There are 15 possible objects that may appear in any level. With 15 objects, the level would create at most 15^{23} states in the state set S, as Mario occupies 2 cells in the middle of the grid. Fortunately almost all the MDP states are sparse matrices, because a background cell counts as an empty cell, thus the number of the state elements in S is far less than 15^{23}. During the game Mario and the various enemies are not fixed to a particular block position, so their positions are rounded to the nearest block position. This discretization creates some variance in the state transition probabilities as sometimes the same action (for example, jumping to stomp a goomba) may not always result in the same end state depending on the actors' distance to transitioning between blocks.

Data is recorded sparsely, by the column, row, and item ID. For example, Figure 1 can be expressed as: 0,2: -4;(a mud brick) 0,3:-16;(a brick) 0,4:-16; 1,3:2;(a coin) 1,4:2; 2,3:2; 2,4:2; 3,1:-16; 3,2:-16; 3,3:-16; 3,4:-16; 4,0:-128;(a platform) 4,1:-128; 4,2:-128; 4,3:-128; 4,4:-128;.



3	3	2	1	1
4	4		0	0
4	4		0	0
5	5	6	7	7
5	5	6	7	7

Fig. 2. Defining direction in a 5×5 grid.

We also gathered 20 significant features of gameplay by watching human demonstrations and analysing Mario's

interaction with the MDP environments. These values include noting the direction of Mario, if he is jumping, the closest block, enemy, coin, etc. For features that describe the relative position of an object to Mario, a specific value was assigned relating to the direction of Mario to the object in the grid. In a small 5×5 grid, the direction is a more effective indicator compared to small displacements of distance between blocks. Directions were defined by "slicing" the grid into 8 discrete directions, shown in Figure 2. The 23 pieces of block data plus the extended feature data combines to form each unique state in the MDP.

The basic actions in the gameplay simply mirror keyboard presses. There are four keys: A, S, ←, →, which indicate Sprint, Jump, Move left and Move right respectively. Often players press more than one key at any given time. For correctly catching the player's actions, key presses are represented as binary digits and obtained as a binary string. We denote actions as a combination of four digits, which represent boolean values of Move left, Move right, Jump and Sprint, from left to right. For example, 0110 means that Mario is jumping to the right and 1001 indicates that Mario is sprinting to the left.

B. Data Gathering and Pre-processing

The data of the state-action trajectories of an expert is captured by a log file, and consists of 4 parts: the time, action, states and features. Time is expressed in milliseconds; actions and features have been explained previously.

For the sake of legibility, we create a state mapping file. The list of possible transitions can be easily created based on the actions taken in a particular state. As previously mentioned, in an MDP [12], at each discrete time step t, given a state s and any action a, the probability of transition to the next state is denoted as:

$$p^a_{ss'} \;=\; P_r\{\; s_{t+1} = s' \mid s_t = s,\; a_t = a \;\},$$

To calculate the probability $P^a_{ss'}$, we use the formula shown as below:

$$p^a_{ss'} \;= \frac{The\ number\ of\ s \to s'\ when\ taking\ action\ a}{Total\ times\ of\ taking\ action\ a\ at\ state\ s} \quad (3)$$

C. Running IRL algorithm

The Inverse Reinforcement Learning Toolkit (IRL Toolkit) [13] is a package developed in Matlab which contains a collection of inverse reinforcement learning algorithms and provides a framework for evaluating these algorithms on the environment of MDPs. The package was originally developed as a reference implementation for the Gaussian Process Inverse Reinforcement Learning algorithm described in [13], but it also provides a general framework for the researchers who are interested in IRL. In this project, we adopted the algorithms described in [5].

Once modelled in the MDP environment, the expert trajectories and the probability transitions are imported to Matlab for setting up the algorithm running environment. When the IRL calculations complete, we get an optimal policy obtained through the AL/IRL algorithm from the expert's performance. The optimal policy is then imported and applied into the testbed game for observation.

IV. Experimental Design and Results

In this section we present experimental results, focusing on convergence results i.e., on analysing if the optimal policy devised by AL/IRL converges to the expert policy provided by demonstrations. Secondly we explain the design and construction of a self-reporting experiment and discusses the results from various perspectives.

A. Apprenticeship learning Via IRL Analysis

In Section II-C, we described the apprenticeship learning algorithm as presented in Abbeel's paper [5]. Here we firstly introduce how we captured the experiment trajectory data from the users' demonstrations, and display and analyse the AL convergence results obtained from the experiment.

This leads to our first contribution, designing an appropriate knowledge representation in Super Mario game testbed which allows a controller to learn via imitation learning.

In section III-A, we proposed the approach to represent the MDP environment, using a 5 × 5 grid whose size is much smaller than that of the best controller (VLS) used in the 2012 Turing Test Mario Championship, which is 22 × 22 [3]. Using a smaller size grid still allows us to capture key information from the environment without encountering the same computational trade-offs and limitations described by the authors of the VLS controller.

To search information in this smaller size grid precisely and quickly, we discretised the possible directions in the grid surrounding Mario (as described in Section III-A on MDP representation). This is a typical approach in several pattern recognition problems (e.g., gesture recognition) employing template matching [14]. By assuming that the reward function is expressed as a linear function of the known features we were able to obtain convergence of the the learned policy to the feature expectations of the expert after a finite number of iterations.

1) Experimental Setup and Data Logging: As mentioned in Section II, the testbed game is the modified version of Markus Persson's *Infinite Mario Bros*. The experiments were run on an Intel Core I7- 3610QM, 2.3GHZ, with a 4 GB RAM. The computer captured one line of log data every 40 milliseconds.

We invited 3 players for the tests since all of them have extensive experience in playing several kinds of video games. Each player was asked to complete a specific level for six times. After that, we chose the one who had highest average skills score (such as the number of coins collected, enemies killed and stomped in the level), which is returned by the Mario simulator at the end of the level, as our expert. For the purpose of covering the states as much as possible, we asked the expert to play three more levels and each level was successfully played six times. We ended up with 3413 states and 10 actions. The maximum size of a state transition set was 25.

2) Convergence Results : In this experiment, we assumed that the expert is trying to maximize a reward function. For finding a policy π that can induce feature expectation $\mu(\pi)$ close to the expert feature expectation $\mu(\pi_E)$, we applied apprenticeship learning to our expert's trajectories. The IRL

algorithm was stopped after 10 iterations as it reached a satisfactory minimum distance to the expert policy. We have plotted the margin (distance to expert's policy) in Figure 3. The distance is calculated using the Euclidean distance to the expert's feature expectations. Our results return a optimal policy π^* which almost exactly matches the expert's policy in terms of feature expectations.

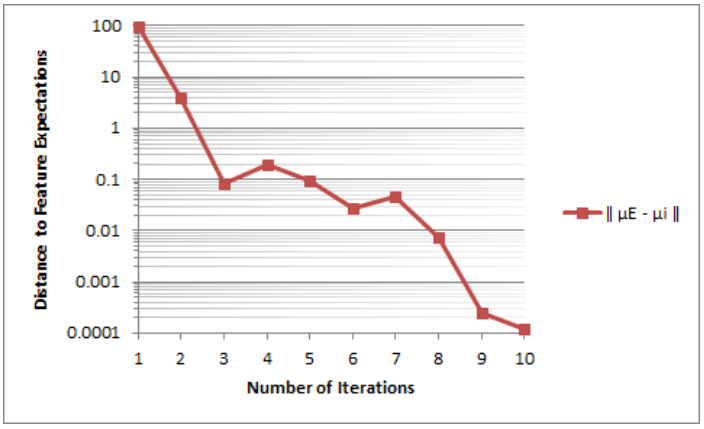


Fig. 3. Convergence speed of the IRL algorithm in Super Mario.

From the results displayed in Figure 3, we can see that apprenticeship learning is a fast learning method to study our expert's performance. Another important property of apprenticeship learning is that it is able to mimic the expert's style, which means each instance of the controller's policy closely matches the expert's policy. We list some features in Table I from the learned policy for analysis of the expert's style.

On examination of Table I, the data indicates a strong preference for moving right, which is necessary to complete the game. The expert places a higher priority on killing monsters by stomping on them, and places the highest priority on dealing with Koopa shells in his immediate vicinity (Mario might be facing a shell, stomping on a shell, kicking a shell or jumping over a shell). These results indicate that apprenticeship learning should be able to mimic an expert, however to prove this we have performed a user study to confirm our findings.

B. Questionnaire Experimental Study

We administered an online questionnaire to test the hypothesis that the controller learned via AL/IRL was "human-like", to the point where it could pass, at least in some instances, a modified version of a Turing Test (the Mario AI Turing Test [4]). The design of the questionnaire is available online[4] or on request. A summary of the results we gained from it is analysed and discussed.

1) Evaluating the controller's performance: To evaluate the controller's performance, the benchmark test in assessing the believability of a controller's performance versus that of a human player is conducted via a Turing Test, as done in the Mario AI Championship. In addition, we were concerned that the participants might have biased judgements if they knew that one of the video was generated by a computer, thus we ran two kind of tests, biased and unbiased. The results are presented in Section IV-C

2) Evaluating the controller's behaviour as human-like: Judging a controller's believability can usually be done from a first-person (playing with) or third-person (observing) perspective. In this project, it is impossible to run a first-person assessment as the game only has one character. It is reasonable and feasible to judge the performance from the third-person perspective. We decided to use the results from the 2012 Mario AI Championship as our baseline, since they had already been published when our work was carried out.

3) Design: To ensure the evaluation was correct, we set up the questionnaires in the following steps which can help us organize the clear comparison between human and computer performance, as in the modified Turing Test for dialogue provided in [15]:

- **Items**
  The items used in the questionnaires were five video clips of Mario game playing: three of them are the expert's demonstration and the other two are learned controller's performance.

- **Groups**
  We developed two types of questionnaires, investigating both within-subject and between-subject results. Each questionnaire consisted of two groups, which observed different combinations of gameplay video. The purpose was to assess how our controller performed in simulating human play. In addition, we would be able to see if having knowledge of the computer's participation would have biased these performance judgements.

  The first questionnaire required the participants detailed analysis of the controller's performance (referred to as QDPJ). Questions involved asking the user to agree/disagree with statements such as "Mario is collecting items proficiently" and "Mario appears to be exploring the world around him." The second questionnaire asked about Mario's overall performance judgements (denoted as QOPJ). This questionnaire informed the subjects of the AI computer participation, and it involved asking the user to agree/disagree with statements such as "Mario's performance is human-like" and "Mario's performance is akin to one of an expert".
  The subjects were divided into in four groups: Group A1 and Group A2 denote the subjects who took QDPJ first and QOPJ second, while, Group B1 and Group B2 denote those who took the questionnaires in the opposite order. Subjects in Group A1 and Group A2 are unbiased as they are not informed of AI participation until QDPJ, and those in Group B1 and Group B2 are biased.

- **Subjects**
  We were able to invite 70 participants. The subjects were divided into four groups: 17 of them were in Group A1, 31 of them in Group A2, 10 of them in Group B1 and 12 of them in Group B2. Having

[4]https://dl.dropboxusercontent.com/u/48945922/2014-04-01_Mario_Paper_Survey/MarioQuestionnaire.pdf

	Right Movement	Jumping	Collecting Coins	Stomping Killing	Hitting Brick	Close Shell
$\hat{\mu}E$	6.420	1.739	0.007	9.346	0.124	9.842
$\mu(\tilde{\pi})$	6.071	1.751	0.019	9.420	0.107	9.865
$\tilde{w}$	-0.076	-0.607	-0.019	0.489	-0.045	.0134

TABLE I. SOME OF THE FEATURES LEARNED VIA IRL. DISPLAYED ARE THE FEATURE EXPECTATION FROM THE EXPERT POLICY $\hat{\mu}E$, THE FEATURE EXPECTATION OF THE LEARNT POLICY $\mu(\tilde{\pi})$, AND THE LEARNT FEATURE WEIGHT $\tilde{w}$. HIGHER EXPECTATIONS INDICATE THE PREFERENCE TO ACTIVATE A GIVEN FEATURE IF AVAILABLE. FEATURE WEIGHTS ARE RELATIVE, THE HIGHER THE VALUE THE HIGHER THE PRIORITY OF THE FEATURE OVER OTHERS.

experience of playing video games was an important prerequisite.

C. Result and analysis

In this section, the figures provided represent an overall result of the experiments. Individual testing groups (A1, A2, B1, B2) are discussed to highlight minor discrepancies between survey results.

We were interested in testing how the subjects' judgement would be affected if they have been told there was a computer playing the game. We have used the χ^2 test for independence to see if the two factors response and questionnaire pattern (Group A: unbiased and Group B: biased) are independent. The degree of freedom for the test is two, since we compare two patterns of three cells (frequency of Agree, Neutral and Disagree). The result is displayed in Table II.

	Agree	Neutral	Disagree	Total
Group A	12	6	30	48
	12.34	6.86	28.80	
	0.010	0.107	0.050	
Group B	6	4	12	22
	5.66	3.14	13.20	
	0.021	0.234	0.109	
Total	18	10	42	70

TABLE II. SIGNIFICANCE TEST RESULTS: CHI-SQ = 0.530, DF = 2, P-VALUE = 0.767

The p-value is 0.767, which is higher than the usual values used in hypothesis testing (typically 0.01 or 0.05). Therefore, we can conclude that knowing a computer was used to generate some of the clips did not provide a bias in the users' judging the game play video clips.

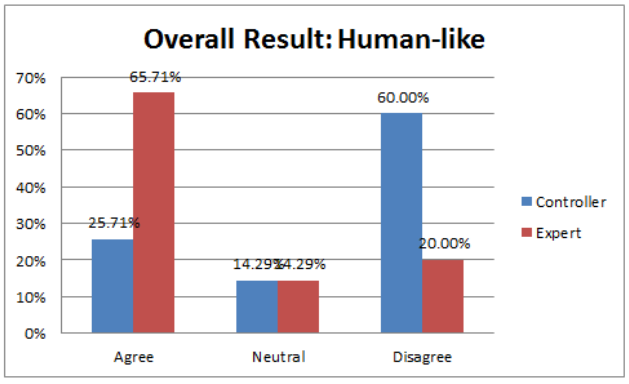


Fig. 4. Survey results: Comparison of participants' perception of human-like behaviour of the AI Controller and the human Expert.

Overall, 70 participants took part in the evaluation process. The scores are calculated as the percentage of judges' opinions:

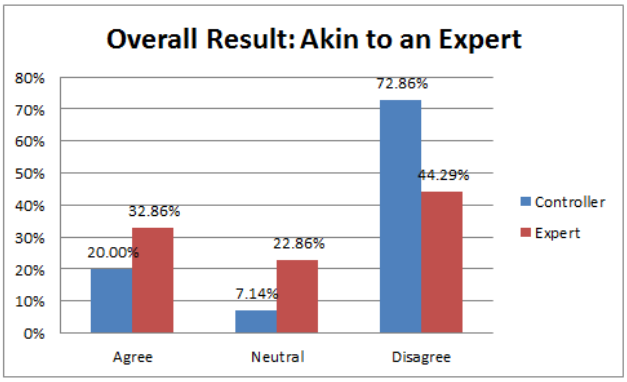


Fig. 5. Survey results: Comparison of participants' perception of Expert Gameplay of the AI Controller and the human Expert.

Agree, Neutral or Disagree. As per Figure 4, 25.71% of subjects agreed that the controller's performance was human-like. This result is only 0.08% lower than the best agent VLS in the Turing Test Track 2012 Mario Championship. We have noticed that there is an interesting phenomenon where 20% of the survey participants disagreed that the expert's play clips were 'humanlike'. Both the controller and the expert received 14.29% Neutral responses respectively, so we can see that the controller's performance did appear even more human, or at least not obviously non-human.

As shown in Figure 5, the controller also picked up on some of the player's expertise. According to survey results the player only received a 32.86% 'expert' result, and that the IRL controller performed at 20%. This means that it is performing just as well for over half of those respondents.

Overall performance human-like: IRL Controller

	A1	A2	B1	B2
Agree	29.41%	22.58%	40.00%	16.67%
Neutral	17.65%	9.68%	10.00%	25.00%
Disagree	52.94%	67.74%	50.00%	58.33%

Overall performance human-like: Human Expert

	A1	A2	B1	B2
Agree	76.47%	58.06%	80.00%	58.33%
Neutral	11.76%	16.13%	10.00%	16.67%
Disagree	11.76%	25.81%	10.00%	25.00%

Akin to an Expert - IRL Controller

	A1	A2	B1	B2
Agree	41.18%	9.68%	20.00%	16.67%
Neutral	5.88%	19.35%	10.00%	16.67%
Disagree	52.94%	58.71%	10.00%	66.67%

Akin to an Expert - Human Expert

	A1	A2	B1	B2
Agree	5.88%	41.94%	50.00%	33.33%
Neutral	17.65%	19.35%	40.00%	25.00%
Disagree	76.47%	58.71%	10.00%	41.67%

TABLE III. OVERALL GROUP RESULTS.

The significance results for each group are listed in Table III. Of note, in the individual groups there were a number of observable discrepancies and highlights. In Group A1, 29.41% of the subjects agreed that the controller's performance was human-like and 41.18% thought the controller was akin to an expert. For the expert's performance result, while 76.47% agreed the performance was human-like, only 5.88% thought the performance was akin to an expert, which indicates that for this group that our controller performed more skilfully than the expert.

Group B1 results indicate that 40% of the subjects thought the controller's performance was human-like. Compared to the result of Mario AI Championship 2012, it is quite a high score, although with the small sample size of the group the individual result is not significant. Group B2 results diverged in that only 16.67% of the participants thought the IRL controller's behaviour was human-like, which is much lower than the overall result. However the results for the expert are also lower, only 58.33% of them agreed that the performance was human-like.

For the evaluation of the controller's performance, Group A1 and Group B1 have much higher score than Group A2 and Group B2. Some possible causes for this are:

1) The video clip duration for Group A2 and B2 may not long enough to let the judges have a good observation about Mario's performance;
2) The video clip of the controller for Group A2 and B2 might not provide enough information of Mario's skills. We checked the clip and found that the level environment in Group A2 and B2 was quite empty compared to the other clips so that it was not able to give the judges sufficient details about Mario's skills;
3) The optimal policy obtained from AL/IRL constrains Mario to taking an 'optimum' unique action in a particular state, so this may limit the diversity of transitions in the real gameplay.

From the results above, we can prove that our controller's performance has been able to pass this modified Turing test at times. We can say apprenticeship learning via IRL has been feasible and effective for this project and it has potential for learning and imitating human-like game play in Super Mario. However, we have also noticed that the score of the controller's exploring ability is low in each group (see Figure 6 and Figure 7)

As mentioned in Section III-A, we have generalized 20 features of Mario in the gameplay for the apprenticeship learning method. We have extracted 4 features to investigate the drawbacks in our current approach. As seen in Figure 7, all feature values are uniformly normalized in the range [0, 1]. The figure illustrates that the controller has better killing skill and stomping skill than the expert, but there is a remarkable difference existing in devouring flowers skill, the controller has a null score. The controller also has lower score in the collecting coins behaviour. Possible reasons for this may include:

1) AL/IRL cannot generate a consistent enough policy for an event rarely demonstrated by the expert;

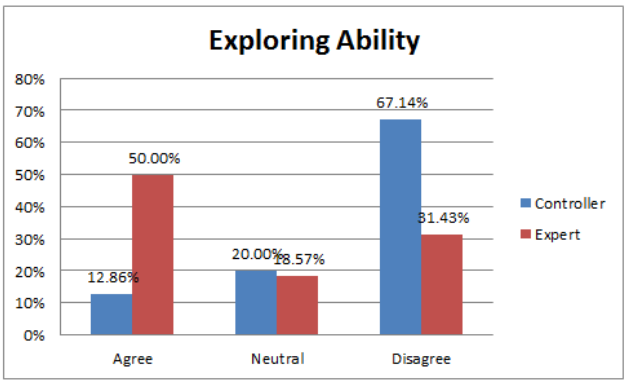


Fig. 6. Comparison between the perceived exploration ability of the controller and the expert.

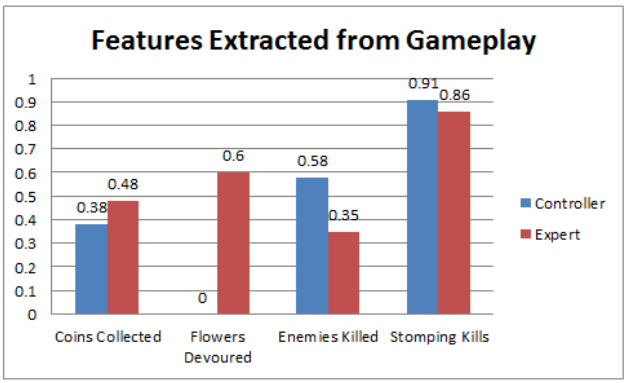


Fig. 7. Features extracted from gameplay, weights normalised in [0,1]. A comparison between the behaviours of the controller and the expert.

2) The collected states in the state set are not sufficient, which could be improved upon by letting the expert play more levels for a longer period of time;
3) The MDP environment was represented via a 5×5 gird, which limits Mario's vision. There was a trade off between grid size and scalability, compared to the in-game visible area the size of the grid is fairly small, and flowers and coins often sit outside of the range. Thus there will always be some items available in a human player's eye view that the agent is not able to detect;
4) Finally, the optimal policy we obtain from AL/IRL is deterministic, which means that the single optimal action is taken in a particular state every time. This limits the variety of taken actions in this particular state and it could be improved upon by using some of the candidate non-deterministic policies generated by AL/IRL, when iterating in search of the optimal policy.

V. Conclusion

In this paper, we proposed that apprenticeship learning as an imitation learning method to develop a human-like controller for the game Super Mario. We have provided a

solution to represent the knowledge in Super Mario and have demonstrated that apprenticeship learning converges rapidly (see Section IV-A2) to the expert policy. We have established the advantages of AL/IRL and have also highlighted limitations inherent in our current knowledge representation and its use by AL/IRL. We also found limitations in the current implementation that affected Mario's ability to explore the game environment.

We have been able to apply the obtained optimal policy in a Super Mario controller and have been able to assess its human-likeness via users' self-reporting. Our preliminary results perform at the same level to state-of-the-art techniques used in the latest Super Mario AI competition results published to date (see Section IV-B). Finally, we proved that the subjects' responses in the self-reporting experiment were not affected by knowing that a computer had generated some of the video clips presented in the experiment (again, see Section IV-B).

There are two areas of particular interest for future work. Firstly, increasing Mario's perception by using larger or shaped detection grids for determining the MDP states. To some degree, this can be mitigated but cannot be completely solved by enlarging the size of the grid. Growing the grid size has implications on state size, its storage and the convergence of the IRL algorithms and it may affect the algorithm efficiency. The trade-off between these factors could result a fairly complex problem, which will be very interesting to tackle in our future work.

Secondly, when a human is playing the game, they may at times take different actions in a repeated state. The obtained optimal policy returns one action at a certain state through calculating the weights of the expert known features. This limits the controller imitating the versatility of human behaviour as it lacks the variety and flexibility a human player expresses. For instance, if a player runs into a wall ten times and finally jumps over it once, the controller may learn that running into the wall was the player's preferred method of acting in that state and only perform that action. For solving this problem, we may borrow some idea from Bayesian models of imitation learning which formulate the probability of each action based on the previous action at a current state [16] [17] [18]. This obviously represents another avenue for future work, which we look forward to developing in the near future.

ACKNOWLEDGMENT

Geoffrey Lee, Min Luo, Fabio Zambetta and Xiaodong Li are with the School of Computer Science and Information Technology, RMIT University Melbourne, Australia. Some of the authors of this research were supported under Australian Research Council's Discovery Projects funding scheme (project number LP130100743)

REFERENCES

[1] C. G. Atkeson and S. Schaal, "Learning tasks from a single demonstration," in *ICRA*. IEEE, 1997, pp. 1706–1712.

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

[3] N. Shaker, J. Togelius, G. N. Yannakakis, L. Poovanna, V. S. Ethiraj, S. J. Johansson, R. G. Reynolds, L. K. Heether, T. Schumann, and M. Gallagher, "The turing test track of the 2012mario ai championship: Entries and evaluation," 2012.

[4] J. Ortega, N. Shaker, J. Togelius, and G. N. Yannakakis, "Imitating human playing styles in super mario bros," 2012.

[5] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, ser. ACM International Conference Proceeding Series, C. E. Brodley, Ed., vol. 69. ACM, 2004.

[6] M. Y. Lim, J. Dias, R. Aylett, and A. Paiva, "Creating adaptive affective autonomous npcs," *Autonomous Agents and Multi-Agent Systems*, vol. 24, no. 2, pp. 287–311, 2012.

[7] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *Advances in Computer Entertainment Technology*, N. Lee, Ed. ACM, 2005, pp. 429–433.

[8] G. Xiao, F. Southey, R. C. Holte, and D. F. Wilkinson, "Software testing by active learning for commercial games," in *AAAI*, M. M. Veloso and S. Kambhampati, Eds. AAAI Press / The MIT Press, 2005, pp. 898–903.

[9] B. E. John and A. H. Vera, "A goms analysis of a graphic machine-paced, highly interactive task," in *CHI*, P. Bauersfeld, J. Bennett, and G. Lynch, Eds. ACM, 1992, pp. 251–258.

[10] S. Karakovskiy and J. Togelius, "The mario ai benchmark and competitions," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 4, no. 1, pp. 55–67, March 2012.

[11] S. Mohan and J. E. Laird, "An object-oriented approach to reinforcement learning in an action game," in *AIIDE*, V. Bulitko and M. O. Riedl, Eds. The AAAI Press, 2011.

[12] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. A Bradford book. Bradford Book, 1998. [Online]. Available: http://books.google.com.au/books?id=CAFR6IBF4xYC

[13] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *NIPS*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, Eds., 2011, pp. 19–27.

[14] S. Korman, D. Reichman, G. Tsur, and S. Avidan, "Fast-match: Fast affine template matching," in *CVPR*. IEEE, 2013, pp. 2331–2338.

[15] A. P. Saygin and I. Cicekli, "Pragmatics in human-computer conversations," *Journal of Pragmatics 34(2002) 227-258*, 2002.

[16] R. P. N. Rao, A. P. Shon, and A. N. Meltzoff, "A bayesian model of imitation in infants and robots," *Cambridge University Press*, 2004.

[17] S. Ross and J. Pineau, "Model-based bayesian reinforcement learning in large structured domains," *CoRR*, vol. abs/1206.3281, 2012.

[18] D. Ramachandran and E. Amir, "Bayesian Inverse Reinforcement Learning," in *International Joint Conference on Artificial Intelligence*, 2007, pp. 2586–2591.

[19] R. S. Sutton, D. Precup, and S. P. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, no. 1-2, pp. 181–211, 1999.

[20] S. Levine, Z. Popovic, and V. Koltun, "Feature construction for inverse reinforcement learning," in *NIPS*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 1342–1350.

[21] G. N. Yannakakis and M. Maragoudakis, "Player modeling impact on player's entertainment in computer games," in *User Modeling*, ser. Lecture Notes in Computer Science, L. Ardissono, P. Brna, and A. Mitrovic, Eds., vol. 3538. Springer, 2005, pp. 74–78.

[22] S. Wintermute, J. Z. Xu, and J. E. Laird, "Sorts: A human-level approach to real-time strategy ai," in *AIIDE*, J. Schaeffer and M. Mateas, Eds. The AAAI Press, 2007, pp. 55–60.

[23] J. Laird, "Using a computer game to develop advanced ai," *Computer*, vol. 34, no. 7, pp. 70–75, 2001.

[24] S. Mohan and J. E. Laird, "Relational reinforcement learning in infinite mario," in *AAAI*, M. Fox and D. Poole, Eds. AAAI Press, 2010.

[25] J. E. Laird and M. van Lent, "Human-level ai's killer application: Interactive computer games," *AI Magazine*, vol. 22, no. 2, pp. 15–26, 2001.

[26] B. Gorman and M. Humphrys, "Towards integrated imitation of strategic planning and motion modeling in interactive computer games," *Computers in Entertainment*, vol. 4, no. 4, 2006.

[27] G. w. Lee, "Learning Reward Functions Underlying Player Behaviour in Computer Role Playing Games," Honour's thesis, RMIT University, Melbourne, Nov. 2012.