Predicting skill from gameplay input to a first-person shooter

DOI:
10.1109/CIG.2013.6633655

Predicting Skill from Gameplay Input to a First-Person Shooter

David Buckley, Ke Chen and Joshua Knowles
School of Computer Science
University of Manchester, UK
david.buckley@cs.man.ac.uk; ke.chen@manchester.ac.uk; j.knowles@manchester.ac.uk

***Abstract*—One way to make video games more attractive to a wider audience is to make them adaptive to players. The preferences and skills of players can be determined in a variety of ways, but should be done as unobtrusively as possible to keep the player immersed. This paper explores how gameplay input recorded in a first-person shooter can predict a player's ability. As these features were able to model a player's skill with 76% accuracy, without the use of game-specific features, we believe their use would be transferable across similar games within the genre.**

I. Introduction

One of the challenges of video game design is catering for an audience with different styles of play [1]. The unpredictability of player preferences and increased costs of game production have discouraged larger publishers from venturing into new territory [2]. In response to this, the emerging field of player modelling seeks to learn player preferences through the use of existing techniques such as neural networks or genetic algorithms [3].

Models of player preferences provide the developer with valuable feedback, allowing them to adjust their game manually, or dynamically adapt it after release. For instance, particularly challenging sections in a game can be identified [4], or the game could be altered in real-time to optimise a particular emotion [5]. Further examples of research in the field of adaptivity can be found in the survey by Lopes and Bidarra [6].

One common form of adaptivity found in video games is *Dynamic Difficulty Adjustment* (DDA) [7], in which the game accommodates for the differences in players' abilities and skills by adjusting the difficulty settings. Successful examples of this can be found both in academia [8], [9] and the industry [10]. However, before adaptation, the game must have some method to work out how competent the player is and how well they are currently doing. This concept is known as a *challenge function* [7].

In order to work out a player's current state, some video game companies have taken to instrumenting their games to construct player models [11]. Game events, such as player deaths, are recorded unobtrusively in *log files* that can be mined for features. The events that make up this log file are described in the field of Human Computer Interaction as *high-level* or *low-level* [12]. High-level events such as player deaths are composed of lower-level events; damage taken and health restored. In a first-person shooter, player movement around their environment is composed of more frequent key and mouse movement events.

While high-level game events like player jumps or deaths have been explored previously [13], a player's direct input has received less attention. Previous work has successfully clustered players by their playing style [14]. For a 2-D arcade game *Snakeotron*, the features corresponding to player input were considered very important. In the same study, player behaviour in a second game, *Rogue Trooper*, clustered around movement and firing which are both composed of low-level input events.

Therefore, for the purposes of modelling a challenge function, this research proposes that features derived from low-level events, the player input, can be used to construct a model of the player's skill. First-person shooter (FPS) games require continuous interaction with the player and the real-time nature of this input can provide a snapshot of the player. Moreover, the fast-paced nature of their gameplay and input from both mouse and keyboard offers a good use case for exploring hardware input.

In single-player games, this model could be used for the purposes of DDA, determining how well players are doing at a particular instant [8]. Its use would also apply to other dynamic contexts, including tailoring tutorials towards less experienced players. Within multi-player games, it can be extended to matchmaking, where players are partnered with opponents of a similar skill. By quickly determining the level of experience a player has, they can be matched against a comparable player without having to play many games.

In order to test the theory that skill can be modelled from player input data, 34 participants with varying degrees of experience played an FPS. A selection of low-level events were recorded during play, and, after each game, players were asked to answer questionnaires about their experience. The choice of data and questions has been described in Section II.

Random forests were used to construct models from this data. A previous study on player modelling [11] explored a variety of techniques suitable for this task, including multilayer perceptrons, common in this domain [13], and support vector machines. The choice of model and reasoning behind it has been outlined in further detail in Section III.

Results, presented in Section IV, show that players can be classified by skill with 76% accuracy after only a minute of play.

II. Data Collection

The experimental method is described in this section, including the reasoning behind the data collected.

A. Red Eclipse

The gameplay within the FPS genre is typically fast and relies on quick reflexes and skilful use of input devices. For players on the computer, these are the mouse, used for aiming, and the keyboard, typically used for movement. The richness of the data provided by two input devices and their use in-game offer a good example for exploring player input. Moreover, the use of multiplayer games within this experiment removes dependencies on storylines and any effects they may have on the player.

Red Eclipse[1] is a free and open-source, multiplayer FPS built on the *Cube Engine 2*[2]. It draws on similar themes as the popular *Quake*, and is largely representative of the FPS genre, offering several different modes and considerable customisability. Its open-source licence also allows us to instrument the game for the purpose of this experiment.

Modification of *Red Eclipse* was performed on the client, potentially allowing players to download an instrumented version of the game and connect to existing servers. Events, such as key presses and releases, were logged to a text file when registered with the engine. Logs were output to a text file, extending on the game engine's existing logging system.

Some of the game mechanics and rules of *Red Eclipse* are unique or uncommon among other FPS games. One such game mechanic is an advanced movement system, with which players can perform double-jumps and run along walls. While uncommon, this did not subtract from the experience, and was only used by a few of the more experienced players. The rules also included a weapon limit so that the player could only carry up to three weapons at a time. This weapon limit, alongside unlimited ammo, decreased the complexity of the game for newer players.

In addition, these game mechanics and rules can be customised by the user, allowing us to configure the game for the experiment. The settings specified included the game mode, time limit, difficulty and game level, known as a *map*.

B. Experimental Setup

Data collection ran over a two week period, and was done in-house in order to ensure the quality of the data collected. During this time, 212 games were played by 34 participants who completed an average of 6 games each, with one player completing as many as 22 games.

Before taking part in the experiment, every participant was asked to sign a consent form, read a tutorial and complete a short, demographic questionnaire. The tutorial was provided to the player to ensure everyone started the game with a minimum level of knowledge. It explained the basic controls used in FPS games, the weapons found in *Red Eclipse*, and some of the key game mechanics such as health regeneration. The player was given as long as they needed to read through this before starting the experiment. They were then allowed to play as many games as they wished, filling in a questionnaire about their experience after each.

Two of the game's parameters - the length of game and the game mode - were fixed throughout the experiment. The first of these, the game time, was set to three minutes. Choosing a shorter game time made it easier for players to remember their experience [15] and for the questions asked to more accurately reflect the state of the whole game. The second parameter was set to 'deathmatch', a standard game mode in which players fight individually to earn points through killing each other. This game mode is one of the most common in FPS games and its simplicity controlled the learning required by players.

The enemy difficulty and map were then selected randomly between each game. Six different difficulties and eight maps were available to choose from. These allowed the player to experience a variety of games while retaining the simplicity of the experiment. The maps chosen, for instance, were limited so the player might have the opportunity to become familiar with some levels, while still including a range of environments, including simple terrains, complex building structures or wide, spacious arenas.

C. Player Feedback

For the purpose of player modelling, a measure of the player's emotion was taken through the use of questionnaires. While objective and quantifiable measurements of a player's emotions is preferred for prediction, use of physiological data is still an emerging field [16] and equipment is expensive. Therefore, self-assessment is a typical form of capture for player experience [3], [17], as in this study.

A separate questionnaire was used to collect simple demographic information about participants. From this, the majority of people who took part were male, with three female participants who played 12 games out of the total 212. There was, however, an even split of players 18 to 25 years old and those 26 and older. Participants were also asked about their playing habits in this questionnaire, in particular the hours spent a week playing games and the number of FPS games played. The first category, the hours, was split up into four categories to determine how often players spent playing games on a regular basis. Participants were also asked how many first-person shooters they had played previously to use as a quantifiable measure of player experience. While they were asked to select one of five categories as best as they could, no participants selected the option 'None'. The other categories were:

- 1 or 2
- 2 - 5
- 5 - 10
- More than 10

For each of the games played, the player was asked four questions corresponding to *fun*, *frustration*, *challenge* and *map*

[1]http://www.redeclipse.net

[2]http://cubeengine.com

complexity, each on a five-point Likert scale. The first three of these were chosen due to previous success in predicting them [13]. The last was selected for use as preliminary research into map design, exploring the connections between experience and map layout. *Predictability*, while showing a measure of success in the previous research, did not apply to the non-linear nature of the maps.

The work on which this was based [13] used a 4-alternative forced choice (4-AFC) approach in obtaining feedback. Using this method, players offer a comparison between two games, stating which elicited that emotion the most, or whether both or neither elicited it the same amount. The differences between rating and preference based reporting has been explored briefly [18], finding that there are some inconsistencies due to 'order-of-play' effects found in rating questionnaires. However, despite these, a Likert scale provided twice as many examples for learning, and required only one game to be remembered per questionnaire.

D. Captured Data

During each game, all mouse and key events produced by the game were recorded in a log file. Alongside these data, particular game events, such as player deaths and damage dealt, were also logged. While not the focus of the research, these higher-level events were used for comparison and a better perspective of the games played. The resulting data files can be found on the author's website[3].

E. Features

Statistical features were extracted from each game due to the size of the log files, which could contain over 10,000 events. These features, the most interesting of which have been described here, are more accessible to machine learning algorithms. In addition, the game was split up into time windows to explore the properties of different sections of the game, and to find the smallest size section of gameplay that could be used for prediction [19]. The window sizes presented in this research were 5s, 10s, 60s and 180s, the full length of the game.

1) General Features: Each of the events was placed into a category such as *mouse*, *key* or *damage dealt*, and features were extracted from each group to describe the distribution of events. The most notable were:

- Number of events
- Measure of distance from centre of screen
- Mean time between events

2) Mouse Movement: By far the most frequent event recorded in the log file is that of mouse movement, taking up around three quarters of the total events. However, as this data is analogue, it is also the most unpredictable. Each event generated by mouse movement records the displacement of the mouse in the x and y directions, referred to here as δx and δy. These were compiled to work out the absolute positions of the mouse for each event, x and y. Using these four values, the following features were constructed in order to best describe the movement of the mouse over the time window:

- The average x and y position
- The maximum and minimum x and y positions
- The standard deviation of x and y
- The largest δx and δy value
- The average δx and δy value
- The sum of the absolute values of δx and δy

3) Button Presses: After the mouse, the second form of player input into the game is from button presses, both presses and releases. For the purpose of this study, this includes both key and mouse button data, as the two are only distinguishable by an ID to the game engine. These events allow the player to perform actions in *Red Eclipse* such as moving or firing their weapon. Logging this data can therefore provide a low-level view of what the player is doing.

In addition to the number of key press events in the time window, we also extracted the following features:

- Key that was pressed the most
- Most keys pressed at one time
- Time spent holding forward key
- Time spent holding backward key
- Time spent holding left key
- Time spent holding right key.

III. Skill Modelling

A. Definition of Skill

The skill of a player is particularly important in multiplayer games as it, more often than not, determines the winner of a game. For players to enjoy a game, they should typically be matched against players of a similar skill level, thereby increasing competition. Measurements to calculate or compare skill levels can be used to accomplish this.

Some measures of skill are already widely used in games. *StarCraft*, the national e-sport of South Korea, makes use of *actions per minute* (apm) to judge a player, with professional players capable of over 300apm. In first-person shooters, *accuracy* and *kill-to-death ratio* are popular metrics. However, while common, these can depend on the game type or the opponents.

Hit accuracy, the number of times a player hits an enemy divided by the total number of shots, is a common way of measuring a player's skill. An experienced player is likely to be more apt at aiming the mouse and therefore hitting opponents. However, this measure is highly dependent on the type of weapons used. Within *Red Eclipse*, nine different weapons are available, all designed to provide unique mechanics. Players might have differing preferences, and therefore, while style might be predicted from hit accuracy, an accurate measure of skill may not. The second measure mentioned, kill-to-death ratio, has been explored briefly in this research.

In order to create a measurement of player skill, players were asked how many hours of games they currently played per week and how many first-person shooters they had previously played. Both of these seek to remove subjectivity,

[3] http://www.cs.man.ac.uk/~buckled8/automatic.html

the first used to determine how much time people invest into games, the second to measure their previous experience. However, it was found in this study that sometimes players with a lot of experience had not played recently. It was therefore considered useful to examine both metrics, *Hours* and *FPSs Played*.

B. Previous Modelling Techniques

A variety of machine learning techniques have already been used for the purpose of player modelling, one study examining the performance of several [11]. This study, also working towards predicting player behaviour, had some success in prediction, and showed that some techniques were more suited than others. Most notably, multilayer perceptrons (MLPs), decision trees and support vector machines (SVMs) performed well. The first of these, MLPs, are a popular choice for prediction in games [13], [20]. However, they are relatively slow to train, requiring genetic algorithms to optimise, and the resulting network is harder to interpret. SVMs are also difficult to optimise, requiring adjustment for each data set and, with regards to this previous research [11], its performance was notably worse than that of the other methods. The performance and flexibility of the third technique, decision trees, led us to explore random forests.

C. Random Forests

Random forests [21], an ensemble method constructed from decision trees, can provide state-of-the-art performance, and have most notably been used in Microsoft's Kinect [22]. They are fast and effective, performing well on high-dimensional data. In addition, decision trees have a white box property in that the constructed models can be understood. Contrasting with the black box property of an SVM, random forests therefore have a 'grey box' property. The features used in the random forest models can be ranked by their utility, providing an opportunity to analyse the features used.

In this task, where each game session was split into smaller time windows, the number of features grew rapidly. As such, random forests were suitable to our data set, able to cope with the size and even present the most interesting features. Moreover, random forests can model both classification and regression problems, allowing for flexibility during analysis.

In order to train a random forest, a data set and corresponding labels are provided to learn from. For our research, this data set consisted of all features taken from a particular window for a subset of games. For example, the features extracted from the first ten seconds of the game. These windows could also be combined and passed as training data, for instance the second and fourth ten second windows. The labels then corresponded to either information about each game, such as the points scored, or the player of the game, such as their age.

As mentioned, a random forest is an ensemble method, and, therefore, is made up of many decision trees, each of which sees a different view of the data. For classification, each decision tree votes for a particular class and that with the majority vote is taken. Regression can also be done by averaging the resulting class for each model. To achieve these differing views for each model, each decision tree is trained on a subset of data, randomly selected with replacement.

The model behind a random forest, the decision tree, is constructed by recursively splitting the training data into separate classes. At each node in the tree, the split is made on the feature that produces the most information gain. As such, the final nodes in an ideal tree will each only contain instances of a single class. A new example can then be classified by following the correct branches down the tree.

One other feature of the models used in a random forest is that not all features are used at once. During node creation, a subset of features are chosen to calculate the split. This furthers variation of features used and thereby the performance of models, increasing the overall performance.

Once constructed from player data, random forest models can be used for predicting information about the game or player. This could be categorical data like that supplied in the questionnaires, or the result of a game, such as the player's final score. The random forest can be constructed for either case; classification or regression.

IV. Results

In this section the results from data analysis and motivations for exploring each of the areas are presented.

A. Experimental Setting

The first step of analysis was to consider how well the most coarse data sets could model each of the labels provided by the players' feedback. This put each of the tasks in perspective. In order to accomplish this, classification models were constructed from the largest window sizes, 180s and 60s. Two labels were omitted from these tests due to large imbalance in the sample: previous experience of *Red Eclipse*, of which only two participants had any, and gender, as there were only 3 female participants that took part out of 34.

Each random forest in this experiment was trained using 500 decision trees and the number of features selected randomly for each tree was left as the default. This was calculated as the square root of the total number of features available. The models were then tested using 5-fold cross validation, each test repeated a further 5 times and averaged in order to ensure a meaningful result. In line with this, all accuracies shown are testing errors.

Following on from these results, two labels, *FPSs Played* and *Hours*, were found to be of particular interest. These are both representative of the players' experience and both showed some level of prediction. The two labels also have slightly different interpretations of skill. The first, *FPSs Played*, provides a representation of a players' entire gaming history and indicates experience specific to first-person shooters. *Hours*, on the other hand, is a snapshot of each players' most recent gaming experince, regardless of genre. A high *Hours* with low *FPSs Played*, for instance, may indicate preference towards a different genre.

TABLE I

TEST ACCURACY (%) OF RANDOM FOREST MODELS FOR DIFFERENT LABELS GIVEN FEATURES EXTRACTED FROM THE WHOLE GAME (180S) AND SECTIONS OF THE GAME (60S). LABELS ARE SPLIT INTO 'OBJECTIVE' AND 'SUBJECTIVE'. THE BASELINE IS PROVIDED AS A MINIMUM EXPECTED PERFORMANCE. '#' REPRESENTS THE NUMBER OF CLASSES FOR EACH LABEL. STANDARD DEVIATIONS PRESENTED IN BRACKETS.

Label	#	Base	180s	60s			
			All	1	2	3	All
Map	8	15.6	55.6 (3.28)	35.9 (3.28)	38.4 (2.50)	48.7 (3.85)	50.1 (3.24)
Difficulty	6	18.9	26.5 (3.03)	18.2 (2.53)	24.7 (2.48)	30.4 (3.13)	27.3 (4.06)
Hours	4	33.0	62.6 (2.98)	49.2 (2.14)	57.0 (2.81)	63.0 (2.96)	57.2 (3.55)
FPSs	4	42.5	49.2 (2.72)	54.4 (2.64)	44.9 (2.58)	54.8 (3.63)	54.1 (2.47)
Fun	5	46.2	39.4 (2.66)	38.4 (.639)	39.7 (1.67)	41.9 (1.28)	42.7 (.261)
Frustration	5	38.2	41.3 (3.17)	34.8 (2.62)	32.9 (2.45)	45.1 (2.43)	43.6 (2.79)
Challenge	5	32.1	31.7 (3.95)	40.6 (2.38)	28.3 (1.71)	46.1 (2.56)	36.0 (3.36)
Complexity	5	31.1	29.3 (3.13)	25.5 (2.78)	35.7 (2.23)	31.5 (2.86)	34.8 (3.61)

In order to explore these two labels, models were constructed using subsets of the available data. Features corresponding to player input, *player input features*, were examined first, demonstrating that skill could be predicted using only player input. Then concept drift was explored by training the models on only the first games for each player. The third step was to make use of the more fine data sets, training with smaller windows to determine how quickly a players' skill could be predicted.

Finally, we exploited the regression available in random forests by attempting to predict more common metrics of skill such as points scored or the kill-to-death ratio.

B. All Labels

From player responses to questionnaires, a subset of categorical labels were selected. These were then used to train random forests, for the larger window sizes 180s and 60s, listed in Table I. For the latter, which only covered a third of the game, the performance of different windows has been presented. The labels have also been separated into two categories, 'objective' and 'subjective' to differentiate between the information about the game and player and the player's reported preferences. The baseline and number of categories have also been shown for each of the labels. As in [11], the baseline is equivalent to guessing the majority class for each label to account for unbalanced data, and is used here as a minimum expected performance.

C. Classifying Skill

Once trained, a random forest is capable of reporting the importance of each feature used by the decision trees. This allowed us to rank the features used for predicting both *Hours* and *FPSs Played* and understand which had the most impact.

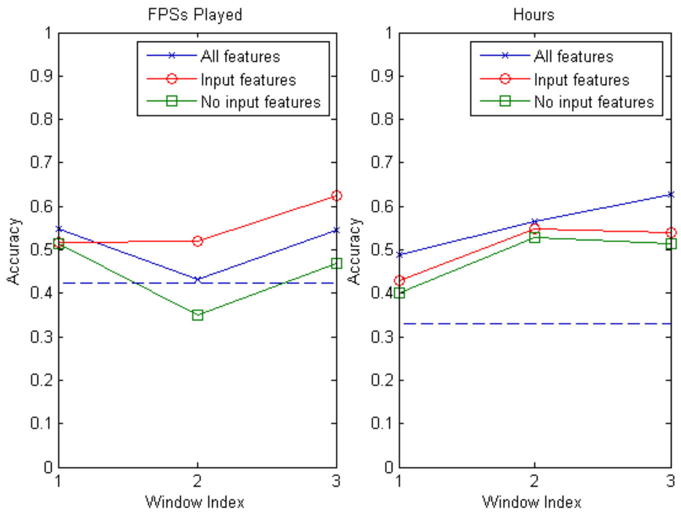

Fig. 1. Comparison of classifiers trained on different combinations of features for 60s window sizes. Baseline shown as a dashed line.

The top five features for *FPSs Played*, in order of importance, were:

1) Time spent holding backwards key.
2) Time spent holding forwards key.
3) Time spent holding left key.
4) Total mouse distance moved in the x direction.
5) Damage dealt over the game.

The top five features for *Hours*, were:

1) Number of kills.
2) Damage received over the game.
3) Number of kills by weapon two.
4) Points scored.
5) Total dominations of killers.

The use of player input features in the first label was of interest, and led us to train models with only features taken from the keyboard and mouse. The classification accuracies for these models are shown in Fig. 1, with comparisons to a model trained on all features and one trained without player input features. The highest accuracy for both labels was at 62% in the final window.

The next stage of analysis was to explore concept drift of skill, where players familiarise themselves with the game over time. Models were therefore trained with and without the first game for each player. Given the 34 participants, there were 34 games with which to train and test the smallest model. The results are presented in Fig. 2 with baselines. The change in baselines for each model is indicative of less skilled players playing fewer games.

The bias that might have been caused by the most active player, with 22 games, was considered. The first of these two tests, Fig. 1, was trained and tested on the same data set without this player. The results were very similar, but raised the baseline accuracy to 47.4%.

Once we had examined the effect of player input data and players' first games, we turned to classifying smaller windows

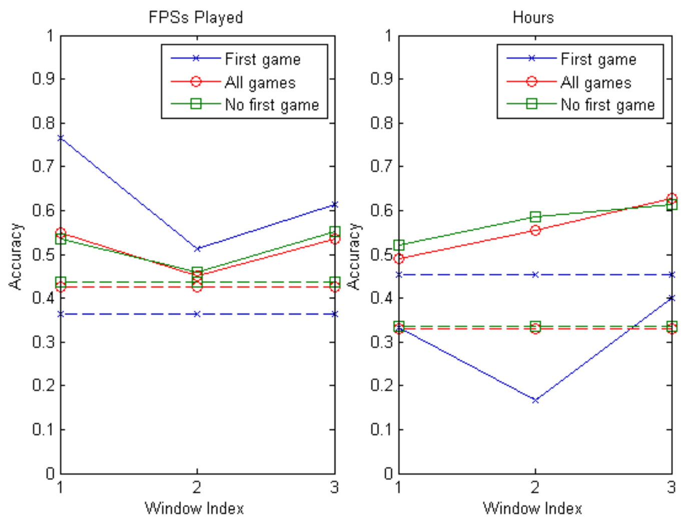

Fig. 2. Comparison of classifiers trained on different games for each player with all features for 60s window sizes. Baseline shown as a dashed line.

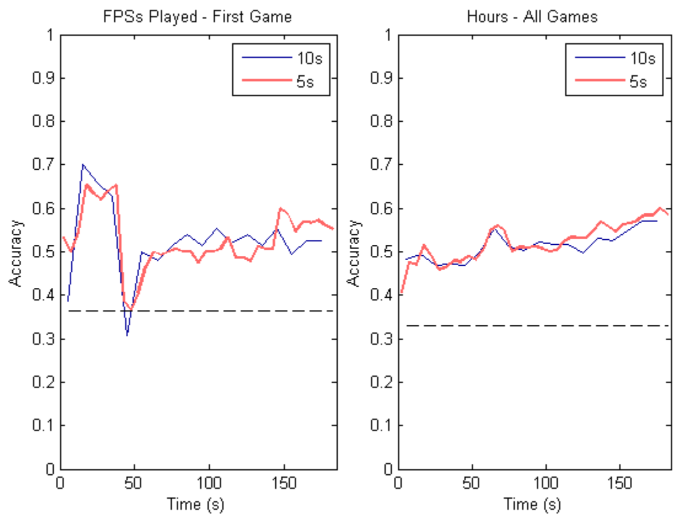

Fig. 3. Accuracy of classifiers trained by cumulating successive windows for sizes 5s and 10s using all features. Baseline shown as a dashed line.

with more fine-grained data. Fig 3 compares models using two different window sizes, at each window training the model cumulatively with features from the previous windows.

D. Regression

While classification could be done for simple measures of a player's previous experience, we then turned to more common measures of skill. As in [11], where completion time was estimated, we made use of regression to predict points scored and the kill-to-death ratio at the end of each game. For both of these factors, we predicted the value for the game and the mean for each player using only their input features collected from cumulative 10s windows.

The relative absolute error (RAE), used here, is defined by the sum of the differences between the prediction and actual value divided by the sum of the differences between the mean and the actual value. This allows us to compare performance

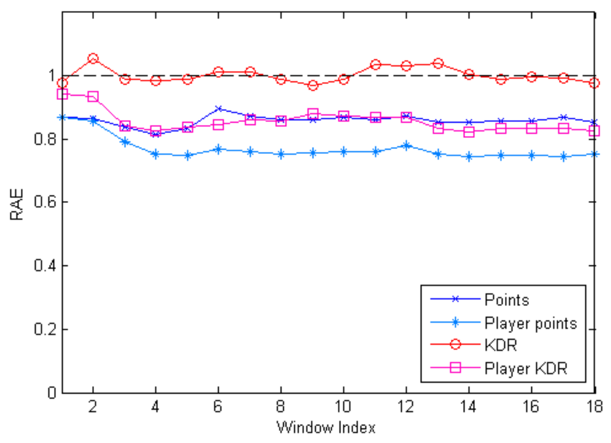

Fig. 4. Relative absolute error of models trained on player input data for different continuous metrics of player skill using different 10s windows. Baseline shown as a dashed line.

for labels with different measures. Fig. 4 shows that predicting the player's mean score outperforms the other categories, and predicting the kill-to-death ratio for each game did not return promising results. Random guessing, used for the baseline in these experiments, returns a value of 1 for RAE.

V. Discussion

Input to any application is likely to be very unpredictable, particularly in a fast-paced game such as *Red Eclipse*. The user may hit keys accidentally or idly trail the mouse. The performance seen in Fig. 2 and Fig. 3 is therefore promising, as, for certain cases, it is capable of predicting a player's skill better than guessing.

The claim that a player's input can be used to predict their skill has been reinforced in Fig. 1, in which the performance of a model trained only on *FPS Played* out-performs that trained with higher-level game features. *Hours*, however, performs better with both sets of data. As game statistics such as player points and kills can be found in this, it is understandable that these increase performance of the model.

For visualisation of player input, Fig. 5 shows the difference between a skilled player, red, and an unskilled player, blue. Each line is a crude representation of their movement with each of the four movement keys over time, the darkest part of the line indicating the end of the game. The more experienced player can be seen to use more complex keyboard input, while the newer player makes less use of the input in a simpler fashion, and does not hold the movement keys for as long.

Player skill was expected to increase between games, particularly from the first game. This should be especially true of their input, while players familiarise themselves with the game's controls. In line with this, *FPSs Played* can be seen to perform better for the first section of the first game, while *Hours*, more reliant on game data, performs incredibly poorly on the first game and strengthens over time, also seen in Fig. 3. From these observations, both labels could be employed in

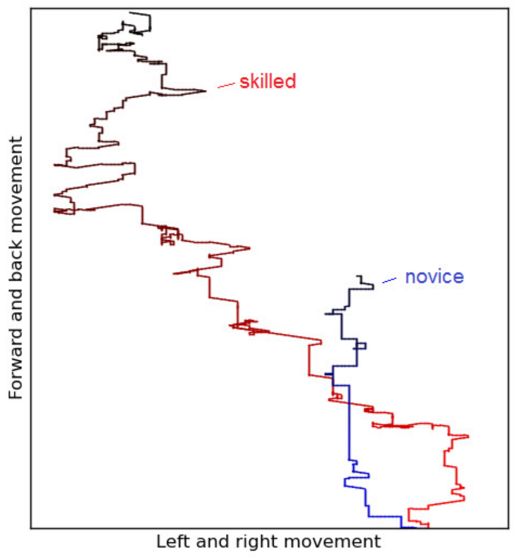

Fig. 5. Comparison of keyboard input between two players of different skill

a game; *Hours* taking over after a time when players are more likely to be comfortable with controls.

The imbalance in the data set and below average performance for some experiments indicate that more rigorous studies should be undertaken. These might explore a player's input over both the first few games and how it changes after several games and the player becomes more familiar with the controls and the game mechanics.

The windows in Fig. 3 are representative of different periods in the game. Accuracy for models trained on the first few windows of the game were as high as 70%, indicating that player skill can be determined within the first 30 seconds of a new player starting to play. This may be useful for automatically accommodating for different skills in both single-player and multiplayer FPS games.

Interestingly, models trained on the end of the game also see an increased performance. This effect is seen on all models, including those only trained on player input features. One possible explanation for this is that more experienced players, aware of the end of the game, changed their style of play to adjust to maximise their points.

While exploring more common measures of skill, presented in Fig. 4, we were unable to predict the points or kill-to-death ratios for each game with much accuracy. Models predicting the mean for each player were more successful, however, indicating either that the regression task was made easier through averaging, or supporting the claim that an individual player's skill can be predicted from their input data.

Some preliminary research went into predicting player emotion, presented in Table I. The results of this were very poor, performing no better than guessing. Emotions like frustration have been shown to be predictable for certain games [13], and correlations were found with some notable features like kills and deaths. There was a slight increase in performance for later windows, however, which complies with Kahneman's findings that the memory of an event is strongly affected by the latter parts of the experience [15]. Due to the noise in player input and the fluctuation of their emotions, this may be possible in the future with more fine-grained feedback from players.

Another predicted label of note is that of the current map. An accuracy of 56% is notably high for an eight-class problem, as seen in Table I. This, however, is explained by the nature of the data. The points scored at the end of each game is highly dependent on the current map, as evidenced by better prediction in later windows. Such a relationship makes it easy to predict the map played. Similarly, the difficulty of the map could be predicted in a similar, but less reliable, fashion.

VI. Conclusions and Future Work

Having recorded the in-game input from 34 players, we successfully predicted their previous experience with 76% accuracy for a four-class problem, as shown in Fig. 2. Moreover, similar performance can be achieved after only 20s of gameplay. This means players can be classified by skill when they first start playing a game, allowing us to either present a tutorial or automatically set the difficulty as appropriate.

Matchmaking by skill may also be possible without requiring players to play many games. Furthermore, the impact of the input-based features is that external applications such as a digital download service could be used to model its users more closely. These techniques may also be generalisable to other games in the genre. This would, however, require more game modes, such as team-based play, and games with a different pace, such as the more tactical *Counter-Strike*, to be explored.

One of the main drawbacks of this work is the fixed game time. Longer games, which are typical in the multiplayer scene, may cause discrepancies in training. If a model is only required for the first 30s of gameplay, however, then this method would be suitable.

Now that it has been shown that player input is a viable predictor of player skill, further analysis is required in order to uncover more patterns in play. As in other research [14], unsupervised techniques could be used to cluster players by their input. This could help prediction accuracy and aid in more rigorous extraction of features. Patterns found in combinations of key presses, for instance, is one avenue of research.

The measure of skill used for learning a skill model should also be explored. While a player-reported measure of skill was used, other, more objective criteria, may provide a more solid foundation for these models. In particular, these models should be compared to long-term rankings such as the Elo rating system [23].

One of the issues touched on here is that player feedback of emotions is too coarse, even more so for a longer game. The creation of new techniques for monitoring player emotions continuously during a game session may allow player input to be mapped to emotions.

Finally, these models were able to predict skill using input to a mouse and keyboard. Other genres, such as role-playing games, make very different use of these devices, while some, such as flight simulators or consoles, make use of completely different devices, such as joysticks and gamepads. It would, therefore, be interesting to explore the success of predicting skill on other devices.

Acknowledgements

The authors would like to thank all students and staff at the University of Manchester that participated in the experiment, particularly A. Apaolaza and N. Matentzoglu for the help they provided to this project, and the *Red Eclipse* community for their assistance during instrumentation. The authors would also like to thank the anonymous reviewers for their constructive and insightful comments.

This work was supported by the Engineering and Physical Research Council [EP/I028099/1].

References

[1] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit MUDs," *The Journal of Virtual Environments*, vol. 1, 1996.

[2] N. Brown. (2012, May) Free radical founder on leaving the FPS behind. [Online]. Available: http://www.edge-online.com/features/free-radicals-founder-leaving-game-industry-behind/

[3] G. N. Yannakakis, M. Maragoudakis, and J. Hallam, "Preference learning for cognitive modeling: A case study on entertainment preferences," *IEEE Trans. Syst., Man, Cybern.*, vol. 39, no. 6, pp. 1165–1175, 2009.

[4] J. H. Kim, D. V. Gunn, E. Schuh, B. Phillips, R. J. Pagulayan, and D. Wixon, "Tracking real-time user experience (TRUE): a comprehensive instrumentation solution for complex systems," in *Proc. SIGCHI Conf. Human Factors Comput. Syst. (CHI'08)*, Florence, Italy, 2008, pp. 443–452.

[5] G. N. Yannakakis and J. Hallam, "Real-time game adaptation for optimizing player satisfaction," *IEEE Trans. Comput. Intell. AI Games*, vol. 1, pp. 121–133, Jun. 2009.

[6] R. Lopes and R. Bidarra, "Adaptivity challenges in games and simulations: A survey," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, pp. 85–99, Jun. 2011.

[7] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin, "Polymorph: dynamic difficulty adjustment through level generation," in *Proc. Workshop Procedural Content Generation Game.*, Monterey, CA, 2010, pp. 11:1–11:4.

[8] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinorcement learning to provide dynamic game balancing," in *Proc. IJCAI Workshop Reasoning, Representation and Learning in Computer Games*, Jul. 2005, pp. 7–12.

[9] C. H. Tan, K. C. Tan, and A. Tay, "Dynamic game difficulty scaling using adaptive behavior-based AI," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, pp. 289–301, 2011.

[10] M. Booth, "The AI systems of left 4 dead," in *Keynote, Fifth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE'09)*, Stanford, CA, Oct. 2009.

[11] T. Mahlmann, A. Drachen, J. Togelius, A. Canossa, and G. N. Yannakakis, "Predicting player behaviour in Tomb Raider: Underworld," in *Proc. IEEE Symp. Comput. Intell. Games (CIG'10)*, 2010, pp. 178–185.

[12] D. M. Hilbert and D. F. Redmiles, "Extracting usability information from user interface events," *ACM Comput. Surv.*, vol. 32, no. 4, pp. 384–421, Dec. 2000.

[13] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience for content creation," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, pp. 54–67, Mar. 2010.

[14] J. Gow, R. Baumgarten, P. Cairns, S. Colton, and P. Miller, "Unsupervised modeling of player style with LDA," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 3, pp. 152–166, 2012.

[15] D. Kahneman, "Choices, values and frames," in *Experienced Utility and Objective Happiness: A Moment-Based Approach*, D. Kahneman and A. Tversky, Eds. New York, NY: Cambridge University Press, 2000, ch. 37, pp. 673–692.

[16] F. Levillain, J. Orero, M. Rifqi, and B. Bouchon-Meunier, "Characterizing player's experience from physiological signals using fuzzy decision trees," in *Proc. IEEE Symp. Comput. Intell. Games (CIG'10)*, 2010, pp. 75–82.

[17] G. van Lankveld, P. Spronck, J. van den Herik, and A. Arntz, "Games as personality profiling tools," in *Proc. IEEE Conf. Comput. Intell. Games (CIG'11)*, Sep. 2011, pp. 197–202.

[18] G. N. Yannakakis and J. Hallam, "Ranking vs. preference: a comparative study of self-reporting," in *Proc. Conf. Affect. Comput. Intell. Inter. (ACII'11)*, Berlin, Heidelberg, 2011, pp. 437–446.

[19] N. Shaker, G. N. Yannakakis, and J. Togelius, "Feature analysis for modeling game content quality," in *Proc. IEEE Conf. Comput. Intell. Games (CIG'11)*, 2011, pp. 126–133.

[20] G. N. Yannakakis and J. Hallam, "Game and Player Feature Selection for Entertainment Capture," in *CIG'07*, 2007, pp. 244–251.

[21] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[22] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proc. IEEE Conf. Vis. Pattern Rec. (CVPR'11)*. Washington, DC: IEEE Computer Society, 2011, pp. 1297–1304.

[23] A. Elo, *The Rating of Chessplayers, Past and Present*. Arco, 1972.