Rapid Skill Capture in a First-Person Shooter

David Buckley, Ke Chen, *Senior Member, IEEE*, and Joshua Knowles

***Abstract*—Various aspects of computer game design, including adaptive elements of game levels, characteristics of "bot" behavior, and player matching in multiplayer games, would ideally be sensitive to a player's skill level. Yet, while game difficulty and player learning have been explored in the context of games, there has been little work analyzing skill *per se*, and how this is related to the interaction of a player with the controls of the game—the player's input. To this end, we present a data set of 476 game logs from over 40 players of a first-person shooter game (*Red Eclipse*) as a basis of a case study. We then extract features from the keyboard and mouse input and provide an analysis in relation to skill. Finally, we show that a player's skill can be predicted using less than a minute of their keyboard presses. We suggest that the techniques used here are useful for adapting games to match players' skill levels rapidly, arguably more rapidly than solutions based on performance averaging such as TrueSkill.**

***Index Terms*—First-person shooter, player modeling, skill capture, skill measures, skill prediction.**

I. Introduction

SKILL is an important component of any recreational or competitive activity. Not only does it contribute to the outcome, but the relationship between skill and the difficulty of the activity affects the experience of those taking part. More specifically, players in a game often have the most fun when their skill is matched equally, rather than dominating novices or being dominated by highly accomplished players [1].

In our research, skill is a property of a player, defined in terms of their average performance. This definition discounts notions of "skillful" behavior other than those that aid in winning the game [2]. The definition used here falls in line with existing skill measures [3], [4], and allows skill to be explicitly measured.

If a player's skill were known before they played, their opponents could be selected in a way that would optimize their experience of the game. In competitive games, this is known as matchmaking, and is widely used in online gaming. Single player games, on the other hand, use dynamic difficulty adjustment (DDA) [5], [6], where the game's difficulty is changed according to the player's progress. *Left 4 Dead*'s AI director is an example of this in action [7].

Unfortunately, there is currently no quick and accurate way of measuring a player's skill. Bayesian methods, such as TrueSkill [4], require several games before converging, depending on the number of players, and DDA relies on heuristic methods which are not necessarily representative of a player's skill [5]. In a domain where a single bad experience can alienate a player, two or three games are too many, so we seek to reduce this to a single game or less.

Whereas a player's performance may depend on several factors including their opponents, their input, e.g., mouse and key presses, is consistent over several games. It is intuitive to assume that a skilled player will interact with the controls differently to a novice [8]. Instead of relying on performance as a measure for each player, we consider using their input and hypothesize that it could better predict skill from fewer games.

Towards this goal, we have performed a systematic study based on *Red Eclipse*, a first-person shooter (FPS). Game logs were automatically recorded during the study, storing input events, some game events and a few common measurements of performance. In order to understand these measurements, we present a thorough analysis of them and the features extracted from the input events. Building on the success of random forests in previous work [9], we then predict the player's skill with reasonable accuracy from only 30 s of data.

Our main contribution is a model capable of predicting a player's skill within a single game. As a minor contribution, we also provide a complete data set of games containing player input and game results, and some analysis of this data set, exploring its connection to player skill.

The rest of this paper is organized as follows. After a review of previous work in Section II, the data set is described in detail in Section III. We then use the techniques presented in Section IV to analyze the data in Section V and present the skill prediction in Section VI. Finally, we discuss the implications of this research in Section VII and offer our concluding remarks in Section VIII.

II. Previous Work

We define skill as the average level of performance over a set of games. A value of skill only holds meaning for a particular set and for a particular averaging technique. This definition does not consider concept drift or learning, and assumes skill is averaged over a reasonable length of time. Note that the definition of skill used here is distinct from the term "ability" defined by Parker and Fleishman [10]: "Ability refers to a more general, stable trait of the individual which may facilitate performance in a variety of tasks. … The term skill is more specific: it is task oriented."

Performance is the value assigned to a person after a particular task has been completed. This value, or measurement, is defined by a measure, where different measures may yield different performance measurements for the same task (e.g., the player with the highest score may have taken the most damage),

Manuscript received November 06, 2014; revised April 04, 2015, July 06, 2015; accepted August 21, 2015. Date of publication October 27, 2015; date of current version March 15, 2017. This work was supported by the Engineering and Physical Research Council under Grant EP/I028099/1.

The authors are with the School of Computer Science, University of Manchester, Manchester M13 9PL, U.K. (e-mail: david.buckley@cs.man.ac.uk; ke.chen@manchester.ac.uk; j.knowles@manchester.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCIAIG.2015.2494849

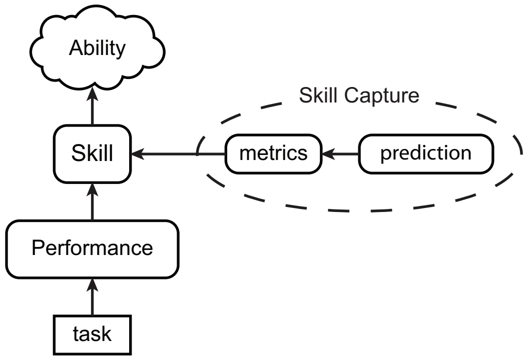

Fig. 1. Connections between relevant concepts used in this paper.

and the choice of measure used affects the rankings of players. The connection between skill and performance has been illustrated in Fig. 1, and is similar to the connection Chomsky draws between competence and performance [11].

We differentiate between a skill measure, which is calculated by averaging performance over time, and skill prediction, the process of predicting a skill measure using less information than that required by the measure. Thus, while the prediction may share the same unit as the measure, it is not guaranteed to produce the same ranking. Although both are considered methods of skill capture, this research assumes that a skill measure always has higher validity than a prediction.

A. *Performance and Skill Measures*

There are numerous ways to measure performance of a task, and each video game has its own common measures. *StarCraft* and *Counter-Strike*, for instance, use win-loss measures and rankings respectively, which determine the winner of each game. However, players of these games use their own measures, e.g., actions-per-minute or kill-to-death ratio, to compare themselves. Regardless of their purpose, these can, and often are, averaged to provide players with skill measures.

A common problem with measures is "inflation," where players change their gameplay to manipulate their performance (and consequently their skill measurement), contrary to how the developers intended them to play. Combining and adjusting different measures is done in order to encourage desired behavior [12]. The WN6 algorithm used for *World of Tanks*, for example, takes a variety of measures and combines them using weightings and a series of mathematical operations to produce a single skill measure [13].

TrueSkill, unlike the simple measures previously mentioned, averages performance using Bayesian updating [4]. The model, which is based on the Elo rating [3], actually represents a belief in a player's skill, which can be reduced to produce a skill measure. The model uses rank as its performance measure, and can therefore cope with multiple teams of varying player sizes. The main criticisms of TrueSkill are its time to convergence, which can take several games to find a confident representation, and that values cannot be compared across different leagues [14].

B. *Skill Prediction*

Skill measures have the distinct disadvantage that performance measurements must be taken over a set period of time in order to determine an average. Users of the TrueSkill algorithm, for instance, need to play anywhere between 3 and 100 games, depending on the number of players in each game. Skill prediction techniques seek to determine an individual's skill in significantly less time.

Regan *et al.* extend a chess end-game performance measure [15] to complete chess games [16]. Using the assumption that computers can play better than humans, a player's move is compared with those of a computer to produce a prediction of the player's performance. The authors then use Bayesian averaging over several moves in order to produce a skill prediction.

The task of skill prediction is not limited to games, and also extends to domains such as teleoperations [17] and Human Computer Interaction (HCI) [18], [19]. The work in HCI uses several features of the user's mouse input to predict their skill for a specific task and for the whole system. The useful features that these authors found have been tested in our own research to explore how well the mouse performs within the context of games. The major difference between their research and ours is that the work in HCI focused on a predefined task with specific instructions that the users can learn very quickly. This contrasts with the task used in our own experiments, which is more analogous to "system skill."

Within the domain of video games, there have been a few attempts at skill prediction, using techniques such as physiological monitoring, recording game events, and logging player actions. The first of these, monitoring physiological responses, explored skill in a fighting game [2]. The researchers distinguished between players of different skill using the performance measure "success rate" when inputting commands. However, while the work provides a foundation for further research, there was a very small number of participants and little analysis of the differences between player types. Moreover, physiological data collection can be intrusive, potentially distancing players from immersion, thus changing how they play.

An alternative to physiological data is using information about the game and high-level game events. This sort of data is easy to collect, and useful for other methods of prediction [20]. Mahlmann *et al.* consider this data for predicting completion time in *Tomb Raider: Underworld* [21]. The main focus of the paper was not on player skill, however, and the results of prediction were inconclusive.

Finally, the most closely related research was done in the real-time strategy (RTS) game *StarCraft II* [22]. In this work, Tetske *et al.* successfully predict a player's skill level using "actions," the interactions between the player and the interface, from a substantial data set. *StarCraft II* groups players into leagues that represent different categories of skill. These categories were used for skill prediction. Building on this, our research uses hardware input events to predict continuous skill measures, which are more descriptive of a player's skill than leagues.

III. Data Set

In this work, we provide a medium-sized data set of game logs recording keyboard and mouse input, some basic game events, and a selection of player survey responses. To our knowledge, there does not exist a publicly available data set based on an

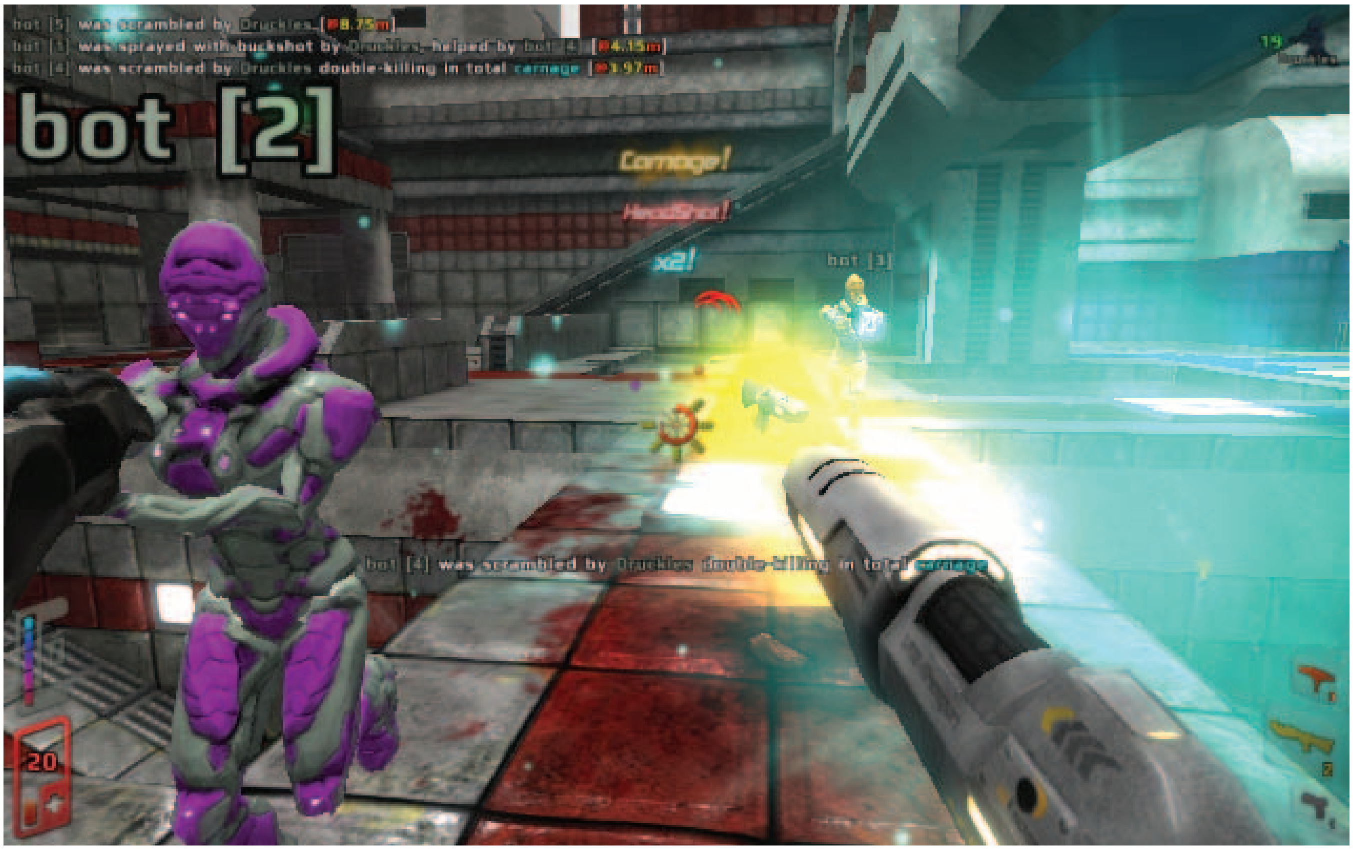
bot [5] was scrambled by Druckles [8.75m]
bot [3] was sprayed with buckshot by Druckles, helped by bot [4] [4.15m]
bot [4] was scrambled by Druckles double-killing in total carnage [3.97m]
bot [4] was scrambled by Druckles double-killing in total carnage

Fig. 2. Screenshot of the game used in our study, *Red Eclipse*.

FPS, a popular video game genre. This data set compliments existing data sets in the literature, such as the MazeBall data set [23], a 3-D predator–prey game; the Platformer Experience Dataset [24]; and a *StarCraft* data set [25], which all include a variety of player feedback types, including keyboard input, physiological data and survey responses. In contrast to these, our data set also provides mouse movement events for each game.

Designed for balance and representation of different player types, the data, and how it was collected is described here. The data set, scripts for manipulating it and further information can be found on our website [26].

A. Red Eclipse

The test-bed for this experiment was an open-source first-person shooter, *Red Eclipse*,[1] which is a fully customizable, fast-paced action game that includes many common game mechanics from the FPS genre. A screenshot of the game can be seen in Fig. 2.

While *Red Eclipse* strives to emulate traditional game mechanics, it also provides a "parkour" system, which is not present in most first-person shooters. The system allows players greater freedom in moving around their environment. The key associated with this parkour system was pressed by many of the players, but only two players used it consistently throughout their games.

The data collected from the games were limited to logging the inputs of the player and select information about the game. A timestamped log file was constructed for each, recording the game's settings and a selection of events, including keyboard and mouse events and some game features such as kill and damage events.

Red Eclipse allows users to modify game settings in order to customize their experience. This includes the type of game they play (the game mode), the arena in which they play (the map), and the difficulty of simulated enemies (bots).

The game mode was set to *deathmatch*, in which players compete to kill each other for the most points. This limited the complexity of rules and tactics used, and meant players were less dependent on the skill of their teammates. In this experiment, however, players only played with bots, and not against human opponents. Each game was set to three minutes; considered long enough for players to become immersed, but short enough to avoid boredom [27].

Eight different maps were chosen in order to represent a range of playing environments. Some maps were more difficult for players, whereas others were harder for the bots. Six ranges of bot difficulty were used (40–50 to 90–100) defining the minimum and maximum difficulty. From a given range, inclusive of the two limits, the engine randomly selects an integer for each bot which defines its skill for that game.

B. The Log File

Each log file has a set of metadata that describes the game and a variable-length list of events. The log files, originally text-based, have been published as `JSON` objects. This is for flexibility and human-readability.

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

TABLE I
METADATA FOR EACH GAME

Name	Description	Example
Game ID	A unique identifier for the game.	127
Player ID	A unique identifier for the current player.	26
Client Number	The number assigned to the player by the game. Always 0 in this experiment.	0
Game Number	From the set of games played by one player, the position this game appears (starting from 0).	5
Map Name	The name of the map that was selected for this game.	wet
Bot Min	Each bot's difficulty is chosen randomly from between Bot Min and Bot Max. Possible values range from 0 to 101.	60
Bot Max		70
Connect time	The time the user connected to the game (ms).	1
Disconnect time	The time the game ended (ms).	185010
Scoreboard	The final scoreboard for the game, including number of points and kills for each player (given by their client number).	0: 'points': 8, 'kills': 3 ...
Date & time	The date the game was played and the time it started.	2013-02-26, 14:40:54

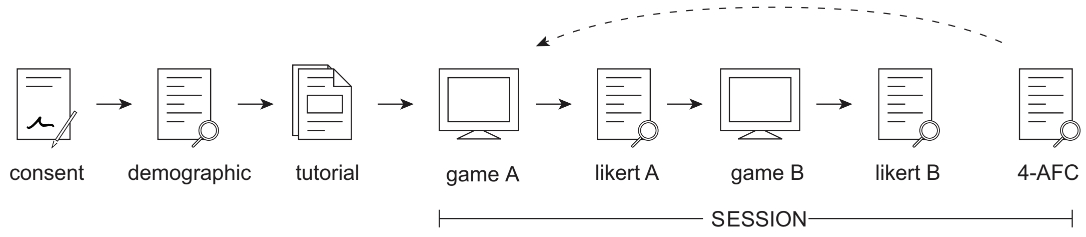

Fig. 3. Overall format of the experiment.

Each game comes with information that describes its settings. The list of metadata can be found in Table I along with a brief description. Although the bot difficulties had a larger range, they were restricted to 40 and 100 in this experiment, as difficulties lower than 40 were considered minimally different.

Two types of events were extracted from the game: input events and game events. Input events were further separated into key presses, mouse button presses and mouse motion. Keyboard and mouse button events contain a key identifier, the final state of the button and the action the button caused in the game. Mouse motion events have an x and y value (the number of pixels the mouse was moved), and were triggered roughly once every three milliseconds while the mouse was in motion.

The second category of events is a simplified summary of game events. These events, generated by the game, only concern events that happen to the player; in other words, interactions between bots is not considered. The events were chosen with the consideration of skill as a focus of the experiment. Some examples included damage taken and dealt by the player, and when points were awarded.

C. Data Collection

The data set was compiled from an in-house experiment. This level of control gave both consistency and reliability to the data set. It also allowed the experimenters to ensure the data set remained balanced throughout.

Although the terms participant and player can be used interchangeably, we have attempted to attribute participant to the context of the experiment, and player to the context of the game.

The overall format for the experiment is presented in Fig. 3 and was similar to a previous study [27], although the questionnaires differed. Each participant started by completing a demographic questionnaire, a summary of which can be found on the website [26]. They were then presented with a written tutorial and given as much time as they needed to read through it. This included a summary of general first-person shooter mechanics and more specific details about *Red Eclipse*. Participants were allowed to ask questions at any point through the experiment or refer back to the tutorial, but the experimenter did not provide information voluntarily.

The main part of the experiment was split into "sessions," where a single session consists of a pair of games and a respective set of questionnaires, as shown in Fig. 3. A participant was allowed to complete as many sessions as they wanted. After each game, the participant answered questions about their experience, and at the end of each session, the participant would compare the two experiences. The questionnaires are described in the next subsection.

All participants used the same keyboard and mouse, and a headset was provided to wear at their discretion. The researcher was present throughout the experiment to guide participants and answer any questions. On three occasions, the researcher had to intervene to ensure participants completed the questionnaires for the previous games. For each of these games, there is roughly an 18-s gap of missing game data. These games are highlighted on the website [26].

Finally, it is worth noting that the data, while only spanning a few weeks, is separated by several months. After the initial study [9], a further period of data collection was held in order to correct imbalances of skill, increase the overall number of games, and increase the number of games per player. From all 45 participants, 14 took part exclusively in the first period, 11 in the second, and 20 took part in both periods.

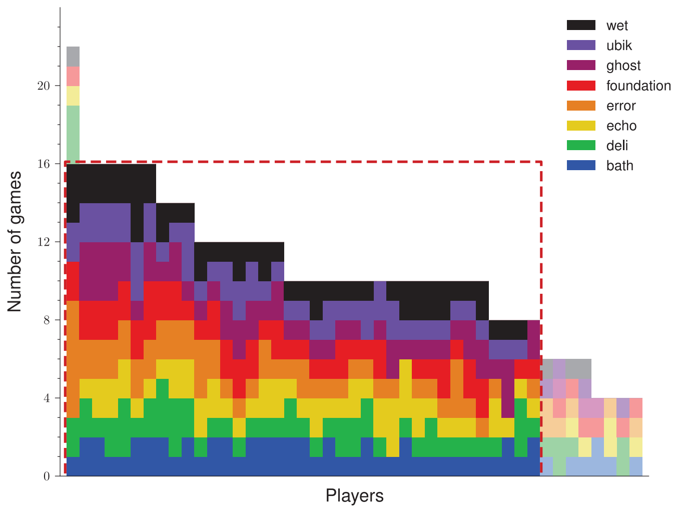

Fig. 4. Number of games played by each player. Games highlighted by the dashed box are those used in this research. Colors indicate which maps each player played.

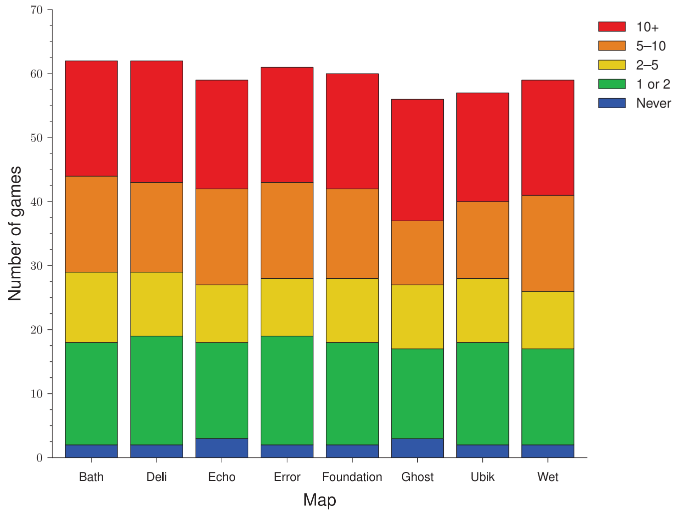

Fig. 5. Number of times each map was played, overlaid by the number of times played by each group in f (FPSs played).

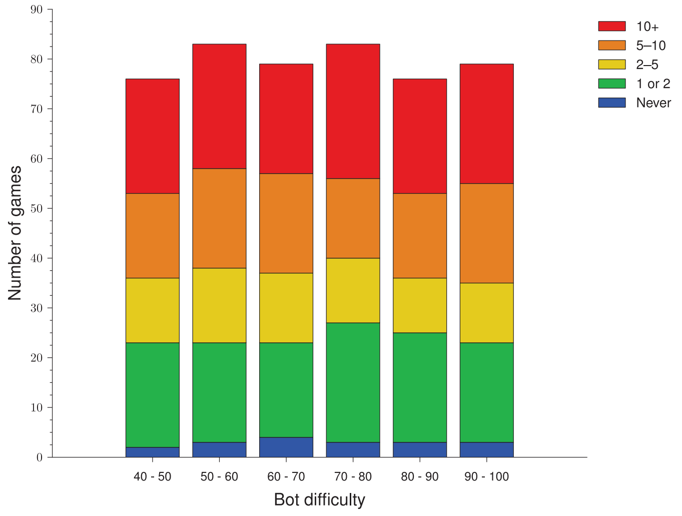

Fig. 6. As in Fig. 5, the number of games played on each difficulty, with additional grouping over f (FPSs played).

D. Questionnaires

There were three different questionnaires used in total throughout the experiment: a demographic questionnaire, an experience-based questionnaire using the Likert scale [28], and an experience-based questionnaire using 4 Alternative Forced Choice (4-AFC) [29].

The demographic questionnaire was presented to participants before they started. This questionnaire gleaned information such as age, gender and, most notably, two self-reported measures of skill. The first measure, how many hours the participant plays per week, is a common question in research [30], [31]. The second, the number of first-person shooters played (f), was conceived to discount the effect of other genres, and account for the player's entire gaming experience, rather than playing habits. These questions were designed to be objective and avoid self-assessment, which players are notoriously poor at [32]. The questions were multiple choice.

- How many hours do you usually play video games in a week? 0–2, 2–5, 5–10, or 10+.
- How many first-person shooters have you played previously? Never, 1 or 2, 2–5, 5–10, or 10+.

From the group of 20 players that took part in both testing periods, there were 6 players who gave different answers for this self-reported skill measure between the two experiments. While most of these discrepancies were off by one category, one player reported playing more (1 or 2 went up to 5–10), while another reported playing less (10+ down to 2–5). Any effects caused by the time delay between experiments has been ignored in this research.

The two experience-based questionnaires used the same questions in two different forms. The first was Likert, to allow the participant to rate each game separately, and the second 4-AFC, comparing the last two games. There are advantages and disadvantages to each method, which are discussed more thoroughly in [33]. Each of these questionnaires had four questions concerning the fun, the frustration, the challenge and the player's impression of the map. The first three questions have been used previously with some degree of success [20], [34]. In our research, the Likert questionnaire was worded as follows.

- How much would you want to keep playing the game?
- How frustrating did you find the game?
- How challenging did you find the game?
- How lost did you feel while playing the map?

The first question, regarding fun, was chosen to allow players to question their current state of feeling, rather than remembering how they felt during the game. This was to mitigate the effects of memory on self-reported affect [35].

E. Data Distribution

The complete data set consists of 476 games from 45 participants. The range of number of games played varied from 4 to 22, and has been visualized in Fig. 4.

As player skill was the main focus of this research, some effort went towards ensuring balance. The number of FPSs played (f) was found to be a better indicator of the two self-reported measures, and was therefore used to validate the balance of the data set. The maps and bot difficulties were selected independently and uniformly at random, adjusted by the experimenter to ensure players and groups of f did not have a biased experience of the game. The final distribution of maps over players is represented in Fig. 4, while the maps and bot difficulties for each skill group is shown in Figs. 5 and 6, respectively.

Although there is an overall imbalance of players according to f, the distribution of the original population is unknown, and the range of different skills observed was considered acceptable for this experiment.

From a preliminary analysis of the first period of data, average player skill leveled out near the sixth game. For this study, we therefore discarded players with fewer than eight games and ignored games played after the 16th, in order to minimize bias. This selection of data (430 games from 37 players) is highlighted in Fig. 4 and has been used throughout the rest of this paper.

IV. Methods

This section reviews the existing measures and algorithms that are used in our experiments. In order to perform the analysis and prediction, we present some skill measures and highlight some methods for evaluating them. We finally introduce some techniques used to extract features from the players' input that are used in Section V.

A. Skill Measures

Many different measures are used to measure skill in a game, some by the developers, and others by the community. In multiplayer games, there is often a ranking of players that determines the order of winners. In *Red Eclipse*, score is used to determine this ranking at the end of each game. Two other performance measures, kill-to-death ratio (KDR) and accuracy, are also used in FPS games to measure how efficient a player is. These are more commonly used by the community to compare players.

The performance measures score (s) and KDR (k) are averaged for each player over all their games to produce a single skill measure, player score $(\bar{s})$ and player KDR $(\bar{k})$. An in-depth comparison between these skill measures and others such as TrueSkill is presented separately [26]. While there are advantages to using other measures, we found $\bar{s}$ the most descriptive measure of skill. $\bar{k}$, on the other hand, may represent play style to some extent, and has been included as a comparison for some experiments.

In order to perform classification and to help analyze the data, four groups were created using $\bar{s}$. Each bin was defined using limits defined in Table II. These groups are used as a substitute for $\bar{s}$ where groups of skill are more appropriate.

B. Evaluating Skill Measures

In classification problems, it is common to evaluate a model using its testing accuracy (or error rate). Within regression (predicting a continuous measure), the proportion of explained variance (R^2) is a common evaluation criterion. This measure and others, including relative absolute error (RAE) [21], punish offset results and those suffering from scaling effects. The values we are comparing, however, are skill measurements; measurements which are ultimately used for ranking players.

TABLE II
Different Groups Separated by Player Score ($\bar{s}$)

$\bar{s}$	Name	Number of Players
< 14	Novice	9
14–22	Intermediate	10
22–27	Skilled	9
≥ 27	Expert	9

TABLE III
Feature Groups Used Within This Research

Group name	Description	Features
Keyboard	From keyboard events	83
Mouse	From mouse movement events	66
Clicks	From mouse clicks	14
Ungrouped	-	11
Event Frequency	Frequency of events over the game	31
Complexity	Complexity of input	75
Kinetics	Describing how the player or mouse moves	19
Ungrouped	-	49
Context-Free	No prior knowledge of game required	78
Dependent	Some knowledge of game semantics needed	96

We therefore use Spearman's rank correlation coefficient (Spearman's ρ) to evaluate our models. This has the added advantage that the ranking of two different skill measures can be compared. Spearman's ρ is defined as the Pearson correlation coefficient [36] between two ranked variables.

C. Measuring Player Input Complexity

A reasonable hypothesis is that skilled players use controls in a more complex way than novices. We therefore use a number of techniques to measure this complexity—some for compression of a sequence and others for analysis on a time-series. These techniques are used to extract features which are then used in Sections V and VI.

The first two, Lempel–Ziv–Welch (LZW) [37] and Huffman coding, can all be used for compression of data. Simple, or more predictable data, should be easier to compress, allowing these to be used to measure complexity. The first, LZW, has the advantage of being simple to implement. The algorithm is as follows.

1) Initialize a dictionary with single-character strings.
2) Find the next longest string, W, in the dictionary.
3) Replace W with the dictionary index.
4) Add ($W+$ next character) to the dictionary.
5) Go to Step 2.

The second algorithm, designed by Huffman [38], constructs a Huffman tree based on probability distributions. Common characters are given smaller codes and placed towards the left of the tree. Encoding involves replacing characters with codes from the tree. If the population distribution of the characters is known, Huffman encoding is close to the theoretical minimum.

In addition to the compression techniques above, two measures of entropy are used: Shannon entropy and sample entropy. The first measures the amount of information in a given sequence

$$H(X) = -\sum_i P(x_i) \log P(x_i).$$

TABLE IV
TOP TEN FEATURES RANKED ACCORDING TO THEIR CORRELATION TO PLAYER SCORE ($\bar{s}$)

Name	Description	Hardware	Type	Context	Pearson's r
lz-bintostr-keybin	The LZ complexity of the keys' binary string.	Keyboard	Complexity	Dependent	**0.7994**
mean-numkeys	The average number of keys pressed at one time.	Keyboard		Free	**0.7896**
length-movement	The total length of the player's movement path.	Keyboard	Kinetics	Dependent	**0.7804**
multikeys/2	The total time for which at least two keys are pressed.	Keyboard		Free	**0.7802**
sum-magnitude-displacement-clickBursts-movement	The player's movement during clicking bursts (periods where the player performs lots of clicking).	Clicks		Dependent	**0.7738**
keytime-requiredkeys	The total time the basic set of keys are pressed.	Keyboard	Event Frequency	Dependent	**0.7663**
length-position	The length of the player's position path.		Kinetics	Dependent	**0.7662**
lz-inttostr-keybin	The LZ complexity of the keys' ASCII string.	Keyboard	Complexity	Dependent	**0.7579**
zlib-bintostr-keybin	The zlib complexity of the keys' binary string.	Keyboard	Complexity	Dependent	**0.7578**
num-keyevents	The number of key press events.	Keyboard	Event Frequency	Free	**0.7573**

The second measure, sample entropy, based on approximate entropy [39], is performed on continuous data and was originally designed for physiological time series. As it is independent of data length, it is potentially useful in understanding the complexity of either mouse or keyboard input.

The final complexity measure used was a discrete Fourier transform [40]. This method reveals regularities in the data and relative strengths of periodic components. Assuming complexities vary with skill, it would be interesting to see how the frequencies of the mouse input compare between users.

V. PLAYER INPUT FEATURE ANALYSIS

Using the methods presented in Section IV and previous work [19], 174 global features were extracted from the keyboard and mouse events of each game.[2] These features are grouped and analyzed in this section in order to better understand player input and how it relates to skill.

Three different schemes, summarized in Table III, were used to group the features. By grouping these features, we can start to see how different types of player input are affected by skill. While the groups of each scheme were designed to be mutually exclusive, some features could not be categorized, so are left ungrouped, and were not used in analysis.

In order to understand the relevance of the features, the Pearson correlation coefficient (Pearson's r) was calculated with respect to player score ($\bar{s}$), chosen as a major index of skill. Strong correlation has been defined here as 0.6, slightly greater than that suggested in previous work [41]. The features that are most correlated with skill have been summarized in Table IV.

A. Hardware: Keyboard, Mouse Movement, and Clicks

The first set of groups separates features according to which input device generated the events. As one of the first obstacles to playing a game, use of the input devices is likely to contribute to skill. In addition, different types of games may have different dependencies on each of the devices.

The features extracted from the *Keyboard* events concerned the complexity of the input or the frequency with which they were pressed. Some of these features were based specifically on the movement keys, which allow the player to move around. An example of this is the amount of time the player spent strafing left and right. A number of mouse movement events have already been used in related HCI research [19], and these formed the basis for the *Mouse* features. Mouse *Clicks*, having been used less in the literature and far more simple in nature, had the fewest features. One set of features (an estimate of the player's position) was created using knowledge of both the mouse and keyboard and, as such, did not fall into one single category. These were ignored for this particular grouping.

The overall correlation of each group has been presented in Fig. 7. Although *Keyboard* contains the most features, it was also one of the more interesting groups, as most features were correlated in some way. The *Mouse* group, on the other hand, correlated significantly less with skill overall. This contrasts previous work in HCI, in which mouse features played a key role [19]. A possible cause of this is the increased noise in a first-person shooter, where the mouse is constantly in motion, and the objectives quickly change. By contrast, the objectives in an HCI task are static. *Clicks* were also generally uncorrelated to skill, the most interesting being the LZW complexity of a player's clicks, with a correlation of only 0.418.

B. Type: Event Frequency, Complexity, and Kinetics

The second grouping scheme is slightly less obvious, in that features are grouped according to what type of input they describe. Some features, for instance, describe the motion of the mouse, referred to here as kinetics, whereas others describe how complex a user's input was (according to the algorithms presented in Section IV). These groups allow us to see what types of player input are most relevant to skill. Unfortunately, there were 49 ungrouped features which did not fall into any of the three groups within this category.

The *Complexity*-based features that well correlated to skill described how complex a player's keyboard input was. For example, the LZW complexity of the four movement keys (forward, left, right and back) correlates highly with skill (Pearson's $r = 0.799$). Skilled players had a higher LZW value, implying their skill is more complex according to the LZW algorithm.

The *Kinetics* group was much smaller than its counterparts. The most interesting features, corresponding to $r \approx 0.48$, include the number of times the player changed the x-direction of the mouse and the average angle of change in a player's movement. That there were few well correlated features in this group suggests that this type of feature is less descriptive of player skill.

[2]The complete list of features and their associated groups have been provided online as a separate document [26].

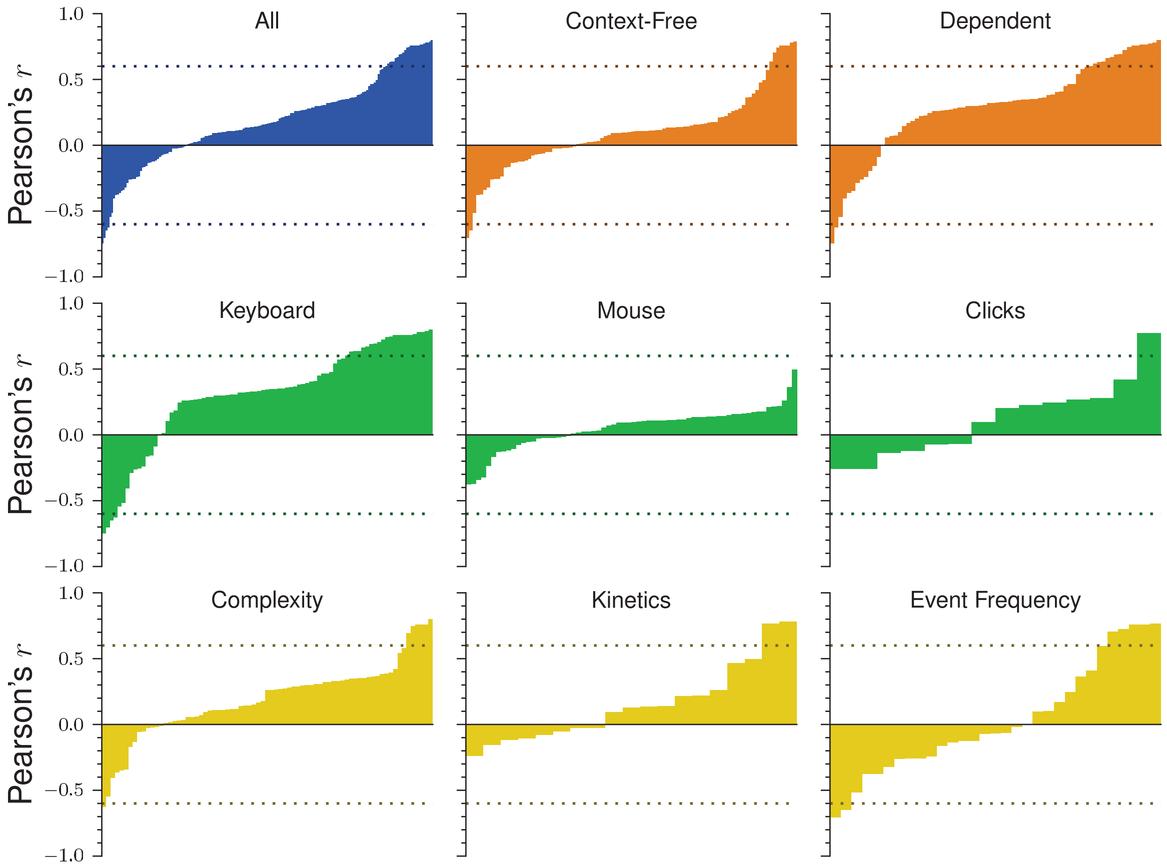

Fig. 7. Pearson correlation coefficient for each feature to player score ($\bar{s}$), grouped by feature group and ordered by correlation. Dotted lines indicate correlation of ± 0.6.

Event Frequency described how often a player generated events with the input devices. While not as prominent as *keyboard*, there were several features in this group that correlated highly with skill, as illustrated in Fig. 7. In general, the higher a player's skill, the greater the number of presses, and the longer each key was pressed. In addition to being well correlated to skill, these features are also relatively easy to compute.

C. Context: Free and Dependent

In an ideal scenario, data collection could be done independently of each game. By splitting the features into those that require some prior knowledge about the game (e.g., the user pressed a key that moves the player forward), and those that do not (e.g., the user pressed the "w" key), we start to understand how independent the features are from the game. This category had the most balanced grouping out of each set. Although similar in size, the *Dependent* group generally had more features that correlated well with skill, as seen in Fig. 7. This indicates that it was easier to create features given knowledge about the game, which is not unexpected. On the other hand, features extracted from the keyboard without knowing anything about the game still contained some information about skill. The length of time any two keys were pressed at once, for instance, had a correlation to $\bar{s}$ of 0.780.

D. Player Learning

The cumulative average score for each score group has been presented in Fig. 8. There is a notable increase in average performance over the first few games for groups *Skilled* and *Expert* which is less visible in the other two groups. Given that only one person had played *Red Eclipse* before, this is consistent with previous research that found skilled players learned faster [42].

Selecting a feature that was particularly highly correlated with player score (the average number of keys pressed at once, which had a Pearson's $r = 0.7896$), we plot the cumulative average value for this over successive games in Fig. 9, again grouping by score group. In contrast to Fig. 8, there is much less variation in value over several games. This may suggest that this feature is a poor indicator of skill because it is invariant to learning effects. However, we argue that the initial learning effects present in Fig. 8 are caused by player acclimatization to the game, rather than increase in skill.

VI. Skill Prediction

This section presents how a player's skill can be predicted from their input to a game. The experiments presented include predicting different classes of skill, predicting continuous skill measures and finally attempting to learn from smaller sections of gameplay. Each experiment uses random forests, which is introduced first.

A. Random Forests

There are several techniques that could be used for predicting player skill. Previous research [22], [43] successfully used support vector machines [44]. However, random forests [45] were chosen for their ability to generalize well, even with a large number of features with unknown properties. A random forest

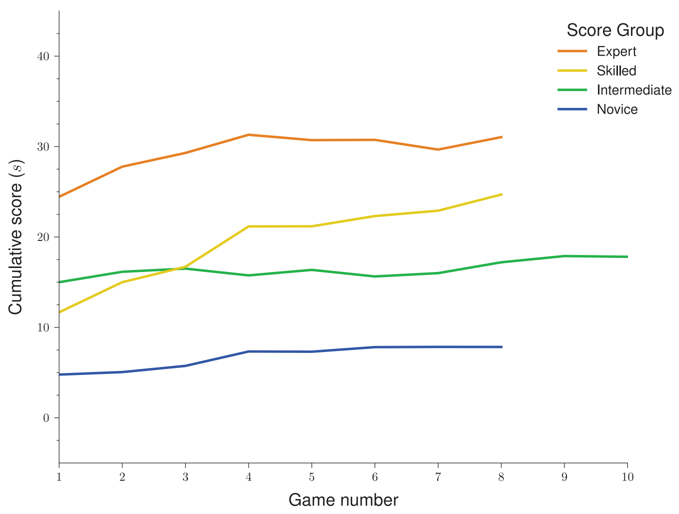

Fig. 8. Cumulative average score (s) over several games for each score group.

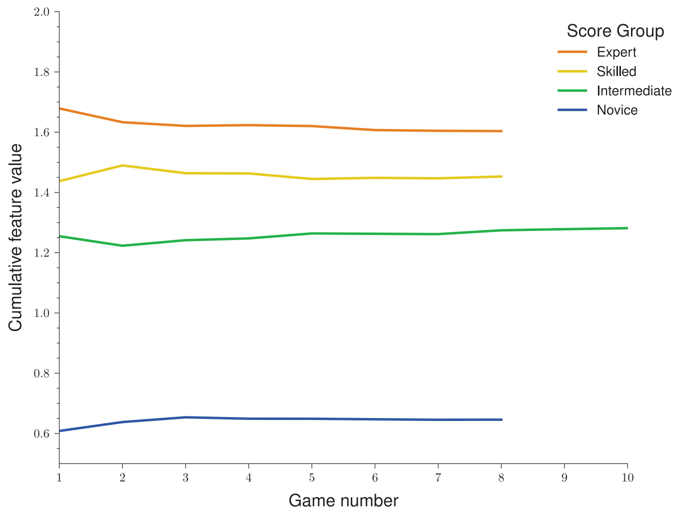

Fig. 9. Cumulative average value for a feature over several games for each score group.

also has the added advantage of being a "gray box," in that it can be used with little knowledge of its internal mechanics, but can tell us which features were the most important during training. Finally, a random forest model can be trained for either classification or regression, which can accommodate the different shapes and sizes of skill measures.

Random forests are an ensemble method that train several trees on different subsets of the data. The MATLAB implementation used was an interface to the R implementation by Andy Liaw *et al.* [46]. Two settings are used during training this model. The first, `ntree`, dictates how many trees to use. This was left on its default setting of 500 for all the given experiments. The second setting, `mtry`, determines how many features are sampled from when a tree is split. This variable was also left on its default setting, $\lfloor\sqrt{D}\rfloor$, where D is the total number of features.

Each experiment presented in this experiment used random forests with 5-fold cross-validation, and the feature groupings from Section V, regardless of correlation to skill measures.

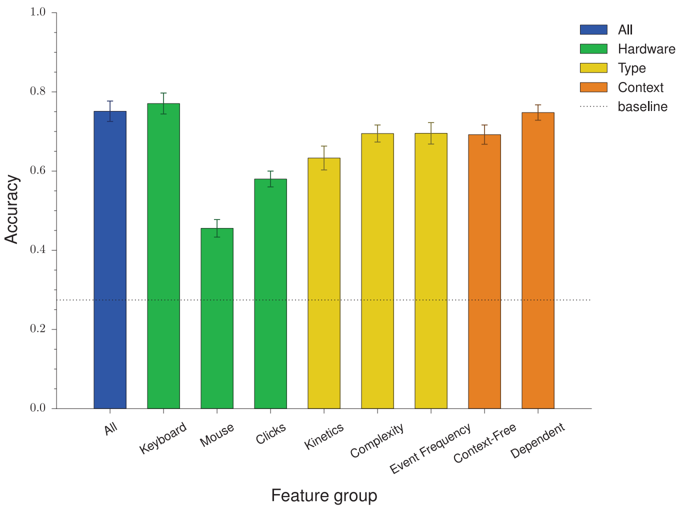

Fig. 10. Mean accuracy of random forest trained to predict a player's score group using different feature groups. Error bars indicate standard error of each model.

TABLE V
HOW EACH GAME WAS CLASSIFIED FOR A RANDOM FOREST TRAINED TO PREDICT GROUPS OF PLAYER SCORE ($\bar{s}$)

	Novice	Intermediate	Skilled	Expert	
Novice	**93**	5	0	0	98
Intermediate	15	**64**	19	20	118
Skilled	0	19	**57**	30	106
Expert	0	9	14	**85**	108
	108	97	90	135	

B. Predicting a Skill Category

Categories of player can be used to get a general idea of how skillful players are. *StarCraft*, for instance, groups players into leagues, where players in the same league are generally comparable in skill [22]. The score groups introduced in Section IV are therefore used to construct a classification model.

The average accuracy for such a model trained on the different feature groups is presented in Fig. 10. An average accuracy of 77.1% is achieved by training on *Keyboard* features, significantly higher than the majority class baseline of 27.4%. The confusion matrix of this model is given in Table V, and shows that many mistakes (77.9% of all misclassifications) are in neighboring classes. The *Intermediate* group was, however, the most difficult to predict.

For some applications, it is often sufficient to be able to distinguish between two kinds of players: those who have never played before, and those who have. For this binary classification, we split the data into two groups: *Novice* and all others. As shown in Fig. 11, the *Context-Free* group achieves an accuracy of 94.9%, whereas the worst group, *Mouse*, performed at 86.2%.

C. Predicting Skill Measures

Most measures of skill use a continuous measurement, allowing detailed comparisons between players. A regression model would allow these continuous values to be predicted for each player, but has not been studied in the literature as

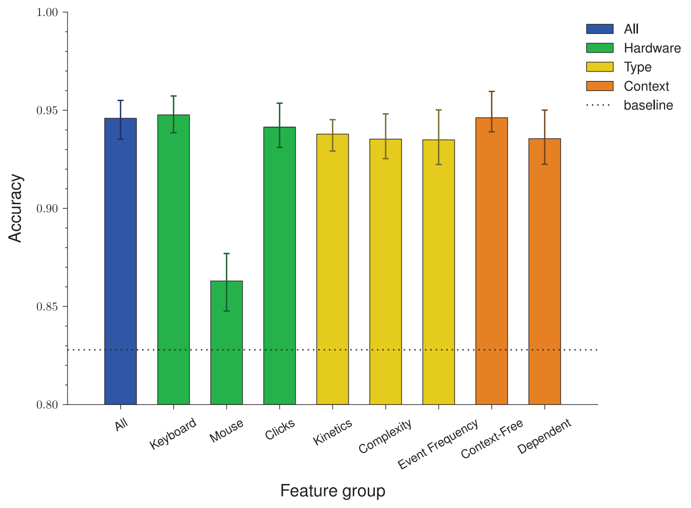

Fig. 11. Mean accuracy of random forest trained to detect *Novice* players using different feature groups.

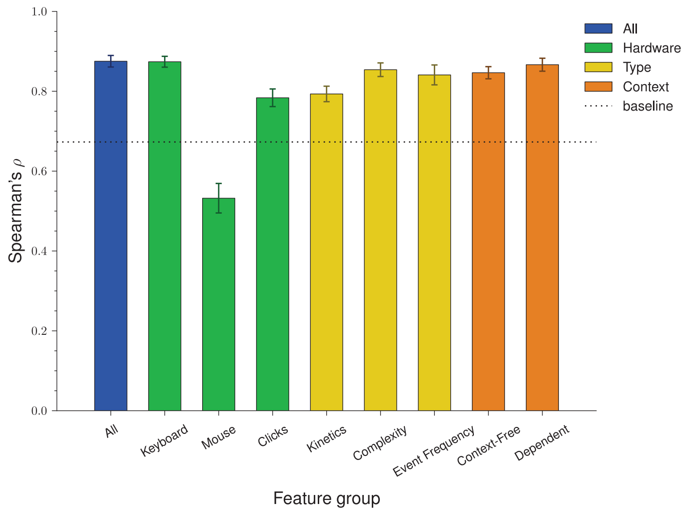

Fig. 12. Performance (Spearman's ρ) of models trained to predict player score ($\bar{s}$). Baseline indicates mean ρ between score (s) and $\bar{s}$.

thoroughly. Predicted values are represented in this research with a hat (i.e., $\hat{\bar{s}}$ is the prediction of $\bar{s}$).

We constructed several models to predict player score ($\bar{s}$) using different feature groups, measuring the performance for each model using Spearman's ρ. The performances for these models are summarized in Fig. 12. The comparative baseline for this experiment is to use the player's performance s (which can be collected after one game), as a substitute for the skill measure, $\bar{s}$, as this is known in real time. $\bar{s}$ is successfully predicted with $\boldsymbol{\rho} = 87.4$, notably higher than s, which has a correlation of only $\rho = 67.3$.

We visualized the predicted values of player score ($\hat{\bar{s}}$) against the predicted values of player KDR ($\hat{\bar{k}}$) for each game in Fig. 13. Overlaid are the actual skill groups for each game. It is clear that the two models agree with each. It also clusters the games into three groups, around $\hat{\bar{s}} < 19$ and $\hat{\bar{s}} > 27$. From this graph, it seems particularly difficult for the model to distinguish between the two highest skilled groups. It may be that the clusters created here related to both skill and player style [8].

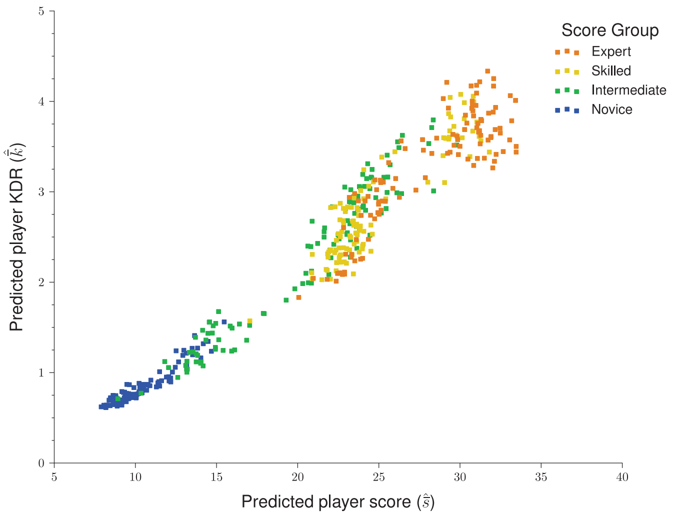

Fig. 13. Relationship between predicted player score ($\hat{\bar{s}}$) and predicted player KDR ($\hat{\bar{k}}$), colored by score group.

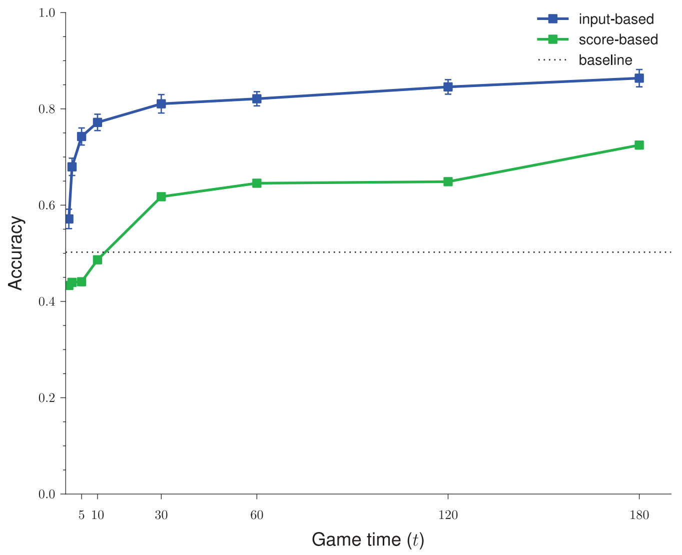

Fig. 14. How fast a classification model is able to predict binary score group. Dotted line indicates mean accuracy guessing the majority class.

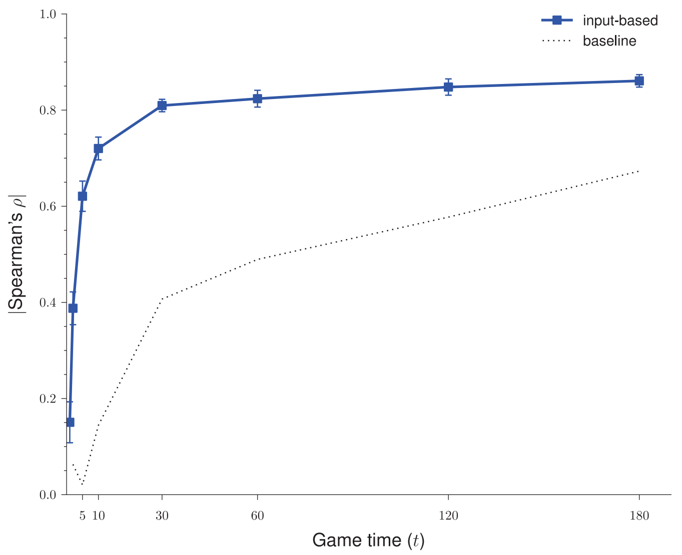

Fig. 15. How fast a regression model is able to predict player score ($\bar{s}$). Baseline indicates mean correlation of the current score at t.

D. Prediction Convergence Rate

The features used up to this point were all extracted from the entire three minutes of gameplay. However, in order to explore how soon a player's skill could be predicted, the same features were extracted from smaller portions of the game, referred to here as *segments*. In addition to the full 180-s segment already used, data was extracted from the first t s of the game, where t was selected from between 1 and 120 s.

Splitting the players into two roughly equally sized groups, *Novice* and *Intermediate* players in one group, the *Skilled* and *Expert* players in the other, we trained the model on the different segment sizes. The result of this is presented in Fig. 14 and compared to a model trained using score (s) as a feature. We performed the same test for a regression model, predicting $\bar{s}$ for each segment of the game. The performance of this is compared to how well the current score at t correlates to $\bar{s}$ in Fig. 15. Not only are the input-based models more accurate than their baselines (using only the current score as a predictor), they start to converge in a very short time (e.g., $t = 30$ s).

Having attempted to use keyboard and mouse data to predict the player's skill in different ways, we highlight some of the more interesting results. Primarily, we find that it is more feasible to predict a player's skill within 30 s of game data using the player's input than it is using the player's performance for that game. In addition to predicting categories of players [22], we were also able to predict continuous skill measures such as a player's average score $(\bar{s})$.

VII. Discussion

This section will discuss the implications and limitations of the data set and predictive model, and outline future work that this research leaves open.

Our data set, presented in detail in Section III, is potentially useful to anyone delving into player input, particularly where two distinct input devices are required (e.g., a keyboard and mouse). Although our research did not offer promising results with regards to the mouse input, it may be that there still exist features of mouse input that can describe a player, their style, or even their skill level. There is a small set of game events available, although limited to basic interactions. This means prediction cannot be done based on higher-level game states such as player positions. The player experience feedback has also been left unexplored, leaving open an entirely different subject of research: how a player's input relates to their experience.

The two major limitations with this data set that may directly affect this research, are: 1) that players only competed against bots; and 2) the lack of expert players for this specific game. Although the measures used here were comparable between players, it is feasible that players would perform differently when competing directly against each other. Additionally, real-world expert players would have more experience than those that took part in our experiment, limiting the direct usefulness of our model. Both of these issues would require further research to explore.

During our analysis of the features, we found that the keyboard was the most descriptive input device of skill. The mouse features, on the other hand, were very weakly correlated to player skill. While useful in previous research [19], it may simply be too random for use in this application.[3] We also showed that even though knowledge of the game was preferred when extracting features, there were features that correlated with skill that required no previous knowledge of the game. Using this, models based on a game-independent approach may be implemented externally to a game. On the other hand, the features extracted were limited to input features. As such, the predictive models may be limited to predicting skill at using the input, or a player's mechanical dexterity, discounting other aspects of skill, such as tactical thinking [47].

There are several key differences between the prediction done in this research and that in previous work [9], [22]. The first is the skill measures used, which are more objective. Secondly, we showed that skill measures such as average score $(\bar{s})$ could be predicted relatively accurately—more so than using a performance measure like s. And finally, this could be achieved within the first half minute of gameplay. Using this model, matchmaking algorithms could initialize skill values for players, and, after a few games, switch to another slower, but more reliable, model such as TrueSkill. This model also addresses TrueSkill's limitation, where players have to compete with each other for TrueSkill values to become meaningful. The presented model takes each new player and predicts their skill independently.

We defined our task as the average skill at deathmatch over a preselected number of maps. This meant that our "ground truth" was the average rank of the player compared to other players, $\bar{r}$. If the task changed, however, to a different game mode, or to a different game, the ground truth would undoubtedly change, and as such, the meanings of these skill measurements. In addition, although each player in our data set experienced a well-balanced proportion of content, traditional games may offer more content to the player, and a player may have a preference for particular maps, skewing a measure such as $\bar{s}$, which was calculated using the mean. As such, different averaging techniques should be compared to account for differences in content. This research also assumed skill to be measured on a singular dimension, averaged from final performance. It may, however, be the case that skill is multidimensional, and, as such, requires a multidimensional model [48].

The most obvious next step with this research is to show how these techniques can be applied. An obvious example, as already mentioned, is matchmaking. Would using a rapid model presented here help improve matchings over the first few games in a matchmaking system? And similarly, in single-player games, it is worth asking whether a rapid model can reliably select the difficulty for players, removing the need for players to learn what the developer means by "normal" or "hard."

Many of the features we collected are relevant to all PC-based first-person shooters. Two possible extensions on this work are either generalizing to other games in the genre or attempting to predict skill on console devices. Difficulties may start to arise with the former when a game's pace changes. The *Counter-Strike* series, for example, are much slower paced than *Red*

[3]Any findings in this research are limited to the types of features extracted. There may yet be other features of mouse movement that correlate well with skill.

Eclipse, and *Team Fortress 2* lets players compete as different classes, each with different styles of play. Console games, on the other hand, control player movement using analogue sticks, which may require completely different types of features.

VIII. Conclusion

This research has presented a solid model for skill prediction in a first-person shooter, reliably doing so within 30 s of gameplay. Moreover, this was done exclusively using a player's input to the game.

The applications for this research can be directly applied to matchmaking and DDA systems, potentially improving player satisfaction in the short-term. However, the models will need to be further refined or adapted when applied to other domains or when using different input devices. The area of research that we intend to explore next is skill capture in single-player games, applying the same methods presented here, and showing how they can be used to improve DDA.

Acknowledgment

The authors would like to thank the people at the University of Manchester for assisting with experiments, and the reviewers for their valuable feedback. A preliminary version of the prediction section can be found in [9].

References

[1] M. Csíkszentmihályi, *Flow: The Psychology of Optimal Experience*. New York, NY, USA: Harper & Row, 1990.

[2] T. Taneichi and M. Toda, "Fighting game skill evaluation method using surface EMG signal," in *Proc. IEEE Conf. Consumer Electron.*, Oct. 2012, pp. 106–107.

[3] A. Elo, *The Rating of Chessplayers, Past and Present*. New York, NY, USA: Arco, 1972.

[4] R. Herbrich, T. Minka, and T. Graepel, "TrueSkill(TM): A Bayesian skill rating system," *Adv. Neural Inf. Process. Syst.*, vol. 20, pp. 569–576, Jan. 2007.

[5] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinforcement learning to provide dynamic game balancing," in *Proc. IJCAI Workshop Reason. Represent. Learn. Comput. Games*, Jul. 2005, pp. 7–12.

[6] 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.

[7] M. Booth, "The AI systems of left 4 dead," in *Proc. 5th Artif. Intell. Interactive Digital Entertain. Conf.*, Stanford, CA, USA, Oct. 2009.

[8] 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.

[9] D. Buckley, K. Chen, and J. Knowles, "Predicting skill from gameplay input to a first-person shooter," in *Proc. IEEE Conf. Comput. Intell. Games*, Aug. 2013, pp. 105–112.

[10] J. F. Parker and E. A. Fleishman, *Ability Factors and Component Performance Measures as Predictors of Complex Tracking Behavior*, ser. Psychological Monographs: General and Applied. Worcester, MA, USA: APA, 1961.

[11] N. Chomsky, *Aspects of the Theory of Syntax*. Cambridge, MA, USA: MIT Press, 1965.

[12] S. Samothrakis, D. Perez, P. Rohlfshagen, and S. Lucas, "Predicting dominance rankings for score-based games," *IEEE Trans. Comput. Intell. AI Games*, Aug. 2014.

[13] "Curious about WN6? read here. Forum," [Online]. Available: http://forum.worldoftanks.com/index.php?/topic/203366-curious-about-wn6%-read-here/ Jan. 2013

[14] P. Dangauthier, R. Herbrich, T. Minka, and T. Graepel, "TrueSkill through time: Revisiting the history of chess," in *Advances in Neural Information Processing Systems 20*. Cambridge, MA, USA: MIT Press, 2008, pp. 931–938 [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=74417

[15] G. M. Haworth, "Gentlemen, stop your engines!," *ICGA J.*, vol. 30, no. 3, pp. 150–156, 2007.

[16] G. Di Fatta, G. Haworth, and K. W. Regan, "Skill rating by Bayesian inference," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, Mar. 2009, pp. 89–94.

[17] Y. Xu and J. Yang, "Towards human-robot coordination: Skill modeling and transferring via hidden Markov model," in *Proc. IEEE Conf. Robot. Autom.*, May 1995, vol. 2, pp. 1906–1911.

[18] A. Hurst, S. E. Hudson, and J. Mankoff, "Dynamic detection of novice vs. skilled use without a task model," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2007, pp. 271–280.

[19] A. Ghazarian and S. Noorhosseini, "Automatic detection of users' skill levels using high-frequency user interface events," *User Model. User-Adapt. Interact.*, vol. 20, no. 2, pp. 109–146, 2010.

[20] 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.

[21] T. Mahlmann, A. Drachen, J. Togelius, A. Canossa, and G. N. Yannakakis, "Predicting player behavior in Tomb Raider: Underworld," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010, pp. 178–185.

[22] T. Avontuur, P. Spronck, and M. van Zaanen, "Player skill modeling in starcraft II," in *Proc. AAAI Conf. AI Interactive Digit. Entertain.*, 2013.

[23] G. N. Yannakakis, H. P. Martínez, and A. Jhala, "Towards affective camera control in games," *User Model. User-Adapt. Interact.*, vol. 20, no. 4, pp. 313–340, Oct. 2010.

[24] K. Karpouzis, G. N. Yannakakis, N. Shaker, and S. Asteriadis, "The platformer experience dataset," in *Proc. Conf. Affect. Comput. Intell. Inter.*, Xi'an, China, 2015.

[25] G. Synnaeve and P. Bessire, "A dataset for StarCraft AI and an example of armies clustering," in *Proc. AIIDE Workshop AI Adversarial Real-Time Games*, 2012.

[26] D. Buckley, K. Chen, and J. Knowles, "Keyboard and mouse data from a first-person shooter: Red Eclipse," [Online]. Available: http://dx.doi.org/10.15127/1.262244

[27] S. Tognetti, M. Garbarino, A. T. Bonanno, M. Matteucci, and A. Bonarini, "Enjoyment recognition from physiological data in a car racing game," in *Proc. Int. Workshop Affect. Inter. Natural Environ.*, 2010, pp. 3–8.

[28] R. Likert, A Technique for the Measurement of Attitudes 1932.

[29] G. T. Fechner, *Elemente der Psychophysik*. Leipzig, Germany: Breitkopf & Härtel, 1860.

[30] A. Canossa, J. Martinez, and J. Togelius, "Give me a reason to dig Minecraft and psychology of motivation," in *Proc. IEEE Conf. Comput. Intell. Games*, Aug. 2013, pp. 1–8.

[31] N. Shaker, S. Asteriadis, G. Yannakakis, and K. Karpouzis, "A game-based corpus for analysing the interplay between game context and player experience," in *Proc. Conf. Affect. Comput. Intell. Inter.*, S. D. Mello, A. Graesser, B. Schuller, and J.-C. Martin, Eds., 2011, vol. 6975, pp. 547–556.

[32] J. Kruger and D. Dunning, "Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments," *J. Personal Social Psychol.*, vol. 77, pp. 1121–1134, 1999.

[33] G. N. Yannakakis and J. Hallam, "Ranking vs. preference: A comparative study of self-reporting," in *Proc. Conf. Affect. Comput. Intell. Inter.*, Berlin/Heidelberg, Germany, 2011, pp. 437–446.

[34] V. M. lvarez Pato and C. Delgado-Mata, "Dynamic difficulty adjusting strategy for a two-player video game," *Procedia Technol.*, vol. 7, pp. 315–321, 2013.

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

[36] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proc. Roy. Soc. Lond.*, vol. 58, no. 347–352, pp. 240–242, 1895.

[37] T. A. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, Jun. 1984.

[38] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[39] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *Amer. J. Physiol.—Heart Circulatory Physiol.*, vol. 278, no. 6, pp. H2039–H2049, Jun. 2000.

[40] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297–301, 1965.

[41] J. Cohen*, Statistical Power Analysis for the Behavioral Sciences*. Mahwah, NJ, USA: L. Erlbaum Associates, 1988.

[42] N. J. Hogle, W. D. Widmann, A. O. Ude, M. A. Hardy, and D. L. Fowler, "Does training novices to criteria and does rapid acquisition of skills on laparoscopic simulators have predictive validity or are we just playing video games?," *J. Surgical Educ.*, vol. 65, no. 6, pp. 431–435, 2008.

[43] A. Drachen, A. Canossa, and G. N. Yannakakis, "Player modeling using self-organization in Tomb Raider: Underworld," in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 1–8.

[44] J. C. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines, advances in kernel methods—Support vector learning Microsoft Research, Tech. Rep. MSR-TR-98-14, 1998.

[45] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[46] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R News* vol. 2, no. 3, pp. 18–22, 2002 [Online]. Available: http://CRAN.R-project.org/doc/Rnews/

[47] D. Conroy, Apr. 2012, A Simple Model for Measuring Skill Ceilings in Video Games. Blog. [Online]. Available: http://tinmangdj.blogspot.co.uk/2012/04/simple-model-for-measuring-skill-in.html

[48] O. Delalleau, E. Contal, E. Thibodeau-Laufer, R. Ferrari, Y. Bengio, and F. Zhang, "Beyond skill rating: Advanced matchmaking in Ghost Recon online," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 3, pp. 167–177, Sep. 2012.

David Buckley received the B.Sc. degree (with first-class honors) in 2011 from the University of Manchester, Manchester, U.K., where he is a currently pursuing the Ph.D. degree in computer science.

His research mainly concerns adapting games to match player skill, with a focus on first-person shooters. He is also interested in general player modelling, level adaptation, and skill capture in other domains, including language acquisition.

Ke Chen (M'97–SM'00) received the B.Sc., M.Sc., and Ph.D. degrees in computer science in 1984, 1987, and 1990, respectively.

He has been with The University of Manchester, Manchester, U.K., since 2003. He was previously with The University of Birmingham, Peking University, The Ohio State University, Kyushu Institute of Technology, and Tsinghua University. His current research interests lie in machine learning, machine perception, and their applications in intelligent system development, including AI video games.

Dr. Chen has been on the Editorial Board of several academic journals, including *Neural Networks* (2011–present) and the IEEE TRANSACTIONS ON NEURAL NETWORKS (2005–2010). He was a Technical Program Co-Chair of the International Joint Conference on Neural Networks (IJCNN'12) and has been a member of the Technical Program Committee of numerous international conferences. In 2008 and 2009, he chaired the IEEE Computational Intelligence Society's Intelligent Systems Applications Technical Committee and the IEEE Computational Intelligence Society's University Curricula Subcommittee. In 2012 and 2013, he was a member of IEEE Biometrics Council AdCom. He is a recipient of several academic awards, including the NSFC Distinguished Principal Young Investigator Award in 2001 and JSPS Research Award in 1993.

Joshua Knowles received the B.Sc. (Hons.) degree in physics with a minor in maths in 1993, the M.Sc. (with distinction) in information systems engineering in 1997, and the Ph.D. degree in computer science in 2002, all from the University of Reading, Reading, U.K.

He is a Professor of Natural Computation in the School of Computer Science, University of Birmingham, Birmingham, U.K., and an Honorary Professor in the Decision and Cognitive Sciences Research Centre, Alliance Manchester Business School, Manchester, U.K. His Ph.D. thesis was part of a second wave of work on evolutionary multiobjective optimization (EMO), helping to establish the field and some of its central topics. Contributions (with various co-authors) have included theoretical work generalizing No Free Lunch, local search methods with provable approximation properties, and ParEGO–one of the better known surrogate-based methods for expensive optimization problems. He currently leads a group (split between Manchester and Birmingham) studying EMO, computational biology, artificial life, and applied optimization. He has worked on optimization or machine learning projects with a number of organizations, including BT, Astra Zeneca, Theo Chocolate, GlaxoSmithKline, Thermo Instruments, and Jodrell Bank Centre for Astrophysics.

Dr. Knowles appeared as Keynote Speaker at the IEEE Symposium Series on Computational Intelligence (SSCI), Cape Town, South Africa, in 2015, and Plenary Speaker at the Surrogate-Assisted Multi-Criteria Optimization (SAMCO) Workshop, Lorentz Centre, The Netherlands, in 2016.