Two Human-Like Imitation-Learning Bots with Probabilistic Behaviors

Chris Pelling
Inventive Dingo *
3507 Palmilla Dr Unit 3044
San Jose CA 95134, USA
chrisp@inventivedingo.com
* *Formerly at The Australian National University*

Henry Gardner
Research School of Computer Science, CECS
The Australian National University
Canberra, ACT 2600, Australia
Henry.Gardner@anu.edu.au

***Abstract*—We present details of two imitation-learning, game-playing agents – "SVMBot" and "PETBot" – that feature probabilistic modelling of some low-level combat behaviours. Both bots used a support-vector-machine approach for *aiming* and a novel, probabilistic model for *jumping* behaviours. PETBot also used a probability-estimation-tree (PET) technique for combat *movement*. The bots were developed for the FPS game *Unreal Tournament 2004* and one of each was submitted to the qualification round (SVMBot) and the final round (PETBot) of the 2009 2K BotPrize competition where they obtained good results. They were then compared with each other as independent variables in a human-computer-interaction (HCI) between-subjects experiment.**

***Index Terms*—competitions, support vector machine, imitation learning, bots, humanness, Turing test, human computer interaction, decision trees, first person shooters, evaluation**

I. Introduction

Many genres of video games incorporate artificial intelligence (AI) agents ("bots") that compete or collaborate with human players. In competitive games, bots can provide opponents of a given skill-level for training purposes or can augment or replace human opponents when true multi-player participation is inconvenient. However, it has been observed that playing against bots can be less enjoyable than playing against human opponents [1] and research has addressed the construction of bots which are more human-like in various ways.

Bots have been evaluated for their "humanness" in Turing-test-like competitions. One of the best known of these competitions is the 2K BotPrize that was introduced at the IEEE Symposium on Computational Intelligence and Games in 2008 [2]. This competition provided cash prizes for developing human-like bots for the first-person-shooter (FPS) game *Unreal Tournament 2004* (UT2004). In the 2008 and 2009 2K BotPrize competitions, each human judge played several rounds of UT2004 with both a human player (a "confederate") and a bot without knowing which was which. Prizes were awarded on the basis of evaluations of the bots by the judges in a Turing-like test. The competition rules were subsequently revised in 2010 to make the evaluation of humanness part of the game itself [3]. The 2K BotPrize is one of several humanness bot-gaming competitions and a review can be found in [4].

In games development, it has historically been the case that expert AI programmers create bots manually for each separate game that requires them. Many effective techniques are available to support this process, from finite state machines (FSMs) to sophisticated planners and behavior trees. UT2004 has its own built-in bots which use hand-created scripts driven by a "fuzzy" FSM. However, whereas it is relatively straightforward to create very challenging FPS bots (by coding near-perfect aim or instant reactions) it is much more difficult to make bots appear as though they are being controlled by a human opponent. One way to reduce the effort involved might be to use *imitation learning*, which performs tasks by attempting to replicate the observed actions of another actor such as a human opponent. The potential of this approach for producing human-like bots is appealing and has been explored by various authors. In [5]–[8], the authors used imitation learning for *reactive behaviors* in low-level tasks such as aiming, firing, selecting weapons and combat movement. A neural network approach to building human-like bots, based on a global workspace architecture, has been described by [9] and the resulting "Neurobot" was very successful in the 2011 BotPrize competition.

This paper describes two bots that were entered into the 2009 2K BotPrize competition together with their evaluation in a human-computer-interaction (HCI) experiment. The first bot, "SVMBot", used a support vector machine (SVM), imitation-learning algorithm for *combat movement* together with a probabilistic approach for jumping. SVMBot was entered into the qualification stage of the competition. A variant of SVMBot, denoted here as "PETBot", was constructed by replacing the SVM modelling of *combat movement* with a probabilistic approach that was based on a probability estimation tree derived from player data. We describe these our two bots in Sections II and III below. We describe their evaluation in competition and in a controlled experiment in Section IV and we conclude with Section V.

II. SVMBot

The starting point for the development of the two bots described in this paper was the AMIS bot by Michal Stolba and

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

Juraj Simlovic [10]. The AMIS bot is an open-source bot for *Unreal Tournament 2004* with behaviors entirely hand-coded in Java using the Pogamut API to interface with UT2004 [11], [12]. The AMIS bot was the winner of the 2008 2K BotPrize competition and we are very grateful that its authors agreed to let us use it to develop our imitation learning bots.

Typically, the designer of an imitation-learning system seeks to reproduce observed behavior by constructing function approximators with defined inputs and outputs and training them on data obtained from human players. Much of the prior work in this field has used Artificial Neural Networks [6]–[8]. We decided to instead use Support Vector Machines (SVMs), partly for convenience, partly because support vector techniques have a number of theoretical advantages (such as always finding a global minimum), and also in the interest of breaking new ground in games bot development.

The Support Vector Machine algorithm involves maximising the following expression with respect to the coefficients a_n for $1 \leq n \leq N$: [13]

$$\sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k\left(\mathbf{x}_n, \mathbf{x}_m\right)$$

where each of the $\mathbf{x}_n$ is an input vector and where each $t_n \in \{-1, 1\}$ denotes the corresponding output category (one of two output classes) for that input vector. Here $k\left(\mathbf{x}_n, \mathbf{x}_m\right)$ is a kernel function with specific properties. Each a_n satisfies $0 \leq a_n \leq C$ and $\sum_{n=1}^{N} a_n t_n = 0$ where C is a tuning parameter. C is typically determined by trial and error for each model, using separate training and test data sets or cross-validation to find the value that gives the best possible classification rate. The minimisation is a quadratic programming problem, for which well-known algorithms exist. Once a solution has been found, the values of a_n are used to calculate a parameter b such that $t_n \left(\sum_{m=1}^{N} a_m t_m k\left(\mathbf{x}_n, \mathbf{x}_m\right) + b \right) = 1$ for any vector $\mathbf{x}_n$ where $a_n \neq 0$. Finally, new data points $\mathbf{x}$ can be *classified* by examining the sign of the following function:

$$y\left(\mathbf{x}\right) = \sum_{n=1}^{N} a_n t_n k\left(\mathbf{x}, \mathbf{x}_n\right) + b$$

If $y(\mathbf{x}) < 0$ then $\mathbf{x}$ is classified as belonging to one class, otherwise it is assigned to the other class. Several techniques exist for extensions to problems involving more than two classes. For our work we used the "one against one" method, where a two-class SVM is trained for all possible pairs of classes, and points are classified according to which class receives the highest number of "votes" [13].

A. Combat Movement

In our construction of SVMBot, we set out to replace many of the AMIS bot's behaviors with imitation-learning components. Training data were extracted from several hours of recordings of a human player competing against one of UT2004's built-in bots. Firstly, we desired to reproduce the *combat movement* patterns which the human player performed in order to evade enemy fire. We modified the approach of [6] to use the following inputs: the player's current health and shield values, the enemy's position (in a Cartesian coordinate system centered on the player), the enemy's velocity, the distance from the player to the walls in four directions (in the hope of modelling spatially-aware movement behaviors) and the identity of the weapons held by both the player and the enemy. The output was a category, with one level for each of eight directions in the plane (Back, Back Left, Back Right, Forward, Forward Left, Forward Right, Left, Right) plus an additional category (Stay) for no movement at all.

Training data were obtained from five separate play sessions with a total duration of 27.5 minutes using the UT2004 levels *DM-1on1-Albatross* and *DM-1on1-Idoma*. After filtering out data points from non-combat situations, the training data set contained 18,427 data points. We then deployed the *libsvm* library [14] to train a support vector machine (SVM) for mapping input into output categories. We used a radial basis function kernel, and tuned its parameters using a brute-force grid search and five-fold cross-validation. This achieved a correct classification rate of 91% on our data-set.

B. Probabilistic Jumping

Jumping is an important part of combat in many FPS games, particularly when avoiding enemy area-of-effect weapons such as rocket launchers. Testing showed that the bot never jumped when jumping was treated as another movement direction in our SVM combat-movement model. This was possibly because there were many more non-jumping than jumping data points in the training data or because all of the jumping data points were very close to non-jumping data points in the input space. Upon reflection, it seemed reasonable to try to model jumping *probabilistically*: Human players do not necessarily jump predictably in any given situation and both the size and the precise timing of jumps can vary. The approach we took was to count the number of data points where the imitated player had jumped, and to divide this by the total number of data points to arrive at a probability value. On each frame, we generated a uniform random number between 0 and 1 and we compared this to the probability value to decide whether to jump or not. We conditioned the probability value on the weapons in play by dividing the counts into one of 81 bins (since there are 9 Deathmatch weapons in UT2004). The bot's jump decision routine looks at its weapon and the weapon of its current enemy to determine the appropriate bin, and uses the probability value calculated for that bin. Jumping lengths were held constant and were comparable to evasive running with a similar time-step.

C. Aiming

We trained two aiming models – one for the pitch and one for the yaw of an angle offset relative to "perfect" aim. The inputs chosen were the current weapon, the distance from the enemy to the bot and the enemy's velocity in a rotated reference frame (having one axis as the perfect aiming direction). We found that it was necessary to reduce this set

of inputs from that reported by [8] in order to obtain aiming behavior which was sufficiently reliable. In particular, the inclusion of the pitch and yaw of the enemy position (in spherical coordinates), as described in [8], led to unreliable aiming behavior in our model. We used the same dataset as for combat movement, and trained the aiming models using Support Vector Regression (SVR). As with our SVM models, our implementation of SVR used *libsvm* [14].

III. PETBot

SVMBot achieved the highest score of all the bots in the qualification trial of the 2009 2K BotPrize competition. In spite of this good result, our in-house testing revealed a number of flaws. In particular, the deterministic combat movement failed to display sufficient spatial awareness. Even though distances to nearby walls were provided as inputs to the movement model, the bot frequently bumped into them. Furthermore, the deterministic nature of SVM was a liability, causing the bot to sometimes become "stuck" in invalid movement states (e.g. running into walls). Noise was added to some inputs to correct this flaw but with only partial success.

A possible solution to these problems appeared to be to include an element of randomness in the models. The need for randomness was supported by the fact that players tend to "dance around" in first-person shooter games to evade enemy fire. In order to be effective, this dance needs to be have an element of randomness in order to keep an adversary off guard. The nature of this dance can be observed in our training data, captured from the human player, shown in Fig. 1. This figure plots traces of enemy locations (in Cartesian coordinates centred on the human player) which are labelled by the human player's choice of movement direction in each frame. It can be seen that there are short sequences of identical markers (representing movement directions) and that these patterns tend to be circular. The identical markers result from the fact that human players tend to stick to moving in one particular direction for a number of frames. The orientation of these sequences reflects the classic "circle strafe" tactic, where opponents circle each other to make themselves harder to hit while remaining at a distance.

Perhaps the simplest possible probabilistic approach to movement is to construct a frequency histogram from the data of the human player's selections from each of the nine movement directions and to sample from this histogram as if it were a discrete probability distribution. The *duration* of each movement also needs to be modelled, since (as is apparent from Fig. 1) human players do not change direction every frame. To give this approach situational awareness, the input space can be split into regions and each region can be given its own histogram. The choice of a combat movement direction is then a matter of finding which region of the input space the data point belongs to and then sampling from the associated histogram. The question remains of how to split the input space into regions. One approach is to use a decision tree [15] which is a binary tree where an inequality on some variable is attached to each internal node. Nodes within the

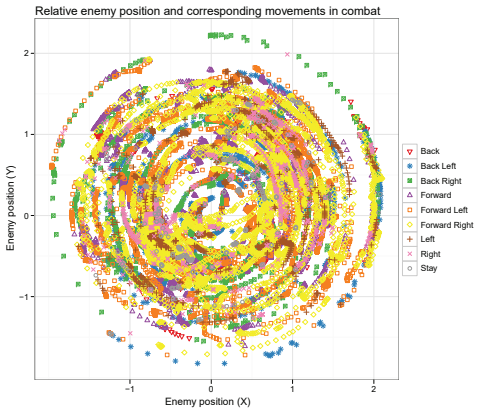

Fig. 1. Plot of enemy locations relative to the human player in the training data, with markers showing the player's current movement direction in each case.

left subtree correspond to situations in which the inequality is false, and nodes within the right subtree correspond to situations in which the inequality is true. Decision trees are simple and can be easily interpreted and modified by humans but their principal disadvantage is that the region boundaries are always hard and axis-aligned. In a classic decision tree, each leaf node specifies a single outcome but this can be easily extended to the probabilistic case by replacing those single outcomes with discrete probability distributions. The resulting tree can be called a *probability estimation tree* (PET) [16] or a *class probability tree* [17] (although the undifferentiated term *decision tree* has also been used [18]).

We constructed a PET based on probability distributions derived from the same combat movement training data as for the SVMBot. The tree was constructed automatically using the "C4.5 algorithm" of [19] with leaf nodes being replaced by probability distributions in a post-processing step. The data set contained all of the input parameters used for SVMBot but was augmented by an additional "recent damage" parameter to represent the amount of damage done to the bot during the past two seconds. It turned out that our decision-tree algorithm made heavy use of the "distance to walls" parameters which gave us confidence that the model would go some way towards correcting some of the flaws of the SVMBot. An excerpt from the resulting tree is shown in Fig. 2.

A. Movement Duration

Movement duration was modelled by recording player movement durations and placing them in one of nine lists (one for each of the eight directions in the plane as well as one for no movement) and then selecting randomly from the appropriate list. The shapes of the probability distributions as a function of movement duration are shown in Fig. 3. It can be seen that these distributions are noticeably different from each other and that there is a tendency of straight movements (Forward, Back, Left, Right) to be shorter in duration than the diagonal movements. This might reflect the imitated player's

Recent damage ≤ 0

DIST_BACK ≤ 0.139815

DIST_RIGHT ≤ 0.089745

	Left	-	Right
Forward	38	80	115
-	21	30	30
Back	0	24	18

DIST_RIGHT > 0.089745

DIST_LEFT ≤ 0.12852

(...)

DIST_LEFT > 0.12852

DIST_RIGHT ≤ 0.27881

Weapon = Shield (no data)

Weapon = Assault (...)

Weapon = Bio (...)

Weapon = Shock (...)

Weapon = Link (...)

Weapon = Mini (...)

Weapon = Flak (...)

Weapon = Rocket

	Left	-	Right
Forward	7	12	34
-	18	6	41
Back	13	4	20

Weapon = Sniper

	Left	-	Right
Forward	0	0	0
-	2	0	1
Back	3	1	0

DIST_RIGHT > 0.27881

(...)

DIST_BACK > 0.139815

(...)

Fig. 2. Excerpt from the decision tree used for the final PET-based combat movement implementation. Each variable beginning with "DIST_" is the distance to the nearest wall or other map obstacle in the indicated direction. The frequency histograms are shown as tables under the relevant leaf nodes for the forward/neutral/backward and left/neutral/right directions of movement. The neutral columns and rows correspond to the probability of no movement.

preference for diagonal movements, or it might have been a consequence of the interface: Since diagonal movements were performed by holding down two straight-direction control keys simultaneously, many short straight movements might have been generated as a result of failing to press or release both keys at the same time.

IV. Humanness Evaluation

Our two bots have been evaluated for humanness against other bots in competition, and against each other in a controlled laboratory experiment. The 2009 2K BotPrize competition had two phases – a qualification round (in late July 2009) and a final round (in September 2009). The competition rules allowed qualifying bots to be modified prior to the final competition which, in our case, led to SVMBot being replaced by PETBot in the final round. The controlled laboratory experiment was run following the final BotPrize round using a traditional HCI experimental methodology.

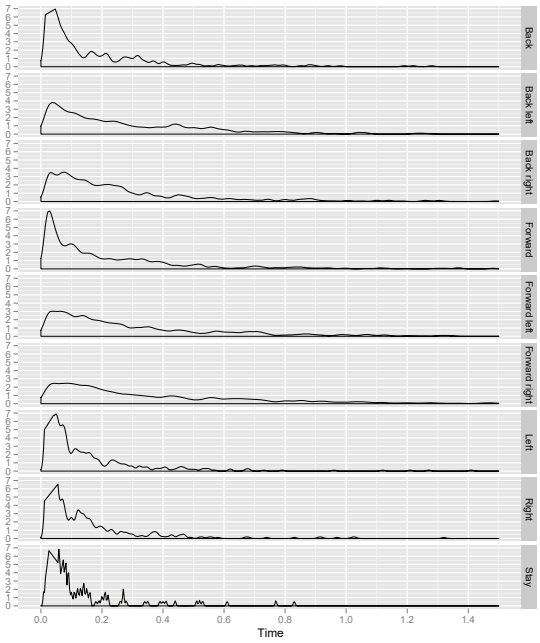

Fig. 3. Density plots of movement duration for each possible movement direction, including the no-movement direction ("Stay").

TABLE I
Categories used to judge players in the 2009 2K BotPrize

Label	Category text
0	This player is a not very human-like bot.
1	This player is a fairly human-like bot.
2	This player is a quite human-like bot.
3	This player is a very human-like bot.
4	This player is human.

A. The 2009 2K BotPrize competition

In the qualification trial of the 2009 2K BotPrize competition, each of the 15 entrants took part in three rounds of five ten-minute games with three humans and three bots playing together in each game. After each game, the three humans rated their opponents on the scale shown in Table I and the top five bots went through to the final round. Of the bots in the qualification round, SVMBot obtained the highest mean score (of 1.75) but further details of this round have not been made available.

TABLE II
SCORING OF BOTS IN THE FINAL ROUND OF THE 2009 2K BOTPRIZE COMPETITION (TAKEN FROM [21])

	Judge scores					
Entry	#1	#2	#3	#4	#5	Mean
sqlitebot	2	4	2	2	2	2.4
PETBot	0	4	2	3	1	2.0
ICE-2009	3	4	1	1	1	2.0
BradBot	2	1	4	0	0	1.4
UTAustin	0	4	0	1	0	1.0

Twenty-five games were played in the final round of the 2009 2K BotPrize. Five human judges and five human confederates were recruited. The judges were all game developers or AI researchers. Each game was played between one judge, one bot, and one human confederate, such that each judge met each bot and confederate exactly once. Judges were tasked with identifying which of their opponents was human, and rating each opponent using the numerical scale of Table I. Confederates were instructed to play normally as if trying to win the game, and they were motivated to do this with a 100 Euro prize for earning the most game points. Judges and confederates were optionally allowed to provide comments together with their scores [20].

The five finalists are listed in Table II together with the scores they received from each judge in the final trial. (This data has been taken from [21] – with the name "PETBot" substituted for "Anubot" in that reference.) Each bot managed to fool *exactly one* judge into giving it a rating of 4 ("is human"). Rankings derived from the *mean scores* in that table showed "sqlitebot" to be the winner of the final round with PetBot tied for second place.

There is evidence that the humanness of bots has improved since 2009. Figure 1 of [22] aggregates humanness scores (according to a metric related to judges' ratings) of a number of bots, including "Anubot", in the BotPrize competitions from 2008 to 2012. Later years show a marked increase in aggregate humanness by these bots. Even though the protocols of competitions after 2009 changed significantly [3], this evidence of steady improvement in bot humanness is compelling.

B. Controlled Experiment

We decided to run a formal laboratory test to attempt to distinguish SVMBot and PETBot and to gauge the humanness of each. We adopted an HCI methology to run this experiment. Participants were formally recruited from an advertisement requesting volunteers with first-person-shooter gaming experience. Before playing the game, participants were provided with an information sheet and a consent form. After the game, they were asked to fill in a short questionnaire that requested a small amount of demographic information and asked them to rate the humanness of the "bots" they were playing against (as in the final round of the 2006 2K BotPrize competition, one "bot" was a human). Participants were also asked for any observations that they had about each bot and any suggestions that they had to make that bot appear more human. The experiment protocol was approved by the Human Ethics Committee of our university.

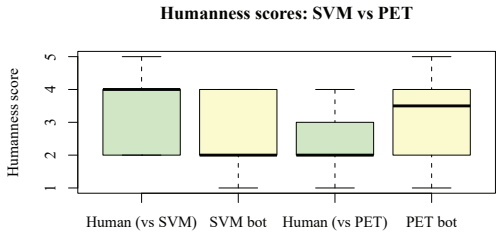

Fig. 4. Box plots of humanness ratings per player and per bot type.

Twenty participants were recruited for our experiment. All were males with fifteen between the ages of 15 and 24, three between the ages of 25 and 34, and one each in the age groups 35-44 and 45-54. All had previously played first-person shooter games with most rating themselves as slightly above average in skill. Most participants had played *Unreal Tournament 2003* or *2004* a few times.

Each individual test was run as a three-way game of UT2004 with the three players being: the study participant, a human confederate, and a bot. The bot was chosen to be one of SVMBot and PETBot, alternating from one experiment to the next. Each participant only played against one of SVMBot or PETBot and the human confederate was the same in each game. Each game was run for 15 minutes (compared to the 10 minutes per game used in the 2009 2K BotPrize).

For this experiment, we asked participants to rate humanness using a rating scale that kept the spirit of the 2009 2K BotPrize scale but conformed better to traditional anchors used in Likert surveys. Our rating scale was similar to the "Likelihood" set of anchors described by [23] but used terminology taken from the BotPrize Turing test:

(1) This player is definitely a bot.
(2) This player is probably a bot.
(3) I can't tell whether this player is a human or a bot.
(4) This player is probably a human.
(5) This player is definitely a human.

Figure 4 shows box plots of rating data obtained from this experiment. From these plots, it can be seen that the *median* score for the SVMBot (of 2) was less than the median score (of 3.5) for the PETBot. Had this been a competition, these numbers would have made PETBot the winner. However,

statistical significance for rejecting the null hypothesis of the two bots having identical humanness ratings was not obtained (a p-value of $p \approx 0.5$ was obtained using the non-parametric Wilcoxon test for this condition). Of the various comparisons evident in Fig. 4, the most significant was that comparing the perceived humanness of the human confederate when playing in games including SVMBot as compared to those using PETBot ($p \approx 0.13$) but this was still not statistically significant at the traditional $p = 0.05$ level.

C. Qualitative Analysis

In order to derive recommendations for improvement in our bots, we combined the written comments made by the participants in our HCI test with written feedback we received from the judges in the 2009 2K BotPrize and performed a textual analysis. We drove this analysis by the two questions "What went right?" (i.e. Why were players tricked into thinking that our bots were human and *vice versa?*) and "What went wrong?" (i.e. Why did players make the correct identifications?). We obtained a corpus of about 3,500 words which we split into 229 separate comments by topic in order to perform this analysis following a focus group methodology [24]. Several distinct trends were identified, and the major recommendations for improving on our bots are listed below. Apart from these recommendations, we found that there were no clear differences in comments received regarding *combat movement* in SVMBot and PETBot. This was surprising to us because improving combat movement was the rationale for replacing SVMBot by PETBot in the first place. We also found that comments indicated that the human confederate was sometimes considered to be just "too accurate" and "too fast" to be really human!

The strongest recommendations that we obtained for improving our bots were the following:

1) Bots should visibly respond when fired upon by unseen opponents. About half of the untricked participants commented on the bots' lack of awareness when shot from behind. This would have been an easy fix for our bots.
2) Bots should have more unpredictable, evasive, tactical and spatially-aware movements. Avatars identified as bots had movements which were predictable, repetitive and became stuck on walls. Human avatars, when correctly identified, had movements which displayed good dodging, avoidance and jumping.
3) Bots should not display reaction time delays in their movements at inappropriate times. About half of the untricked participants noted that the bots paused or stopped at inappropriate times.
4) Bots should aim *intelligently*. For example, they should cleverly lead the target. But at the same time, bots need not aim *accurately*! Some tricked participants cited the poor aim of the bot as being a reason why they thought it was human. Some cited the the real human's accurate aim as a reason for it being a bot. Some participants commented on the correctly-identified human avatar's precise aiming and good use of leading.
5) Bots should be intelligent about "running the map" to collect items such as health, shields, powerups and weapons. Some participants commented on the (correctly identified) human avatar's good acquisition of these items. A few participants, and one BotPrize judge, commented that our bot was bad at acquisition.
6) Bots should know when to attack and when to avoid engagement or to retreat. They should, however, continue to display aggressiveness. Some participants noted that the bots never retreated. About half of the participants mentioned the human avatar's consistently aggressive playing style (although a few mentioned that he was too aggressive!).

V. Conclusion

We have discussed two designs for imitation-learning bots. Both were based on support vector techniques and include a *novel probabilistic model for in-combat jumping*. One bot also includes a *application of probability estimation trees to in-combat movement*. Both designs appeared viable when tested under laboratory conditions and in the competition format of the 2009 2K BotPrize.

As mentioned above, later years of the BotPrize competition have seen a continued interest by bot developers and improvements in bot humanness [22]. In the hope that our experiences might be useful as this field progresses further, we have listed the suggestions we received on how to improve our bots in the previous section. In addition to these recommendations, future work might build on our bots in several ways. In particular, we constructed PETBot's Probability Estimation Tree by adapting an algorithm designed to construct decision trees with *deterministic leaf nodes*. It is possible that a PET algorithm which operates directly on probability distributions might give better results. Other methods of estimating probability distributions could also be applied and the nature of the density plots obtained for movement duration showed an interesting structure which could be another matter worthy of future investigation.

The design of Turing-test competitions such as the BotPrize continues to be a topic of active research (see, for example, [4], [25]). A crowd-sourced version of our HCI experimental approach could provide an alternative to such competitions for a developer who wished to verify that improvements made to bots actually did appear to be more human. In such an evaluation, both human confederates and human participants could be sourced on the internet using a platform such as Amazon Mechanical Turk. The larger number of participants obtained using such an approach would enable statistical significance to be more easily obtained than in our experiment and that significance could be used as a decision point to adopt one version of a bot over another as improvements are made.

Acknowledgment

We gratefully acknowledge Michal Stolba and Juraj Simlovic, for agreeing to let us use their AMIS bot as a basis

for our development and competition entry. Thanks to Penny Kyburz for useful discussions and encouragement.

REFERENCES

[1] P. Sweetser, D. Johnson, J. Sweetser, and J. Wiles, "Creating engaging artificial characters for games," in *Proceedings of the second international conference on Entertainment computing*. Carnegie Mellon University Pittsburgh, PA, USA, 2003, pp. 1–8.

[2] P. Hingston, "A Turing test for computer game bots," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 3, pp. 169–186, 2009.

[3] ——, "A new design for a Turing Test for Bots," in *2010 IEEE Symposium on Computational Intelligence and Games (CIG)*. IEEE, 2010, pp. 345–350.

[4] C. Even, A.-G. Bosser, and C. Buche, "Analysis of the protocols used to assess virtual players in multi-player computer games," in *International Work-Conference on Artificial Neural Networks*. Springer, 2017, pp. 657–668.

[5] B. Geisler, "An empirical study of machine learning algorithms applied to modeling player behavior in a first person shooter video game," Ph.D. dissertation, University of Wisconsin, 2002.

[6] S. Zanetti and A. E. Rhalibi, "Machine learning techniques for FPS in Q3," in *ACE '04: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*. New York, NY, USA: ACM, 2004, pp. 239–244.

[7] C. Bauckhage and C. Thurau, "Towards a Fair'n'Square Aimbot–Using Mixtures of Experts to Learn Context Aware Weapon Handling," in *Proc. GAME-ON*, 2004, pp. 20–24.

[8] B. Gorman and M. Humphrys, "Imitative Learning of Combat Behaviours in First-Person Computer Games," in *Proceedings of the Tenth International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games (CGAMES)*, 2007.

[9] D. Gamez, Z. Fountas, and A. K. Fidjeland, "A neurally controlled computer game avatar with humanlike behavior," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 1, pp. 1–14, 2013.

[10] M. Stolba, J. Simlovic, and J. Gemrot, "AMIS bot description and source code," http://web.archive.org/web/20120814174348/-https://artemis.ms.mff.cuni.cz/pogamut/tiki-

[21] ——, "2K BotPrize 2009 results," http://web.archive.org/web/20090923153541/-http://www.botprize.org/Results2009.xlsx. (Retrieved 28 May 2019), 2009.

index.php?page=Botprize+2008+winning+bot. (Retrieved 28 May 2019), 2009.

[11] O. Burkert, R. Kadlec, J. Gemrot, M. Bída, J. Havlícek, M. Dorfler, and C. Brom, "Towards fast prototyping of IVAs behavior: Pogamut 2," *Lecture Notes in Computer Science*, vol. 4722, p. 362, 2007.

[12] J. Gemrot, R. Kadlec, M. Bida, O. Burkert, R. Pibil, J. Havlicek, L. Zemcak, J. Simlovic, R. Vansa, M. Stolba, T. Plch, and B. C., "Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents," *Lecture Notes in Computer Science*, vol. 5920, p. 1, 2009.

[13] C. Bishop, *Pattern recognition and machine learning*. Springer, 2007.

[14] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm (Retrieved 28 May 2019).

[15] J. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[16] F. Provost and P. Domingos, "Well-trained PETs: Improving probability estimation trees," *Raport instytutowy IS-00-04, Stern School of Business, New York University*, 2000.

[17] W. Buntine, "Learning classification trees," *Statistics and Computing*, vol. 2, no. 2, pp. 63–73, 1992.

[18] D. M. Magerman, "Statistical decision-tree models for parsing," in *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 1995, pp. 276–283.

[19] J. Quinlan, *C4.5: programs for machine learning*. Morgan Kaufmann, 2003.

[20] P. Hingston, "2K BotPrize 2009 rules," http://web.archive.org/web/20090221151001/-http://www.botprize.org/rules.html. (Retrieved 28 May 2019), 2009.

[22] M. Polceanu, "Mirrorbot: Using human-inspired mirroring behavior to pass a turing test," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. IEEE, 2013, pp. 1–8.

[23] W. M. Vagias, "Likert-type scale response anchors. clemson international institute for tourism," *& Research Development, Department of Parks, Recreation and Tourism Management, Clemson University*, 2006.

[24] R. A. Krueger and M. A. Casey, *Focus Groups: A Practical Guide for Applied Research*, 4th ed. Sage Publications, 2008.

[25] C. Even, A.-G. Bosser, and C. Buche, "Bot believability assessment: A novel protocol & analysis of judge expertise," in *2018 International Conference on Cyberworlds (CW)*. IEEE, 2018, pp. 96–101.