We can only see a short distance ahead, but we can see plenty there that needs to be done.

– Alan Turing, 1950.

University of Alberta

Passing a Hide and Seek Turing Test

by

Andrew Cenkner

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Spring 2012
Edmonton, Alberta

Abstract

Hiding and seeking are cognitive abilities frequently demonstrated by humans in both real life and video games. To test to which extent this ability can be replicated by Artificial Intelligence, we introduce a specialized version of the Turing test for hiding and seeking. We then develop an agent that passes the test by appearing indistinguishable from human behavior to a panel of human judges. We analyze the artificial intelligence techniques that enable the agent to capture human hide and seek behavior and their relative contribution to the agent's performance.

Acknowledgements

We would like to acknowledge the following people for assisting in the creation of this thesis: National Science and Engineering Council, for providing funding; Vadim Bulitko for initially suggesting a modified Turing test, constant guidance, and providing a working lab; Valve for developing software and making it available for academic researchers; The IRCL group (Vadim Bulitko, David Thue, Daniel Huntly, and Gregory Lee) for support and revision suggestions; and Marcia Spetch's psychology group (Marcia Spetch, Eric Legge, Craig Anderson, and Matthew Brown) for providing access to research participants, a room to conduct the research in, data from previous experiments as well as revision suggestions from a psychology point of view.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Hiding and seeking are cognitive abilities of humans studied in psychology (Talbot et al., 2009). Many video games invoke this ability in some form. For instance, competitive on-line first-person shooters such as *Counter-strike: Source* (Valve Software, 2008) have players searching for members of the opposing team (e.g., snipers). Role-playing games such as *Borderlands* (Gearbox Software, 2009) or *Fallout: New Vegas* (Bethesda Softworks, 2011) encourage the player to explore the environment and reward them with weapons, side-quests and information on the story and environment.

To support these hide and seek activities, game developers face several challenges. First, level designers need to place desirable items ("loot") in locations that would reward both casual and hardcore players. Deciding on which kinds of items to place at which locations can be made easier and more efficient by predicting, at the game development stage, where the players will search and how their search patterns will be different depending on the player type (e.g., from a casual player to a completionist).

Second, game designers can also enhance the intelligence of their computer opponents with knowledge of where the players will be looking for other players (e.g., in *Counter-strike: Source*) or other player's units (e.g., in *StarCraft 2* (Blizzard Entertainment, 2010)).

Finally, artificial intelligence (AI) developers need to develop non-playable characters that search for the player in a compelling way. A common approach is to give such characters a perfect knowledge of the player's position and then add hard-

coded behavior in an attempt to obfuscate such omniscience. The obfuscation is labor-intensive as it requires extensive trial-and-error and may be fragile insomuch as every once in a while the characters demonstrate their omniscient knowledge of player's position. This is viewed as "cheating" in video games and is despised by the players.

Beyond video games, understanding hiding and seeking is valuable to law-enforcement agencies (e.g., predicting hiding spots for illegal substances) and the military (e.g., predicting locations of stashes of weapons, improvised explosive devices, etc.) Automated searching is especially useful in situations where human interaction is limited (e.g., planetary or disaster-response rovers). Finally, if hiding and seeking are indeed fundamental cognitive abilities of humans then understanding them via a computer program/model may bring us closer to building strong Artificial Intelligence.

The rest of the thesis is organized as follows. We formalize the problem and describe our performance measures in the next section. In Section 3 we review the existing related work and argue that it is insufficient to solve the problem at hand. Our own approach is presented in Section 4, followed by an empirical evaluation. We then discuss the results (Section 6), consider directions for future work (Section 7) and conclude the thesis.

This thesis extends our conference publication (Cenkner et al., 2011) by offering substantially more details on the approach, an extensive walkthrough example, two new agent designs and extended empirical results.

Chapter 2

Problem Formulation

In this thesis we consider the problem of building an AI agent capable of exhibiting human-like hide and seek behavior in a novel environment. We will consider this problem to be solved when an agent passes the following specialized version of Turing test (Turing, 1952). An agent must hide and seek objects in a realistic environment in such a way that human judges are unable to reliably distinguish the agent from humans hiding/seeking in the same environment. Note that the test environment is novel to the agent insomuch as the agent is not given any samples of human behavior in the test environment. An agent must be able to perform on novel environments in order to be portable.

2.1 Telemetry

We formalize the test as follows. Two different environments are prepared (an example of such an environment is shown in Figure 2.1). Each environment has a finite set $L = \{l_1, \ldots, l_n\}$ of n discrete bins where objects can be hidden. The term bin denotes an arbitrary location that participants can interact with to hide or seek. The bins do not need to be containers. Some examples of bins could be a desk drawer, a windowsill, a floor tile or a discolored area of paint on a wall. The human participants are tasked with repeatedly hiding and seeking objects in both environments. They do so by moving about the environment and occasionally hiding an object at one of the bins or checking a bin for a hidden object.

Each time a participant completes a hide or seek task, their path, P, and

Figure 2.1: A office-like environment with hide and seek bins shown as black tiles on the floor.

selection history, S, are recorded. $P = \{P_1, \ldots, P_m\}$ is a list of the participant's Cartesian coordinates and orientations at different points in time. Similarly, $S = \{S_1, \ldots, S_p\}$ is a list of the participant's bin selections and the corresponding times. We define the following variables. $P_i = (x_i, y_i, \phi_i, t_{Pi})$ where $1 \leq i \leq m$; $x_i, y_i, t_{Pi} \in \mathbb{R}$; and $\phi_i \in [0, 360°)$. $S_j = (l_{Sj}, t_{Sj})$ where $1 \leq j \leq p$; $l_{Sj} \in L$; and $t_{Sj} \in \mathbb{R}$.

To illustrate, Figure 2.2 shows a hypothetical sample of one participant completing the seek task in a simple room, with $m = 11$ and $p = 6$. The filled circles and arrows indicate participant's locations and orientations, at the moments in time their position was recorded. The room entrance is at the bottom, where the path starts and ends. The participant's final path, $P = (P_1, P_2, \ldots, P_{11})$, and selection history, $S = (S_1, S_2, \ldots, S_6)$, are shown on the right.

2.2 The Test

Such data recorded for all subjects define human *hide/seek behavior*. Hide/seek behavior from one of the two environments (called the *training environment*) constitutes the *training data* and is made available to the AI agent. The AI agent is

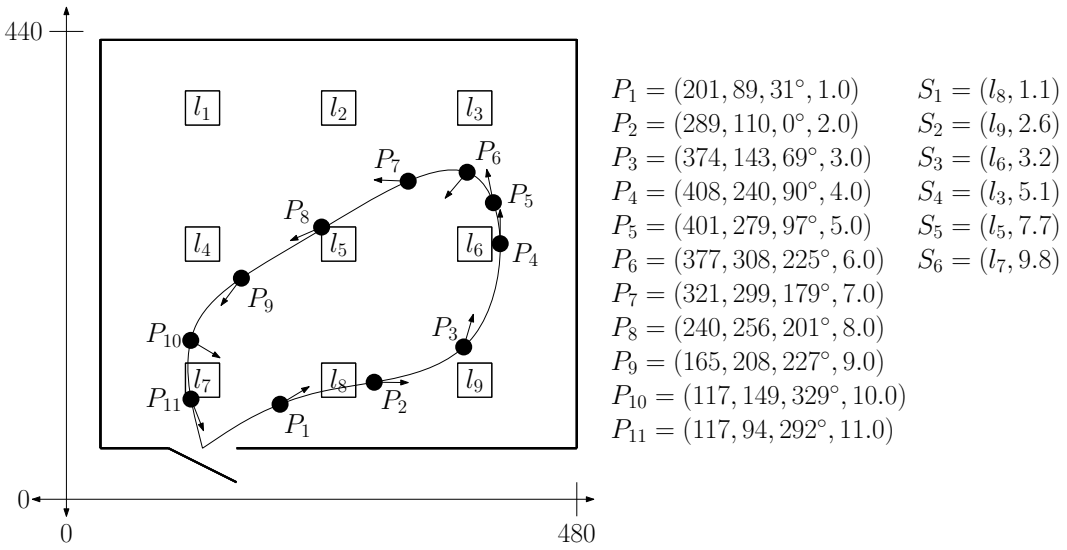


Figure 2.2: A top-down view of an environment with nine possible bins shown as squares. The subject performed the seek task and their path is shown. The path is sampled at 11 moments of time $\{P_1, \ldots, P_{11}\}$. The bins the subject selected for seeking are recorded as $\{S_1, \ldots, S_6\}$.

then asked to hide and seek objects in the other environment (called the *test environment*) and its behavior (i.e., path and selection histories) is recorded. A panel of human judges is then presented with a playback of hide/seek behavior of humans and the AI agent in the test environment. Each panelist is asked to label each behavior sample as either 'human' or 'computer'. The entire process is then repeated with the two environments swapped in their training/test roles. The AI agent is said to have passed the hide/seek Turing test if the human judges were unable to detect the agent statistically better than by random guessing. Note that human judges are trained at this task by being exposed to human hide/seek behavior prior to the test.

We argue that the specialized version of the Turing test is cognitively rich. Research in psychology has demonstrated that hiding and seeking appear to involve skills in orientation, navigation and the theory of mind (Talbot et al., 2009). These skills are non-trivial and humans hone these skills as they mature (Moffat and Resnick, 2002). Other animals vary greatly in their performance in hiding and seeking with more evolved species exhibiting better performance (Emery and Clayton, 2004). The complexity of hiding and seeking is also supported by our experiments in this thesis, where simpler yet non-trivial versions of our AI agent failed to capture

human behavior well enough to pass the test.

We will also briefly investigate whether human judges become better with practice and whether they build their own model of human hide/seek behavior.

Chapter 3

Related Work

3.1 Related Work in Psychology

Psychologists have researched animal hiding and seeking behavior, especially with respect to food caching (Clayton et al., 2007, 2006; Dally et al., 2006). There have also been studies on hiding and seeking behavior in children (Cornell and Heth, 1986; Cornell et al., 1987). Unlike our Turing test, none of these studies consider the hiding/seeking behavior of human adults.

A more recent study (Talbot et al., 2009) considered hiding and seeking behavior of adult humans in a simple virtual environment. The observed behavior was analyzed only at an aggregate level (e.g., mean distance traveled from the room entrance to the first hiding bin). No generative computational model that can be used for an AI agent mimicking the hide/seek behavior was proposed.

3.2 Related Work in Computing Science

The Turing test, previously called the imitation game, has been around since 1950 (Turing, 1950). Livingstone (2006) provides a good introduction to the Turing test and believability in AI. Since the creation of the Turing test there have been two popular public competitions calling for AI agents (bots) attempting to pass the test. The first, known as the Loebner Prize (Loebner, 2011), represents the classic teletype version of the Turing test. Livingstone's (2006) analysis that, "To date, no program entered for the Loebner prize has managed to fool the judges into thinking that it is human - and the prospects of such happening in the near future are arguably

remote" is still valid (Loebner, 2011).

This sentiment of difficulty and a divide and conquer approach has led to many restricted or altered versions of the classic Turing test. The second popular public Turing competition, the BotPrize (2K Games, 2011), replaces the teletype environment with a competitive FPS environment. The competing AI are no longer required to communicate, but must successfully move and combat like humans. Hirono and Thawonmas (2009) and Wang et al. (2011) placed 2nd and 3rd in the 2008 BotPrize competition. To appear human Hirono and Thawonmas' (2009) design broke human behavior into five factors and hand coded each one. Conversely Wang et al.'s (2011) design used reinforcement learning with a neural net on sample human data. Although these designs received a high place in the competition's rankings, to date none of the submissions to the BotPrize have passed the contest's conditions to be labeled human like. Tastan and Sukthankar (2011) used Inverse Reinforcement Learning to train a bot at game time with a database of previous human activity. Their study demonstrated their bot appeared, with statistical significance, more human in the areas of attacking, movement and overall performance, than the default bot shipped with the game. Laird and Duchi (2001) measured human responses to the perceived skill and humanness of their bot and stated "The preliminary nature of this study does not let us draw any hard and fast conclusions". An interesting addition to the BotPrize suggested by Hingston (2010) is to add a game mechanic, essentially a special gun, that rewards players for correctly identifying bots while the game is playing. This would enable bots to learn in real time with feedback regarding their actions. In summary, creating an agent to perform the tasks of movement and combat in a first person environment is not a new or easily solved problem. We simplified the task by requiring our agent to make human-like movements in a first person environment, with the judges viewing only from a top down perspective, and there is no interaction with other agents.

3.3 Related work in Video Games

Historically in first-person shooters, the AI is created by developers selecting behaviors believed to be important to appear human and hard-coding them into the game. For example, in Counter-Strike the bots were programmed to specifically mimic the slow reaction time and attention prioritization displayed by humans (Booth, 2004). Lidén (2002) used visibility maps and Boolean logic to make first-person-shooter bots select positions of cover more intelligently.

There has been work in the area of predicting possible player locations in first-person shooter (FPS) games. Bererton (2004) proposed a model using particle filters to predict human locations. The idea of the particle filter is to give the AI agent an opponent prediction model, removing the dependence on cheating with omniscient vision. The particle filter approximates opponent locations by simulating possible opponent paths from the last known location. Each particle is is a potential path, and is updated with Gaussian noise to represent possible human movement. Hladky and Bulitko (2008) and Darken and Anderegg (2008) have improved on the particle filter method by adding hidden Markov models and simularca respectively. Darken's simularca are similar to the particles in a particle filter, except there are fewer of them, and each has a simple intelligence to replace the Gaussian noise. These AI methods are promising, but do not demonstrate how to actually create a believable agent. The papers do not present any way to collapse their predictions into a single believable path for a player. Any individual particle or simularca only performs a naive attempt at acting human. Also their methods are domain-specific and hard-coded to game specifics (items and environments). Therefore they do not model hide/seek behavior in a portable, environment-independent fashion. Our thesis demonstrates how to create believable, Turing test verified, paths, and deals with a more general case of hiding and seeking arbitrary objects in a novel environment.

Similar to the human prediction models for FPS games, there has been some work in creating human prediction models for real-time strategy (RTS) games. Southey et al. (2007) and Butler and Demiris (2007), using an assumption that units only make small deviations from optimal paths, independently demonstrate how to

predict multiple unit's paths given limited observations. This could be useful when seeking enemy units as one could extrapolate the units location on the predicted path. Although these approaches are helpful to an RTS bot creator, the optimal path assumption results in not modeling how humans hide or seek. Weber et al. (2011) gives an approach to track units after loosing sight using a particle model. The model shows some promising predictions, but like the FPS particle counterparts, provides no way to collapse the predictions into single believable human actions.

Chapter 4

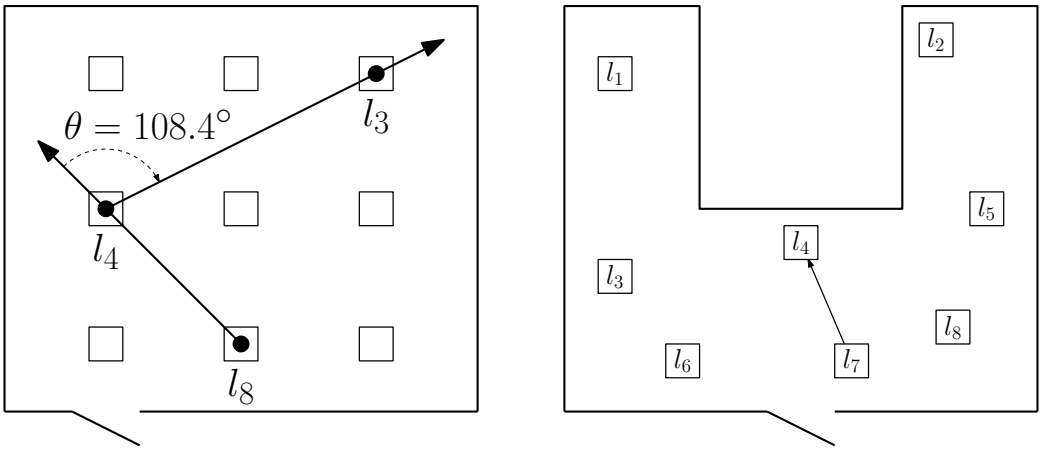
Proposed Approach

In order to pass the specialized Turing test described in Chapter 2, we needed a generative model of human hide and seek behavior. The model was to be trained on human behavior recorded in one environment and then applied to generate new behavior in another environment.

We added several additional requirements as follows. First, the model was to generate varied hide/seek behavior over multiple runs. Second, the model was to be derived automatically from the training data. Third, the model was to operate on a novel environment without any annotation of it beyond a navigation graph.

Given that our environments have a finite number of bins, we decomposed hide/seek behavior into two sub-behaviors, selecting bins and navigating among them. The decomposition made the design more modular as well as gave us an insight on the relative contributions of each sub-behavior.

For the movement sub-behavior we created a simple (M1) and an advanced (M2) strategy. Similarly, we created a simple (L1) and an advanced (L2) strategy for the bin selection sub-behavior. In order to explain each strategy in detail we have a small example running throughout this section. The training and testing environments for our small example are displayed in Figures 4.1(a) and 4.1(b) respectively. In our example an agent starts at the door in the testing room. Figure 4.1(b) displays the room with the 8 squares representing floor tiles and their respective tile ID numbers. The floor tiles are acting as the bins in our example. The agent entered the room, selected tile l_7, then tile l_4, and is now looking for a new tile selection. This example will be demonstrating the seek task.



(a) Example of calculating $G(l_8, l_4, l_3)$ (b) Tile IDs for testing room

Figure 4.1: Training room and testing room.

4.1 Bin Selection Strategies

Each selection strategy was given an ordered list of bins selected so far, and asked to make a new selection. Given this list, the strategies assigned probabilities to each bin in the room. The selections were made with replacement for the seek task and without replacement for the hide task to reflect bin revisitation practices of human subjects.

Let the functions P_{L1} and P_{L2}, represent the probability distribution functions (PDFs) for strategies L1 and L2 respectively. Every bin receives a probability between 0 and 1, and the sum of all the bin's probabilities is 1.

4.1.1 Strategy L1: Uniform random selection

L1 assigns uniform probability to all bins: $\forall i\ [P_{\text{L1}}(l_i) = 1/n]$. For example, for the room shown in Figure 4.1(b), $\forall i\ [P_{\text{L1}}(l_i) = 1/8]$.

4.1.2 Strategy L2: Data-driven bin selection

First we create three probability distributions, P_{L2D}, P_{L2A}, and P_{L2R}, from the training data (i.e., human bin selections recorded in the training environment). P_{L2D} is based on the distance between consecutively selected bins, P_{L2A} is based on the rotation angle between consecutively selected bins, and P_{L2A} is based on the last time

the bin was selected.

Online, L2 uses these three probability distributions each time it is asked to select the next bin. Each bin receives three probability values, one for each distribution. A product of these three distributions comprises L2's final distribution: $P_{\text{L2}} = P_{\text{L2D}} \cdot P_{\text{L2A}} \cdot P_{\text{L2R}}$. In the following we detail each of the three factors.

Spatial Distance - P_{L2D}: The PDF for spatial distance between consecutively selected bins is computed as follows. First, we rank all possible distances between bin pairs in the training room. Then we assign a weight to each distance as the proportion of times that distance occurred between consecutive human selections over the number of times the distance occurs between all bin pairs in the room.

Formally, we first build a set of all possible unique bin pair distances: $D = \{E(i,j) \mid 1 \leq i,j \leq n\}$ where $E(i,j)$ is the Euclidean distance between bins l_i and l_j. We sort the set D in an ascending order and build an index function $I_D : D \to \{1,\ldots,|D|\}$ such that for any possible distance d between two bins, $I_D(d)$ gives d's index in the sorted set D. Clearly, $I_D(\max D) = |D|$ (the maximum distance index) and $I_D(0) = 1$ (the minimum distance index). Therefore $I_D(E(i,j))$ represents the rank of the distance between bin i and j among all distances between bin pairs.

For example, in the room in Figure 2.2 there are 6 unique distances between all pairs of bins. We have $I_D(E(1,1)) = I_D(0) = 1$ (the shortest distance is zero), $I_D(E(5,6)) = 2$ (the distance between bins 5 and 6 is tied for the second shortest distance with many other bin pairs) and $I_D(E(1,9)) = 6$ (the distance between bins 1 and 9 is tied for the largest distance). Figures 4.2(a) and 4.2(b) show a complete definition of $E(*)$ and $I_D(E(*))$ for the training room.

Next we construct the PDF itself $P_{L2D} : \{1,\ldots,|D|\} \to \mathbb{R}$. To build the distribution, we first initialize the PDF to zero: $P_{\text{L2D}}(i) = 0$ where $1 \leq i \leq |D|$. Next we consider the selection history $S = \{(l_{S1}, t_{S1}), \ldots, (l_{Si}, t_{Si}), \ldots (l_{Sm}, t_{Sm})\}$ for each participant in the training data. For each $1 \leq i \leq m-1$, we compute the Euclidean distance $d = E(l_{Si}, l_{S(i+1)})$ between the bin selected at time t_{Si} and the next bin selected at time $t_{S(i+1)}$. We then increase the frequency of the corresponding index $I_D(d)$ in the PDF. The increase is scaled by the number of times that distance occurs in the room. Formally, if X represents the number of bin pair distances in

	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9
l_1	0	1	2	1	1.41	2.24	2	2.24	2.83
l_2	1	0	1	1.41	1	1.41	2.24	2	2.24
l_3	2	1	0	2.24	1.41	1	2.83	2.24	2
l_4	1	1.41	2.24	0	1	2	1	1.41	2.24
l_5	1.41	1	1.41	1	0	1	1.41	1	1.41
l_6	2.24	1.41	1	2	1	0	2.24	1.41	1
l_7	2	2.24	2.83	1	1.41	2.24	0	1	2
l_8	2.24	2	2.24	1.41	1	1.41	1	0	1
l_9	2.83	2.24	2	2.24	1.41	1	2	1	0

(a) Bin pair distances

	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9
l_1	1	2	4	2	3	5	4	5	6
l_2	2	1	2	3	2	3	5	4	5
l_3	4	2	1	5	3	2	6	5	4
l_4	2	3	5	1	2	4	2	3	5
l_5	3	2	3	2	1	2	3	2	3
l_6	5	3	2	4	2	1	5	3	2
l_7	4	5	6	2	3	5	1	2	4
l_8	5	4	5	3	2	3	2	1	2
l_9	6	5	4	5	3	2	4	2	1

(b) Distance rankings

Figure 4.2: A demonstration of calculating distance ranks.

the room with the same rank as d:

$$X = |\{(l_a, l_b) \mid E(l_a, l_b) = d, 1 \leq a, b \leq n\}| \tag{4.1}$$

$$P_{\text{L2D}}(I_D(d)) \leftarrow P_{\text{L2D}}(I_D(d)) + \frac{1}{X} \tag{4.2}$$

The reason we scale the increase is so that the PDF will generalize better to new rooms. For example in our training room (Figure 2.2) $E(l_1, l_2)$ ties for the second shortest distance with 23 other bin pairs out of 81 possible bin pairs. Alternatively in the testing room only 2 bin pairs tie for the second lowest ranked distance out of 64 possible bin pairs. A non scaled increase would disproportionately favor ranks based only on room geometry. For example, we would expect a participant strategy of uniform bin selection in our training room to map to a strategy of uniform selection in our test room. If we do not scale the updates, then the final PDF from out training room will not be uniform across distance ranks, and will not produce uniform behavior in a testing room that has a different rank distribution. In our example these frequency updates turn the human training data (in Figure 4.3(a)) into the second row of Table 4.1.

Once all training data is processed, we normalize the frequencies so they sum

to 1 resulting in the discrete point density function over bin pair distance ranks.

$$P_{\text{L2D}}(i) \leftarrow \frac{P_{\text{L2D}}(i)}{\sum_{j=1}^{|D|} P_{\text{L2D}}(j)}, 1 \leq i \leq |D| \tag{4.3}$$

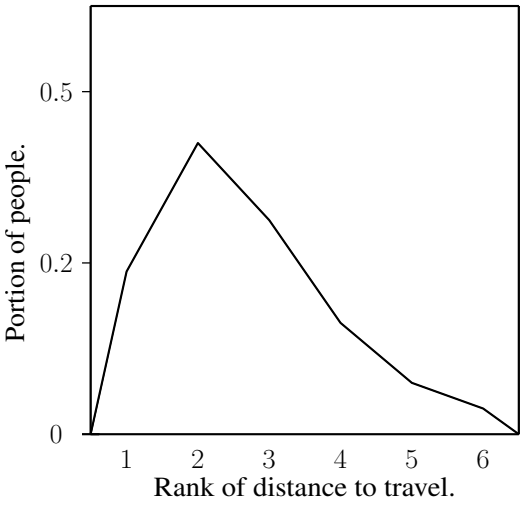
Rank : $1 \ldots \lvert D \rvert$	1	2	3	4	5	6
Number of times rank is selected	$\frac{528}{9}$	$\frac{2512}{24}$	$\frac{1227}{16}$	$\frac{492}{12}$	$\frac{305}{16}$	$\frac{33}{4}$
$P_D(i)$	0.190	0.339	0.249	0.133	0.062	0.027

Table 4.1: Summing, scaling and normalizing frequencies for each distance rank.

The normalized frequencies are displayed in the third row of Table 4.1. A graph of the PDF is displayed in Figure 4.3(b).

	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9
l_1	60	101	40	98	80	22	45	27	6
l_2	109	52	90	82	111	76	13	39	21
l_3	35	110	67	19	73	101	11	23	32
l_4	93	89	14	43	82	41	123	72	26
l_5	69	125	62	76	48	112	68	109	73
l_6	25	77	99	40	107	72	15	76	100
l_7	38	34	7	92	83	12	76	134	76
l_8	18	43	9	80	88	81	89	62	121
l_9	9	19	35	8	86	120	28	122	48

(a) History of human bin selections



(b) P_{L2D} : Spatial distance

Figure 4.3: The human training data and the resulting PDF.

Rotation Angle - P_{L2A}: The rotation angle for a bin is calculated as the angle rotation needed to face that bin after traveling from the second most recent bin to the most recent bin. With the angle values defined in this way we construct P_{L2A} in the same way as we constructed P_{L2D}.

To compute P_{L2A} we start by considering all triplets of bins. For any three bins, l_a, l_b, and l_c, we define $G(l_a, l_b, l_c)$ as the angle between $\overrightarrow{l_a l_b}$ and $\overrightarrow{l_b l_c}$.[1] This angle is shown in Figure 4.1(a). We form the set of all unique angles $\Theta = \{G(l_a, l_b, l_c) \mid l_a \neq$

[1]We ignore all cases where l_a, l_b, l_c line up in a way that makes the angle $G(l_a, l_b, l_c)$ undefined (e.g., $l_b = l_c$).

$l_b \neq l_c, 1 \leq a, b, c \leq n\}$, sort the set in an ascending order and build an index function $I_A : \Theta \rightarrow \{1, \ldots, |\Theta|\}$ such that for any angle $\theta \in \Theta$, $I_A(\theta)$ gives its index in the sorted set Θ. For clarity we note that this angle $\theta \in [-180, 180°)$ is not the same as $\phi \in [0, 360°)$ defined in the problem specification. θ calculated from $G(l_a, l_b, l_c)$ can be visualized as the difference between ϕ when the participant is standing on l_a looking at l_b and ϕ when the participant is standing on l_b looking at l_c. We decided to use θ instead of ϕ because in general the participants do not look at their destination.

We then process the training data. Once again, we consider each subject's selection history $S = \{(l_{S1}, t_{S1}), \ldots, (l_{Si}, t_{Si}), \ldots (l_{Sm}, t_{Sm})\}$. Then for each bin triplet, $(l_{S(i-1)}, l_{Si}, l_{S(i+1)})$, with $l_{S(i-1)} \neq l_{Si} \neq l_{S(i+1)}$ and $2 \leq i \leq m-1$ we compute the angle $\theta = G(l_{S(i-1)}, l_{Si}, l_{S(i+1)})$ and update the frequency of the corresponding index $I_A(\theta)$ in the PDF. The increase is scaled by the number of times that angle occurs in the room. Formally, if X represents the number of angles in the room with the same rank as θ:

$$X = |\{(l_a, l_b, l_c) \mid I_A(G(l_a, l_b, l_c)) = I_A(\theta), 1 \leq a, b, c \leq n\}| \tag{4.4}$$

$$P_{\text{L2A}}(I_A(\theta)) \leftarrow P_{\text{L2A}}(I_A(\theta)) + \frac{1}{X} \tag{4.5}$$

Finally we normalize the frequency for each rank to get the discrete point density function over the angle ranks:

$$P_{\text{L2A}}(i) \leftarrow \frac{P_{\text{L2A}}(i)}{\sum_{j=1}^{|\Theta|} P_{\text{L2A}}(j)}, 1 \leq i \leq |\Theta|. \tag{4.6}$$

Selection recency - P_{L2R}: The selection recency PDF gives the probability that bin i will be selected at time t given its recency number R_i. The recency number is defined as the number of (other) bin selections made by a subject before t and after the last time the bin i was previously selected. For instance, if the participant selects bin i at time t_1 and no further bin selections until time t_2 then $R(i, t_2) = 1$. The first time t any bin i is selected, its $R(i, t) = \infty$.

To illustrate, suppose a user's bin selection history is $S = \{(l_8, 1.1), (l_9, 2.6), (l_6, 3.2), (l_3, 5.1), (l_5, 7.7), (l_7, 9.8)\}$ (given in Figure 2.2). Then, we have $R(l_6, 6) = 2, R(l_1, 6) = \infty, R(l_3, 6) = 1, R(l_3, 3) = \infty, R(l_7, 9.8) = \infty$.

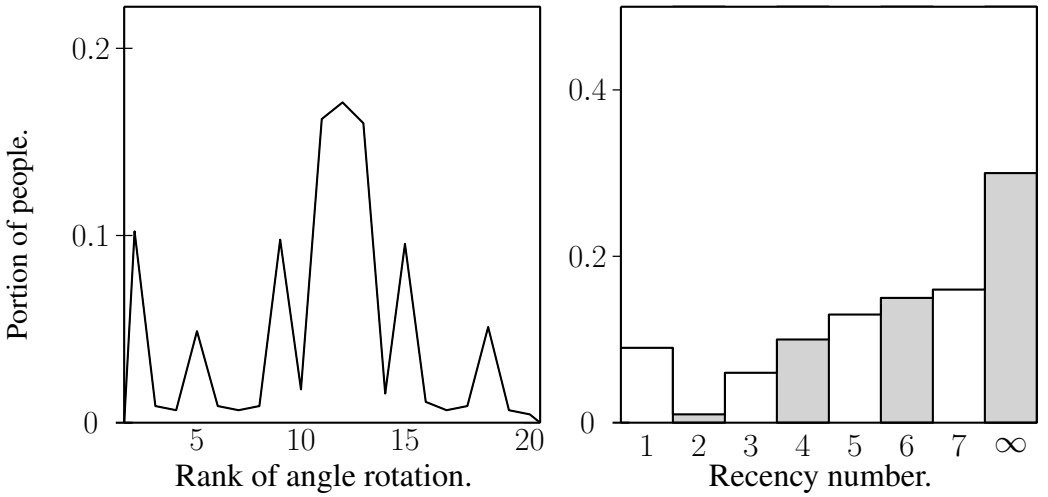
The corresponding PDF is denoted by P_{L2R}. Once again to compute the PDF, we consider each subject's selection history $S = \{(l_{S1}, t_{S1}), \ldots, (l_{Si}, t_{Si}), \ldots (l_{Sm}, t_{Sm})\}$. Then for each bin selection, (l_{Si}, t_{Si}) we calculate its recency value $r = R(l_{Si}, t_{Si})$, and update the frequency of the corresponding recency value r in the PDF via the following update rule:

$$P_{\text{L2R}}(r) \leftarrow P_{\text{L2R}}(r) + 1 \tag{4.7}$$

Let $\mathbf{R}$ denote the set of all recency values observed in the human training data. We normalize the observed frequencies P_{L2R} to compute the probability density function $P_{\text{L2R}}(i)$ defined for $i \in \mathbf{R}$:

$$P_{\text{L2R}}(i) \leftarrow \frac{P_{\text{L2R}}(i)}{\sum_{j \in \mathbf{R}} P_{\text{L2R}}(j)}. \tag{4.8}$$

Examples of the angle and recency probability density functions are found in Figure 4.4.



(a) P_{L2A} : Rotation angles (b) P_{L2R} : Selection recency

Figure 4.4: The probability density functions computed for the room in Figure 2.2.

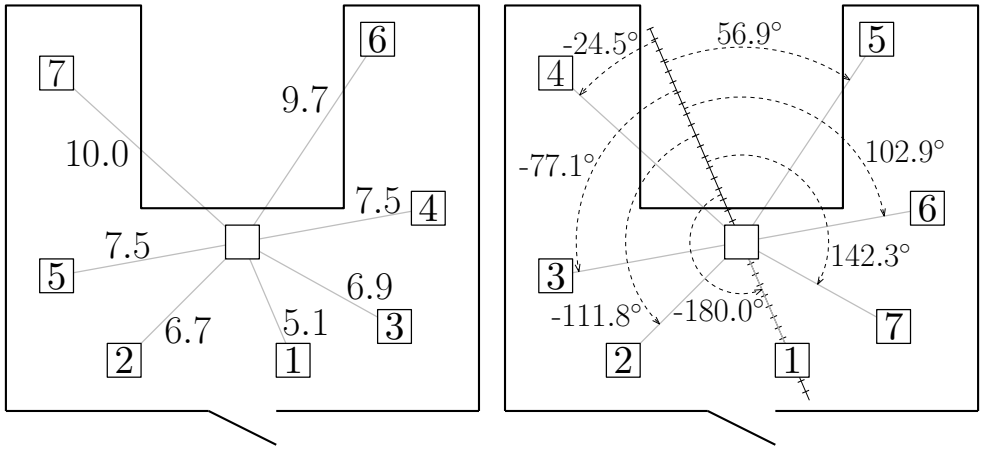
Porting L2's final distribution to a novel environment

The three probability density functions, $P_{\text{L2D}}, P_{\text{L2A}}, P_{\text{L2R}}$, are multiplied together to produce L2's distribution over bins. More precisely if we just selected l_c and then l_b and are considering selecting l_a at time t:

$$P_{L2}(l_a, l_b, l_c) = P_{L2D}(I_D(E(l_a, l_b))) \cdot P_{L2A}(I_A(G(l_a, l_b, l_c))) \cdot P_{L2R}(R(l_a, t)) \tag{4.9}$$

Now that we have explained how to create the PDFs offline we will explain how they are used online to create the final probabilities.

Before we use any of the PDFs we need to rank the tiles in their respective orders for each feature. We calculate all distance pairs and angle triplets in the same way we did for the training room. Next we calculate the distance and what would be the resulting angle to each candidate tile if it were to be selected. We can see the calculations for the distance and angle features in Figure 4.5. The ranking for selection recency is trivial as there have been only 2 tile selections. Tile l_7 is ranked 1, L_4 is ranked 2 and the rest of the tiles tie at a rank of ∞. Due to the non-gridlike layout of the test room we see many more unique values for the feature ranks (31 for distance and 310 for angle triplets).

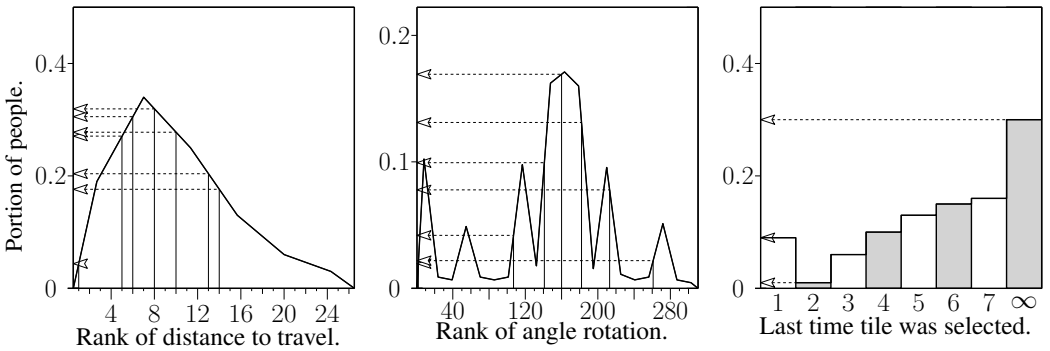


(a) Distance calculations (b) Angle calculations

Figure 4.5: A demonstration of how the bins are ranked by distance and angle. Numbers outside the square display the calculated value, and numbers inside display the rank corresponding to that value.

Once we have each tile's ranking relative to all possible values for each feature we can use the PDFs to find the probability mass. We will use the same distributions created from the training room to give an example of portability. The probability mass is found by taking the area under the curve. Since we know all the ranks are the same width and that we scale values at the end, we can simply take the probability mass as the height of the PDF. Computing the value of the PDFs for each tile is visualized in Figure 4.6. In the Figure each tile traces a line from it's rank on the x-axis to find the height of the PDF at that point. For example tile

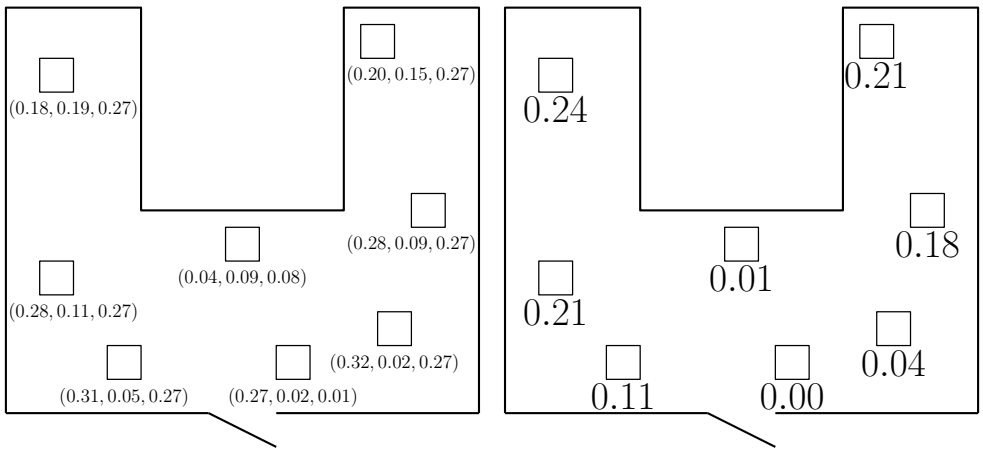
number l_1 (see Figure 4.1(b)) has an angle value of $-24.5°$. Out of the 310 different possible angles in this room $-24.5°$ ranks 160. Therefore in Figure 4.6(b) we can see tile l_1 tracing a line from rank 160 up to the value of 0.19.



(a) P_{L2D} : Spatial distance (b) P_{L2A} : Angle triplet (c) P_{L2R} : Selection recency

Figure 4.6: Applying the ranks of the new room to the previously created PDFs and evaluating.

The resulting probability masses are then associated with each tile, resulting in each tile having three probability mass values, shown in Figure 4.7(a). We can see here from our previous example that tile l_1 represents its correct value of 0.19 ($P_{\text{L2A}}(l_1) = 0.19$).



(a) Partial PDF evaluated for each tile. (P_{L2D},P_{L2A},P_{L2R}) (b) Final weight assigned to each tile

Figure 4.7: Probability breakdown for tile selection.

Finally these probability masses are multiplied together on each tile and scaled to sum to 1 to form the joint probability function (Figure 4.7(b). A tile is stochasti-

cally selected using these weights.

4.2 Movement Strategies

Once a sequence of bins was selected using L1 or L2, a movement strategy was used to navigate between them. For the simple movement strategy (M1) we used cubic spline interpolation between the desired bins. For the advanced movement strategy (M2) we searched a library of human paths and selected one that passes through the chosen bins.

4.2.1 Strategy M1: Spline interpolation

Given a sequence of bins, an A* search was used to convert the sequence of bins into a sequence of guide points adjacent to the bins. Then, a cubic spline interpolation was used to construct a path through the sequence of guide points. The agent's angle, $\phi \in [0, 360°)$ is not splined, and instead is set to always be the current direction the agent is traveling. If a segment between guide points a and b of the resulting path intersected an obstacle in the environment, an A* search was used to construct a valid path between a and b and its middle point was inserted as a new guide point between a and b in the sequence of bins. The spline fitting was then repeated for the new sequence of guide points. The process stopped when the resulting path fit completely inside the environment. This strategy was complete in the sense that A* search is complete, and given enough additional guide points inserted in the path the path would degenerate into A* search. In other words if there was a path between a and b this strategy would find one.

In Figure 4.8 we can see an example of a how M1 creates a path. For the purpose of this example we will assume that the agent started at the door, selected tiles l_7, l_4, l_1, l_6 and returned to the door to leave the room. First the M1 strategy creates an A* path connecting the sequence of desired bins (Figure 4.8). Next M1 replaces each bin in the sequence with a guide point. Each bin's guide point is the first point along the A* path to be within a maximum reachable distance of the bin. The maximum reachable distance is the largest distance the agent can be from a bin and still select

it. The maximum reachable distance radii and new path guide points are displayed in Figure 4.8. The next step is for M1 to fit a spline through the desired guide points and trace along the spline to ensure it falls completely in the room. As shown in Figure 4.8, when we trace the path we intersect a wall (dotted line). M1 remedies this by calculating an A* path (shown with railroad style line) between the sections of the path outside the room, adding the midpoint of that path as a new guide point, and then starting over. We can see that the second spline iteration does not intersect any obstacles and M1 returns the complete path.

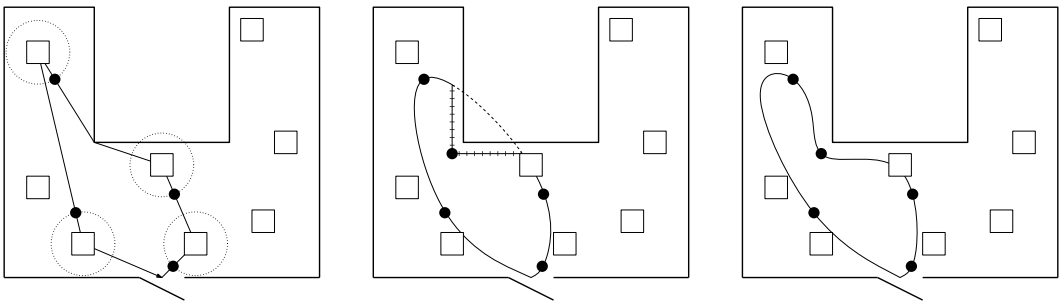


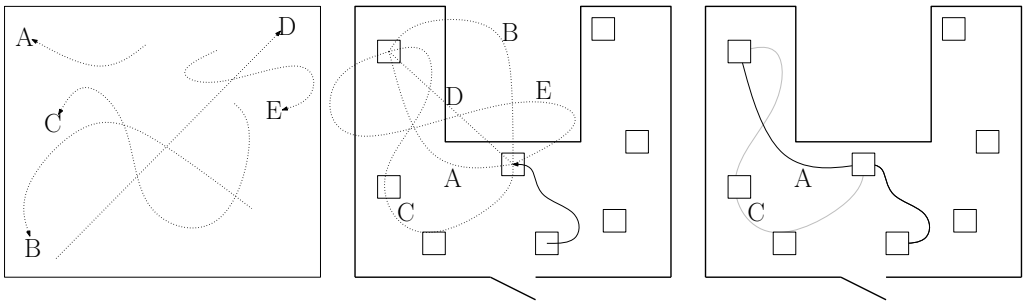
Figure 4.8: Fitting paths from a library to a desired goal.

4.2.2 Strategy M2: Data-driven movement selection

Given a sequence of bins $[l_1, \ldots, l_n]$ to traverse, we considered them sequentially. For each pair of bins (l_i, l_{i+1}), we translated, rotated and linearly scaled every recorded path in the path library so that it connected l_i and l_{i+1}. The quality of each such fit was determined as the product of the scaling quality and the continuity quality. We took the product so that if either quality was completely unacceptable (equal to 0) than the overall quality would be completely unacceptable (equal to 0). Any quality metric can be used for the definition of a path's scaling quality and a path's continuity quality. The particular metrics we used in our experiment are described in Section 5.2. A uniform stochastic selection from the highest quality paths was used for navigation. The reason we did not simply take the best fitting segment was to guard against the possibility of always selecting a few segments that happen to fit a room's geometry well. If an agent only selected from just a few paths it might become predictable to judges. The larger the library of human

movements was the more fluid the segment transitions appeared. If all the segments are discarded M2 falls back on spline interpolation for that segment.

For the purpose of this example M2 will also be planning a path to tile l_1. M2 starts with a large library of paths collected from previous hide and seek tasks performed by humans in a different room. Five of these paths are shown in Figure 4.9(a). Each of the five paths is rotated, translated, and scaled so it starts at tile l_4 and ends at tile l_1. The resulting paths are displayed in Figure 4.9(b). Next, the paths that intersect obstacles are discarded. Finally the paths are ranked on the product of their scaling quality and continuity quality. For example, we can see paths A and C are the only ones staying inside the room. Path A is ranked higher than path C because it is scaled less than C and has little difference in angle from the last path.



(a) Path library (b) Fitting the library paths (c) Final path in bold

Figure 4.9: Fitting paths from a library to a desired goal.

4.3 Agent Structure

The four possible combinations of these strategies: (M1,L1), (M1,L2), (M2,L1) and (M2,L2) — gave us a total of four agents (A1 through A4) shown in Table 4.2. The four agents were of different complexity and had different performance. We will discuss their performance after we present the experimental results.

All of these strategies are portable. L1 and M1 do not have any training data. L2 scales its PDFs to fit the environment given and M2 scales the paths in its library to fit the environment. Therefore agents $1-4$ are portable, and provided with a traversal map and a list of bins can perform in a new environment.

	Uniformly random selection (L1)	**Data-driven selection (L2)**
Spline movement (M1)	Agent 1 (A1)	Agent 2 (A2)
Data-driven movement (M2)	Agent 3 (A3)	Agent 4 (A4)

Table 4.2: Agents A1 through A4.

We have also developed two special agents: A5 and A6. A5 was meant to give away its artificial nature by purposely displaying the most non-human behavior we could generate. A5 was designed to assess judges' performance when the non-human behavior is presented. A6 was designed to behave like A5 at first but then switch to A4 behavior. A6 was meant to check if judges can be tricked into a below-chance-level performance by giving them a fake tale-telling sign.

Chapter 5

Empirical Evaluation

We used the following implementation of the hide and seek tasks in our experiment. Each participant was briefed with a description of the task and trained on how to control a first-person avatar with the *Half-Life 2* (Valve Corporation, 2004) mouse-and-keyboard controls in a small specifically designed training environment built with *Hammer* (Valve Coorporation, 1996). In our environment the selection bins were black tiles on the floor (Figure 2.1). The participants were then put in one of two virtual environments within *Half-Life 2* and asked to perform the hide and seek tasks. In the hide task the participant was asked to hide three objects and to "make your objects difficult for other people to find." In the seek task the participant was asked to select tiles until three previously hidden items were found. The seek task was limited to 1 minute, while the hide task was not time limited. Participants were free to move about the room, but had to wait a delay of 1 second between selecting tiles.

The two environments were: room A, a simple rectangular room, and room B, an office style room modeled after an existing laboratory (Figure 5.1). The former contained 73 designated bins for hiding and seeking. The latter had 75 bins. Both environments had realistic lighting and office furniture.

In our experiment we pursued two objectives to demonstrate that the version of Turing test we used is indeed meaningful. The objectives are (i) showing that the task is cognitively rich enough that a simple AI agent would fail the test and (ii) that the human judges are given enough information to render an informed judgment. An example of a test that violates condition (i) would be "to sit in the chair" since

a statically placed agent model would easily pass the test. An example of violating condition (ii) would be withholding the actual contents of a conversation from the judge in the original Turing test (Turing, 1952) and, instead, showing them only a light when the agent is using their teletype.

We satisfied these conditions by presenting the judges with a video of a top-down view (Figure 5.1) of the agent (human or AI) moving about in the environment and selecting tiles. We demonstrated that a non-trivial, manually designed AI agent was reliably distinguished from humans. This shows that the test is not passed trivially, and that the judges were given enough information to correctly label our non-trivial agent.

Figure 5.1: The two environments used in our study (a top-down view).

Additionally we investigated the following questions raised by our previous work (Cenkner et al., 2011) regarding the nature of Turing tests. Do judges treat each video independently? If judge's choices are not independent, can we exploit this to create an agent that is identified correctly less than 50% of the time? Finally, are judge scores affected by practice?

The study was carried out in three phases: data collection, model/agent training and judging.

5.1 Data Collection

The dataset we used to create our models was the collective recordings of 1071 human participants in virtual environments. Our participants were recruited from a first-year course in psychology.

As the subjects performed their tasks, their avatar's locations and orientations were recorded once per second. Additionally, each tile selections and the time the selection occurred at were recorded. Overall, 5142 paths were recorded and used as training data, each containing between 6 and 125 data points.

5.2 Model/Agent Training

First we will describe the creation of the two bin selection strategies and then the two movement strategies. We implemented the L1 strategy by drawing uniformly random tile ID numbers.

The domain for our experiment, *Half-Life 2*, did not require any specific changes to the domain-independent specification of the L2 strategy outlined in Section 4.2.1. The distance and angle metrics were provided by the *Half-Life 2* engine. The shape of our PDFs generally corresponds to the ones displayed in Figure 4.4. Specifically humans are biased to making selection choices closer to where they already are, with this bias increasing the higher their current selection number is. The PDF for angle rotation had the highest peak at $0°$ and two shorter peaks at $-90°$ and $90°$. This indicated humans are strongly biased to making selections in a line (turning $0°$), and somewhat biased to making right angle turns (turning $-90°$ and $90°$). Finally the more recently a tile was selected the less likely humans are to select it again, except the most recent selection. This double selecting a tile phenomenon can potentially be explained by humans missing the selection confirmation and re-selecting the tile until they see the second selection confirmation. This explanation is suggested because the participants were able to click the tiles as rapidly as they liked, however the clicks would only register as a new selection after a delay of one second, and the confirmation of the new selection only appeared as a minor change in the corner of the screen.

The first attempt at creating the M1 strategy was to fit a cubic spline through all the 2-space coordinates of the given list of tiles. The z coordinate was ignored because the judges were presented with top down views. This proved to be ineffective as the agent ended up standing right on top of each tile before selecting it. In prac-

tice most participants do not walk over the tile they wish to select while selecting it, and almost none stood directly on top of the tile before selecting it. To remedy this problem, instead of making M1 fit a spline through the actual selected tiles, we made it fit a spline through points beside the actual tiles. The points beside the actual tiles were created by running an A* path through the actual tiles and taking the first point along the A* path within a reachable distance of the actual tile as described in Section 4.2.1. The reachable distance is the farthest the avatar can be from an object and still activate it. In our experiment the reachable distance was approximately the height of the avatar (2 meters). In other words, an agent using the M1 strategy, did not quite travel to each tile selection. Instead it came within 2 meters and started heading to the next tile. The cubic spline interpolation was done by parameterizing the desired x and y coordinates with respect to time and performing a one dimensional cubic spline interpolation on each. The agents' yaw was set to the direction it was traveling. In other words the yaw was always tangential to the curve created from the spline fitting.

The paths mentioned in data collection were used to create the library of human movements for the M2 strategy. Each path was divided into the segments between tile selections. When M2 was asked to create a path from a to b it rotated, translated and scaled each of these segments so the endpoints lined up with a and b. Next, a cubic spline interpolation was fit to each segment, and traced from a to b. If a segment intersected an object or wall it was discarded, and the rest were ranked based on how well they fit. For our experiment we defined the scaling quality and continuity quality mentioned in Section 4.2.2 as follows. If we have X as target length over the segment length, the stretching was calculated as the minimum of X and $1/X$. This mapping was motivated by the following results: a segment half the desired length was of the same quality as a segment twice the desired length; segments of length 0 or ∞ map to a quality of 0; and a segment requiring no scaling (length equal to target length) maps to the highest possible quality of 1. The continuity quality is a measure of how much an agent would need to turn at the segments' join point when transitioning from one to the next. Let ϕ_1 be the yaw at the end of the last segment, and ϕ_2 be the yaw at the start of the next segment. For

example if we are considering segment C's quality in Figure 4.9(c) ϕ_1 would be approximately $180°$ (pointing left) and ϕ_2 would be approximately $270°$ (pointing down). The continuity quality is calculated as:

$$\frac{(180° + \phi_1 - \phi_2) \mod 360°}{360°} \tag{5.1}$$

This relation results in a segment requiring no turn at the join point mapping to the highest possible quality of 1, and a segment requiring a $180°$ turn mapping to the lowest possible quality of 0. After all the segments were ranked by the product of the two quality measures, M2 uniformly selected a segment from the top 10 highest quality segments.

These 4 strategies were combined according to Table 4.2 to create agents 1 to 4. A5 tries to give itself away by constantly standing between 2 tiles in the corner selecting both in an alternating fashion until running out of time. While standing between the 2 tiles, A5 spins at a constant rate. We use the performance of A5's non-human behavior as an approximation of how poorly a trivial bot can perform in our test. For each trial of 10 videos, the random video selection averaged 5 agent and 5 human videos. A6 returned videos from agent A5 for the first 2 requested agent videos and then then switched to the agent A4 for the remaining agent videos. Our results in Section 5.4 show agent A5 to be the weakest agent and A4 to be the strongest agent. This resulted in A6 using an average of 2 weak and 3 strong videos in each trail.

Each agent uses its current bin selection strategy to make a list of tiles to select. In the hide task the list contained 3 tiles and in the seek task the list contained enough tiles to ensure that the task timed out before running out of tiles. Next, the list of tiles, created by L1 or L2, was passed to the movement strategy. The movement strategy, M1 or M2, then fit a path from the player starting area, near the room's door, within selecting distance of each sequential tile in the list and back to the door.

5.3 Judging

For this part of the study we ran a group of 288 human participants recruited from a first-year course in psychology. None of these students participated in the previous study collecting the initial hide and seek data. Each participant was briefed with a description of the task and was asked to judge videos of agents in 4 trials: hiding and seeking in each of the two rooms. These participants were divided into 6 groups of 48 participants, one for each agent. These groups were further broken down into two groups of 24, for the two possible room orderings. Each trial consisted of 5 training videos followed by 10 test videos. The judges knew that the training videos were of human behavior. The judges were told that some test videos may be of a computer and some may be of humans with no particular proportion given. In actuality, the proportion was approximately half. The judges were not told that there may be multiple agents. Each video was played at double speed and was between 3 and 44 seconds long. At the end of each test video the judge labeled it as "Human" or "Computer" which queued in the next video. The judges were also able to re-watch the most recent video before labeling it. In total each judge labeled 40 videos, 20 for hide and 20 for seek. Each judge's score out of 20 for hide and 20 for seek was calculated as the number of correct labelings. These scores were tied to the agent the judge was assigned. The judges were not told their scores.

5.4 Study Results

The correct labeling rates in Figures 5.2, 5.3 and 5.4 are the percentage of videos the judges correctly labeled, with the boxes indicating the 95% confidence intervals. The intervals displayed are the Wilson score intervals, which are used when there is a chance of the mean being close to the boundaries (0% or 100%). The closer an agent is to 50% the closer it is to passing the Turing test. To clarify, an agent identified 55% of the time is stronger than one identified 30% of the time. This is because a theoretical judge could obtain a 70% identification rate by inverting every answer. If one confidence bar occurs completely under another bar we can conclude that the first is correctly labeled less often than the second with greater than 95%

confidence. The agent's identification rates have been broken down in 3 ways to compare across task, video type and experience.

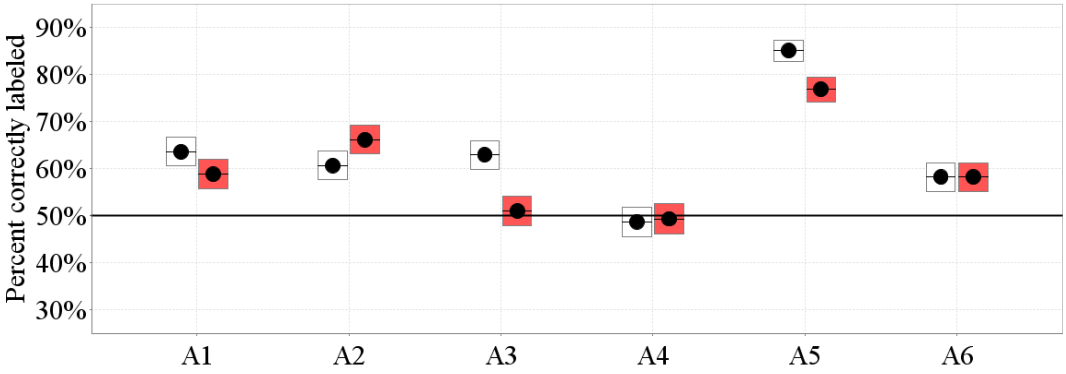


Figure 5.2: 95% significance boxes for scores broken down by task. The seek task is represented with empty boxes and the hide task with filled boxes.

In Figure 5.2 we show agent identification rates in the hide task and seek task. In the hide task (filled boxes), we can see that when comparing the agents with the same bin strategy, the agents implementing the M2 strategy were correctly labeled significantly less than their M1 counterparts (A3's filled box is under A1's filled box and A4's is under A2's). This implies that the M2 strategy made a significant difference in the hide task. It is interesting to note the same is not true when comparing agents with the same movement strategy (A2's empty box is not under A1's and A4's is not under A3's).

In the seek task, A4 was correctly labeled significantly less than A1, A2 and A3. This implies that the M2 and L2 strategies made a significant difference when used together in the seek task.

The results above indicate that A4 is correctly labeled as an AI agent less frequently than A1 with 95% confidence. The following analysis allows us to derive an even better confidence bound. A distribution of the judge scores (with hide and seek mixed together) for A1 and A4 can be found in Figure 6.1. We performed an independent two sample (A1 and A4) Student's t-test, for both hide and seek. These scores out of 20 can be approximated as a normally distributed random variable since it is the sum of individual Bernoulli trials. The number of degrees of freedom was 94 in both the hide and seek tasks (48 participants judged A1 and 48

judged A4). The t-test produced a t value of 3.129 in the hide task and 4.750 in the seek task. Applying these values to a two tailed t-test, we can claim that A4 was correctly labeled less often than A1 with confidence of 99.77% for the hide task and greater than 99.999% for the seek task.

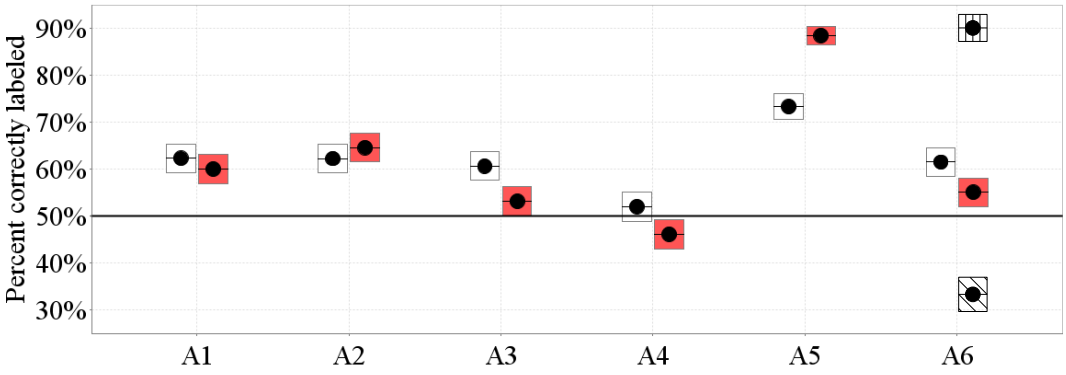


Figure 5.3: 95% significance boxes for scores broken down by video type. The score on human videos is represented with empty boxes and the score on agent videos with filled boxes. The vertically and diagonally striped boxes respectively represent the breakdown of the trivial and advanced videos agent A6 uses.

In Figure 5.3 we show agent identification rates for the human videos and agent videos. One may predict that the identification rate on human videos would be the same across all agents. This would be true if judges treated each video independently (for example had no memory of previous videos). However, we can see a general trend that identification rate of human videos rises when mixed with weaker bots and lowers when mixed with stronger bots. The empty box (human video identification rate) for A4 is below the empty boxes of A1, A2 and A3. Similarly the empty box for A5 is above A1, A2 and A3. A4 and A5 were designed to be the strongest and weakest bots respectively, and this is verified by viewing the filled boxes. We propose two approaches to judging to explain the different identification rates on the same human videos, and use the results of A6 to investigate.

In the first approach the judges build two agent models, one for computer and one for human and then they label a video with the model it most closely fits. If a clear model for an agent is easy to formulate, as it is with A5, then the other videos are deemed to be human videos. Although intuitive this is not sound. Indeed, there may be multiple models of bots and humans, and the judges were not given the

ratio of human videos to agent videos. It appears that the judges are unable to tell the difference between A4 and human videos. This means that if the judges are actually using this approach they should have an identification rate close to 100% on the human videos for A6 and an identification rate close to 0% for the strong A6 videos (diagonally shaded box). We see a different result indicating that even in the presence of clearly computer videos (weak videos are the same as the ones used in A5 achiving a 94% identification rate) judges classify some of the other videos as computers. It is interesting to note that although a two model approach is not perfect it would achieve a higher score than the judges did. The two model approach would score $2/2$ weak agent videos correct, $5/5$ human videos correct and $0/3$ strong agent videos correct totaling a 70% identification rate, which is much higher than the 57.5% identification rate human judges recived on agent A6.

The second approach proposed is that human judges always assume an equal mixture of videos. In other words if a judge knows they have identified 2 agents and have 8 videos left, they will try to label 3 out of the 8 remaining videos as agents. In randomly selecting 3 of the 8 videos to label as agents, and labeling the other 5 as human, judges have an expected value of correctly identifying 1.125 of the 3 agent videos (37.5%) and 3.125 of the 5 human videos (62.5%). The proximity of these scores to the experimental means of 35.3% and 61% resectively, favors the second approach over the first.

It is interesting to note that humans incorrectly assume they can maximize their score by balancing their classifications. For instance, if a judge knows they have labeled 2 agents correctly and are presented with 8 more indiscernible videos of which they know exactly 3 are agents, the optimal strategy is to label all 8 as human, instead of randomly labeling 3 as agent and 5 as human.

In Figure 5.4 we explore the possibility of a learning effect. In the experiment each participant judged 20 trials in the first room, and then 20 trials in the second room. In our experiment half the participants experienced room A (simple rectangle room) before room B (more complicated room), and half experienced room B before room A. The reason we divided the participants into these two groups was to isolate the room and experience variables. If all participants viewed room A first,

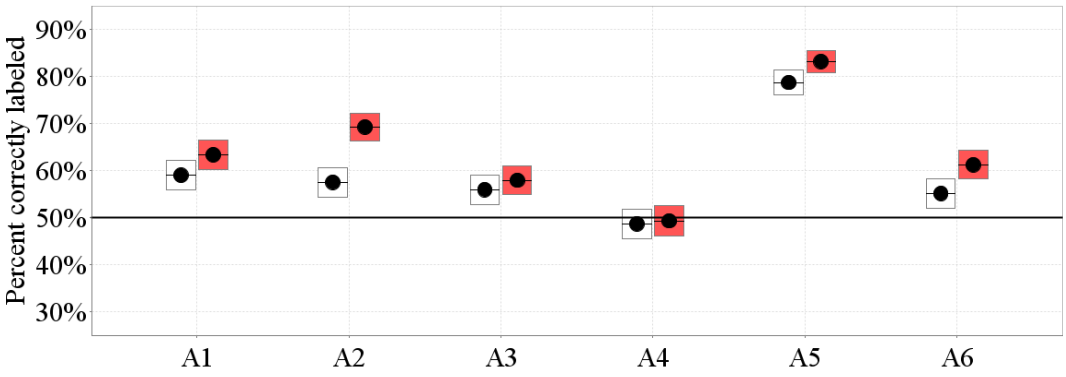


Figure 5.4: 95% significance boxes for scores broken down by experience. The score on the first 20 videos (less experience) is represented with empty boxes and the score on the second 20 videos (more experience) with filled boxes.

we would not be able to attribute a change in judge scores to a specific variable, as we would not know what caused the change, room configuration or experience. It is possible that the score for a group of video labelings is affected by the proportion of room A videos to room B videos. Since this proportion is equal in both the video group receiving less experienced labelings and the video group receiving more experienced labelings, the affect will be equal. Therefore we can attribute a change in judge scores in Figure 5.4 to an experience effect. We can see that there is a general trend of higher identification rates in the second room. This indicates the judges perform better with practice. It is interesting to note that even though the judges did not receive feedback regarding their choices, they were able to improve their scores. We see that, although other agents became identified correctly more often, there is no increase in identification of A4. This means even if judges do improve with practice, A4 is still not reliably discernible from humans.

Chapter 6

Discussion

In the hide task, the lower correct labeling rates of the agents implementing M2 indicates that, in our experiment, the advanced movement strategy made a significant difference. The L2 strategy produced no significant difference in the hide task, but did in the seek task. We attribute this to the limited number of bin selections (3) in the hiding task. The bin selection strategy plays a much smaller part in the hiding task because 3 choices are often not enough to draw an informed guess to the identity of the agent.

Agent A4 performs significantly better than Agent A1 in both hiding and seeking. The mean near 50% in both hiding and seeking indicates agent A4 has passed the Turing test we set out to pass. That is to say, judges do no better than chance with agent A4.

Note it is possible, although unlikely, that in a Turing test a bimodal distribution in judge scores can appear. This would lead to average judge score of 50%, and possible incorrect inference that an agent is indistinguishable. For example let us say all human videos start with one turn clockwise and all computer videos start with one turn counterclockwise, but are otherwise indiscernible. If this is the case, every judge can trivially sort the two types of videos into two groups, clockwise and counter clockwise. However, they will not be sure which group to label human and which to label agent. This may result in half of the judges making the correct guess and labeling 100% of videos correctly and the other half making the wrong guess and labeling 0% correct. This would create a bimodal distribution in which the mean is 50%, but the agent is fully distinguishable from a human. Any judge

who is given this "tell" will be able to correctly label the agent from a human every time.

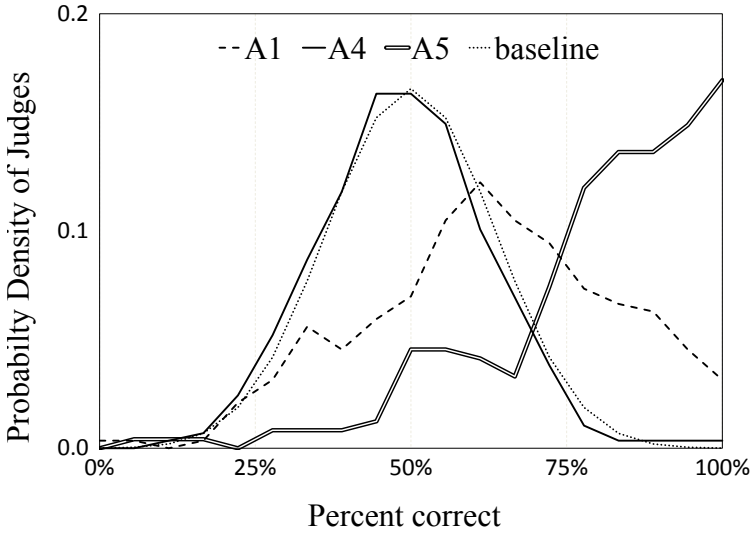


Figure 6.1: Distribution of judge scores.

If all the judges were equal and none could tell the difference between humans and agents, all the scores would be left purely up to chance. This means, for both the hide and seek tasks, some would score higher than 10 out of 20, some lower, but most would be centered around 10. The distribution of these scores out of 20 are displayed in Figure 6.1. The shape of the purely chance distribution is included as a baseline for comparison. The closer a distribution is to the baseline the closer it is to matching pure chance. The data in Figure 6.1 suggests that agent A4 is normally distributed about a mean of 50%, and is close to the baseline. Conversely, the distributions for agents A1 and A5, demonstrate less normal shapes. The probability mass near the 100% side of the distribution for agents A1 and A5 show that there exist judges that can consistently label A1 and A5 correctly. It is the absence of these expert judges in the distribution of agent A4 that suggests that agent A4 does not consistently exhibit any "tells" to human judges.

Chapter 7

Future Work and Conclusion

Hiding and seeking may be a fundamental cognitive ability of humans and animals and has several applications in video game design. This thesis made the following contributions. We proposed a first computational generative model of hide and seek behavior in adult humans. The model is built automatically by data-mining observed human behavior. We implemented a model within an AI agent and demonstrated its validity via a restricted version of the Turing test. Specifically, a series of four AI agents based on the model was constructed and evaluated. The most complex of the four agents appears to have passed the Turing test. Additionally we provided evidence that human judges, while behaving suboptimally, are not easily swayed by injecting the trials with a "tell". Finally we show that there is an increase in judge identification rate over time on the bots that are identified higher than a pure chance rate.

Future work may pursue training our model on data regarding pigeons seeking for food. It will also be of interest to incorporate our model into a combat agent in an on-line game such as *Counter-strike: Source*.

Bibliography

2K Games. The 2k BotPrize : Home, November 2011. URL http://botprize.org/.

Curt Bererton. State estimation for game ai using particle filters. In *AAAI Workshop - Technical Report*, volume WS-04-04, pages 36–40, 2004. URL www.scopus.com. Cited By (since 1996): 2.

Michael Booth. The official Counter-Strike bot. http://www.gdcvault.com/play/1013625/, 2004.

Simon Butler and Yiannis Demiris. Partial observability during predictions of the opponent's movements in an RTS game. In *Computational Intelligence and Games. IEEE Symposium. 2010. (CIG 2010)*, Copenhagen, Denmark, 2007. Institute of Electrical and Electronics Engineers (IEEE). URL http://portal.acm.org/citation.cfm?id=1625275.1625699.

Andrew Cenkner, Vadim Bulitko, and Marcia L. Spetch. A generative computational model for human hide and seek behavior. In *Proceedings of the Seventh Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2011)*, pages 128–133, Stanford, Palo Alto, California, 2011. AAAI Press.

N.S. Clayton, N.J. Emery, and A. Dickinson. The prospective cognition of food caching and recovery by western scrub-jays (aphelocoma californica). *Comparative Cognition & Behavior Reviews*, 1, 2006.

N.S. Clayton, J.M. Dally, and N.J. Emery. Social cognition by food-caching corvids. the western scrub-jay as a natural psychologist. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 262, 2007.

E. H. Cornell and C. D. Heth. The spatial organization of hiding and recovery of objects by children. *Child Development*, 57, 1986.

E. H. Cornell, C. D. Heth, L. S. Broda, and V. Butterfield. Spatial matching in $1\frac{1}{2}$- to $4\frac{1}{2}$-year-old children. *Developmental Psychology*, 23, 1987.

J.M. Dally, N.S. Clayton, and N.J. Emery. The behavior and evolution of cache protection and pilferage. *Animal Behaviour*, 72, 2006.

C.J. Darken and B.G. Anderegg. Particle filters and simulacra for more realistic opponent tracking. In *Game AI Programming Wisdom 4*, pages 419–428. Charles River Media, Inc., Hingham, Massachusetts, 2008.

N.J. Emery and N.S. Clayton. The mentality of crows: Convergent evolution of intelligence in corvids and apes. *Science*, 306, 2004.

P. Hingston. A new design for a turing test for bots. In *2010 IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 345 –350, Aug. 2010. doi: 10.1109/ITW.2010.5593336.

Daichi Hirono and Ruck Thawonmas. Implementation of a human-like bot in a first person shooter: Second place bot at BotPrize 2008. In *Proceedings of Asia Simulation Conference 2009 (JSST 2009)*, Shiga, Japan, 2009. Japan Society for Simulation Technology.

S. Hladky and V. Bulitko. An evaluation of models for predicting opponent locations in first-person shooter video games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 39–46, Perth, Australia, 2008. IEEE.

John E. Laird and John C. Duchi. Creating human-like synthetic characters with multiple skill levels: A case study using the soar quakebot. In *Papers from the 2001 AAAI Spring Symposium on Artificial Intelligence and Computer Games*, pages 54–58, 2001.

Lars Lidén. Strategic and tactical reasoning with waypoints. In *AI game programming wisdom*, pages 211–220. Charles River Media, Inc., Hingham, Massachusetts, 2002.

Daniel Livingstone. Turing's test and believable AI in games. *Computers in Entertainment (CIE) - Theoretical and Practical Computer Applications in Entertainment*, 4, 2006.

Hugh Loebner. Home page of the Loebner prize in artificial intelligence, November 2011. URL `http://www.loebner.net/Prizef/loebner-prize.html`.

Bethesda Softworks. Fallout: New Vegas. `http://fallout.bethsoft.com/`, 2011.

Blizzard Entertainment. Starcraft 2. `http://www.starcraft2.com/`, 2010.

Gearbox Software. Borderlands. `http://www.borderlandsthegame.com/`, 2009.

Valve Coorporation. Hammer. `http://developer.valvesoftware.com/wiki/Valve_Hammer_Editor/`, 1996.

Valve Software. Counter-Strike: Source. `http://store.steampowered.com/app/240/`, 2008.

S.D. Moffat and S.M. Resnick. Effects of age on virtual environment place navigation and allocentric cognitive mapping. *Behavioral Neuroscience*, 116, 2002.

Finnegan Southey, Wesley Loh, and Dana Wilkinson. Inferring complex agent motions from partial trajectory observations. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2631–2637, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. URL `http://portal.acm.org/citation.cfm?id=1625275.1625699`.

Katherine J. Talbot, Eric L.G. Legge, Vadim Bulitko, and Marcia L. Spetch. Hiding and searching strategies of adult humans in a virtual and a real-space room. *Learning and Motivation*, 40, 2009.

Bulent Tastan and Gita Sukthankar. Learning policies for first person shooter games using inverse reinforcement learning. In *Proceedings of the Seventh Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2011)*, Stanford, Palo Alto, California, 2011. AAAI Press.

A.M. Turing. Computing machinery and intelligence. *Mind*, 49, 1950.

A.M. Turing. Can automatic calculating machines be said to think? in Copeland (1999), 1952.

Valve Corporation. Half-Life 2. `http://www.half-life2.com/`, 2004.

Di Wang, Budhitama Subagdja, Ah-Hwee Tan, and Gee-Wah Ng. Creating human-like autonomous players in real-time first person shooter computer games. In *Proceedings of the Twenty-First Conference on Innovative Applications of Artificial Intelligence (IAAI 2009)*, Stanford, Palo Alto, California, 2011. AAAI Press.

Ben G. Weber, Michael Mateas, and Arnav Jhala. A particle model for state estimation in real-time strategy games. In *Proceedings of the Seventh Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2011)*, pages 103–108, Stanford, Palo Alto, California, 2011. AAAI Press, AAAI Press.