Passing a Hide and Seek Third-Person Turing Test

Andrew Cenkner and Vadim Bulitko and Marcia Spetch and Eric Legge and Craig Anderson and Matthew Brown
Departments of Computing Science and Psychology and Psychiatry
University of Alberta
Edmonton, AB, T6G 2E8, CANADA
{ cenkner | bulitko | mspetch | elegge | cganders | mbrown2 } @ ualberta.ca

Abstract—Hiding and seeking are cognitive abilities frequently demonstrated by humans in both real life and video games. To determine to which extent these abilities can be replicated with Artificial Intelligence, we introduce a specialized version of the Turing test for hiding and seeking. We then develop a computer 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 imitate human hide and seek behavior and their relative contribution to the agent’s performance.

Index Terms—Turing test, AI bots, hide and seek behavior

I. INTRODUCTION

Hiding and seeking are considered to be non-trivial human cognitive behaviors that develop with age [1], [2]. These behaviors are present in a variety of video games in some form. For instance, competitive on-line first-person shooters such as Counter-strike: Source [3] have players searching for members of the opposing team (e.g., snipers). Role-playing games such as Borderlands [4] or Fallout: New Vegas [5] encourage the player to explore the environment and reward such exploration with weapons, side quests and information on the story and the 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 depend on the player type (e.g., from a casual player to a completionist).

Second, game designers can enhance behavior of in-game artificial intelligence (AI) controlled agents 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 [6]). This would allow the AI agents to better control the level of stealth they exhibit in the game.

Finally, game 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 obfuscating such omniscience. Naturally looking obfuscation is labor-intensive as it requires extensive trial-and-error iterations 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 can break player’s immersion.

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 weapon stashes, improvised explosive devices, enemy troops or sniper positions). From a theoretical and cognitive perspective, if hiding and seeking are fundamental cognitive abilities of humans then understanding them via a computer program/model may shed light on human cognition and/or bring us closer to building strong Artificial Intelligence.

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

This paper extends our conference publication [7] by offering more details on the approach, an extensive walkthrough example, two new AI agent designs and new empirical results.

II. PROBLEM FORMULATION

The problem we are tackling in this paper is to develop an Artificial Intelligence agent that hides and seeks objects in a virtual environment. The criteria for success will be whether the agent does so in a human-like fashion, replicating peculiarities of human hiding and seeking behavior. How similar the agent’s behavior is to that of humans is to be judged by a panel of humans. Note that the AI agent is tested in a novel environment where it has not previously seen any humans hiding or seeking.

Tests in which humans judge how human-like a computer is are known as Turing tests, after the original test proposed by Alan Turing [8]. In game-like settings such tests are also known as player believability tests since to pass such a test a computer agent controlling the player’s avatar needs to mislead the judges to believe that the player behind the avatar is human [9]. Because of the difficulty of passing the full Turing test, specialized or restricted Turing tests are commonly used. The Loebner competition [10] and the BotPrize [11], [12] use restricted Turing tests to assess player believability of computer programs.
Early Turing tests assumed that the task the agent is being judged on allows the agent to interact with the human judge (e.g., via a conversation in the Loebner competition or in first-person multiplayer shooter settings in the BotPrize). Later Turing tests have been generalized to single-player tasks where the judge is necessarily a passive observer of the agent. Some researchers have even argued that such third-person tests are actually more accurate than first-person tests as the judge is able to separate their judging from their game-play and therefore “is able to concentrate more on the assessment of believability via a higher cognitive focus on the task” [9]. A recent example of such a third-person Turing test is the Turing test track of the international Mario AI Championship [13].

Our specialized version of the Turing test is to have an AI agent exhibit hide and seek behavior in a novel environment that is indistinguishable (by a panel of human judges) from humans operating in the same environment. This hide-and-seek task we are interested in is inherently single-player and thus our Turing test will use third-person assessment of player believability by having the human judges passively observe agent behavior.

A. Telemetry

We formalize our test as follows. Two different environments are prepared. Each environment has a finite set \( L = \{l_1, \ldots, l_n\} \) of \( n \) discrete locations where objects can be hidden or sought. Examples of such locations are a desk drawer, a window sill, a floor tile or a discoloured area of paint on a wall. In our experiments, the locations were floor tiles. The participants (humans and AI agents) are tasked with repeatedly hiding or seeking objects in both environments. They do so by moving about the environment and occasionally hiding an object at one of the locations or checking a location for a hidden object.

We define a route to be a recording of one participant completing either the hide or seek task. The route is recorded as their path, \( P \), and location selection history, \( S \). The path \( P = (P_1, \ldots, P_m) \) is a list of the participant’s Cartesian coordinates and orientations at different points in time. Similarly, the selection history \( S = (S_1, \ldots, S_p) \) is a list of the participant’s location selections and the corresponding times. Mathematically: \( P_i = (x_i, y_i, \phi_i, t_{P_i}) \) where \( 1 \leq i \leq m; \ x_i, y_i, t_{P_i} \in \mathbb{R} \); and \( \phi_i \in [0^\circ, 360^\circ] \); \( S_i = (l_{S_i}, t_i) \) where \( 1 \leq i \leq p; \ l_{S_i} \in L \); and \( t_i \in \mathbb{R} \).

To illustrate, Figure 1 shows a hypothetical sample of one participant completing the seek task in a simple room, with \( m = 11 \) and \( p = 6 \). The participant enters the room at the door on the bottom left. He/she then travels around the room along the path shown in the figure. The participant’s location is recorded every second as \( P_i, 1 \leq i \leq 11 \) and shown in the figure as filled circles. The participant also seeks at the six locations \( S_j, 1 \leq j \leq 6 \); at the location \( l_b \) at time 1.1, at the location \( l_a \) at time 2.6 and so on as listed in the figure.

When combining several routes together, we will use superscripts to group locations by route. For instance, suppose in addition to the route shown in Figure 1, we have another route consisting of a single location access: \((l_1, 4.4)\). Then we can represent both routes as \( S = (S^1, S^2) \) where the first route, \( S^1 \), consists of six locations: \( S^1 = (l_1, l_2, l_3, l_4, l_5, l_6) = ((l_8, 1.1), (l_9, 2.6), (l_6, 3.2), (l_3, 5.1), (l_5, 7.7), (l_7, 9.8)) \) while the second route has one: \( S^2 = (S^2_l) = ((l_1, 4.4)) \). Also, when the specific time points at which the locations were accessed are not important, we will use a simplified notation. The aforementioned routes \( S^1 \) and \( S^2 \) can be timelessly represented as \( Q^1 \) and \( Q^2 \): \( Q^1 = (Q^1_1, Q^1_2, Q^1_3, Q^1_4, Q^1_5, Q^1_6) = (l_8, l_9, l_6, l_3, l_5, l_7) \), \( Q^2 = (Q^2_l) = (l_1) \).

B. The Test

Human routes recorded from one of the two environments (called the training environment) constitute the training data and are made available to the AI agent. The AI agent is then asked to create its own routes in the other environment (called the test environment) and its routes are recorded. A group of human judges are then individually presented with a playback of the routes created by humans and the AI agent in the test environment. Each judge is asked to label each route as either ‘human’ or ‘computer’. The entire process is then repeated with the two environments swapped in their training/test roles. We define an agent’s identification rate as the percentage of the agent’s routes the judges correctly label. An agent passes our Turing test if the 95% confidence interval for its identification rate falls between the identification rates of 45% and 55%. That is to say that an agent passes our test if we can claim with 95% confidence that the identification rate falls within 5% of the chance level. Note that the human judges are trained at this task by being exposed to human routes prior to the test.
Our objective of minimizing the difference between the agent’s identification rate and the chance level rate raises some important questions. What if the judges are unmotivated and randomly guess at all times? What if the routes are not presented in an intelligible way to the judges? Is the task at which the agent is being tested cognitively rich so that passing the test is a meaningful achievement in believable agent design or understanding intelligence?

We will first 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 [1]. These skills are non-trivial and humans hone these skills as they mature [2]. Other animals vary greatly in their performance in hiding and seeking with species having more highly evolved spatial abilities exhibiting better performance [14].

The ecological validity of virtual environments as tests of navigation and memory has been suggested through experiments that have shown transfer of spatial knowledge from virtual environments to the real space counterpart in both adults [15] and children [16], [17], [18]. Although differences in performance between virtual and real environments are sometimes observed (e.g., a greater tendency to underestimate distance in virtual environments [19]) many fundamental findings hold in both environments. For instance, researchers found that in both virtual and real environments Alzheimer disease patients were impaired relative to controls in navigating to a goal when cues on the walls provided the only cues for navigation but not when start position provided the only cues for navigation [20]. Others found that the developmental process of spatial knowledge acquisition was comparable in real and virtual maze tasks [21]. Moreover, fMRI studies in humans [22] showed distinct neural activation for wayfinding and route following in a virtual town, which corresponds to the neural dissociations shown in rodents for place and response learning in real maze tasks. Consequently, virtual environments have been used extensively over the past couple of decades to investigate neural and behavioural processes underlying spatial navigation [23], [24], [22] and to investigate, diagnose or rehabilitate spatial cognitive impairments [20], [25], [26], [18], [27], and even to develop strategies to interdict terrorists carrying radioactive material [28].

Directly relevant to the present work, we have shown in two previous studies [1], [29] that hiding and searching behavior in virtual rooms is similar in many respects to that seen in real-life rooms. In these experiments, people searched for and hid objects in bins or under tiles on the floor of a room. In our first set of studies [1], the room was a simple rectangular space with nine bins that provided hiding and searching locations. Importantly, the same systematic differences between hiding and searching emerged in both real and virtual versions of the room. Specifically, in both environments people chose locations farther from origin and they dispersed their choices more when hiding than when searching. In our second set of studies [29], people were tested in more complex virtual and real rooms that contained several dozen hiding locations provided by tiles on the floor of the room. In some conditions the rooms were empty and in others they were cluttered with furniture. For our purposes, the main finding of interest was that in both empty and cluttered rooms, people continued to choose locations farther from origin and to disperse their choices more when hiding than when searching. The generality of these differences between hiding and searching behavior across real and virtual tasks and across simple empty spaces and more cluttered complex spaces justified the use of an empty virtual room for the purpose of our study.

To argue that the information given to the judges is sufficiently rich and that the judges are properly motivated and skilled, we develop several reasonably complex hide-and-seek agents and show that they are easily distinguished from humans by the judges. Admittedly, being “reasonably complex” is a matter of opinion and we, as the authors, may be biased. So we present the “reasonably complex” designs in the paper for the reader to judge.

Finally we will briefly investigate whether human judges become better with practice and whether they appear to distribute labels evenly between “human” and “computer”.

III. RELATED WORK

A. Related Work in Psychology

Work in psychology indicates that hiding and seeking strategies are cognitive processes. Birds of higher spatial and social intelligence develop more complex hide and seek strategies related to their survival, especially with respect to food caching [30], [31], [32]. Also, the complexity of children’s hide and seek strategies varies with age [33], [34].

More recent studies [1], [29] considered hiding and seeking behavior of adult humans in a 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 location). The published results focus on correlations between hide or seek frequency at certain locations and room features. For example, [29] shows that there is a positive correlation between a location’s distance from walls and hiding frequency at such locations, but a negative correlation between distance from walls and seeking frequency. This indicates that humans do not seek in the same places they hide. These kinds of observations are helpful when mapping out the basic framework of an agent, but are not sufficiently specific to predict particular hide/seek locations. To the best of our knowledge, no generative computational models useful for an AI agent mimicking the hide/seck behavior have been published in psychology literature to date.

B. Related Work in Computing Science

The Turing test, originally called the “imitation game”, was introduced in 1950 [35] and is often linked to believability of AI agents [36]. There are two well known public competitions related to the Turing test. The Loebner Prize [10] test is the classic teletype version of the Turing test and has not been passed to date. The BotPrize [12], replaces the teletype environment with a competitive first-person shooter environment. To compete, an AI agent is no longer required to communicate, but must move and engage in combat in a way indistinguishable from humans (as deemed by human
judges). There have been a number of unsuccessful attempts to pass the test (e.g., [37], [38], [39]) and only very recently the first positive results were reported [40]. While the technical details are still scarce, it is unclear how to apply the winning designs to our problem of human-like behavior in the tasks of hiding and seeking objects in a novel environment. There are no combat and no opponents – both critical components of the BotPrize test.

C. Related work in Video Games

Historically in first-person shooters, an AI agent is created by selecting behaviors believed to be important for the agent to appear human and hard-coding them into the game. For example, in the game Counter-Strike: Source the agents were programmed to specifically mimic the slow reaction time and attention prioritization displayed by humans [41]. Other researchers [42] used visibility maps and Boolean logic to make non-playable characters (NPCs) take cover safely. These types of hand-coded approaches are game/task-specific and may not be portable to other tasks.

There has been work on predicting possible player locations in first-person shooter games. Some proposed a model using particle filters to predict human locations, removing the dependence on cheating with omniscient knowledge [43]. The particle filter approximates opponent locations by simulating possible opponent paths from the last known location. Others have improved on the particle filter method by adding hidden Markov models and simulacra [44], [45]. These prediction models are promising but they do not demonstrate how to actually create an autonomous agent. Additionally, these models do not offer portability as they require training data for the very environment they are used in.

Similarly to the human prediction models for first-person shooter games, there has been work in creating human prediction models for real-time strategy (RTS) games. Researchers have demonstrated how to predict multiple unit’s paths given limited observations, using an assumption that units only make small deviations from optimal paths [46], [47]. Although these approaches are helpful to an RTS game designer, the optimal path assumption may not be human-like. More recently, researchers tracked units after losing sight using a particle model [48]. The model shows promising predictions, but similarly to its first-person shooter particle counterparts, provides no way to collapse the predictions into a single human-like course of action that would be needed to pass our version of the Turing test.

IV. PROPOSED APPROACH

In order to pass the specialized Turing test described in Section II, we needed a model capable of generating human-like hide and seek routes. The model was to be trained on human routes recorded in one environment and then applied to generate new routes in another, previously unseen, environment. We also wanted to have variability in our model so that it generates different routes over multiple runs in the same environment. Finally, we set out to automatically learn as much of the model as possible using the training data, thereby reducing the manually engineered component.

We decomposed route generation into two sub-behaviors, selecting locations and navigating among them. The decomposition made the design more modular as well as gave us insight into the relative contributions of each sub-behavior. For location selection sub-behavior, we created a simple (L1) and an advanced (L2) strategy. Likewise, for the movement sub-behavior we developed a simple (M1) and an advanced (M2) strategy. In order to explain each strategy in detail we include a small artificial example running throughout this section.

The training and testing environments for our running example are in Figures 2(a) and 2(b) respectively. In the example, an agent starts at the door in the testing room which contains eight floor tiles where the participant can hide/set objects. The agent entered the room, sought at the location \( l_7 \), then at the location \( l_4 \), and is now looking for the next location to select. We will now demonstrate how the next location is selected under the L1 and L2 strategies.

A. Location Selection Strategies

Each of the two selection strategies (L1 and L2) is given an ordered list of locations already selected by the agent, and asked to make a new selection. Given the list, the strategies assign probabilities to each location in the room, thereby imposing a probability distribution. The next selection is then stochastically drawn from the distribution, with replacement for the seek task and without replacement for the hide task to reflect the location revisitation practices of humans.

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

1) Strategy L1: uniform random selection: the strategy assigns equal probability of selection to all eligible locations.

2) Strategy L2: data-driven location selection: the strategy is based on three probability distributions: \( P_D \), \( P_A \), and \( P_R \), created automatically from the training data (i.e., human location selections observed in the training environment). The distribution \( P_D \) is based on the distance between consecutively selected locations, \( P_A \) is based on the rotation angle between consecutively selected locations, and \( P_R \) is based on the last time the location was selected. Then L2 uses these three probability distributions each time it is asked to
select the next location. Specifically, each location receives three probability values, one for each distribution. A product of these three distributions comprises L2’s final distribution: \(P_{L2} = P_D \times P_A \times P_R\). We will now detail each of the three distributions.

**Spatial distance probability distribution function.** The PDF \(P_D\) for spatial distance between consecutively selected locations is computed as follows. First, we rank all possible distances between location pairs in the training room. Then we assign a weight to each distance as the ratio of the number of times the distance occurred between consecutive human selections to the number of times the distance occurs between all location pairs in the room.

Mathematically, we first build a set of all possible unique location pair distances: \(D = \{E(l_i, l_j) \mid 1 \leq i, j \leq n\}\) where \(E(l_i, l_j)\) is the Euclidean distance between the centers of locations \(l_i\) and \(l_j\). We sort the set \(D\) in the ascending order and build an index function \(I_D: D \rightarrow \{1, \ldots, |D|\}\) such that for any possible distance \(d\) between two locations, \(I_D(d)\) gives \(d\)’s index in the sorted set \(D\). Thus, \(I_D(E(l_i, l_j))\) represents the rank of the distance between the location \(l_i\) and \(l_j\) among all distances between location pairs.

For example, in our training environment there are six unique distances between all pairs of locations. We have \(I_D(E(l_1, l_1)) = I_D(0) = 1\) (the shortest distance is zero), \(I_D(E(l_5, l_6)) = 2\) (the distance between locations 5 and 6 is tied for the second shortest distance with many other location pairs) and \(I_D(E(l_1, l_9)) = 6\) (the distance between locations 1 and 9 is tied for the largest distance). Figure 3 lists \(E(\cdot)\) and \(I_D(E(\cdot))\) for the training environment.

![Fig. 3. Pairwise location distances (left) and their ranks (right).](image)

To illustrate: the location pair counts in the training data are shown in Figure 4. For instance, the location \(l_1\) was selected immediately after the location \(l_9\) nine times. The distance between these two locations is \(d = E(l_1, l_9) = 2.83\) as per Figure 3. The distance 2.83 has a rank of 6 in this environment (Figure 3) and will thus be contributing to \(P_D(I_D(2.83)) = P_D(6)\). Three other location pairs contribute to the value of the PDF for rank 6: \((l_7, l_9), (l_3, l_7), (l_1, l_9)\) which collectively happen in the training data 7+11+6 times. Thus, after the training data is processed, the PDF value for rank 6 is set to \(P_D(6) = (9 + 7 + 11 + 6)/4 = 33/4\) since \(X = 4\) (there are four location pairs with the distance of 2.83). The ratio is found in the last column of Table I. Once all training data is processed, the \(P_D\) values are normalized:

\[
P_D(i) = \frac{P_D(i)}{\sum_{j=1}^{|D|} P_D(j)}. \quad 1 \leq i \leq |D|. \tag{3}
\]

The resulting values are listed in the bottom row of Table I as well as displayed in Figure 4.

| Rank : 1 \ldots |D| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Number selected | 528 | 2512 | 1227 | 492 | 305 | 33 |
| \(P_D(i)\) | 0.190 | 0.340 | 0.249 | 0.133 | 0.062 | 0.027 |

**TABLE I**

**Spatial distance PDF for our example.**

![Fig. 4. Location pair selection counts from the training data (left). Spatial distance PDF \(P_D\) built from it (right).](image)

**Rotation angle probability distribution.** To compute \(P_A\) we start by considering all triplets of locations. For any three locations, \(l_a, l_b,\) and \(l_c\), we define \(G(l_a, l_b, l_c)\) as the angle between \(l_a l_b\) and \(l_b l_c\). An example of \(G(l_a, l_b, l_c) = 108.4^\circ\) is shown in Figure 2(a). We form the set of all unique angles \(A = \{G(l_a, l_b, l_c) \mid l_a \neq l_b \neq l_c, 1 \leq a, b, c \leq n\}\), sort the set in an ascending order and build an index function \(I_A: A \rightarrow \{1, \ldots, |A|\}\) such that for any angle \(\theta \in A, I_A(\theta)\) gives its index in the sorted set \(A\).

We then process the training data. Once again, we consider each selection history \(Q^j = \{q_i^j \mid 0 \leq j \leq |Q^j|\}\). Then for each location triplet \((q_{i-1}^j, q_i^j, q_{i+1}^j)\) in it, we compute the angle \(G(l_a, l_b, l_c)\) defined (e.g., \(l_b = l_c\)).

*1*This artificial example is given for illustration purposes only. The data in Figure 4 is made up. The PDF for the actual data is shown in Figure 8.

*2*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., \(l_b = l_c\)).
angle $\theta = G(q_{i-1}^l, q_i^l, q_{i+1}^l)$ and update the frequency of the corresponding index $I_A(\theta)$ in the PDF. The update is scaled by the number of times that angle occurs in the environment. Formally, if $X$ represents the number of angles in the environment with the same rank as $\theta$:

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

(4)

then we update $P_A$ as:

$$P_A(I_A(\theta)) \leftarrow P_A(I_A(\theta)) + \frac{1}{X}. \quad \text{(5)}$$

Finally, we normalize the frequency for each rank to get the probability distribution function over the angle ranks:

$$P_A(i) \leftarrow \frac{P_A(i)}{\sum_{j=1}^{|A|} P_A(j)}, 1 \leq i \leq |A|. \quad \text{(6)}$$

Selection recency probability distribution. For the last distribution, we will first define a recency number. Given a sequence of locations the agent has already visited/sought at, $Q = (q_1, \ldots, q_n)$, the recency number of the location $l$ is the number of locations since and including the most recent access to $l$ in $Q$. Mathematically: $R(l | Q) = |Q| - \max \{i | q_i = l\} + 1$. If the location $l$ has not been previously accessed ($l \not\in Q$) then $R(l | Q) = \infty$. To illustrate, suppose a location selection history is $Q = (l_6, l_5, l_6, l_3, l_5, l_7)$ as per Figure 1. Then $R(l_7 | Q) = 1$, $R(l_5 | Q) = \infty$, $R(l_6 | Q) = 4$.

Suppose the training data consists of $J$ location sequences $Q^j = (q_{i_1}^j, \ldots, q_{i_k}^j)$ where $1 \leq j \leq J$ is the sequence (i.e., route) number. Then each location $q^j_i$ has a recency number with respect to the part of its route ending immediately before $q^j_i$. Taking over the entire training data set $Q$, such recency numbers run between 1 and $\infty$. With a finite data set, there is a highest finite recency number, denoted by $r_{\text{max}}$.

To illustrate: let us suppose the training data consists of two routes $Q^1 = (l_8, l_9, l_6, l_9)$ and $Q^2 = (l_7)$. Then the following recency numbers are computed for all five locations encountered in the two routes. Going through the route $Q^1$ we compute $R(l_8 | (l)) = \infty$, $R(l_9 | (l)) = \infty$, $R(l_6 | (l_8, l_9)) = \infty$, $R(l_9 | (l_8, l_9, l_6)) = 2$, $R(l_7 | (l_8, l_9, l_6)) = 2$. For the second route $Q^2$ a single recency number is computed $R(l_7 | (l)) = \infty$. Thus, the maximum finite recency number $r_{\text{max}} = 2$.

For each recency number $r$ between 1 and $r_{\text{max}}$ we can compute the ratio of the number of times $r$ actually happened to the number of times $r$ could have possibly happened. Mathematically, this quantity is:

$$P^*(r) = \frac{\sum_{j=1}^J \{i | R(q_{i}^j | (q_{j-1}^j, \ldots, q_{1}^j)) = r\}}{\sum_{j=1}^J \max \{|Q^j| - r, 0\}}. \quad \text{(7)}$$

When $r$ exceeds $r_{\text{max}}$, the quantity $P^*(r)$ is undefined. Continuing with the example above, we compute $P^*(1)$ as the ratio of the number of times the recency number $r = 1$ was encountered in the training data, 0, to the number of times it could have possibly been encountered in the data: $\sum_{j=1}^J \max \{|Q^j| - 1, 0\} = 3 + 0 = 3$. Thus: $P^*(1) = 0/3 = 0$. For the recency number $r = 2$ which was encountered once in the training data as $R(l_9 | (l_8, l_9, l_6)) = 2$, we compute $P^*(2) = 1/(2 + 0) = 0.5$.

After training, given a possible route length $n \in \mathbb{N}$ we define $x_n^* = \min\{n, r_{\text{max}}\}$. Then the probability distribution over the recency numbers ($r \in \{1, 2, \ldots, x_n^*, \infty\}$) is defined as:

$$P_R^n(r) = \begin{cases} P^*(r) & 1 \leq r \leq x_n^*; \\ 1 - \sum_{k=1}^{x_n^*} P^*(k) & r = \infty. \end{cases} \quad \text{(8)}$$

Intuitively, given a route of $n$ locations, $P_R^n(r)$ gives the probability that the next, $(n + 1)$-th, location will yield the recency number $r$. Mathematically, for any given route length $n$, the function $P_R^n(r)$ is indeed a probability distribution over $r \in \{1, 2, \ldots, x_n^*, \infty\}$ insomuch as:

$$\forall n \in \mathbb{N}, \sum_{r \in \{1, 2, \ldots, x_n^*, \infty\}} P_R^n(r) = 1. \quad \text{(9)}$$

Using the example data above, we will demonstrate computing the distribution $P_R^2$ for $n \in \{1, 2, 3\}$. For $n = 1$, $x_1^* = x_1 = \min\{1, 2\} = 1$. Thus we can define the probability distribution $P_R^1$ over $\{1, 2, \ldots, x_1^*, \infty\}$ which becomes $\{1, \infty\}$. Specifically, $P_R^1(1) = P^*(1) = 0$ as we computed earlier. The second value of the distribution is $P_R^1(\infty) = 1 - \sum_{k=1}^{x_1^*} P^*(k) = 1 - 0 = 1$. Intuitively, this means that for any route of $n = 1$ location, the probability of getting the recency number of 1 with the next (i.e., second) location is 0. This makes sense as the recency number of 1 means immediately repeating choice and we never saw such in the training data. The probability mass is concentrated entirely over the recency number of $\infty$ is 1.

For $n = 2$, $x_2^* = x_2 = \min\{2, 2\} = 2$ which allows us to define $P_R^2$ for $r \in \{1, 2, \ldots, x_2^*, \infty\} = \{1, 2, \infty\}$. As before, $P_R^2(1) = P^*(1) = 0$. For $r = 2$, we have $P_R^2(2) = P^*(2) = 0.5$. Finally, for $r = \infty$ we have $P_R^2(\infty) = 1 - \sum_{k=1}^{x_2^*} P^*(k) = 1 - (0 + 0.5) = 0.5$.

The last example is for $n = 3$ which exceeds $r_{\text{max}} = 2$. Thus, $x_3^* = x_3 = \min\{3, 2\} = 2$ which allows us to define $P_R^3$ for $r \in \{1, 2, \ldots, x_3^*, \infty\} = \{1, 2, \infty\}$. The calculations are the same as in the previous paragraph, leaving us with the probability mass evenly divided between the recency numbers of 2 and $\infty$: $P_R^3(2) = P_R^3(\infty) = 0.5$. As before, $P_R^3(1) = 0$.

Using the distributions. Above we defined three probability distribution functions, $P_D, P_A, P_R$ which model the training data (i.e., location sequences recorded in the training environment). However, our AI agent is required to operate in a novel test environment for which it lacks any training data and, consequently, has no distributions. To solve this problem, we port the three distributions to the novel environment in the following fashion.

Suppose an agent hiding or seeking in a novel test environment has already visited $n$ locations $Q = (q_1, \ldots, q_{n-1}, q_n)$. Then the strategy L2 stochastically selects the location $l$ as the next location $q_{n+1}$ with the probability $P_D(i) \times P_A(j) \times P_R^n(R(l | Q))$ where the quantities $i$ and $j$ are defined below.

Let there be $|D|$ distinct inter-location distances in the training environment and $|D'|$ distinct inter-location distances in the test environment. Then, given the latest already selected
location \( q_n \) and a candidate location \( l \), we first compute the distance between them, \( E(l, q_n) \), and then take its rank \( i' = I^{D'}(E(l, q_n)) \) among all pairwise distances in the test environment. We then linearly scale the rank \( i' \) to the range of the training environment: \( i = i' = 1 + (i' - 1) \frac{|D - 1|}{|D' - 1|} \). The result, \( i \), is in the range \([1, |D|]\) and becomes an input to the distance distribution for the training environment: \( P_{D}(i) \).\footnote{Note that the distribution \( P_{D} \) for the training environment was defined only for the discrete inputs \([1, \ldots, |D|]\). We use linear interpolation to extend \( P_{D} \) to all points in the continuous interval \([1, |D|]\). This is done because the scaled rank, \( i \), can be anywhere in \([1, |D|]\) and not just in the set \([1, \ldots, |D|]\).} Note that for the first location to be selected the expression \( E(l, q_n) \) is not defined as there is no preceding location \( q_0 \). In that case, \( q_0 \) is artificially set to be the entry point to the environment (e.g., room’s door).

A similar scaling and interpolation procedure is applied to the angle distribution \( P_A \). Specifically, the quantity \( j' \) is the rank of the angle \( G(l, q_n, q_{n-1}) \) among all \( |A'| \) distinct angles possible induced by the locations in the test environment. Mathematically, \( j' = I_{A'}(G(l, q_n, q_{n-1})) \) which puts \( j' \) in the set \([1, \ldots, |A'|]\). We scale it linearly to place it in the range \([1, |A|]\) of angle ranks for the training environment: \( j = 1 + j' - 1) \frac{|A| - 1}{|A' - 1|} \). The resulting scaled rank \( j \) becomes the input to \( P_A \) with linear interpolation used to handle non-integer values of \( j \). Note that the expression \( G(l, q_n, q_{n-1}) \) is undefined in two cases: when \( n < 2 \) and when the angle among \( l, q_n, q_{n-1} \) is ill-defined (i.e., when two of the three locations coincide). In either case, we set \( P_A(j) = 1/|L'| \) where \( L' \) is the set of locations in the test environment.

No scaling is required for the input to the recency distribution \( P_R \) since it does not directly depend on the environment geometry. However, we have a special case when the recency number (finite or infinite) for the candidate location \( l \) exceeds any finite recency number observed in the training data: \( R(l|Q) > r_{\text{max}} \). Then we set the probability of selecting the location \( l \) as the \((n + 1)\)-th choice as follows. We compute the probability mass of the recency number of infinity scaled by the number \( \Omega \) of all test locations whose recency number exceeds \( r_{\text{max}} \): Mathematically: 
\[
P^n_R(R(l|Q)) = P^n_R(\infty)/\Omega \]
where \( \Omega = |\{P \in L'| R(P|Q) > r_{\text{max}}\}|. \)

B. Movement Strategies

Once a sequence of locations is selected using the strategies L1 or L2, a movement strategy is used to navigate between them. We designed two movement strategies as follows.

1) Strategy M1: spline interpolation: this strategy starts with the desired sequence of locations to be visited. In the example in Figure 5, the agent begins at the door and intends to visit the locations \( l_7, l_4, l_1, l_6 \) and then return to the door. First, the strategy M1 runs \( A^* \) on a rectangular grid overlaid on the environment. As a result, it gets a path connecting the locations (Figure 5, left).

Next, M1 replaces each location in the sequence with a guide point. The guide point is the earliest point on the path from which the agent can access the location (i.e., perform hiding or seeking at that location). This was done because humans usually perform the hide and seek action at a selected location as soon as they get close enough to it. In Figure 5, left we show the “close enough” radii as the dashed circles around the locations and the guide points as the black dots at the intersection of the \( A^* \) path and the circles.

M1 then fits cubic splines between the guide points to smooth the path. The resulting smooth path is checked for intersecting obstacles and walls in the environment. If it does (as shown in the middle plot of the figure) then a new \( A^* \) search is run between the guide points on each side of the intersection. The middle of the resulting path becomes a new guide point and the spline fitting is conducted again. The process is repeated until all splines fit inside the environment and do not intersect any obstacles/walls. The final path is shown at the right of the figure. This process produces a smooth trajectory visiting the location sequence selected by the strategy L1 or L2. The agent always faces the direction of its velocity vector.

2) Strategy M2: data-driven route fitting: for this strategy we first constructed a database of movements recorded in the training data. To do this, we took each route (e.g., Figure 1) and split it into segments between each consecutive location selected by the participant (Figure 6, left). Each segment was stored in the database with each point’s coordinates (including the timestamps) recorded relative to the start of the segment.

This database was then used to compute the AI agent’s paths in the test environment. Specifically, given a sequence of locations \( (l_1, \ldots, l_n) \) to visit, the agent considered all consecutive location pairs \( (l_i, l_{i+1}), i < n \). For each such pair, the strategy M2 translated, rotated and linearly scaled every segment in the database so that it connected \( l_i \) and \( l_{i+1} \). An example of such fitted segments for a particular pair of locations is shown in Figure 6, middle. All segments not fully contained within the environment were discarded.\footnote{If no segments remained then the agent fell back on the strategy M1 for that pair of locations.} All remaining segments were then evaluated for their quality of fit – a product of the scaling fit and the continuity fit, described in detail in Section V-B. In Figure 6 right the segment \( A \) is fitter than the segment \( C \) because it is scaled less than \( C \) and has less angular difference from the previous segment.

Once all segments connecting the pair of locations \( l_i \) and \( l_{i+1} \), \( 1 \leq i < n \) were evaluated for their fit quality, a uniform random selection from the top 10 fittest segments was used as the path between the locations. The stochasticity was introduced to add diversity to M2 and, consequently, make the agent less predictable while still using scaled, rotated and translated training (i.e., human) paths.

C. Agent Structure

The four possible combinations of these strategies: (M1,L1), (M1,L2), (M2,L1) and (M2,L2) — gave us the four agents (A1 through A4) shown in Table II. The agents were of different complexity and had different performance as discussed later in the paper. Note that all four agents are designed to operate in a novel environment for which they have no training (i.e., human) data available.
Fig. 5. The movement strategy M1.

Fig. 6. The movement strategy M2.

<table>
<thead>
<tr>
<th></th>
<th>Random selection (L1)</th>
<th>Data-driven selection (L2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spline-interpolation movement (M1)</td>
<td>Agent A1</td>
<td>Agent A2</td>
</tr>
<tr>
<td>Data-driven movement (M2)</td>
<td>Agent A3</td>
<td>Agent A4</td>
</tr>
</tbody>
</table>

TABLE II
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 non-human-like behavior. It was meant to study performance of the judges when such behavior is presented. A6 was designed to behave like A5 at first but then switch to A4 behavior. It was meant to check if judges can be tricked by displaying obvious non-human behavior at first, followed by more human-like behavior.

V. EMPIRICAL EVALUATION

In this section we present an empirical evaluation of our model by running the agents A1 to A6 through our specialized version of the Turing test. We implemented the hide and seek environments using the Source engine and the art assets of Half-Life 2 [50]. Two environments shown in Figure 7 were built using the Hammer editor [51]. The locations are represented by black floor tiles.

We then ran a group of human participants in the environments and recorded their routes while both hiding and seeking (Section V-A). The collected data were used to develop agents A1 through A6 (Section V-B). Finally, another group of human participants were recruited as the judges (Section V-C).

A. Data Collection

The dataset we used to create our models was the recordings of human participants in the two virtual environments. Our participants were recruited from first-year psychology courses and received a partial course credit for the participation.

The subjects were first trained to use the keyboard-and-mouse controls of Half-life 2 by performing hiding and seeking in a separate training room. Then they were asked to perform a hide or a seek task in one of the two environments. In the hide task the participants were asked to hide three objects and to “make your objects difficult for other people to find.” In the seek task the participants were asked to select locations until three previously hidden items were found. The seek task was limited to one minute, while the hide task was not time limited. Participants were free to move about the room, but had to wait for one second between selecting locations.

As the subjects performed their tasks, their avatar’s locations and orientations were recorded once per second. Additionally, each location selection and the time at which the selection occurred were recorded. Overall, 5142 paths were recorded, each containing between 6 and 125 data points. They constituted the training data for the agent development.

B. Agent Development

As described in detail earlier, the strategy L1 selected locations uniformly randomly while the strategy L2 used the training data collected by observing human behavior in both rooms. Under L2, three probability distribution functions \( P_D, P_A, P_R \) were built. Several observations can be drawn from their shape as follows.
First, humans tend to prefer making selection choices near their current position (Figure 8). This appears intuitive as accessing floor locations around the current position is easier and faster for it requires less travel. Second, the highest peak in angle rotation is 0° which, in Figure 9, corresponds to the 50th percentile in the ranking spanning the range of \(-180°\) to \(180°\). The two secondary peaks correspond to \(-90°\) and \(90°\) rotation angles (i.e., the two right angle turns). This is likely due to the regular placement of the locations on a rectangular grid (Figure 7) and the preference of some humans to traverse the said grid systematically, row by row or column by column. Third, the more recently a location was selected the less likely humans were to select it again (Figure 10) although there was a tendency to select the very same location twice in a row. This tendency is possibly due to the human participants’ missing the selection confirmation message and thus re-clicking on the floor tile right after selecting it.

For the movement strategy M2, the scaling quality and the continuity quality were defined as follows. Suppose a segment
of length $a$ was fitted between two locations $b$ distance apart (by shortest path). Then the scaling quality of the fit is $\min\{a/b, b/a\}$. Intuitively, a segment that needs to be shrunk to half of its original length to fit between given locations is of the same fit quality as the segment that needs to be enlarged two times. The continuity quality of fitting a line segment was computed as $1 - \Delta^\circ/180^\circ$ where $\Delta$ is the absolute value of the difference in the agent’s heading angle at the end of the previous segment and the beginning of the segment being fit.

The two additional agents A5 and A6 were implemented as follows. The agent A5 picked two neighboring locations in a corner of the environment, stood between them so that it could access both of them without moving and then alternately selected them (i.e., hid or sought at them). While doing so it rotated in place with a constant angular speed. This was meant to give away its non-human nature. The agent A6 was equivalent to A5 for the first two routes and to agent A4 for the following routes. This was done to investigate whether the judges can be affected by demonstrating non-human behavior first, followed by human-like behavior.

C. Judging

We recruited 288 human participants from first-year courses in psychology as our judges. This group did not overlap with the one described in Section V-A. The judges were divided into six groups of 48, one group for each AI agent design (A1-A6). Each of the six groups was further divided into two sub-groups of 24, for the two possible room orderings (room A followed by room B or room B followed by room A).

Human and computer hide/seek behavior was presented to the judges in the form of top-down video recordings of routes (Figure 7(b)). In the figure, the agent is standing by the door and it just accessed the location immediately on its right, indicated with a yellow glow. Each judge watched 5 training recordings followed by 10 test recordings with no overlap between training and test recordings. The judges knew that the training recordings were of humans. The judges were told that some test recordings may be of an AI agent and some may be of humans but no specific proportion given. In actuality, the proportion was approximately half. The judges were not
told that there may be multiple agent designs. Each recording, including the training recordings, was played at double speed, lasting between 3 and 44 seconds. At the end of each test recording, the judge labeled it as “Human” or “Computer” using the interface shown in Figure 11 which queued in the next recording. The judges were also able to re-watch the most recent recording multiple times before labelling it. In total each judge labeled 40 recordings, 20 for hide task and 20 for seek task. Each judge’s identification rate was calculated as the percentage of correct labelings. The judges were not informed of their identification rates.

VI. Study Results

The mean identification rates across all judges are shown in Figures 12, 13 and 14, together with 95% Wilson confidence intervals. If such a confidence interval lies completely below another interval we can conclude that the former agent is correctly labeled less often than the latter agent with greater than 95% confidence. The identification rates are divided by task (hide/seek), by agent (human/AI) and by judge experience (low/high). We will analyze them below.

A. Hiding versus Seeking

Figure 12 shows agent identification rates for the hide and seek tasks. For the hide task, AI agents using the data-driven movement strategy appear significantly more human-like than agents using the spline-based M1 strategy. Indeed, the 95% confidence interval for A3 (M2,L1) is below the interval for A1 (M1,L1) and the interval for A4 (M2,L2) is below the interval for A2 (M1,L2). On the other hand, data-driven selection strategy L2 is not significantly different from the random location-selection strategy L1.

For the seek task, the AI agent A4 was judged significantly more human-like than the agents A1, A2, A3 (which are not significantly different from each other). The L2 strategy produced no significant difference in the hide task, but did so in the seek task when paired with M2 strategy (the agent A4 was judged significantly more human-like than the agent A3).
The location selection strategy appears to play a smaller part in the hide task because its low number of location choices (three) may not suffice to judge a route as human or AI.

B. Humans versus AI Agents

Figure 13 shows judge identification rates for human versus AI agent routes. Note a general trend that the identification rate of human routes rose when such routes were presented in the same batch with weaker agents and fell when mixed with stronger agents. The 95%-confidence interval for the human identification rate when paired with the agent A4 was below the human intervals of A1, A2 and A3. Similarly the human interval for A5 was above such for A1, A2 and A3. The agents A4 and A5 were designed to be the strongest and weakest agents respectively (which is supported by the positions of their computer intervals in the figure).

![Figure 13. 95% confidence intervals for identification rates presented by agent type. Two additional intervals are shown: ‘A6/A5’ shows the first two routes of A6 (when it is equivalent to A5). ‘A6/A4’ shows the last three routes of A6 (when it is equivalent to A4).](image)

The fact that judges’ success in identifying humans as humans appears to depend on the type of the AI agent presented in the same batch is curious. To explain it, we conjecture that human judges tend to assume an equal mixture of routes in a batch. Specifically, if there are 10 routes a judge is to label as “human” or “computer” then he/she will tend to assign 5 “human” and 5 “computer” labels. This is despite the fact that the judges were never told that there is an expected balance in the routes presented to them.

The agent A6 generated an intentionally non-human route (using A5) as the first two routes it was run for and then used the most human-like design (A4) for the following three routes it was run for. Most judges correctly labeled the first two routes as “computer” (the mean identification rate for A5 is 88.5%). The remaining eight routes (three produced by the agent A4 and five produced by actual humans) appear all human-like. By guessing randomly but maintaining the equal balance proportions (i.e., labelling random three out of the eight routes as “computer” and the remaining five routes as “human”) the judges would have an expected identification rate of 37.5% for the computer (A6, last three routes) and 62.5% for humans. The empirical means of 35.3% and 61% appear to support this conjecture.

C. Effects of Practice

In Figure 14 we investigate the effects of practice on judges performance. We show the identification rates broken down by judge experience. Each judge labeled 20 routes in each of the two rooms (the room order was counter-balanced). In the figure, the ‘less experience’ intervals show 95% confidence intervals for judge performance for different agent types averaged over the first 20 routes across all judges. The ‘more experience’ confidence intervals are for the last 20 routes.

![Figure 14. 95% confidence intervals for identification rates by judge experience.](image)

While the judges did not receive any feedback on their labelling, their performance appears to improve with practice for all AI agent types but the amount of improvement varies.

VII. CONCLUSION AND FUTURE WORK

Hiding and seeking are important cognitive abilities of humans and animals and have several applications in video game design. This paper made the following contributions. We proposed the first computational generative model designed to mimic people hide and seek behavior in a virtual room. The model is built automatically from observed human hide and seek behavior. We implemented a model within an AI agent and demonstrated its validity via a restricted version of the Turing test. Specifically, four AI agents based on the model were constructed and evaluated. The most complex of the four agents used machine-learned patterns of human behavior and appears to have passed the restricted version of the Turing test. Specifically, the 95% confidence intervals for A4 fall within the range of 45% to 55% for both the hide and seek tasks. In other words, we are 95% confident that the identification rate of human judges for the agent A4 is within 5% of chance.

Future work will investigate applicability of this model to other animals and other spatial tasks. Furthermore, while this study used only sparsely populated geometrically simple rooms in the empirical evaluation, future work will investigate the extent to which the agent designs evaluated in this paper manage to capture human behavior in environments with more complex geometry.
varied hiding locations and higher visual fidelity found in many commercial video games. If this is successful then our agents can be incorporated into a video game to take advantage of predicting player’s search patterns.

VIII. ACKNOWLEDGMENTS

We appreciate funding from the National Science and Engineering Research Council. We appreciate fruitful discussion with the IRCL group: David Thue, Daniel Huntley, and Gregory Lee. We thank Valve Software Corp. for developing software and releasing their toolset. Finally, we appreciate feedback from our reviewers.

REFERENCES


