ABSTRACT
Most path planning techniques use exact, global information of the environment to make optimal or near-optimal plans. In contrast, humans navigate using only local information, which they must augment with their understanding of typical building layouts to guess what lies ahead, while integrating what they have seen already to form mental representations of building structure. Here, we propose Scene Planning Networks (SPNets), a neural network based approach for formulating the long-range navigation problem as a series of local decisions similar to what humans face when navigating. Agents navigating using SPNets build additive neural representations of previous observations to understand local obstacle structure, and use a network-based planning approach to plan the next steps towards a fuzzy goal region. Our approach reproduces several important aspects of human behavior that are not captured by either full global planning or simple local heuristics.

CCS CONCEPTS
• Computing methodologies → Real-time simulation; Animation; Machine learning; Planning and scheduling.

KEYWORDS
Path planning, navigation, character simulation, deep neural networks
either full global planning or simple local heuristics: SPNet agents will integrate the history of what they have seen to improve their navigation, make different decisions in response to varying levels of certainty with the task at hand, and will stochastically follow a variety paths in situations where multiple different paths to reach the goal are reasonable. Importantly, the training approach used in SPNet reaches human-like performance on these tasks without requiring any human training data, allowing the approach to be easily adopted in a variety of applications.

2 RELATED WORK

Our method draws from research in Computer Graphics, Deep Learning, and Cognitive Science. Here we examine some closely related work from these fields.

2.1 Local & Global Navigation

Previous work in human-like navigation has considered both local and global contexts. Related work in local navigation has focused on realistic behaviors [Kapadia et al. 2015; Karamouzas et al. 2017; McDonnell et al. 2008], often with data-driven [Charalambous and Chrysanthou 2014; Karamouzas et al. 2014; Wang et al. 2016] or geometric [van den Berg et al. 2008] reactive approaches. Others have looked to produce realistic local behaviors by modeling problems similarly to what humans experience [Lee et al. 2019; Ondřej et al. 2010]. Here, we take a similar approach, broadening the scope to considering compelling human-like behavior in the context of global navigation.

Graph search has been used for global navigation in interactive games, VR, and animation to allow virtual characters to plan over a graph-like representation of their environment, producing graph-optimal global routes to their goals, [Botea et al. 2013; van Toll et al. 2016]. Other works have focused on acceleration techniques [Lee and Lawrence 2013], or exploiting heuristics and optimized structures to accelerate the search [Kallmann and Kapadia 2014]. Planners such as Rapidly-exploring Random Trees and its variants [LaValle 1998] use a sampling based approach to build plans in continuous environments under control constraints, while others apply computer vision to directly predict next actions [Rabiee and Biswas 2019], perform waypoint navigation [Bansal et al. 2020], or vision-based navigation [Gupta et al. 2017].

Very related to our work is the area of agent-centered search. Works in this field provide approaches for real-time heuristic search under local information and time constraints. Approaches such as D* Lite [Koenig and Likhachev 2002] and LRTA* [Korf 1990] incrementally learn heuristic information in an online fashion, and extensions focus on improving convergence properties [Hernández and Meseguer 2005], learning speed ([Koenig and Sun 2009; Sturtevant and Bulitko 2011]), and memory efficiency [Bulitko and Björnsson 2009]. LRTA provides a learning framework with a tunable relationship between performance and optimality [Bulitko 2004]. Here, our work will focus on creating human-like behavior (as opposed to guarantees on heuristic admissibility) while allowing the goal to be fuzzy (our agents only have a distribution describing where the goal may be).

2.2 Deep Learning

Deep Learning (DL) techniques have been broadly applied to train large neural networks in many fields [Sze et al. 2017]. Approaches for navigation can either be supervised with known training data (as in [Pfeiffer et al. 2017]), or the network can be trained to optimize path cost using reinforcement learning (as in [Tamar et al. 2016] and [2018]). Deep reinforcement learning (DRL) has recently shown promise for navigation as a fine-tuning final step, as was done in [Pfeiffer et al. 2018] and [Luo et al. 2020]. Very recent work has sought to improve the practicality of DRL navigation by augmenting it with classical, analytical planners ([Chaplot et al. 2020]). Neural networks have also proven useful in transforming raw input into rich and meaningful representations [Cai et al. 2019; Doersch 2016]. Additionally, works such as [Peters and O’Sullivan 2002] have used learning techniques to model ways people understanding sensory information. Eslami et al. used an additive representation network to encode virtual scene descriptors that can be used to query views from novel perspectives [Eslami et al. 2018]. A similar cognitive structure has recently been used for efficient robot navigation on a grid [Gupta et al. 2017]. We adapt a similar architecture to build scene representations in a continuous environment.

3 LOCAL-GLOBAL PLANNING

Like humans, SPNet agents are assumed to have only local, limited information about their environment and imprecise knowledge of their long term goals. The agent’s task, then, is to integrate its local observations, together with a learned understanding of typical building layouts, to plan efficient paths toward likely goal locations. We assume the agent is updated in real-time following a sense-plan-act loop where it computes its direction of travel frequently based on its history of recent observations.

This section details the underlying problem formulation and path execution strategy of the SPNet framework. In the next section we will introduce a novel network structure which makes the actual predictions of the next optimal step for the agent.

3.1 Problem Formulation

Given a single agent tasked with navigating to given goal whose location is not precisely known, we represent each aspect of the problem as follows:

Agent Representation. The agent is represented by a circle with a radius $r$ large enough to encompass the collision model associated with the agent’s animation.

Environment Representation. The agent’s environment is defined by a series of line-segments which represent impassible obstacles, and the agent is assumed to be able to move otherwise freely and continuously in $\mathbb{R}^2$. An example map can be seen in Figure 2.

Goal Representation. An agent is not given the exact location of its goal, but rather only a general fuzzy notion of its location. We represent this as a 2D Gaussian distribution centered at ($\mu_x, \mu_y$), with a standard deviation of $\sigma$ in each dimension.

Sensing and memory. An agent can only directly observe the portion of line segment obstacle walls which are visible from its current position. In practice, we represent this as a series of $i$ rays
which specifies the maximum number of times the agent can visit walls [Lozano-Pérez and Wesley 1979]. We refer to any such (potentially optimal) point as a navigation node, and statically compute all such navigation nodes for any map (purple x’s in Figure 2). This discretizes the continuous navigation problem without sacrificing any potential path quality by choosing between reachable navigation nodes. We move navigation nodes to be $r$ away from the closest wall to account for the agent’s radius.

Our fundamental execution strategy is as follows. Given a set of navigation nodes, a start node, and a goal distribution, our agents utilize a neural-network to select the next navigation node to travel to. At each step, given the current and recent local visibility isovists, the network predicts the next optimal node position. The agent then moves to whichever of the neighboring navigation nodes (e.g., currently visible nodes) is closest to the network’s prediction. To guarantee the agent eventually reaches the goal (and avoids infinite travel loops), we divide the tasks into separate responsibilities: a Scene Representation Network whose job is to build a representation of map seen so far, and a Planning Network, which takes this representation along with the (fuzzy) goal location and predicts the next step to take. This separation of responsibilities also leads to greater flexibility in how the network is used in execution.

We implement the Scene Representation Network using a three layer neural network (500-300-100 nodes respectively), with a ReLU activation function between each layer. The input of this network is the normalized 60 isovist ray lengths, together with the x and y offset of where the isovist was seen relative to the agent’s current position. The output of this network is a 200 valued feature vector which represents the environment. The Scene Representation Network is run once for each isovist in the agent’s path history, defined as the most recent $n$ isovists (and their offsets) that the agent has seen. The resulting encoding from each isovist are then summed to produce the final 200-dimensional scene representation vector. Critically, this additive representation does not require the agent to use a fixed path history size in execution or training. Summing the Scene Representation Network output over more or fewer previously observed isovists will change the agents behavior in a natural fashion. This behavior tuning is discussed further in Section 5.2.

The Planning Network combines the 200-dimensional scene representation together with the goal region to determine the likely next actions. The goal input is represented as three parameters defining a Gaussian which describes a distribution over possible goal locations. These values are concatenated with the encoding, and the resulting 203 dimensional vector is then passed through another three layer network (100-150-200 nodes respectively), with a ReLU activation function between each layer. The output of this network encodes the next step for the agent to take.

4.1 Network Structure

The structure of our prediction network (shown in Figure 1) draws on the idea of mental maps. Rather than taking in a visibility isovist (or series of isovists) and directly predicting the next location, we divide the tasks into separate responsibilities: a Scene Representation Network whose job is to build a representation of map seen so far, and a Planning Network, which takes this representation along with the (fuzzy) goal location and predicts the next step to take. This separation of responsibilities also leads to greater flexibility in how the network is used in execution.

4.2 Navigation Prediction

Rather than producing a single 2D coordinate for the next navigation step, our network outputs a stochastic prediction split over two alternatives: $p_0 = (x_0, y_0)$ and $p_1 = (x_1, y_1)$ represent the probable offsets (relative to the agent) of optimal next navigation nodes, and $c$ ranges continuously from 0 to 1 represent the network’s choice between $p_0$ and $p_1$ (0 for choice 0, 1 for choice 1).

We train this output using a three-part loss function:

$$L(p_0, p_1, c, y) = k_ml_{\text{min}} + k_0l_{\text{avg}} + k_pl_{\text{pred}}$$

with $p_0$, $p_1$, and $c$ as defined above, $y$ as the true optimal prediction for this scenario, and $k_m$, $k_0$, and $k_p$ as tuning parameters that relatively weight the three components of the loss.

Each of the three components of the loss function serves a different purpose. The first term:

$$L_{\text{min}}(p_0, p_1, y) = \min(s_0\|p_0 - y\|, s_1\|p_1 - y\|)$$

Figure 2: Feature Representation. The goal distribution is shown as 1 and 2 standard deviation rings ($\sigma = 2m$).
measures how close the better of the two predictions is to the optimal next node. The $s_i$ terms, computed as $s_i = 5 \cdot \sigma(||p_i - y|| - ||p_a - y||)$ where $p_a$ is the agent position, add penalties to discourage overly short predictions. The $\min()$ ensures this component of the loss only provides a reward for improving the better of the two predictions. This allows the network to split the prediction, as there is no penalty for the wrong half of the split. Combined over training data representing ambiguous scenarios, the two predictions are driven towards different, but equally promising next steps.

Simply optimizing the better of two predictions during training is insufficient when one of the two predictions is much further away from the correct node than the other. The second loss term, $L_{\text{avg}}$, addresses this by providing a small penalty for the average of the two predictions:

$$L_{\text{avg}}(p_0, p_1, y) = (\|p_0 - y\| + \|p_1 - y\|)/2 \quad (3)$$

This term must have a small weight to ensure the primary loss comes from $L_{\text{min}}$ to maintain split predictions.

The network also must learn to indicate which of the two predictions is the correct using an output value $c$. This is accomplished via the third loss term, $L_{\text{pred}}$:

$$L_{\text{pred}}(p_0, p_1, c, y) = |(c - \sigma(||p_1 - y|| - ||p_0 - y||))| \quad (4)$$

where the sigmoid function, $1/(1+e^{-x})$, maps the relative difference between the errors of the two options to the continuous domain $[0, 1]$. This loss will be driven to zero when $c$ matches the (sigmoid of) the relative error. When $c$ is exactly 0 or 1, the agent should move to $p_0$ or $p_1$ respectively. When $c$ is between these values, the network can stochastically select an option, leading to a spread of reasonable paths. Unless otherwise stated, for all results we used a $k_m$ of 0.999, a $k_a$ of 0.001, and a $k_p$ of 10.

Figure 3 shows the effect of the network’s split prediction. At the top left of the map, the network does not have enough information to know which side of the courtyard is most optimal, and bifurcates the prediction between the two hallways. When the agent is in a less ambiguous part of the building, the network is more confident in the next step. We performed a user study to compare SPNet paths to those taken by real humans for the same tasks (see Appendix E). As Figure 4 shows, in this and other ambiguous scenarios, both user and SPNet paths bifurcate similarly.

Figure 4: Human-Like Behavior: SPNet produces efficient, human-like paths both for maps on which it was trained (left), and maps not seen in training (right).

4.3 Data Generation & Training
The Scene Representation Network and the Planning Network are trained together as one system. The training data is generated by an exhaustive sampling of every reachable start-goal pair of navigation nodes in training maps (see Appendix D for dataset details). Given a start-goal pair, we permute possible path histories to generate training rows.

In training, the network is never given the actual goal position, but a distribution with a mean sampled from a gaussian centered at the true goal location with uncertainty $\sigma$. In order to integrate this uncertain representation into a gradient-based optimization architecture, we employ a custom network layer which samples the fuzzy goal region to generate a specific training instance (the dark shaded box in Figure 1). Importantly, the $\sigma$ used to perturb the goal location is also an input to the navigational network, enabling the input $\sigma$ to be used as a run-time tuning parameter that controls how much the agent is willing to explore.

The network weights are trained using Keras [Chollet et al. 2015] using stochastic minibatch gradient decent with an ADAM optimizer. Further training details are given in Appendix B.1.
5 RESULTS & ANALYSIS

5.1 Network Analysis

As discussed above, allowing the network to split its prediction lead to more human-like behaviors. Additionally, these split predictions improve the training performance of the network. Figure 5 shows a comparison of the SPNet network trained on a single map with 6m of goal uncertainty with and without split predictions. With split predictions enabled, there is over a 60% reduction in the final loss $L$ (from 3.68 down to 2.26). Similarly, the additive representation enabling path history also led to improved loss in training. Figure 5 shows that incorporating path history lead to an additional 15% reduction in the training loss.

5.2 Behavioral Analysis

Our problem formulation and custom network structure allows SPNet agents to display several human-like navigation behaviors that are not possible either with optimal planning techniques or simple local heuristics. SPNet agents respond only to their local conditions, explore in search of vague goals, integrate their observations over time, and intelligently backtrack when they get stuck in local minima. The framework runs in real-time, with a full planning step taking 1-2ms (see Appendix B.2 for runtime details).

As the SPNet predictions are a distribution over two likely possible actions, stochastically selecting one can create a natural diversity of paths. Because the choice of which node to go to next follows the distribution predicted by the network, the agent paths tend to vary more in ambiguous situations (see Figure 4).

Figure 6 shows a SPNet agent navigating past a fork juncture on an apartment-style map. Here, when the goal uncertainty is small (left), the agent navigates straight to the goal, taking a near-optimal path. While this path might be expected from a person who is very familiar with both this building and the exact goal location, its not very reflective of the ambiguity inherent in the navigation task. Increasing the goal uncertainty (right) naturally leads to an exploration behavior.

SPNet agents have two main tunable parameters which can control the behaviors: the size of the goal region $\sigma$ (creating less certainty in the agents decisions), and the maximum path history $n$ (creating more intelligent backtracking behaviors). These two parameters can also interact with each other. For example, for low $\sigma$, paths are more efficient regardless of the size of the path history.

Figure 7 explores the interaction of goal uncertainty and path history by comparing SPNet paths to optimal averaged over 200 random tasks spread over the 5 training maps. A larger history size increases path optimality across goal uncertainties, even for history sizes and uncertainties beyond those used in training. A 2-way ANOVA reveals that goal uncertainty has a more significant effect on the agent’s path than history (history $F=2.4$, $p < 0.1$, uncertainty $F=7$, $p<0.05$). These results confirm the ability of the scene representation and planning networks to generalize outside the range of parameter values used in training.

Figure 8 (and Appendix C.3) show SPNet agents navigating in a variety of challenging paths on untrained maps. The agents typically find relatively efficient and natural paths that reach their goal despite having never seen the map in training.

6 LIMITATIONS & FUTURE WORK

While SPNet covers an important aspect of a human-like global navigation, it focuses on single agent navigation. Planning in environments with multiple agents typically require specialized search techniques [Atzmon et al. 2020] or hierarchical models [Musse and...
Thalmann 2001]. Expanding the network input to include the relative position and velocity of other agents could combine local and global planning into an integrated network. While SPNet is already realtime for a single agent, faster performance may be needed to support large crowd simulations with thousands of agents in a shared scene. Here, it may be possible to accelerate isovist construction using spatial data structures, and to speed up network computation by using GPUs.

In conclusion, we have presented SPNet, a comprehensive framework for human-like global navigation under local information in building environments. The result is a real-time system capable of intelligent navigation behaviors, including exploration and backtracking, environment familiarity, varied routes per task, and efficient but sub-optimal routes toward fuzzy goals.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grants IIS-1748541 and CHS-1526693.

REFERENCES


A EXECUTION DETAILS

Algorithms 1 & 2 detail the SPNet execution strategy. Because SPNet executes very quickly, it is possible to plan fully to the goal and smooth away minor inefficiencies in the resulting path. Here, we generally avoid such smoothing techniques as they can suppress desired exploration behaviors. One exception is that if a network predicts visiting a node it has already visited (or a node for which the agent has already seen all of the node’s neighbors), the we allow the agent to move directly toward the next predicted node as soon as it comes into view. In practice, the particular character animation system used may provide some additional path smoothing as the character animates between nodes.

Algorithm 1: SPNet

Input: map (Walls and Nodes), start (Node), goal (Node), goalRegionMean (Position), goalRegionSigma (float)
Output: path (Node List)

#global maxVisitCount = 3;
/* visit count for each node */
#global visits = [0] * map.numNodes;
/* path is full route (will include back-tracking sequences) */
path = [];
/* history is path with cycles removed */
history = [];
current = start;
while current != goal do
    next = SPNetStep(map, current, goalRegionMean, goalRegionSigma, history, path);
    path.push(next);
    current = next;
end
return path;

B NETWORK DETAILS

B.1 Training Details

The network was trained on 10 cores of an 2.3 GHz Intel(R) Xeon(R) CPU. Training was run for 230 epochs that iterated through all the training data, with each epoch taking about 3 minutes, and the entire run taking 12 hours. As can be seen in Figure 9, most of the improvements in loss came at the start of training, but the longer run ensured the network had sufficient time to converge. Training was performed across 5 maps and included all possible paths with a history of length three for a total of 2,315,665 training examples, each of which was sampled with three levels of goal uncertainty (sampled from 0, 1, and 2m gaussian distributions respectively), leading to about 7 million samples trained each epoch. Epochs were trained in batches of size 100,000 using the Keras framework with the ADAM optimizer and a dynamic learning rate that started at 3e-4 and ended at 1e-5.

To improve numerical stability, isovist ray lengths are normalized to lie on the interval [0, 3] by dividing by 1/3 of the maximum length of 30m. Unless otherwise stated below, the network was trained with a goal uncertainty \( \sigma \in \{0, 1, 2\} \) meters, and a maximum path history of size 3. We note that while 2m may seem like a small distance, in practice sampling from a 2D gaussian with i.i.d dimensions at \( \sigma = 2m \) often yields points significantly farther than 2m from the mean (in fact, the 1 std dev circle having diameter 4m will probabilistically contain less than half of sampled points). As discussed in Section 5.2, the network was able to extrapolate good behavioral performance beyond the uncertainties and history sizes on which it was trained.

Algorithm 2: SPNet Step

Input: map (Walls and Nodes), current (Node), goal(Node), goalRegionMean (Position), goalRegionSigma (float), history (Node List), path (Node List)
Output: next (Node) visits[current]++; if isVisible(goal) then
    next = goal;
else
    /* gets all visible nodes from current node visited less than maxVisitCount times */
    candidates = map.getVisibleNodes(current, maxVisitCount);
    if candidates.size == 0 then
        /* retrace our steps */
        next = history.pop();
    else
        position = getNetworkPrediction(current, goalRegionMean, goalRegionSigma, candidates, path);
        next = map.getClosestNode(position);
        history.push(current);
    end
end
return next;
Table 1: User study and participant summary.

<table>
<thead>
<tr>
<th></th>
<th>Users</th>
<th>Tasks</th>
<th>Avg. Samples/Task</th>
<th>Game Experience (hrs/mo)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trained</td>
<td>Untrained</td>
<td></td>
<td>None/Rare</td>
</tr>
<tr>
<td>Study 1 (selected tasks)</td>
<td>48</td>
<td>8</td>
<td>2</td>
<td>40.45</td>
</tr>
<tr>
<td>Study 2 (random tasks)</td>
<td>27</td>
<td>25</td>
<td>15</td>
<td>9.85</td>
</tr>
</tbody>
</table>

C.1 Comparison to Local Heuristics

SPNets bridge the gap between global planning techniques and local navigation heuristics. Human route selection studies have revealed high-quality local route selection heuristics that strongly influence human paths. Two of these techniques from the cognitive science domain that can be directly implemented in the broader navigation framework of this paper are:

- Traveling as far as possible towards a goal [Bailenson et al. 2000], which we will refer to as Closest-to-Goal (CTG)
- Maintaining a small relative angle in heading with respect to the goal [Lima et al. 2016], which we will refer to as Angle-to-Goal (ATG)

Figure 11 shows the results of replacing the network predictions with CTG and ATG-based heuristics. When compared to the tasks in the user study data (see Appendix E), both CTG and ATG had path lengths which were more similar to human paths that the optimal route selection. This result is consistent with the existing findings in psychology of the general importance and applicability of these local navigation heuristics to humans. On the tasks tested in the user study, SPNet matches human path lengths as well as (or better than) these heuristics, while exhibiting more natural behaviors (ATG tends to oscillate and CTG can get stuck in obstacle local minima) and supporting uncertain goals and memory integration.

Some over-fitting effects can be observed in trained maps where SPNet produces path lengths closer to optimal than human paths, while matching human path lengths better on untrained maps. This suggests that techniques to help combat over-fitting such as larger training data sets and early-stopping may further improve SPNet's path quality on trained maps. Alternatively, this over-fitting can be exploited as a feature when agents should exhibit familiarity with a particular environment.

C.2 Effect of History

Figure 12 shows an example of the key navigational behaviors SPNet agents exhibit. On the left is an agent with a max path

Figure 10: SPNet Runtime: Average total runtime for each step of planning in SPNet across history and map size. 50 random tasks were executed for a small (House) and large (Business Park) map to generate timing samples. Runtime cost is primarily split between executing the prediction network, scene network, and computing isovist features.

Figure 11: Differences from mean user path lengths on Round 2 of the User study for trained and untrained map sets.

Figure 12: Effect of Path History: Two simulated trajectories for the same task. The left path was generated by an agent with no path history, and the right trajectory the agent incorporates the past three visited map nodes in planning.
Table 2: Datasets: the collection of maps analyzed in this work. The table indicates the size of the maps, and the number of instances of training data that would be generated from that map. The three largest maps and one smaller map were excluded from training and used to test generalization.

<table>
<thead>
<tr>
<th>Walls</th>
<th>Nodes</th>
<th>Size (m)</th>
<th>Data (#)</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>29</td>
<td>28</td>
<td>12x8</td>
<td>16 K</td>
</tr>
<tr>
<td>Apartment</td>
<td>65</td>
<td>65</td>
<td>14x9</td>
<td>157 K</td>
</tr>
<tr>
<td>House</td>
<td>72</td>
<td>84</td>
<td>29x19</td>
<td>168 K</td>
</tr>
<tr>
<td>Courtyard</td>
<td>75</td>
<td>97</td>
<td>17x12</td>
<td>867 K</td>
</tr>
<tr>
<td>Medical</td>
<td>101</td>
<td>109</td>
<td>23x27</td>
<td>1.1 M</td>
</tr>
<tr>
<td>Office</td>
<td>84</td>
<td>95</td>
<td>28x17</td>
<td>781 K</td>
</tr>
<tr>
<td>University Labs</td>
<td>129</td>
<td>154</td>
<td>44x26</td>
<td>4.66 M</td>
</tr>
<tr>
<td>Business Park</td>
<td>161</td>
<td>197</td>
<td>33x27</td>
<td>4.60 M</td>
</tr>
<tr>
<td>Conference</td>
<td>276</td>
<td>332</td>
<td>61x33</td>
<td>15.44 M</td>
</tr>
</tbody>
</table>

C.3 Generalizing to Large Maps

SPNet agents can perform well on large maps not seen in training even for relatively long or complex tasks (Figures 16, 15).

Figure 13: Effect of history size on path optimally averaged across all paths with goal size $\sigma=2$.

Figure 14: Remembering the Past: Our representation network allows for the integration of past observations that influences future decisions, resulting in more human-like path lengths on untrained maps.

history of 3 isovists and a relatively large goal distribution ($\sigma = 2m$ of uncertainty, shown as the shaded circle at 1 std dev radius). After initially exploring the room overlapping the bottom half of the goal distribution, the agent rules out the area as a potential goal location and remembers this as it travels up the left side of the environment. In contrast, an agent who is using only its single, current isovist (no path history) is unable to remember where it has been. Figure 12 right shows such an agent, who must return to places it had been before, eventually finding a longer route to the goal.

We can effectively increase the size of the history simply by summing additional encodings together before sending them to the planning network. As shown in Figure 7 and Figure 14, this additive encoding scheme allows the SPNet to make meaningful use of additional history even outside the sizes it was trained on. Figure 13 shows the effect of planning over longer histories on the optimality of the resulting paths. Plans that account for more history lead to more optimal paths, even when using longer histories than were seen in training.

Figure 15: Selected SPNet trajectories on a business suites map (history size 3, uncertainty = 0m).

Figure 16: Selected SPNet trajectories on a conference center map (history size 3, uncertainty = 0m).
While we employ a supervised learning approach, it would be impractical to gather enough human trajectory data to demonstrate good behavior on a meaningful percentage of the potential tasks. For example, the 6 maps we use in our training dataset contain 6.9 million training examples of different start-goal pairs permuted with possible previous observations. In order to create this data, we instead use a graph-optimal search to determine the optimal path from start to goal, and use the resulting step-wise decisions in training.

While the SPNet network is trained on optimal paths, our goal is not to produce purely optimal behavior. Rather, attempting to be globally optimal while restricting the network to only local, uncertain information results in paths with human-like sub-optimal behaviors, even on maps used in training.

To facilitate training and evaluation, we create a dataset of environments based on real floor plans from a variety of buildings of different shapes and sizes (see Figures 19 and 20). Maps used in training the network include three smaller buildings, two larger buildings, and a small test map made by hand. This dataset consists of a wide range of different environments, with very different types of layouts, and very different sizes (see Table 2). Additionally, we withheld a set of maps from training to be used in analysis of how the method performs in untrained scenarios. All maps are rendered in Figures 20 and 19.

E USER STUDY
To validate the human-like properties of SPNet paths, we gathered human paths via a user study approved by the Institutional Review Board (IRB). During the study, participants performed navigation tasks via a 3D game-like interface in a web browser (on their own machines) through a rendering of the maps in Table 2. A minimap displayed a live update of the relative goal location, but not walls in order to provide some natural uncertainty in the exact goal location. Participants were asked to aim to complete 15 navigation tasks, but were allowed to do as many or few as they chose, with the average user completing 16.9 navigation tasks (see Table 1 for additional details). Each user was given a sample "warm-up" task to acclimate to the controls. As the user moved through each task, their trajectory was tracked at a frequency of 10 samples per meter displacement. Trajectories for the same tasks were generated using SPNet, with the goal region centered on the true goal position. Before performing analyses, both user and simulated paths are smoothed using a low-pass running median filter as implemented in the runmed function of the R statistical programming language with k=11 to remove any small jittering in the paths. Users took one of two versions of the study: the first consisted of randomized hand-picked tasks for qualitative comparison (879 trajectories captured), and the second consisted of random tasks across all maps for quantitative comparisons (394 trajectories captured).

Quantitatively, we see SPNet has more similar path lengths to humans than graph-optimal planning. Figure 18 shows that path lengths of SPNet agents are more similar to study participants than paths generated using the graph-optimal routing. An ANOVA with Tukey post-hoc analysis shows strong separation between optimal path lengths and user path lengths as well as SPNet path lengths (p < 0.05), while showing the User and SPNet distributions as similar (p 0.43). Qualitatively, the user study results show SPNet captures the natural, efficient, and varied paths real humans take. Figure 4(left) shows SPNet on an ambiguous task in the Courtyard map, which was included in training. Here, SPNet agents follow different paths that encompass the majority of the types taken by the users in the study. In a similarly ambiguous task on the untrained University Labs map, Figure 4(right), SPNet agents are capable of finding both homotopies portrayed in human routes. In practice, we find SPNet agents tend to produce compelling split predictions in untrained environments, but are overly confident between the two choices as compared to maps seen in training. Therefore, while SPNet agents show exploration behaviors, and human-like path lengths in both trained and untrained maps, it may be best in practice to train on maps when possible to ensure more path variety.
Below we show the maps used in training (Figure 19) and testing (Figure 20). All environments were based on real floor plans, abstracting away irrelevant obstacle detail.

Figure 19: Training Maps.

Figure 20: Testing Maps.