To appear in AAAI Spring 2014 Symposium on Qualitative Representations for Robots (AAAI-SSS), March 2014.

Multi-robot Human Guidance using Topological Graphs

Piyush Khandelwal and Peter Stone
Department of Computer Science
The University of Texas at Austin
2317 Speedway, Stop D9500
Austin, TX 78712, USA
{piyushk,pstone}@cs.utexas.edu

Abstract

Prior approaches to human guidance using robots inside a building have typically been limited to a single robot guide that navigates a human from start to goal. However, due to their limited mobility, the robot is often unable to keep up with the human's natural speed. In contrast, this paper addresses this difference in mobility between robots and people by presenting an approach that uses multiple robots to guide a human. Our approach uses a compact topological graph representation of the environment, and we first present the procedure for generating this representation. Next, we formulate the multi-robot guidance problem as a Markov Decision Process (MDP). Using a model of human motion in the presence of guiding robots, we define the transition function for this MDP. Finally, we solve the MDP using Value Iteration to obtain an optimal policy for placing robots and evaluate this policy's effectiveness.

Indoor environments such as airports, shopping malls, hospitals, and warehouse stores are characteristically full of people hurrying towards a destination or trying to locate a particular item. Often, they are unfamiliar with the environment and spend a fair amount of time locating these resources. With recent advancements in service robots, it is becoming far more feasible to deploy a large number of robots to aid humans in these environments. This paper studies how ubiquitous robots in an environment can be used to guide people efficiently to their destinations.

Past research has explored the possibility of using a single robot to guide people (Thrun et al. 1999; Philippsen and Siegwart 2003). However, in environments densely packed with moving people and goods, navigating a guide robot the entire length from a human's start location to their goal can be a significant challenge. The same navigation task can be completed far more efficiently by people, and yet they become limited by the navigation speed of the guide. A multi-robot solution can make use of a human's ease of navigation by proactively placing robots where the human is likely to need help in the future. Whenever the system needs to guide a human at a specific location, it can commission a nearby robot to direct the human towards the next objective, whether it be another guide robot or the goal. Once that robot's task is completed, it can go back to performing its other duties. Potentially, this approach can greatly reduce the time each individual robot has to spend guiding the human, allowing robots to assist more people in the same time.

This paper specifically studies the problem of deciding where to place robots in an environment to guide a human as he or she moves around. First, we formulate this multi-robot guidance problem as a Markov Decision Process (MDP), and use a hand-coded model of human motion in the presence of guide robots to define the transition function for this MDP. We then use Value Iteration (Sutton and Barto 1998) to solve this MDP, generating an optimal solution for placing robots. Such a solution can take the uncertainty in a human's movement into account and avoid actions that have a significant probability of failure. Finally, we evaluate the generated policy by comparing it against a heuristic solution for deciding robot placements. Experiments are run using the model of human motion, as well as with avatars controlled by real humans in a simulation environment.

To reason about human movement and robot placements, a representation of the environment is required. This paper uses a topological graph representation of the environment for reasoning (see Fig. 1h). Topological graphs can provide a compact representation of the environment while still retaining all key locations and the connectivity between these locations. The process of topological graph generation described in this paper is built on previous work (Thrun and Bücken 1996). As such, the main contribution of this paper is the procedure for generating topological graphs, formulating the multi-robot guidance problem as an MDP using these topological graphs, and solving the MDP using Value Iteration to produce the best graph nodes for placing robots. All code in this paper has been implemented using the ROS middleware package (Quigley et al. 2009) and is available in the public domain along with videos from the experiments[1].

Related Work

Single robot guides have been used over the past two decades to guide humans on a tour (Thrun et al. 1999), or provide navigation assistance to the elderly and visually impaired (Montemerlo et al. 2002; Lacey and Dawson-Howe 1998). In contrast, the goal of our work is to navigate able-

[1] https://github.com/utexas-bwi/bwi_guidance

bodied individuals inside a building by using multiple robots to guide them. The work of *Rogers et al.* has taken a much more subtle approach to influencing human routes inside a building using ambient displays (Rogers et al. 2010), and aims to change people's behavior towards making healthier choices such as taking the staircase instead of an elevator. In contrast, we use a much more direct interface by using a robot's screen to influence short term navigation decisions.

To efficiently guide a person inside a building, it is necessary to have a model of human motion based on their interaction with guide robots. Past work has looked into building models of how humans interpret natural language navigation instructions (Chen and Mooney 2011), and how such instructions can be used by robots to get to a navigation goal (Tellex et al. 2011). In this paper, we use directional arrows on a robot's screen to guide a person, which simplifies how we formulate the multi-robot guidance problem as a Markov Decision Process (MDP). In future work, we hope to test additional interfaces for human robot interaction.

In this paper, we use a topological graph representation of the environment to formulate the multi-robot guidance problem as an MDP and reason about human movement. Topological graphs have been used in the past for learning maps of the environment (Choset and Nagatani 2001), and for robot path planning and navigation (Thrun and Bücken 1996; Konolige, Marder-Eppstein, and Marthi 2011). The approach for topological map generation discussed in this paper is built upon previous work (Thrun and Bücken 1996), and produces graphs similar to Generalized Voronoi Graphs (Choset and Burdick 1995). We will explain why graphs generated in prior work are insufficient for reasoning about human movement, and present a refinement procedure to improve connectivity between key real world locations.

Problem Statement

Given an occupancy grid representation of the environment, along with the start and goal locations, a multi-robot guidance solution should produce locations for placing multiple robots to guide a human towards the goal. Solutions are evaluated based on how well they minimize the overall distance traveled by the human before reaching the goal. Our desiderata for such a solution are presented below:

- The solution should reason about the uncertainty in the human's movement, and avoid situations that result in a significant probability of the human making a costly mistake.
- The solution should be reactive as the human moves around in the environment, and commission robots when necessary to direct the human.

Additionally, in this paper we make the following assumptions regarding multi-robot guidance solutions:

1. The maximum number of robots that the solution can place is pre-specified.
2. Each robot can direct a human by displaying a directed arrow on its screen, and there may be some uncertainty as to how this information will be interpreted by the human. Other interfaces are not studied in this paper.
3. Whenever a robot is commissioned and assigned to a specific location, the robot gets there instantaneously.
4. Robots in the environment that have not been commissioned (or have been decommissioned) do not influence the human being guided.

With regards to assumptions 3 and 4, we leave consideration of robot path planning and travel cost to future work in order to focus on the robot placement problem in isolation. In a small environment with robots that move at moderate speed, this assumption, though not fully realistic, does not significantly affect the decision-making problem in question.

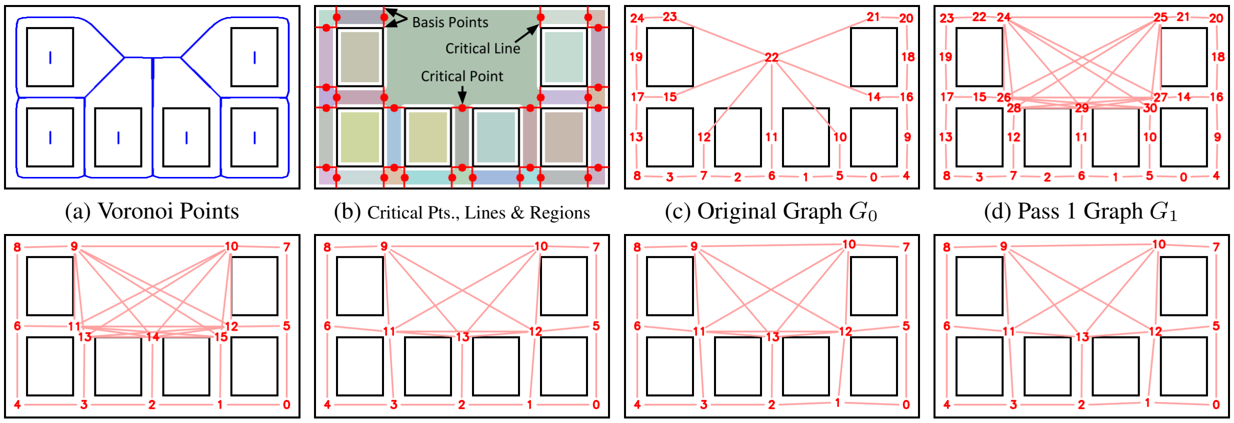
Topological Map Generation

An occupancy grid is a 2D matrix representation of the environment, where each cell in the matrix corresponds to a tile in the real world. The value contained in each cell reflects a probability measure of the corresponding real world tile being occupied. While such a representation is suitable for robot navigation and mapping, it does not support reasoning about key locations where humans decide to move in a particular direction. In contrast, this section discusses the construction of a topological graph from the occupancy grid representation. The graph representation is more compact, and only contains key locations along with the connectivity between these locations (see Fig. 1h), making it well suited for reasoning about human motion.

We formalize a topological graph as $G = \langle V_G, E_G \rangle$, where V_G is the set of all vertices and E_G is the set of all edges in the graph. Each vertex $v \in V_G$ is defined as a 3-tuple $\langle x, y, size \rangle$ representing the location of the vertex and the size of the underlying region. We define some functions on vertices $u, v \in G$ that will be used throughout the text:

$$
\begin{aligned}
isVisible(u,v) &= true \text{ if } \exists \text{ a line-of-sight obstacle free} \\
&\quad \text{path between } u \text{ and } v \text{ in the grid} \\
&\quad \text{representation, } \textit{false} \text{ otherwise.} \\
nodeAngle(u,v) &= \arctan((v.y - u.y)/(v.x - u.x)) \\
euclidDist(u,v) &= \sqrt{(v.y - u.y)^2 + (v.x - u.x)^2} \\
pathDist(u,v,G) &= \text{Distance of the shortest path between} \\
&\quad u \text{ and } v \text{ in } G \text{ using Dijkstra's Alg.} \\
&\quad \text{over Euclidean distance.} \\
adjacentNodes(v) &= \{w : e_{vw} \in E_G\} \\
visibleNodes(v) &= \{w : w \in V_G \mid isVisible(v,w)\}
\end{aligned}
$$

Constructing the graph follows an approach closely modeled after the work of *Thrun et al.* (Thrun and Bücken 1996). First, the Voronoi diagram (Aurenhammer 1991) of the grid-based representation of the environment is computed. The Voronoi diagram is the set of points in free space which are equidistant to their closest obstacles (see Fig. 1a). This distance is defined as the *clearance* of that Voronoi point, and the closest obstacles are called the *basis points*. From the set of Voronoi points, *critical points* are selected such that no Voronoi point within an ϵ-*neighborhood* of a critical point



(a) Voronoi Points (b) Critical Pts., Lines & Regions (c) Original Graph G_0 (d) Pass 1 Graph G_1

(e) Pass 2 Graph G_2 (f) Pass 3 Graph G_3 (g) Pass 4 Graph G_4 (h) Final Graph G_5

Figure 1: Fig. 1a shows the Voronoi diagram in blue generated over the occupancy grid representation of the environment. The environment is $29.6m \times 18.4m$ in size. Fig. 1b shows the critical points and lines demarcating space into different regions, which can be used to directly construct the graph G_0 in Fig. 1c. Figs. 1d-1h show the graph refinement process as explained in the paper.

has lower clearance than that critical point (with $\epsilon = 0.25m$). Critical points when connected to their basis points form *critical lines* that divide the environment into a number of *regions* (see Fig. 1b). The centroids of these regions can be joined together to produce the graph G_0 (see Fig. 1c).

Even with good region segmentation, the graph produced by directly connecting adjacent regions may be unsuitable for reasoning about human motion in large spaces. For instance, a human walking directly from node 15 to node 14 in G_0 will pass through the region represented by node 22 ($n22$), but not necessarily the point $\langle n22.x, n22.y \rangle$. G_0 does not capture the fact that when people move in large open spaces, they tend to move directly towards their intended exit from that space. Since this information is valuable when reasoning about human motion, we present a 5-pass graph refinement strategy that incorporates direct movements in large spaces into the graph:

1. In pass 1, any vertex $v \in G_0$ such that $v.size >$ *splitRegionSize* is split into its constituent critical points to produce G_1 (Fig. 1d). *splitRegionSize* is a user defined parameter and $15m^2$ was used in this paper.
2. There are a number of vertices in G_1 that are not necessary to reason about human movement, such as node 3 while moving between nodes 7 and 8. In this pass, graph G_2 (Fig. 1e) is produced by removing any vertex $v \in G_1$ that satisfies the following two conditions:
$$|adjacentNodes(v)| = 2 \wedge$$
$$\forall\, u, w \in adjacentNodes(v), u \neq w\ [isVisible(u, w)]$$
3. Pass 3 aims to remove nodes in G_2 that are too close to other nodes, as close nodes can introduce unnecessary transitions in the system while mapping real world locations to a graph node. This pass forms graph G_3 by merging any vertices $u, v \in G_2$ that satisfy:
$$euclidDist(u, v) < vertexMergeDistance$$
where *vertexMergeDistance* is user-defined (set to $2m$).
4. Some vertices $u, v \in G_3$ may not be visible as the line-of-sight path may skim a wall in the environment, such as the edge between nodes 10 and 12 in G_3. Pass 4 produces graph G_4 (Fig. 1g) by nudging every vertex $v \in G_3$ to any randomly selected point within a ϵ*-neighborhood* ($\epsilon = 0.25m$) that maximizes its visibility $|visibleNodes(v)|$.
5. Multiple close edges are irrelevant for reasoning about human motion. In G_4, when a human moves right from node 11, it is unnecessary to have a direct edge to node 12, and the movement can be captured by transitioning through node 13. Pass 5 removes any edge e_{uv} in G_4 that can be represented by a suitable combination of other edges:
$$pathDist(u, v, G_4 \setminus \{e_{uv}\}) \leq$$
$$edgeMergeFactor \times euclidDist(u, v)$$
where *edgeMergeFactor* is another user defined parameter set to 1.05. Fig. 1h shows the final graph G_5.

Additional examples of the result of this procedure appear in Fig. 4. Once a graph suitable for reasoning about human movement is constructed, it can be used by the multi-robot guidance solution as elaborated in the next section.

Multi-robot Guidance Solution

In this section, we frame the multi-robot guidance problem as an episodic Markov Decision Process (MDP). This MDP defines the choices the system can make for placing robots and directing people, and treats the choices of the human moving around in the system as part of the environment's transition function. Once the MDP has been framed, any MDP solver can be used to compute an optimal or approximate policy for placing robots. We use Value Iteration (Sutton and Barto 1998) to find this policy.

MDP Formulation

Given a topological graph G and a goal node g, we define the MDP as $M = \langle S, A_s, P^a_{ss'}, R^a_{ss'} \rangle$, where S represents the environment's state space. A_s is the set of actions the system can take at all states $s \in S$, where actions control placing robots around and directing people, and P and R are the transition and reward functions, respectively. The following sections define these elements of the MDP in detail.

State Each state $s \in S$ is represented by the 5-tuple $\langle$*curNode*, *curDir*, *visRobotLoc*, *pointToDir*, *robotsLeft*$\rangle$. All members of the state tuple are discrete and described below:

1. *curNode* is the node mapped from the human's current location. *curNode* can take any value in V_G. Any state s where *s.curNode* is the goal node is considered terminal.
2. *curDir* is an estimate of the humans' current direction of motion, which we discretize into 16 bins in the state representation. At the start of an episode, *curDir* is initialized by discretizing the human's true orientation and subsequently computed as the human moves to a new graph node as explained in Alg. 1 at Line 18.
3. *visRobotLoc* - A robot can be present at any node visible from *curNode*, and *visRobotLoc* tracks if and where such a robot exists. It assumes any value in:

   $$\{\text{NONE}\} \cup \mathit{visibleNodes}(\mathit{curNode}) \setminus \{\mathit{curNode}\}$$

   It is sufficient for *visRobotLoc* to track one robot only as only one robot is required to serve as the human's next objective. *curNode* is excluded from the set of possible values as the case where a robot is present at the current node is handled by *pointToDir*.
4. *pointToDir* - Once the human reaches a robot, the system uses that robot to guide the human in a given direction. *pointToDir* reflects whether the current node contains a robot, and if that robot has been assigned an adjacent node towards which it will guide the human. *pointToDir* assumes any value in:

   $$\{\text{NONE, UNASSIGNED}\} \cup \mathit{adjacentNodes}(\mathit{curNode})$$
5. *robotsLeft* is the number of robots left to be placed. It can take any value in $\{0, 1, \ldots, \mathit{maxRobots}\}$, where *maxRobots* is problem dependent.

Actions There are 3 different types of actions that the system can perform, and each individual state has a variable number of actions depending on $|\mathit{adjacentNodes}(s.\mathit{curNode})|$ and $|\mathit{visibleNodes}(s.\mathit{curNode})|$.

1. *placeRobot*(v) - Placing a robot provides the human with his next objective. The system can place a robot at any location v visible from the current location, and the following actions are available at state $s \in S$:

   $$\{\mathit{placeRobot}(v) : v \in \mathit{visibleNodes}(s.\mathit{curNode})\}$$

   placeRobot(v) can only be taken if *s.robotsLeft* > 0. The transition to the next state s' is deterministic and $s' = s$ except the number of robots left to be placed is one less (s'.*robotsLeft* = *s.robotsLeft* $-$ 1). There are additional conditions depending on the choice of v:

   - **Case 1:** $v \neq$ ***s.curNode***. The robot can only be placed if a robot is not visible (*s.visRobotLoc* = NONE). In the next state s', the system starts tracking this robot (s'.*visRobotLoc* = v).
   - **Case 2:** v = ***s.curNode***. A robot can only be placed on the current node if no robot is present already (*s.pointToDir* = NONE). The next state s' has s'.*pointToDir* = UNASSIGNED.
2. *directHuman*(v) - When a human reaches a robot, the robot has yet to be assigned a direction to guide the human (*s.pointToDir* = UNASSIGNED). The system can only take one of the following *directHuman* actions to guide the human towards an adjacent node:

   $$\{\mathit{directHuman}(v) : v \in \mathit{adjacentNodes}(s.\mathit{curNode})\}$$

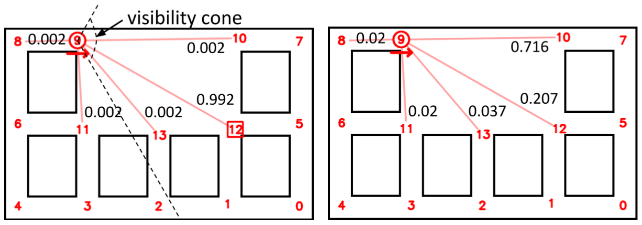
   Once *directHuman*(v) is taken, the system deterministically transitions to a next state s' where the robot guides the human by displaying an arrow towards v (s'.*pointToDir* = v).
3. *wait* - The system does not change the state of the system directly, but waits for the human to move inside the environment. This action is nondeterministic, as the transition depends on the graph node to which the human moves to.

Transitions The *placeRobot* and *directHuman* actions are completely deterministic and transition to a known state. Defining the transition function when the *wait* action is taken is considerably more complex, as it depends on how a human decides to move in the environment, and a model of human motion in the presence of guiding robots is required. In this paper, a hand-coded model of human movement is used which will be discussed in the next section.

When a human moves to an adjacent node *nNode*, the movement is represented as a transition to a unique MDP state s', which we compute as explained in Alg. 1. If no robot was visible in s, it is impossible for any robots to be present in s' (Lines 3-5). On the other hand, if a robot was visible in s, the human may have moved to the node containing it. The system updates *pointToDir* to reflect that a robot without an assigned direction is present(Lines 7-9). The human may also move such that a previously visible robot is no longer visible, and the system stops tracking that robot (Lines 10-12). If that robot is still visible, the system continues to track it (Lines 13-15). The human's direction of

Algorithm 1 State transition when a human moves

```
 1: $s \leftarrow$ Current State, $s' \leftarrow$ new State()
 2: $nNode \leftarrow$ Graph node after human movement
 3: if $s.visRobotLoc$ = NONE then
 4:     $s'.pointToDir \leftarrow$ NONE
 5:     $s'.visRobotLoc \leftarrow$ NONE
 6: else
 7:     if $s.visRobotLoc = nNode$ then
 8:         $s'.pointToDir \leftarrow$ UNASSIGNED
 9:         $s'.visRobotLoc \leftarrow$ NONE
10:     else if $s.visRobotLoc \notin visibleNodes(nNode)$ then
11:         $s'.pointToDir \leftarrow$ NONE
12:         $s'.visRobotLoc \leftarrow$ NONE
13:     else    $\triangleright$ the case where tracked robot is still visible
14:         $s'.pointToDir \leftarrow$ NONE
15:         $s'.visRobotLoc \leftarrow s.visRobotLoc$
16:     end if
17: end if
18: $s'.curDir \leftarrow discretize(nodeAngle(nNode, s.curNode))$
19: $s'.robotsLeft \leftarrow s.robotsLeft$
20: $s'.curNode \leftarrow nNode$
```



(a) Visible robot at node 12 (b) No Robots

Figure 2: Fig. 2a shows that the transition distribution is heavily in the favor of node 12 if a robot is visible there. On the other hand, Fig. 2b shows more variation if no robots are present.

motion is computed based on the transition to *nNode*, and other state members are updated accordingly (Lines 18-20).

Rewards A transition reward simply reflects the distance traveled by the human when the system goes from s to s':

$$R^a_{ss'} = -euclidDist(s.curNode, s'.curNode)$$

By this reward formulation, the optimal MDP solution minimizes the expected distance traveled by the human. Additionally, a reward is non zero if and only if the *wait* action is taken and a human moves in the environment:

$$R^a_{ss'} \neq 0 \iff a = wait$$

Human Motion Model

The human motion model needs to account for transitions to adjacent graph vertices under the influence of guiding robots (*nNode* in Alg. 1). This model can then be used to define the transition function of the MDP when the *wait* action is taken. A fully realistic human motion model should be stochastic and determined empirically by recording the decisions that real people make when moving around real maps.

Since the focus of this paper is on map representation and multi-robot guidance, a hand-coded human model has been used for experiments. This model computes transition probabilities according to these 2 cases:

- **Case 1: *s.visRobotLoc* $\neq$ NONE**. If the human sees a robot clearly visible from the current node, then it is expected that the human moves towards that robot. We define a robot as being clearly visible when it lies in the *visibility cone* by satisfying the following condition:

$$absAngleDiff(expDir, radians(s.curDir)) < \pi/3,$$
$$\text{where } expDir = nodeAngle(s.curNode, s.visRobotLoc)$$

  If this condition is true, then we compute the likely adjacent node v the human should transition to as:

$$v = \arg\min_{w \in W}(absAngleDiff(expDir, wAngle)), \text{where}$$
$$wAngle = nodeAngle(s.curNode, w) \text{ and}$$
$$W = adjacentNodes(s.curNode)$$

  In case a robot is additionally present at the current node and points to v, or is not present (*s.pointToDir* = v $\vee$ *s.pointToDir* = NONE), then the model states that with 99% probability the human moves to v. On the other hand, if a robot at the current node does not point to v (i.e. in the direction of a robot further up that the human can clearly see), the human is presented with a confusing situation. In this case, the model splits 99% probability evenly between v and $s.pointToDir$. The remaining 1% probability is split evenly among all outgoing edges to give each transition a finite probability, as illustrated in Fig. 2a.
- **Case 2: *s.visRobotLoc* = NONE**. If no robot is present at the current node (*s.pointToDir* = NONE), then the human is expected to continue moving in the current direction of motion, i.e. the expected direction of motion (*expDir*) is *radians*(*s.curDir*). Alternatively, if a robot is present at the current node and pointing towards an adjacent node (*s.pointToDir* $\neq$ NONE), then the human is expected to move towards that adjacent node along the direction:

$$expDir = nodeAngle(s.curNode, s.pointToDir)$$

  If there are multiple adjacent nodes in the direction of *expDir*, there may still be some uncertainty about the node the human decides to move to. The hand-coded model simulates this uncertainty by assigning a weight $weight_v$ to every outgoing transition to adjacent node $v \in adjacentNodes(s.curNode)$ as:

$$\begin{aligned} edgeAngle &= nodeAngle(s.curNode, v) \\ weight_v &= exp\left(-\frac{absAngleDiff(edgeAngle, expDir)^2}{2\sigma^2}\right) \end{aligned}$$

  where σ^2 = 0.1 controls the spread of the Gaussian function. 90% of the transition distribution is assigned using edge weights $\{weight_v\}$, and the remaining 10% is split evenly among all outgoing edges to give each transition a positive probability, as illustrated in Fig. 2b.

Solving the MDP using Value Iteration

Given an MDP and a model of human motion in the presence of robots, a complete description of the domain is available including the transition model $P^a_{ss'}$. Any MDP solver can now be used to compute an optimal or approximate policy for placing robots as a human moves around in the environment. In this paper, Value iteration (VI) (Sutton and Barto 1998) has been used to solve the MDP.

Value Iteration defines an expected value $V(s)\ \forall\ s \in S$, which is the maximum possible long term expected reward that the system can receive from the state s. Given the formulation of the reward structure in our MDP, $V(s)$ reflects the expected distance a human has to walk from state s to the goal, given that the system takes the best possible actions to reduce this distance. $V(s)$ can be computed iteratively using a simple backup operation:

$$V_{k+1}(s) = \max_{a \in A_s} \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma V_k(s')\right]$$

In practice, the backup is run iteratively until the maximum value change becomes less than a threshold ϵ:

$$\forall\ s \in S\ [\max(V_{k+1}(s) - V_k(s)) < \epsilon]$$

In this paper, we used $\epsilon = 0.05m$ and ran VI without any discounting ($\gamma = 1$). Additionally, $V(s)$ was never allowed to go below a threshold δ to speed up convergence. In states where no robots are left to be placed, $V(s)$ can be extremely low and take many iterations to converge. It is sufficient to identify these states as being unfavorable by selecting a low enough δ, set to -500m in experiments, and the following additional step was run after every backup:

$$V_{k+1}(s) = \max(V_{k+1}(s), \delta)$$

Once $V(s)$ has been computed, the system can select the optimal policy $\pi(s)$ for placing robots and directing humans:

$$\pi(s) = \arg\max_{a \in A_s} \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s')\right]$$

On a fixed map, this policy $\pi(s)$ can be pre-computed offline for any given goal beforehand, and looked up as needed.

Experiments

In this section, we present two experiments that evaluate the effectiveness of the VI-based solution for guiding a human, and compare it against the following heuristic solution.

Heuristic Solution

For comparison purposes, we also define a simple heuristic solution that iteratively finds a suitable location in the path in front of a human to place a robot. Given the human's current location and direction of motion, the likely path $P \subseteq G$ is the set of nodes that the human is likely going to traverse while walking as straight as possible. Alg. 2 gives the precise methodology for computing P, as illustrated in Fig. 3a.

Once the likely path P is available, the node n in P which is closest to the goal g is selected such that:

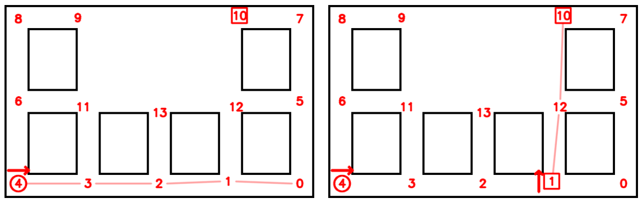
$$n = \arg\min_{p \in P}(\mathit{pathDist}(p, g, G))$$

A robot is then placed at node n to guide the human to the goal through the shortest path, as shown in Fig. 3b.

A robot is decommissioned when it is no longer required. This situation can happen if the human successfully reaches the robot's position and moves forward, or if the human moves such that the placed robot is no longer visible. Once a robot is decommissioned, the algorithm is re-run to compute the next placement for a robot if robots are available.

Algorithm 2 Heuristic for computing the likely path P

```
 1: $curNode \leftarrow$ Human's current location (graph node)
 2: $curDir \leftarrow$ Human's direction of movement (radians)
 3: $P \leftarrow \emptyset$
 4: while $true$ do
 5:     $P.insert(curNode)$
 6:     $A \leftarrow adjacentNodes(curNode)$
 7:     $diff \leftarrow Map()$, $dir \leftarrow Map()$
 8:     for all $a \in A$ do
 9:         $dir[a] \leftarrow nodeAngle(curNode, a)$
10:         $diff[a] \leftarrow absAngleDiff(curDir, dir[a])$
11:     end for
12:     $nextNode \leftarrow \arg\min(diff)$
13:     if $diff[nextNode] > \pi/4$ then
14:         break
15:     end if
16:     $curDir \leftarrow dir[nextNode]$
17:     $curNode \leftarrow nextNode$
18: end while
```



(a) Likely Straight Path P (b) Robot Placement

Figure 3: Fig. 3a shows the likely path P that the human is expected to follow starting at node 4 and pointing rightwards. With the goal at node 10, Fig. 3b shows that node 1 in p is closest to the goal. A robot is then placed at node 1 guiding the human in the direction of the shortest path towards the goal.

Simulated Human Agent

The first experiment compares the performance of both solutions through an agent exactly following the hand-coded human motion model described in this paper. Since the same model was used while generating the VI solution, the policy computed by the VI solution is optimal for this agent.

All evaluations in this experiment are made on the three maps shown in Fig. 4. The first map is a grid-like environment, and where 2 or 3 robots should be sufficient to guide the human to the goal. The second map introduces barricades in this grid-like environment, thereby inducing a need for more robots. The third and final map introduces some open spaces into the environment with the goal of creating interesting transition dynamics as the human moves around.

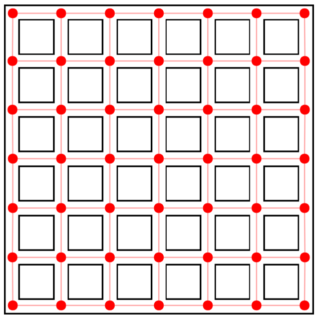
The experiment was run by first generating 1000 different problem instances on each map. Each problem was set up with randomly selected start and goal locations as follows:

$$\langle \mathit{startNode}, \mathit{startDir}, \mathit{goalNode}\rangle, \mathit{startNode} \neq \mathit{goalNode}$$

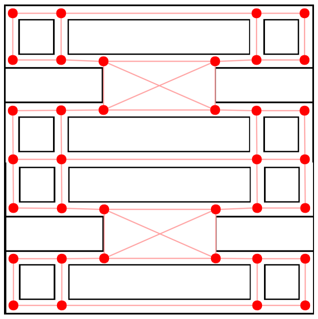
Each problem instance was further split into 5 separate sub-problems by setting the maximum number of robot placements (*maxRobots*) to every value $\in \{1, \ldots, 5\}$, and every sub-problem was evaluated once through the agent using both the VI and heuristic solutions. *startNode*, *startDir* and *maxRobots* were used to compute the system's state s at the start of each episode.

Since the distance between start and goal in each problem is variable, it is necessary to normalize the agent's distance in each problem before this information can be aggregated. We normalize the distance the agent travels in each problem by the shortest distance between *startNode* and *goalNode* for that problem (i.e. $\mathit{pathDist}(\mathit{startNode}, \mathit{goalNode}, G)$). These normalized distances have been averaged across the 1000 instances to produce the plots shown in Fig. 4.

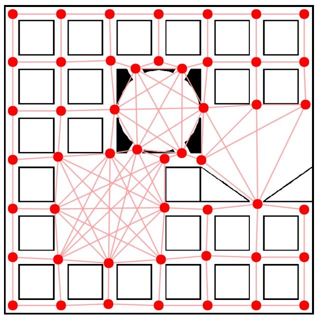
Results from map 1 in Fig. 4d show that as expected, both the VI and the heuristic solution reach a normalized distance close to optimal once three robots are available for placement. In a grid-like environment 2 robots are sufficient to guide the person to the goal node, and a final robot is placed at the goal node itself by both approaches to entice the human to walk towards it. The difference in performance between the approaches when only a single robot is available



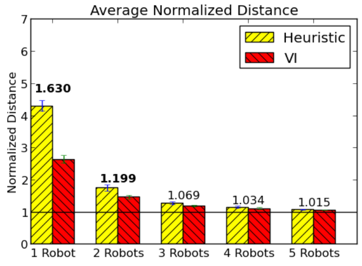
(a) Map 1



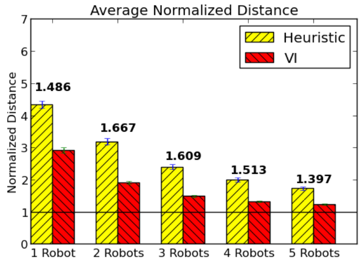
(b) Map 2



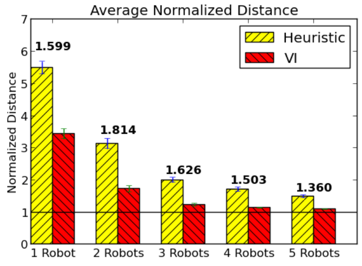
(c) Map 3



(d) Map 1 Results



(e) Map 2 Results

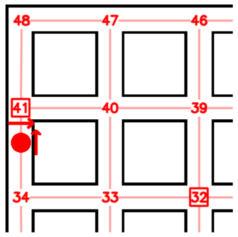


(f) Map 3 Results

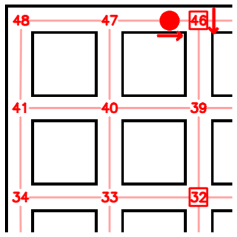
Figure 4: Fig. 4 shows different evaluation maps (each of size $48.8m \times 46.8m$) and the average performance of the heuristic and VI approaches on these maps with standard error bounds. The value above each set of bars indicates the ratio between the heuristic and VI solutions, and is bold when the difference between the two approaches is significant using a two sample t-test with 95% confidence.

is because the VI solution is much better at reasoning about human movement, and an example is illustrated in Fig. 5.

When barricades are introduced in Map 2, the VI approach significantly outperforms the heuristic solution which runs out of robots to place. On average, the heuristic solution would require 9 robots to achieve the same expected distance as that of the VI solution with 5 robots. With an insufficient number of robots being available, it becomes necessary to reason about human movement so that robots are only placed at strategic locations. With large open spaces in Map 3, the VI approach is much better at realizing that simply placing a robot with a directional arrow when a person enters an open space is insufficient to guide the person towards the intended exit from the space. Consequently, the VI approach typically also places a robot at this exit to ensure that the human walks towards it.



(a) Heuristic - Map 1



(b) VI - Map 1

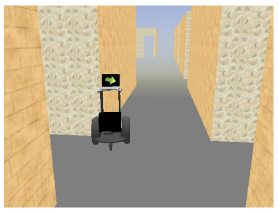
Figure 5: Fig. 5a shows that in the situation where the goal is at 32 and the human is currently moving upwards at node 41, the heuristic solution immediately directs the human to the goal by placing the last remaining robot at 41 itself. In a large majority of cases, the human misses the goal. On the other hand, VI reasons about human movement and waits for the person to naturally turn right before directing him to the goal at node 46.

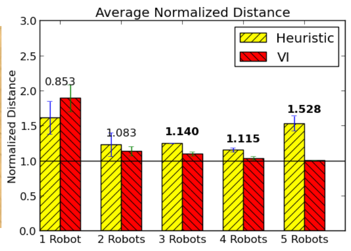
Human experiments

The second experiment evaluates both the VI and heuristic solution with real humans inside a 3D simulation environment. We set up the simulation using Gazebo (Koenig and Howard 2004), and constructed a 3D maze based on Map 3 (Fig. 4c). The simulator allows a human to control an avatar inside the environment, and provides sensory information through a simulated camera as shown in Fig. 6a. Human position inside the environment is mapped to a graph node to compute the system state inside the MDP, and the simulator queries the policy being evaluated for the action at that state to move robots into their assigned locations as requested. In this experiment, since real humans control what happens in the environment, their behavior may significantly differ from the hand-coded human motion model, and the VI solution is no longer guaranteed to be optimal.

This experiment required some small changes to the human motion model and the MDP, as explained below:

1. Visibility in the simulator is limited up to 30 meters, and *isVisible*(u,v) $\forall\, u, v \in G$ was modified appropriately.
2. There is a slight delay between when the state of the MDP is updated and when the robots are moved into place. This duration gives the possibility for the human to collide with or bypass a robot being placed at the current node. Consequently, placing robots at the current node (*placeRobot*(*s.curNode*)) was disabled.
3. The humans are told to look for a red ball in the environment. Since the ball is visible in the simulator, it induces an effect in the human motion model similar to that produced by a visible robot in the distance. Case 1 in the hand-coded human motion model was modified accordingly to also account for the goal being visible.


(a) Simulator Interface



(b) Distance - Experiment 2

Figure 6: Fig. 6a shows the input available for the user to move around in the environment. Fig. 6b shows the average distance traveled in each problem instance. Significant differences are in bold.

With real humans in the loop, it quickly becomes impractical to test performance across an extensive set of problems. Instead, we randomly selected 5 problem instances such that the length of the shortest path between start and goal in each instance was greater than $40m$ (the side of the square map is $46.8m$). Each problem instance was tested with a specific value of *maxRobots*, and between the 5 problems all values in $\{1, \ldots, 5\}$ were covered. Each human subject first went through 2 small tutorial problems to first get acquainted with the simulator interface. Next, either the heuristic or the VI policy was randomly selected for the system to use while guiding them over these 5 problem instances. 9 and 10 humans were guided using robot placements provided by the heuristic and VI solutions, respectively.

As before, distances traveled by the humans have been normalized against the shortest graph distance and averaged to produce the results in Fig. 6b. In all experiments, the difference between the 2 policies is either insignificant, or the VI solution significantly outperforms the heuristic. Of particular note is the case with 1 robot, where the heuristic performs better than the VI solution (not significantly). The heuristic places its single robot immediately to point the human in the general vicinity of the goal, and the human executes a search strategy in the correct region to find the goal. Since the hand-coded human motion model does not account for this strategy, the VI solution is quite conservative about placing its single robot unless it can place a robot on a location directly visible from the goal, and the human wanders aimlessly for some time. A more realistic human model may be able to improve the VI solution in such cases.

Discussion

In this paper, we have introduced the multi-robot human guidance problem, formulated the problem as an MDP using a model of human motion and solved the MDP to generate an optimal policy for placing multiple robots in an environment to efficiently guide a human. Our MDP formulation uses a topological graph representation of the environment, and we have discussed how such a representation can be constructed. Our ongoing research agenda includes extending this work by collecting data from real humans interacting with real guide robots to construct realistic models of human motion. Given such a model, solving the MDP will have the potential to produce equivalent or better solutions than any heuristic-based approach.

Acknowledgments

The authors would like to thank Samuel Barrett for many useful discussions during the seminal stages of this work. The authors would also like to thank the ROS and GAZEBO open-source communities for infrastructure used in this work.

This research has taken place in the Learning Agents Research Group (LARG) at the AI Laboratory, University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-1330072, CNS-1305287), ONR (21C184-01), and Yujin Robot. LARG research is also supported in part through the Freshman Research Initiative (FRI), College of Natural Sciences, University of Texas at Austin.

References

Aurenhammer, F. 1991. Voronoi diagrams–a survey of a fundamental geometric data structure. *Computing Surveys (CSUR)*.

Chen, D. L., and Mooney, R. J. 2011. Learning to interpret natural language navigation instructions from observations. In *National Conf. on Artificial Intelligence (AAAI)*.

Choset, H., and Burdick, J. 1995. Sensor based planning, Part I: The Generalized Voronoi Graph. In *Int'l. Conference on Robotics and Automation (ICRA)*.

Choset, H., and Nagatani, K. 2001. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *Transactions on Robotics and Automation*.

Koenig, N., and Howard, A. 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Int'l. Conf. on Intelligent Robots and Systems (IROS)*.

Konolige, K.; Marder-Eppstein, E.; and Marthi, B. 2011. Navigation in hybrid metric-topological maps. In *Int'l. Conf. on Robotics and Automation (ICRA)*.

Lacey, G., and Dawson-Howe, K. M. 1998. The application of robotics to a mobility aid for the elderly blind. *Robotics and Autonomous Systems*.

Montemerlo, M.; Pineau, J.; Roy, N.; et al. 2002. Experiences with a mobile robotic guide for the elderly. In *Innovative Applications of Artificial Intelligence (AAAI/IAAI)*.

Philippsen, R., and Siegwart, R. 2003. Smooth and efficient obstacle avoidance for a tour guide robot. In *International Conference on Robotics and Automation (ICRA)*.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; et al. 2009. Ros: an open-source robot operating system. In *Workshop on Open Source Software in Robotics at ICRA*.

Rogers, Y.; Hazlewood, W. R.; Marshall, P.; et al. 2010. Ambient influence: Can twinkly lights lure and abstract representations trigger behavioral change? In *International conference on Ubiquitous computing (UBICOMP)*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*. Cambridge Univ Press.

Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M. R.; Banerjee, A. G.; et al. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *National Conf. on Artificial Intelligence (AAAI)*.

Thrun, S., and Bücken, A. 1996. Integrating grid-based and topological maps for mobile robot navigation. In *National Conference on Artificial Intelligence (AAAI)*.

Thrun, S.; Bennewitz, M.; Burgard, W.; Cremers, A. B.; Dellaert, F.; Fox, D.; et al. 1999. MINERVA: A second-generation museum tour-guide robot. In *International Conference on Robotics and Automation (ICRA)*.