A Graph-Search based Navigation Algorithm for Traversing A Potentially Hazardous Area with Disambiguation

Xugang Ye, Johns Hopkins University, USA
Carey E. Priebe, Johns Hopkins University, USA

ABSTRACT

The authors consider the problem of navigating an agent to safely and swiftly traverse a two dimensional terrain populated with possible hazards. Each potential hazard is marked with a probabilistic estimate of whether it is indeed true. In proximity to any of these potential hazards, the agent is able to disambiguate, at a cost, its true-false status. The method presented in this paper is to discretize the terrain using a two dimensional grid with 8-adjacency and approximately solve the problem by dynamically searching for shortest paths using the A* algorithm in the positively weighted grid graph with changing weight function.

Keywords: A* algorithm, Disambiguation, Mark Information, Path Planning, Shortest Path

INTRODUCTION

The problem of navigating an agent with disambiguation capability to safely and swiftly traverse a potentially hazardous area is currently receiving considerable attention due to its application to minefield. The operational concepts of minefield reconnaissance via an airborne payload and minefield path planning given the detection data were first introduced in Witherspoon et al. (1995). Most of the early research efforts were put on the mine detection part. A thorough discussion of the mine detection can be found in Holmes et al. (1995). As the output of the mine detection, the detection map consists of a collection of objects with each marked with the estimate of probability of being a true hazard.

The focus of our work presented in this paper is the navigation part. That is to navigate the agent given the detection map a priori. In this problem, it is assumed that in proximity to any of the potential hazards the agent is able to disambiguate, at a cost, its true-false status. Since the agent can disambiguate, the safety is not a concern. Therefore, the navigation objective is to

DOI: 10.4018/joris.2010070102
minimize total traversal cost, which consists of the traveling cost and the disambiguation cost.

Due to the randomness sources like the locations, the true-false status, and the detection markers, the agent’s traversal is also random. If a disambiguation is performed, the traversal is named random disambiguation walk (RDW), which is a very novel object that appears in very recent probabilistic path planning literature.

The object random disambiguation walk first appeared in Priebe et al. (2005). However, in this original study, the object was named as random disambiguation path (RDP). We rename the object as random disambiguation walk because the agent may revisit a location. If the world is a (directed) graph so that the agent only travels along the arcs, then the agent may pass some node or arc more than once. In this discrete case, the agent’s final trajectory is precisely a walk, which is defined in the most popular graph theory books (e.g., West, 2000).

Despite the name of the object, an important analytical result in Priebe et al. (2005) is that there exists a positive probability that there will be a RDW that strictly reduces the expected traversal cost compared with any deterministic shortest path that circumvents any risk. This result suggests that any relevant navigation algorithm should be able to exploit the potential benefit of disambiguation(s). Both the work in Priebe et al. (2005) and the follow-up by Fishkind et al. (2007) explored the navigation methods that yield a disambiguation-included traversal with small expected cost. However, a naïve assumption in both works is that any marker is the probability of being true hazard. The reason for this assumption is that both aimed to formulate the problem as a minor modification of the Stochastic Obstacle Scene Problem (SOSP) of Papadimitriou & Yannakakis (1991) since SOSP had been studied. A discrete version of SOSP is well known as the Canadian Traveler Problem (CTP). In CTP, the goal is to minimize the expected cost of traveling from a starting node to a destination node in a finite directed graph in which arcs are marked with their respective probabilities of being traversable or nontraversable, and each arc’s actual nature can be discovered only when encountered. Also, finding the nature of an arc leads to a cost, which has the same meaning as what we mentioned as disambiguation cost.

CTP is also a special case of the Stochastic Shortest Paths with Recourse (SPR) problem of Andreattta & Romeo (1988), who presented a stochastic dynamic programming formulation for SPR and noted its intractability. Another stochastic dynamic programming formulation for SPR was presented in Polychronopoulos & Tsitsiklis (1996), in which several variants were shown to be intractable. In Provan (2003), the SPR problem was proven to be intractable even if the underlying graph is directed and acyclic.

Although there are heuristics suggested for CTP and some versions of SPR (Blei & Kaelbling, 1991; Polychronopoulos & Tsitsiklis, 1996; Baglietto et al., 2003; Ferguson et al., 2004). They are not applicable to the RDW problems of Priebe et al. (2005) and Fishkind et al. (2007) either because of the different problem settings or because of an assumption made on the underlying graph. The assumption is the independence of the arc probabilities. In fact, the RDW problem in Fishkind et al. (2007), when discretized, reduces to a CTP with dependent arc probabilities. In our RDW problem, we even relax the assumption that any marker is the probability of being true hazard and more realistically model the markers as the realizations of the underlying random variables.

The up-to-date framework for solving a continuous RDW problem was summarized in Fishkind et al. (2007) as three steps. The first step is to discretize the world by constructing a directed graph. In Fishkind et al. (2007), the tangent arc graph (TAG) was built for a RDW setting that contains $m$ disk-shape regions with centers at the detection locations and radii as the effectiveness ranges. The TAG is a precise map representation. The downside is that constructing a TAG is computationally demanding when the number of detections is large (e.g., $m \geq 100$). The second step is to assign weights to the arcs of the graph. This step is the most important since the weight function largely determines the quality of the final traversal. This step actually reflects how a navigation algorithm uses the markers. The third step is to invoke an efficient shortest path algorithm to compute a shortest path from where the agent is to the target node in the graph. In Fishkind et al. (2007), the Dijkstra algorithm with binary heap implementation (Ahuja et al., 1993) was used. In this step, the efficiency of...
implementation also depends on the data structure rendered in the first step. Functionally, completion of the third step specifies the plan of the agent’s next move. If the planned next move is risk-free, then the agent moves on; otherwise, the agent disambiguates. The disambiguation result(s) will be incorporated into the new weight function and then both the second and third steps will be repeated.

Within the same framework, the work of this paper aims to develop an efficient approximate navigation algorithm for a RDW problem formulated in next section. As mentioned, a new assumption made on the markers constitutes an important distinct feature of this work. The motivation for this new assumption comes from the fact that in reality, the markers are usually obtained from sensor. Another distinct feature of this work is a new weight function. As an effort of overcoming the shortcoming of the weight function in Fishkind et al. (2007) that it does not contain the disambiguation cost, we design a weight function that not only considers the Euclidean distance and the traversability uncertainty but also considers the disambiguation cost.

The rest of this paper is organized as follows. We first present our formulation of the RDW problem. We then present our method and some experimental results from the computer simulations. We also present some results of processing a set of real-world minefield data using the navigation algorithm we developed. Finally, we give some conclusions of this study and some suggestions for future work.

**THE FORMULATION**

Suppose a point-size agent needs to travel from a starting location \( s \in \mathbb{R}^2 \) to a target location \( t \in \mathbb{R}^2 \). Suppose the agent faces \( m \in \mathbb{Z}^+ \) detected risks \( d_1, d_2, \cdots, d_m \in S \subset \mathbb{R}^2 \), with each \( d_i \) representing the center of either a true hazard or a false hazard and \( S \) denoting a bounded area. For \( i = 1, 2, \cdots, m \), let \( X_i \in \{0, 1\} \) be the indicator of \( d_i \), that is, \( X_i = 1 \) if \( d_i \) is a true hazard; \( X_i = 0 \) otherwise. For \( i = 1, 2, \cdots, m \), let \( D_i \) be a disk-shape region that is centered at \( d_i \) and has a radius \( r_i > 0 \). For each \( i \), \( D_i \) can be viewed as the range of effectiveness of \( d_i \) if \( X_i = 1 \). We assume that there are no more potential hazards except for \( d_1, d_2, \cdots, d_m \).

Suppose for each \( d_i \), there is an independent random marker \( 0 < Y_i < 1 \) such that \( Y_i \big|_{X_i=0} \sim F_0(y) \) and \( Y_i \big|_{X_i=1} \sim F_1(y) \), where \( F_0: [0, 1] \rightarrow [0, 1] \) and \( F_1: [0, 1] \rightarrow [0, 1] \) are two distribution functions such that \( F_0(y) \geq F_1(y) \) for any \( 0 \leq y \leq 1 \). Any realization \( y_1, y_2, \cdots, y_m \) of \( Y_1, Y_2, \cdots, Y_m \) can be viewed as a probabilistic estimate of the nontraversability of those risk disks \( D_1, D_2, \cdots, D_m \). It is a pre-knowledge of the world.

A disambiguation of some \( d_i \) at a cost \( c_i > 0 \) happens when the agent is right outside \( D_i \) but about to enter \( D_i \) and \( d_i \) has not been disambiguated yet. If the disambiguation discloses that \( X_i = 1 \), then \( d_i \) is confirmed to be a true hazard, in this case, the region covered by \( D_i \) should be forbidden, that is, the marker \( Y_i \) should be updated into 1. If the disambiguation discloses that \( X_i = 0 \), then \( d_i \) should be removed (and hence its marker \( Y_i \) should be updated into 0), but the region covered by \( D_i \) may still be questionable since \( D_i \) may intersect some other risk disks that have not been disambiguated yet. During the travel, the agent dynamically collects the new information through local disambiguations and updates its knowledge of the world.

The cost for the agent to go from \( s \) to \( t \) has two-fold meanings. One is the actual distance the agent travels; the other is the cost paid for disambiguations. A realistic explanation of the disambiguation cost is that the agent spends time when it disambiguates. To unify the cost measures, we define the disambiguation cost as the additional “distance” that the agent travels. Hence the total cost for the agent to go from \( s \) to \( t \) is measured by a total distance with part as the actual length of the traversal and the remaining as the measure of the total disambiguation cost. The agent is supposed to go from \( s \) to \( t \) with a total cost as small as possible.

Note that the agent can always circumvent all the risk disks and travel along a deterministic \( s-t \) shortest path that does not penetrate any risk disk even if it is indeed safe. The central issue is how to exploit the agent’s disambiguation capability to gain a better performance than just using the deterministic shortest path. It’s not hard to imagine the cases in which a deterministic
shortest path could be very long. For example, there might be too many risk disks that cover a very large area so that to detour all the potential barriers will be very costly.

The NAVIGATION ALGORITHM

The preprocessing step is to discretize $\mathbb{R}^2$ to obtain a directed graph representation. In this paper, we adopt a grid with 8-adjacency as shown in the following Figure 1, in which a node $v_{ij}$ has 8 outgoing neighbors $v_{i+1,j}$, $v_{i+1,j-1}$, $v_{i-1,j}$, $v_{i-1,j+1}$, $v_{i,j+1}$, and $v_{i,j}$, and $\delta > 0$ represents the resolution.

Compared with the TAG, which is a topological superimposition of all the visibility graph generated by $s$, $t$, and $D$ over all $D \subseteq \{D_1, D_2, \ldots, D_m \}$, the 8-adjacency grid is not a precise map representation. However, since according to Fishkind et al. (2007), the time complexity of setting up a TAG is $O(m^3 \log m)$, where $O$ stands for the “big O” notation, as $m$ is large, building a TAG representation will be inefficient. One advantage of the grid representation is that the time complexity of constructing a directed grid graph with weighted arcs is not so sensitive to $m$. This can be reflected from the assembling of the weight function that we introduce the next.

Let $a = (u, v)$ denote the arc from node $u$ to node $v$ in our grid graph, say $G$. Let $W(a)$ denote the weight of arc $a$. $W(a)$ contains three types of information: the marker, the length, and the disambiguation cost. The marker measures the uncertainty of the arc. The length measures the cost of traversing the arc. The disambiguation cost measures how much to pay to find out whether the arc is traversable or not. If the arc is already confirmed to be nontraversable, then its disambiguation cost becomes zero but its length should be infinity; otherwise, it has non-zero disambiguation cost and its length might be just its usual Euclidean length. We can see that the information on both the length and the disambiguation cost is affected by the marker. Hence, we first derive the markers of the arcs.

We first define the extended markers of $d_1, d_2, \ldots, d_m$ as

$$Y_i^+ = \begin{cases} 
0 & \text{if } d_i \text{ has been disambiguated as } X_i = 0; \\
1 & \text{if } d_i \text{ has been disambiguated as } X_i = 1; \\
Y_i & \text{if } d_i \text{ has not been disambiguated.}
\end{cases}$$

for $i = 1, 2, \ldots, m$. Thus, the real-time knowledge of the true-false status of $d_1, d_2, \ldots, d_m$ can be represented as \{ $Y_1^+, Y_2^+, \ldots, Y_m^+$ \}, which can also

Figure 1. The grid representation, with 8-adjacency
be viewed as the in-situ markers.

We then define $Y_I$: the derived marker of the region $R_I$ that is covered by $\bigcap_{i \in I} D_i$ for any $I \subseteq \{1, 2, \ldots, m\}$ as

$$Y_I = 1 - \prod_{i \in I} (1 - Y_i^+)$$  \hspace{1cm} (2)

with the convention that $\prod_{i \in I} (1 - Y_i^+) = 1$ if $I$ is empty. In fact, (2) is to mimic the probability that $R_I$ is forbidden. The heuristic idea is that if $Y_i^+$ is viewed as the independent probability that $D_i$ is forbidden, then the probability that $R_I = \bigcap_{i \in I} D_i$ is forbidden is expressed as (2). Note that the independence of $Y_1$, $Y_2$, $\ldots$, $Y_m$ is necessary.

For arc $a = (u, v)$, let $I_u = \{i \mid u \text{ is covered by } D_i\}$ and $I_v = \{i \mid v \text{ is covered by } D_i\}$. We claim that the extended marker

$$Y^*(a) = Y_{I_u,I_v}$$  \hspace{1cm} (3)

is the in-situ marker of arc $a$. To justify this claim, we define $I_{da} = \{i \mid d_i \text{ has not been disambiguated}\}$ and notice that

$$Y_{I_u,I_{da}} = 1 - \left[\prod_{i \in I_u \cap I_{da}} (1 - Y_i^+)\right] \cdot \left[\prod_{i \in I_v \cap I_{da}} (1 - Y_i^+)\right].$$  \hspace{1cm} (4)

The justification follows upon arguing three cases. First, $Y_{I_u,I_{da}} = 1$ if and only if $\prod_{i \in I_u \cap I_{da}} (1 - Y_i^+) = 0$, which is equivalent to the existence of a $j \in (I_u \setminus I_d) \cap I_d$ such that $Y_j^+ = 1$, i.e. $d_j$ has been disambiguated and $X_j = 1$, hence $Y_{I_u,I_d} = 1$ if and only if $a$ is confirmed to be nontraversable. Second, $Y_{I_u,I_{da}} = 0$ if and only if $\prod_{i \in I_u \cap I_{da}} (1 - Y_i^+) = 1$ and $(I_u \setminus I_d) \cap I_d$ is empty, hence $Y_{I_u,I_d} = 0$ if and only if $a$ is confirmed to be traversable. Third, $0 < Y_{I_u,I_{da}} < 1$ if and only if $\prod_{i \in I_u \cap I_{da}} (1 - Y_i^+) = 1$ and $(I_u \setminus I_d) \cap I_d$ is nonempty, hence $0 < Y_{I_u,I_d} < 1$ if and only if $a$ is potentially traversable but still uncertain.

Based on (3), we now define the extended length function of arc $a$ as:

$$l^+(a) = \begin{cases} \ell(a) & \text{if } 0 \leq Y^*(a) < 1; \\ +\infty & \text{if } Y^*(a) = 1, \end{cases}$$  \hspace{1cm} (5)

where $\ell(a)$ denotes the Euclidean length of $a$. Note that $\ell(a) = \delta$ or $\sqrt{2}\delta$ in our 8-adjacency grid graph $G$ with resolution $\delta$. This extended length function says that if $a$ is confirmed to be nontraversable, its length should be infinity; otherwise, its length is treated as the usual Euclidean length.

Also based on (3), we define the extended disambiguation cost of arc $a = (u, v)$ as

$$c^+(a) = \sum_{i \in (I_u \setminus I_d) \cap I_d} c_i \quad \text{if } 0 < Y^*(a) < 1;$$  \hspace{1cm} (6)

$$0 \quad \text{otherwise.}$$

That is, $a$ needs to be disambiguated if and only if it is nondeterministic, and furthermore, disambiguating $a$ means disambiguating all risk disks that cover $v$ but not $u$ and that have not been disambiguated yet.

With $Y^*(a)$, $l^+(a)$, and $c^+(a)$ available, we are now ready to assemble the weight $W(a)$. Our guideline is $W(a) = \ell(a)$ when $Y^*(a) = 0$; $W(a) = +\infty$ when $Y^*(a) = 1$; and $l(a) + c^+(a) < W(u, v) < +\infty$ when $0 < Y^*(a) < 1$. That is, if $a$ is confirmed to be traversable, then its weight should be its usual Euclidean length; if $a$ is confirmed to be nontraversable, its weight should be infinity; if $a$ is uncertain, its weight should be greater than the total cost of traversing $a$. The last case says we place penalties on all the nondeterministic arcs. By this guideline and inspired by Aksakalli et al. (2009), our weight function is

$$W(a) = l^+(a) + \frac{c^+(a)}{1 - Y^+(a)}$$  \hspace{1cm} (7)

for all $a$ of $G$.

We now give the description of our
navigation algorithm under discrete setting:

**The Navigation Algorithm**

Steps:

Given the grid graph $G$, the starting node $s$, the target node $t$, the risk disks $D_1, D_2, \ldots, D_m$, and the associated markers $Y_1, Y_2, \ldots, Y_m$ of the risk centers $d_1, d_2, \ldots, d_m$.

Step 1. For $i = 1, 2, \ldots, m$, set $Y_i = Y$. Initialize the weight function $W$ of $G$ according to (7). Set the agent’s location $v = s$.

Step 2. Find a shortest path from $v$ to $t$ in $G$ relative to $W$. Move along this path (hence update $v$) until the agent reaches $t$ or encounters an arc $a = (v, v')$ such that $Y'(a) > 0$. In former case, stop; otherwise, continue.

Step 3. Disambiguate $d_i$ for all $i \in (I_r \cap I_t) \cap I_2$. Update $Y_i$ for $i = 1, 2, \ldots, m$ according to (1). Update the weight function $W$ of $G$ according to (7). Go to Step 2.

Note that the updates on both the extended markers $Y_1^+, Y_2^+, \ldots, Y_m^+$ and the weight function $W$ are local since the disambiguations happen locally. There is a shortest path subproblem that calls for an efficient shortest path algorithm. Due to the fact that the grid graph $G$ is embedded in $\mathbb{R}^2$. The A* algorithm (Hart et al., 1968; Hart et al., 1972; Nilsson, 1980; Pearl, 1984) is an ideal choice for the shortest path subproblem. We employ a best-first search (Pearl, 1984) version of the A* algorithm that can be applied to our grid graph $G$. For convenience, let $G = (V, A, W)$, where $V$ is the set of nodes, $A$ is the set of arcs, $W: A \rightarrow \mathbb{R}^+$ is the weight function. A good merit of $G$ is that it is equipped with a natural consistent heuristic function $h: V \rightarrow R$ relative to the target node $t$. The consistency means $h(v) \geq 0$ for all $v \in V$, $h(t) = 0$, and $W(a) + h(v) \geq h(u)$ for all $a = (u, v) \in A$. For our grid graph $G$, the form of $h$ can be expressed as

$$h(v) = d^{Eu}(v, t), \tag{8}$$

for all $v \in V$, where $d^{Eu}(v, t)$ stands for the Euclidean distance between node $v$ and node $t$.

The A* algorithm is stated as follows:

**The A* Algorithm**

Notations:

$h$: heuristic function  
$O$: Open list  
$E$: Closed list  
$d$: distance label  
$f$: node selection key  
$\text{pred}$: predecessor

Steps:

Given the grid graph $G$, the starting node $s$, the target node $t$, and the heuristic function $h$.

Step 1. Set $O = \{s\}$, $d(s) = 0$, and $E = \phi$.

Step 2. If $O = \phi$ and $t \notin E$, then stop (there is no $s$-$t$ path); otherwise, continue.

Step 3. Find $u = \arg \min_{v \in O} f(v) = d(v) + h(v)$. Set $O = O \setminus \{u\}$ and $E = E \cup \{u\}$. If $t \in E$, then stop (a shortest $s$-$t$ path is found); otherwise, continue.

Step 4. For each node $v \in V$ such that $a = (u, v) \in A$ and $v \notin E$,

- if $v \notin O$, then set $O = O \cup \{v\}$, $d(v) = d(u) + W(a)$, and $\text{pred}(v) = u$;
- otherwise, if $d(v) > d(u) + W(a)$, then set $d(v) = d(u) + W(a)$ and $\text{pred}(v) = u$.

Go to Step 2.

According to Hart et al. (1968), Hart et al. (1972), Nilsson (1980), and Pearl (1984), this A* algorithm can find a shortest $s$-$t$ path in $G$ as long as there exists an $s$-$t$ path in $G$. Since there always exists a deterministic $s$-$t$ shortest path that circumvents all the risk disks, this A* algorithm can always return a shortest path plan from $s$ to $t$.

When $G$ is a large but sparse finite graph in the sense that $b << |V|$, where $b$ is a constant integer such that $0 < b < |V|$ and $|\{v \mid (u, v) \in A\}| \leq b$ for all $u \in V$ (e.g., $b = 8$ and $|V| = 10^4$), then the A* algorithm can be efficiently implemented by storing $G$ in the form of adjacency lists (Cormen et al., 2001) and maintaining the Open...
list $O$ as a binary heap, or pairing heap, or Fibonacci heap (Ahuja et al., 1993; Cormen et al., 2001). For example, using a binary heap to maintain $O$ yields a total number of operations bounded by $O(|E_{final}| \cdot b \cdot \log_2 |O_{max}|)$ upon successful termination, where $E_{final}$ is the final Closed list, $O_{max}$ is the Open list of the largest size during the iterations, and “$O$” stands for the “big O” notation.

If we consider two consistent heuristic functions $h_1$ and $h_2$, relative to $t$, such that $h_1(v) > h_2(v)$ for all $v \in V \setminus \{t\}$, then A dominance theorem (Hart et al., 1968; Nilsson, 1980; Pearl, 1984) on the A* algorithm says that $E_{final,1} \subseteq E_{final,2}$, where $E_{final,i}$ denotes the final Closed list corresponding to $h_i$, $i = 1, 2$. Note that replacing the right hand side of (8) with zero reduces the A* algorithm to the Dijkstra’s algorithm. Since according to (Pearl, 1984), for the $R^2$ embedded 8-adjacency grid graph $G$, $|E_{final}| = O(|O_{max}|^2)$, we have that running the A* algorithm in $G$ can be more efficient than running the Dijkstra’s algorithm.

**EXPERIMENTAL RESULTS**

In our experiments, we simulate $m = 100$ risk disks within $[0, 100] \times [0, 100]$. The locations of $d_1$, $d_2$, $\cdots$, $d_{100}$ are independently and uniformly drawn from $[10, 90] \times [10, 90]$. For $i = 1, 2, \cdots, 100$, we set $r_i = 4.5$. Based on this risk disk size, we choose $\delta = 1$, which means the resolution of the grid is $100 \times 100$. We let $s = (0, 0)$ be the starting location and $t = (100, 100)$ be the target location. An example of this setting is illustrated in the following Figure 2.

We simulate the markers of $d_1$, $d_2$, $\cdots$, $d_{100}$ from Beta distribution, which has the probability density function

$$f_{Beta}(x; \xi, \eta) = \left\{ \begin{array}{ll} \frac{\Gamma(\xi+\eta)}{\Gamma(\xi)\Gamma(\eta)} x^{\xi-1}(1-x)^{\eta-1} & \text{if } 0 < x < 1; \\ 0 & \text{otherwise,} \end{array} \right.$$

(9)

where $\xi > 0$ and $\eta > 0$ are the two parameters, and $\Gamma$ is Gamma function. In our experiments, for $i = 1, 2, \cdots, 100$, independently, we draw $Y_i \mid \xi, \eta = 0 \sim f_{Beta}(y; 3.5 - \lambda, 3.5 + \lambda)$ and draw $Y_i \mid \xi, \eta = 1 \sim f_{Beta}(y; 3.5 + \lambda, 3.5 - \lambda)$, where $0 < \lambda < 3.5$ is the

![Simulated Risk Centers and Disks](image-url)

Figure 2. A collection of $m = 100$ detected risks. The little squares are the simulated risk centers, the circles are the boundaries of the simulated risk disks.
uniparameter. In fact, such parameter setting renders us a sensor \((\text{Beta}(3.5 - \lambda, 3.5 + \lambda), \text{Beta}(3.5 + \lambda, 3.5 - \lambda))\) for each \(\lambda\). And the larger the value of \(\lambda\), the better the sensor. As \(\lambda \to 0\), the sensor approaches “useless”; as \(\lambda \to 3.5\), the sensor approaches “perfect”.

For \(i = 1, 2, \cdots, 100\), we use a constant disambiguation cost \(C_d > 0\) of \(d_i\). In our experiments, we use \(C_d = 2.25\). It should be noted that the larger the value of \(C_d\), the more likely it is that the planned path does not traverse any risk disks. We choose \(C_d = 2.25\) in order to observe the cases in which disambiguation does happen.

Finally for \(i = 1, 2, \cdots, 100\), we independently draw \(X_i \sim \text{Bernoulli}(0.6)\).

In our experiments, we coded both our navigation algorithm and the embedded A* algorithm with Matlab 7.1 and tested the code on a PC with Intel dual core CPU T2050 at 1.6 GHz and 2.0 G RAM. A sample simulated traversal is displayed in Figure 3.

Figure 3 visualizes a sample run of simulating a point agent traveling from \(s = (0, 0)\) to \(t = (100, 100)\). In this run, we first draw the locations of \(d_1, d_2, \cdots, d_{100}\), then draw \(X_1, X_2, \cdots, X_{100}\), and then draw \(Y_1, Y_2, \cdots, Y_{100}\). The marked map on the left is color-coded so that a large marker of a region produces deep pink (probable danger) and a small marker of a region produces light pink (probable safety). The right subfigure shows that 2 disambiguations of false alarms happen. The first disambiguation happens when the agent is right outside the first risk disk (the lower one) and about to enter. Since it turns out to be safe, a new plan directs the agent to enter this disk. The second disambiguation happens when the agent is still inside the lower disk and right outside the upper risk disk, but about to enter. Since it again turns out to be safe, a new plan directs the agent to enter this upper disk. Since there is no new information injection after the second disambiguation, the agent simply follows the plan made after the second disambiguation and finally reaches \(t\). This sample run actually shows a good traversal.

A very interesting question is how the quality of the sensor affects the quality of the traversal. More concretely, how the value of \(\lambda\) affects the total cost of the traversal under our experimental setting?

To have an empirical answer, we did the
following experiment:

1. We still maintain the experimental setting as the one that generates the sample traversal displayed in Figure 3. We consider 8 values of $\lambda$: $\lambda_1 = 0.01$, $\lambda_2 = 0.5$, $\lambda_3 = 1.0$, $\lambda_4 = 1.5$, $\lambda_5 = 2.0$, $\lambda_6 = 2.5$, $\lambda_7 = 3.0$, and $\lambda_8 = 3.49$.

2. For each $i = 1, 2, \ldots, 8$, we generate 2500 sample traversals under the sensor parameter $\lambda_i$. Each sample traversal is generated in the same manner as that for generating the sample traversal displayed in Figure 3. There are totally 20000 runs and each run returns a total cost.

3. We generate 2500 sample deterministic shortest paths with each run executed as finding the shortest s-t path that circumvents $D_1, D_2, \ldots, D_{100}$ after drawing the locations of $d_1, d_2, \ldots, d_{100}$.

This experiment generated 20000 costs of nondeterministic traversals in step 2 and 2500 lengths of deterministic shortest paths in step 3. The data was processed and the results are graphically illustrated in Figure 4.

As the left plot of the Figure 4 shows, the effect of the value of $\lambda$ is monotone, that is, the greater the value of $\lambda$, the less the average cost of the nondeterministic traversals. This empirical monotonicity result indicates that the better informed, the better our navigation algorithm performs. Hence, empirically speaking, our navigation algorithm is reliable. The critical value $\lambda^* = 1.856$ says that the mark information provided by the sensor (Beta($3.5 - \lambda$, $3.5 + \lambda$), Beta($3.5 + \lambda$, $3.5 - \lambda$)) with $\lambda < \lambda^*$, on average, is not worthy of being considered when compared with the strategy of using the deterministic shortest path that detours all the risk disks.

To gain more insight, we extended the same experiment to 4 different settings of the value $C_d$. That is, $C_d = 1.0, 2.25, 4.5, 6.75$. The associated average costs are displayed in the right plot of the Figure 4. We can see that the effects of the values of $\lambda$ are monotone under different values of $C_d$. We can also see that the larger the value of $C_d$, the smaller the value of $\lambda^*$ and the more our navigation algorithm performs like the strategy of just using the deterministic shortest path that detours all the risk disks. Actually, this is in

**Figure 4.** Left plot: average costs of 2500 nondeterministic traversals under $\lambda = 0.01, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.49$ vs. average length of 2500 deterministic shortest paths. $C_d = 2.25$. The error bars corresponds to the standard deviations of the means. The intersection of two curves yields a critical value $\lambda^* = 1.856$. Right plot: average costs of 2500 nondeterministic traversals under $\lambda = 0.01, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.49$ vs. average length of 2500 deterministic shortest paths. $C_d = 1.0, 2.25, 4.5, 6.75$. The respective four critical values are $\lambda^* = 2.385, 1.856, 1.486, 1.194$. 

Copyright © 2010, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited
accordance with (7), and for enough large value of $C_d$, our navigation algorithm reduces to non-disambiguation planning. Moreover, it shows that the smaller the value of $C_d$, the more accurate sensor it needs.

Figure 5 displays a vivid example of bad traversal under $\lambda = 0.5$ vs. good traversal under $\lambda = 3.0$ for a same terrain and the same $C_d = 2.25$. In the bad case, the mark information not only leads the agent into dead ends but also results in many unnecessary disambiguations. The final trajectory is long and the total disambiguation cost is high. Obviously, this traversal is worse than the deterministic shortest path that detours all the risk disks. In contrast, the mark information provided by the sensor with $\lambda = 3.0$ yields a superior performance, which is better than the deterministic shortest path that circumvents all the risk disks.

Figure 5. An example of bad traversal under $\lambda = 0.5$ (the upper two subfigures) vs. good traversal under $\lambda = 3.0$ (the lower two subfigures) for a same terrain. Both have disambiguation cost per detection: $C_d = 2.25$. The bad traversal has total cost 242.14, traveling cost 215.14, and disambiguation cost 27 (there are totally 12 disambiguations). The good traversal has total cost 174.77, traveling cost 168.02, and disambiguation cost 6.75 (there are totally 3 disambiguations).
REAL WORLD DATA

We applied our navigation algorithm to a minefield data from the Coastal Battlefield Reconnaissance and Analysis (COBRA) Group. The data was reported in Witherspoon et al. (1995) and then referred to in Priebe et al. (1997), Olson et al. (2002), Priebe et al. (2005), and Fishkind et al. (2007). This data has 39 mine detections with x-y coordinates and associated markers listed in Table 1. The markers were generated by the post-classification rule of Olson et al., (2002). Each risk disk has radius 50. The later-found true-false status is illustrated in the upper left plot of Figure 6. The locations of the risk centers and the markers of those detections are fixed (as given in Table 1). We used the same markers of those detections are fixed (as given in Figure 6. Each risk disk has radius 50. The later-found true-false status is illustrated in the upper left plot of Figure 6.

The locations of the risk centers and the markers of those detections are fixed (as given in Table 1). We used the same s = (0, 800) and t = (0, 100) as used in both Priebe et al. (2005) and Fishkind et al. (2007). We choose δ = 10 in our discretization step. We experimented with three constant costs: C_d = 5, 50, 500. Running our navigation algorithm produced three traversals under C_d = 5, 50, 500 respectively. They are also displayed in Figure 6. We can see that when the disambiguation cost per detection increases, there will be less and less disambiguations. An enough large C_d (e.g., 500) leads to a deterministic shortest path plan with no disambiguation plan involved. All of these three traversals are optimal because setting \( Y_i \mid x_i = 0 \rightarrow 0 \) and \( Y_i \mid x_i = 1 \rightarrow 1 \) for \( i = 1, 2, \ldots, 39 \) (i.e. changing the markers to approach perfect) for our navigation algorithm yields exactly the same traversals under \( C_d = 5, 50, 500 \) respectively.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>Marker</th>
<th>x</th>
<th>y</th>
<th>Marker</th>
<th>x</th>
<th>y</th>
<th>Marker</th>
</tr>
</thead>
<tbody>
<tr>
<td>321.17</td>
<td>158.27</td>
<td>0.59017</td>
<td>-105.75</td>
<td>262.2</td>
<td>0.25748</td>
<td>95.39</td>
<td>248.12</td>
<td>0.18868</td>
</tr>
<tr>
<td>54.23</td>
<td>201.12</td>
<td>0.54178</td>
<td>185.31</td>
<td>182.18</td>
<td>0.65266</td>
<td>-78.75</td>
<td>396.14</td>
<td>0.0731</td>
</tr>
<tr>
<td>158.17</td>
<td>416.48</td>
<td>0.43525</td>
<td>116.39</td>
<td>110.84</td>
<td>0.44124</td>
<td>-245.28</td>
<td>372.05</td>
<td>0.52154</td>
</tr>
<tr>
<td>215.13</td>
<td>428.31</td>
<td>0.6189</td>
<td>-128.6</td>
<td>274.12</td>
<td>0.62001</td>
<td>-166.45</td>
<td>180.33</td>
<td>0.61082</td>
</tr>
<tr>
<td>-145.67</td>
<td>703.06</td>
<td>0.61714</td>
<td>-61.19</td>
<td>345.12</td>
<td>0.17183</td>
<td>-134.53</td>
<td>769.27</td>
<td>0.19386</td>
</tr>
<tr>
<td>-151.01</td>
<td>572.15</td>
<td>0.56076</td>
<td>-91.27</td>
<td>664.45</td>
<td>0.16675</td>
<td>-258.45</td>
<td>641.03</td>
<td>0.5657</td>
</tr>
<tr>
<td>221.12</td>
<td>557.31</td>
<td>0.64047</td>
<td>-82.87</td>
<td>248.29</td>
<td>0.58308</td>
<td>111.6</td>
<td>640.1</td>
<td>0.5629</td>
</tr>
<tr>
<td>-166.36</td>
<td>299.42</td>
<td>0.49173</td>
<td>105.47</td>
<td>509.8</td>
<td>0.85147</td>
<td>-219.32</td>
<td>313.68</td>
<td>0.57449</td>
</tr>
<tr>
<td>296.16</td>
<td>163.31</td>
<td>0.11649</td>
<td>-19.93</td>
<td>568.04</td>
<td>0.59937</td>
<td>-455.72</td>
<td>742.57</td>
<td>0.63987</td>
</tr>
<tr>
<td>163.31</td>
<td>186.14</td>
<td>0.65636</td>
<td>-310.23</td>
<td>402.92</td>
<td>0.64163</td>
<td>-157.1</td>
<td>441.96</td>
<td>0.16675</td>
</tr>
<tr>
<td>28.31</td>
<td>205.03</td>
<td>0.15269</td>
<td>-320.73</td>
<td>532.23</td>
<td>0.3092</td>
<td>-242.22</td>
<td>321.51</td>
<td>0.65428</td>
</tr>
<tr>
<td>-79.26</td>
<td>709.99</td>
<td>0.56085</td>
<td>-35.11</td>
<td>242.61</td>
<td>0.1033</td>
<td>-237.86</td>
<td>546.19</td>
<td>0.13793</td>
</tr>
<tr>
<td>100.4</td>
<td>376.47</td>
<td>0.51487</td>
<td>-169.99</td>
<td>438.9</td>
<td>0.6413</td>
<td>-269.98</td>
<td>379.65</td>
<td>0.52802</td>
</tr>
</tbody>
</table>

CONCLUSION

We studied the problem of navigating a point agent with disambiguation capability to safely and swiftly traverse a two dimensional terrain populated with possible hazards. As a distinct feature of this work, we model the markers of the potential hazards as random numbers conditionally drawn from two distribution functions \( F_0 \) and \( F_1 \). In other words, the markers are the readings of a sensor characterized by the pair \((F_0, F_1)\). Our method is to discretize the terrain using an 8-adjacency grid and approximately solve the problem by dynamically searching for shortest paths using the A* algorithm. Each shortest path plan tells the agent where to go and which to disambiguate. As another distinct feature of this work, we proposed a new weight function of the underlying graph and this weight function considers not only the length and the risk but also the disambiguation cost.

The experimental results show that our navigation algorithm is both efficient and effective. An empirical discovery is that the better informed, the less average cost our navigation algorithm will yield. Consequently, there exists a threshold for the quality of the sensor that provides the mark information. Below that threshold, the mark information is not worthy of...
being considered when compared with the strategy of just using the deterministic shortest path that circumvents all risks.

We also applied our navigation algorithm to a real-world minefield data from the COBRA group. The results are encouraging since all the three traversals we generated are optimal.

As for the directions of the future research, we have the following four immediate suggestions:

1. Three dimensional setting. This is for the consideration of some real-world application like underwater navigation.
2. Location pattern. This means the distribution pattern of the locations of the potential hazards. In our work, we assumed uniform distribution. However, it is practically important to consider other distributions. For example, a minefield may be set up with purpose other than just randomly.
3. Location uncertainty. It is of great interest to relax the assumption that the locations of the potential hazards are fixed throughout the

Figure 6. The COBRA terrain and three traversals under \(C_d = 5, 50, 500\). When \(C_d = 5\), the total cost is 715, with 3 disambiguations; when \(C_d = 50\), the total cost is 807.99 with one disambiguation; when \(C_d = 500\), the total cost is 1043.3 with no disambiguation. Total simulation run time on a PC with Intel dual core CPU T2050 at 1.6 GHz and 2.0 G RAM is 1.609 seconds, 1.906 seconds, and 4.063 seconds, respectively.
entire navigation process. In reality, the potential hazards do not have fixed locations. An example is the drifting sea mines.

4. Dependence pattern. We mean, in reality, there might be dependence relations among the markers. In minefield application, a potential mine at one location might indicate another potential mine at a different location. If there exists dependence structure in the mark information, then our heuristic idea expressed in (2) must be adjusted. Henceforth the building of the weight function will be utterly different.

ACKNOWLEDGMENTS

We thank the support from Office of Naval Research under grant N00014-04-1-0483. We thank the Center for Imaging Science of Johns Hopkins University for providing us computing resources to perform large-scale simulations. We would also like to thank the editor-in-chief of the journal and two anonymous reviewers for their valuable comments and suggestions that improved the paper significantly.

REFERENCES


strategies for computer problem solving. Addison-Wesley.


Xugang Ye obtained his Ph.D. degree from the Department of Applied Mathematics and Statistics, the Johns Hopkins University. Dr. Ye has broad research interests in Probability, Statistics, and Optimization, as well as their various real-world applications. He has presented his research in INFORMS 2008, SIAM 2009, and ISMB 2010. He was selected as the Counselman Fellow in the Johns Hopkins University and is currently selected as the Stephen Altschul Fellow at the National Center for Biotechnology Information, the National Library of Medicine, the National Institutes of Health.

Carey E. Priebe is a professor in the Department of Applied Mathematics and Statistics, the Johns Hopkins University. He obtained his Ph.D. degree in Information Technology and Computational Statistics from the George Mason University. Professor Priebe is a renowned statistician and currently is a Fellow of the American Statistical Association. He also holds joint appointments as a professor in the Department of Computer Sciences and as a senior research scientist in the Center for Imaging Science, both at the Johns Hopkins University. Professor Priebe is also a research professor in the National Security Institute at the Naval Postgraduate School, and has been named one of six inaugural National Security Science and Engineering Faculty Fellows. Professor Priebe’s research spans a broad spectrum of probability, statistics, and operations research, and has more than 100 journal publications and dozens of patents in those areas.