Visibility-Based Pursuit-Evasion with Probabilistic Evader Models

Nicholas M. Stiffler Jason M. O'Kane

***Abstract*— We propose an algorithm for a visibility-based pursuit-evasion problem in a simply-connected two-dimensional environment, in which a single pursuer has access to a probabilistic model describing how the evaders are likely to move in the environment. The application of our algorithm can be best viewed in the context of search and rescue: Although the victims (evaders) are not actively trying to escape from the robot, it is necessary to consider the task of locating the victims as a pursuit-evasion problem to obtain a firm guarantee that all of the victims are found. We present an algorithm that draws sample evader trajectories from the probabilistic model to compute a plan that lowers the Expected Time to Capture the evaders without drastically increasing the Guaranteed Time to Capture the evaders. We introduce a graph structure that takes advantage of the sampled evader trajectories to compute a path that would "see" all the evaders if they followed only those trajectories in our sampled set. We then use a previous technique to append our path with actions that provide a complete solution for the visibility-based pursuit-evasion problem. The resulting plan guarantees that all evaders are located, even if they do not obey the given probabilistic motion model. We implemented the algorithm in a simulation and provide a quantitative comparison to existing methods.**

I. INTRODUCTION

In disaster situations such as fires, earthquakes, floods, or mine collapses, search and rescue (SAR) efforts can be difficult, especially when some locations are inaccessible or dangerous for human rescuers. Autonomous search-and-rescue robots have the potential to alleviate this difficulty by reducing the need for human presence in such locations. Although substantial progress has been made on the problem of robotic SAR in recent years [1], full autonomy in such contexts remains an unsolved problem. A key component of this application is the need for robust search plans that enable robots to locate victims, who may be moving themselves, quickly and reliably.

At the heart of this task is a computational *pursuit-evasion problem*, in which a pursuer must systematically search an environment to locate one or more evaders. See Figure 1. Although the victims are unlikely to avoid detection actively, their movements might be erratic and unpredictable. As a result, a strategy that treats the victims as evaders is necessary to guarantee that they are found.

Algorithms are known to solve a wide variety of pursuit-evasion problems [6], [11], [18]. Our research builds upon those existing methods by incorporating a predictive model for the evaders' motions. Such a model might, for example, be derived from *a priori* simulations of victims' responses to various disasters [9], [15], or from other domain specific knowledge. The intuition is that, in a disaster scenario, there are likely to be some predictable patterns to the victims' movements, and that these patterns can be exploited to decrease the expected time needed to locate them. A final phase of the pursuer's strategy can then use more traditional pursuit-evasion methods to guarantee that all evaders, regardless of whether or not they obey the predictive motion model, are eventually found by the robot.

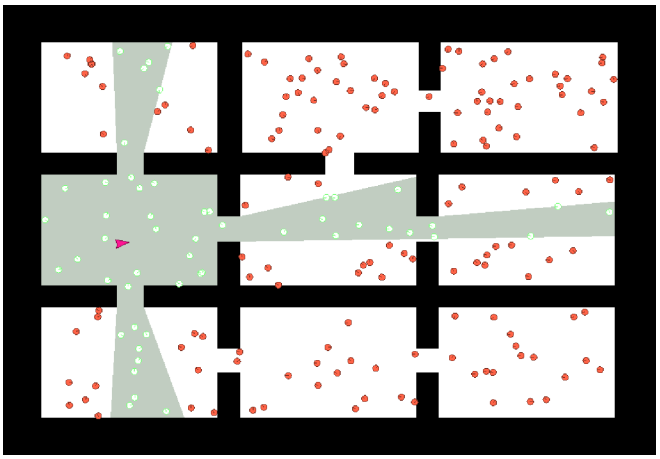


Fig. 1. A pursuer robot (triangle) navigating through an environment. The pursuer detects all of the evaders caught in its field of view (visibility polygon). The evaders that are visible to the pursuer appear as open circles, while those that are outside appear as closed circles.

The specific problem we consider is a variation on the visibility-based pursuit-evasion problem in which a robot moving through a simply-connected polygonal environment seeks to locate an unknown number of evaders, each of which may move arbitrarily fast. Guibas, et al. presented a complete algorithm for this problem [7]. However, that research also demonstrates that a solution for this problem may require the robot to revisit some parts of the environment repeatedly, to ensure that evaders have not "recontaminated" those regions. Our work can be viewed as a "rescheduling" of this strategy to visit first those regions most likely to actually contain evaders, before completing the worst-case search.

To ensure that our approach is useful in a broad variety of contexts, we do not assume any specific form for the evader model. Instead, we treat this model as "black box," from which the pursuer can generate a collection of sample evader trajectories, drawn from some probability density over all possible such trajectories. We use this sample set as an implicit representation for the underlying distribution.

The contribution of this work is an algorithm that uses

N. M. Stiffler and J. M. O'Kane are with the Department of Computer Science and Engineering, University of South Carolina, 301 Main St., Columbia, SC 29208, USA. {stifflen,jokane}@cse.sc.edu

a predictive model of evader motions to generate plans that reduce the expected time to locate all evaders in the environment, without sacrificing the worst-case guarantees provided by known algorithms. We present simulations that demonstrate that this algorithm succeeds in this goal.

The remainder of this paper is structured as follows. Section II describes related work, including the guaranteed visibility-based pursuit-evasion algorithm that our work extends. A formal problem statement appears in Section III, followed by a description of our algorithm in Section IV. Section V presents our simulations of this algorithm and a quantitative evaluation of its effectiveness. Discussion and a preview of future work conclude the paper in Section VI.

II. RELATED WORK

A. Pursuit-Evasion

Pursuit-evasion was introduced as graph problem in which multiple pursuers and an evader move from vertex to vertex within the graph until one of the pursuers lies on the same vertex as the evader [12]. The visibility-based pursuit-evasion problem proposed by Suzuki and Yamashita [17] is an extension of the watchmen route problem [3], in which the objective is to compute the shortest path that a guard should take to patrol an entire area populated with obstacles, given only a map of the area.

The visibility-based pursuit-evasion problem has been studied under various constraints. A complete algorithm was presented by LaValle et al. [11] for the case when pursuers move along the boundary of an environment and have only a single ray of visibility. A strategy for searching an environment in which one or more pursuers have a limited field-of-view was presented by Gerkey, et al. [6]. Pursuit-evasion has also been studied in the context of velocity-bounded evaders [18]. While there has been substantial research in visibility-based pursuit-evasion, to the authors' knowledge, a strategy for solving the pursuit-evasion problem that takes into account information regarding the evaders' probable trajectories has not been considered.

Others have considered the problem of rapidly searching a polygonal environment for a target [16]. That work presents a hardness result for a search problem related to ours, and a heuristic greedy search technique that performs well in practice. Our work differs because we directly consider motion models for evader movements.

The most closely related research to our problem is the complete algorithm for the visibility-based pursuit-evasion problem in which a single pursuer with an omni-directional field of view attempts to locate all of the evaders in a polygonal environment [7]. This algorithm uses "gaps", segments of the pursuer's visibility polygon that are not part of the environment, to identify regions of the environment in which evaders may be located. If it is possible for an evader to be in an area beyond the gap, it is considered to be "contaminated" and is labeled with a 1. If it is not possible for an evader to be beyond a gap, that gap is "cleared" and is labeled with a 0.

The algorithm begins by computing a cell decomposition of the environment. The cell decomposition is based on critical visibility events that occur because of inflections and bitangents of the environment boundary. This subdivides the environment into conservative regions in which the gap representation remains the same as the pursuer moves through the region. Simple rules describe how the gap labels change when the pursuer crosses into a new conservative region. These rules define a graph in which each conservative region is associated with one node for each possible assignment of the gap labels. Details appear in [7]. The pursuer maintains the value of the gap labels, which change when the pursuer moves from one conservative region to another, until all of the gap labels are "cleared" or the algorithm determines that a solution does not exist. We use and extend this strategy in our algorithm to guarantee completeness, in the sense of locating every evader in an environment.

B. Informed Evacuation Models

The novelty of our approach to the pursuit-evasion problem is the use of probabilistic evader models to compute the most probable trajectories that the evaders will take. In the context of SAR it is useful to think of evacuation behavior during emergencies. These behaviors, commonly referred to as emergency egress, tend to have three distinct analytical dimensions [15]:

1) the environment and its configuration at the time of the evacuation,
2) the existing policies and procedures that are in place prior to the evacuation occurring,
3) the social psychological and social organizational characteristics effecting those participating in the evacuation.

Every scenario is comprised of these three dimensions, and there are many factors that contribute to a particular evacuation behavior. For instance, it has been shown that by varying evacuation instructions and changing the number of posted exit signs, people react differently when they hear a fire alarm [8]. A behavioral-based evacuation framework that focuses on the interaction between agents as they attempt to navigate through an environment in the presence of hazards, was presented by Rodríguez and Amato [13].

Our implicit representation of the distribution of evader trajectories using a collection of samples is similar in spirit to the particle filter methods that have been used for mobile robot localization [4], [5].

III. PROBLEM STATEMENT

This section formalizes the visibility-based pursuit-evasion problem where a single pursuer, equipped with a model for the motion of the evaders, attempts to visually locate one or more evaders in a polygonal environment.

A. Representing the Environment, Pursuer, and Evaders

The environment is a simply connected closed set $P \subset \mathbb{R}^2$, with polygonal boundary ∂P. We define R as the set of reflex vertices in P. For any point $r \in P$, let $VR(r)$ denote the

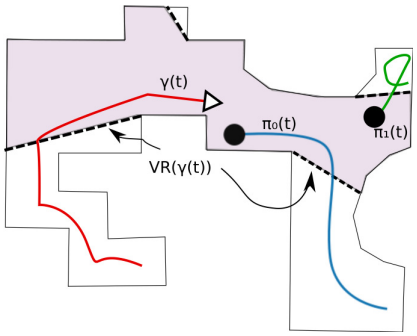


Fig. 2. An illustration of our notation. A single pursuer (triangle) moves through an environment in search of a unknown number of evaders (circles).

visibility region at point r, which consists of the set of all points in P that are visible from point r. That is, any line segment formed by connecting point r and a point from the set $VR(r)$ lies in the polygonal free-space.

A single pursuer and an unknown number of evaders are modeled as single points that translate in the polygonal free-space. Let $\pi_i(t) \in P$ represent the position of the i^{th} evader at time $t \geq 0$. The trajectory π_i is a continuous function $\pi_i : [0, \infty) \rightarrow P$, in which the evader is capable of moving arbitrarily fast (i.e. a finite, unbounded speed) within P. Let Π denote the function space of all such trajectories. Similarly, let $\gamma(t) \in P$ denote the position of the pursuer at time $t \geq 0$. Without loss of generality, we say that the pursuer moves along γ with maximum speed 1. Figure 2 shows this notation.

The pursuer's goal is to see every evader. A pursuer trajectory γ is a *solution strategy* if, for all evader mappings $\pi \in \Pi$, there exists a finite time of capture, denoted $t_c(\gamma, \pi)$ and defined as

$$t_c(\gamma, \pi) = \min\{t \geq 0 \mid \pi(t_c) \in VR\big(\gamma(t_c)\big)\}.$$

The time $t_c(\gamma, \pi)$ is the time of capture for an evader following trajectory π_i when the pursuer follows solution strategy γ. Thus, the pursuer's task is to find a solution strategy with a finite time of capture for every possible evader trajectory.

B. Probabilistic Evader Models

Although the exact positions of the evaders are unknown to the pursuer at time t_0, the pursuer has at its disposal a predictive model for the motions of the evaders. The motion model reflects a behavior that the evaders are likely to exhibit throughout the run.

Specifically, we assume that there exists a probability density function $p : \Pi \rightarrow [0, 1)$, that models the likelihood of each possible trajectory in Π being selected by each evader. A critical assumption is that evaders' motions are independent of the motions of the pursuer.[1] Because p may be difficult to express explicitly, we assume only that the pursuer is able to sample evader trajectories, drawn from this density. For example, p might model evader behavior in which there are nonzero probabilities for trajectories that (1) seek a known "exit" in the environment, (2) move erratically, and (3) remain largely motionless. Equipped with this information, the pursuer can compute a solution strategy that minimizes the expected time to locate the evaders.

[1]This assumption is, of course, unreasonable for traditional pursuit-evasion problems. We believe that it is appropriate for this problem because the evaders are merely somewhat unpredictable, rather than being truly antagonistic. Pursuit-evasion concepts are still relevant in this context because we are interested in a *guarantee* that all evaders are located, regardless of the trajectories they select.

C. Success Metrics

We consider two different outcomes when evaluating the solution strategies generated by our algorithm.

1) First, we are interested in minimizing the expected time to locate each evader, defined as the Expected Time to Capture (ETC):

$$E(\gamma) = \int_{\pi \in \Pi} p(\pi) t_c(\gamma, \pi) \; . \tag{1}$$

The intuition of $E(\gamma)$ is to compute the average time to capture each evader as the pursuer follows its solution strategy. However, because the ETC is difficult to evaluate exactly, in practice we instead approximate the ETC as

$$E(\gamma) \approx \sum_{i=1}^{n} \frac{t_c(\gamma, \pi_i)}{n} \; , \tag{2}$$

in which n trajectories are sampled from Π as described above.

2) Second, we consider how long it would take for the pursuer to execute its solution strategy to completion, defined as the Guaranteed Time to Capture (GTC):

$$G(\gamma) = \max_{\pi \in \Pi} (\, t_c(\gamma, \pi) \,) \; . \tag{3}$$

For a given solution strategy γ, $G(\gamma)$ can be computed using the technique introduced by Guibas, et al. [7].

We use both of these criteria when computing the effectiveness of the algorithm. We seek to decrease the ETC without drastically increasing the GTC.

IV. ALGORITHM DESCRIPTION

This section describes our algorithm to solve the visibility-based pursuit-evasion problem introduced in Section III. The algorithm goes through several stages. In the first stage, a set T of potential evader trajectories is created by sampling from Π. In the next stage, a forward search is conducted to generate a path that allows the pursuer to see each evader in T. That is, when the trajectory set T is used as the input in our search, the forward search returns a path that would qualify as a solution strategy if the evaders followed only those trajectories in T. The last stage appends motions to the path returned from the forward search so that the path is a solution strategy for Π.

A. Pursuer Trajectory Graph

The graph that we search to find a path that can view all trajectories in T is constructed from the reduced visibility graph (RVG) [10] of P. Our motivation for using the RVG is twofold. First, the RVG contains shortest paths in P, so pursuer paths along the RVG represent locally optimal plans for traversing the environment. Second, note that for every non-convex[2] polygonal environment P, the union of the visibility regions of the reflex vertices of P is equal to P itself. This means that every element in P is visible from at least one reflex vertex of P; the intuition is that the RVG is "big enough" to allow the pursuer to see evaders wherever they travel in the environment.

Based on the RVG, we define the *purser trajectory graph* with vertex set $[0,\infty) \times R \times \{0,1\}^n$. For a specific vertex $v = (t, r, b_1 \cdots b_n)$, the interpretation is that after time t has elapsed, the pursuer is at reflex vertex r, and evader trajectory i has been seen if and only if $b_i = 1$. Each vertex $(t, r, b_1 \cdots b_n)$ has one outgoing edge for each reflex vertex adjacent to r in the reduced visibility graph [14] of P. The "child" node's value of t is equal to that of the "parent" node plus the time it takes the pursuer to travel from the child's r to the parent's r. As the pursuer travels between reflex vertices, the pursuer updates the b_i values. That is, for all b_i in the parent that are equal to 0, the pursuer checks if there is any $t \in [t_{parent}, t_{child}]$ such that $T_i(t) \in VR\big(\gamma(t)\big)$, and if so sets b_i equal to 1 in the child node. Any b_i equal to 1 in the parent node is carried over as a 1 in the child node.

B. Forward Search

The pursuer performs a graph search on the pursuer trajectory graph for a path from the root vertex, $V_r = \big(0, \gamma(0), b_1 \ldots b_n\big)$ in which $b_i = 1$ if and only if $\pi_i(0) \in VR(\gamma(0))$, to a vertex in which all values of b_i are equal to one. The forward search that we implemented was a variation of Dijkstra's Algorithm using t as the cost metric. See Algorithms 1 and 2.

The primary difference between our search and the standard Dijkstra's algorithm is a pruning of redundant nodes. For a vertex $v = (t, r, b_1 \cdots b_n)$ to be considered redundant there must already be a vertex $v' = (t', r', b'_1 \cdots b'_n)$, in the *closed* list, such that

1) both nodes share the same physical location, so that $r' = r$, and
2) the vertex v' has already seen every evader trajectory that v has seen, so that $b'_i \geq b_i$ for all $1 \leq i \leq n$.

We perform the redundancy check immediately before the vertex is expanded, instead of when the vertex is added to the *open* list, to account for the fact that it may become redundant in the interim. For example, this situation may occur when a vertex $h = (t_h, r_h, b_h)$ is at the top of the expansion queue and creates a child, $k = (t_k, r, b)$. As a result, k is added to the *open* list. Later, a vertex $j = (t_j, r_j, b_j)$ is at the top of the expansion queue and creates a child, $l = (t_l, r, b)$, so l is added to the *open* list. Although vertex k has the same r and b values as vertex l and was added to the *open* list earlier, l will appear at the top of the expansion queue before k if $t_l < t_k$. Assuming that when l reaches the top of the expansion queue it is determined that l is not redundant, then l will be added to the *closed* list. This would mean that when k reaches the top of the expansion queue, k would fail the redundancy check because l is already in the *closed* list.

All vertices that fail the redundancy check are discarded. Figure 3 shows a simple example of the search, in which the redundant node pruning takes effect.

[2]Note that if P is convex, every evader is immediately visible, trivially solving the problem.

Algorithm 1 FORWARDSEARCH$\big(\gamma(0), T\big)$

```
open ← empty priority queue of vertices ordered by t
closed ← empty set of vertices
for all π ∈ T do
    bπ ← π(0) ∈VR(γ(0))
end for
add vertex (0, γ(0), b) to open
while open is not empty do
    v ← open.top()
    (t, r, b_1 ··· b_n) ← v
    if b_1 ··· b_n = 1 ··· 1 then
        return success
    end if
    if not REDUNDANT(v) then
        add v to closed
        EXPANDNODE(v, T)
    end if
end while
```

Algorithm 2 EXPANDNODE$\big(v, T\big)$

```
(t, r, b_1 ··· b_n) ← v
for each element r' in RVGNEIGHBORS(r) do
    t' ← t + (time for pursuer to travel from x to x')
    path ← SEGMENTBETWEEN(r, r')
    for all π ∈ T do
        b'_π ← b_π
        for time ∈ [t, t'] in increments of Δt do
            b'_π ← b'_π or π(time) ∈ VR(path(time))
        end for
    end for
    v' ← (t', r', b'_1 ··· b'_n)
    add v' to the open list
end for
```

C. Appended Forward Search

Although our initial forward search is guaranteed to find a path that "sees" every evader that follows a trajectory in T, the forward search does not account for evaders that do not follow a trajectory that is part of the sampled set T. As a result, it is possible that the path returned by the forward search does not see all of the evaders.

To account for this possibility, we append the path returned

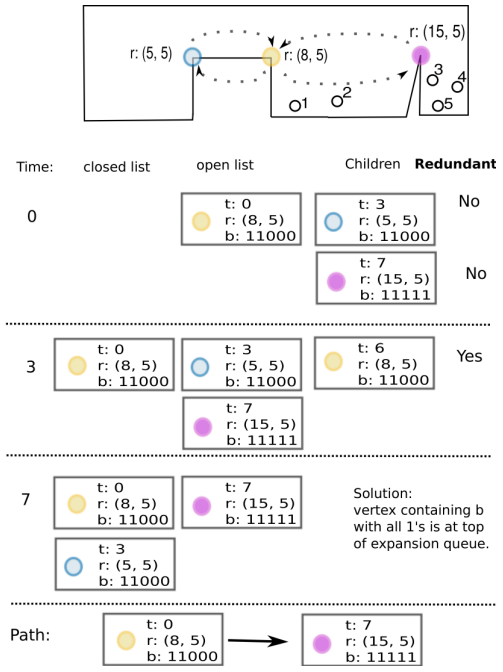


Fig. 3. A basic example demonstrating how the forward search is conducted. Starting at the middle vertex the pursuer searches for a path that can see all 5 evaders(open circles) in the environment. The dotted lines represent the possible transitions from one vertex of the RVG to another.

by the forward search with actions that eliminate all of the remaining possible contaminations in the environment after the pursuer reaches the end of the path generated in Section IV-B.

We use the technique described by Guibas, et al. [7] (for which some details appear in Section II-A) to do a cell decomposition of P, partitioning P into conservative regions. We then find the sequence of conservative regions traversed by the path returned by our forward search, maintaining the gap labels through this sequence of conservative regions. We then run the the algorithm of Guibas et al., modified to start its search in the last conservative region of the sequence instead of the pursuer's original location, and with the updated gap labels instead of initially labeling all gaps as contaminated. In the best case, it is possible that the path generated by the forward search travels through a sequence of conservative regions that eliminate all contaminations in P, which means that our path does not require any additional actions to become a solution strategy. If there are contaminations, the prior algorithm supplies the necessary actions to make our path a solution strategy.

V. RESULTS

We used C++ and CGAL [2] to implement this algorithm. We tested our algorithm in several different environments and under various evader motion models. The following section describes some of the scenarios in which our algorithm was run, followed by the results comparing the ETC and GTC from our plan to the ETC and GTC of the prior, uninformed algorithm. Throughout our results, the pursuer moves through the environment at speed 1, and each evader moves at a constant velocity in the range of $1/2$ to $3/2$.

Figure 4a depicts a very simple environment that has nine vertices in the RVG. For our simulation in this environment we used $n = 1,000$ sample trajectories with uniformly random starting points. Of these, 250 do not move, another 250 wander aimlessly in the environment, while the other 500 have a uniform probability of heading along shortest paths to ends of each of the four hallways.

The solution generated by our forward search explores the hallways in the following order: top left, bottom left, bottom right, and then top right. At this time, the entire left section of the environment and the middle hallway remain as possible hiding places for the evaders. The additional actions appended to catch theses evaders require the pursuer to re-visit the top left and bottom left portions of the environment.

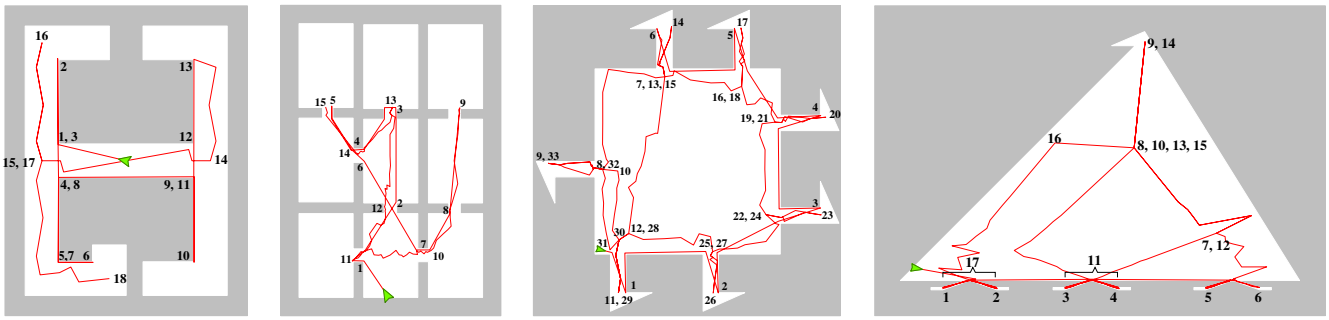
Figure 4b depicts an environment resembling an office building floor plan. For this scenario, we used $n = 200$ sample evader trajectories. Of these, 25% move randomly through the environment (modeling victims with no knowledge of the location of exits), 25% move toward the primary exit located at the far left wall in the middle room, 25% move toward secondary exits located at the far right wall in the bottom right corner of the middle room, the far right wall in the bottom right corner of the south room, and along the north facing wall in the middle room. The remaining 25% of the trajectories model victims that remain motionless due to injury or indecision. The evaders' starting positions are evenly distributed across five designated meeting spaces located in all three middle rooms, the middle room along the south wall, and the room in the top left corner. However, the group of evaders without knowledge of the exit locations start scattered randomly through the environment.

Figure 4c depicts an environment originally used by Suzuki and Yamashita [17] in their explanation of the visibility-based pursuit-evasion problem. In this example, the sampled evader trajectories all start within the the six rightmost legs of the environment. A small fraction (20%) of these evaders travel across the center chamber to the leftmost leg. The remaining evaders remain close to their starting locations. The intuition is to model a scenario, such as certain kinds of potential disasters in schools, in which many of the victims may elect not to try to escape.

Figure 4d shows an environment well known to require multiple recontaminations of the upper "peak." A correct solution strategy requires the pursuer to make repeated trips through the center chamber. We tested this our algorithm using evader models in which the evaders remain clustered in the "feet" of the environment. The algorithm successfully exploits this knowledge to locate all of the sampled evaders relatively quickly, in contrast to the extremely conservative plan generated by the algorithm of Guibas et al.

A. *Quantitative Comparison*

For each of these four environments, we executed both our algorithm and the uninformed algorithm of Guibas et al. For each generated plan, we computed both the ETC and GTC.



[a] [b] [c] [d]

Fig. 4. Solution strategies generated by our algorithm for four environments. Integer labels show the sequence of pursuer movements.

TABLE I

RESULTS OF SIMULATIONS

Environment	ETC	Guibas et al. ETC	GTC	Guibas et al. GTC
1 (Figure 4a)	294.44	441.79	2340.17	1514.72
2 (Figure 4b)	334.96	561.57	2564.17	1596.06
3 (Figure 4c)	394.08	754.45	4341.05	2796.63
4 (Figure 4d)	303.40	842.93	2515.15	1984.52

Table I shows the results. The difference in ETC between the two algorithms can be interpreted as the amount of performance increase obtained by our algorithm's exploitation of the probabilistic motion models for the evaders. In these environments, our algorithm is successful in improving the ETC, without sacrificing the completeness of the algorithm.

Notice that each of these environments have a specific "solution path" that must be followed in order to guarantee that all the evaders have been captured. For instance, in Environment 1, the two right corridors must be cleared sequentially, followed by the top left corridor, before finishing in the bottom left corridor. As result, the path generated by our forward search may not necessarily clear any of the gaps tracked by the Guibas et al. algorithm. This inherent limitation explains the increase in GTC shown in Table I. However, because our algorithm maintains gap labels throughout its execution to determine the starting node for the Guibas et al graph search, it is able to take advantage of any gaps that actually are cleared during the forward search.

VI. CONCLUSION

We presented an algorithm for visibility-based pursuit-evasion that uses a collection of sampled evader trajectories to improve the expected time required to locate the evaders. This algorithm is motivated by the need to efficient search strategies in robotic SAR, and by the extremely conservative paths generated by uninformed pursuit-evasion planners.

ACKNOWLEDGMENT

We gratefully acknowledge support for this work from NSF (IIS-0953503) and DARPA (N10AP20015).

REFERENCES

[1] D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi. Multi-objective exploration and search for autonomous rescue robots: Research articles. *Journal of Field Robotics*, 24(8-9):763–777, 2007.

[2] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[3] W. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete Computational Geometry*, 6(1):9–31, 1991.

[4] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. IEEE International Conference on Robotics and Automation*, 1999.

[5] A. Doucet, N. J. Gordon, and V. Krishnamurthy. Particle filters for state estimation of jump markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624, 2001.

[6] B. P. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. In *International Journal of Robotics Research*, pages 20–27, 2004.

[7] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. *International Journal of Computational Geometry and Applications*, 9(5):471–494, 1999.

[8] N. R. Johnson and W. E. Feinberg. The impact of exit instructions and number of exits in fire emergencies: A computer simulation investigation. *Journal of Environmental Psychology*, 17(2):123 – 133, 1997.

[9] E. D. Kuligowski and R. D. Peacock. Review of building evacuation models. Technical report, National Institute of Standards and Technology, July 2005. NIST TN 1471; NIST Technical Note 1471; 153 p.

[10] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic, 1990.

[11] S. M. LaValle, B. Simov, and G. Slutzki. An algorithm for searching a polygonal region with a flashlight. *International Journal of Computational Geometry and Applications*, 12(1-2):87–113, 2002.

[12] T. D. Parsons. Pursuit-evasion in a graph. *Theory and Application of Graphs, Lecture Notes in Mathematics*, pages 426–441, 1976.

[13] S. Rodríguez and N. M. Amato. Behavior-based evacuation planning. In *Proc. IEEE International Conference on Robotics and Automation*, pages 350–355, 2010.

[14] H. Rohnert. Shortest paths in the plane with convex polygonal obstacles. *Information Processing Letters*, 23(2):71–76, 1986.

[15] G. Santos and B. E. Aguirre. A critical review of emergency evacuation simulation models. *NIST Workshop on Building Occupant Movement during Fire Emergencies*, pages 27–52, 2004.

[16] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. A multi-robot strategy for rapidly searching a polygonal environment. In *Advances in Artificial Intelligence IBERAMIA 2004*, volume 3315 of *Lecture Notes in Computer Science*, pages 484–493. Springer Berlin / Heidelberg, 2004.

[17] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21(5):863–888, 1992.

[18] B. Tovar and S. M. LaValle. Visibility-based pursuit-evasion with bounded speed. In *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2006.