2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Active Target Search from UAVs in Urban Environments

Christopher Geyer

Abstract— **In this paper we consider the problem of searching for a target from a camera-equipped unmanned aerial vehicle (UAV) flying in an urban area. Urban areas present challenges because buildings can hamper the ability to see regions on the ground. We describe an algorithm that constructs paths that take into account obstructions due to buildings or other large objects. The approach combines search trees and a particle filters to evaluate a large number of possible paths, while at the same time performing all the Bayes' filter innovations that would need to occur during the evaluation of each path.**

I. Introduction

In this paper we consider the problem of searching a region with complex geometry, looking for a moving target from an aerial sensing platform such as an unmanned aerial vehicle that has a limited field of view. In addition, we are interested in environments that have complex geometry, where targets may be behind buildings, and without taking these into account we may fail to find the target. As an example of such an environment, see Figure 1. This is an approximate three-dimensional model of the McKenna MOUT site located at Fort Benning, which we have reconstructed manually from orthographic imagery. The target in red lies somewhere in this area, and as the target moves around within the environment, we would like to locate the target. This problem has immediate application to military operations in urban or mountainous terrain, but may also be applicable to search, rescue, or wildlife monitoring operations in mountainous terrain, or planning for movie production, for example.

A. Related Work

In 1965 Isaacs [8] first presented a unifying treatment of *differential games*, where the prototypical problem is to find a strategy that guarantees collision with, or capture of, another vehicle. Later, Parsons [11] considered a pursuit-evasion game on a graph, where the goal is to occupy the same edge as the evader.

Even earlier, much work had been done in World War II with the aim of finding search strategies for locating submarines. Koopman assembles much of this knowledge in the monograph [10]. This topic was explored again by Stone in [13]. These works treat macroscopic issues such as resource allocation, as well as optimal sweep widths for searching in wide open areas, with uniform geometry and target distribution.

C. Geyer is at the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA; e-mail: cgeyer@ri.cmu.edu .

I would like to thank Geoff Hollinger, Athanasios Kehagias, Ben Grocholsky, and Sanjiv Singh for their insight and comments on this paper.

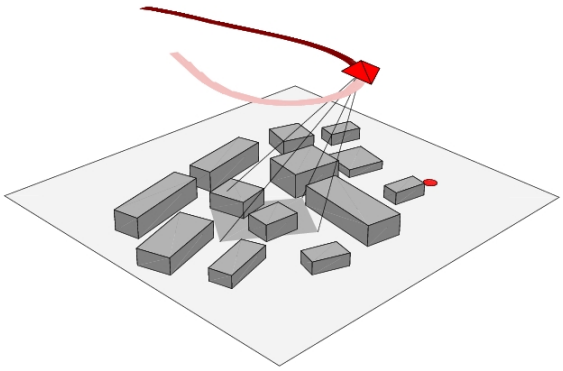


Fig. 1. A UAV searching for a target in an urban environment with complex geometry and occlusions. The model shown is an approximate 3D model of Fort Benning's McKenna MOUT site reconstructed manually from orthographic imagery.

Several works have considered exact algorithms that guard and search indoor polygonal areas while guaranteeing capture if feasible; we mention only a few. Suzuki and Yamashita [14] first examined the problem of how to guarantee capture in polygonal environments—where capture only requires that the pursuer see the evader with a beam. Guibas et al. [6] consider the same, except that instead of a beam, they assume a pursuer with an omnidirectional sensor for capture. Recently Gerkey et al. [5] extend this to allow for a limited field-of-view.

Often the solutions or representations for the exact approaches become infeasible with large regions or multiple search, or they do not provide solutions in the case where no solution with a guarantee exists. Other approaches solve the problem probabilistically, most commonly using partially observable Markov decision processes (POMDP).

Roy [12], for example, computes policies on the space of densities representing where the evader may be. He represents the high-dimensional densities using a small basis of distributions, thereby expressing the density in a lower-dimensional space. He then solves the POMDP in the smaller space.

Yu et al. [16] enumerate several criteria for deciding when it is sufficient to compute open-loop plans instead

of full policies. They claim that under certain conditions, such as that there is no sensor feedback, the agent's plant can be assumed to be noiseless, or that there is a "stopping-time" sensor, then open-loop plans are sufficient.

Approaches to pursuit-evasion can be applied to search and rescue, and Bourgault et al. [2] consider spotting survivors of a marine accident from an unmanned aerial vehicle (UAV). They formulate the problem similar to that of a POMDP and compute an open-loop plan over a finite horizon.

Vidal et al. [15] demonstrate a pursuit-evasion implementation collaboratively with both unmanned aerial and ground vehicles. They provide approximate greedy policies, and represent the state of the evader on a grid, and assume Markov motion model, or non-adversarial model, for the target.

Hollinger et al. [7] investigates pursuit-evasion in an in-door environment, and provide algorithms for approximating a POMDP solution on a graph representation of the environment, and test the effects of coupling or decoupling multiple pursuers.

B. Contribution

We express the problem as a pursuit-evasion problem in which the target is ignorant of the pursuers presence, or is non-adversarial. In a formulation similar to that of [2], we derive the probability of finding the target as a function of an open-loop path. Here we approximate the integral by a sampling scheme. By doing so we can more efficiently evaluate possible control laws over a finite horizon. Our primary contribution:

- *Efficient filtering.* We present a planner that updates a particle filter based on the expected measurements, and does so with reasonable efficiency for horizons up to 16 timesteps deep. We can perform what is equivalent to $16,000$ particle filter updates on $1,000$ particles per second on a 2 Ghz Pentium M processor.
- *Compression of visibility function.* We present a method to compress the visibility function so as to efficiently evaluate whether a given UAV point can see a specified point on the ground. The visibility representation is stored in a set of images, which give the minimum visible elevation of the sky that is visible from each point on the ground in a fixed number of directions.
- *Memory-less planners do just as well.* We also find a negative result: a planner that updates the expected posteriors along the way does only marginally better than a memory-less planner, which does not update the posterior based on expected measurements.

II. A Search Game

In this section we describe a simplified search game which is a simplification of a pursuit-evasion game in which the evader's movements are conditionally independent of the pursuer's state. This models a situation in which the evader is non-adversarial, or indifferent to the evader, or if the pursuer is sufficiently stealthy that the evader is unaware of the pursuer's state. Specifically, we make the following assumptions:

- *Target.* A target of interest exists in an environment described by state-space $\mathscr{X}$. State transitions occur at discrete timesteps, and on a finite horizon, T-timesteps into the future, P_{x^T} gives a probability distribution on sequences during this horizon, e.g. $x^T = (x_1, \ldots, x_T) \in \mathscr{X}^T$. This distribution need not be stationary, that is, have identical distributions across time periods, but we assume for notational simplicity that P_{x^T} is the most current distribution. We further assume that the transitions are Markov.
- *Pursuer.* Our pursuer(s) exist in a state-space $\mathscr{Y}$, and our agent(s) transitions in $\mathscr{Y}$ occur deterministically. We assume that the effects of high-level controls are modeled by a kinematic (or dynamic) model specified by:

$$y_{k+1} = F(y_k, u_k) \tag{1}$$

  where $u_k \in \mathscr{U}$. Noisy control effects are assumed to be handled by a low-level motion planner (see for example or [16]).
- *Capture event.* C_{x_i,y_i} represents the event that when the agent is at y_i at timestep i, it captures the target at x_i. For a capture event, it is necessary—but not sufficient—that the target be within the field-of-view of the sensor.
- *Conditional distribution.* The conditional distribution $P(C_{x_i,y_i}|x_i)$ depends not only on x_i and y_i, but also on the properties of a target detection system (e.g., false negative rate or the signal-to-noise ratio of the target against the background), as well as the environment (such as whether the environment occludes the target for the given x_i and y_i). We further assume that conditioned on x_i and x_j, C_{x_i,y_i} and C_{x_j,y_j} are independent. We discuss visibility in section II-B.
- *Stopping time τ_c.* The stopping-time τ_c gives the first timestep that our agent(s) captures—i.e., sees—the target.
- *Non-adversarial agent.* We assume that the sequence x^T is independent of y^T; i.e., the target's transition $P_{x_k|x_{k-1}}$ is independent of y_{k-1}. This could be the case if the target is indifferent to or unaware of being pursued, or alternatively the agent may be stealthy, so that the evader is unaware of the agent's position.

The goal is to find an open-loop path that optimizes some criteria that favors finding the evader. One proposal would be to find a path that minimizes the expected time-to-capture, which we can calculate using the path y^T and

the distribution P_{x^T}. In doing so, though, we need to provide a cost-to-go value at the end of the horizon. Various heuristics could be used, or values could be obtained via assumptions about further paths (e.g. Brownian motion). Instead of the minimizing τ_c, we choose to maximize the CDF of τ_c evaluated at time T. If the capture event were the flip of a coin, then the capture time would be the first time we get tails, and the CDF is the sum of probabilities of getting heads on the first $k-1$ tries, and getting tails on the kth try:

$$P[\tau_c \leq T] = \sum_{k=1}^{T} P\left[\overline{C}_{x_1,y_1}, \ldots, \overline{C}_{x_{k-1},y_{k-1}}, C_{x_k,y_k}\right] \quad (2)$$

where $\overline{C}_{x_i,y_i}$ is the complement of the event $\overline{C}_{x_i,y_i}$. The latter equation is implicitly a function of the control inputs in u^{T-1} that generate the path y^T according to equation (1).

By conditioning on x^T we can expand each of the terms in the sum as follows:

$$\begin{aligned} &P\left[\overline{C}_{x_1,y_1}, \ldots, \overline{C}_{x_{k-1},y_{k-1}}, C_{x_k,y_k}\right] \\ &= \int P[x^T] P\left[\overline{C}_{x_1,y_1}, \ldots, \overline{C}_{x_{k-1},y_{k-1}}, C_{x_k,y_k} | x^T\right] dx^T \\ &= \int P[x^T] P[C_{x_k,y_k}|x_k] \prod_{j=1}^{k-1} P\left[\overline{C}_{x_j,y_j}|x_j\right] dx^T \end{aligned} \quad (3)$$

If $\dim \mathscr{X} = d$, then this integral is dT dimensional. We can compute the integral using Monte Carlo integration, drawing samples of the chain over the entire horizon. We draw N samples $x^{T(i)}$, each of length T, from the Markov chain with joint PDF P_{x^T}. We can then write the CDF that is a function of the open-loop control input sequence u^{T-1} as:

$$\begin{aligned} &P[\tau_c \leq T](u^{T-1}) \\ &= \sum_{k=1}^{T} \int P[x^T] P[C_{x_k,y_k}|x_k] \prod_{j=1}^{k-1} P\left[\overline{C}_{x_j,y_j}|x_j\right] dx^T \\ &= \sum_{k=1}^{T} \left[\frac{1}{N} \sum_{i=1}^{N} f(y_k, x_k^{(i)}) \prod_{j=1}^{k-1} \overline{f}(y_j, x_j^{(i)}) \right] \end{aligned} \quad (4)$$

where the latter equality holds in distribution as $n \to \infty$, and where $f(y_k, x_k^{(i)}) = P[C_{x_k,y_k}|x_k = x_k^{(i)}]$ and $\overline{f}(y_k, x_k^{(i)}) = 1 - f(y_k, x_k^{(i)})$, i.e., the probability that the target is not scene given the agent's current position, and the position of the target represented by the ith particle. Since the probability is bounded, the Monte Carlo approximation converges in distribution by the central limit theorem. Note, though, that as a function of the sequence u^{T-1}, uniform convergence is *not* guaranteed. In other words, the number of samples required to achieve a given accuracy may depend on u^{T-1}.

A. Maximizing Probability of Capture

Next we write out the probability of capture as a function of the control input sequence, and maximize it by using a dynamic programming principle. In doing so we will also show how we can ...

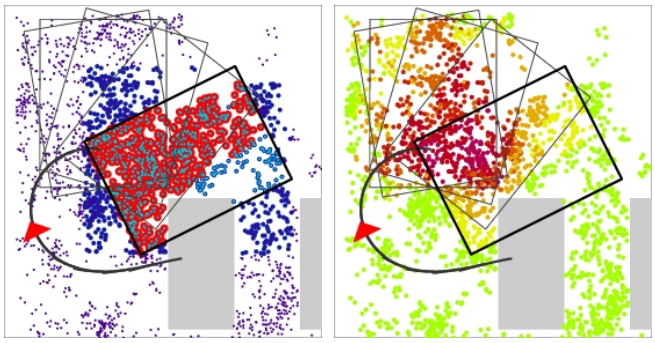


Fig. 2. Left: State of the algorithm during evaluation at a single node. The dark rectangle represents the FOV of the UAV projected to the ground. The red-backed/dark-backed particles are visible from the UAV. Right: State of the visibility counts at the eight step in the horizon. Redder particles have greater counts. The thinner rectangles represent previous FOVs of the UAV projected to the ground. In both cases the red triangle is the position of the UAV. The FOV is fixed and points down and to the right.

We let $C(u^{t-1})$ be the cost of the control sequence u^{t-1} on a horizon of length t, wherein y^t is implicitly computed from u^{t-1}:

$$C(u^{t-1}) = \sum_{k=1}^{t} \frac{1}{N} \sum_{i=1}^{N} f(y_k, x_k^{(i)}) \prod_{j=1}^{k-1} \overline{f}(y_j, x_j^{(i)}). \quad (5)$$

We wish to find the maximum of C over controls, and we let C^* be the maximizing cost:

$$C^* = \max_{u^{T-1} \in \mathscr{U}^{T-1}} C(u^{T-1}) \quad (6)$$

Now, since in the sum comprising $C(u^t)$, the first $t-1$ terms do not depend on u_t, we may apply a principle similar to that used for dynamic programming principle [1], and write the maximizing cost function recursively, as a function of the control inputs up to and including $t-1$:

$$C^*(u^t) = D(u^{t-1}) + \arg\max_{u_t \in \mathscr{U}} C^*\left([u^{t-1}, u_t]\right). \quad (7)$$

with terminating condition $C^*(u^{T-1}) = D(u^{T-1})$ and where $D(u^{t-1})$ is the last term of the sum from $C(u^{t-1})$:

$$D(u^{t-1}) = \frac{1}{N} \sum_{i=1}^{N} f(y_t, x_t^{(i)}) \prod_{j=1}^{t-1} \overline{f}(y_j, x_j^{(i)}). \quad (8)$$

Unfortunately we can not get away from the fact that costs in the future depend on the path we took to get there, and so $C^*(u^t)$ must be computed for all possible u^t. This is the case, first, because of the dependence of y^t on u^{t-1}; and, second, the fact that the executed path affects the belief about where the target is. The chief advantage of writing it in the form of (7) is so that we may use memoization to build up $C^*(u^t)$.

Given that we need to compute to $C^*(u^t)$ for all possible u^t, how can we best do so efficiently? We re-write $D(u^{t-1})$

as follows:

$$D(u^{t-1}) = \frac{1}{N}\sum_{i=1}^{N} f\left(y_t(u^{t-1}), x_t^{(i)}\right) g_i(u^{t-1}) \qquad (9)$$

where we have made more explicit the dependence of y_t on u^{t-1}, and where h is the product of $\overline{f}$'s given a single sample $x^{T(i)}$:

$$\begin{aligned} g_i(u^k) &= \prod_{j=1}^{k} \overline{f}\left(y_j(u^{j-1}), x_j^{(i)}\right) \\ &= \overline{f}\left(y_k(u^{k-1}), x_k^{(i)}\right) g_i(u^{k-1}) \end{aligned}$$

Note the properties of g_i:

- If sample i is not visible from y_t then $f(y_t, x_t^{(i)}) = 0$. Therefore, at any one timestep the set of non-zero summands in $C(u^{t-1})$ is non-zero on a set that is likely to be small compared to N, the total number of samples, and could be computed from a single range query.
- If sample i is visible from y_t, and if the conditional distribution only depends on visibility, i.e., $P(C_{x,y}|x) = \rho$ if x is visible from y and 0 otherwise, then $g(y^t, i) = \rho\,(1-\rho)^n$ where n is the number of times in the preceding $t-1$ steps that sample i was visible.

Therefore in order to calculate the sum of g's efficiently, we do the following. We do a depth first traversal of control sequences u^T. We keep an array that maintains a count on the times sample i was visible. At each y_t generated from a sequence u^{T-1}, we determine the set of indices $\mathscr{I}(y_t)$ of samples visible from y_t at time t. For each $i \in \mathscr{I}(y_t)$, we evaluate the sum of g using the array of visibility counts; then we increment visibility counts for each $i \in \mathscr{I}(y_t)$ and we recurse on each $u_t \in \mathscr{U}$; when all the recursions have returned, we go back to each $i \in \mathscr{I}(y_t)$ and decrement the corresponding visibility counts, and return. The algorithm is presented in Algorithm 1.

B. Visibility

For each query particle $x_t^{(i)}$ and waypoint y_t we evaluate whether the particle lies within the field-of-view of the UAV given it's current state, and also whether the particle is occluded by buildings or other structures from the UAV's position. In order to do this efficiently, we first only consider points which are within in an axis-oriented bounding box of the FOV of the UAV projected to the ground. We construct this potential list using a 2D range query, which is constructed for all particles at time t—that is, we create T $2D$ range trees. Of the results from the query, we remove those which lie outside the FOV of the UAV projected to the ground. Then we remove the particles that are occluded.

Algorithm 1 Compute optimal path

```
function BestPath(y_t, u^{t-1}, v : ref, T)
    V ← 𝒱(y^t, {x_t^(i)})                ▷ Get visible particles
    p ← 0
    for i ∈ V do                          ▷ Increment visible particles
        v_i ← v_i + 1                     and calculate D(u^{t-1})
        p ← p + ρ (1−ρ)^{v_i}
    end for
    if t < T then                         ▷ If not at end of horizon...
        for u_t^(j) ∈ 𝒰 do                ▷ We assume 𝒰 is discrete
            y_{t+1} ← F(y_t, u_t^(j))
            (u^{t(j)}, C^(j)) ←
                BestPath(y_{t+1}, [u^{t-1}, u_t^(j)], v, T)
        end for
        j ← arg max_j C^(j)               ▷ Save the best sub-path
        p ← p + C^(j*)
        u* ← [u^{t-1}, u_t^(j*)]          ▷ Concatenate the input
    else                                  to the current sequence
        u* ← u^{t-1}
    end if
    for i ∈ V do                          ▷ Restore visibility counts
        v_i ← v_i − 1
    end for
    return (u*, p)
end function
```

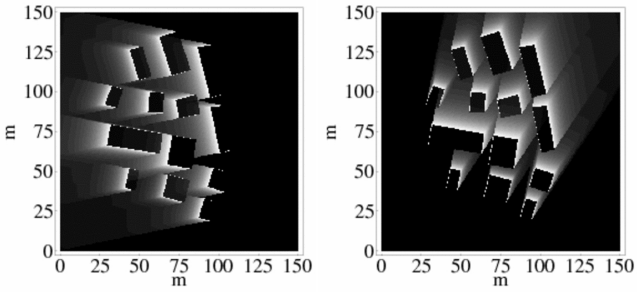


Fig. 3. Left and right: Two of the 16 channels of the visibility representation for the McKenna MOUT site at Fort Benning. The axes are measured in meters. The left corresponds to points in a $\pi/8$ wedge due east; the right to a wedge to the south-south-west. Lighter colored regions are more occluded in the given direction, meaning that they are only visible at higher elevations, or directly up.

We use a visibility function $\mathscr{V}(x,y)$ which is 1 is a target at x on the ground is visible from the UAV in state y. Though this is a 6D function depending on the positions of both the target and the UAV, we compress this function into a single $2D$ array. To do this we assume that the target is on the ground, and that heights in the world are given by a $2D$ height function $\mathscr{H}(p)$ where $p \in \mathbb{R}^2$. In other words, there are no tunnels, and the outward normals of building façades have non-negative z-components. Then we let

$$\tilde{\mathscr{V}}(x,k) = \max_{\substack{p\,:\,p=(x,\mathscr{H}(x)),\\ \frac{2\pi k}{n} \le \theta(p-x) < \frac{2\pi(k+1)}{n}}} \phi(p-x)$$

where $\theta(p)$ and $\phi(p)$ are the spherical coordinates of $p \in \mathbb{R}^3$, where $\phi(0,0,1) = \pi/2$. In other words, $\tilde{\mathcal{V}}(x,k)$ gives the minimum elevation angle at which the target on the ground can be guaranteed to be seen from points within the kth pie wedge emanating from the target. Figure 3 shows the encoding for McKenna MOUT site at Fort Benning.

C. Control Loop and Filtering

The control loop is as follows:

1) Use the sensor to evaluate whether the target is in view—if the target is seen, we are done.
2) Update a particle filter (see [4] for a review of sequential Markov Chain Monte Carlo methods) based on whether the target was seen or not.
3) Sample N chains from the distribution P_{x^T} represented by the particle filter.
4) Using the N samples, evaluate the optimal control input sequence u^{T*} over the horizon T using Algorithm 1.
5) Apply $u_0{}^*$, the first coordinate of u^{T*}, to the plant.
6) Repeat steps 1 through 5 until the target is found.

To compute P_{x^T} we use simple 2nd order Brownian motion with bounded velocities. We are investigating the usage of more realistic and predictive models such as those described by Bruce et al. in [3].

III. Experiments

In this section we describe some of the results of running the closed-loop controller in a simulation environment. We use four simulated environments, three of which are shown in 4, the fourth is the McKenna MOUT site shown in Figure 1. We assume that the target motion model is a 2nd order Brownian motion model with bounded velocities, i.e. velocities are Brownian except for the fact that they are restricted to a box. In the McKenna simulations, the target has a maximum velocity of $2\,\mathrm{m/s}$. The UAV flies at a low altitude of $12.5\,\mathrm{m}$, and the maximum roof height is $9\,\mathrm{m}$. In the simpler maps the roof heights are 150 units of height in a map with 400×400 pixels; in these scenes, the UAV has an altitude of 200 units.

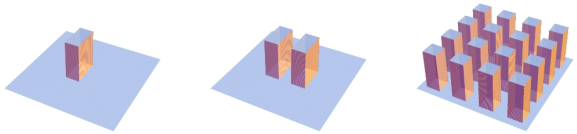
We measure four different algorithms: *Full*: The Algorithm as stated in Algorithm 1; *Blind*: The same as *Full* except the visibility check due to occlusions is not done; *Memory-less*: A modification of Algorithm 1 that is memory-less, meaning that there is no term that discounts for having seen the same particle in the past—then, line 6 of Algorithm 1 becomes $p \leftarrow p + \rho$; *Memory-less+Blind*: A combination of *Memory-less* and *Blind*. To measure performance we report the probability that the target was captured, as well as the median time to capture. In all cases, when measuring whether we have captured the target, we make sure that it is not occluded.

The left column of figure 6 shows the probability of capture and median time to capture as a function of the map number. The map numbers are coded according to Figure 4. In general, what turns out to be surprising is the little difference between the *Memory-less* and *Full* planners. Also of note, in the case of *Map 3*, both blind controllers perform not as well in terms of probability of capture. Thus, the visibility does yield an improvement in the more complex environment.

The middle column shows the effects of horizon length. Surprises here are the low median time for the *Memory-less+Blind* planner with only 4-step look-ahead, however, this is coupled with a lower probability of capture.

The right column gives probability of capture and median time to capture for the McKenna MOUT site, using 6- and 11-step lookaheads. Here it is noticeable that the *Full* and *Blind* planners, neither of which are memory-less, excel in the 6-step lookahead.

Overall, it is notable that horizon has little effect, nor does whether the planner is memory-less or not. The visibility computations help when it counts, in the more complicated *Map 3* scene.



Map 1 *Map 2* *Map 3*

Fig. 4. Three simple maps used for testing.

IV. Conclusion

In this paper we have described an algorithm and framework for generating trajectories for UAVs in search of a target, which is either non-adversarial, or is unaware of the UAV's presence. Nevertheless, several issues need to be solved to make this method more effective:

- *Pruning.* Planning horizon is limited by the need to test many paths. What heuristics can be used to prune branches likely to be unfruitful?
- *Gimbal cameras.* We assumed a fixed camera mount, and we assumed the aircraft does not roll, which is unrealistic. We need to the UAV state extra dimensions for the camera gimbal. This, however, would increase the dimension of the state space, and may require more branching.
- *Convergence.* Convergence properties, and uniformity of convergence are unknown, and need to be determined.
- *Transitioning to optimal pursuit plans.* If the optimal UAV trajectories for search differ for those for pursuit when the target's position is known, are there optimal configurations for transitioning from search to pursuit,

Fig. 5. Left: 3D view of the state of McKenna MOUT simulation in the 92nd time step. Right frames: Bird's eye view at time steps 30, 60, and 90 showing the evolution of the path.

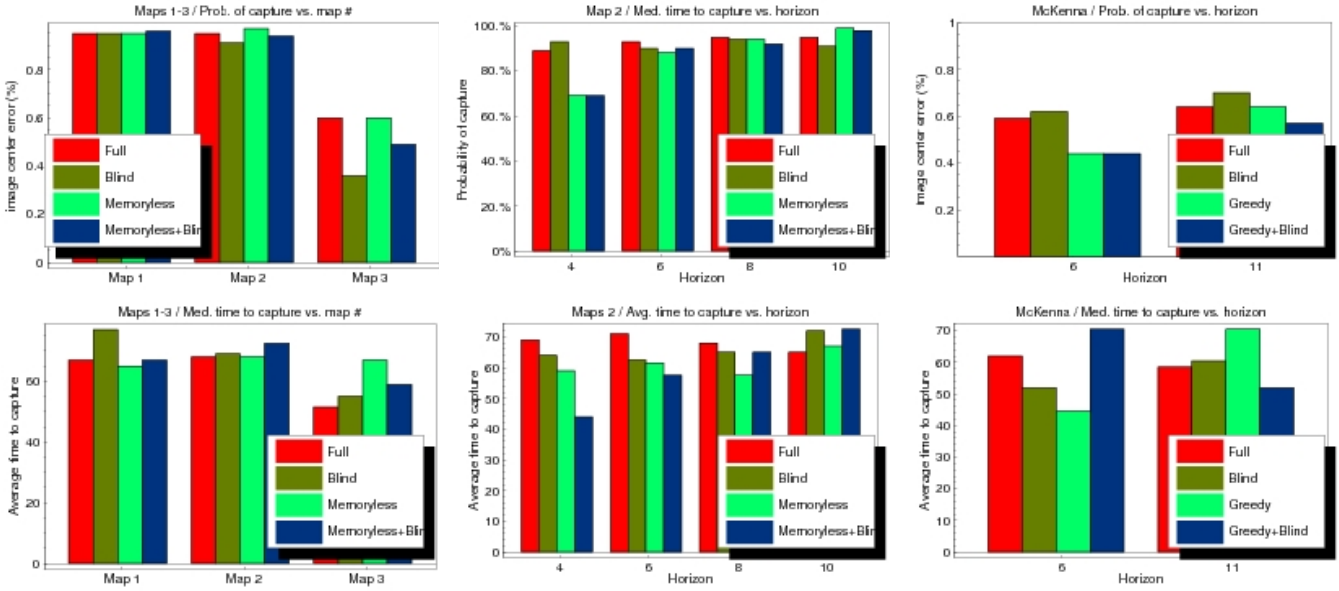


Fig. 6. Results of experiments; see text in section III for explanation.

and should this be taken into account when putting a metric on search trajectories?

These are some of the issues that we plan to address in the future. In addition, we are investigating how this algorithm might be applied to a real UAV in a MOUT environment.

REFERENCES

[1] D. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[2] F. Bourgault, A. Göktogan, T. Furukawa, and H. F. Durrant-Whyte. Coordinated search for a lost target in a bayesian world. *Advanced Robotics*, 18(10):979–1000, 2004.

[3] A. Bruce and G. Gordon. Better motion prediction for people-tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2004.

[4] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[5] B. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. *International Journal of Robotics Research*, 25(4):299–315, 2006.

[6] L. Guibas, J. Latombe, S. 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.

[7] G. Hollinger, A. Kehagias, and S. Singh. Probabilistic strategies for pursuit in cluttered environments with multiple robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2007.

[8] R. Isaacs. *Differential Games*. Wiley, New York, NY, 1965.

[9] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 5(21):864–875, 2005.

[10] B. Koopman. *Search and Screening*. Pergamon, 1980.

[11] T. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D. Lick, editors, *Theory and Applications of Graphs*, pages 426–441. Springer, 1976.

[12] N. G. D. Roy. *Finding approximate pomdp solutions through belief compression*. PhD thesis, Carnegie Mellon University, 2003. Chair-Tom Mitchell and Chair-Sebastian Thrun.

[13] L. Stone. *Theory of Optimal Search*. Operations Research Society of America, 1989.

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

[15] R. Vidal, S. Rashid, C. S. Sharp, O. Shakernia, J. Kim, and S. Sastry. Pursuit-evasion games with unmanned ground and aerial vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2948–2955, 2001.

[16] C. Yu, J. Chuang, B. Gerkey, G. Gordon, and A. Ng. Open loop plans in multi-robot pomdps. Technical Report, Stanford CS Department, 2005.