Appeared in the Proceedings of the American Control Conference (ACC 2006). Minneapolis, June 2006.

Dynamic Action Spaces for Information Gain Maximization in Search and Exploration

Nicholas Roy and Caleb Earnest

Abstract— **The problem we investigate is how an autonomous, mobile agent can search for a hidden, moving target efficiently. A good control strategy will plan more informative sensing of the world, allowing the agent to find the target quickly. Searching for moving targets typically involves planning over probability distributions, or beliefs, that characterize the possible locations of the target. Most motion strategies choose actions that reduce the uncertainty of the current belief by maximizing the predicted information gain of the next action, but computing good multi-step plans is usually computationally intractable [1] due to the high dimensionality of the action and belief spaces.**

In this paper, we describe a novel algorithm for generating search plans using dynamic action spaces. The algorithm clusters a particle filter description of the current belief at each point in time, and uses search to compute a trajectory through the clusters in order to maximize information gain. This model allows us to efficiently compute finite-horizon multi-step plans in extremely high dimensional problems. We show preliminary results for an unknown target tracking problem.

I. Introduction

The problem we investigate is how an autonomous, mobile agent can search for moving opponents efficiently. A good motion planner will generate plans that allow the agent to gather as much information as possible, narrowing the space of possible locations for the opponent as quickly as possible. At the same time, a good planner should incorporate the cost of moving around the environment, producing plans that trade off the gain from sensing against the cost of acting.

There are a number of applications of intelligent motion controllers for search and exploration, including military target surveillance for unmanned vehicles and homeland security surveillance in populated environments. Additionally, search problems have gained research attention in the health care domain for patient tracking and monitoring, especially for eldercare. Finally, good motion strategies for information gain are essential for autonomous agents building environmental models, such as mobile robots building maps autonomously.

Planning good motion strategies for search and exploration can be computationally intractable. Modern target tracking techniques generally rely on probabilistic techniques to estimate the likelihood of target locations, such as the Kalman filter or the particle filter. As sensor data is acquired, at each point in time, probabilistic inference algorithms provide a full probability distribution, or belief, over the space of possible target locations. The advantage to probabilistic inference is that the posterior probability distribution of the target location is usually robust to sensor noise and motion uncertainty, allowing for reliable inference of the target location over time.

The disadvantage to using probability distributions for planning, however, is the computational intractability of computing plans based on the full distribution. The planning problem grows exponentially with the size of the input space (the space of distributions of target locations), which also grows in complexity with the size of the underlying state space (the space of actual possible target locations). However, *not* incorporating the probability distribution into planning process, or not planning at all, can lead to poor performance. Figure 1 shows an example search problem, where a hidden agent enters from the right, and can pass either above or below the small island obstacle before heading to one of the three goals. The mobile agent starts to the left of the obstacle, and must find the target before the target reaches the goals. The agent must make motion planning decisions based not only on its current location, but also on the relative likelihood that the target passes above or below the island obstacle.

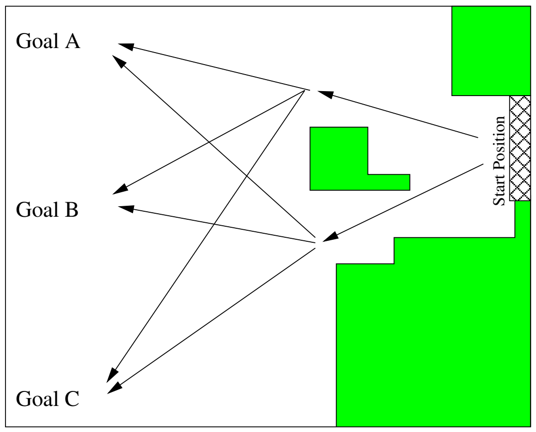


Fig. 1. An example search and track problem. The hidden target enters from the right, and is headed to one of the three goals (A, B or C), but the sensor does not know which goal. The mobile sensor's goal is to detect the target before the target reaches its goal.

Figure 2 shows two example trajectories using a simple greedy strategy to search for the next most likely location of the target, and failure to incorporate knowledge of the full probability distribution into the planner leads to suboptimal trajectories. In the upper panel (1-3), the agent causes the posterior distribution to split into three modes; this means that there are three distinct areas where the target can be,

N. Roy is a member of the Computer Science and Artificial Intelligence Laboratory and the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge MA 02139 USA

C. Earnest performed this work while at Draper Laboratories, Cambridge, MA.

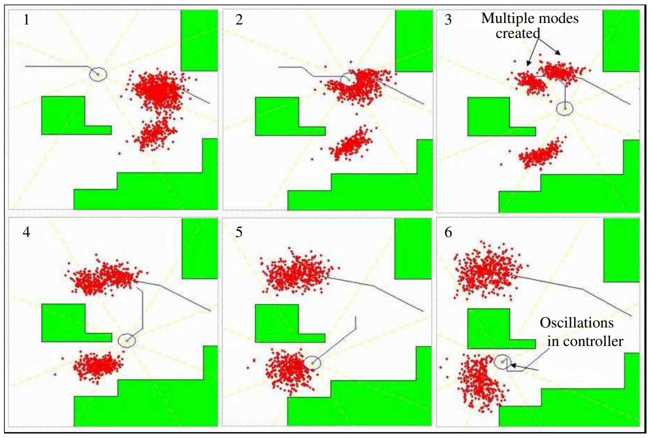


Fig. 2. Two examples of bad motion planning. The solid blocks are obstacles, the open circle is the mobile agent, and the small particles are samples drawn according the distribution of possible target locations at each point in time. In the upper series of panels, the search trajectory causes the posterior distribution to split into three distinct modes, making the search problem harder. In the lower series of panels, the two modes are approximately equal in information gain, and so the controller oscillates between the two modes as the modes move away.

making the target detection substantially harder. In the lower panel (4-6), the agent's control does not recognize that it is oscillating between two approximately equal modes in the distribution.

Computing good motion plans is a planning and control problem in the high-dimensional space of possible probability distributions, or possible beliefs. If the underlying state space is discretized into n possible states, then the space of possible beliefs (the input space to the planner) has dimensionality $n - 1$. If the underlying state space is continuous, then the input space is potentially of infinite dimension. The challenge is how to compute good control actions in this very high dimensional problem space.

II. Dynamic Target Tracking

The problem of our mobile agent is to find a moving target as quickly as possible. We assume that the mobile agent and the target occupy the same (x, y) space; without loss of generality we will neglect orientation. We assume that the agent operates by issuing some control action u that causes its own position to change, and it then receives some (binary-valued) observation z that indicates whether the agent can detect the target or not. Once the agent has inferred (from the sequence of observations) that it has detected the target, the agent declares that it has found the target and the search problem ends.

If we model the motion of the agent, target and the target detection observations probabilistically, we can compute a probability distribution for the locations of the agent and target at time t, $\mathbf{a}_t$ and $\mathbf{x}_t$ respectively, from the sequence of actions $u_0, u_1, \ldots, u_t$ and observations $z_0, z_1, \ldots, z_t$:

$$p(\mathbf{a}_t, \mathbf{x}_t | z_t, u_t, z_{t-1}, u_{t-1}, \ldots, z_0, u_0). \tag{1}$$

If we assume that the states of the agent and target are Markovian and we make use of some basic rules of probability, we can factor this distribution into

$$\begin{aligned} p(\mathbf{a}_t, \mathbf{x}_t | z_t, u_t, z_{t-1}, ut-1, \ldots) = \\ \eta p(z_t|\mathbf{x}_t, \mathbf{a}_t) \iint p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{a}_t|u_t, \mathbf{a}_{t-1}) \cdot \\ p(\mathbf{x}_{t-1}, \mathbf{a}_{t-1}|z_{t-1}, u_{t-1}) d\mathbf{x}_{t-1} d\mathbf{a}_{t-1}. \end{aligned} \tag{2}$$

In this equation, $p(z_t|\mathbf{x}_t, \mathbf{a}_t)$ is the probabilistic model of the detection observations, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the motion model of the target, $p(\mathbf{a}_t|u_t, \mathbf{a}_{t-1})$ is the motion model of the agent and η is a normalizer. Note that we have factored the motion models of the agent and target to reflect conditional independence, i.e., we assume that the target moves independently of the agent. If the two models are dependent, we would simply have a more complex, joint motion model. This equation for the posterior distribution, a special form of the general Bayes' filter [2], gives a recursive way to update the current belief over agent and target location by incorporating a new control and observation into the previous belief.

We do not have a closed form representation of the exact posterior distribution $p(\mathbf{x}_t, \mathbf{a}_t|z_t, u_t)$. A common approach to representing posteriors in tracking problems is to the Extended Kalman Filter approximation [3], however, the EKF assumes that the distribution can be well-approximated by a single Gaussian distribution, and the filter computes the maximum *a posteriori* mean and covariance. An examination of the data (such as the posterior distributions shown in figure 2) for the target tracking problem indicates that the EKF is likely to provide a poor approximation as the distribution contain several modes.

Instead we will represent the posterior distribution using the non-parametric model of particle filtering, in which the posterior is approximated by a set of sample states that are distributed according to the distribution $p(\mathbf{a}_t, \mathbf{x}_t|z^t, u^t)$. We will use an algorithm called Importance Sampling [4], [5], which assumes knowledge of a *proposal* distribution that we can easily sample from and is reasonably "close" to the desired *target* distribution to be sampled (in this case, the posterior of equation 2); the assumption is that the target distribution can be evaluated for a given state but cannot be easily sampled. The basic process is to generate a proposal set of samples $\{\mathbf{x}^i\}$ according to the proposal distribution $q(\mathbf{x})$. (Again without loss of generality we will temporarily ignore the agent position $\mathbf{a}$ for clarity.) Each sample is assigned an importance weight

$$w^i = \frac{p(\mathbf{x}^i)}{q(\mathbf{x}^i)} \tag{3}$$

where $q(\mathbf{x}^i)$ is the probability of sample $\mathbf{x}^i$ according to the proposal distribution and $p(\mathbf{x}^i)$ is the probability of sample $\mathbf{x}^i$ according to the target distribution. The weighted samples are then resampled according to their weights, and in the limit of an infinite number of samples, the sample density will converge to the target distribution density[5].

In the case of Bayesian filtering, we have access to samples from the prior distribution which can be propagated through the motion model to become a proposal distribution for the posterior distribution:

$$w^i = \frac{p(\mathbf{x}_t^i|z_t, u_t)}{p(\mathbf{x}_t^i|u_t)} = \frac{\eta p(z_t|\mathbf{x}_t^i) \int p(\mathbf{x}_t^i|u_t, \mathbf{x}_{t-1}) d\mathbf{x}_{t-1}}{\eta \int p(\mathbf{x}_t^i|u_t, \mathbf{x}_{t-1}) d\mathbf{x}_{t-1}} \quad (4)$$

$$= p(z^t|\mathbf{x_t^i}). \quad (5)$$

The full Importance Sampling algorithm for Bayesian filtering is given in table I.

```
Input: Particle set $p_{t-1} = \{\mathbf{x}_{t-1}^0, \mathbf{x}_{t-1}^1, \ldots, \mathbf{x}_{t-1}^n\}$,
control $u_t$, observation $z_t$.
  1) Sample from proposal distribution:
       • $\forall \mathbf{x}_{t-1}^i$: sample $\mathbf{x}_t^i$ according to $p(\mathbf{x}_t|u_t, \mathbf{x}_{t-1}^i)$
  2) Compute importance weights for each particle:
       • $\forall \mathbf{x}_t^i : w^i = p(z_t|\mathbf{x}_t^i)$
  3) Sample from $n$ particles from $\{\mathbf{x}_t^i\}$ according to weights $\{w^i\}$
     to generate $p_t = \{\mathbf{x}_t^0, \mathbf{x}_t^1, \ldots, \mathbf{x}_t^n\}$
```

TABLE I
IMPORTANCE SAMPLING FOR BAYESIAN FILTERING.

The advantage to the particle filter method for tracking is that arbitrary distributions such as those in figure 2 can be easily represented; additionally, there are fewer restrictions on the prediction and measurement models. The two major disadvantages to particle filtering involve the complexity of the particle set, that is, the number of particles required to accurately represent the distribution. Firstly, the complexity of the particle set is exponential in the number of variables being tracked; in our example of agent and target position tracking, we have a small number of variables and therefore do not pay a heavy penalty in our problem. Secondly, the number of samples required to approximate the posterior distribution depends on the divergence between the proposal and target distributions; the greater the divergence between the two distributions, the more samples are required to accurately represent the target distribution. However, in the case of Bayesian filtering for real-world dynamical systems, the proposal distribution offered by the motion model is frequently a good approximation to the posterior, and the number of particles can be kept fairly low.

III. Motion Planning for Target Tracking

Given that we are using probabilistic inference to compute a posterior over possible locations of the target, we can use decision theory to choose good measurements. Decision theory phrases the problem as one of choosing measurements that maximize information, where the information gain is the change in the entropy of the posterior distribution, $H(p) = \int_x p(x) \log p(x) dx$. Because we are using a non-parametric representation of a continuous distribution, however, the entropy cannot be easily computed; we approximate the information gain with the relative change in variance of the distribution. Our agent must therefore choose trajectories that minimize the posterior variance of the set of particles.

This planning problem suffers from computational complexity in that we must compute a plan based on the current belief (probability distribution) as represented by our particle set, rather than a single state estimate. The cost of planning grows exponentially with the horizon length (the number of actions considered in the plan). As a result, we are unlikely to be able to compute trajectories consisting of a long sequence of actions. A second pressure on the search depth is that evaluating the expected information gain of any particular trajectory will require integrating over a set of possible observations that will also grow exponentially with the trajectory length. Careful consideration of the appropriate action space is required in order to ensure that useful trajectories can be generated with relatively short sequences of actions.

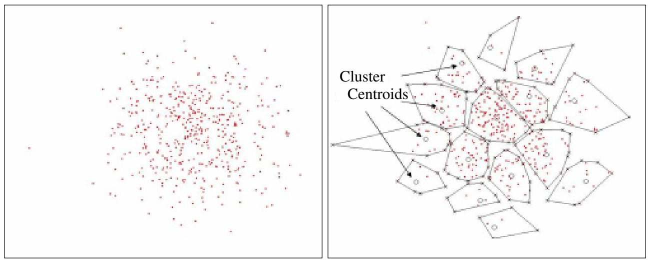


Fig. 3. An example particle set and the resulting clusters.

```
Input: Particle set $p_t = \{\mathbf{x}_t^0 = (x_t^0, y_t^0), \ldots, \mathbf{x}_t^m = (x_t^m, y_t^m)\}$
  1) Initialize:
     $clusters_t = \{\}$
  2) while $p$ is not empty:
       a) Choose $\mathbf{x}_t^i$ from $p_t$ at random
          and remove $p_t = p_t \backslash \mathbf{x}_t^i$
       b) Initialize cluster: $\mathbf{c} = \mathbf{x}_t^i$
       c) Find all remaining particles in range of $\mathbf{x}_t^i$ and add to
          cluster center
          $\forall \mathbf{x}_t^j \in p_t$:
          if $||\mathbf{x}_t^i - \mathbf{x}_t^j|| < r_{sensor}$
              $\mathbf{c} = \mathbf{c} + \mathbf{x}_t^j$
              $count = count + 1$
       d) If cluster contains more than one particle, compute
          cluster centroid and retain cluster
          if $count > 1$
              $\mathbf{c}_x = \mathbf{c}_x / count$
              $\mathbf{c}_y = \mathbf{c}_y / count$
              $clusters_t = \{\mathbf{c}\} \cup clusters_t$
```

TABLE II
THE CLUSTERING ALGORITHM.

Dynamic Action Spaces

Many planning processes assume a static action space. For example, kinematic motion planners assume discrete waypoints constructed using a visibility graph, randomized sampling or grid-based discretization, with a static set of actions to connect the waypoints. The planning process is then a search for an optimal sequence of actions that will transition the agent from the start state through an appropriate set of waypoints to the goal. However, a complete set of static waypoints and actions is less useful when planning with probability distributions; the probability mass obeys a principle of locality. For example, if the current belief has little or no probability mass near a waypoint, then the waypoint and associated actions should be temporarily pruned from the planning problem. We can then rely on some lower-level motion planner to compute trajectories between waypoints, allowing the search process to consider trajectories only between points with information gain.

Clustering Algorithm

In order to generate a set of candidate actions (and associated waypoints), we use the fact that the particle filter representation of the posterior gives us a potential set of waypoint samples directly from the sample states. At each time step we divide the particle set into clusters consisting of particles that are mutually within the sensor range of the agent, r_{sensor}. The centroid of each cluster constitutes a potential destination waypoint during the search process. Table II describes this clustering algorithm. Note that outliers (clusters of only one particle) are ignored.

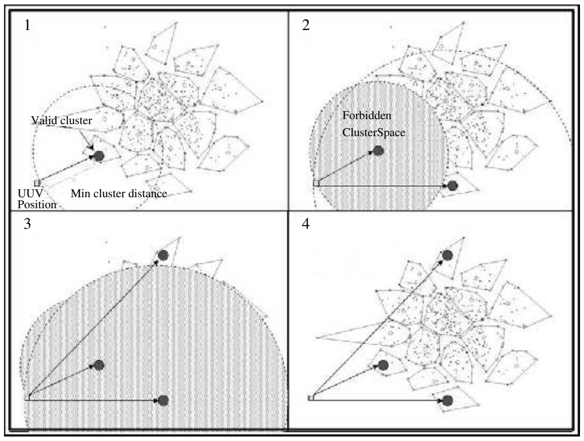


Fig. 4. The action selection process. The cluster centroids are sorted by distance to the agent, and then the closest cluster centroid in each range is kept as a valid waypoint for the search, eliminating other cluster centroids in the same range from the search process.

Once the particle set has been clustered, we further prune the clusters based on relative distances to the agent. Clusters are sorted according to the distance to the agent, and the closest cluster centroid of radius r is considered as a valid waypoint in the trajectory search. All clusters closer than radius r are then eliminated from the set of valid waypoints. The process is then repeated with the next closest cluster of radius r'. Figure 4 shows three steps of the clustering process, resulting in three waypoints being selected for the first step of the search process.

Search

Once the set of actions (cluster centroids acting as trajectory waypoints) has been constructed, we use a conventional search process as outlined in table III. The major bottleneck of the search process is determining the posterior distribution after each potential action. There is no closed-form solution for computing this distribution, so we simulate the agent and particle motion and agent observations along the trajectory. The particle weights are then updated and the particles resampled to arrive, at time $t+1$, a hypothesized posterior $\hat{p}_{t+1}$. This particle set is then reclustered to form the action space for the following step of search. The process operates to some maximum depth (a maximum number of actions), and the information gain of a particular sequence of actions (or waypoints) is computed with respect to the final distribution.

```
Input: Particle set $p_t = \{\mathbf{x}_t^i\}$ and current recursion depth $depth$
  1) Generate waypoint set $\{\mathbf{c}^i\}$ from particle set $\{\mathbf{x}_t^i\}$ using
     clustering and selection algorithm in table II as shown in
     figure 4.
  2) For each waypoint $\mathbf{c}^i$:
       a) Simulate agent motion to waypoint $c_t^i$ and generate new
          potential particle set $\hat{p}_{t+1}$ during motion
       b) Incorporate simulated measurements and compute
          weights of $\hat{p}_{t+1}$
       c) Resample $\hat{p}_{t+1}$ according to weights
       d) If $depth = max\_depth$
              Return information gain of $\hat{p}_{t+1}$
          else
              $gain(\mathbf{c}^i)$ = step 1) with $(\hat{p}_{t+1}, depth + 1)$
  3) Return $\text{argmax}_{\mathbf{c}}\, gain(\mathbf{c})$
```

TABLE III
THE SEARCH ALGORITHM.

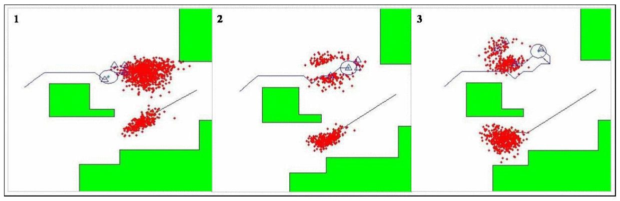
Measurement Updates and Resampling

During the search process, the particle filter treats the motion to a cluster centroid as a single action. As a result, the particle weights are not updated with measurements or resampled until some time has passed. The advantage is that the search process is accelerated, as the process of simulating observations and resampling the weighted particles is the slowest part of the search process. However, a low measurement update rate during the search can lead to two problems: firstly, the predicted information gain at the end the trajectory will not be accurate. Secondly, if the particle weights are updated but the particles are not resampled, then a phenomenon known as "particle death" can occur where, as the result of accumulating a long sequence of observations, none of the particle weights has any reasonable probability mass. Regular resampling has the effect of ensuring that probability mass is retained by the set of particles. However, the appropriate rate at which the importance weights need to be updated and resampled is an open question. Figure III shows the results of different resampling rates. In the upper figure, the particle filter is only resampled at the end of each action, resulting in a trajectory that causes the agent to split a mode of a distribution (figure III(a), second panel). Even though the algorithm search process allows the agent to recover from this problem (figure III(a), third panel), an alternative action selection may have contained the mode and therefore further decreased the uncertainty in the location of the contact.

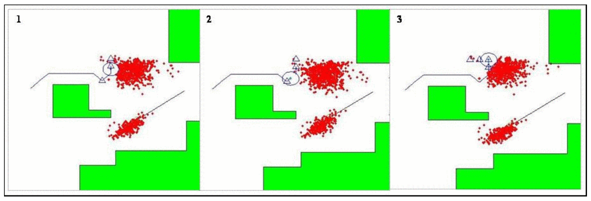
In contrast, if we update the particle weights and resample ten times along each potential trajectory, the search process identifies the trajectory in figure III(b) as having higher information gain; as a result, the agent does not split the mode but instead constricts the location of the mode.

IV. EXPERIMENTAL RESULTS

We performed preliminary experiments to compare different search strategies. We assume that the agent has a fixed detection radius of .9 distance units, and the relative speed of the agent compared to the target is 1.75 units/time step. We assume that the agent motion model is deterministic, the



(a) Low Measurement Update Rate



(b) Fast Measurement Update Rate

target motion model is uniform probability towards one of the three goals shown in figure 1. With 75% probability the target moves over the top of the barrier in the middle of the scene (Scenario 1), and 25% probability through the narrow channel under the barrier (Scenario 2). Gaussian noise is added to each action taken by the target. We assume that the observation model consists of binary observations (target detected/undetected) with accuracy 90%. The particle filter estimation process used 1,000 particles.

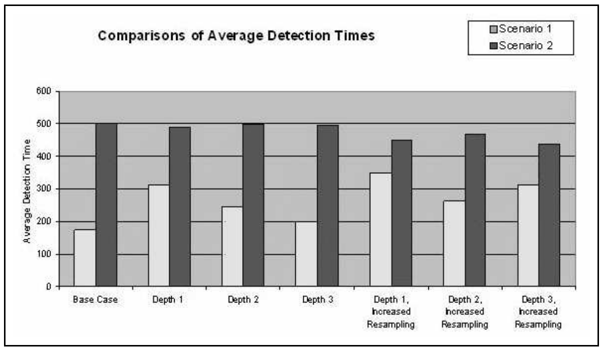


Fig. 5. Average time to detection for the different algorithms. In scenario 1 (light coloured bars) the target takes the (higher probability) upper route, and in scenario 2 (dark coloured bars), the target takes the (lower probability) lower route. The base case is the direction with maximum single-step information gain. Note that each run was stopped at 500 iterations. Each result is the average of 20 trials.

The baseline strategy maximizes single-step information gain, but does not perform any prediction or measurement updates in computing the information – the agent simply moves to the location where a single observation would cover the maximum probability mass. The search algorithms search for full trajectories with search depth of 1, 2 and 3 actions to maximize information gain, and increased resampling refers to performing measurement updates and particle resampling 10 times between path waypoints.

Figure 5 shows the average time to detect the target for various algorithms. The light coloured bars show the performance in scenario 1 (high probability) and the dark coloured bars show the performance in scenario 2. The search algorithms search for full trajectories with 1, 2 and 3 waypoints to maximize information gain, and increased resampling refers to additional particle resampling between path waypoints. In situations where the target is found, performing search and increasing the search depth appears to have a mild effect on detection time, although the trend is not statistically significant (error bars not shown). Note that the baseline algorithm performs well only when the target is actually following the high probability trajectory, suggesting that the baseline algorithm is brittle, as we might expect.

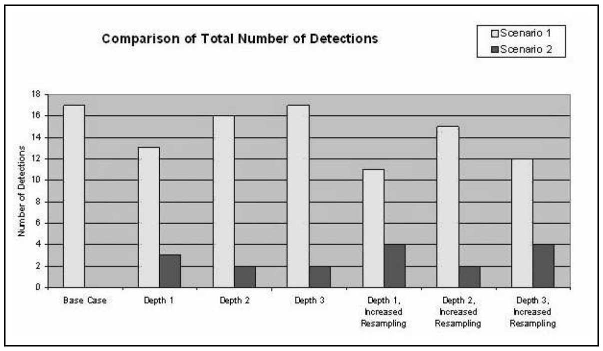


Fig. 6. Number of successful detections for the different algorithms, out of 20 trials. In scenario 1 (light coloured bars) the target takes the (higher probability) upper route, and in scenario 2 (dark coloured bars), the target takes the (lower probability) lower route. Note first that the additional search depth can help find the target even in the harder scenario 2, but that regular resampling is key for accurately estimating the posterior distribution.

Figure 6 shows the number of successful target detections of each algorithm, which highlights the advantage of using search in this problem domain. In particular, we see that increasing search depth improves performance in both scenarios; again, although the baseline algorithm performs well in high probability scenarios, it never detects the target in the low probability scenario. In contrast, searching to depth 3 allows the agent to find the target with equal success in the high probability case, and some moderate success in the low probability case. Note that the algorithms fail to find the target in some trails because we stop each trial after 500 steps. The increased resampling clearly has an effect in the harder search cases in that low probability modes can be extinguished more quickly. However, there is still high variance in the performance at depth 3, suggesting that deeper search may be useful.

Finally, figure 7 shows a longer run of an example trajectory, where the motion planning occurs out to depth 3. After each action, the plan is recalculated. Notice that the planner is careful to minimize the number of mode splits, and spends time extinguishing complete modes before moving on. (The depicted trajectory is not smooth as an artifact of the underlying dynamic controller.)

V. Related Work

The search and exploration problem has been studied in a number of different contexts. The simplest form of information gathering is to use a greedy strategy, where

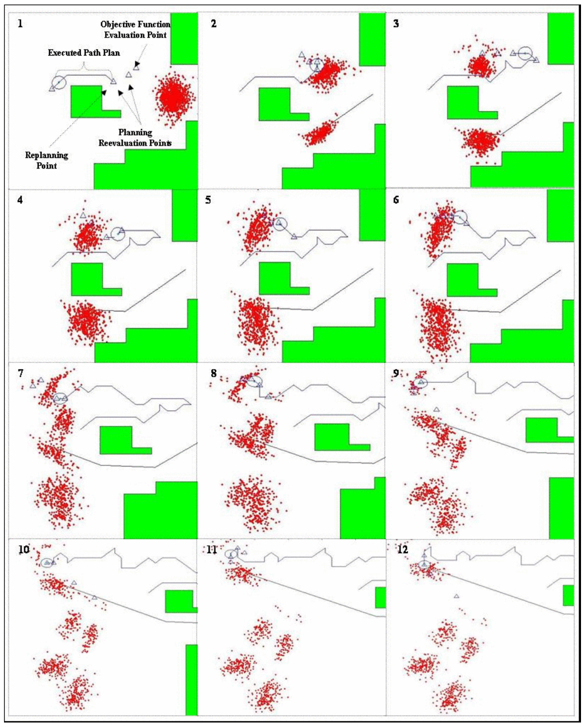

1
Objective Function Evaluation Point
Executed Path Plan
Planning Reevaluation Points
Replanning Point
2
3
4
5
6
7
8
9
10
11
12

Fig. 7. An example trajectory for a search depth of 3. Notice that the motion planner attempts to extinguish complete modes before moving on.

the motion planning chooses the single best destination to take the next measurement [6], [7]. However, we have shown that the greedy strategy can be improved by planning longer-horizon trajectories. One of the most general forms of long-term sequential decision making with uncertainty is the Partially Observable Markov Decision Process [1] (POMDPs), but exact algorithms for solving POMDPs are computationally intractable [8], suffering from complexity that is potentially doubly exponential in the horizon length and observation space [9]. More recent efforts to reduce the complexity of solving POMDPs have used different forms of approximation, including variants of Principal Components-based approximations of the belief space [10]. Similarly, Poupart & Boutilier [11] used a linear model reduction algorithm on the value function directly. More recent efforts have shown that sampling the belief-space can lead to efficient computation of good approximate value functions [12], [13], [14], [15], and in particular Pineau et al. [16] have applied their PBVI algorithm to specifically the search problem. The Policy-Contigent Abstraction (PolCA) algorithm [17] is one of the closest algorithms in spirit to the Dynamic Action Space algorithm described in this paper, in that the planning problem is decomposed into a hierarchy of problems, where different actions are applicable in different regions of the belief space. However, the PolCA algorithm assumes a fixed action decomposition *a priori*, whereas our action decomposition strategy is dynamic, dictated by the current state of the particle filter.

VI. Conclusion

In this paper we have described an algorithm for a mobile agent to plan its motion and sensing to minimize the time to detection of a moving hidden target. The contribution of this work is to show how planning using search can be made tractable by dynamic selection of actions, allowing useful trajectories to be planned even with relatively small search depths. We showed preliminary results that demonstrated the practicality of this approach. However, a number of issues must be resolved in future work, including the particular action selection algorithm; our clustering approach here was based on relative distances between particles. Improved performance may come from more appropriate clustering approaches. Additionally, further complexity gains may be had by performing smarter sample propagation, such as using the unscented Kalman filter [18], which we will study in future work.

Acknowledgements

The authors gratefully acknowledge the support of Draper Laboratories.

References

[1] E. Sondik, "The optimal control of partially observable Markov decision processes," Ph.D. dissertation, Stanford University, Stanford, California, 1971.

[2] S. Thrun, D. Fox, and W. Burgard, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[3] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[4] A. Smith and A. Gelfand, "Bayesian statistics without tears: a sampling-resampling perspective," *American Statistician*, vol. 46, pp. 84–88, 1992.

[5] R. Neal, "Probabilistic inference using markov chain monte carlo methods," Dept. of Computer Science, University of Toronto, Tech. Rep. CRG-TR-93-1, 1993.

[6] D. MacKay, "Information-based objective functions for active data selection." *Neural Computation*, vol. 4, no. 4, pp. 590–604, 1992.

[7] H. J. S. Feder, J. J. Leonard, and C. M. Smith, "Adaptive mobile robot navigation and mapping," *Inter. J. of Robotics Research, Special Issue on Field and Service Robotics*, vol. 18, no. 7, pp. 650–668, July 1999.

[8] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[9] M. L. Littman, "Algorithms for sequential decision making," Ph.D. dissertation, Brown University, Providence, RI, March 1996.

[10] N. Roy, G. Gordon, and S. Thrun, "Finding approximate pomdp solutions through belief compression," *Journal of Artificial Intelligence Research*, vol. 23, pp. 1–40, 2005.

[11] P. Poupart and C. Boutilier, "Value-directed compression of POMDPs," in *Advances in Neural Information Processing Systems 15 (NIPS)*, S. Becker, S. Thrun, and K. Obermayer, Eds., 2002.

[12] H.-T. Cheng, "Algorithms for partially observable Markov decision processes," Ph.D. dissertation, University of British Columbia, 1988.

[13] P. Poupart and C. Boutilier, "Vdcbpi: an approximate scalable algorithm for large scale pomdps," in *Advances in Neural Information Processing Systems 17 (NIPS)*, Vancouver, BC, 2004, pp. 1081–1088.

[14] T. Smith and R. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[15] M. T. J. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

[16] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[17] J. Pineau and S. Thrun, "An integrated approach to hierarchy and abstraction for POMDPs," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-02-21, August 2002.

[18] E. Wan and R. van der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proc. of IEEE Symposium 2000 (AS-SPCC)*, 2000.