The Belief Roadmap: Efficient Planning in Linear POMDPs by Factoring the Covariance

Sam Prentice[1] and Nicholas Roy[2]

[1] Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, Cambridge, MA. prentice@mit.edu
[2] Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, Cambridge, MA. nickroy@mit.edu

Summary. In this paper we address the problem of trajectory planning with imperfect state information. In many real-world domains, the position of a mobile agent cannot be known perfectly; instead, the agent maintains a probabilistic belief about its position. Planning in these domains requires computing the best trajectory through the space of possible beliefs. We show that planning in belief space can be done efficiently for linear Gaussian systems by using a factored form of the covariance matrix. This factored form allows several prediction and measurement steps to be combined into a single linear transfer function, leading to very efficient posterior belief prediction during planning. We give a belief-space variant of the Probabilistic Roadmap algorithm called the Belief Roadmap (BRM) and show that the BRM can compute plans substantially faster than conventional belief space planning. We also show performance results for planning a path across MIT campus without perfect localization.

1 Introduction

Sequential decision making with incomplete state information is an essential ability for most real world autonomous systems. For example, robots without perfect state information can use probabilistic inference to compute a distribution over possible states from sensor measurements, leading to robust state estimation. Incorporating knowledge of the uncertainty of this state distribution, or belief, into the planning process can similarly lead to increased robustness and improved performance of the autonomous system. Unfortunately, despite the recent growth of approximation algorithms and considerable progress in the applicability of algorithms such as the partially observable Markov decision process (POMDP) [1, 2], planning in belief space has met limited success in addressing large real-world problems. These models almost always rely on a discrete representation of the agent state, dynamics and perception and finding a plan usually requires optimizing a policy across the entire belief space, inevitably leading to problems with scalability.

In contrast, the motion planning community has realized considerable success in using direct search to find paths through configuration space with algorithms such as the Probabilistic Roadmap [3] or Rapidly-Exploring Randomized Trees [4]. However, adapting these search algorithms to belief space has generally not been feasible. Computing the reachable part of belief space can be expensive; predicting the evolution of the agent's belief over time involves costly non-linear operations such as matrix inversions. Secondly, the reachable belief space depends on the initial conditions of the robot and must be re-computed when the robot's state estimate changes. Therefore, any work done in predicting the effect of a sequence of actions through belief space must be completely reproduced for a query from a new start position.

In this paper, we present a formulation for planning in belief space which allows us to compute the reachable belief space and find minimum expected cost paths efficiently. Our formulation is inspired by the Probabilistic Roadmap, and we show how a graph representation of the reachable belief space can be constructed for an initial query and then re-used for future queries. We develop this formulation using the Kalman filter, a common form of linear Gaussian state estimation. We first provide results from linear filtering theory and optimal control [5] showing that the covariance of the Kalman filter can be factored, leading to a linear update step in the belief representation. As a result, the mean and covariance resulting from a *sequence* of actions and observations can be combined into a single prediction step for planning. The factored form allows an initial graph of the reachable belief space to be computed while avoiding expensive non-linear computations, and the graph can be efficiently updated for additional queries based on new initial conditions. Optimal paths can therefore be found in time linear with the size of the graph, leading to greatly accelerated planning times compared to existing techniques. We give experimental results demonstrating this algorithm for motion planning of a mobile robot.

2 Trajectory Planning and Search

Given a map, model of robot kinematics, and the start and goal positions, the goal of trajectory planning is to find the minimum-cost collision-free path from start to goal. We will restrict the discussion in this paper to kinematic motion planning; we plan to extend this work to kinodynamic planning in future work. $\mathcal{C}$ denotes the configuration space [9], the space of all robot poses, $\mathcal{C}_{free}$ is the set of all collision-free poses (based on the map of obstacle positions) and $\mathcal{C}_{obst}$ is the set of poses resulting in collision with obstacles, so that $\mathcal{C} \equiv \mathcal{C}_{free} \cup \mathcal{C}_{obst}$. When the state is fully observable, the Probabilistic Roadmap (PRM) algorithm [3] can be used to find a path through $\mathcal{C}_{free}$ by generating a discrete graph approximation of $\mathcal{C}_{free}$. The PRM computes vertices in the graph by sampling poses from $\mathcal{C}$ and rejecting those samples that lie in $\mathcal{C}_{obst}$ (i.e., that collide with obstacles). Edges in the graph are placed between mutually-visible vertices, i.e., where a straight-line path between the vertices also lies entirely in $\mathcal{C}_{free}$. The PRM then finds a feasible path by searching through the graph from the start vertex to the goal vertex. The power of the PRM resides in the fact that even if $\mathcal{C}_{free}$ cannot be tractably computed, it is relatively efficient to determine if an arbitrary vertex or edge lies in $\mathcal{C}_{free}$.

3 Belief Estimation in Linear Gaussian Systems

If, however, the agent does not have access to perfect state information, it cannot plan robustly in the configuration space and must instead plan in the space of beliefs, or possible distributions over its state. Let us denote the (unknown) state of the vehicle at time t as s_t. If the vehicle takes an action according to some control u_t, then at time $t+1$ the vehicle has moved to some new state s_{t+1} that is drawn stochastically according to some transition probability distribution $p(s_{t+1}|s_t, u_t)$. After each motion, the vehicle receives an observation z_t that is drawn stochastically according to some observation probability distribution $p(z_t|s_t)$. With knowledge of the transition and observation distributions, the vehicle can estimate the probability of its current state $b_t = p(s_t|u_{1:t}, z_{1:t})$ after a sequence of controls and observations.

A common assumption is that the posterior state s_t after some control input u_t depends only on the prior state s_{t-1} such that $p(s_t|s_{t-1}, u_{1:t}, z_{1:t-1}) = p(s_t|s_{t-1}, u_t)$. Similarly, the likelihood of an observation depends only on the current state, $p(z_t|s_t, u_{1:t}, z_{1:t-1}) = p(z_t|s_t)$. These assumptions allow the posterior belief to be expressed as

$$b_t = p(s_t|u_{1:t}, z_{1:t}) = \frac{1}{Z} p(z_t|s_t) \int_S p(s_t|u_t, s_{t-1}) p(s_{t-1}) ds_{t-1}, \tag{1}$$

where Z is a normalization factor. Equation (1) is the standard Bayes' filter equation, and provides an efficient, recursive form of updating the state distribution.

Implementing the Bayes' filter requires committing to a specific representation of the state distribution $p(s_t)$, with consequences on how the transition $p(s_t|s_{t-1}, u_t)$ and observation $p(z_t|s_t)$ functions are represented. One of the most common representations is the Kalman filter [6], in which the state distribution is assumed to be Gaussian and the transition and observation functions are linear with Gaussian noise. If the true system transition and observation functions are non-linear, the extended Kalman filter (EKF) [7] linearizes the transition and observation functions at each step. A full derivation of the EKF is outside the scope of this paper, but briefly, the assumption is that

$$s_t = g(s_{t-1}, u_t, w_t), \qquad w_t \sim \mathcal{N}(0, W_t), \tag{2}$$

$$\text{and} \qquad z_t = h(s_t, q_t), \qquad q_t \sim \mathcal{N}(0, Q_t), \tag{3}$$

where w_t and q_t are random, unobservable noise variables. In the presence of this unobserved noise, the EKF estimates the state at time t from the estimate at time $t-1$ in two separate steps: a process step based only on the control input u_t leading to an estimate $p(\overline{s}_t) = \mathcal{N}(\overline{\mu}_t, \overline{\Sigma}_t)$, and a measurement step to complete the estimate of $p(s_t)$. The process step follows as

$$\overline{\mu}_t = g(\mu_{t-1}, u_t) \tag{4}$$

$$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t W_t V_t^T, \tag{5}$$

where G_t is the Jacobian of g with respect to s and V_t is the Jacobian of g with respect to w. For convenience, we denote $R_t \triangleq V_t W_t V_t^T$. Similarly, the measurement step updates the belief as follows:

$$\mu_t = \overline{\mu_t} + K_t(H_t \overline{\mu}_t - z_t) \tag{6}$$

$$\Sigma_t = (I - K_t H_t) \overline{\Sigma}_t, \tag{7}$$

where H_t is the Jacobian of h with respect to s and K_t is known as the Kalman gain,

$$K_t = \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t \right)^{-1}. \tag{8}$$

If the measurement function is conditionally independent of a large number of state variables, the information form of the EKF may be more computationally efficient. The distribution $p(s_t|u_{1:t}, z_{1:t})$ can be represented by the information vector and the information matrix $\Omega_t = \Sigma_t^{-1}$ [8]. The information matrix updates can be written as

$$\overline{\Omega}_t = \overline{\Sigma}_t^{-1} = \left(G_t \Sigma_{t-1} G_t^T + R_t \right)^{-1} \tag{9}$$

$$\Omega_t = \overline{\Omega}_t + H_t^T Q_t^{-1} H_t. \tag{10}$$

For convenience, we denote $M_t \triangleq H_t^T Q_t^{-1} H_t$ such that $\Omega_t = \overline{\Omega}_t + M_t$.

4 Belief Space Planning

The goal in extending motion planning to belief spaces is to enable the planner to make decisions not just based on the mean of the distribution, as in the conventional PRM planner, but also by incorporating additional statistics of the distribution. If the planner has access to both the mean and the covariance of the EKF, different plans can be computed depending on whether the position estimate is very certain (the norm of the covariance is small), or if the position estimate is uncertain (the norm of the covariance is large). The robot can then detour from shorter paths for those with greater potential to reduce belief uncertainty through sensor information gain, leading to more conservative motion plans. In order to incorporate the covariance into the decision making, the planner must find a sequence of actions $\{u_0, \ldots, u_t\}$ such that the resulting beliefs $\{b_0, \ldots, b_t\}$ maximize the objective function J of the robot.

Conventional motion planners generally search for a collision-free path that minimizes the distance to the goal location, such that $J(s_t) = ||s_t - s_{goal}||$. However, planning in a Gaussian belief space requires a different objective function since every belief has some non-zero probability that the robot is at the goal state (although this probability may be extremely small for beliefs where the mean is far from the goal). A more appropriate objective function is therefore to maximize the *probability* of the goal state, such that $J(b_t) = \sum_{s \in \mathcal{S}} b_t(s) ||s - s_{goal}||$. A trivial extension of the PRM for solving this optimization problem would first generate a graph by sampling belief nodes randomly and creating edges between nodes where an action exists to move the robot from one belief to another. Graph search would then find a trajectory to the belief with highest probability of being at the goal.

The difficulty with this approach is that the control problem is underactuated and, thus, only a specific subset of beliefs $\mathcal{B}^*$ is actually realizable. Even if the robot has full control of the mean of its belief, the covariance evolves as a complicated, non-linear function of both the robot controls and environmental observations. If the robot has full control over its n-dimensional mean, then the reachable part of the belief space is an n-dimensional manifold embedded in the n^3-dimensional belief space, and therefore a subset of measure 0. It is vanishingly unlikely that any belief $b' \in \mathcal{B}^*$ would ever be sampled such that there exists a control u to reach it.

A different approach must therefore be used for building the belief graph. Since the robot does have control authority over its mean, it is possible to sample mean components of the belief, then *predict* the corresponding covariance components. Let us sample a set of mean poses $\{\mu_i\}$ from $\mathcal{C}_{free}$ as the nodes in the PRM graph. We add an edge e_{ij} between pairs (μ_i, μ_j) if a sequence of controls $u_{ij} = \{u_{t_i}, \ldots, u_{t_j}\}$ exists to move along the straight line between μ_i and μ_j without collision. We then perform search in the graph to find a sequence of controls starting from the initial belief b_0 such that the posterior covariance at the end of the sequence is minimized, computing this posterior covariance using equations (5) and (7). For each step along the edge e_{ij}, the G_t, R_t and M_t matrices are computed using the appropriate models.

5 Linearizing Belief Space Planning

A major computational bottleneck with the planning algorithm described above is that standard search optimization techniques cannot be used, such as re-using portions of previous solutions. While the EKF is an efficient model for tracking the probability distribution of both linear and well-behaved non-linear systems, the update steps in the filtering process itself are non-linear. In particular, any trajectory optimization that depends on the covariance requires solving the following Riccati equation (from equations 5 and 7):

$$\begin{aligned}\Sigma_t = {} & (G_t \Sigma_{t-1} G_t^T + R_t) \\ & - (G_t \Sigma_{t-1} G_t^T + R_t) H_t^T \left(H_t (G_t \Sigma_{t-1} G_t^T + R_t) H_t^T + Q_t\right)^{-1} H_t (G_t \Sigma_{t-1} G_t^T + R_t).\end{aligned}$$

As a result, if the intial conditions or the trajectory itself are modified in any way, the covariance must be recomputed from scratch. If the planner computes a predicted posterior state (μ_t, Σ_t) from an initial distribution (μ_0, Σ_0) and a predicted sequence of actions and observations using a set of t EKF updates, the non-linearity prevents us from computing a new posterior state (μ_t', Σ_t') from a different set of initial conditions (μ_0', Σ_0'), except by repeating the entire sequence of t EKF updates. This is *not* the case for the mean μ_t for most real-world systems; for a sequence of controls $\{u_0, \ldots, u_t\}$, under some reasonably mild assumptions, once μ_t is calculated from μ_0, a new μ_t' can be calculated in a single step from a different μ_0'. The EKF update of the mean becomes linear during predictive planning when the measurement z_t is assumed to be the maximum likelihood observation $z_t = H_t \overline{\mu}_t$, which simplifies equation 6 to $\mu_t = \overline{\mu}_t$.

For a trajectory formed from a sequence of k graph edges each of length l, $\mathcal{O}(kl)$ EKF process and measurement updates are required. The asymptotic complexity of the overall problem is $\mathcal{O}(lb^d)$ for a search depth of d edges in the graph with a branching factor of b; the computational cost of the specific EKF updates along each edge may seem negligible as a constant multiplier of the exponential growth, but this term has a significant effect on the overall time to plan. However, if the covariance is factored appropriately, we can show that the EKF update equations for each factor separately are in fact linear. Along with other benefits, the linearity will allow us to combine multiple EKF updates into a single transfer function, ζ_{ij}, associated with each edge e_{ij}, to efficiently predict the posterior filter state from a sequence of controls and measurements in a single step. Although initial construction of the graph and transfer functions requires a cost of $\mathcal{O}(l)$ per edge, this construction cost can be amortized, leading to a planning complexity of $\mathcal{O}(b^d)$, equivalent to the fully-observable case.

5.1 Linear Covariance Updates

To show the linearization of the EKF covariance update, we rely on previous results from linear filtering theory and optimal control [5] to make use of the following matrix inversion lemma:

Lemma 1.

$$(A + BC^{-1})^{-1} = (ACC^{-1} + BC^{-1})^{-1} = C(AC + B)^{-1} \tag{11}$$

Theorem 1. *The covariance can be factored as $\Sigma = BC^{-1}$, where the combined EKF process and measurement update gives B_t and C_t as linear functions of B_{t-1} and C_{t-1}.*

Proof. We proceed by proof by induction.

Base case: We can show the theorem to be trivially true, as

$$\Sigma_0 = B_0 C_0^{-1} = \Sigma_0 I^{-1}. \tag{12}$$

Induction step:

$$\begin{aligned}
\text{Given: } \Sigma_{t-1} &= B_{t-1}C_{t-1}^{-1} && (13)\\
\text{From equation (5) , } \overline{\Sigma}_t &= G_t B_{t-1} C_{t-1}^{-1} G_t^T + R_t \\
\overline{\Sigma}_t &= (G_t B_{t-1})(G_t^{-T} C_{t-1})^{-1} + R_t && (14)\\
\text{From lemma 1, } \overline{\Sigma}_t &= \left((G_t^{-T} C_{t-1}) \left(G_t B_{t-1} + R_t (G_t^{-T} C_{t-1}) \right)^{-1} \right)^{-1} \\
\overline{\Sigma}_t &= \left(\overline{D}_t \overline{E}_t^{-1} \right)^{-1} && (15)\\
\Rightarrow \overline{\Sigma}_t &= \overline{E}_t \overline{D}_t^{-1}, && (16)
\end{aligned}$$

where $\overline{D}_t = G_t^{-T} C_{t-1}$ and $\overline{E}_t = G_t B_{t-1} + R_t (G_t^{-T} C_{t-1})$. As a result, we can see that the process update preserves the factored form of Σ. Similarly, if we start with the information form for the covariance update,

$$\begin{aligned}
\text{From equation (10), } \Sigma_t &= (\overline{\Sigma}_t^{-1} + H_t^T Q_t^{-1} H_t^T)^{-1} && (17)\\
\text{Substituting in } M_t \text{ and equation (16), } \Sigma_t &= (\overline{D}_t \overline{E}_t^{-1} + M_t)^{-1} && (18)\\
\text{Again from lemma 1, } \Sigma_t &= \overline{E}_t (\overline{D}_t + M_t \overline{E}_t)^{-1} && (19)\\
\Rightarrow \Sigma_t &= B_t C_t^{-1}, && (20)
\end{aligned}$$

where $B_t = \overline{E}_t$ and $C_t = \overline{D}_t + M_t \overline{E}_t$. If we collect terms, we see that

Algorithm 1 The Belief Roadmap Build Process.

Require: Map $\mathcal{C}$ over mean robot poses
1: Sample mean poses $\{\mu_i\}$ from $\mathcal{C}_{free}$ using a standard PRM sampling strategy to build belief graph node set $\{n_i\}$ such that $n_i = \{\mu = \mu_i\}$
2: Create edge set $\{e_{ij}\}$ between nodes (n_i, n_j) if the straight-line path between $(n_i[\mu], n_j[\mu])$ is collision-free
3: Build one-step transfer functions $\{\zeta_{ij}\}\ \forall\, e_{ij} \in \{e_{ij}\}$
4: **return** Belief graph $\mathcal{G} = \{\{n_i\}, \{e_{ij}\}, \{\zeta_{ij}\}\}$

$$B_t = \overline{E}_t = G_t B_{t-1} + R_t(G_t^{-T} C_{t-1}) \tag{21}$$

and

$$C_t = \overline{D}_t + M_t \overline{E}_t = G_t^{-T} C_{t-1} + M_t(G_t B_{t-1} + R_t(G_t^{-T} C_{t-1})). \tag{22}$$

In both cases, B_t and C_t are linear functions of B_{t-1} and C_{t-1}.

Collecting terms again, we can re-write equations (21) and (22), such that

$$\Psi_t = \begin{bmatrix} B \\ C \end{bmatrix}_t = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix}_t \begin{bmatrix} B \\ C \end{bmatrix}_{t-1} \tag{23}$$

$$= \begin{bmatrix} 0 & I \\ I & M \end{bmatrix}_t \begin{bmatrix} 0 & G^{-T} \\ G & RG^{-T} \end{bmatrix}_t \begin{bmatrix} B \\ C \end{bmatrix}_{t-1}, \tag{24}$$

where Ψ_t is the stacked block matrix $\left[\begin{smallmatrix} B \\ C \end{smallmatrix}\right]_t$ consisting of the covariance factors and $\zeta_t = \left[\begin{smallmatrix} W & X \\ Y & Z \end{smallmatrix}\right]_t$ is the one-step transfer function for the covariance factors.

5.2 The Belief Roadmap Algorithm

The belief space planning algorithm is therefore a two stage process. First, mean positions of the robot are sampled, as in the Probabilistic Roadmap algorithm, and edges between visible graph nodes are added. The corresponding process and measurement Jacobians are calculated along each edge and assembled via matrix multiplication into a one-step transfer function for the covariance, ζ_{ij}, according to equation (24). Each ζ_{ij} now allows us to compute the posterior covariance Σ_j that results at node j by starting at node i with covariance Σ_i, moving along edge e_{ij} and performing an EKF update for each observation received along that path.

In the second stage, a standard search algorithm is used to compute the sequence of edges through the graph, starting at b_0, that maximizes the probability of being at the goal (or, equivalently, results in minimal belief covariance at the goal). We call this algorithm the *Belief Roadmap* (BRM) planner. The build and search phases of the BRM planner are shown in Algorithms 1 and 2, respectively.

There are several issues that for reasons of space we will only address briefly. Firstly, note that this must be a forward search process; the terminal node of this path cannot be determined *a priori*, as the covariance (and hence $b_t(s_{goal})$) depends on the specific path.

Secondly, the BRM search process in Algorithm 2 assumes a queue function that orders the expansion of (μ, Σ) nodes. Breadth-first search sorts the nodes in a first-in, first-out order. More intelligent search processes rely on an $A^\star$ heuristic to find

Algorithm 2 The Belief Roadmap Search Process.

Require: Start belief (μ_0, Σ_0), goal location μ_{goal} and belief graph $\mathcal{G}$
1: Append $\mathcal{G}$ with nodes $\{n_0, n_{goal}\}$, edges $\{\{e_{0,j}\}, \{e_{i,goal}\}\}$, and one-step transfer functions $\{\{\zeta_{0,j}\}, \{\zeta_{i,goal}\}\}$
2: Augment node structure with best path p=$\emptyset$ and covariance Σ=$\emptyset$, such that n_i=$\{\mu, \Sigma, p\}$
3: Create search queue with initial position and covariance $Q \leftarrow n_0 = \{\mu_0, \Sigma_0, \emptyset\}$
4: **while** Q is not empty **do**
5:   Pop $n \leftarrow Q$
6:   **if** $n = n_{goal}$ **then**
7:     Continue
8:   **end if**
9:   **for all** n' such that $\exists e_{n,n'}$ **and not** $n' \ni n[p]$ **do**
10:     Compute one-step update $\Psi' = \zeta_{n,n'} \cdot \Psi$, where $\Psi = \begin{bmatrix} n[\Sigma] \\ I \end{bmatrix}$
11:     $\Sigma' \leftarrow \Psi'_{11} \cdot {\Psi'_{21}}^{-1}$
12:     **if** $tr(\Sigma') < tr(n'[\Sigma])$ **then**
13:       $n' \leftarrow \{n'[\mu], \Sigma', n[p] \cup \{n'\}\}$
14:       Push $n' \rightarrow Q$
15:     **end if**
16:   **end for**
17: **end while**
18: **return** $n_{goal}[p]$

the goal state faster; however, good $A^\star$ heuristics for planning in belief space are a topic for future work. For the results given in this paper, we used exclusively breadth-first search for both shortest-path (PRM) and belief-space (BRM) planning problems. Additionally, considerable work has been devoted to finding good sampling strategies in fully-observable motion planing problems. For the results in this paper, we used the same medial-axis sampling strategy for both the shortest-path (PRM) and belief-space (BRM) planning problems. However, it is likely that better belief-space planning would result from sampling strategies that are aware of the sensor model. Similarly, a sampling stategy that incorporates the cost function would also lead to improved planning, especially for cost functions that are not solely a function of the distribution over the goal state. By iteratively computing expected costs and re-sampling the roadmap, an upper-bound on the expected cost of the entire computed plan can be achieved. The exact algorithm for iteratively planning-resampling is outside the scope of this paper.

Thirdly, note in Algorithm 2 that we only expand nodes where the search process has not already found a posterior covariance $n'[\Sigma]$ such that some measure of uncertainty such as the trace or determinant is less than the measure of the new posterior covariance Σ'. It is also assumed that a node n' replaces any current queue member n' when pushed onto the queue.

Finally, the BRM search process can be adapted to optimize different objective functions. One alternative minimizes the maximum covariance encountered along the entire path, rather than at the goal location. This is implemented by augmenting the node structure (in line 2) with the maximum covariance Σ^p_{max} along path $n[p]$, such that $n_i = \{\mu, \Sigma, p, \Sigma^p_{max}{=}\infty\}$. The decision-making step in lines 12-13 is then

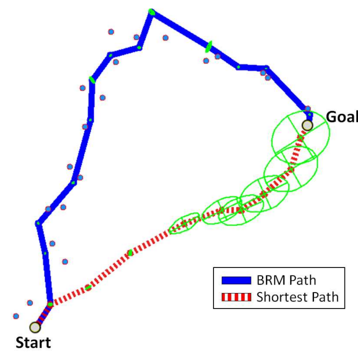


Figure 1. Experimental Setup. Range sensor locations (shown as blue circles) were sampled along randomized trajectories between a start and goal location. The solid and dashed line show the plans generated by the BRM and shortest path algorithms, respectively, with green ellipses indicating the covariances Σ along each trajectory. This experimental design tests the ability of the BRM to find paths with greater potential information gain to stay well-localized during execution.

replaced as follows:

$$\textbf{if } max(tr(\Sigma'), tr(n[\Sigma^p_{max}])) < tr(n'[\Sigma^p_{max}]) \textbf{ then}$$
$$n' \leftarrow \{n'[\mu], \Sigma', n[p] \cup \{n'\}, max(\Sigma', n[\Sigma^p_{max}])\}.$$

6 Experimental Results

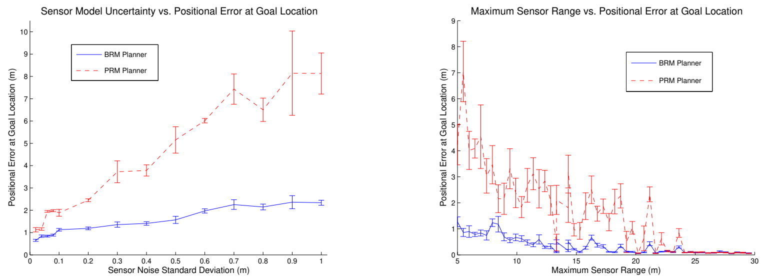
In order to evaluate the BRM algorithm, we performed a series of evaluations on a small planning domain in simulation. The testing consisted of two objectives: (1) to evaluate the quality of plans produced by the BRM algorithm in terms of uncertainty reduction; and (2), to assess the computational advantage of employing the linearized EKF covariance update during the search process.

The experimental setup consisted of small-sized maps with randomly placed ranging beacons using a generalized range sensor model (which, in practice, could be implemented as GPS or RF range sensors). The range measurement was modelled as the Euclidean distance, d, between the mobile robot and each visible sensor, with additive, distance-varying Gaussian bias and random error components, $\mathcal{N}(\mu_{bias}(d), \sigma_{bias}(d)^2)$.

The environment was assumed to be free of obstacles to avoid experimental bias resulting from artifacts in sensor measurements and random trajectory graph generation in environments with varying contours.

Localization Performance:

In the first set of analyses, we compared the quality of paths produced using the BRM algorithm to those resulting from a shortest path search. In each test iteration, sensor locations were sampled along randomized trajectories between a start and goal location in an environment with $100m$ sides, as shown in Figure 1. This experimental setup tests the ability of the BRM to detour from shorter paths for those with lower



(a) Varying Sensor Noise (b) Varying Sensor Range

Figure 2. In these figures we characterized the positional accuracy of the robot for each planner as a function of the sensor noise and sensor range. (a) Accuracy vs. Sensor Noise. The positional accuracy of the shortest path algorithm suffered with increased noise. The positional accuracy of the BRM increased slightly but not substantially with noise. (b) Accuracy vs. Range. The positional accuracy of the shortest path algorithm increased with sensor range as the agent had more ranges for localization. Even with very short range, the BRM algorithm was able to find paths that maintained relatively good localization.

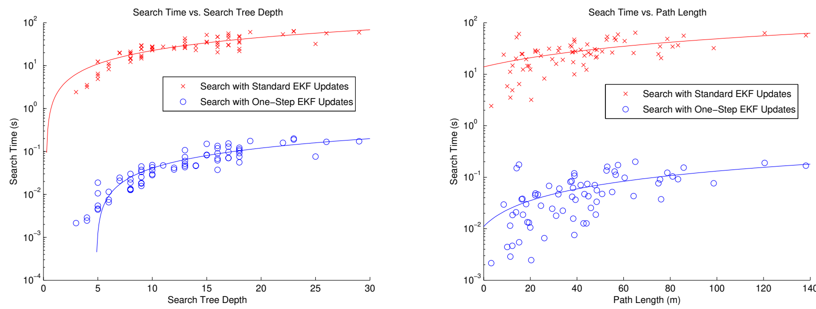
expected uncertainty. We tested the quality of paths computed by the BRM and shortest path planning algorithms by evaluating the average position error obtained at the goal location after executing the prescribed path.

We performed two analyses to demonstrate that the BRM provided more accurate localization; we artificially varied the random noise of the range beacons, and we artificially limited the range of the beacons by discarding measurements beyond a maximum range. In Figure 2(a), we see the performance of the two planning algorithms under conditions of varying noise. As the sensor noise increases, both algorithms have worsened positional accuracy at the goal, but the shortest path algorithm degrades more quickly. In Figure 2(b), we see that with a small maximum sensor range, the BRM is able to find trajectories in close proximity to sensors, yielding a reasonable level of positional accuracy. As maximum sensor range increases, trajectories farther from sensors provide sufficient information for localization and the positional errors in both planners converge to similar values.

Algorithmic Performance:

Secondly, we assessed the speed improvement of utilizing the linearized EKF covariance update during planning. We compared the time required by the planning search process when using the one-step linearized EKF covariance update to that of the standard EKF covariance updates. Planning experiments were performed using randomized sensor locations in maps of varying size ($30-100m$ per side). To reduce variability in the speed comparison results, the number of sensors was held constant throughout the experiments and the number of nodes was sampled randomly in proportion to the area of the environment to maintain consistent trajectory lengths.

Figure 3(a) shows the relative search times with respect to the depth of the search tree in the corresponding trajectory graph. The BRM maintains a consistent improve-



(a) Time vs. Tree Depth

(b) Time vs. Path Length

Figure 3. Algorithmic Performance. (a) Time to Plan vs. Tree Depth (b) Time to Plan vs. Path Length. Note that these graphs are semi-log graphs, indicating two orders of magnitude increase in speed.

ment by over two orders of magnitude, with search costs scaling logarithmically with increasing tree depth. Similar results are obtained when comparing the search times with respect to the length of the resulting path, shown in Figure 3(b), reiterating the significant scalable improvement of the one-step update. The one-step covariance update presents a consistent speedup and scales with the size of the trajectory graph, making planning in belief space with the BRM computationally tractable.

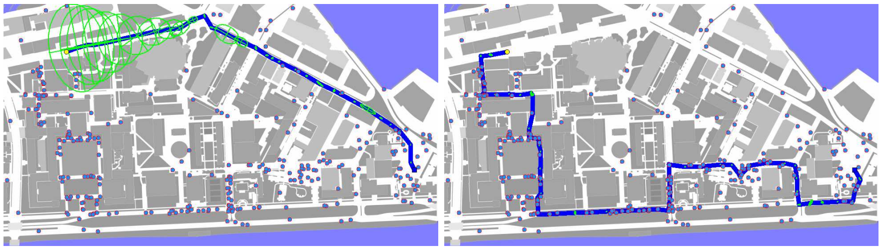
Note that to construct update matrices for each trajectory in the graph, the one-step linearized search incurs a one-time build cost comparable to the cost of *one* path search using the standard covariance model. However, this cost is amortized; the BRM reuses this graph to enable efficient search in replanning.

Example trajectories:

Finally, Figure 4 shows example trajectories for a very large planning problem. The robot must navigate across the MIT campus from the bottom right corner to the top left corner. Scattered throughout the environment are ranging beacons with known position, shown as the small blue circles. The robot can localize itself according to the ranges, but the ranging accuracy varies across campus according to the proximity and density of the beacons. The robot is also constrained to the outside paths (and cannot short-cut through buildings, the light-grey blocks). The shortest path planner shown in Figure 4(a) finds a direct route (the solid blue line) but the positional uncertainty grows quite large, shown by the green uncertainty ellipses. In contrast, the BRM algorithm finds a path that stays well-localized by finding areas with a high sensor density. The uncertainty ellipses are too small to be seen for this trajectory.

7 Conclusion

In this paper, we have addressed the problem of planning in belief space for linear-Gaussian POMDPs, where the belief is tracked using Kalman-filter style estimators. We have shown that the computational cost of EKF predictions during planning can be reduced by factoring the covariance matrix and combining multiple EKF update



(a) PRM: Shortest Path (b) BRM: Lowest Expected Uncertainty Path

Figure 4. Example paths for a mobile robot navigating across MIT campus. The solid blue line in each case is the robot path, the blue dots are the range beacons being used for localization, and the green ellipses are the covariances Σ of the robot position estimate along its trajectory. Notice that the shortest path trajectory grows very uncertain, whereas the lowest expected uncertainty path always stays well-localized at the cost of being slightly longer.

steps into a single, one-step process. We have presented a variant of the Probabilistic Roadmap algorithm, called the Belief Roadmap (BRM) planner, and shown that it substantially improves planning performance and positional accuracy. We demonstrated our planning algorithm on a large-scale environment and showed that we could plan efficiently in this large space. This kind of trajectory has been reported elsewhere [10] but in limited scales of environments. We believe that our demonstration of belief-space planning in the MIT campus environment is considerably larger than existing results.

References

1. J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.
2. M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.
3. L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
4. Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378-400, 2001.
5. D. Vaughan. A non-recursive algebraic solution for the discrete Riccati equation. *IEEE Transactions on Automatic Control*, 15(5):597-599, 1970.
6. R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering*, 82(Series D):35–45, 1960.
7. R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In Ingemar J. Cox and Gordon T. Wilfong, editors, *Autonomous Robotic Vehicles*. Springer-Verlag, Orlando, FL, 1990.
8. S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of the American Control Conference*, pages 1628–1632, 1995.
9. T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Trans. on Computers*, C-32(2):108–120, Feb. 1983.
10. Nicholas Roy and Sebastian Thrun. Coastal navigation with mobile robots. In *Advances in Neural Processing Systems 12*, volume 12, pages 1043–1049, 1999.