The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance

Samuel Prentice and Nicholas Roy

***Abstract*—When a mobile agent does not known its position perfectly, incorporating the predicted uncertainty of future position estimates into the planning process can lead to substantially better motion performance. However, planning in the space of probabilistic position estimates, or belief space, can incur substantial computational cost. In this paper, 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 conclude with performance results for an agent using ultra-wide bandwidth (UWB) radio beacons to localize and show that we can efficiently generate plans that avoid failures due to loss of accurate position estimation.**

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; the most general formulation of this problem is known as the partially observable Markov decision process (POMDP) (Kaelbling et al., 1998). Unfortunately, despite the recent development of efficient exact and approximate algorithms for solving POMDPs, planning in belief space has had limited success in addressing large real-world problems. The existing planning algorithms almost always rely on discrete representations 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 stochastic search to find paths through high-dimensional configuration spaces with algorithms such as the Probabilistic Roadmap (PRM) (Kavraki et al., 1996) or Rapidly-Exploring Randomized Trees (RRT) (LaValle and Kuffner, 2001). Some approaches have extended these techniques to allow uncertainty over the effects of actions by modelling the planning problem as a Markov Decision Process (MDP). MDP-based planning through a Probabilistic Roadmap optimizes over a stochastic action space (Alterovitz et al., 2006, 2007), but still assumes perfect knowledge of the state. More recently, Pepy et al. (2008) used an RRT-based approach to plan over a set representation of uncertain states; however, only the effects of action uncertainty were considered in state prediction, resulting in the monotonic growth of belief uncertainty without exteroceptive observations. Incorporating the full probability distribution provided by state estimation into planning algorithms such as the PRM or RRT has generally not been feasible. Computing the reachable part of belief space can be expensive; predicting the full evolution of the agent's belief over time, incorporating both stochastic actions and noisy observations, involves costly non-linear operations such as matrix inversions. Furthermore, 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.

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 (Kalman, 1960), a common form of linear Gaussian state estimation. We first provide results from linear filtering theory and optimal control (Vaughan, 1970) showing that the covariance of the Kalman filter can be factored, leading to a linear update step in the belief representation. This technique has been well-established in the optimal control and filtering literature; however, its use for prediction in planning is both novel and powerful. Using this 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 not only allows the graph of reachable belief space to be computed efficiently, but also updated online for additional queries based on new initial conditions. Optimal paths with respect to the roadmap can therefore be found in time linear with the size of the graph, leading to greatly accelerated planning times compared to existing techniques.

The specific problem we address is an agent navigating in a GPS-denied environment. The Global Positioning System (GPS) provides position estimates of a user's location on the Earth to within a few meters, enabling geolocation in areas with a clear line-of-sight (LOS) to GPS satellites, but

Samuel Prentice and Nicholas Roy are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 02139. E-mail: {prentice|nickroy}@mit.edu.

in indoor and dense urban environments, GPS becomes unreliable or altogether unavailable. One approach to mitigating the loss of GPS involves placing anchor radio beacons around the target environment, enabling localization by computing range measurements from the RF signals. However, even with spread-spectrum technologies such as ultra-wide bandwidth (UWB) radio, the quality of positional information available to a mobile agent varies across the environment. When a signal is transmitted, the channel properties of the environment may result in multiple signal path reflections; interference between these path reflections can result in signal degradation, or even cancellation, which inhibits the receiver's ability to successfully resolve the original signal for time-based ranging. Obstacles in the environment may occlude the radio transceivers, blocking signal transmission or adding substantial material propagation delays, both of which are problematic for ranging applications. We show that integrating predictions of position uncertainty into the planning algorithm enables the robot to choose plans that maximize the probability of reaching the goal, avoiding trajectories through regions of low density of UWB signals that lead to very uncertain estimates of the agent position. We give experimental results demonstrating this algorithm for motion planning of a mobile robot, in which the robot must use UWB range estimates for tracking its position. We conclude with results that show planning using a simulated UWB ranging model to navigate across MIT campus.

2. Trajectory Planning and Search

Given a map, model of robot kinematics, and the start and goal positions, the objective 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 (Lozano-Perez, 1983), 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 (Kavraki et al., 1996; Bohlin and Kavraki, 2000) can be used to find a path through $\mathcal{C}_{free}$ by generating a discrete graph approximation of $\mathcal{C}_{free}$. The PRM provides a general framework for efficiently solving fully observable motion planning problems in two stages, as follows:

1) **Pre-processing phase**: The PRM first constructs a graph, or roadmap, that is a simplified representation of $\mathcal{C}_{free}$. Robot poses are sampled from $\mathcal{C}$ according to a suitable probabilistic measure and tested to determine if each pose lies in $\mathcal{C}_{free}$ or $\mathcal{C}_{obst}$. Poses within $\mathcal{C}_{free}$ (i.e., that do not collide with obstacles) are retained and added as nodes to the graph. Edges in the graph are placed between nodes where a straight-line path between the nodes also lies entirely in $\mathcal{C}_{free}$. Typically, edges are limited to the k nearest neighbor nodes or to the neighbors that are closer than some bounding distance.
2) **Query phase**: Given a start and goal pose, a graph search algorithm is used to find a path through the graph that connects the corresponding start and goal nodes. As a result, the PRM returns a path consisting of a series of straight-line trajectories between waypoints. If the start and goal poses are not already nodes in the graph, additional nodes are added to the graph and edges to any reachable graph nodes are added as in the pre-processing phase.

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 node or edge lies in $\mathcal{C}_{free}$. As a result, when planning in high-dimensional spaces, $\mathcal{C}_{free}$ can be approximated as a discrete graph by sampling poses from $\mathcal{C}$, retaining the collision-free samples and straight-line trajectories. Note that there is an implicit assumption in using the PRM, specifically, that some controller exists that can be used to follow a straight-line trajectory from way-point to way-point without collisions when the edge is known to lie entirely in $\mathcal{C}_{free}$. It should also be noted that any sampling-based roadmap approach inherently sacrifices guaranteed global optimality for the computational tractability of solutions within the discrete graph approximation. "Optimal" plans within the discrete roadmap are approximations of optimal trajectories in the continuous state space, which converge as the size of the roadmap grows.

3. Belief Estimation in Linear Gaussian Systems

When the agent does not have access to near-perfect information about its state in the world using some external positioning system such as GPS, the agent can infer its position from a history of sensor measurements and control actions. The sensor measurements constitute observations of the environment, and can be matched against a prior model of the world, such as a map. Typically, these observations are noisy and imperfect; as a result, statistical inference can be used to maintain a distribution over possible positions, essentially averaging out errors in the observations over time. While the inference results in a probability distribution over the agent's state, a common assumption is that the maximum likelihood state under the distribution can be used in place of the true state when executing a plan.

Let us denote the (unknown) state of the agent at time t as s_t. If the agent takes an action according to some control u_t, then at time $t+1$ the agent 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 agent 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 agent 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 b_t to be computed recursively as

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

where Z is a normalization factor. Equation (1) is the standard Bayes' filter equation, and provides a 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, and on the tractability of performing the integration in Equation (1). One of the most common representations is the Kalman filter (Kalman, 1960), 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) (Smith et al., 1990) 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

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

and

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

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

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

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

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:

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

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

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

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

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 $\Omega_t = \Sigma_t^{-1}$ (Julier et al., 1995). The information matrix updates in the process and measurement steps of the extended information filter (EIF) can be written respectively as

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

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

For convenience, we use $M_t \triangleq H_t^T Q_t^{-1} H_t$ to denote the information matrix corresponding to a given measurement, such that $\Omega_t = \overline{\Omega}_t + M_t$.

4. Belief Space Planning

The assumption of most planning and control algorithms is that knowledge of the mean of the belief distribution is sufficient for good performance. However, if the planner uses both the mean and the covariance of the belief in choosing actions, different plans can be computed depending on whether the position estimate is very certain (for example, a norm of the covariance is small), or if the position estimate is uncertain (a norm of the covariance is large). The robot can then balance shorter paths against those with greater potential to reduce belief uncertainty through sensor information gain, choosing longer but more conservative motion plans when appropriate. Notice that by incorporating additional statistics from the belief such as the covariance, we are doing nothing more than increasing the state space of the agent. Instead of planning in configuration space, the agent is now planning in *belief space* $\mathcal{B}$, or *information space*, but the basic problem is essentially the same: 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 collision-free paths that minimize the cost of moving to the goal location, such that

$$J(s_t) = \min_{u_{0:T}} C(s_t - s_{goal}) + \sum_t^T c(s_t, u_t), \quad (11)$$

where C is the cost of the distance to the goal location and c is the cost of executing control u_t from state s_t.

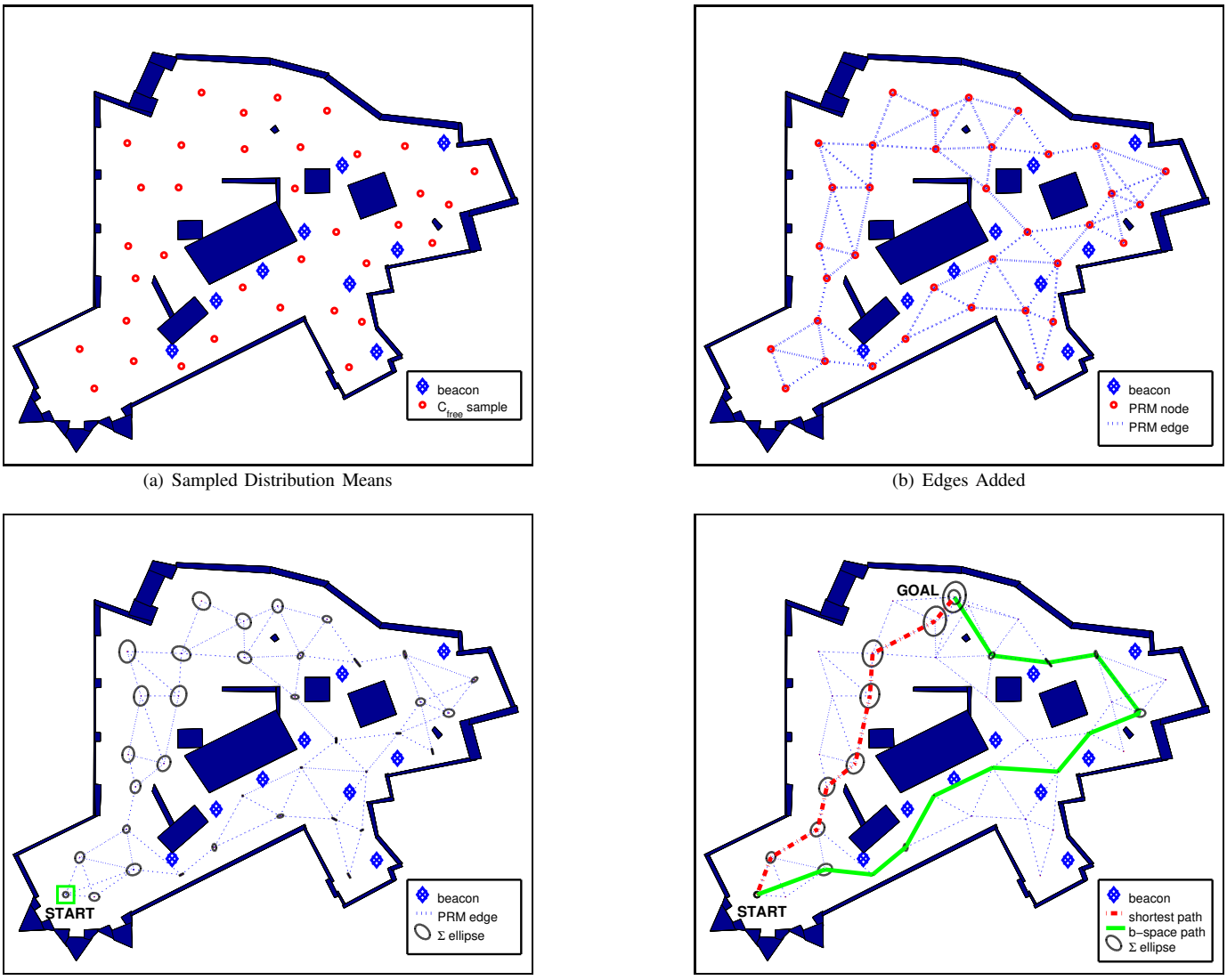
Planning in a Gaussian belief space requires a different objective function since every belief has some non-zero probability that the robot is at the goal state (although this probability may be extremely small for beliefs where the mean is far from the goal). A more appropriate objective function is therefore to minimize the *expected* cost of the path, $E_{s_{0:T}}[J(s_t)]$. In practice, the effect of uncertainty along the trajectory is dominated by the effect of uncertainty at the goal, allowing us to approximate the cost with a more efficient cost function:

$$J(b_t) \approx \min_{u_{0:T}} E_{b_{goal}|b_t, u_{0:T}}[C(s_t - s_{goal})] + \sum_t^T c(b_t, u_t), \quad (12)$$

where the expectation is taken with respect to the belief state at the 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 under-actuated 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



(a) Sampled Distribution Means

(b) Edges Added

(c) Resulting Belief Space Graph

(d) Advantage of Belief Space Planning

Fig. 1. A basic example of building a belief-space roadmap in an environment with ranging beacons. (a) Distribution means are sampled, and the means in C_{free} are kept. (b) Edges between distributions that lie in C_{free} are added to the graph. (c) Once the graph is built, an initial belief (lower right, labelled START) can be propagated through the graph by simulating the agent's motion and sensor measurements, and performing the appropriate filter update steps in sequence along edges. The posterior distribution at each node is drawn with $1-\sigma$ uncertainty ellipses, and results from a single-source, minimum uncertainty search path to that node. In this example, we artificially increased the noise in the robot motion to make the positional uncertainty clearly visible throughout the environment. Figure (d) reiterates the benefit of incorporating the full belief distribution in planning. The belief space planner detours from the shortest path through an sensing-rich portion of the environment to remain well-localized.

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. This process is shown in Figure 1. Let us sample a set of mean poses $\{\mu_i\}$ from $\mathcal{C}_{free}$ as the nodes in the PRM graph (Figure 1a). 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 (Figure 1b). We then simulate a sequence of controls and measurements along each edge; for each step along the edge e_{ij}, the G_t, R_t and M_t matrices are computed using the appropriate models. Finally, we 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). Figure 1(c) shows the result of the search process and the minimum covariance achieved at each graph node. Figure 1(d) demonstrates the advantage of planning in belief space. The belief space planner detours from the shortest path, finding a trajectory that is rich with sensor information to enable superior localization.

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)):

$$\Sigma_t = (G_t\Sigma_{t-1}G_t^T+R_t) - (G_t\Sigma_{t-1}G_t^T+R_t)H_t^T\times (H_t(G_t\Sigma_{t-1}G_t^T+R_t)H_t^T+Q_t)^{-1}H_t(G_t\Sigma_{t-1}G_t^T+R_t). \quad (13)$$

As a result, if the initial 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 $z_t = h(\overline{\mu}_t)$, 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.

The one-step covariance transfer function ζ_{ij} is demonstrated in Figure 2. In the top, standard EKF updates are used to predict the evolution of an initial belief (μ_0, Σ_0) along edge e_{ij}. The belief is updated in multiple filter steps across the edge; the mean μ propagates linearly (with the assumptions stated above), while the covariance requires computing a corresponding series of non-linear updates according to Equation (13). The posterior belief (μ_T, Σ_T) resulting from the initial belief (μ_0, Σ_0) is recovered after T update steps, and must be fully re-computed for a change in initial conditions. The one-step transfer function is shown in the bottom of Figure 2. An initial belief at node i is propagated through ζ_{ij} in one efficient step to recover the posterior covariance at node j. The following section derives this transfer function by showing that EKF covariance updates can be composed.

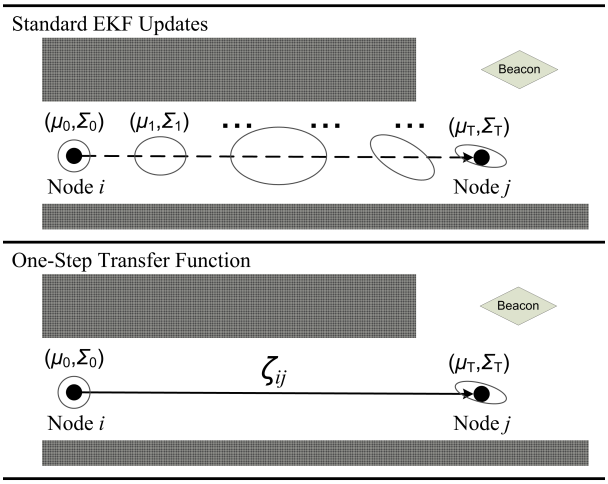


Fig. 2. Belief Prediction With One-Step Covariance Transfer Function. Nodes i and j are connected by an edge in C_{free}; a beacon (diamond) provides range measurements. (Top) Standard EKF updates are used in succession to propagate the initial covariance Σ_0 along edge e_{ij} in multiple filtering steps. (Bottom) The one-step transfer function ζ_{ij} encodes the effects of multiple EKF process and measurement covariance updates, enabling the posterior covariance Σ_T to be computed in one efficient step given any novel initial covariance Σ_0.

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 (Vaughan, 1970) to make use of the following matrix inversion lemma:

Lemma 1.

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

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

$$\Sigma_0 = B_0C_0^{-1} = \Sigma_0 I^{-1}. \quad (14)$$

Induction step:

$$\text{Given: } \Sigma_{t-1} = B_{t-1}C_{t-1}^{-1} \quad (15)$$

From equation (5),

$$\overline{\Sigma}_t = G_tB_{t-1}C_{t-1}^{-1}G_t^T + R_t \quad (16)$$

$$\overline{\Sigma}_t = (G_tB_{t-1})(G_t^{-T}C_{t-1})^{-1} + R_t \quad (17)$$

From lemma 1,

$$\overline{\Sigma}_t = \Big((G_t^{-T}C_{t-1})\big(G_tB_{t-1}+R_t(G_t^{-T}C_{t-1})\big)^{-1}\Big)^{-1} \quad (18)$$

$$\overline{\Sigma}_t = \left(\overline{D}_t\overline{E}_t^{-1}\right)^{-1} \quad (19)$$

$$\Rightarrow \overline{\Sigma}_t = \overline{E}_t\overline{D}_t^{-1}, \quad (20)$$

where $\overline{D}_t = G_t^{-T}C_{t-1}$ and $\overline{E}_t = G_tB_{t-1} + R_t(G_t^{-T}C_{t-1})$. 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,

From equation (10),

$$\Sigma_t = (\overline{\Sigma}_t^{-1} + H_t^TQ_t^{-1}H_t^T)^{-1} \quad (21)$$

Substituting in M_t and equation (20),

$$\Sigma_t = (\overline{D}_t\overline{E}_t^{-1} + M_t)^{-1} \quad (22)$$

Again from lemma 1,

$$\Sigma_t = \overline{E}_t(\overline{D}_t + M_t\overline{E}_t)^{-1} \quad (23)$$

$$\Rightarrow \Sigma_t = B_tC_t^{-1}, \quad (24)$$

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

$$B_t = \overline{E}_t = G_tB_{t-1} + R_t(G_t^{-T}C_{t-1}) \quad (25)$$

and

$$C_t = \overline{D}_t + M_t\overline{E}_t \quad (26)$$

$$= G_t^{-T}C_{t-1} + M_t(G_tB_{t-1} + R_t(G_t^{-T}C_{t-1})). \quad (27)$$

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 (25) and (26), such that

$$\Psi_t = \begin{bmatrix}B\\C\end{bmatrix}_t = \begin{bmatrix}W & X\\Y & Z\end{bmatrix}_t\begin{bmatrix}B\\C\end{bmatrix}_{t-1} \quad (28)$$

$$= \begin{bmatrix}0 & I\\I & M\end{bmatrix}_t\begin{bmatrix}0 & G^{-T}\\G & RG^{-T}\end{bmatrix}_t\begin{bmatrix}B\\C\end{bmatrix}_{t-1}, \quad (29)$$

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. Initial Conditions

In order to use this factored form of the covariance, we need to ensure that this factorization applies across all possible initial conditions. To verify that the factorization and update are independent of the initial conditions, we show that our assumed initial condition $\Sigma_0 = X_0Y_0^{-1} = \Sigma_0I^{-1}$ is an achievable result of performing a boundary update from each of two possible boundary conditions, which we will denote with a minus subscript as $\Sigma_- = \frac{X_-}{Y_-}$, or equivalently $\Omega_- = \frac{Y_-}{X_-}$, with a slight abuse of notation. The first boundary condition we will consider is that of infinite uncertainty, or equivalently zero information. The second is that of zero uncertainty, or equivalently infinite information. The linear system corresponding to the boundary update is given as

$$\begin{bmatrix}X\\Y\end{bmatrix}_0 = \begin{bmatrix}G & RG^{-T}\\MG & MRG^{-T}+G^{-T}\end{bmatrix}_0\begin{bmatrix}X\\Y\end{bmatrix}_- \quad (30)$$

We consider each boundary condition in turn, by solving the system in equation (30) and imposing the constraint $\Sigma_0 = X_0Y_0^{-1}$.

5.2.1. Boundary Case: Infinite Uncertainty, Zero Information: The boundary condition of infinite uncertainty $\Sigma_- = \infty$, or zero information $\Omega_- = 0$, corresponds to the case where $Y_- = 0$, which is shown as follows:

$$\Sigma_- = \frac{X_-}{Y_-} = \frac{X_-}{0} = \infty, \quad (31)$$

$$\Omega_- = \frac{Y_-}{X_-} = \frac{0}{X_-} = 0, \quad (32)$$

$$X_- \neq 0. \quad (33)$$

Using equation (30), the covariance factors are written as

$$X_0 = G_0X_- + 0 = G_0X_- \quad (34)$$

$$Y_0 = M_0G_0X_- + 0 = M_0G_0X_-. \quad (35)$$

Solving for the initial condition using equations (34-35), we obtain,

$$X_0Y_0^{-1} = G_0X_- \cdot X_-^{-1}G_0^{-1}M_0^{-1} = M_0^{-1},$$

which implies the following constraint:

$$\Sigma_0 = M_0^{-1}. \quad (36)$$

By denoting $\mathcal{A} = G_0X_-$ and applying the constraint in equation (36) to equations (34-35), we obtain the following solution set:

$$\begin{bmatrix}X\\Y\end{bmatrix}_0 = \begin{bmatrix}\mathcal{A}\\\Sigma_0^{-1}\mathcal{A}\end{bmatrix}, \quad \mathcal{A} \neq 0. \quad (37)$$

Note that our trivial initial condition of $X_0 = \Sigma_0$ and $Y_0 = I$ is valid with $\mathcal{A} = \Sigma_0$.

The result shown above is intuitive when considering EKF/EIF control and measurement updates from the boundary condition of zero information $\Omega_- = 0$ and, equivalently, infinite uncertainty $\Sigma_- = \infty$. The EKF control update is irrelevant, since adding any process noise to infinite covariance results in infinity:

$$\overline{\Sigma}_0 = G_0\Sigma_-G_0^T + R_0 = G_0\infty G_0^T + R_0 = \infty. \quad (38)$$

The measurement update, however, shows that an update from this boundary condition corresponds to receiving a measurement $M_0 = \Omega_0$:

$$\Omega_0 = \Omega_- + M_0 = 0 + M_0 = M_0. \quad (39)$$

Thus, this boundary update is equivalent to beginning in a state with zero information and increasing certainty by incorporating a measurement of value $M_0 = \Omega_0 = \Sigma_0^{-1}$.

5.2.2. Boundary Case: Zero Uncertainty, Infinite Information: The boundary condition of zero uncertainty $\Sigma_- = 0$, or

infinite information $\Omega_- = \infty$, corresponds to the case where $X_- = 0$, which is shown as follows:

$$\Sigma_- = \frac{X_-}{Y_-} = \frac{0}{Y_-} = 0, \tag{40}$$

$$\Omega_- = \frac{Y_-}{X_-} = \frac{Y_-}{0} = \infty, \tag{41}$$

$$Y_- \neq 0. \tag{42}$$

As before, the covariance factors are written as

$$X_0 = G_0 \cdot 0 + R_0 G_0^{-T} Y_- = R_0 G_0^{-T} Y_- \tag{43}$$

$$Y_0 = M_0 G_0 \cdot 0 + (M_0 R_0 G_0^{-T} + G_0^{-T}) Y_- \tag{44}$$

$$= (M_0 R_0 + I) G_0^{-T} Y_-. \tag{45}$$

Computing the initial covariance corresponding to equations (43-45) gives us

$$\begin{aligned} X_0 Y_0^{-1} &= R_0 G_0^{-T} Y_- \cdot Y_-^{-1} G_0^T (M_0 R_0 + I)^{-1} \\ &= R_0 (M_0 R_0 + I)^{-1}, \end{aligned}$$

implying the following constraint on R_0 and M_0:

$$\Sigma_0 = R_0 (M_0 R_0 + I)^{-1}. \tag{46}$$

We make the substitution $\mathcal{B} = G_0^{-T} Y_-$ for the free variables, and apply the constraint in equation (46) to equations (43-45), yielding the solution set:

$$\begin{bmatrix} X \\ Y \end{bmatrix}_0 = \begin{bmatrix} R_0 \mathcal{B} \\ (M_0 R_0 + I)\mathcal{B} \end{bmatrix}, \quad \mathcal{B} \neq 0, \tag{47}$$

This result is also intuitive, although not as straightforward as the previous boundary update in which the control update had no effect. Due to the ordering of control and measurement updates, the initial covariance can result from a combination of both adding uncertainty R_0 to the boundary state of perfect information, and then subsequently adding measurement information M_0. It is for this reason that the constraint set is a function of both R_0 and M_0. However, our assumed initial condition of $X_0 = \Sigma_0$ and $Y_0 = I$ is the trivial result of adding only process noise $R_0 = \Sigma_0$ and zero measurement information $M_0 = 0$, with $\mathcal{B} = I$.

6. Redheffer Star Product

The factored covariance representation and matrix form of the update given in equation (29) represents a non-recursive solution to the Riccati equation (13) in the symplectic form. The $2n \times 2n$ matrix Ψ is by definition symplectic if

$$\Psi J \Psi^T = J \tag{48}$$

where

$$J = \begin{bmatrix} 0 & -I_n \\ I_n & 0 \end{bmatrix} \tag{49}$$

and I_n is the $n \times n$ identity matrix. The eigenvalues of symplectic matrices such as Ψ occur in reciprocal pairs such that if λ is an eigenvalue of Ψ, then so is λ^{-1}. Unfortunately, as a result, composition of symplectic matrices is known to become numerically unstable as round-off errors in computation can result in loss of the symplectic eigenvalue structure (Fassbender, 2000). An alternate form that provides greater numerical stability through inherent structure preservation is the Hamiltonian form, with the corresponding composition operator known as the Redheffer "star" product (denoted with a '$\star$') (Redheffer, 1962). A $2n \times 2n$ block matrix

$$\mathcal{S} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{50}$$

is called *Hamiltonian* if

$$J\mathcal{S} = (J\mathcal{S})^T = -\mathcal{S}^T J, \tag{51}$$

noting that $J^T = J^{-1} = -J$. We will show that there is a Hamiltonian representation and composition operator equivalent to the symplectic form that does not share the same numerical instability.

6.1. Derivation of Star Product

We can formally derive the Hamiltonian method of composition by starting with descriptor matrices of a Hamiltonian system at two adjacent timesteps as follows:

$$\begin{bmatrix} x_2 \\ y_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x_1 \\ y_2 \end{bmatrix}, \tag{52}$$

$$\begin{bmatrix} x_3 \\ y_2 \end{bmatrix} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} \begin{bmatrix} x_2 \\ y_3 \end{bmatrix}. \tag{53}$$

For clarity it should be stated that these two system matrices have the same block structure, where each block actually corresponds to a time-varying quantity. Our goal is to determine the Hamiltonian matrix corresponding to the aggregate system that represents both equation (52) and equation (53). We will show that the resulting system can be computed using the star product in the following manner:

$$\begin{bmatrix} x_3 \\ y_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \star \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} \begin{bmatrix} x_1 \\ y_3 \end{bmatrix}. \tag{54}$$

Our explicit goal is to derive equations for the variables x_3 and y_1 in terms of x_1 and y_3 to determine the star product operation in equation (54). We begin by writing the given equations from the systems in equations (52) and (53), as follows:

$$x_2 = Ax_1 + By_2 \tag{55}$$

$$y_1 = Cx_1 + Dy_2 \tag{56}$$

$$x_3 = Wx_2 + Xy_3 \tag{57}$$

$$y_2 = Yx_2 + Zy_3. \tag{58}$$

Substituting y_2 from equation (58) into equation (55) we solve for x_2 as follows:

$$\begin{aligned} x_2 &= Ax_1 + B(Yx_2 + Zy_3) \\ (I - BY)x_2 &= Ax_1 + BZy_3 \\ x_2 &= (I - BY)^{-1} Ax_1 + (I - BY)^{-1} BZy_3 \end{aligned} \tag{59}$$

We also substitute x_2 from equation (55) into equation (58)

to solve for y_2 as follows:

$$\begin{aligned} y_2 &= Y(Ax_1 + By_2) + Zy_3 \\ (I - YB)y_2 &= YAx_1 + Zy_3 \\ y_2 &= (I - YB)^{-1}YAx_1 + (I - YB)^{-1}Zy_3. \end{aligned} \tag{60}$$

Now with x_2 and y_2 both written in terms of x_1 and y_3, it is possible to similarly solve for x_3 and y_1. To solve for x_3, we substitute x_2 from equation (59) into equation (57):

$$\begin{aligned} x_3 &= Wx_2 + Xy_3 \\ &= \Big((I - BY)^{-1}Ax_1 + (I - BY)^{-1}BZy_3\Big) + Xy_3. \end{aligned}$$

which is simplified to give the desired result

$$x_3 = W(I-BY)^{-1}Ax_1 + (X + W(I-BY)^{-1}BZ)y_3. \tag{61}$$

We also substitute y_2 from equation (60) into equation (56) to solve for y_1 as follows:

$$\begin{aligned} y_1 &= Cx_1 + Dy_2 \\ &= Cx_1 + D(I - YB)^{-1}YAx_1 + D(I - YB)^{-1}Zy_3, \end{aligned}$$

which simplifies to become

$$y_1 = (C + D(I-YB)^{-1}YA)x_1 + D(I-YB)^{-1}Zy_3. \tag{62}$$

Now, with equations (61) and (62), our solution is obtained as the aggregate system in equation (54), which can now be written in terms of one matrix as

$$\begin{bmatrix} x_3 \\ y_1 \end{bmatrix} = \begin{bmatrix} W(I-BY)^{-1}A & X + W(I-BY)^{-1}BZ \\ C + D(I-YB)^{-1}YA & D(I-YB)^{-1}Z \end{bmatrix} \begin{bmatrix} x_1 \\ y_3 \end{bmatrix}, \tag{63}$$

where we have now derived the star product as a set of matrix block operators, given as

$$S_1 \star S_0 = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \star \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} = \tag{64}$$

$$\begin{bmatrix} W(I - BY)^{-1}A & X + W(I - BY)^{-1}BZ \\ C + D(I - YB)^{-1}YA & D(I - YB)^{-1}Z \end{bmatrix}. \tag{65}$$

6.2. Scattering Theory Parallel

The Hamiltonian method of composition can be demonstrated most intuitively with an analogy in scattering theory stemming from Redheffer's original work and developed in the context of optimal filtering by Kailath et al. (1976).

The key to this analogy lies in the forward-backward Hamiltonian system associated with the discrete Riccati difference equation: in filtering, this corresponds to a system which simultaneously produces filtered and smoothed estimates; in scattering theory it is interpreted as waves traveling through a medium in opposite directions with forward and backward transmission and reflection operators, whose interactions are determined by the state space parameters $\{G, R, M\}$. Given these parameters for a set of consecutive scattering medium layers, or equivalently a set of consecutive Kalman filter updates, the descriptor matrices for each update can be combined using the star product to produce *one* descriptor matrix. This resulting descriptor matrix represents the aggregate scattering medium, or equivalently the aggregate filter update.

The Riccati difference equation is represented as follows:

$$\Sigma_t \;=\; R_t + G_t\Sigma_{t-1}(I - M_t\Sigma_{t-1})^{-1}G_t^T, \tag{66}$$

where at filtering time t (*or scattering layer* t)

$G_t =$ state transition matrix (*or scattering transmission matrix*)
$R_t =$ error covariance (*or right reflection coefficient*)
$M_t =$ measurement information (*or left reflection coefficient*)

and the associated Hamiltonian matrix is called the *scattering matrix* and has the form:

$$\mathcal{S}_t = \begin{bmatrix} G & R \\ -M & G^T \end{bmatrix}_t \tag{67}$$

Composition of multiple layers can be performed using the star product as

$$\mathcal{S}_{t:T} = \mathcal{S}_t \star \mathcal{S}_{t+1} \star \cdots \star \mathcal{S}_T, \tag{68}$$

where $\mathcal{S}_{t:T}$ is the aggregate scattering matrix capturing the effects of layers t through T.

It can be shown that individual control and measurement steps have corresponding scattering matrices by noting the direct correspondence between EKF/EIF updates and the Riccati equation (66). The control update yields the control update scattering matrix

$$\mathcal{S}_t^C = \begin{bmatrix} G & R \\ 0 & G^T \end{bmatrix}_t. \tag{69}$$

Similarly, the measurement update gives the measurement update scattering matrix

$$\mathcal{S}_t^M = \begin{bmatrix} I & 0 \\ -M & I \end{bmatrix}_t. \tag{70}$$

Thus, multiple filter updates can be composed in the Hamiltonian form as in equation (68) by star-producing the corresponding scattering matrices in succession:

$$\mathcal{S}_t = \mathcal{S}_t^C \star \mathcal{S}_t^M. \tag{71}$$

6.2.1. Initial Conditions: The initial conditions in this formulation are handled in similar fashion to our discussion in Section 5.2. The insight in applying the initial covariance is to create a *boundary layer*, which is a scattering layer that is attached to the *null* scattering layer.

It is straightforward to see that the initial covariance Σ_0 could be represented as a null layer with process noise $R = \Sigma_0$, or alternatively with additional measurement information $M = \Sigma_0^{-1} = \Omega_0$. These are identical to the cases derived in Section 5.2. The first two cases would be boundary layers described as

$$\mathcal{S}_0 = \begin{bmatrix} I & \Sigma_0 \\ 0 & I \end{bmatrix}, \quad \text{or} \quad \mathcal{S}_0 = \begin{bmatrix} I & 0 \\ -\Omega_0 & I \end{bmatrix}, \tag{72}$$

where this boundary layer can be attached to a scattering medium to impose an initial condition.

6.3. Computing the Hamiltonian One-Step Update

The key to applying the star product for composition lies in the associative property of the star product operation

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 $\mathcal{G} = \{\{n_i\}, \{e_{ij}\}, \{\zeta_{ij}\}\}$

(Redheffer (1962)), which ensures that the star product of any scattering matrices yields another scattering matrix of the same block form. To compose the filter updates for T time steps, we begin by computing the aggregate scattering matrix,

$$\mathcal{S}_{1:T} = \begin{bmatrix} G_{1:T} & R_{1:T} \\ -M_{1:T} & G_{1:T}^T \end{bmatrix} = \mathcal{S}_1 \star \mathcal{S}_2 \star \cdots \star \mathcal{S}_T. \tag{73}$$

We can then apply a novel initial condition Σ_0 and use equation (66) to solve for the posterior covariance,

$$\begin{bmatrix} \cdot & \Sigma_T \\ \cdot & \cdot \end{bmatrix} = \begin{bmatrix} I & \Sigma_0 \\ 0 & I \end{bmatrix} \star \mathcal{S}_{1:T}. \tag{74}$$

(The matrix elements $\cdot$ are irrelevant to the final solution for the covariance.) As mentioned previously, the advantage to this formulation is that the aggregation of Hamiltonian matrices is numerically more stable than the symplectic form of equation (29).

7. The Belief Roadmap Algorithm

The belief space planning algorithm can now be shown as 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 at steps along each edge and assembled via matrix multiplication into a one-step transfer function for the covariance, ζ_{ij}, according to equation (29).

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). During search, 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 in one efficient step along edge e_{ij}. 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 points of discussion that we address in turn. Firstly, note that this must be a forward search process; the terminal node of this path cannot be determined *a priori*, for while the goal location μ_{goal} is known, 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. Note that cycles in the path are disallowed in line 9 of

Algorithm 2 The Belief Roadmap Search Process.

Require: Start belief (μ_0, Σ_0), goal location μ_{goal} and belief graph $\mathcal{G}$

Ensure: Path p from μ_0 to μ_{goal} with minimum goal covariance Σ_{goal}.

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 $\Sigma{=}\emptyset$, 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** $n_{goal}[p]$

Algorithm 2, where a neighboring node n' is only considered if it is not already in the search state path $n[p]$. This guarantees that upon termination of the breadth-first search process, the minimum covariance path stored at the goal node is optimal with respect to the roadmap. More intelligent search processes rely on an $A^\star$ heuristic to find the goal state faster; however, common admissible heuristics do not apply as the evolution of the covariance through the roadmap is non-monotonic (due to the expansion and contraction of uncertainty in the underlying state estimation process). Developing suitable $A^\star$ heuristics for planning in belief space are a topic for future work. Similarly, the applicability of dynamic search processes, such as the $D^\star$ family of algorithms (Stentz, 1995; Koenig and Likhachev, 2002) and anytime methods (Van den Berg et al., 2006), is a direction for future research. 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, note in lines 12-13 of 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 in line 14.

Further, considerable work has been devoted to finding good sampling strategies in fully-observable motion planing problems (we refer the reader to (Hsu et al., 2006; Missiuro

Algorithm 3 The Min-Max Belief Roadmap (minmax-BRM) algorithm.

Require: Start belief (μ_0, Σ_0), goal location μ_{goal} and belief graph $\mathcal{G}$

Ensure: Path p from μ_0 to μ_{goal} with minimum maximum covariance.

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

and Roy, 2006)). Such strategies can bias samples towards different topological features and areas of interest to improve both the quality and efficiency of the roadmap. For the results in this paper, we used a 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 strategy 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.

7.1. Modified BRM for MinMax Path Uncertainty

In the BRM formulation shown in Algorithm 2, the search process finds the series of trajectories that results in minimal uncertainty at the goal location; however, it may be desirable to instead limit the maximum uncertainty encountered along an entire path. One approach could be to impose bounds on the maximum allowable uncertainty during the BRM search $tr(\Sigma) < tr_{max}$ to discard undesirable goal paths. In a more principled approach, one could modify the BRM search process to optimize an alternative objective function that minimizes the maximum predicted uncertainty along the entire path. Within the context of the BRM graph, this approach would consider the posterior covariance predicted at each intermediate belief node in a series of trajectories. The goal would be to minimize the objective function $\hat{J}$, which is given as

$$\hat{J}(b_t) \approx \min_{u_{0:T}}(\sum_{t}^{T} c(b_t[\mu], u_t) + \max_{b_{0:T}} D(b_t[\Sigma])), \qquad (75)$$

where $\hat{J}(\ldots)$ is the cost of a path, c is the cost of executing control u_t from belief pose $b_t[\mu]$, and D is the cost associated with the maximum uncertainty of all discrete belief nodes along the path.

The BRM search process is adapted to minimize the objective function $\hat{J}$ in Algorithm 3, which we have named the Min-Max Belief Roadmap (minmax-BRM) algorithm. There are two key changes from the standard BRM. First, the augmented search node structure in line 3 stores the best path p to the given node and the maximum covariance Σ^p_{max} along p. The best path $n_i[p]$ to node n_i corresponds to the series of nodes beginning at the start node n_0 that collectively has the minimum maximum (min-max) covariance of all such paths considered to node n_i. The maximum covariance $n_i[\Sigma^p_{max}]$ along this best path $n_i[p]$ is also stored in the search state for computing the associated cost D in the objective function $\hat{J}$ (equation (75)) and for decision-making during search. Note that the covariance $n_i[\Sigma]$ stored at node n_i is no longer the minimum achievable covariance, but rather the posterior covariance resulting from the best path $n_i[p]$.

Secondly, the primary decision-making step in line 13 is modified for the new objective function. In this case, the path being considered in the current search state $\{n[p], n'\}$ is deemed better than the existing path $n'[p]$ to node n' if its maximum uncertainty is less than that of the existing path. Note that the maximum uncertainty of the current search path $\{n[p], n'\}$ is computed by taking the max function of the associated uncertainty of each portion of this path, which is $tr(n[\Sigma^p_{max}])$ for $n[p]$ and $tr(\Sigma')$ for n'. If the current search path is better than the existing path, then the node n' is updated accordingly in line 14 and placed on the queue in line 15.

A key consideration of the minmax-BRM algorithm is that it can only guarantee an optimal solution within the roadmap for a specific resolution of the uncertainty evolution along a path. In Algorithm 3, we only consider the covariance at node locations along the path. While lines 11-12 exactly compute the posterior covariance of multiple EKF updates along a trajectory, the underlying multi-step process is not monotonic. This means that it is possible for the covariance at an intermediate point on an edge between two graph nodes to be larger than both the prior covariance and posterior covariance for the full trajectory. It is possible to revert to some form of multi-step approach to this problem, but, without further assumptions, the guarantee of min-max covariance will always be limited to the resolution of discretization. We leave

the analysis of this problem for future work, and place our focus on the standard BRM for experiments.

It is important to note the generality of the BRM formulation, which was demonstrated in this section by modifying the search process to optimize an alternative objective function. The BRM technique presents a general approach to planning in belief space that can be adapted to solve a broad class of planning problems.

8. UWB Localization

In this work, we applied the belief roadmap algorithm to the problem of navigating in a GPS-denied environment using ultra-wide bandwidth (UWB) radio beacons for localization. UWB is a nascent technology that is amenable to ranging applications in dense indoor and urban environments, overcoming weaknesses of traditional narrowband counterparts with its fine delay resolution and large bandwidth that provide immunity to multipath fading and the ability to penetrate building materials (Win and Scholtz, 1998; Cassioli et al., 2002). Ranges can be computed between a pair of UWB radio beacons by measuring the round-trip time-of-flight of UWB pulse exchanges. While a complete characterization of the UWB channel is still an active area of research and beyond the scope of this paper, here we briefly develop the general ranging model used in this work. A supplementary discussion of the details of this UWB ranging scheme was presented by Prentice (2007).

8.1. Ultra-Wideband Measurement Model

The general UWB sensor model can be written as

$$r_t = d_t + b_t + n_t, \tag{76}$$

where r_t is the range, d_t is the distance between UWB sensors, b_t is the range bias, and n_t is additive noise. The round-trip time calculation is approximate in nature, leading to uncertainty in the range calculation which can be modelled as a stochastic process. In a testing campaign, we developed a Gaussian model to describe ranging uncertainty in LOS scenarios. We characterized the Gaussian process by gathering range data between a pair of sensors at various distances with LOS visibility. The distance was increased in 0.25 meter increments from 1 to 14 meters of separation and at each point, 1,000 range samples were gathered. The resulting data is plotted in Figure 3, showing the mean bias μ_{bias} and standard deviation σ_{bias} errorbar at each distance.

This data suggests that the LOS range bias can be reasonably modeled as distance-varying Gaussian noise, with mean $\mu_{bias}(d_t)$ and standard deviation $\sigma_{bias}(d_t)$. Computing a linear regression yields

$$\mu_{bias}(d_t) = \mu^m_{bias} d_t + \mu^b_{bias} \tag{77}$$

$$\sigma_{bias}(d_t) = \sigma^m_{bias} d_t + \sigma^b_{bias}. \tag{78}$$

The range function in equation (76) then becomes

$$r_t = d_t + \mu_{bias}(d_t) + \mathcal{N}(0, \sigma_{bias}(d_t)^2), \tag{79}$$

where the bias b_t is now a linear function of the distance $\mu_{bias}(d_t)$, and the noise term n_t is zero-mean Gaussian noise

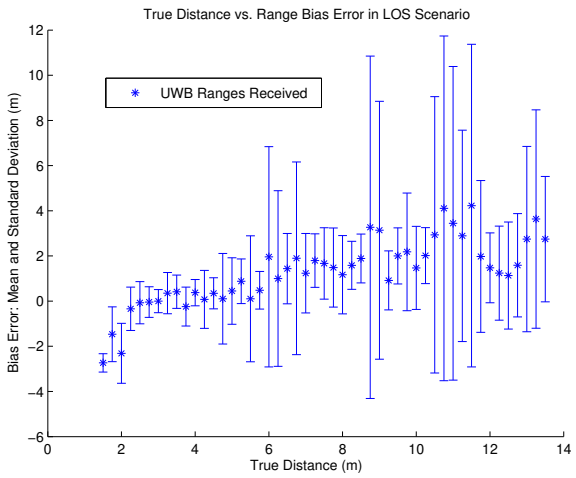


Fig. 3. The Ultra-Wide Band Ranging Model in LOS. The x-axis is the true range between sensors and the y-axis is the measured systematic error (unbiased measurements are at 0), and the error bars give the standard deviation of the random noise.

with variance $\sigma_{bias}(d_t)^2$. When used in filtering problems, the range function in equation (79) corresponds to the observation function $z_t = h(x_t) + v_t$, with $z_t = r_t$, $v_t = \mathcal{N}(0, \sigma_{bias}(d_t)^2)$ and $h(x_t)$ is given as

$$h(x_t) = d_t + \mu_{bias}(d_t) \tag{80}$$

$$= \mu^b_{bias} + (1 + \mu^m_{bias}) \sqrt{(x - x_{beacon})^2 + (y - y_{beacon})^2}, \tag{81}$$

where x_t is assumed to be the robot pose $(x, y, \theta)_t$ at time t, and (x_{beacon}, y_{beacon}) is the ranging beacon location.

9. 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 the stochastic range sensor model from Section 8.1. 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.

We begin by presenting the motion and sensor models used in our experiments, which are linearized versions of the motion and sensor models for use in our EKF-based formulation. Note that, for readability, we omit time index subscripts in the next two sections; however, all matrices derived are time-varying quantities.

9.1. Linearized Motion Model

We use the following non-linear probabilistic motion model,

$$\begin{aligned} g_x &= x + D\cos(\theta + \frac{T}{2}) + C\cos(\theta + \frac{T+\pi}{2}) \\ g_y &= y + D\sin(\theta + \frac{T}{2}) + C\sin(\theta + \frac{T+\pi}{2}) \\ g_\theta &= \theta + T \mod 2\pi, \end{aligned}$$

where g_x, g_y and g_θ are the components of g corresponding to each state variable, and the control variable u_t is given by $u_t = \begin{bmatrix} D & C & T \end{bmatrix}^T$ with down-range D, cross-range C and turn T components.

In the EKF, the state transition matrix G is the Jacobian of the motion model with respect to the state, and is computed by linearizing the state transition function g about the mean state μ as follows:

$$G = \begin{bmatrix} \frac{\delta g_x}{\delta x} & \frac{\delta g_x}{\delta y} & \frac{\delta g_x}{\delta \theta} \\ \frac{\delta g_y}{\delta x} & \frac{\delta g_y}{\delta y} & \frac{\delta g_y}{\delta \theta} \\ \frac{\delta g_\theta}{\delta x} & \frac{\delta g_\theta}{\delta y} & \frac{\delta g_\theta}{\delta \theta} \end{bmatrix}_\mu = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix},$$

where

$$\begin{aligned} a &= -D\sin(\mu_\theta + \frac{T}{2}) - C\sin(\mu_\theta + \frac{T+\pi}{2}) \\ b &= D\cos(\mu_\theta + \frac{T}{2}) + C\cos(\mu_\theta + \frac{T+\pi}{2}). \end{aligned}$$

The linearized process noise in state space is computed as $R \triangleq VWV^T$ where W is the process noise covariance in control space

$$W = \begin{bmatrix} \sigma_D^2 & 0 & 0 \\ 0 & \sigma_C^2 & 0 \\ 0 & 0 & \sigma_T^2 \end{bmatrix}$$

and V is the mapping from control to state space, computed as the Jacobian of the motion model with respect to the control space components

$$V = \begin{bmatrix} \frac{\delta g_x}{\delta D} & \frac{\delta g_x}{\delta C} & \frac{\delta g_x}{\delta T} \\ \frac{\delta g_y}{\delta D} & \frac{\delta g_y}{\delta C} & \frac{\delta g_y}{\delta T} \\ \frac{\delta g_\theta}{\delta D} & \frac{\delta g_\theta}{\delta C} & \frac{\delta g_\theta}{\delta T} \end{bmatrix}_{\mu, \mu^{[u]}}.$$

Computing these partial derivatives leads to

$$V = \begin{bmatrix} \cos(\theta + \frac{T}{2}) & \cos(\theta + \frac{T+\pi}{2}) & -\frac{1}{2}\left(D\sin(\theta + \frac{T}{2}) + C\sin(\theta + \frac{T+\pi}{2})\right) \\ \sin(\theta + \frac{T}{2}) & \sin(\theta + \frac{T+\pi}{2}) & \frac{1}{2}\left(D\cos(\theta + \frac{T}{2}) + C\cos(\theta + \frac{T+\pi}{2})\right) \\ 0 & 0 & 1 \end{bmatrix},$$

where the components of V are evaluated at the mean state μ and the mean control $\mu^{[u]}$ (D, C, and T take on the mean values of the respective normal distributions).

9.2. Linearized LOS Sensor Model

Similarly, the UWB sensor model developed in Section 8.1 can be linearized for use in the EKF. For convenience we restate the distance-varying Gaussian noise model of the bias $\mathcal{N}(\mu_b(d), \sigma_b(d))$ and the observation function z. The noise model is given by,

$$\begin{aligned} \mu_b(d) &= \mu_b^m d + \mu_b^b \\ \sigma_b(d) &= \sigma_b^m d + \sigma_b^b \end{aligned}$$

and the observation function $z = h(x) + v$ is determined by,

$$\begin{aligned} h(x) &= \mu_b^b + (1 + \mu_b^m)\sqrt{(x - x_b)^2 + (y - y_b)^2} \\ v &= \mathcal{N}(0, \sigma_b(d)^2), \end{aligned}$$

where (x, y) is the robot pose, (x_b, y_b) is the beacon location, d is the Euclidean distance and the parameters of the Gaussian bias distribution are linear functions of the distance.

The linearized transformation from measurement space to state space is computed as the measurement Jacobian H, computed as the partial derivatives of the measurement function with respect to each component of the state:

$$H = \begin{bmatrix} \frac{\delta h}{\delta x} & \frac{\delta h}{\delta y} & \frac{\delta h}{\delta \theta} \end{bmatrix},$$

which becomes:

$$H = \begin{bmatrix} \frac{(1+\mu_b^m)(x-x_b)}{\sqrt{(x-x_b)^2+(y-y_b)^2}} & \frac{(1+\mu_b^m)(y-y_b)}{\sqrt{(x-x_b)^2+(y-y_b)^2}} & 0 \end{bmatrix}. \quad (82)$$

Note that $(x - x_b) = d\cos\theta_m$ and $(y - y_b) = d\sin\theta_m$, where $\theta_m = \text{atan2}(y - y_b, x - x_b)$ is the angle of measurement relative to the robot pose. Thus, equation (82) becomes

$$H = \begin{bmatrix} (1 + \mu_b^m)\cos\theta_m & (1 + \mu_b^m)\sin\theta_m & 0 \end{bmatrix}. \quad (83)$$

As can be seen, the range measurements yield no information on bearing, and thus only affect the estimation of the x and y components of the robot state. The measurement noise covariance $Q \triangleq cov(q, q)$ for a given beacon is the 1×1 matrix

$$Q = \begin{bmatrix} (\sigma_b^m d + \sigma_b^b)^2 \end{bmatrix}. \quad (84)$$

Recall that the information matrix update in the measurement step is additive as $\Omega_t = \overline{\Omega}_t + M_t$. Thus, the range measurements $z_t^{[i]}$ to each of N visible beacons i at time t can be incorporated by computing the aggregate measurement information as

$$M_t = \sum_{i=0}^{N} M_t^{[i]}, \quad (85)$$

where $M_t^{[i]} = H_t^{[i]^T} Q_t^{[i]^{-1}} H_t^{[i]}$.

9.3. 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 a conventional PRM, which does not incorporate uncertainty. In each test iteration, sensor locations were sampled along randomized trajectories between a start and goal location in an obstacle-free environment with

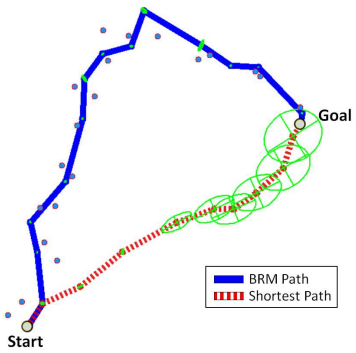


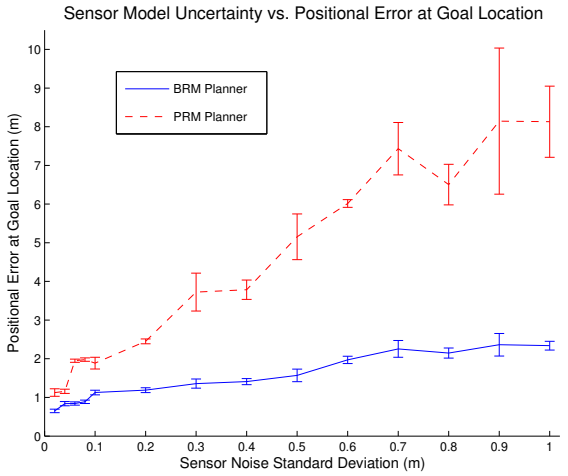
Fig. 4. Experimental Setup. Range sensor locations (shown as small circles) were sampled along randomized trajectories between a start and goal location. The solid and dashed line show the plans generated by the BRM and shortest path algorithms, respectively, with ellipses indicating the covariances 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.

$100m$ sides, as shown in Figure 4. This experimental setup tests the ability of the BRM to detour from shorter paths for those with lower expected uncertainty. We tested the quality of paths computed by the BRM and PRM shortest path planning algorithms by evaluating the average position error obtained at the goal location after executing the prescribed path.

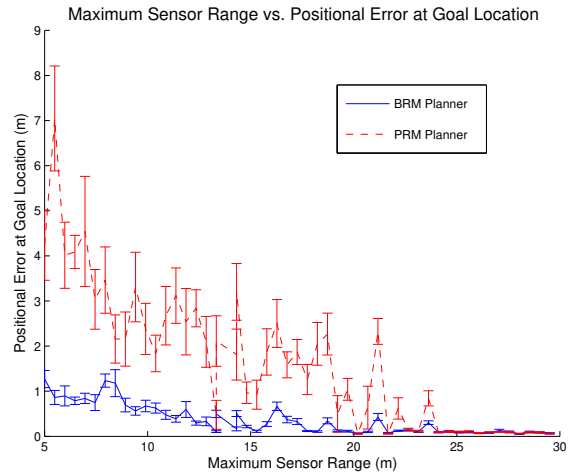
We performed two analyses to demonstrate that the BRM provided more accurate localization; we artificially varied the random noise of the range beacons, and we artificially limited the range of the beacons by discarding measurements beyond a maximum range. In Figure 5(a), we see the performance of the 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. The BRM planner contends with increased sensor noise by finding trajectories with higher quality measurements. In Figure 5(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 the maximum sensor range increases, trajectories farther from sensors provide sufficient information for localization and the positional errors in both planners converge to similar values. Intuitively, as the information space becomes artificially saturated with abundant state information, the agent can remain well-localized regardless of its trajectory. Conversely, when the information space has greater disparity, the BRM excels at finding higher-quality paths that provide greater state information.

9.4. 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 (ζ_{ij}) to that of the standard EKF covariance updates. Note once again that the one-step covariance transfer function produces the same resulting covariance as performing each of multiple standard EKF updates in succession; there is no trade-off in



(a) Varying Sensor Noise

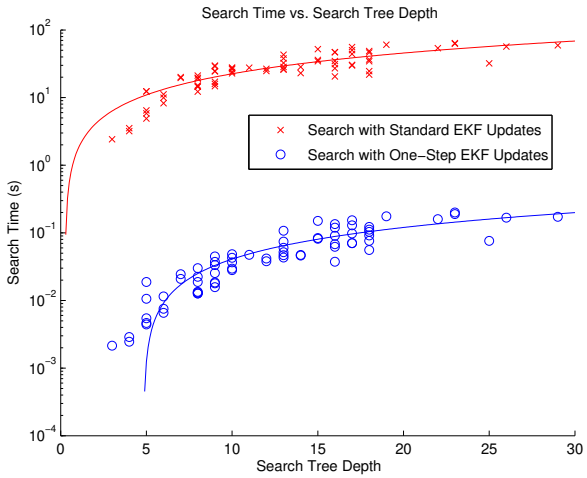


(b) Varying Sensor Range

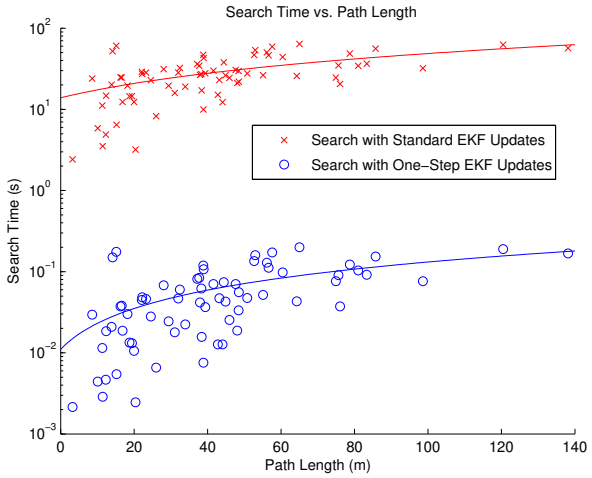
Fig. 5. In these figures we characterized the positional accuracy of the robot for both the PRM shortest path and BRM planners as a function of the sensor noise and sensor range. (a) Accuracy vs. Sensor Noise. The positional accuracy of the PRM shortest path algorithm suffered with increased noise. The positional accuracy of the BRM increased slightly but not substantially with noise. (b) Accuracy vs. Range. The positional accuracy of the PRM 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.

accuracy. This experiment evaluates the effect of using the one-step transfer function on planning speed. 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 6(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 improvement by over two orders of magnitude, with search costs scaling logarithmically with increasing tree depth. Similar results



(a) Time vs. Tree Depth



(b) Time vs. Path Length

Fig. 6. 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.

are obtained when comparing the search times with respect to the length of the resulting path, shown in Figure 6(b), reiterating the significant scalable improvement of the one-step update. The one-step covariance update presents a consistent improvement in planning speed 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.

9.5. Example trajectories:

Finally, example trajectories are shown in Figures 7-8. In Figure 7, a mobile robot navigates through a small-sized indoor environment (∼70m in length) providing ranging beacons

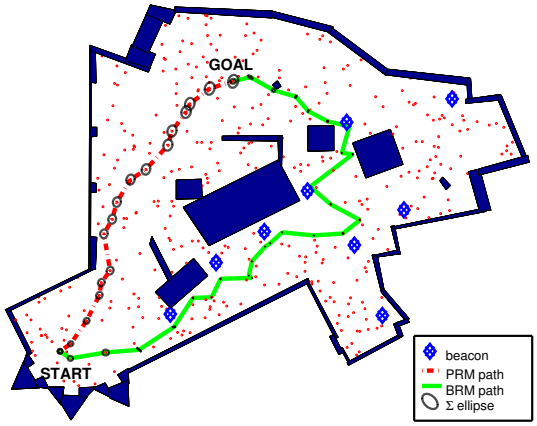


Fig. 7. Example trajectories for a mobile robot in an indoor environment (∼70m across) with ranging beacons. The robot navigates from START (lower left) to GOAL (top). The BRM finds a path in close proximity to the ranging beacons, balancing the shorter route computed by the PRM in configuration space against a lower cost path in information space. The positional uncertainty over the two paths is shown as the bold covariance ellipses.

for localization. The BRM planner detours from the direct route chosen by the shortest path planner for sensor-rich regions of the environment. Whereas the shortest path planner accumulates positional error by relying on dead-reckoning, the BRM path incorporates ranging information to maintain lower uncertainty.

Figure 8 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 8(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.

10. Conclusion

In this paper, we have addressed the problem of planning in belief space for linear-Gaussian systems, 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 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 (Roy and Thrun, 1999) but in limited

(a) PRM: Shortest Path

(b) BRM: Lowest Expected Uncertainty Path

Fig. 8. Example paths for a mobile robot navigating across MIT campus. The solid line in each case is the robot path, the small dots are the range beacons being used for localization, and the dark 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.

scales of environments. We believe that our demonstration of belief-space planning in the MIT campus environment is considerably larger than existing results.

References

Alterovitz, R., Branicky, M., and Goldberg, K. (2006). Constant-curvature motion planning under uncertainty with applications in image-guided medical needle steering. In *Workshop on the Algorithmic Foundations of Robotics*. Springer.

Alterovitz, R., Simeon, T., and Goldberg, K. (2007). The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In *Robotics: Science and Systems*.

Bohlin, R. and Kavraki, L. (2000). Path planning using lazy PRM. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Cassioli, D., Win, M. Z., and Molisch, A. F. (2002). The ultra-wide bandwidth indoor channel: from statistical model to simulations. *IEEE Journal on Selected Areas in Communications* 20(6): 1247–1257.

Fassbender, H. (2000). *Symplectic Methods for the Symplectic Eigenvalue Problem*. Kluwer Academic/Plenum Publishers.

Hsu, D., Latombe, J., and Kurniawati, H. (2006). On the Probabilistic Foundations of Probabilistic Roadmap Planning. *The International Journal of Robotics Research* 25(7).

Julier, S., Uhlmann, J., and Durrant-Whyte, H. (1995). A new approach for filtering nonlinear systems. In *Proc. ACC*.

Kaelbling, L., Littman, M., and Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101: 99–134.

Kailath, T., Friedlander, B., and Ljung, L. (1976). Scattering theory and linear least squares estimation—II discrete-time problems. *J. Franklin Institute* 301: 77–82.

Kalman, E., Rudolph (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering* 82(Series D): 35–45.

Kavraki, L., Svestka, P., Latombe, J.-C., and Overmars, M. (1996). Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4): 566–580.

Koenig, S. and Likhachev, M. (2002). Improved fast replanning for robot navigation in unknown terrain. In *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA'02*, volume 1.

LaValle, S. and Kuffner, J. (2001). Randomized kinodynamic planning. *International Journal of Robotics Research* 20(5): 378–400.

Lozano-Perez, T. (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers* C-32(2): 108–120.

Missiuro, P. and Roy, N. (2006). Adapting probabilistic roadmaps to handle uncertain maps. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Pepy, R., Kieffer, M., and Walter, E. (2008). Reliable robust path planner. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, 1655–1660.

Prentice, S. (2007). *Robust Range-Based Localization and Motion Planning Under Uncertainty Using Ultra-Wideband Radio*. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Redheffer, R. (1962). On the relation of transmission-line theory to scattering and transfer. *Journal of Mathematical Physics* 41: 1–41.

Roy, N. and Thrun, S. (1999). Coastal navigation with mobile robots. In *Advances in Neural Processing Systems 12*, volume 12.

Smith, R., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robotic Vehicles*. Springer-Verlag, Orlando, FL.

Stentz, A. (1995). The Focussed Dˆ* Algorithm for Real-Time Replanning. In *International Joint Conference on Artificial Intelligence*, volume 14, 1652–1659.

Van den Berg, J., Ferguson, D., and Kuffner, J. (2006). Anytime path planning and replanning in dynamic environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 2366–2371.

Vaughan, D. (1970). A nonrecursive algebraic solution for the discrete Riccati equation. *IEEE Transactions on Automatic Control* 15(5): 597–599.

Win, M. Z. and Scholtz, R. A. (1998). Impulse radio: how it works. *IEEE Communications Letters* 2(2): 36–38.