MIT-CSAIL-TR-2011-039 August 27, 2011

A hypothesis-based algorithm for planning and control in non-Gaussian belief spaces

Robert Platt Jr, Leslie Kaelbling, Tomas Lozano-Perez, and Russ Tedrake

A hypothesis-based algorithm for planning and control in non-Gaussian belief spaces

Robert Platt, Leslie Kaelbling, Tomas Lozano-Perez, and Russ Tedrake

Abstract We consider the partially observable control problem where it is potentially necessary to perform complex information-gathering operations in order to localize state. One approach to solving these problems is to create plans in *belief-space*, the space of probability distributions over the underlying state of the system. The belief-space plan encodes a strategy for performing a task while gaining information as necessary. Most approaches to belief-space planning rely upon representing belief state in a particular way (typically as a Gaussian). Unfortunately, this can lead to large errors between the assumed density representation and the true belief state. We propose a new computationally efficient algorithm for planning in non-Gaussian belief spaces. We propose a receding horizon re-planning approach where planning occurs in a low-dimensional sampled representation of belief state while the true belief state of the system is monitored using an arbitrary accurate high-dimensional representation. Our key contribution is a planning problem that, when solved optimally on each re-planning step, is guaranteed, under certain conditions, to enable the system to gain information. We prove that when these conditions are met, the algorithm converges with probability one. We characterize algorithm performance for different parameter settings in simulation and report results from a robot experiment that illustrates the application of the algorithm to robot grasping.

1 Introduction

A fundamental objective of robotics is to develop systems that can function robustly in unstructured environments where the state of the world is only partially observed and measurements are noisy. For example, robust robot manipulation is well modeled as partially observable problem. It is common to model control problems such as these as partially observable Markov decision processes (POMDPs). However,

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar St., Cambridge, MA, e-mail: {rplatt,lpk,tlp,russt}@csail.mit.edu

in general, finding optimal solutions to POMDPs has been shown to be PSPACE complete [12]. Even many approximate approaches are computationally complex: the time complexity of standard point-based algorithms, such as HSVI and SARSOP, is exponential in the planning horizon [17, 9, 15]. A growing body of work is focused on finding correct rather than optimal solutions to the partially observable control problem. Many of these approaches search for plans in *belief space*, the space of probability distributions over the underlying state space. The idea of planning in belief space can be traced back to some of the early dual control work where differential dynamic programming was used to find robust policies in stochastic domains [1]. More recent work has explored the application of different planning and re-planning mechanisms to the belief space planning problem [13, 6, 11]. Although these approaches are well suited to finding complex information-gathering behavior, they do so at the expense of solving a planning problem that is higher dimensional than the underlying perfectly-observable planning problem. Another recent class of approaches avoids this complexity by evaluating large numbers of candidate trajectories in the underlying state space in terms of the information that is likely to be gained during execution and the chances of colliding with problem constraints [18, 14, 5]. Although these approaches plan directly in the (lower-dimensional) state space, it may be necessary to create a large number of plans before finding one with satisfactory information-gathering properties.

One drawback with the belief space planning work cited above is the assumption that belief state (the probability distribution over underlying system state) is Gaussian. Unfortunately, this assumption is unwarranted in many robot navigation and manipulation applications (witness the popularity of the particle filter in these applications). Furthermore, directly extending an approach such as in [13] to non-Gaussian distributions quickly results in a computationally complex planning problem because of the high dimensionality of typical non-Gaussian parametrizations (for example, see [2]). This paper considers the problem of planning in non-Gaussian belief spaces. We propose an algorithm that, under certain conditions, is provably correct and also computationally efficient. Belief space planning implicitly necessitates tracking belief state using a Bayes filter. Our key idea is to separate the representation used to track belief state from the representation used for planning. During execution of the plan, system state is tracked using an arbitrary Bayes filter implementation that is selected by the system designer (a particle filter, for example). For the purposes of planning, however, this potentially high-dimensional belief state representation is projected onto a low-dimensional sampled representation. Plans are created that generate observations that differentiate a hypothesis sample from the other samples while also reaching a goal state. If the true belief state diverges too far from the nominal belief space trajectory during execution of the plan, then a re-planning cycle is triggered and the process iterates. The dimensionality of this planning problem is linear in the dimensionality of the underlying state space. This compares favorably with other algorithms [13, 6, 11, 1] which must solve planning problems quadratically larger than the fully observable problem. Perhaps surprisingly, this approach can be proved to solve the belief space planning problem (under certain conditions) with probability one when as few as two samples are used for planning. Moreover, our experiments indicate that, for relatively simple problems at least, it is unnecessary to use large numbers of samples in order to obtain good plans. After defining the problem in Section 2, this paper describes the algorithm in Section 3 and proves convergence in Section 4. In Section 5, we experimentally characterize the performance of algorithm as a function of the number of samples used. Finally, in Section 6, we apply the algorithm to a robot grasping problem where a robot must simultaneously localize and grasp objects in the environment.

2 Problem Statement

We are concerned with the class of control problems where it is desired to reach a specified goal state even though state may only be estimated based on partial or noisy observations. Consider a discrete-time system with continuous non-linear deterministic process dynamics [1], $x_{t+1} = f(x_t, u_t)$, where state, x, is a column vector in $\mathbb{R}^n$, and action, $u \in \mathbb{R}^l$. Although state is not directly observed, an observation, $z_t = h(x_t) + v_t$, is made at each time t, where $z \in \mathbb{R}^m$ is a column vector and v_t is zero-mean Gaussian noise with covariance Q.

Bayes filtering can be used to estimate state based on the previous actions taken and observations perceived. The estimate is a probability distribution over state represented by a probability density function (pdf), $\pi(x;b) : \mathbb{R}^n \rightarrow \mathbb{R}^+$ with parameter vector, $b \in \mathscr{B}$. The parameter vector is called the *belief state* and the parameter space, $\mathscr{B}$, is called the *belief-space*. For deterministic process dynamics, the Bayes filter update can be written:

$$\pi(f(x,u_t); b_{t+1}) = \frac{\pi(x; b_t) P(z_{t+1} | x, u_t)}{P(z_{t+1})}. \tag{1}$$

The Bayes update calculates a new belief state, b_{t+1}, given b_t, u_t, and z_{t+1}. It will sometimes be written, $b_{t+1} = G(b_t, u_t, z_{t+1})$. In general, it is impossible to implement Equation 1 exactly using a finite-dimensional parametrization of belief-space. However, a variety of approximations exist in practice [4].

Starting from an initial belief state, b_1, the control objective is to achieve a task objective with a specified minimum probability of success, $\omega \in [0,1)$. Specifically, we want to reach a belief state, b, such that

$$\Theta(b, r, x_g) = \int_{x \in B_n(r)} \pi(x + x_g; b) \geq \omega, \tag{2}$$

where $B_n(r) = \{x \in \mathbb{R}^n, x^T x \leq r^2\}$ denotes the r-ball in $\mathbb{R}^n$ for some $r > 0$, and ω denotes the minimum probability of success. There are strong similarities between this

[1] Although we have formally limited ourselves to the case of deterministic process noise, we find in Section 6 that empirically, our algorithm performs well in environments with limited amounts of process noise.

control problem and the more general Partially Observable Markov Decision Process (POMDP) problem. Both define a partially observable control problem. However, whereas the objective of a POMDP is to minimize expected cost, our objective is to reach a goal region with a specified minimum probability. Also, in contrast to the more general POMDP problem, we have only allowed deterministic process dynamics.

3 Algorithm

Our algorithm can be viewed as a receding horizon control approach that creates and executes nominal belief space plans. During execution, we track a belief distribution over underlying state based on actions and observations. If the true belief state diverges from the nominal trajectory, our algorithm re-plans and the process repeats. Our key contribution is a planning problem that, when solved optimally on each re-planning step, is guaranteed, under certain conditions, to enable the system to gain information.

3.1 Creating plans

The key to our approach is a mechanism for creating horizon-T belief-space plans that guarantee that new information is incorporated into the belief distribution on each planning cycle. Given a prior belief state, b_1, define a "hypothesis" state at the maximum of the pdf, $x^1 = \arg\max_{x \in \mathbb{R}^n} \pi(x; b_1)$. Then, sample $k-1$ states from the prior distribution, $x^i \sim \pi(x; b_1), i \in [2, k]$, such that the pdf at each sample is greater than a specified threshold, $\pi(x^i; b_1) \geq \varphi > 0$, and there are at least two unique states among the $k-1$. We search for a sequence of actions, $\mathbf{u}_{T-1} = (u_1, \ldots, u_{T-1})$, that result in as wide a margin as possible between the observations that would be expected if the system were in the hypothesis state and the observations that would be expected in any other sampled state. As a result, a good plan enables the system to "confirm" that the hypothesis state is in fact the true state or to "disprove" the hypothesis state. If the hypothesis state is disproved, then the algorithm selects a new hypothesis on the next re-planning cycle, ultimately causing the system to converge to the true state.

To be more specific, let $F_t(x, \mathbf{u}_{t-1})$ be the state at time t if the system begins in state x and takes actions $\mathbf{u}_{t-1}$. Recall that the expected observation upon arriving in state x_t is $h(x_t)$. Therefore, the expected sequence of observations is:

$$\mathbf{h}_t(x, \mathbf{u}_{t-1}) = \left(h(F_1(x, \mathbf{u}_1))^T, \ldots, h(F_{t-1}(x, \mathbf{u}_{t-1}))^T \right)^T .$$

We are interested in finding a sequence of actions that minimizes the probability of seeing the observation sequence expected in the sampled states when the system is actually in the hypothesis state. In other words, we want to find a sequence of actions, $\mathbf{u}_{T-1}$, that minimizes

$$\tilde{J}(x^1,\ldots,x^k,\mathbf{u}_{1:T-1}) = \sum_{i=2}^{k} N\left(\mathbf{h}(x^i,\mathbf{u}_{T-1})|\mathbf{h}(x^1,\mathbf{u}_{T-1}),\mathbb{Q}\right)$$

where $N(\cdot|\mu,\Sigma)$ denotes the Gaussian distribution with mean μ and covariance Σ and $\mathbb{Q} = diag(Q,\ldots,Q)$ is the block diagonal of measurement noise covariance matrices of the appropriate size. When this sum is small, Bayes filtering will more accurately be able to determine whether or not the true state is near the hypothesis in comparison to the other sampled states.

The above expression for observation distance is only defined with respect to the sampled points. However, we would like to “confirm” or “disprove” states in regions about the hypothesis and samples – not just the zero-measure points themselves. This can be incorporated to the first order by defining small Gaussian distributions in state space with covariance, V, about the samples and taking an expectation:

$$\begin{aligned} J(x^1,\ldots,x^k,\mathbf{u}_{1:T-1}) &= \sum_{i=2}^{k} \mathbb{E}_{y^i\sim N(\cdot|x^i,V),y^1\sim N(\cdot|x^1,V)} N\left(\mathbf{h}(y^i,\mathbf{u}_{T-1})|\mathbf{h}(y^1,\mathbf{u}_{T-1}),\mathbb{Q}\right) \\ &= \sum_{i=2}^{k} N\left(\mathbf{h}(x^i,\mathbf{u}_{T-1})|\mathbf{h}(x^1,\mathbf{u}_{T-1}),\Gamma(x^i,\mathbf{u}_{T-1})\right), \quad (3) \end{aligned}$$

$$\text{where} \quad \Gamma(x,\mathbf{u}_{T-1}) = 2\mathbb{Q} + \mathbf{H}_T(x,\mathbf{u}_{T-1})\mathbf{V}\mathbf{H}_T(x,\mathbf{u}_{T-1})^T + \mathbf{H}_T(x^1,\mathbf{u}_{T-1})\mathbf{V}\mathbf{H}_T(x^1,\mathbf{u}_{T-1})^T, \quad (4)$$

$\mathbf{H}_t(x,\mathbf{u}_{1:t-1}) = \partial\mathbf{h}_t(x,\mathbf{u}_{1:t-1})/\partial x$ denotes the Jacobian matrix of $\mathbf{h}_t(x,\mathbf{u}_{1:t-1})$ at x, and $\mathbf{V}$ is the appropriately sized block diagonal matrix of V. Rather than optimizing for $J(x^1,\ldots,x^k,\mathbf{u}_{1:T-1})$ (Equation 3) directly, we simplify the planning problem by dropping the normalization factor in the Gaussian and optimizing the exponential factor only. Let

$$\Phi(x^i,\mathbf{u}_{T-1}) = \|\mathbf{h}(x^i,\mathbf{u}_{T-1}) - \mathbf{h}(x^1,\mathbf{u}_{T-1})\|^2_{\Gamma(x^i,\mathbf{u}_{T-1})}.$$

The modified cost function is:

$$\bar{J}(x^1,\ldots,x^k,\mathbf{u}_{1:T-1}) = \frac{1}{k}\sum_{i=1}^{k} e^{-\Phi(x^i,\mathbf{u}_{T-1})}. \quad (5)$$

The optimization problem becomes:

Problem 1.

$$\begin{aligned} \text{Minimize} \quad & \bar{J}(x^1,\ldots,x^k,\mathbf{u}_{T-1}) + \alpha\mathbf{u}_{T-1}^T\mathbf{u}_{T-1} & (6) \\ \text{subject to} \quad & x^i_{t+1} = f(x^i_t,u_t), i\in[1,k] & (7) \\ & x^1_T = x_g, x^i_1 = x^i, i\in[1,k]. & (8) \end{aligned}$$

Equation 6 adds an additional quadratic cost on action that adds a small preference for short trajectories. The associated weighting parameter should be set to a small value ($\alpha \ll 1$). Problem 1 can be solved using a number of planning techniques such as rapidly exploring random trees [10], differential dynamic programming [8], or sequential quadratic programming [3]. We use sequential quadratic programming to solve the direct transcription [3] of Problem 1. Although direct transcription is only guaranteed to find locally optimal solutions, we have found that it works well for many of the problems we have explored. The direct transcription solution will be denoted

$$\mathbf{u}_{T-1} = \textsc{DirTran}(x^1, \ldots, x^k, x_g, T), \tag{9}$$

for samples, $x^1, \ldots, x^k$, goal state constraint, x_g, and time horizon, T. Note that the dimensionality of Problem 1 is nk – linear in the dimensionality of the underlying state space with a constant equal to the number of samples. This compares favorably with the approaches in [13, 6, 11] that must solve planning problems in n^2-dimensional spaces (number of entries in the covariance matrix).

3.2 Re-planning

After creating a plan, our algorithm executes it while tracking the belief state using the user-supplied belief-state update, G. If the actual belief state diverges too far from a nominal trajectory derived from the plan, then execution stops and a new plan is created. The overall algorithm is outlined in Algorithm 1. The outer *while* loop iteratively creates and executes plans until the planning objective (Equation 2) is satisfied. Step 2 sets the hypothesis state to the maximum of the prior distribution. Step 3 samples $k-1$ additional states. Step 4 of Algorithm 1 calls the CreatePlan function (Algorithm 2). CreatePlan has two steps. First, it solves Problem 1 with the final value (first condition, Equation 8) constraint. Then, CreatePlan calculates a corresponding belief trajectory forward by assuming that the hypothesis state is equal to the true state. If the resulting trajectory does not reach a belief state that satisfies the *while* loop condition in step 1 of Algorithm 1, then CreatePlan solves Problem 1 again, this time without the final value constraint. Steps 6 through 12 execute the plan. Step 9 updates the belief state given the new action and observation using the user-specified Bayes filter implementation. Step 10 breaks plan execution when the actual belief state departs too far from the nominal trajectory, as measured by the KL divergence, $D_1\left[\pi(\cdot; b_{t+1}), \pi(\cdot; \bar{b}_{t+1})\right] > \theta$. The second condition, $\bar{J}(x^1, \ldots, x^k, \mathbf{u}_{t-1}) < 1-\rho$, guarantees that the *while* loop does not terminate before a (partial) trajectory with cost $\bar{J} < 1$ executes. We show in Section 4 that the second condition guarantees that the algorithm makes "progress" on each iteration of the *while* loop.

```
Input : initial belief state, $b$, goal state, $x_g$, planning horizon, $T$, and belief-state update, $G$.
while $\Theta(b,r,x_g) < \omega$ do
    $x^1 = \arg\max_{x \in \mathbb{R}^n} \pi(x;b)$;
    $\forall i \in [2,k], x^i \sim \pi(x;b) : \pi(x^i;b) \geq \varphi$;
    $\bar{b}_{1:T}, \mathbf{u}_{T-1} =$ CreatePlan($b, x^1, \ldots, x^k, x_g, T$);
    $b_1 = b$;
    for $t \leftarrow 1$ to $T-1$ do
        execute action $u_t$, perceive observation $z_{t+1}$;
        $b_{t+1} = G(b_t, u_t, z_{t+1})$;
        if $D_1\left[\pi(x;b_{t+1}), \pi(x;\bar{b}_{t+1})\right] > \theta$ and $\bar{J}(\mathscr{G}, \mathbf{u}_{t-1}) < 1-\rho$ then
            break
        end
    end
    $b = b_{t+1}$;
end
```

Algorithm 1: Belief-space re-planning algorithm

```
Input : initial belief state, $b$, sample set, $x^1, \ldots, x^k$, goal region, $\mathscr{G}$, and time horizon, $T$.
Output: nominal trajectory, $\bar{b}_{1:T}$ and $u_{1:T-1}$
$\bar{u}_{1:T-1} =$ DirTran($x^1, \ldots, x^k, \mathscr{G}, T$);
$\bar{b}_1 = b$; $\forall t \in [1:T-1]$, $\bar{b}_{t+1} = G(\bar{b}_t, u_t, h(x_t^1))$;
if $\Theta(b, \mathscr{G}) \leq \omega$ then
    $\bar{u}_{1:T-1} =$ DirTran($x^1, \ldots, x^k, T$);
    $\bar{b}_1 = b$; $\forall t \in [1:T-1]$, $\bar{b}_{t+1} = G(\bar{b}_t, u_t, h(x_t^1))$;
end
```

Algorithm 2: CREATEPLAN procedure

3.3 Illustration

Figures 1 and 2 show a simple example that illustrates belief space planning. Figure 1 shows a horizontal-pointing laser mounted to the end-effector of a two-link robot arm. The end-effector is constrained to move only vertically along the dotted line. The laser points horizontally and measures the range from the end-effector to whatever object it "sees". There are two boxes and a gap between them. Box size, shape, and relative position are assumed to be perfectly known along with the distance of the end-effector to the boxes. The only uncertain variable in this problem is the vertical position of the end-effector measured with respect to the gap position. This defines the one-dimensional state of the system and is illustrated by the ver-

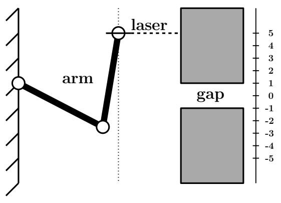


Fig. 1 SLAG scenario. The robot must simultaneously localize the gap and move the end-effector in front of the gap.

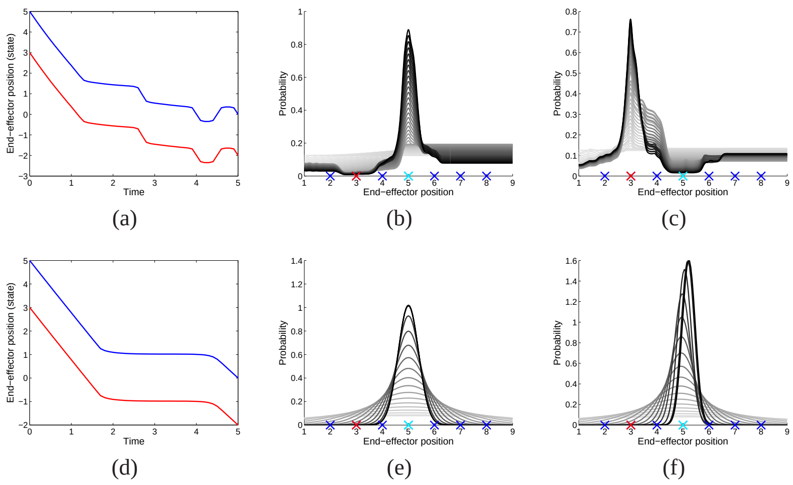


Fig. 2 Illustration of CREATEPLAN. (a) An information-gathering trajectory (state as a function of time) found using direct transcription. Blue denotes the trajectory that would be obtained if the system started in the hypothesis state. Red denotes the trajectory obtained starting in the true state. (b) The planned belief-space trajectory illustrated by probability distributions superimposed over time. Distributions early in the trajectory are light gray while distributions late in the trajectory are dark. The seven "X" symbols on the horizontal axis denote the positions of the samples (red denotes the true state while cyan denotes the hypothesis). (c) The actual belief-space trajectory found during execution. (d-f) Comparison with the EKF-based method proposed in [13]. (d) The planned trajectory. (e) The corresponding nominal belief-space trajectory. (f) Actual belief-space trajectory.

tical number line in Figure 1. The objective is to localize the vertical end-effector with respect to the center of the gap (state) exactly and move the end-effector to the center of the gap. The control input to the system is the vertical velocity of the end-effector.

Figure 2(a) illustrates an information-gathering trajectory found by DIRTRAN that is expected to enable the Bayes filter to determine whether the hypothesis state is indeed the true state while simultaneously moving the hypothesis to the goal state (end-effector at the center of the gap). The sample set used to calculate the trajectory was $x^1,\ldots,x^k = 5,2,3,4,6,7,8$, with the hypothesis sample located at $x^1 = 5$. The action cost used while solving Problem 1 was $\alpha = 0.0085$. DIRTRAN was initialized with a random trajectory. The additional small action cost smooths the trajectory by pulling it toward shortest paths without changing information gathering or goal directed behavior much. The trajectory can be understood intuitively. Given the problem setup, there are two possible observations: range measurements that "see" one of the two boxes and range measurements that "see" through the gap. The plan illustrated in Figure 2(a) moves the end effector such that different sequences of measurements would be observed depending upon whether the system were actually in the hypothesis state or in another sampled state.

Figures 2(b) and (c) show the nominal belief-space trajectory and the actual trajectory, respectively, in terms of a sequence of probability distributions superimposed on each other over time. Each distribution describes the likelihood that the system started out in a particular state given the actions taken and the observations perceived. The nominal belief-space trajectory in Figure 2(b) is found by simulating the belief-space dynamics forward assuming that future observations will be generated by the hypothesis state. Ultimately, the planned trajectory reaches a belief state distribution that is peaked about the hypothesis state, x^1 (the red “X”). In contrast, Figure 2(c) illustrates the actual belief-space trajectory found during execution. This trajectory reaches a belief state distribution peaked about the true state (the cyan “X”). Whereas the hypothesis state becomes the maximum of the nominal distribution in Figure 2(b), notice that it becomes a minimum of the actual distribution in Figure 2(c). This illustrates the main idea of the algorithm. Figure 2(b) can be viewed as a trajectory that “trusts” that the hypothesis is correct and takes actions that confirm this hypothesis. Figure 2(c) illustrates that even when the hypothesis is wrong, the distribution necessarily gains information because it “disproves” the hypothesis state (notice the likelihood of the region about the hypothesis is very low).

Figure 2 (d-f) compares the performance of our approach with the extended Kalman filter-based (EKF-based) approach proposed in [13]. The problem setup is the same in every way except that cost function optimized in this scenario is:

$$\bar{J}(u_{1:T-1}) = \frac{1}{10}\left(\sigma_T^2\right)^T \sigma_T^2 + 0.0085 u_{1:T-1}^T u_{1:T-1},$$

where σ_T^2 denotes covariance. There are several differences in performance. Notice that the two approaches generate different trajectories (compare Figures 2(a) and (d)). Essentially, the EKF-based approach maximizes the EKF reduction in variance by moving the maximum likelihood state toward the edge of the gap where the gradient of the measurement function is large. In contrast, our approach moves around the state space in order to differentiate the hypothesis from the other samples in regions with a small gradient. Moreover, notice that since the EKF-based approach is constrained to track actual belief state using an EKF Bayes filter, the tracking performance shown in Figure 2(f) is very bad. The EKF innovation term actually makes corrections in the wrong direction. However, in spite of the large error, the EKF covariance grows small indicating high confidence in the estimate.

4 Analysis

We are interested in the correctness of Algorithm 1. Can we guarantee that Algorithm 1 eventually reaches a belief state in the goal region? We show that if G is an exact implementation of Equation 1, then Algorithm 1 is expected to localize the true state of the system after a finite number of iterations of the outer loop. As the number of iterations of the outer loop goes to infinity, the probability of having localized the true system state goes to one. We start by providing a lower bound on the expected probability of states in a neighborhood of the true state. On a particular iteration of the outer *while* loop in Algorithm 1, suppose that the system begins in belief state, b_1, while the true state is κ, and executes a sequence of actions, $\mathbf{u} = (u_1, \dots, u_{T-1})$ (subscript dropped for conciseness). During execution, the system perceives observations $\mathbf{z} = (z_2, \dots, z_T)$ and ultimately arrives in belief state b_T. The probability of a state, $y = F_T(x, \mathbf{u})$, estimated by recursively evaluating Equation 1 is:

$$\pi(y; b_T) = \pi(x; b_1) \frac{q_x(\mathbf{z}, \mathbf{u})}{p(\mathbf{z}, \mathbf{u})}, \tag{10}$$

where

$$q_x(\mathbf{z}, \mathbf{u}) = N(\mathbf{z} | \mathbf{h}(x, \mathbf{u}), \mathbb{Q}) \tag{11}$$

is the probability of the observations given that the system starts in state x and takes actions, $\mathbf{u}$, and

$$p(\mathbf{z}, \mathbf{u}) = \int_{x \in \mathbb{R}^n} \pi(x; b_1) N(\mathbf{z} | \mathbf{h}(x, \mathbf{u}), \mathbb{Q}) \tag{12}$$

is the marginal probability of the observations given $\mathbf{u}$. The following Lemma shows that $\pi(y; b_T)$ can be lower-bounded in terms of the proximity of x to the true state, κ.

Lemma 1. *Suppose we are given an arbitrary sequence of actions,* $\mathbf{u}$, *and an arbitrary initial state,* $x \in \mathbb{R}^n$. *Then, the expected probability of* $y = F_T(x, \mathbf{u})$ *found by recursively evaluating the deterministic Bayes filter update (Equation 1) is*

$$\mathbb{E}_{\mathbf{z}} \left\{ \frac{\pi(y; b_T)}{\pi(x; b_1)} \right\} \geq \exp\left(D_1(q_\kappa, p) - D_1(q_\kappa, q_x) \right),$$

where q_κ, q_x, *and* p *are defined in Equations 11 and 12 and* D_1 *denotes the KL divergence between the arguments.*

Proof. The log of the expected change in the probability of x is:

$$\begin{aligned}
\log \mathbb{E}_{\mathbf{z}} \left\{ \frac{\pi(y; b_T)}{\pi(x; b_1)} \right\} &= \log \mathbb{E}_{\mathbf{z}} \left\{ \frac{q_x(\mathbf{z}, \mathbf{u})}{p(\mathbf{z}, \mathbf{u})} \right\} \\
&= \log \int_{\mathbf{z} \in \mathbb{R}^m} \frac{q_\kappa(\mathbf{z}, \mathbf{u}) q_x(\mathbf{z}, \mathbf{u})}{p(\mathbf{z}, \mathbf{u})} \\
&\geq \int_{\mathbf{z} \in \mathbb{R}^m} q_\kappa(\mathbf{z}) \left(\log q_x(\mathbf{z}, \mathbf{u}) - \log p(\mathbf{z}, \mathbf{u}) \right) \\
&= D_1(q_\kappa, p) - D_1(q_\kappa, q_x),
\end{aligned}$$

where the third line was obtained using Jensen's inequality and the last line follows from algebra. Taking the exponential gives us the claim.

Lemma 1 expresses the bound in terms of the divergence, $D_1(q_\kappa, p)$, with respect to the true state, κ. However, since κ is unknown ahead of time, we must find a lower bound on the divergence $D_1(q_y, p)$ for arbitrary values of y. The following lemma establishes a bound on this quantity. We use the notation that $\|a\|_A = \sqrt{a^T A^{-1} a}$ denotes the Mahalanobis distance with respect to A.

Lemma 2. *Given an arbitrary* $\mathbf{u}$ *and a distribution,* ϖ*, suppose* $\exists \Lambda_1, \Lambda_2 \subseteq \mathbb{R}^n$ *such that* $\forall x_1, x_2 \in \Lambda_1 \times \Lambda_2, \|\mathbf{h}(x_1,\mathbf{u}) - \mathbf{h}(x_2,\mathbf{u})\|^2_{\mathbb{Q}} \geq \zeta^2$ *and* $\int_{x\in\Lambda_1} \varpi(x) \geq \gamma$, $\int_{x\in\Lambda_2} \varpi(x) \geq \gamma$. *Then*

$$\min_{y\in\mathbb{R}^n} D_1(q_y, p) \geq 2\eta^2\gamma^2 \left(1 - e^{-\frac{1}{2}\zeta^2}\right)^2,$$

where $\eta = 1/\sqrt{(2\varpi)^n |\mathbb{Q}|}$ *is the Gaussian normalization constant.*

Proof. By Pinsker's inequality, we know that $D_1(q_x, p) \geq 2\sup_{\mathbf{z}} (q_x(\mathbf{z},\mathbf{u}) - p(\mathbf{z},\mathbf{u}))^2$. Notice that $p(\mathbf{h}(x_1,\mathbf{u})) \leq \eta\left(1 - \gamma + \gamma e^{-\frac{1}{2}\zeta^2}\right)$. Since $q_x(\mathbf{h}(x_1,\mathbf{u})) = \eta$, we have:

$$(q_x(\mathbf{h}(x_1,\mathbf{u})) - p(\mathbf{h}(x_1,\mathbf{u})))^2 \geq \gamma^2 \left(1 - e^{-\frac{1}{2}\zeta^2}\right)^2.$$

We obtain the conclusion by using Pinsker's inequality.

As a result of Lemmas 1 and 2, we know that we can lower bound the expected increase in probability of a region about the true state by finding regions, Λ_1 and Λ_2, that satisfy the conditions of Lemma 2 for a given $\mathbf{u}$. Lemma 3 shows that these regions exist for any $\mathbf{u}$ with a cost (Equation 3) $\bar{J} < 1$. The proof uses Lemmas 4 and 5, stated and proved in the appendix.

Lemma 3. *Suppose that* $\mathbf{u}$ *is a plan with cost* $\bar{J} = \bar{J}(x^1, \ldots, x^k, \mathbf{u})$ *defined over the samples,* $x^i, i \in [1,k]$. *If the maximum eigenvalue of the Hessian of* h *is* λ*, then* $\exists i \in [1,k]$ *such that:*

$$\forall r \in \mathbb{R}^+, \forall \delta_1, \delta_2 \in B_n(r), \|\mathbf{h}(x^i + \delta_1) - \mathbf{h}(x^1 + \delta_2)\|^2_{\Gamma(x^i,\mathbf{u})} \geq \left[\sqrt{-\log \bar{J}} - 2(r + cr^2)\right],$$

where $c = \lambda\|\mathbf{1}\|_{\mathbb{Q}}/2$ *and* $B_n(r) = \{x \in \mathbb{R}^n; x^T x \leq r^2\}$.

Proof. Considering Equation 3, we know that a cost, $\bar{J}$, implies that there is at least one sample, x^j, such that

$$\begin{aligned} -\log \bar{J} &\leq \Phi(x^j, \mathbf{u}) \\ &= \|\mathbf{h}(x^j) - \mathbf{h}(x^1)\|^2_{\Gamma(x^j,\mathbf{u})}. \end{aligned}$$

Notice that $\forall y \in \mathbb{R}^n$, the matrix $\mathbf{H}(y)^T \Gamma(y,\mathbf{u})^{-1} \mathbf{H}(y)$ is positive semidefinite with eigenvalues no greater than one. Therefore, we know that $\forall r \in \mathbb{R}^+, \delta \in B_n(r)$,

$$\|\mathbf{H}(y)\delta\|^2_{\Gamma(y,\mathbf{u})} \leq r^2.$$

Using Lemma 4 twice to combine the above equations, we have $\forall r \in \mathbb{R}^+, \delta_1, \delta_2 \in B_n(r)$,

$$\|P(x^j,\delta_1) - P(x^1,\delta_2)\|^2_{\Gamma(x^j,\mathbf{u})} \geq \left(\sqrt{-\log \bar{J}} - 2r\right)^2,$$

where $P(x,\delta) = \mathbf{h}(x) - \mathbf{H}(x)\delta$. Using Lemma 5, we have that $\forall x, \in \mathbb{R}^n$ and $\delta \in B_n(r)$,

$$\|\mathbf{h}(x+\delta) - P(x,\delta)\|^2_{\Gamma(x,\mathbf{u})} \leq (cr^2)^2.$$

Applying Lemma 4 twice gives us the conclusion of the theorem.

We now state our main theoretical result regarding Algorithm 1 correctness. Two conditions must be met. First, the planner in step 4 of Algorithm 1 must always find low-cost plans successfully. Essentially, this guarantees that each plan will acquire useful information. Second, step 8 of Algorithm 1 must use an exact implementation of the Bayes filter. In practice, we expect that this second condition will rarely be met. However, our experiments indicate that good results can be obtained using practical filter implementations (Section 5).

Theorem 1. *Suppose $\exists r, \varepsilon \in \mathbb{R}^+$ such that $\forall i \in [1,k]$ and $\forall \delta \in B_n(r)$, $\pi(x^i+\delta) \geq \varepsilon$ with $k \geq 2$. Suppose:*

1. DIRTRAN *(Algorithm 1, step 4) always finds a horizon-T trajectory, $\mathbf{u}$, with cost,*

$$\bar{J}(x^1,\ldots,x^k,\mathbf{u}) \leq \exp\left[-\left(2r + r^2\lambda_h\lambda_f^{T-1}\|\mathbf{1}\|_{\mathbb{Q}} + \sqrt{\log\varphi^2}\right)^2\right],$$

   where λ_h and λ_f are the maximum eigenvalues of the Hessian matrix of h and f, respectively and $\varphi > 1$ is the threshold parameter in step 3 of Algorithm 1; and
2. *G is an exact implementation of the Bayesian filter update (Equation 1) in step 8 of Algorithm 1.*

Then, when Algorithm 1 executes,

1. *the expected probability of the true state increases on each iteration of the outer* while *loop by at least $2\eta^2\gamma^2(1-1/\varphi)$, where $\eta = 1/\sqrt{(2\pi)^n|\mathbb{Q}|}$ is the Gaussian normalization constant, $\gamma = \varepsilon Vol_n(r)$, and $Vol_n(r)$ is the volume of the r-ball in n dimensions; and*
2. *as the number of iterations of the outer while loop goes to infinity, the true state becomes the maximum of the belief state distribution with probability one.*

Proof. Condition 2 in the premise implies that

$$\sqrt{-\log\bar{J}} - 2r - r^2\lambda_h\lambda_f^{\tau}\|\mathbf{1}\|_{\mathbb{Q}} \geq \sqrt{\log\varphi^2}.$$

Lemma 3, gives us that $\exists i \in [1,k]$ such that $\forall \delta_1,\delta_2 \in B(r)$, $\|\mathbf{h}(x^i+\delta_1) - \mathbf{h}(x^1+\delta_2)\|^2_{\Gamma(x^i,\mathbf{u})} \geq \sqrt{\log\varphi^2}$. Then, Lemma 2 gives us that

$$\min_{y\in\mathbb{R}^2} D_1(q_y,p) \geq 2\eta^2\gamma(1-1/\varphi)^2,$$

where $\gamma = \varepsilon Vol_n(r)$. Lemma 1 gives us the first conclusion. The constraint that $\varphi > 1$ implies that the right side of the above equation is positive. As a result, the probability of the true state is expected to increase on each iteration of the outer *while* loop and we have the second conclusion.

At the end of Section 3.1, we noted that the planning problem solved in step 4 of Algorithm 1 was linear in the dimensionality of the underlying space. Theorem 1 asserts that the algorithm is correct with as few as two samples. As a result, we know that the linear constant can be as small as two.

5 Experiments

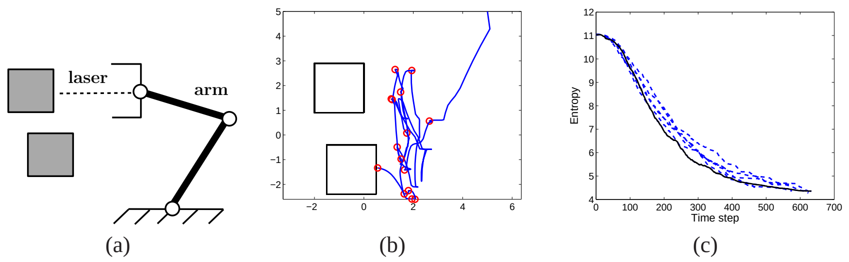


Fig. 3 (a) the experimental scenario. (b) a path found by Algorithm 1 with a nine-sample planner. It starts in the upper right and ends at a point directly in front of the right-most box. The red circles denote where re-planning occurred. (c) belief state entropy as a function of time step. The solid black line corresponds to the trajectory shown in (b). The dashed blue lines correspond to five additional nine-sample runs.

From a practical perspective, the preceding analysis is useful because it tells us that if we execute the *while* loop in Algorithm 1 a sufficient number of times, we can expect to localize the state of the system with arbitrary accuracy (we can drive $\Theta(b, r, x_g)$ arbitrarily low). However, for this result to hold, we require the planner to find low cost paths each time it is called and for the tracking Bayes filter to be an exact realization of Equation 1 (the premise of Theorem 1). Since these conditions are difficult to meet in practice, an important question is how well the approach works for approximately accurate Bayes filter implementations and for planners that only succeed some of the time. Furthermore, we are interested in knowing how the performance of the algorithm changes with the number of samples used to parametrized the planner. Figure 3(a) illustrates the experimental scenario. A two-link robot arm moves a hand in the plane. A single range-finding laser is mounted at the center of the hand. The laser measures the range from the end-effector to whatever object it “sees”. The hand and laser are constrained to remain horizontal. The position of the hand is assumed to be measured perfectly. There are two boxes of known size but unknown position to the left of the robot (four dimensions of unobserved state). The boxes are constrained to be aligned with the coordinate frame (they cannot rotate). The control input to the system is the planar velocity of the end-effector. The objective is for the robot to localize the two boxes using its laser and move the end-effector to a point directly in front of the right-most box (the box with the largest *x*-coordinate) so that it can grasp by extending and closing the gripper. On each time step, the algorithm specified the real-valued two-dimensional hand velocity and perceived the laser range measurement. If the laser missed both boxes, a zero measurement was perceived. The (scalar) measurements were corrupted by zero-mean Gaussian noise with 0.31 standard deviation.

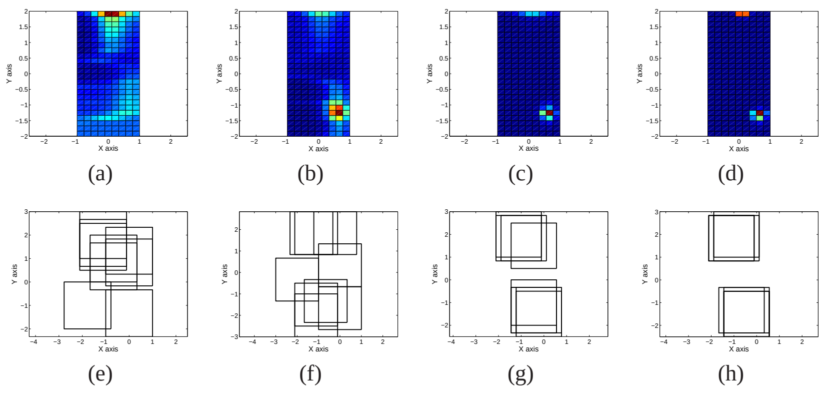


Fig. 4 Histogram probability distributions (a-d) and planner sample sets (e-h) at time steps 10, 100, 200, and 300 during the path shown in Figure 3(b).

Figure 3(b) illustrates the path of the hand (a point directly between the two jaws of the gripper) found by running our algorithm parametrized by nine samples. The state space was four dimensional and comprised of two box locations ranging between $[-1,1]$ on the *x*-axis and $[-2,2]$ on the *y*-axis. The hand starts in the upper right corner at $(5,5)$ and ends at a point directly in front of the lower right box. The blue line shows the path and the red circles identify the points along the path at which re-planning occurred (there are 14 re-plan events in this example). The tracking Bayes filter was implemented using a gridded histogram filter comprised of 62500 bins over the four-dimensional space (the position of each of the two boxes was denoted by a point in a 10×25 grid). At the start of planning, the prior histogram distribution was assumed to be uniform. The cost function optimized by the DIRTRAN planner (Equation 6) was parametrized by $\alpha = 0.01$ and $\mathbf{V} = diag(0.5)$ (Equations 3 and 4). The planning horizon was $T = 50$. The algorithm did not terminate until the histogram Bayes filter was 90% confident that it had localized the right-most box to within ± 0.3 of its true location ($\omega = 0.9$ in step

1 of Algorithm 1). Figure 4(a)-(d) show snapshots of the histogram distribution at time steps 10, 100, 200, and 300. (This is actually a two-dimensional projection of the four dimensional distribution illustrating the distribution over the location of *one* box only.) Figure 4(e)-(h) show the nine samples used to parametrize the planning algorithm at the four snapshots. Initially, (in Figures 4 (a) and (e), the distribution is high-entropy and the samples are scattered through the space. As time increases, the distribution becomes more peaked and the sample sets become more focused. The solid black line in Figure 3(b) shows the entropy of the histogram distribution as a function of time step. As expected, entropy decreases significantly over the trajectory. For comparison, the five additional blue dotted lines in Figure 3(c) show entropy results from five additional identical experiments. Note the relatively small variance amongst trajectories. Even though the algorithm finds a very different trajectory on each of these runs, performance is similar. These results help answer two of the questions identified at the beginning of the section. First, Figure 4 suggests that in at least one case, the histogram filter was adequate to represent the belief state in the context of this algorithm even though it is a coarsely discretized approximation to the true distribution. The black line in Figure 3(c) suggests that DIRTRAN was an effective tool for planning in this scenario. The six additional runs illustrated in Figure 3(c) indicate that these results are typical.

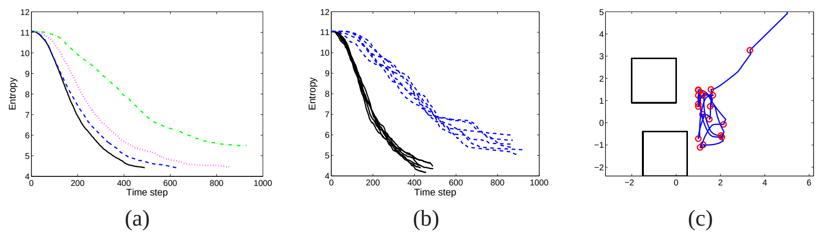


Fig. 5 (a) comparison of entropy averaged over six runs for four different planner sample set sizes (36 samples, solid black line; 9 samples, dashed blue line; 4 samples, dotted magenta line; 2 samples, dash-dot green line). (b) comparison of the six thirty-six-sample runs (solid black) with the six two-sample runs (dashed blue). (c) a path found using a two-sample planner.

The other question to be answered concerns the effect of the number of samples on algorithm performance. To find an answer, we have run the algorithm in the scenario described above for four contingencies where the planner was parametrized by two, four, nine, and thirty-six samples. Figure 5(a) compares the average (over six runs each) information-gathering performance for the four contingencies. Although increasing the number of samples improves algorithm performance, the gains diminish as the number of samples increases. Figure 5(b) compares the two-sample runs with the thirty-six-sample runs and demonstrates that the improvement is statistically significant. The comparison of Figure 5(c) with Figure 3(b) suggests that (in this experiment, at least) the trajectories produced by the high-sample planner are better than those produced by the low-sample planner because the high-sample planner does a better job covering the space in front of the boxes. These results show that it is valuable to expend computational effort planning an information-gathering trajectory, even in this simple example. The results also show that the performance of our algorithm smoothly degrades or improves with fewer or more samples used during planning. Even with the minimum of two samples, the algorithm is capable of making progress.

6 Robot Grasping application

We apply our approach to an instance of the robot grasping problem where it is necessary to localize and grasp a box. We refer to this version of the problem, where perception is incorporated into the problem statement, as "simultaneous localization and grasping" (SLAG). Two boxes of unknown dimensions are presented to the robot. The objective is to localize and grasp the box which is initially found directly in front of the left paddle. This is challenging because the placement of the two boxes may make localization of the exact position and dimensions of the boxes difficult.

6.1 Problem setup

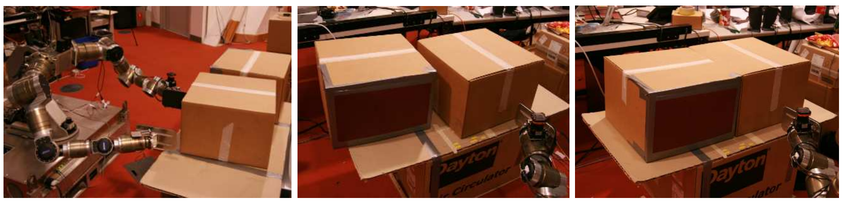

(a) (b) (c)

Fig. 6 Illustration of the grasping problem, (a). The robot must localize the pose and dimensions of the boxes using the laser scanner mounted on the left wrist. This is relatively easy when the boxes are separated as in (b) but hard when the boxes are pressed together as in (c).

Our robot, *Paddles*, has two arms with one paddle at the end of each arm (see Figure 6(a)). Paddles may grasp a box by squeezing the box between the two paddles and lifting. We assume that the robot is equipped with a pre-programmed "lift" function that can be activated once the robot has placed its two paddles in opposition around the target box. Paddles may localize objects in the world using a laser scanner mounted to the wrist of its left arm. The laser scanner produces range data in a plane parallel to the tabletop over a 60 degree field of view.

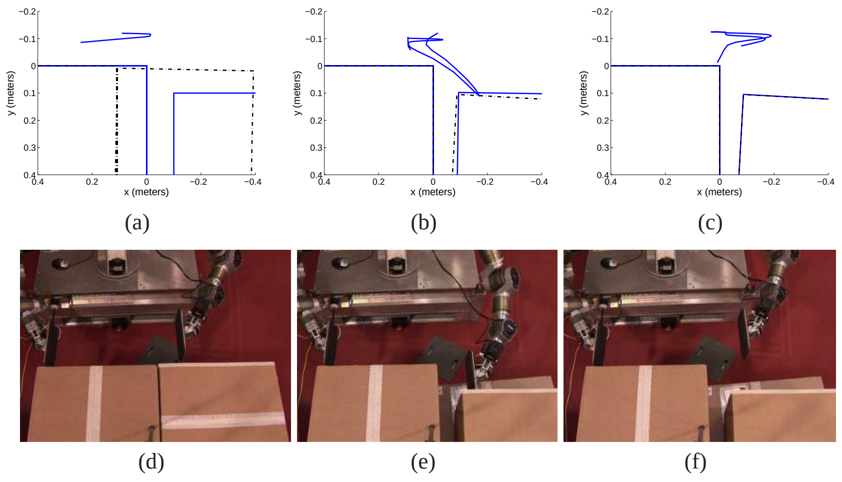


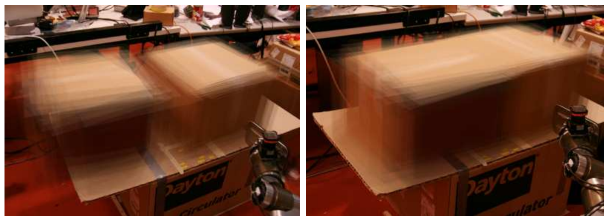
Fig. 7 Example of a box localization task. In (a) and (d), the robot believes the gap between the boxes is large and plans to localize the boxes by scanning this gap. In (b) and (e), the robot has recognized that the boxes abut each other and creates a plan to increase gap width by pushing the right box. In (c) and (f), the robot localizes the boxes by scanning the newly created gap.

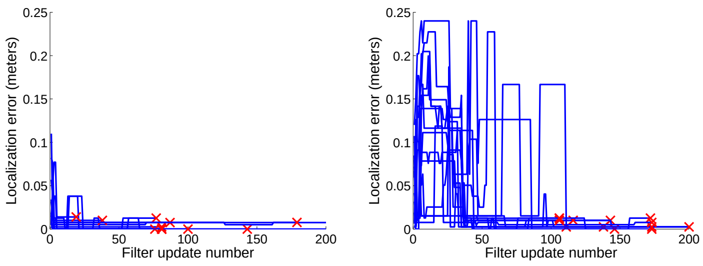
We use Algorithm 1 to localize the planar pose of the two boxes parametrized by a six-dimensional underlying metric space. The boxes are assumed to have been placed at a known height. We reduce the dimensionality of the planning problem by introducing an initial perception step that localizes the depth and orientation of the right box using RANSAC [7]. From a practical perspective, this is a reasonable simplification because RANSAC is well-suited to finding the depth and orientation of a box that is assumed to be found in a known region of the laser scan. The remaining (four) dimensions that are not localized using RANSAC describe the horizontal dimension of the right box location and the three-dimensional pose of the left box. These dimensions are localized using a Bayes filter that updates a histogram distribution over the four-dimensional state space based on laser measurements and arm motions measured relative to the robot. The histogram filter is comprised of 20000 bins: 20 bins (1.2 cm each) describing right box horizontal position times 10 bins (2.4 cm each) describing left box horizontal position times 10 bins (2.4 cm each) describing left box vertical position times 10 bins (0.036 radians each) describing left box orientation. While it is relatively easy for the histogram filter to localize the remaining four dimensions when the two boxes are separated by a gap (Figure 6(b)), notice that this is more difficult when the boxes are pressed together (Figure 6(c)). In this configuration, the laser scans lie on the surfaces of the two boxes such that it is difficult to determine where one box ends and the next begins. Note that it is difficult to locate the edge between abutting boxes reliably using vision or other sensor modalities – in general this is a hard problem.

Our implementation of Algorithm 1 used a set of 15-samples including the hypothesis sample. The algorithm controlled the left paddle by specifying Cartesian end-effector velocities in the horizontal plane. These Cartesian velocity commands were projected into the joint space using standard Jacobian Pseudoinverse techniques [16]. The algorithm was parametrized by process dynamics that modeled arms motions resulting from velocity commands and box motions produced by pushes from the arm. Box motions were modeled by assuming zero slip while pushing the box and assuming the center of friction was located at the center of the area of the box “footprint”. The observation dynamics described the set of range measurements expected in a given paddle-box configuration. For planning purposes, the observation dynamics were simplified by modeling only a single forward-pointing scan rather than the full 60 degree scan range. However, notice that since this is a conservative estimate of future perception, low cost plans under the simplified observation dynamics are also low cost under the true dynamics. Nevertheless, the observation model used for *tracking* (step 8 of Algorithm 1) accurately described measurements from all (100) scans over the 60 degree range. The termination threshold in Algorithm 1 was set to 50% rather than a higher threshold because we found our observation noise model to overstate the true observation noise.

Our hardware implementation of the algorithm included some small variations relative to Algorithm 1. Rather than monitoring divergence explicitly in step 9, we instead monitored the ratio between the likelihood of the hypothesis state and the next most probable bin in the histogram filter. When this ratio fell below 0.8, plan execution was terminated and the *while* loop continued. Since the hypothesis state must always have a maximal likelihood over the planned trajectory, a ratio of less than one implies a positive divergence. Second, rather than finding a non-goal directed plan in steps 3-5 of Algorithm 2, we always found goal-directed plans.

Figure 7 illustrates an example of an information-gathering trajectory. The algorithm begins with a hypothesis state that indicates that the two boxes are 10 cm apart (the solid blue boxes in Figure 7(a)). As a result, the algorithm creates a plan that scans the laser in front of the two boxes under the assumption that this will enable the robot to perceive the (supposed) large gap. In fact, the two boxes abut each other as indicated by the black dotted lines in Figure 7(a). After beginning the scan, the histogram filter in Algorithm 1 recognizes this and terminates execution of the initial plan. At this point, the algorithm creates the pushing trajectory illustrated in Figure 7(b). During execution of the push, the left box moves in an unpredicted way due to uncertainty in box friction parameters (this is effectively process noise). This eventually triggers termination of the second trajectory. The third plan is created based on a new estimate of box locations and executes a scanning motion in front of the boxes is expected to enable the algorithm to localize the boxes with high confidence.


(a) (b)



(c) (d)

Fig. 8 "Easy" and "hard" experimental contingencies. (a) shows images of the 12 randomly selected "easy" configurations (both box configurations chosen randomly) superimposed on each other. (b) shows images of the 12 randomly selected "hard" configurations (boxes abutting each other). (c) and (d) are plots of error between the maximum a posteriori localization estimate and the true box pose. Each line denotes a single trial. The red "X" marks denote localization error at algorithm termination.

6.2 Localization Performance

At a high level, the objective of SLAG is to robustly localize and grasp objects even when the pose or shape of those objects is uncertain. We performed a series of experiments to evaluate how well this approach performs when used to localize boxes that are placed in initially uncertain locations. On each grasp trial, the boxes were placed in a uniformly random configuration (visualized in Figures 8(a) and (c)). There were two experimental contingencies: "easy" and "hard". In the easy contingency, both boxes were placed randomly such that they were potentially separated by a gap. The right box was randomly placed in a 13×16 cm region over a range of 15 degrees. The left box was placed uniformly randomly in a 20×20 cm region over 20 degrees measured with respect to the right box (Figure 8(a)). In the hard contingency, the two boxes were pressed against each other and the pair was placed randomly in a 13×16 cm region over a range of 15 degrees (Figure 8(b)).

Figures 8(c) and (d) show right box localization error as a function of the number of updates to the histogram filter since the trial start. 12 trials were performed in each contingency. Each blue line denotes the progress of a single trial. The termination of each trial is indicated by the red "X" marks. Each error trajectory is referenced to the ground truth error by measuring the distance between the final position of the paddle tip and its goal position in the left corner of the right box using a ruler. There are two results of which to take note. First, all trials terminate with less than 2 cm of error. Some of this error is a result of the coarse discretization of possible right box positions in the histogram filter (note also the discreteness of the error plots). Since the right box position bin size in the histogram filter is 1.2 cm, we would expect a maximum error of at least 1.2 cm. The remaining error is assumed to be caused by errors in the range sensor or the observation model. Second, notice that localization occurs much more quickly (generally in less than 100 filter updates) and accurately in the easy contingency, when the boxes are initially separated by a gap that the filter may used to localize. In contrast, accurate localization takes longer (generally between 100 and 200 filter updates) during the hard contingency experiments. Also error prior to accurate localization is much larger reflecting the significant possibility of error when the boxes are initially placed in the abutting configuration. The key result to notice is that even though localization may be difficult and errors large during a “hard” contingency, all trials ended with a small localization error. This suggests that our algorithm termination condition in step 1 of Algorithm 1 was sufficiently conservative. Also notice that the algorithm was capable of robustly generating information gathering trajectories in all of the randomly generated configurations during the “hard” contingencies. Without the box pushing trajectories found by the algorithm, it is likely that some of the hard contingency trials would have ended with larger localization errors.

7 Discussion

Creating robots that can function robustly in unstructured environments has always been a central objective of robotics. In order to achieve this, it is necessary to develop algorithms capable of actively localizing the state of the world while also reaching task objectives. We introduce an algorithm that achieves this by planning in belief-space, the space of probability distributions over the underlying state space. Crucially, our approach is capable of reasoning about trajectories through a non-Gaussian belief-space. The fact that we can plan effectively non-Gaussian belief spaces makes our algorithm different than most other belief space planning algorithms currently in the literature. The non-Gaussian aspect is essential because in many robot problems it is not possible to track belief state accurately by projecting onto an assumed Gaussian density function (this is the case, for example, in the two-box example described in this paper). This paper provides a novel sufficient condition for guaranteeing that the probability of the true state found by the Bayes filter increases (Lemma 1). We show that our algorithm meets these conditions and, as a result, converges to the true state with probability one (Theorem 1). Although our theoretical results hold only under strict conditions, our experiments indicate that the algorithm performs well in practice. We empirically characterize the effect of changing the number of samples used to parametrize the algorithm on the resulting solution quality. We find that algorithm performance is nearly optimized using very few (between two and nine) samples and that, as a result, the planning step in our algorithm is computationally efficient. Finally, we illustrate our approach in the context of a robot grasping problem where a robot must simultaneously localize and grasp and object that is known only to be found somewhere in front of the robot.

References

1. Y. Bar-Shalom. Stochastic dynamic programming: Caution and probing. *IEEE Transactions on Automatic Control*, pages 1184–1195, October 1981.
2. D. Bayard and A. Schumitzky. Implicit dual control based on particle filtering and forward dynamic programming. *Int'l Journal of Adaptive Control and Signal Processing*, 24(3):155–177, 2010.
3. J. Betts. *Practical methods for optimal control using nonlinear programming*. Siam, 2001.
4. A. Doucet, N. Freitas, and N. Gordon, editors. *Sequential monte carlo methods in practice*. Springer, 2001.
5. N. Du Toit and J. Burdick. Robotic motion planning in dynamic, cluttered, uncertain environments. In *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2010.
6. T. Erez and W. Smart. A scalable method for solving high-dimensional continuous POMDPs using local approximation. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2010.
7. M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.
8. D. Jacobson and D. Mayne. *Differential dynamic programming*. Elsevier, 1970.
9. H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of Robotics: Science and Systems (RSS)*, 2008.
10. S. LaValle and J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.
11. S. Miller, A. Harris, and E. Chong. Coordinated guidance of autonomous uavs via nominal belief-state optimization. In *American Control Conference*, pages 2811–2818, 2009.
12. C. Papadimitriou and J. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.
13. R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2010.
14. S. Prentice and N. Roy. The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance. In *12th International Symposium of Robotics Research*, 2008.
15. S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *The Journal of Machine Learning Research*, 32:663–704, 2008.
16. L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Springer, 2000.
17. T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc. Uncertainty in Artificial Intelligence*, 2005.
18. J. Van der Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. In *Proceedings of Robotics: Science and Systems (RSS)*, 2010.

Appendix

Lemma 4. *If* $\|x\|_A^2 \geq \theta^2$, $\|\delta\|_A^2 \leq \varepsilon^2$, *and* $\theta \geq \varepsilon$, *then* $\|x-\delta\|_A^2 \geq (\theta - \varepsilon)^2$, *where* $x, \delta \in \mathbb{R}^n$, $\theta, \varepsilon \in \mathbb{R}^+$, *and* $A = A^T \geq 0$,

Proof. By the triangle inequality, we have $\|x\|_A \leq \|\delta\|_A + \|x-\delta\|_A$. Rearranging, this becomes $\|x-\delta\|_A \geq \|x\|_A - \|\delta\|_A$. We obtain the conclusion by squaring both sides and substituting θ and ε.

Lemma 5. *If* $\mathbf{f}(x) = (f_1(x), \ldots, f_n(x))^T$ *is a vector-valued function with Jacobian matrix* F, *and each scalar-valued component,* f_i, *has a Hessian matrix with a maximum eigenvalue of* λ_f, *then* $\forall x \in \mathscr{X}, \delta \in B_n(r)$,

$$\|f(x+\delta) - P(x,\delta)\|_A^2 \leq \frac{r^4 \lambda_f^2}{4} \|\mathbf{1}\|_A^2,$$

where $\mathbf{1}$ *is a column vector of n ones,* $P(x,\delta) = f(x) + F(x)\delta$ *is the first-order Taylor expansion of* $\mathbf{f}$, λ_A *is the maximum eigenvalue of A, and* $B_n(r)$ *is the r-ball in dimension n.*

Proof. For all $i \in [1,n]$, the Taylor remainder is $R_i(x,\delta) = f(x+\delta) - P(x,\delta)$. By the Taylor remainder theorem, we know that $|R_i(x,\delta)| \leq \frac{1}{2}\delta^T C_i \delta$, where C_i is the Hessian of f_i. Notice that $\forall \delta \in B_n(r)$, $\delta^T C_i \delta \leq r^2 \lambda_f$. Let $R(x,\delta) = (R_1(x,\delta), \ldots R_n(x,\delta))^T$. Then $\|R(x,\delta)\|_A^2 \leq \frac{r^4 \lambda_f^2}{4} \|\mathbf{1}\|_A^2$ and we have the conclusion.