Efficient planning in non-Gaussian belief spaces and its application to robot grasping

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

Abstract The limited nature of robot sensors make many important robotics problems partially observable. These problems may require the system to perform complex information-gathering operations. 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 of belief state and the true belief state. This paper proposes a new sample-based approach to belief-space planning that has fixed computational complexity while allowing arbitrary implementations of Bayes filtering to be used to track belief state. The approach is illustrated in the context of a simple example and compared to a prior approach. Then, we propose an application of the technique to an instance of the grasp synthesis problem where a robot must simultaneously localize and grasp an object given initially uncertain object parameters by planning information-gathering behavior. Experimental results are presented that demonstrate the approach to be capable of actively localizing and grasping boxes that are presented to the robot in uncertain and hard-to-localize configurations.

1 Introduction

A fundamental objective of robotics is to develop systems that can perform tasks robustly even in unstructured environments. One way to achieve this is to create a planner capable of simultaneously localizing the state of the system and of reaching a particular goal state. It is common to model control problems such as these as partially observable Markov decision processes (POMDPs). However, in general, find-

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

ing optimal solutions to POMDPs has been shown to be PSPACE complete [1]. 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 [2, 3, 4]. These algorithms calculate policies in *belief-space*, the space of probability distributions over the underlying state space. Very few of these algorithms can handle continuous state and action spaces [5, 6].

In an effort to avoid the computational complexity of creating policies, a new set of approaches have recently been proposed which create plans based on expected information content. In one class of approaches, large numbers of candidate trajectories in the underlying state space are evaluated in terms of the information that is likely to be gained during execution [7, 8, 9]. Trajectories are selected that optimize information content or minimize the likelihood of collisions. These approaches work well in scenarios where the likelihood of generating information-gathering trajectories by sampling the underlying space is high. A different class of approaches create plans in a parametrization of belief-space [10, 11, 12]. These approaches are potentially better positioned to generate complex information-gathering plans, but since they plan directly in the belief-space, the dimensionality of the planning problem is potentially very large. With the exception of [12], the planning approaches listed above assume that Bayes filtering will be performed using a Gaussian density function [10, 11, 7, 8, 9]. However, the popularity of the particle filter relative to the extended Kalman filter or unscented Kalman filter suggests that in many robot problems, belief state is not well-represented as a Gaussian. Furthermore, simply extending an approach such as in [10, 11] to non-Gaussian distributions quickly results in an intractable planning problem because of the high dimensionality of typical non-Gaussian parametrizations.

This paper proposes an approach to planning in high-dimensional belief-spaces that tracks belief state using an accurate, high-dimensional filter, but creates plans using a fixed-dimensional sampled representation of belief. We leave the implementation of the high-dimensional filter as a design choice, but expect that it will be a histogram filter or a particle filter. In order to create a new plan, the high-dimensional belief state is projected onto a hypothesis in the underlying state space and a set of sampled competing states. Plans are created that generate observations that differentiate the hypothesis from the other samples while also reaching a goal state. During execution, we monitor KL divergence between the actual (high-dimensional) belief-space trajectory and a belief-space trajectory associated with the plan. If divergence exceeds a threshold, we halt execution and create a new plan starting from the current belief (this re-planning approach is similar to that taken in [10, 11]). In a technical report that expands upon this paper, we have shown that if each new plan found has a below-threshold cost, then the algorithm eventually localizes the true state of the system and reaches a goal region with probability one [13]. We illustrate the approach in the context of a one-dimensional manipulation problem and compare it to the approach proposed in [10]. Then, we show that the approach can be used to solve a version of the grasp synthesis problem where the robot must simultaneously localize and grasp an object. The algorithm generates robot arm trajectories that gain information by "scanning" the boxes using a laser

scanner and pushing one of the boxes as necessary in order to gain information. The algorithm terminates in a pre-grasp configuration that is likely to lead to a successful grasp. The approach is tested over a range of randomly selected box configurations.

2 Problem Statement

This paper is concerned with the problem of reaching a desired goal state when the initial state is uncertain and may only be estimated based on partial or noisy observations. Consider a discrete-time system with continuous non-linear deterministic [1] process dynamics,

$$x_{t+1} = f(x_t, u_t), \tag{1}$$

where state, $x \in \mathbb{R}^n$, and action, $u \in \mathbb{R}^l$, are column vectors. At each time step, the system makes an observation, $z \in \mathbb{R}^m$, that is a non-linear stochastic function of state:

$$z_t = h(x_t) + v_t, \tag{2}$$

where $v_t \sim N(0, Q)$ is zero-mean Gaussian noise with variance Q.

Bayesian filtering can be used to estimate state based on actions taken and observation perceived. The state estimate is represented by a probability distribution function, $\pi(x;b)$, that is a function of the parameter vector, $b \in \mathscr{B}$. We will refer to b, (and sometimes the probability distribution, $\pi(x;b)$) as the *belief state*. Suppose that at time t, the system starts in belief state, b_t, takes action, u_t, and perceives observation, z_{t+1}. Then, belief state can be updated to incorporate the new information using the Bayesian filter update equation. For deterministic process dynamics, it is:

$$\pi(f(x,u_t);b_{t+1}) = \frac{\pi(x;b_t)P(z_{t+1}|x,u_t)}{P(z_{t+1})}, \tag{3}$$

where we implicitly assume that $P(z_{t+1}) \neq 0$. Although, in general, it is impossible to implement Equation 3 exactly using a finite-dimensional parametrization of belief-space, a variety of approximations exist in practice [14].

The objective of belief-space planning is to achieve task objectives with a given minimum probability. Specifically, we want to reach a belief state, b, such that

$$\Theta(b,r,x_g) = \int_{x \in B_n(r)} \pi(x + x_g; b) > \omega, \tag{4}$$

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 $r > 0, x_g \in \mathbb{R}^n$ denotes the goal state, and ω denotes the minimum probability of success. It is important to notice the similarities between this problem and the more general partially observable Markov decision process (POMDP) framework. Both problems are

[1] Although we have formally limited ourselves to the case of zero process noise, we find in Section 4 that empirically, our algorithm performs well in environments with bounded process noise.

concerned with controlling partially observable systems. However, whereas in the POMDP formulation, the objective is to minimize the expected cost, in our problem, the objective is to reach a desired region of state space with a guaranteed minimum probability of success.

3 Algorithm

This paper extends the approach proposed in [10] to non-Gaussian belief spaces. Our algorithm iteratively creates and executes a series of belief-space plans. A re-planning step is triggered when, during plan execution, the true belief state diverges too far from the nominal trajectory.

3.1 Creating plans

The key to our approach is a mechanism for creating horizon-T belief-space plans that guarantees that new information is incorporated into the belief distribution on each planning cycle. The basic idea is as follows. Given a prior belief state, b_1, define a "hypothesis" state to be at the maximum of the distribution,

$$x^1 = \arg\max_{x \in \mathbb{R}^n} \pi(x; b_1).$$

Then, sample $k-1$ states from the prior distribution,

$$x^i \sim \pi(x; b_1), i \in [2,k], \tag{5}$$

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 (including x^1). We search for a sequence of actions, $\mathbf{u}_{1:T-1} = (u_1, \ldots, u_{T-1})$, 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, consider that if the system starts in state x, and takes a sequence of actions $\mathbf{u}_{1:t-1}$, then the most likely sequence of observations is:

$$\mathbf{h}_t(x, \mathbf{u}_{1:t-1}) = \left(h(x)^T, h(f(x,u_1))^T, h(F_3(x,\mathbf{u}_{1:2}))^T, \ldots, h(F_t(x,\mathbf{u}_{1:t-1}))^T\right)^T,$$

where $F_t(x, \mathbf{u}_{1:t-1})$ denotes the state at time t when the system begins in state x and takes actions, $\mathbf{u}_{1:t-1}$. We are interested in finding a sequence of actions over a

planning horizon T, $\mathbf{u}_{1:T-1}$, that maximizes the squared observation distance,

$$\|\mathbf{h}_T(x^i,\mathbf{u}_{1:T-1}) - \mathbf{h}_T(x^1,\mathbf{u}_{1:T-1})\|^2_{\mathbb{Q}},$$

summed over all $i \in [2,k]$, where $\|a\|_A = \sqrt{a^T A^{-1} a}$ denotes the Mahalanobis distance and $\mathbb{Q} = diag(Q,\ldots,Q)$ denotes a block diagonal matrix of the appropriate size composed of observation covariance matrices. The wider the observation distance, the more accurately Bayes filtering will be able to determine whether or not the true state is near the hypothesis in comparison to the other sampled states.

Notice that the expression for observation distance is only defined with respect to the sampled points. Ideally, we would like a large observation distance between a region of states about the hypothesis state and regions about the other samples. Such a plan would "confirm" or "disprove" regions about the sampled points - not just the zero-measure points themselves. We incorporate this objective to the first order by minimizing the Frobenius norm of the gradient of the measurements,

$$\mathbf{H}_t(x,\mathbf{u}_{1:t-1}) = \frac{\partial \mathbf{h}_t(x,\mathbf{u}_{1:t-1})}{\partial x}.$$

These dual objectives, maximizing measurement distance and minimizing the Frobenius norm of the measurement gradient, can simultaneously be optimized by minimizing the following cost function:

$$J(x^1,\ldots,x^k,u_{1:T-1}) = \frac{1}{k}\sum_{i=2}^{k} e^{-\Phi(x^i,\mathbf{u}_{1:T-1})}, \tag{6}$$

where

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

The weighting matrix (*i.e.* the covariance matrix) in the metric above is defined

$$\begin{aligned}\Gamma(x,\mathbf{u}_{1:T-1}) = 2\mathbb{Q} &+ \mathbf{H}_T(x,\mathbf{u}_{1:T-1})V\mathbf{H}_T(x,\mathbf{u}_{1:T-1})^T \\ &+ \mathbf{H}_T(x^1,\mathbf{u}_{1:T-1})V\mathbf{H}_T(x^1,\mathbf{u}_{1:T-1})^T,\end{aligned} \tag{7}$$

where $V \in \mathbb{R}^{n\times n}$ is a diagonal weighting matrix.

In order to find plans that minimize Equation 6, it is convenient to restate the problem in terms of finding paths through a parameter space. Notice that for any positive semi-definite matrix, A, and vector, x, we have $x^T Ax \geq x^T Ax$, where A is equal to A with all the off-diagonal terms set to zero. Therefore, we have the following lower-bound,

$$\Phi(x^i,\mathbf{u}_{1:t-1}) \geq \sum_{t=1}^{T} \phi(F_t(x^i,\mathbf{u}_{1:t-1}),F_t(x^1,\mathbf{u}_{1:t-1})),$$

where

$$\phi(x,y) = \frac{1}{2}\|h(x)-h(y)\|^2_{\gamma(x,y)},$$

$$\gamma(x,y) = 2Q + H(x)H(x)^T + H(y)H(y)^T,$$

and $H(x) = \partial h(x)/\partial x$. As a result, we can upper-bound the cost, J (Equation 6), by

$$\begin{aligned} J(x^1,\ldots,x^k,\mathbf{u}_{1:T-1}) &\leq \frac{1}{k}\sum_{i=1}^{k} e^{-\sum_{t=1}^{T}\phi(F_t(x^i,\mathbf{u}_{1:t-1}),F_t(x^1,\mathbf{u}_{1:t-1}))} \\ &\leq \frac{1}{k}\sum_{i=1}^{k}\prod_{t=1}^{T} e^{-\phi(F_t(x^i,\mathbf{u}_{1:t-1}),F_t(x^1,\mathbf{u}_{1:t-1}))}. \end{aligned} \tag{8}$$

As a result, the planning problem can be written in terms of finding a path through a parameter space, $(x_t^{1:k}, w_t^{1:k}) \in \mathbb{R}^{2k}$, where x_t^i denotes the state associated with the i^{th} sample at time t and the weight, w_t^i, denotes the "partial cost" associated with sample i. This form of the optimization problem is stated as follows.

Problem 1.

$$\text{Minimize} \quad \frac{1}{k}\sum_{i=1}^{k}\left(w_T^i\right)^2 + \alpha\sum_{t=1}^{T-1} u_t^2 \tag{9}$$

$$\text{subject to} \quad x_{t+1}^i = f(x_t^i, u_t), i \in [1,k] \tag{10}$$

$$w_{t+1}^i = w_t^i e^{-\phi(x_t^i, x_t^1)}, i \in [1,k] \tag{11}$$

$$x_1^i = x^i, w_1^i = 1, i \in [1,k] \tag{12}$$

$$x_T^1 = x_g \tag{13}$$

Problem 1 should be viewed as a planning problem in $(x^{1:k}, w^{1:k}) \in \mathbb{R}^{2k}$ where Equations 12 and 13 set the initial and final value constraints, Equations 10 and 11 define the "belief space dynamics", and Equation 9 defines the cost. Notice that we have incorporated a quadratic cost into the objective in order to cause the system to favor short paths. Problem 1 can be solved using a number of planning techniques such as rapidly exploring random trees [15], differential dynamic programming [16], or sequential quadratic programming [17]. We use sequential quadratic programming to solve the direct transcription of Problem 1. The direct transcription solution will be denoted

$$\mathbf{u}_{1:T-1} = \textsc{DirTran}(x^{1:k}, x_g, T), \tag{14}$$

for the sample set, $x^{1:k}$, goal state constraint, x_g, and time horizon, T. Sometimes, we will call DIRTRAN without the final value goal constraint (Equation 13). This will be written, $\mathbf{u}_{1:T-1} = \textsc{DirTran}(x^{1:k}, T)$. It is important to recognize that the computational complexity of planning depends only on the number of samples used (the size of k in step 3 of Algorithm 1) and not strictly on the dimensionality of the underlying space. This suggests that the algorithm can be efficient in high-dimensional belief spaces. In fact, we report results in [13] from simulations that indicate that the algorithm can work well when very few samples (as few as three or four) are used.

3.2 Re-planning

After creating a plan, our algorithm executes it while tracking belief state using an accurate, high-dimensional filter chosen by the system designer. We denote this Bayesian filter update as

$$b_{t+1} = G(b_t, u_t, z_{t+1}).$$

If the true 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. Steps 2 and 3 sample k states from the distribution with the hypothesis state, $x^1 = \arg\max_{x \in \mathbb{R}^n} \pi(x;b)$, located at the maximum of the prior distribution. The prior likelihood of each sample is required to be greater than a minimum threshold, $1 > \varphi \geq 0$. In step 4, CREATEPLAN creates a horizon-T plan, $\mathbf{u}_{1:T-1}$, that solves Problem 1 and generates a nominal belief-space trajectory, $b_{1:T}$. Steps 6 through 12 execute the plan. Step 8 updates the belief state given the new action and observation using the Bayes filter implementation chosen by the designer. Step 9 breaks plan execution when the actual belief state departs too far from the nominal trajectory, as measured by the KL divergence, $D_1\left[\pi(x;b_{t+1}), \pi(x;b_{t+1})\right] > \theta$. The second condition, $J(x^1,\ldots,x^k,\mathbf{u}_{1:t-1}) < 1-\rho$, guarantees that the *while* loop does not terminate before a (partial) trajectory with cost $J < 1$ executes. The outer *while* loop terminates when there is a greater than ω probability that the true state is located within r of the goal state, $\Theta(b,r,x_g) > \omega$ (Equation 4). In the technical report that expands upon this paper [13], we show that if, for each iteration of the *while* loop, the two conditions in step 9 are met on some time step, $t < T$, then it is possible to guarantee that Algorithm 1 will eventually localize the true state of the system and the *while* loop will terminate.

Input : initial belief state, b, goal state, x_g, planning horizon, T, and belief-state update, G.

```
1  while Θ(b,r,x_g) ≤ ω do
2  |   x^1 = arg max_{x∈R^n} π(x;b);
3  |   ∀i ∈ [2,k], x^i ~ π(x;b) : π(x^i;b) ≥ φ;
4  |   b_{1:T}, u_{1:T-1} = CreatePlan(b, x^1, ..., x^k, x_g, T);
5  |   b_1 = b;
6  |   for t ← 1 to T − 1 do
7  |   |   execute action u_t, perceive observation z_{t+1};
8  |   |   b_{t+1} = G(b_t, u_t, z_{t+1});
9  |   |   if D_1[π(x;b_{t+1}), π(x;b_{t+1})] > θ and J(x^1, ..., x^k, u_{1:t-1}) < 1 − ρ then
10 |   |   |   break
11 |   |   end
12 |   end
13 |   b = b_{t+1};
14 end
```

Algorithm 1: Belief-space re-planning algorithm

Algorithm 2 shows the CREATEPLAN procedure called in step 4 of Algorithm 1. Step 1 calls DIRTRAN parametrized by the final value constraint, x_g. If DIRTRAN

fails to satisfy the goal state constraint, then the entire algorithm terminates in failure. Step 2 creates a nominal belief-space trajectory by integrating the user-specified Bayes filter update over the planned actions, assuming that observations are generated by the hypothesis state. If this nominal trajectory does not terminate in a belief state that is sufficiently confident that the true state is located within r of the hypothesis, then a new plan is created in steps 4 and 5 that is identical to the first except that it does not enforce the goal state constraints. This allows the algorithm to gain information incrementally in situations where a single plan does not lead to a sufficiently "peaked" belief state. When the system gains sufficient information, this branch is no longer executed and instead plans are created that meet the goal state constraint.

Input : initial belief state, b, sample set, $x^1,\ldots,x^k$, goal state, x_g, and time horizon, T.
Output: nominal trajectory, $b_{1:T}$ and $\mathbf{u}_{1:T-1}$

1 $\mathbf{u}_{1:T-1} = \texttt{DirTran}(x^1,\ldots,x^k,x_g,T)$;
2 $b_1 = b$; $\forall t \in [1:T-1],\ b_{t+1} = G(b_t,u_t,h(x_t^1))$;
3 **if** $\Theta(b,r,x_g) \leq \omega$ **then**
4 $\quad \mathbf{u}_{1:T-1} = \texttt{DirTran}(x^1,\ldots,x^k,T)$;
5 $\quad b_1 = b$; $\forall t \in [1:T-1],\ b_{t+1} = G(b_t,u_t,h(x_t^1))$;
6 **end**

Algorithm 2: CREATEPLAN procedure

3.3 Illustration

Figures 1 and 2 illustrate the process of creating and executing a plan in a robot manipulation scenario. 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

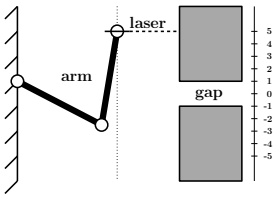

Fig. 1 Localization scenario. The robot must simultaneously localize the gap and move the end-effector in front of the gap.

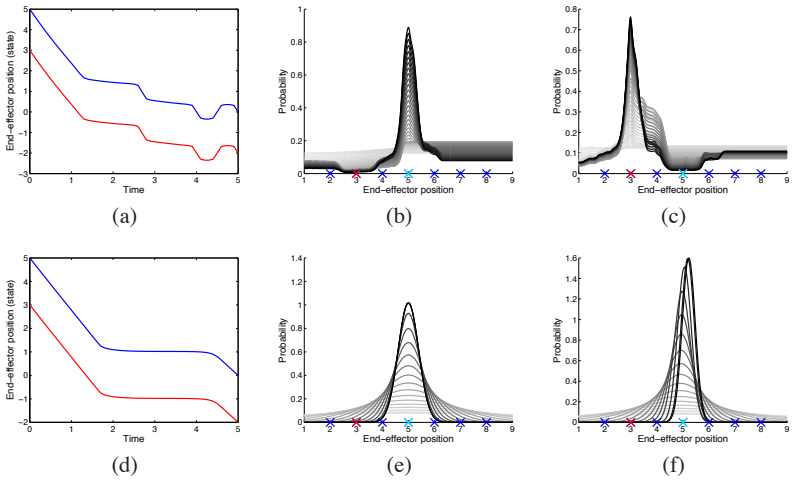

Fig. 2 Illustration of CREATEPLAN. (a) An information-gathering trajectory (state as a function of time) found using direct transcription. Blue denotes the trajectory that would be obtained if the system started in the hypothesis state. Red denotes the trajectory obtained starting in the true state. (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 [10]. (d) The planned trajectory. (e) The corresponding nominal belief-space trajectory. (f) Actual belief-space trajectory.

is illustrated by the vertical number line in Figure 1. The objective is to localize the vertical end-effector with respect to the center of the gap (state) 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 [10]. The problem setup is the same in every way except that cost function optimized in this scenario is:

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

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 Simultaneous localization and grasping

In real-world grasping problems, it is just as important to localize an object to be grasped as it is to plan the reach and grasp motions. We propose an instance of the grasp synthesis problem that we call *simultaneous localization and grasping* (SLAG) where the localization and grasp planning objectives are combined in a single planning problem. In most robot implementations, the robot is able to affect the

type or quality of information that it perceives. For example, many robots have the ability to move objects of interest in the environment or move a camera or LIDAR through the environment. As a result, SLAG becomes an instance of the planning under uncertainty problem. The general SLAG problem is important because good solutions imply an ability to grasp objects robustly even when their position or shape is uncertain.

4.1 Problem setup

(a) (b) (c)

Fig. 3 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 3(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.

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 [18]. 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

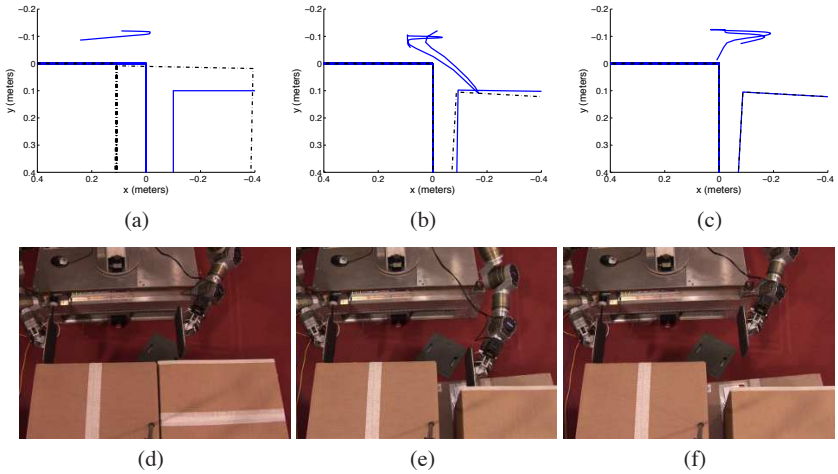

Fig. 4 Example of a box localization task. In (a) and (d), the robot believes the gap between the boxes is large and plans to localize the boxes by scanning this gap. In (b) and (e), the robot has recognized that the boxes abut each other and creates a plan to increase gap width by pushing the right box. In (c) and (f), the robot localizes the boxes by scanning the newly created gap.

(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 3(b)), notice that this is more difficult when the boxes are pressed together (Figure 3(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 [19]. 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 ob-

servation 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 4 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 4(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 4(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 4(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.

4.2 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 5(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 5(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 5(b)).

Figures 5(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

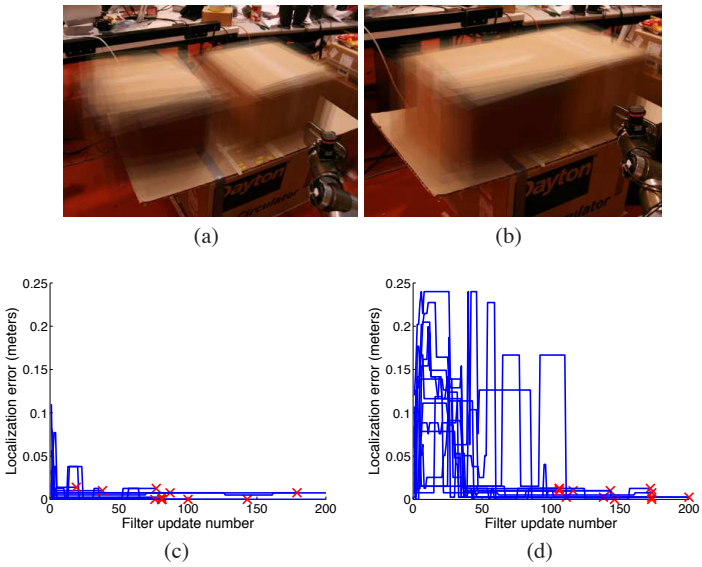

Fig. 5 "Easy" and "hard" experimental contingencies. (a) shows images of the 12 randomly selected "easy" configurations (both box configurations chosen randomly) superimposed on each other. (b) shows images of the 12 randomly selected "hard" configurations (boxes abutting each other). (c) and (d) are 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.

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.

5 Conclusions

Creating robots that can function robustly in unstructured environments is a central objective of robotics. We argue that it is not enough to limit attention to developing better perception algorithms. Robust localization of relevant state in real-world scenarios is not always possible unless the robot is capable of creating and executing information-gathering behaviors. One avenue toward achieving this is the development of algorithms for planning under uncertainty. Our paper proposes a new approach to the planning under uncertainty problem that is capable of reasoning about trajectories through a non-Gaussian belief-space. This is essential because in many robot problems it is not possible to track belief state accurately by projecting onto an assumed density function (Gaussian or otherwise).

The approach is illustrated in the context of a grasping task. We propose a new setting of the grasp synthesis problem that we call simultaneous localization and grasping (SLAG). We test our algorithm using a particular instance of a SLAG problem where a robot must localize two boxes that are placed in front of it in unknown configurations. The algorithm generates information gathering trajectories that move the arm in such a way that a laser scanner mounted on the end-effector is able to localize the two boxes. The algorithm potentially pushes the boxes as necessary in order to gain information. Several interesting questions remain. First, our algorithm focuses primarily on creating information gathering plans. However, this ignores the need for "caution" while gathering the information. One way to incorporate this into the process is to include *chance constraints* into Problem 1 [20]. One approximation that suggests itself is to place constraints on the sample set used for planning. However, since we use a relatively small sample set, this may not be sufficiently conservative. Alternatives should be a subject for future work.

Acknowledgements This work was supported in part by in part by the NSF under Grant No. 0712012, in part by ONR MURI under grant N00014-09-1-1051 and in part by AFOSR grant AOARD- 104135.

References

1. C. Papadimitriou, J. Tsitsiklis, Mathematics of Operations Research **12**(3), 441 (1987)
2. T. Smith, R. Simmons, in *Proc. Uncertainty in Artificial Intelligence* (2005)

3. H. Kurniawati, D. Hsu, W.S. Lee, in *Proceedings of Robotics: Science and Systems (RSS)* (2008)
4. S. Ross, J. Pineau, S. Paquet, B. Chaib-draa, The Journal of Machine Learning Research **32**, 663 (2008)
5. H. Bai, W. Hsu, D. Lee, A. Ngo, in *Workshop on the Algorithmic Foundations of Robotics (WAFR)* (2010)
6. J. Porta, N. Vlassis, M. Spaan, P. Poupart, The Journal of Machine Learning Research **7**, 2329 (2006)
7. J. Van der Berg, P. Abbeel, K. Goldberg, in *Proceedings of Robotics: Science and Systems (RSS)* (2010)
8. S. Prentice, N. Roy, in *12th International Symposium of Robotics Research* (2008)
9. N. Du Toit, J. Burdick, in *IEEE Int'l Conf. on Robotics and Automation (ICRA)* (2010)
10. R. Platt, R. Tedrake, L. Kaelbling, T. Lozano-Perez, in *Proceedings of Robotics: Science and Systems (RSS)* (2010)
11. T. Erez, W. Smart, in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence* (2010)
12. K. Hauser, in *Workshop on the Algorithmic Foundations of Robotics (WAFR)* (2010)
13. R. Platt, L. Kaelbling, T. Lozano-Perez, R. Tedrake, A hypothesis-based algorithm for planning and control in non-gaussian belief spaces. Tech. Rep. CSAIL-TR-2011-039, Massachusetts Institute of Technology (2011)
14. A. Doucet, N. Freitas, N. Gordon (eds.), *Sequential monte carlo methods in practice* (Springer, 2001)
15. S. LaValle, J. Kuffner, International Journal of Robotics Research **20**(5), 378 (2001)
16. D. Jacobson, D. Mayne, *Differential dynamic programming* (Elsevier, 1970)
17. J. Betts, *Practical methods for optimal control using nonlinear programming* (Siam, 2001)
18. M. Fischler, R. Bolles, Communications of the ACM **24**, 381 (1981)
19. L. Sciavicco, B. Siciliano, *Modelling and Control of Robot Manipulators* (Springer, 2000)
20. L. Blackmore, M. Ono, in *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (2009)