Simultaneous localization and motion planning using belief space approaches

Isabella Yu
Massachusetts Institute of Technology
6.4212: Robotic Manipulation
Cambridge, United States
iyu@mit.edu

***Abstract*—Real-world scenarios often present robots with noisy or limited sensor information. In these cases, robots may be required to perform information-gathering actions to maximize sensor information and accomplish their task. One approach to solving these problems is belief space planning, or planning actions according to the probability distribution over the underlying state of the system. This report details the implementation of Platt's paper "Efficient planning in non-Gaussian belief spaces and its application to robot grasping," in which the authors present a belief space planning algorithm with fixed computational complexity that accounts for an arbitrary implementation of a Bayesian filter [2]. The algorithm is applied to a robotic localization problem in simulation, where the robot must simultaneously localize and move its end effector by localizing with a wrist-mounted laser sensor. Although the implementation succeeds at reaching the goal almost all the time, improvements can be made in the laser beam model and the smoothness of trajectories.**

I. Introduction

Robots often use sensors to collect information about their surroundings. However, sensors are not perfect; almost all sensors output noisy observations. Thus, performing actions that maximize the certainty or information content from sensors is crucial for solving common robotics problems such as SLAM and motion planning. In particular, sensor data is an important aspect of robotic motion planning, which involves both localizing the robot as well as reaching a goal state. However, due to limited sensor information, motion planning can be cast as a partially observable Markov Decision Process (POMPD). Obtaining the optimal solution to a POMPD is intractable [3], so approximate algorithms are often practically necessary. One way of solving POMPD's is to use belief space planning, a class of planning algorithms that direct the robot to take information-gathering actions to minimize uncertainty over the robot's state. Thus, belief space planning can maximize the information gained out of noisy sensor data. Such information-gathering actions can be essential for planning in scenarios with dynamic environments, from robotic picking to exploration of caves.

II. Related work

Traditional approaches to planning include Extended Kalman Filter (EKF)-based methods, which are commonly used for nonlinear state estimation. However, these methods are not optimal if the observation model is nonlinear because of the filter's inherent linearization. In addition, process and measurement noise are assumed to be Gaussian, leading to underperformance in non-Gaussian belief spaces [1]. For example, a robot's estimated position in a 2D environment with obstacles may not be represented as Gaussian or a sum of Gaussians because of the irregularity of the obstacles.

Similarly, belief space planning algorithms prior to Platt's, such as [3], assume Bayes filtering will be performed using a Gaussian density function. Furthermore, Platt states that "extending an approach like [3]'s to non-Gaussian distributions quickly results in an intractable planning problem because of the high dimensionality of typical non-Gaussian parametrizations."

Platt's paper proposes an approach to planning in high-dimensional belief spaces that tracks belief state using a Bayesian filter, but creates plans using a set number of samples from the belief space [2]. Thus, the algorithm has a fixed computational complexity dependent on only the sample size, not on the dimension of the underlying belief space representation.

III. Methods

A. Problem statement

The objective of the problem presented in Platt's paper is to estimate the state of a robotic system while reaching the goal state. The robot is equipped with a noisy laser sensor and is assumed to have no process noise.

Thus, the problem can be modeled as a discrete time system. The state of the system evolves via the deterministic dynamics function

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

where $x_t, x_{t+1} \in \mathbb{R}$ are the states at times t and $t+1$, respectively, and $u_t \in \mathbb{R}$ is the input to the system.

The laser observations at time t, z_t can be modeled as a noisy function of the state,

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

where $h(x_t)$ gives the true depth reading at state x_t, and $v_t \sim \text{Normal}(0, Q)$ is zero-mean Gaussian process noise with variance Q.

The robot starts at $t = 0$ with no prior over its state, i.e. with a uniform distribution over the possible states. By taking actions and observing the environment at each timestep, the robot can change this distribution, or the belief space b, to concentrate at the true state via a Bayesian filtering algorithm like a histogram filter.

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

$$\Theta(r, x_g, b) = \int_{x \in B_n(r)} \pi(x + x_g; b) > \omega \tag{3}$$

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.

B. Algorithm

The overarching goal of the algorithm is to maximize the sensor information gathered at each time step of the robot's trajectory in a fixed horizon T. To accomplish this, the algorithm maximizes the difference between the observations (generated by a sequence of velocity commands $\mathbf{u} = (u_1, ..., u_T)$) starting from the most likely state ($\bar{x} = \text{argmax}(\pi(x; b_t))$ to those of $k-1$ randomly sampled other states ($x^i \sim (\pi(x; b_t), i \in \{1, ..., k\})$. Calculating $\Theta(r, x_g, b)$ at the end of the trajecotry either proves that $\bar{x}$ is the true state or disproves it, and in the latter case, the algorithm selects a new $\bar{x}$ and $k-1$ sampled states to compare, ultimately converging to the true state.

The full mathematical details are omitted for brevity and can be found in Platt's paper, but this essentially amounts to solving the optimization problem below:

Problem 1.

$$\text{Minimize } \frac{1}{k} \sum_{i=1}^{k} (w_T^i)^2 + \alpha \sum_{t=1}^{T-1} u_t^2 \tag{4}$$

$$\text{subject to } x_{t+1}^i = f(x_t^i, u_t), i \in \{1, ..., k\} \tag{5}$$

$$\text{subject to } w_{t+1}^i = w_t^i e^{-\phi(x_t^i, x_t^i)} i \in \{1, ..., k\} \tag{6}$$

$$\text{subject to } x_1^i = x^i, w_1^i = 1, i \in \{1, ..., k\} \tag{7}$$

$$\text{subject to } x_T^i = x_g \tag{8}$$

where w_t^i defines the partial cost of sample i, i.e. how far its observation is from $\bar{x}$'s (w_t^i includes terms that minimize the Frobenius norm of $\frac{\partial h(x^i)}{x^i}$, the derivative of sample i's observation with respect to x^i, to accomplish this). The quadratic cost on u_t leads the system to favor shorter paths/smaller velocities. One can obtain the direct transcription solution, the desired trajectories $\mathbf{u}_{1:T}$, by solving the optimization problem via SQP. Platt denotes this as $\mathbf{u}_{1:T} = \text{DIRTRAN}(x^{1:k}, x_g, T)$.

The CREATEPLAN algorithm (Algorithm 2) generates trajectories with or without the goal constraint. It first attemtps to call DirTran with the goal constraint. If the generated trajectory results in too uncertain of a state estimate (line 3), DIRTRAN is called without the goal state to allow for a less constrained exploration of the environment to encourage performing information-gathering actions. In this way, the robot is always gaining information, and its estimated state eventually converges to its true state.

After generating the trajectories with CREATEPLAN, the robot performs each action in $\mathbf{u}_{1:T}$ sequentially while updating its belief state at each timestep, b_t, using a Bayesian filter $G(u_t, x_t, h(x_t)$–a histogram filter in this implementation, as shown in Algorithm 1. The robot continues generating trajectories until $\Theta(r, x_g, b) = \int_{x \in B_n(r)} \pi(x + x_g; b) > \omega$, i.e. the estimated state is within some radius of the true state with probability ω. However, replanning occurs if the KL divergence of the belief state is too high, i.e. the belief state deviates too much from the trajectory (line 9). The algorithms are outlined:

Input : initial belief state, b, goal state, x_g, planning horizon, T, and belief-state update, G.
1 **while** $\Theta(b, r, x_g) \leq \omega$ **do**
2 $x^1 = \arg\max_{x \in \mathbb{R}^n} \pi(x; b)$;
3 $\forall i \in [2, k], x^i \sim \pi(x; b) : \pi(x^i; b) \geq \varphi$;
4 $b_{1:T}, \mathbf{u}_{1:T-1} =$ `CreatePlan(`$b, x^1, \ldots, x^k, x_g, T$`)`;
5 $b_1 = b$;
6 **for** $t \leftarrow 1$ **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** $D_1\left[\pi(x; b_{t+1}), \pi(x; b_{t+1})\right] > \theta$ **and** $J(x^1, \ldots, x^k, \mathbf{u}_{1:t-1}) < 1 - \rho$ **then**
10 break
11 **end**
12 **end**
13 $b = b_{t+1}$;
14 **end**

Algorithm 1: Belief-space re-planning algorithm

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 $\mathbf{u}_{1:T-1} =$ `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 $\mathbf{u}_{1:T-1} =$ `DirTran(`$x^1, \ldots, x^k, T$`)`;
5 $b_1 = b; \forall t \in [1 : T-1],\ b_{t+1} = G(b_t, u_t, h(x_t^1))$;
6 **end**

Algorithm 2: CREATEPLAN procedure

C. Experimental setup

This section delves into the implementation details of the algorithms above that Platt's paper did not go in depth into. In particular, this section will cover how the algorithm was modified to fit Drake's simulation and optimization APIs, as well as the implementation of the histogram filter.

Manipulation setup: Fig. 1 shows a screenshot of the simulation setup in Drake's MeshCat visualizer. The robot used is a 7-link iiwa arm whose base is welded to the world origin. Its initial position is the one shown in the figure. The end effector is a WSG 50 gripper. An Intel RealSense D415 depth camera is attached to the top of the end effector, and it is modified to act as a noisy laser beam; it gives only the center pixel of the depth image with zero-mean Gaussian noise with standard deviation 0.025. Its The end effector faces two evenly placed boxes, whose poses are perfectly known. The end effector is constrained to move in the y-axis from its

inital position, and not beyond the outer ends of the boxes. Notice that the gap between them allows the laser to "see" into the wall behind the boxes, giving the histogram filter a key feature for localization. The system uses differential inverse kinematics to execute actions. The state x is the robot's position along the y-axis, $h(x)$ is the laser sensor reading at state x, and $f(x, u) = x_t + u_t * \delta$ where δ is the simulation timestep. Ultimately, the algorithm does not run in Drake's simulation yet, and the immediate next step would be to run the trajectories generated by the belief space planning algorithm on the iiwa by sending the y positions at each time step to the input of the differential IK controller.

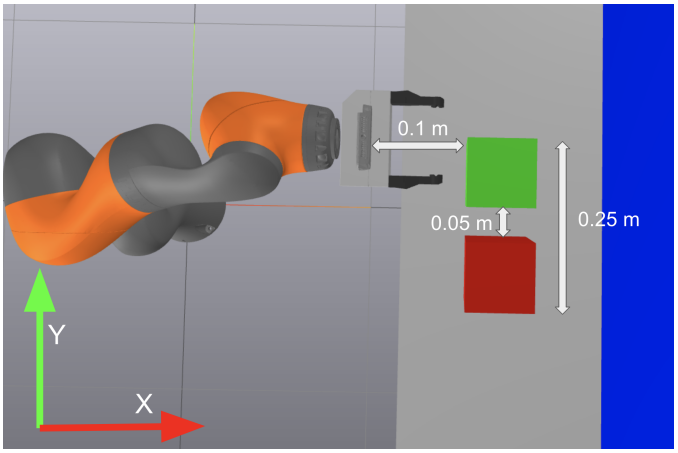


Fig. 1: Simulation setup in Drake's Meshcat, labeled with axes and object positions.

Using Drake's `MathematicalProgram` *and symbolic differentiation*: The implementation of DirTran using Drake's `MathematicalProgram` was straightforward with some exceptions. Notably, since Problem 1 requires the calculation of the derivative of the observation $h(x)$ with respect to state x, and x is a decision variable in the `MathematicalProgram`, $h(x)$ must be a continuous symbolic function. However, in the manipulation setup above, $h(x)$ is discontinuous because of the sharp edges of the boxes, as shown in Fig. 2. Because of this, it was necessary to approximate $h(x)$ as

$$\bar{h}(x) = \frac{2A}{\pi}\arctan\left(\frac{\sin(2\pi f x + \frac{\pi}{4A})}{d}\right) + 0.15 \qquad (9)$$

where $A = 0.1$ (the amplitude), $d = 0.001$ (the steepness of the wave), and $f = 5$ (the frequency). This makes it so that Drake can calculate $\frac{\partial \bar{h}(x^i)}{x^i}$ via autodifferentiation while solving the problem.

The parameters that yielded the least solve failures from the DirTran were $k = 15$, $T = 10$, and $\alpha = 0.0085$.

In addition, the decision variables $u_{1:T}$ are initialized with random values, which was necessary for the solver to generate a solution.

Histogram filter implementation: The system uses a histogram filter to update its state according to its dynamics and observations. The histogram filter algorithm is outlined to the right.

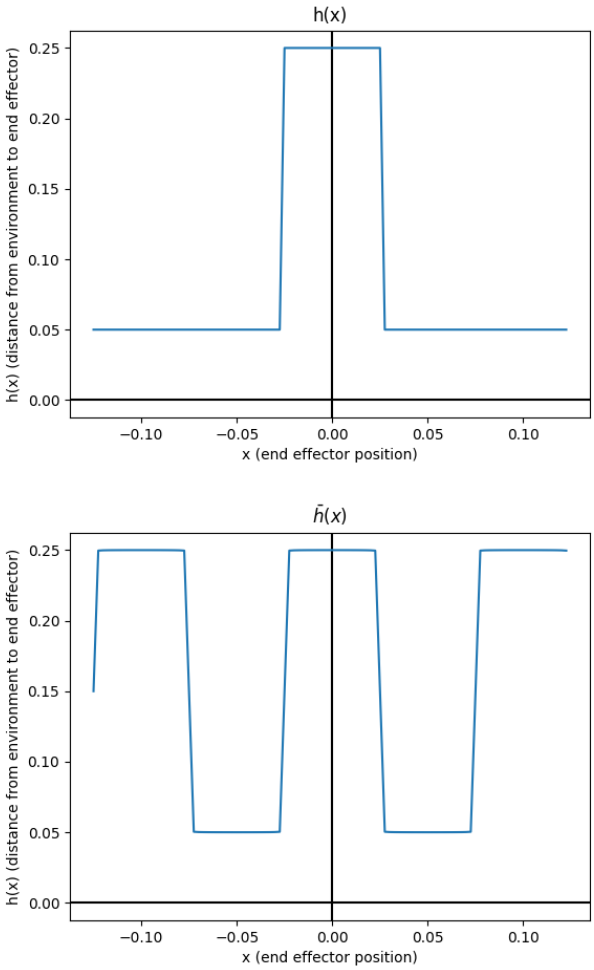


Fig. 2: $h(x)$, the true observation function, and $\bar{h}(x)$, the continuous approximation. Note that the robot is constrained to move between the outer edges of the boxes in the y-axis, in this case between -0.125 m and 0.125 m, so values of $\frac{\partial \bar{h}(x^i)}{x^i}$ outside the range are set to 0.

The first line of the for loop updates the belief state based on the robot's action u_t. In this system with no process noise, this is simply $p(X_t = x_k | u_t, X_{t-1} = x_i) p_{i,t}$, where x_i is the previous step. The second line updates the belief state based on sensor observations. The sensor observation contains Gaussian noise, Normal(0, σ), so $p(z_t | X_t = x_k) =$ NormalCDF(x_{min}, z_t, σ) − NormalCDF(x_{max}, z_t, σ), where x_{min} and x_{max} are the upper and lower bin boundaries of the bin that x is in.

IV. Results

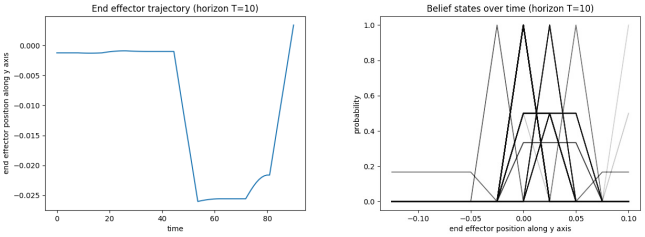
Representative examples of the results of the planning algorithm are shown in Figs. 3-4.

Fig. 3a shows a trajectory generated by the planner with a starting state of $x = -0.0125$. Note that the time axis on the graph represents the total time elapsed, not the planning horizon, as the planner is called multiple times in the while

Algorithm 1: Histogram Filter

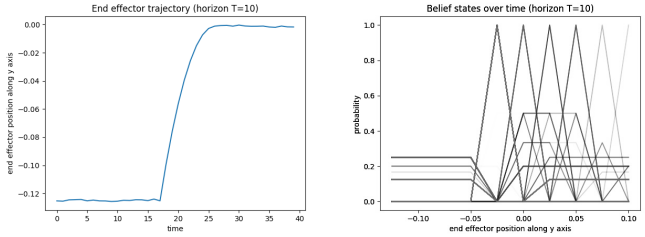
Result: $p_{k,t}$, the belief state over k state bins at time t
initialization;
for *all* k **do**
  $\bar{p}_{k,t} = \sum_i p(X_t = x_k | u_t, X_{t-1} = x_i) p_{i,t}$;
  $p_{k,t} = \eta p(z_t | X_t = x_k) \bar{p}_{k,t}$;
end

loop. Fig. 3b shows the change in probability distributions over the course of the algorithm, with transparent lines representing older distributions. Fig. 4a and 4b show the same plots but for a starting state of $x = -0.125$–the edge of the box.



(a) (b)

Fig. 3: 3a) an example of an information gathering trajectory starting from $x = -0.0125$. 3b) belief state over time. Notice that more recent belief states spike at around $x = 0.0$, the goal state, indicating certainty of reaching the goal.



(a) (b)

Fig. 4: 4a) an example of an information gathering trajectory starting from $x = -0.125$. 4b) belief state over time. More recent belief states spike at around $x = 0.0$, the goal state, but are not as sharp as in the previous figure.

It is unclear why the optimization outputs near-zero velocities in the first few iterations of the algorithm, but it does seem to be a pattern. However, the latter parts of trajectories are representative of information gathering actions. In Fig 3a., even though the robot starts near the goal, the hypothesis state $\bar{x}$ sampled is likely not the goal state, since few actions were taken in the first iterations, leading to repetitive observations. Thus, the robot generates a trajectory to gather information, leading it to observe different features of the environment and thus localize itself. After localizing, it returns to the gap.

V. Discussion

The robot successfully reached the goal state, i.e. the middle of the gap, about 96% of the time (the execution was a success if optimization did not fail and x_T was within 2% of the goal state). Most of the successes were concentrated near the goal state (see Fig. 5). This indicates that although the algorithm was able to reach the goal location most of the time, it was more reliable when it started close to the goal state. One reason not reaching the goal state may be because the histogram filter is too "certain" of its location when it is at the border. The sensor update rule $p_{k,t} = \eta p(z_t | X_t = x_k) \bar{p}_{k,t}$ in the histogram filter returns a small value of $p_{k,t}$ if $\bar{p}_{k,t}$, the previous state, is small, even when $p(z_t | X_t = x_k)$, or the certainty of an observation, is large; when the sensor starts out observing the box instead of the gap, the histogram filter indicates that the probability that the robot is in the gap is near zero. The robot must stay in the gap for an extended period of time This can be alleviated by increasing the sensor's noise variance, but the tradeoff is that the algorithm fails to localize due to high noise. Another way to add noise to decrease the histogram's certainty would be to add process noise. This would both not affect the sensor noise and realistically model how a physical robot moves.

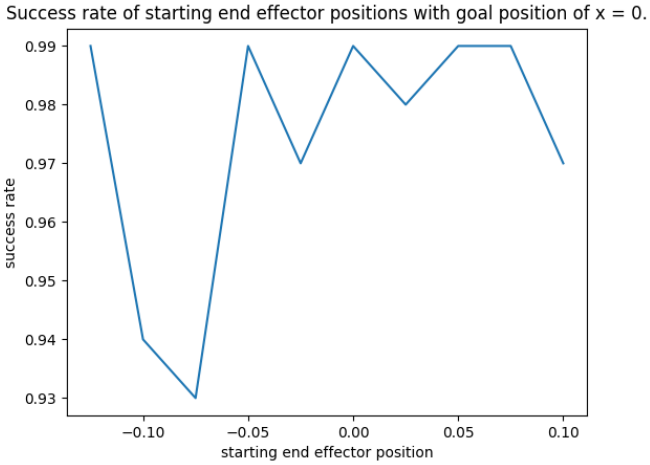


Fig. 5: The rate of success starting from each possible state.

In addition, even though the optimization cost penalized large velocity commands, Figs. 3-4 show that the robot is still incentivized to move large distances in one timestep to get to the goal. This would be undesireable on a real robot due to the large motor torques that sudden accelerations can induce. I tried diminishing this effect by setting bounding constraints $u_{1:T}$, but that caused the optimization to always fail. This may be due to a bug in the optimization code. It may also be caused by poor initialization, as random initialization may not lead to a feasible solution at times.

Though runtime was not explicitly analyzed, qualitatively, the algorithm ran almost instantaneously unless an optimization failure occurred. Increasing the sample size from the

default $k = 15$ did not seem to improve performance, but slowed down the optimizer due to the added number of decision variables.

VI. Conclusion

In this project, I implemented Platt's sampling-based belief space planning planning algorithm to simulaneously localize and move the end effector of a robot with a laser sensor. Overall, the algorithm successfully generated information gathering trajectories. However, improvements to lessen the optimization failure rate and decrease the maximum end effector velocities delivered by the optimization problem would be crucial to allow the system to run reliably on a real robot.

To make the algorithm more applicable to real-world scenarios, I could model the process dynamics with noise, which would affect the probability distributions over the robot's state calculated by the histogram filter. In addition, I modeled the laser sensor with Gaussian noise, but physical laser sensors suffer from occlusions (short returns), missed detections (returning the max depth), and random measurements as well. Thus, a more realistic model of the laser sensor would be Drake's BeamModel, which takes into account these additional problems.

Comparing this planning algorithm with others such as EKF-based approaches could lead to insights on how the algorithm's behavior differs from other approaches, as well as establish a baseline for localization performance.

To expand upon my work, the next step would be to simulate the generated trajectories in Drake. After this, I could implement the simultaneous localization and grapsing (SLAG) algorithm discussed in Platt's paper. The SLAG algorithm localizes the 6-D position of two boxes with a end effector-mounted laser sensor as opposed to the 1-D position of the end effector as implemented in this paper, and is an illustrative example of belief space planning for industrial applications.

References

[1] Courses, E.; Surveys, T. (2006). Sigma-Point Filters: An Overview with Applications to Integrated Navigation and Vision Assisted Control. Nonlinear Statistical Signal Processing Workshop, 2006 IEEE. pp. 201–202

[2] Platt, R., Kaelbling, L., Lozano-Perez, T., & Tedrake, R. (2017). Efficient planning in non-gaussian belief spaces and its application to robot grasping. In Robotics Research (pp. 253-269). Springer, Cham.

[3] Platt Jr, R., Tedrake, R., Kaelbling, L., & Lozano-Perez, T. (2010). Belief space planning assuming maximum likelihood observations.

[4] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. 2005. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press.