16-748 Final Project Report
Implementation for Planning in Generalized Belief Space

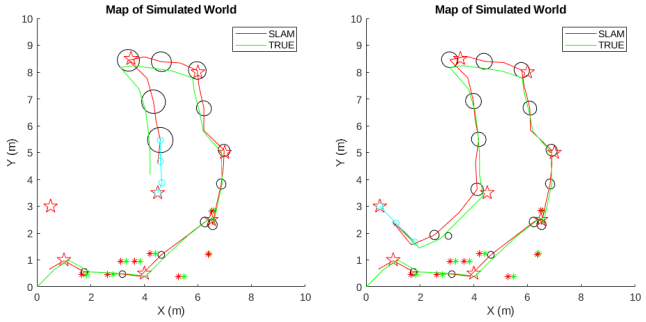
Haowen Shi[1], Hans Kumar[1]

***Abstract*— Belief Space Planning has been of interest in the robotics community because of its ability to take into account robot state uncertainty during planning. [1] presented a method for planning in the belief space given a known map by assuming maximum likelihood observations. [2] extended this work to support Active SLAM tasks by introducing the concept of the generalize belief space (GBS), which contains a joint belief state consisting of the robot state and the world state. By planning in the GBS, no prior knowledge of the map is required to perform planning. This is useful for autonomous exploration tasks such as confined space inspection, etc. where the robot has to plan uncertainty-reducing actions to improve localization accuracy while fulfilling planning objectives. In this project we implement [2] from scratch and evaluate its performance against a naive LQR controller on several different simulated maps.**

I. Introduction

Planning under uncertainty is a Partially Observable Markov Decision Process (POMDP) as formulated in [3]. The uncertainty aspect of this planning problem sets it apart from regular Markov Decision Processes (MDP) because the robot cannot fully observe its own state and the world state. [1], [4] proposed one way of solving optimal controls under uncertainty by maintaining a probabilistic belief distribution of the states, and planning actions in the belief space instead of the state space. This enables us to incorporate state estimation uncertainty (covariance of state belief distribution) as part of the planning objective and plan uncertainty-reducing actions to help cope with sparse features in the environment. Instead of planning straight for the goal pose, the planner *actively* reduces the estimation error. The planner might do this by steering the robot towards previously observed visual landmarks when state uncertainty becomes too big (shown in Fig. 1). Reduced estimation error means better trajectory tracking, and hence better mapping quality. The belief distribution formulated in [1] only contains the robot pose and it assumes a known environment for predicting maximum likelihood observations. This is unrealistic in confined space inspection tasks where the environment is usually unknown. [2] proposed to remove this assumption by maintaining a joint state of both the robot state and the world state. The world state is to be estimated together with the robot state (pose), so there is no need to have any prior knowledge of the environment.

In this work we follow the *Generalized Belief Space* (GBS) planning framework proposed by [2] and implement a planner from scratch because there are no publicly available implementation of this paper available at the time of this writing. Fig. 1 shows the big picture result of our implementation: our simulated robot moves out of the way to re-observe previous landmarks after a build up of uncertainty. We evaluate our planner by comparing its performance against that of a simple LQR planner which only penalizes distance to goal and control effort, using several metrics including: 1) evolution of uncertainty, 2) trajectory estimate error, 3) total control effort expanded, and 4) total planning time.



Before re-observation After re-observation

Fig. 1: Instead of directly reaching the goal, our planner plans uncertainty-reducing action that will lead to re-observation of landmarks and reduce state estimation uncertainty (shown as black ellipses, larger area means larger uncertainty).

II. Methodology

This POMDP autonomous navigation problem can be broken down into two parts - perception and planning. System state belief is updated through observations and system dynamics in the perception component. We use the Square Root SAM to update the system belief state as proposed by [5] because this smoothing based approach can optimize the entire state trajectory efficiently by exploiting sparsity in the information matrix. To implement the planning framework proposed in [2], we first need to formulate the joint belief distribution and its propagation given some measurements and actions. Then, the optimal control action is obtained using Model Predictive Control (MPC). To solve for the optimal control that produces the best objective value, we implemented the dual-layer iterative optimization following [2]. We design the cost function to include a set of concurrent planning objectives including 1) goal-reaching, 2) uncertainty-minimizing, and 3) control-effort-minimizing. Finally, we discuss the choice of weight matrices

This work is done as our final class project for 16-748 Underactuated Robotics offered in fall 2020.

[1]The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

that regulates the balance between conflicting objectives.

A. Square Root SAM

Square Root SAM solves the full Simultaneous Localization And Mapping (SLAM) problem in a smoothing fashion. Compared to the filtering approach used in EKF SLAM methods, Square Root SAM optimizes the entire state trajectory instead of just the latest state. This is useful because it avoids the linearization error that builds up due to a non-linear measurement model. We use the factor graph based GTSAM library implemented by [6] and use the Gaussian state estimate (mean and covariance) as our belief model. The belief is updated by adding measurements and actions as constraints in the factor graph and optimizing the graph.

B. Generalized Belief Space

The generalized belief space is the span of joint states which include both the robot state and the world state. The robot state contains pose estimate of the robot, and the world state contains the pose estimates of the landmarks. The joint state is defined as: $X_k \doteq \{x_0, x_1, \ldots, x_k, W_k\}$ at the k'th planning step. The planning objective function defined at time step $k+l$ is a function of the generalized belief at step $k+l$, where the belief is modelled as:

$$gb(X_{k+l}) \doteq p(X_{k+l}|Z_k, U_{k-1}, Z_{k+1:k+l}, u_{k:k+l-1}) \quad (1)$$

Here X_{k+l} is the joint state including robot poses and landmark locations at time $k+l$. Z_k is all prior observations until time k. U_{k-1} is the history of actions. $Z_{k+1:k+l}$ and $u_{k:j+l-1}$ are the future observations and actions respectively. The belief is then modelled using a Gaussian:

$$gb(X_{k+1}) \sim N(X^*_{k+l}, I_{k+l}) \quad (2)$$

Where X^*_{k+l} is the maximum a posteriori (MAP) estimate:

$$\begin{aligned} X^*_{k+l} &= \arg\max_{X_{k+l}} gb(X_{k+l}) \\ &= \arg\min_{X_{k+l}} -log(gb(X_{k+l})) \end{aligned} \quad (3)$$

And I_{k+l} is the information matrix which is the inverse of the covariance at step $k+l$.

We limit the scope of this work to a 2D environment with sparse landmarks, with the robot state being $[x, y]$ and the control being $[dx/dt, dy/dt]$. The solution to the MAP estimate is discussed in section II-D.

C. Model Predictive Control

In this section we consider the optimal control problem at each planning time k. Model Predictive Control (MPC) is used here to make planning in the large, continuous belief space computationally viable. The optimal control sequence $u_{k:k+L-1}$ is picked over L planning horizons to minimize the objective function $J_k(u_{k:k+L-1})$. Here to formulate the optimization problem we consider generic run-time and terminal cost functions c_l and c_L. For notation convenience,

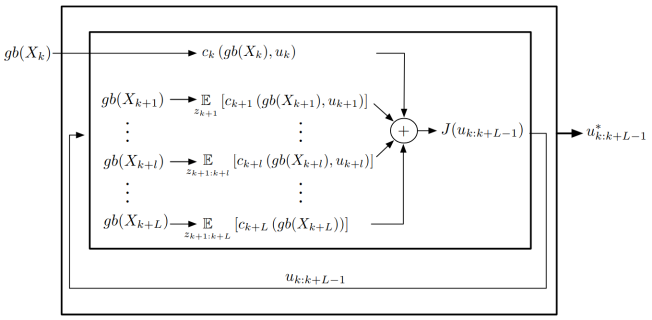


Fig. 2: Dual layer iterative optimization to solve the optimal control problem in generalized belief space. Source: Taken from [2].

c_l is the run-time cost at step $k+l$, defined in this general form:

$$c_l(gb(X_{k+l}), u_{k+l}) \quad (4)$$

In [2], the authors do not assume maximum likelihood observations, and maintain future observations as random variables. This fact means that in the objective function we will not have future observations available. Instead, expectation is taken over the cost over the random variable $Z_{k+1:k+l}$ to account for the uncertainty in future observations. The objective function at step k is then defined as:

$$\begin{aligned} J_k(u_{k:k+L-1}) \doteq \sum_{l=0}^{L-1} & \mathop{\mathbb{E}}_{Z_{k+1:k+l}} [c_l(gb(X_{k+l}), u_{k+l})] \\ & + \mathop{\mathbb{E}}_{Z_{k+1:k+L}} [c_L(gb(X_{k+l}))] \end{aligned} \quad (5)$$

The corresponding optimal control policy is:

$$u^*_{k:k+L-1} = \arg\min_{u_{k:k+L-1}} J_k(u_{k:k+L-1}) \quad (6)$$

Now we discuss the details of this optimization problem in the following section.

D. Iterative Optimization

As shown in Fig. 2, there are two layers of inference in the optimization framework proposed by [2]. The outer layer is a non-linear optimization of the objective function defined in (5) over control. The inner layer is propagating the generalized belief distribution over the planning horizon given current control values in each outer layer iteration. The reason belief propagation is also an optimization problem is because the belief distribution at each future planning step is estimated using MAP as formulated in (3).

We implemented our outer layer non-linear optimization using stochastic gradient descent to generate the optimal control sequence. For the inner layer, we took the observation made in [2] that one iteration of Gauss-Newton sufficiently captures the effect of future observation and action over the belief state, and implemented accordingly.

In the inner layer, we perform one iteration of Gauss-Newton optimization around the linearization points $\bar{X}_{k+l}$ obtained by predicting future states using the motion model

with current control values $u_{k:k+l-1}$ for each look-ahead time step:

$$\bar{X}_{k+l} = \begin{pmatrix} \bar{x}_k \\ \bar{x}_{k+1} \\ \vdots \\ \bar{x}_{k+l} \end{pmatrix} \doteq \begin{pmatrix} \hat{X}_k \\ f(\bar{x}_k, u_k) \\ \vdots \\ f(\bar{x}_{k+l-1}, u_{k+l-1}) \end{pmatrix} \tag{7}$$

Through the mathematical derivations provided in [7], we can solve the MAP estimate problem in (3) using linear least squares:

$$\begin{aligned} X^*_{k+l} = \arg\min_{X_{k+l}} \|X_k - X^*_k\|^2_{I_k} \\ + \sum_{i=1}^{l} \{\|x_{k+i} - f(x_{x+i-1}, u_{k+i-1})\|^2_{\Omega_w} \\ + \|z_{k+i} - h(X^O_{k+i})\|^2_{\Omega_v}\} \end{aligned} \tag{8}$$

Where $\|x-u\|^2_\Omega = (x-u)^\top \Omega (x-u)$ is the Mahalanobis norm with information matrix Ω as the scaling factor. In the one Gauss-Newton iteration we make, the update vector ΔX_k and propagated information matrix I_{k+l} can be calculated by first re-writing function being minimized in (8) in the following form:

$$\|\mathcal{A}_{k+l}(u_{k:k+l-1})\Delta X_{k+l} - b_{k+l}(u_{k:k+l-1}, z_{k+1:k+l})\|^2 \tag{9}$$

Where $\mathcal{A}$ and b are of the following form:

$$\mathcal{A}_{k+l} \doteq \begin{bmatrix} \begin{bmatrix} I^{1/2}_k, 0 \end{bmatrix} \\ \mathcal{F}_{k+l} \\ \mathcal{H}_{k+l} \end{bmatrix}, b = \begin{pmatrix} 0 \\ \Omega^{1/2}_w b^f_{k+l} \\ \Omega^{1/2}_w b^h_{k+l} \end{pmatrix} \tag{10}$$

The information matrix at step $k+l$ can then be obtained using:

$$I_{k+l} \doteq \mathcal{A}^\top_{k+l} \mathcal{A}_{k+l} \tag{11}$$

Here we care the most about the information matrix at planning step $k+l$ because it represents the evolution of state estimate uncertainty and is a great term to be incorporated in the final objective function. We omit the discussion of other details in the inner layer Gauss-Newton update as the derivation is long. More details can be found in [7]. In our implementation we construct the $\mathcal{A}$ as defined in (10) by stacking measurement Jacobians $\mathcal{H}$ and motion Jacobians $\mathcal{F}$ in appropriate locations. We follow the stacking order described in [5] and [8] to make the correct associations between state variables for the least square state estimate solution.

E. Objective Function Design

In this work we experimented with three planning objectives: 1) goal-reaching, 2) uncertainty-minimizing, and 3) control-effort-minimizing. The run-time and terminal cost function we use is defined as follows:

$$\begin{aligned} c_l(gb(X_{k+l}), u_{k+l}) = \|EX^*_{k+l} - X^G\|_{M_x} \\ + tr(M_\Sigma I^{-1}_{k+l} M^\top_\Sigma) \\ + \|\zeta(u_{k+l})\|_{M_u} \end{aligned} \tag{12}$$

$$\begin{aligned} c_L(gb(X_{k+L})) = \|EX^*_{k+L} - X^G\|_{M_x} \\ + tr(M_\Sigma I^{-1}_{k+L} M^\top_\Sigma) \end{aligned} \tag{13}$$

Where $\zeta(u)$ is the function defining control effort given control values. In our case $\zeta(u) = \|u\|$. Substituting (12) and (13) to the objective function (5), we get the final cost function after taking the expectation over future observation random variables:

$$\begin{aligned} J_k(u_{k:k+L-1}) = \sum_{l=0}^{L-1} \|\zeta(u_{k+l})\|_{M_u} \\ + \sum_{l=0}^{L} tr(M_\Sigma I^{-1}_{k+L} M^\top_\Sigma) \\ + \sum_{l=0}^{L} \{\|EX^*_{k+L} - X^G\|_{M_x} \\ + tr(Q_{k+l}(\mathcal{H}_{k+l} \bar{I}^{-1}_{k+l} \mathcal{H}^\top_{k+l} + \Omega^{-1}_v))\} \end{aligned} \tag{14}$$

Where $Q_{k+l} = (EI^{-1}_{k+l} \mathcal{H}^\top_{k+l} \Omega_v)^\top M_x (EI^{-1}_{k+l} \mathcal{H}^\top_{k+l} \Omega_v)$. Here E is the selection matrix that chooses which part of the joint state should be considered in goal-reaching behavior. In our case this E matrix selects the robot pose part of the joint state only. Here we can see a fourth term appearing which is the direct result of not assuming maximum likelihood assumption. The goal achievement behavior now also incorporates uncertainty ($\bar{I}$ in the term) due to stochasticity in future measurements.

The final objective function evaluated in (14) given our run-time and terminal cost is used to compute the cost in the inner layer with propagated generalized belief. The objective value at each planning step in the MPC planning horizon is added to form the final cost which is fed to the outer layer to perform inference over control. The inner rectangle in Fig. 2 shows that the objectives at each planning step are summed together and passed to the outer layer.

III. Implementation

We implemented all of our code in Matlab, and we make it publically available here [1]. Fig. 3 provides a block diagram of how the major parts of our code work together. We split the implementation of this paper into two major subsystems: planning and estimation.

We use GTSAM to do fast incremental estimation based on all of our prior measurements and controls. Then we extract a gaussian belief state by marginalizing out previous states and creating an estimate for the mean and covariance of the current state of the robot and the world. The planning module then takes this belief state and uses a dual layer optimization to output the optimal controls for our given cost functions. The outer layer of this optimization is done by using Matlab's in-built `fmincon` function. In the outer layer, we use the optimal control values from the previous iteration to seed the optimizer's initial guess. Seeding our optimizer

[1] `https://github.com/kumarhans/GBS`

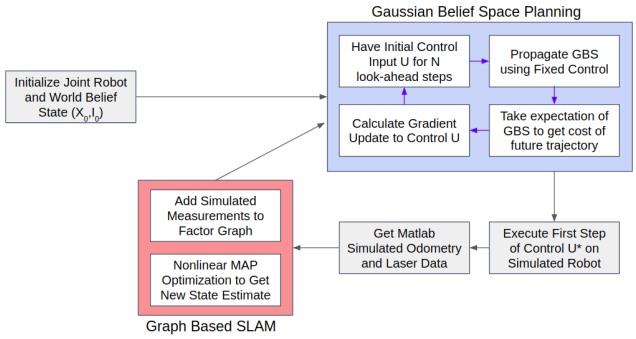


Fig. 3: High level implementation overview. We implement the localization component using graph-based SLAM method based on GTSAM infrastructure. The planning component follows methods described in [2].

with a good initial guess, made the final trajectory smoother and saved a lot of computation time. After optimizing our controls, we execute only the first step of our optimal control trajectory in an MPC style. We repeat this process of estimation and planning until our robot reaches its final destination.

A. Choice of Weight Matrices

There are three tunable weight matrices in the final objective function (14): M_u, M_x and M_Σ. The choice of M_u is straightforward as it penalizes control effort proportionally. However, M_x and M_Σ affect two conflicting control objectives. M_x affects the importance of goal-reaching behavior and M_Σ affects the importance of uncertainty-reducing behavior. [7] discussed the choice of these matrices in detail and provided some guidance on how to balance conflicting control objectives. In our work we hand-tuned the weights until we got a satisfactory balance without too much oscillatory behavior. We acknowledge the inadequacy in our approach and evaluation of parameter tuning, and we leave it as future work.

B. Weighting Future Landmark Observations

When constructing the $\mathcal{H}$ matrix in Eq. 10 we must simulate future observations based on our predicted future state. To decide which landmarks are re-observed in the future plan, one could use a hard sensing range cut-off. The problem with such an observation model is that during the outer layer optimization, unless the perturbed controls resulted in the robot entering the sensing range, there wouldn't be any gradient available. To combat this issue we decided to consider landmarks that were far away in planning, but weight them according to their distance from the robot.

$$\mathcal{H}_{weighted} = \mathcal{H} \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_r \end{bmatrix} \tag{15}$$

This would mean that we would be more certain about measurements from landmarks that are close to us. Adding this additional bit of weighting eliminated the problem of having no gradient to follow and allowed our optimization to converge at more reasonable control values.

IV. Results

A. Experimental Setup

We created several maps with different configurations of waypoints and visual landmarks. In each of these maps, the goal of the robot is to follow the waypoints sequentially and plan actions that would satisfy the three planning objectives listed in II-E. We model the robot as a moving point mass with the pose being its state $[x, y]$. The control is $[dx/dt, dy/dt]$ and control effort is calculated as total trajectory length. The robot is able to sense the distance and bearing of landmarks within a set sensing range. We run the integrated perception and planning components together to concurrently perform online state estimation and planning in the belief space. We swap the planning module of our pipeline with a simple LQR controller to evaluate the performance of our planner on several evaluation metrics listed in IV-B.

B. Evaluation Metrics

We evaluate the performance of the planner using four metrics: 1) state and trajectory estimation uncertainty, 2) final pose error, 3) total control effort expanded, and 4) total planning time. The state estimation uncertainty is computed by evaluating the trace of the state covariance without considering the cross-correlation terms, it reflects the total uncertainty in the estimated state trajectory. We evaluate the quality of estimated state trajectory using the Sums of Squares Error (SSE) metric. The total control effort expanded is the length of trajectory executed and it reflects the controller's ability to efficiently move to the goal location. The total planning time is the total time taken for the planner to finish computing the optimal control, and it represents the computation cost.

C. Experiment Result

We evaluate the performance of our planner on three maps: *Closure*, *Oasis* and *Uniform*.

1) Closure: Fig. 4 show the trajectory visualization and evolution of uncertainty for the LQR planner and our GBS planner respectively in the map called Closure. The black ellipses in the figures represent the uncertainty in the robot state estimate, where a bigger ellipse means higher uncertainty. The naive LQR planner does not take uncertainty into account while planning so, over the entire trajectory, state estimate uncertainty keeps growing. On the other hand, our GBS planner can steer towards prior observed landmarks to actively reduce uncertainty. The bottom right of Fig. 4 shows the sharp decrease in state estimate uncertainty after re-observation of landmarks due to the control actions produced by the GBS planner. This map demonstrated that our GBS controller is able to reach the final landmark with a much higher precision than LQR controller, as shown in table I. The total control effort of the GBS planner is greater than that of the LQR planner because of the extra path travelled

to re-observe landmarks, which is expected and acceptable. However, the total planning time of the GBS planner is one order of magnitude higher than LQR because of the computation overhead from iterative optimization.

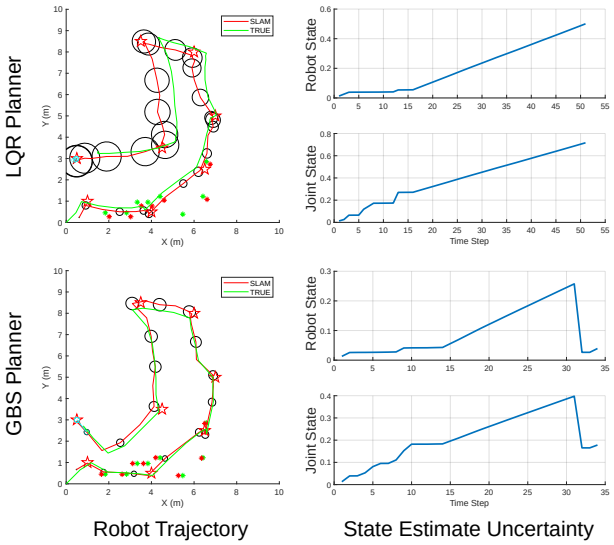


Fig. 4: Performance evaluation on map Closure

Planner	LQR	GBS
Total Control Effort (m)	24.41	26.59
Total Planning Time (s)	2.97	27.67
Final Distance to Goal (m)	1.09	0.41
Trajectory SSE (m^2)	46.33	21.24

TABLE I: Comparison of total control effort and planning time

We follow up our experiments on two additional maps to test the generalizability of this method. Oasis has landmarks located in the middle of the map while all the goal locations are on the outside of the map. This scenario should force the GBS planner to move towards the middle of the map when the uncertainty is too high. Uniform has uniformly distributed landmarks. We picked this map to see how the GBS planner would perform on a map that wasn't specially crafted to induce uncertainty reducing behavior.

2) Oasis: Fig. 5 shows that the GBS planner goes out of the way to move toward the center of the map on two occasions in order to reduce the uncertainty of the robot. The group of landmarks in the center of the map serves as a uncertainty reducing location for the robot to travel to whenever the uncertainty grows to high. Compared to LQR method, the GBS method was able to reach the final goal position with much less overall uncertainty and position error. Table II shows that this increase in goal reaching accuracy comes at the cost of slightly higher control effort and much higher planning time.

3) Uniform: On the Uniform map, Fig. 6 shows the GBS planner does nothing out of the ordinary compared to the LQR method. This is expected because the robot is almost

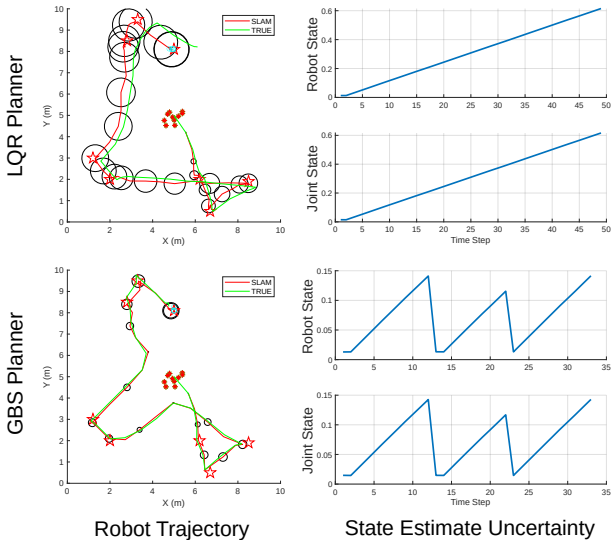


Fig. 5: Performance evaluation on map Oasis

Planner	LQR	GBS
Total Control Effort (m)	24.46	25.81
Total Planning Time (s)	2.98	35.91
Final Distance to Goal (m)	1.11	0.32
Trajectory SSE (m^2)	33.50	24.08

TABLE II: Comparison of total control effort and planning time on Oasis

always able to observe landmarks to suppress uncertainty, so the shortest path objective dominates the cost function. Even so, the GBS planner has better performance than the LQR planner in terms of uncertainty and final position error shown in Table III. We believe that this increase in performance is because the GBS planner is taking less control actions. By taking less control actions, the overall uncertainty of the state also grows less.

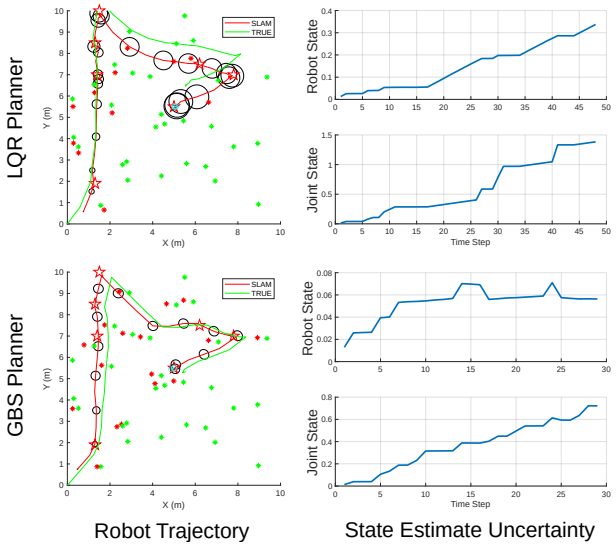


Fig. 6: Performance evaluation on map Uniform

Planner	LQR	GBS
Total Control Effort (m)	21.68	21.36
Total Planning Time (s)	3.29	40.43
Final Distance to Goal (m)	1.09	0.58
Trajectory SSE (m^2)	43.92	19.00

TABLE III: Comparison of total control effort and planning time on Uniform

V. Conclusion

In conclusion, we see that planning in the Generalized Belief Space has distinct advantages and disadvantages. While planning in the GBS is effective at lowering state estimation uncertainty, it results in higher computational cost and sometimes more control effort when compared to a naive LQR planning technique. Our experiments demonstrate the effectiveness of our implementation on three different maps. The first two maps, Closure and Oasis, show that planning in GBS outperforms LQR significantly in environments with unevenly distributed landmarks. The third map, Uniform, shows that in cases where there are always observable landmarks, the performances of the GBS and LQR planners are more similar because uncertainty does not grow as much.

For future work, we plan to generalize this planner to three dimensions with more realistic robot and observation models. We would also like to explore more options for parameter tuning because currently we spend a considerable amount of time hand-tuning parameters to balance conflicting objectives (goal-reaching and uncertainty-reducing). Additionally, we see a lot of parallels between our planning and estimation modules. They both perform MAP inference using non-linear optimization. We want to explore methods for combining these two modules into one single efficient graph based optimization routine using GTSAM. Last but not least, we would like to design more planning objectives to enforce constraints such as distance to landmarks, robot orientation, etc.

References

[1] R. Platt Jr, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," 2010.

[2] V. Indelman, L. Carlone, and F. Dellaert, "Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 849–882, 2015.

[3] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[4] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[5] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[6] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[7] V. Indelman, L. Carlone, and F. Dellaert, "Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 849–882, 2015. [Online]. Available: https://doi.org/10.1177/0278364914561102

[8] Z. Sjanic, M. Skoglund, T. Schön, and F. Gustafsson, *Solving the SLAM problem for unmanned aerial vehicles using smoothed estimates*. Linköping University Electronic Press, 2010.