C-OPT: Coverage-Aware Trajectory Optimization Under Uncertainty

Bobby Davis, Ioannis Karamouzas, and Stephen J. Guy

***Abstract*—We introduce a new problem of continuous, coverage-aware trajectory optimization under localization and sensing uncertainty. In this problem, the goal is to plan a path from a start state to a goal state that maximizes the coverage of a user-specified region while minimizing the control costs of the robot and the probability of collision with the environment. We present a principled method for quantifying the coverage sensing uncertainty of the robot. We use this sensing uncertainty along with the uncertainty in robot localization to develop C-OPT, a coverage-optimization algorithm which optimizes trajectories over belief-space to find locally optimal coverage paths. We highlight the applicability of our approach in multiple simulated scenarios inspired by surveillance, UAV crop analysis, and search-and-rescue tasks. We also present a case study on a physical, differential-drive robot. We also provide quantitative and qualitative analysis of the paths generated by our approach.**

***Index Terms*—Motion and Path Planning; Collision Avoidance; Reactive and Sensor-Based Planning**

I. Introduction

THE coverage planning problem has been extensively studied over the past two decades with important applications in intelligent farming, area surveillance, lawn mowing and mine sweeping [1]. In this problem, a robot has to observe or sweep an environment in an autonomous and efficient fashion. Even for a 2D region, finding an optimal-length solution to coverage planning is NP-hard in the general case [2]. Therefore, several heuristics and approximation algorithms have been proposed. The problem becomes even more challenging in the real world, since real robots do not have perfect sensors, and there is always some uncertainty in their positions and their monitoring. As such, the robots need to account for these uncertainties in their planning, while also avoiding collisions.

Belief space planning methods are designed to address the above problems of motion planning under uncertainty in a principled way [3]. LQG-MP [4] used this approach to evaluate the quality of trajectories, however, it did not consider the problem of generating trajectories. Many recent approaches model the planning under uncertainty problem as a partially-observable Markov decision process (POMDP), and solve it using local optimization techniques [3], [5], [6]. In this paper, we extend belief space planning to the problem of coverage planning under uncertainty. In particular, we focus on the specific problem of planning a locally optimal and collision-free trajectory between two given viewpoints.

We propose a new algorithm, referred to here as C-OPT, to address the coverage-aware trajectory optimization problem. This work makes three main contributions. First, we unify coverage planning, accounting for uncertainty in region sensing, and probabilistic collision avoidance into a single framework (Section III). Second, we present an approach for representing region uncertainty appropriate for our optimization problem (Section IV). Finally, we experimentally show the applicability of our approach across multiple scenarios on both real and simulated robots (Sections V and VI).

II. Related Work

Covering polygonal regions is a well known problem, and solutions can be efficiently calculated by cell decomposition, discretization in grids, or by use of boustrophedon paths [7]. More recent approaches have focused on acceleration through data structures [8], and extensions to nonholonomic robots using neural networks [9]. Complex domains have also been studied including coverage plans for 2.5D elevation maps [10], and 3D domains [11]. Recently, Kartal et al. [12] employed an MCTS-based approach for patrolling coverage problems in adversarial settings.

Accounting for positional uncertainty due to noise adds significant new challenges for motion planning. Recently, a class of techniques known as belief-space planning approaches has made significant progress in this area by explicitly modeling the dynamics of a robot's uncertainty about its environment during planning. Examples include belief trees [13] and belief roadmaps [14], which provide globally (asymptotically) optimal plans, along with local optimization approaches which are typically built on linearized, Gaussian approximations [3], [5], [6].

Recently, there has been work on combining coverage planning and belief space planning, with the goal of accounting for sensing uncertainty in the coverage planning. The work of Hollinger et al. [15] uses Gaussian processes to consider observation uncertainty in order to plan a sequence of sensing waypoints. Kim and Eustice consider positional uncertainty while using active-SLAM techniques to determine when to backtrack along a boustrophedon path [16]. The approach of Bretl and Hutchinson [17] provides guaranteed coverage of a region in a planar workspace assuming bounded worst-case uncertainty. Most recently, Charrow et al. [18] first uses a next-best-view approach to determine which part of an area needs

Manuscript received: August, 31, 2015; Revised December, 8, 2015; Accepted January, 18, 2016.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by University of Minnesota's MnDRIVE Initiative on Robotics, Sensors, and Advanced Manufacturing and NSF Grant No. 1544887.

Authors are with the Department of Computer Science and Engineering, University of Minnesota, USA. {davis,ioannis,sjguy}@cs.umn.edu

Digital Object Identifier (DOI): see top of this page.

to be explored next, and then uses a gradient-based method to locally optimize a collision-free trajectory to get there.

Our work. Similar to the work discussed above, our work seeks to develop belief-space approaches to coverage planning. However, our approach is *complementary* to the above methods. Whereas previous work has typically considered global planning of best viewpoints (while assuming direct paths will be taken between these viewpoints), our work seeks to optimize the path taken in-between these global waypoints to maximize the information they can provide to the robot. Similar to [5], we optimize trajectories over belief-space using iLQG [19] to find a locally optimal solution, and use the LQG-MP formulation [4] to account for the collision probability with the environment. However, our approach extends these works to the coverage planning domain. To the best of our knowledge, no previous work has sought to address this issue of coverage-aware trajectory optimization.

III. C-OPT Framework

Our problem is motivated by cases such as UAV surveillance of a field or campus where a robot knows the general environment (e.g., location of buildings and other obstacles), but would like to record the current conditions on the ground (e.g., field health surveillance, or search-and-rescue). Let $(\mathbf{x}_t, \Sigma_t)$ represent the *belief state* of a robot at timestep t, where $\mathbf{x}_t$ is the robot's state and Σ_t is it's uncertainty. We assume we are given a robot model, $\mathcal{E}$, that, given a previous belief state and a control $(\mathbf{u}_t)$, produces a new belief state:

$$(\mathbf{x}_{t+1}, \Sigma_{t+1}) = \mathcal{E}(\mathbf{x}_t, \Sigma_t, \mathbf{u}_t). \tag{1}$$

This process can be modeled, for instance, by an extended Kalman filter (EKF).

To account for region coverage, let $\mathbf{p}_t$ represent a model of the robot's uncertainty about the target region at timestep t. We assume the robot uses a sensor to obtain information about the region. We also assume we have a model of the sensor, $\mathbf{g}$, which, given a current belief state and a current region uncertainty, produces a new region uncertainty:

$$\mathbf{p}_{t+1} = \mathbf{g}(\mathbf{p}_t, \mathbf{x}_t, \Sigma_t). \tag{2}$$

We are tasked with navigating between two user-given waypoints, denoted by an initial belief state $(\mathbf{x}_0, \Sigma_0)$ and a goal state $(\mathbf{x}_G)$. Our goal, then, is to navigate between these two waypoints, maximizing coverage of the region, while minimizing the chance of collision and the control cost. We write this formally as a constrained optimization problem over a series of controls $(\mathbf{u}_1 \dots \mathbf{u}_{T-1})$:

$$\underset{\mathbf{u}_1 \dots \mathbf{u}_{T-1}}{\arg\min}\ \mathcal{C}_{tot}(\mathbf{x}_0 \dots \mathbf{x}_T, \Sigma_0 \dots \Sigma_T, \mathbf{p}_0 \dots \mathbf{p}_T, \mathbf{u}_1 \dots \mathbf{u}_{T-1}) \tag{3}$$

such that the evolution of the robot state estimate $(\mathbf{x}, \Sigma)$ and the region uncertainty $(\mathbf{p})$ follow Eqs. 1 and 2, respectively.

This formulation allows us to directly optimize over control space, in contrast to the existing paradigm, which, generally first plans a series of states (waypoints) and then finds a controller to drive safely between those waypoints. Our unified framework allows us to simultaneously consider the effect a control sequence has on both collision avoidance and region coverage.

Algorithm 1: C-OPT

Input: Initial trajectory $(\mathbf{u}_1 \dots \mathbf{u}_T)$, $\mathbf{x}_0$, $\mathbf{x}_g$
Output: Optimized trajectory $(\mathbf{u}_1 \dots \mathbf{u}_T)$

1 **while** $\Delta\mathbf{u} > 0$ **do**
2 $\Delta\mathbf{u}_1 \dots \Delta\mathbf{u}_T = computeOptimalControls()$
3 **while** change in $\mathcal{C}_{tot} < 0$ **do**
4 **for** $t \in 1 \dots T$ **do**
5 $\mathbf{u}_t = \mathbf{u}_t + \epsilon Update(\epsilon, \Delta\mathbf{u}_t, \mathbf{b}_t)$
6 $\left[\begin{smallmatrix}\mathbf{x}_{t+1}\\ \Sigma_{t+1}\end{smallmatrix}\right] = EKF(\left[\begin{smallmatrix}\mathbf{x}_t\\ \Sigma_t\end{smallmatrix}\right], \mathbf{u}_t)$
7 Update region uncertainty (Eq. 2)
8 **end**
9 reduce ϵ *//Line scan*
10 **end**
11 **end**

A. Optimization via iLQG

Given the above problem formulation, we can now introduce our proposed coverage-optimization for trajectories algorithm (C-OPT, Alg. 1). Following the approach proposed by van den Berg et al. [5], we model uncertainty in the state and observations as Gaussian distributions, and minimize the total cost function (Eq. 4) using a belief-space variant of iLQG. Here the positional belief dynamics over time are modeled as an extended Kalman filter. The iLQG framework allows us to avoid discretizing the control and state spaces, and allows for any robot dynamics and observation models.

At a high level, iLQG works by taking an initial trajectory and minimizing its cost via gradient descent based on the Hessian of the cost function with respect to controls (line 2). We use a line search approach to reduce sensitivity to the learning rate, and to speed up convergence (lines 3 and 9), by partially updating the controls with a varying learning rate ϵ (line 5). As it is a local optimization technique, iLQG depends on the initial trajectory. Standard global planning techniques, such as RRT* [20] or SST* [21], are able to find optimal control paths between waypoints, but don't consider region coverage. As such, these approaches may not give a good initialization (e.g., consider cases where the goal state is the same as the initial state). Instead, we propose initialization with simple paths which coarsely cover the region, while respecting the robot dynamics. The effect of the initial path on the final trajectory is explored in the accompanying video.

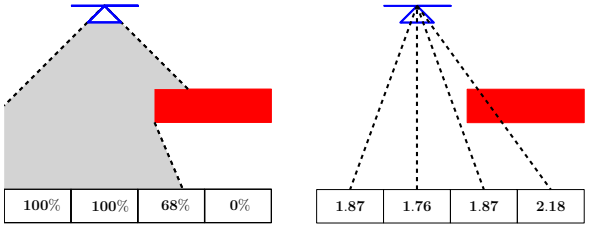
B. Optimization Formulation

To allow our problem to be integrated into the iLQG framework, we split the cost into an intermediate cost, $\mathcal{C}'$, at each timestep, and a separate cost, $\mathcal{C}$, for the final timestep:

$$\mathcal{C}_{tot} = \sum_t \mathcal{C}'(\mathbf{x}_t, \Sigma_t, \mathbf{u}_t) + \mathcal{C}(\mathbf{x}_T, \Sigma_t). \tag{4}$$

We define the cost for each intermediate state as:

$$\mathcal{C}'(\mathbf{x}_t, \Sigma_t, \mathbf{p}_t, \mathbf{u}_t) = \alpha_u \|\mathbf{u}_t\|^2 + \alpha_c \mathcal{C}_{\mathcal{C}}(\mathbf{x}_t, \Sigma_t) + \alpha_p' \mathcal{C}_{\mathcal{P}}(\mathbf{p}_t), \tag{5}$$



(a) Visibility (b) Distance

Fig. 1: **Modeling the viewing uncertainty of a region**. The robot accounts for (a), the visibility, by calculating what percentage of the region is within its field of view (the gray shaded region) and not obstructed by obstacles (the red rectangle). The robot also accounts for (b), the distance from the center of the region to the robot.

where $\mathcal{C}_\mathcal{C}(\mathbf{x}_t, \Sigma_t)$ is a cost accounting for the risk of collision given the robot's belief state, and $\mathcal{C}_\mathcal{P}(\mathbf{p_t})$ is a cost accounting for the uncertainty in the coverage region. Here, α_u is the weight of the control cost, α_c is the weight of the collision cost, and $\alpha_p^{'}$ is the weight of the uncertainty cost.

The cost function for the final state is of the form:

$$\mathcal{C}(\mathbf{x}_T, \Sigma_T, \mathbf{p}_T) = \alpha_d \|\mathbf{x}_T - \mathbf{x}_g\|^2 + \alpha_p \mathcal{C}_\mathcal{P}(\mathbf{p}_T). \quad (6)$$

This penalizes distance from the goal state, and the final uncertainty over the region. Here, α_d is the weight of reaching the goal, and α_p is the weight of the final uncertainty cost. Overall, given Eq. 4, we seek to find the set of controls, $\mathbf{u_t}$, which minimizes the total cost.

IV. Approximation Approach

The above optimization approach requires modeling three aspects of the robot's environment, each with an associated cost: the uncertainty of the coverage region, the chance of a collision, and the dynamics of the robot.

A. Region Uncertainty

To model the spatially varying nature of a robot's uncertainty over a region, we discretize the region into k bins of equal size, where p_i denotes the uncertainty of bin i: $\mathbf{p}_t = \begin{bmatrix} p_1 & \dots & p_k \end{bmatrix}^T$. Here p_i represents the accuracy of our sensing estimate (as measured by the estimated variance). We can then describe the total uncertainty over the region as:

$$\mathcal{C}_\mathcal{P}(\mathbf{p}_t) = s \sum_i p_i, \quad (7)$$

where s is the size of each bin. As k approaches infinity, the quality of our approximation of the uncertainty increases. However, as we will discuss in Section VII, a small value of k is typically sufficient to obtain high-quality paths.

We assume that each observation is independent of each other to create a simple update rule from a Kalman filter. In particular, we assume the measurement of each bin i is sensed as its true value plus some Gaussian noise with variance w_i corresponding to the sensor uncertainty for bin i. Given a previous estimate of the uncertainty of bin i, p_i, and a new measurement of the same bin with uncertainty variance w_i, we can compute a new lower value of the uncertainty using the (one dimensional) Kalman filter equation:

$$p_{i_{t+1}} = p_{i_t} - \frac{p_{i_t}^2}{p_{i_t} + w_i} \quad (8)$$

Using this update rule, we can update the uncertainty of all bins in parallel.

For any given bin i, we are free to model the sensor uncertainty, w_i, as any spatially varying function dependent on the relative state of the robot to the bin. In all of our experiments, we assume the accuracy of the sensor falls off quadratically with the distance to the robot, d_i. In addition, we assume that the uncertainty of a bin rises quickly with the percentage that is obscured from the robot's view by obstacles or limited field-of-view (and is infinite for any non-visible bin). Finally, we also account for a base level of uncertainty always present due to imperfections in the sensor. Our sensing model is therefore:

$$w_i = \begin{cases} \beta_v(1 - v_i) + \beta_d d_i^2 + \beta_b & \text{if } v_i > 0 \\ \infty & otherwise \end{cases} \quad (9)$$

where β_v and β_d are the constants that control the falloff of the view quality, and β_b is the base uncertainty. The visibility term, v_i, measures the percent of bin i that is within the field of view of the robot, and not obscured by obstacles (see Fig. 1(a)). The distance, d_i, is calculated as the Euclidean distance from the robot to the center of the bin (see Fig. 1(b)). To simplify the visibility computation, we make a maximum-likelihood assumption (i.e., assume the agent's state is the mean of the estimated distribution) when computing visibility. Other sensing models can be easily incorporated by simply replacing Eq. 9.

B. Collision Probability

We can conservatively approximate the probability of colliding with an obstacle based on the minimum number of standard deviations the mean position is away from the closest obstacle. Following the LQG-MP approach [4], we compute the distance to collision in a space transformed such that the covariance matrix is the identity matrix. In this transformed space, the number of standard deviations is equivalent to the Euclidean distance. We compute the transformed x distance, d_x, and y distance, d_y, separately and the total chance of collision is conservatively computed as:

$$\mathcal{C}_\mathcal{C}(\mathbf{x}_t) = 4\, cdf(-d_x)\, cdf(-d_y), \quad (10)$$

where cdf is the Gaussian cumulative distribution function. For ease of computation, we model the robot by its bounding disk/sphere. We note that our approach can be extended directly to arbitrary robot shapes at the cost of additional computation.

C. Robot Dynamics and Localization

Our approach supports a wide variety of robot control dynamics, which will in turn affect the computed path. We express the control dynamics in terms of two multi-dimensional

variables: $\mathbf{x}_t$, the robot's state at time t, and $\mathbf{u}_t$, the controls applied at time t. Each dynamics model, then, corresponds to the following equation:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t). \tag{11}$$

We assume this control equation is differentiable everywhere for both $\mathbf{x}$ and $\mathbf{u}$.

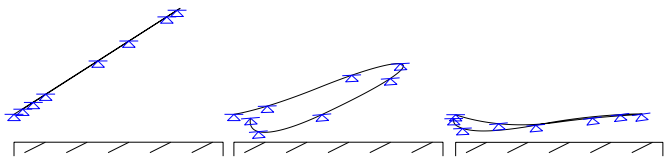
In our experiments below, we employ three different robot dynamics, including both holonomic and non-holonomic models of robot motion (see Appendix A for more details):

- **2D Acceleration Control**. Here the robot's state is a 5D vector (2 position, 2 velocity, and 1 orientation). The robot moves by accelerating in each direction independently. This is a common, simplified model of aerial robot dynamics.
- **3D Acceleration Control**. Here the robot's state is a 9D vector (3 position, 3 velocity, and 3 orientation). This is an extension of the previous dynamics function to 3D.
- **Oriented 2D Acceleration**. Here the robot's state is a 4D vector (2 position, 1 velocity, and 1 orientation). The robot is controlled by a rotational velocity and a linear acceleration. This model serves as a better representation for a car-like robot, which does not have omni-directional acceleration, but rather accelerates in a single direction. Optionally, we can incorporate the influence of gravity to better model an aerial robot, such as a quadrotor.

Localization: In all scenarios, we assume a direct observation of the current position and orientation of the robot. The observation function is then given by $\mathbf{z}_t = \begin{bmatrix} x & y & \theta \end{bmatrix}^T + \mathbf{n}_t$. We also assume a uniform localization uncertainty throughout space (e.g., as one might expect from GPS localization). Our method can support different types of localization, and Appendix B discusses results with more complicated localization models.

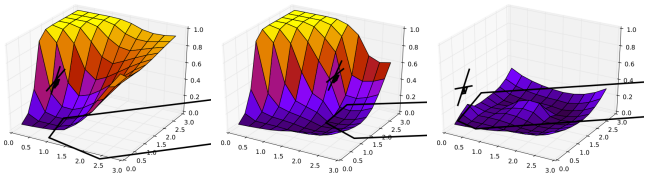
D. Example Optimization via iLQG

An example of iLQG optimization of a trajectory is shown in Fig. 2. The robot in this scenario uses the 2D acceleration control. The robot has the same start and goal position, and has to cover the ground. The initial trajectory is shown on the left where the robot simply goes out and back. Subsequent figures shows the path refinement over iLQG iterations. In the refined path, the robot can be seen to get closer to the ground, while following a more efficient route, and reaching its goal position.



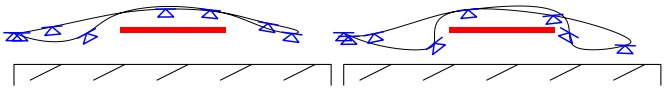
(a) Initial Trajectory (b) Intermediate Plan (c) Final Trajectory

Fig. 2: **Evolution of initial trajectory over iLQG iterations**. Each iteration reduces the overall cost function (Eq. 4) and improves the planned path.



(a) t = 2 (b) t = 5 (c) t = 9

Fig. 3: **3D coverage planning with acceleration controls**. Region uncertainty is shown at time t = 2, 5, and 9 seconds. The black quadrilateral represents the area the robot can see from its current state. The planned path successfully observes the entire region.



(a) $\alpha_p = 5$ (b) $\alpha_p = 20$

Fig. 4: **Effect of uncertainty cost parameter**. An omni-directional, acceleration controlled robot attempting to sense the floor. The rectangle is an opaque obstacle. We show the optimized path for two different weights of the uncertainty cost (α_p). As the uncertainty weight increases, the path becomes much more aggressive in getting to optimal views.

V. Simulations and Results

We implemented C-OPT in C++ and ran several simulated experiments. In all of our experiments, unless otherwise specified, we assume the robot senses through a rigidly mounted camera with a fixed field view of 90° and uses the following values for the cost function: $\alpha_u = 5, \alpha_c = 1000, \alpha_p^{'} = 10, \alpha_d = 10T, \alpha_p = 10T, \beta_v = 9.5, \beta_b = 0.5, \beta_d = 10$, where T is the number of timesteps planned over. We also used a region discretization, k, of 25 bins, and set each bin to an initial uncertainty value of 1. These values were chosen such as the iLQG framework could focus on locally optimal solutions in terms of the coverage of a region, collision-freeness and reachability to the goal. For videos of these and other results, please refer to the accompanying video.

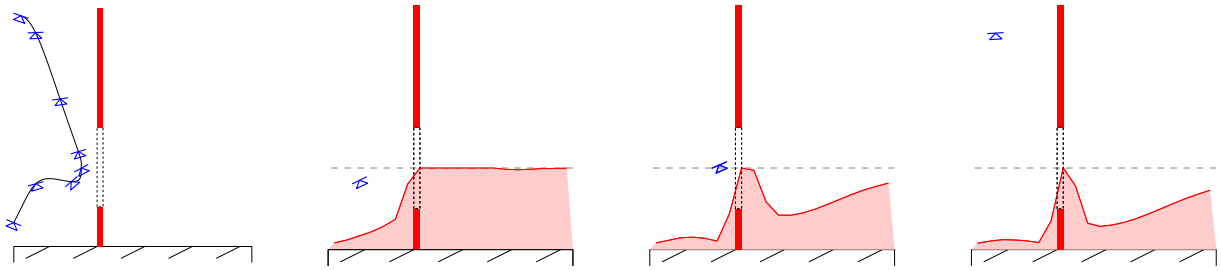
A. 3D Coverage Example

In this 3D scenario, the robot must sense a 2D region using acceleration dynamics. The initial path is a downward facing circular sweep of the region. Figure 3 shows the robot at three different points along its final trajectory, along with the region uncertainty at each of those timesteps. By the end of the path, the robot has successfully observed the entire region at high accuracy.

For ease of illustration, all remaining examples will be performed in 2D.

B. Acceleration Control with an Obstacle

This scenario shows the effect of obstacles in the environment. We utilize the same 2D acceleration and initial path as in Fig. 2. The optimized path is shown in Fig. 4 for two different uncertainty cost parameters. By increasing the uncertainty



(a) Final Trajectory (b) t = 2 (c) t = 5 (d) t = 9

Fig. 5: **A closed window**. Red rectangles are opaque obstacles, while the dashed box is a transparent obstacle. (a) Our C-OPT Path (b)-(d) the evolution of the uncertainty (the filled region) over time. Dashed line shows the initial region uncertainty.

cost, the robot is more willing to execute more aggressive controls to reach more optimal viewing positions. Of particular note is that the robot looks under the obstacle from both sides. This behavior is desired because it greatly reduces the uncertainty under the obstacle. As this is a local optimization method, we can only optimize to paths that belong to the same homotopic class as the initial trajectory. Thus, since our initial trajectory flies over the obstacle, we cannot obtain a solution that steers the robot underneath.

C. Oriented Acceleration

In this scenario, we consider a quadrotor-like robot with the oriented acceleration dynamics function and gravity. We assume the sensor is a downward facing camera rigidly attached to its frame. As a result of the oriented dynamics, the robot can only look in the direction opposite that to which it is accelerating. To illustrate the diversity of scenarios our approach supports, we place an impassable wall in the robot's environment. While the robot cannot fly through this wall, there is a transparent window through which it can see the other side. The goal is located in the upper left, and the initial path flies directly from start to goal.

Despite the challenging nature of this scenario, the robot is still able to find a path which efficiently reaches the goal, and provides good coverage of the ground. Fig. 5(a) shows the planned path, and the intermediate evolution of the region uncertainty over time is shown in Figs. 5(b)-5(d). As can be seen, the robot first flies up partway, uses gravity to fall toward the bottom of the window (in an effort to orient the camera to see outside the window as much as possible), then flies upward slightly away from the window, maximizing the view of the right side of the ground from the rear-facing camera on its way to the goal.

VI. Case Study: Point-to-Point Path Optimization

In this section, we describe the results of our experiments using C-OPT to plan a path for a robot moving through our research lab. For these results, we used a ClearPath Turtlebot 2, equipped with a Kinect for onboard sensing. As an experimental set-up, we assumed the current/initial state of the robot was known (with some uncertainty), and that we had a goal/next state which we assume to be known in advance (e.g., provided by a next-best-view planner). Here, the initial state was on one side of the lab, and the final state on the other. In the process of navigation, the robot must reach its goal state, and avoid collisions with the desks while sensing the desired lab wall (See Fig. 6).

During execution we compared the result from two planning

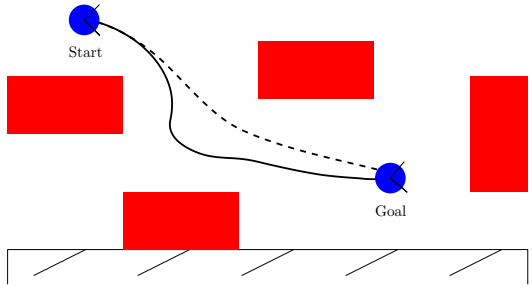
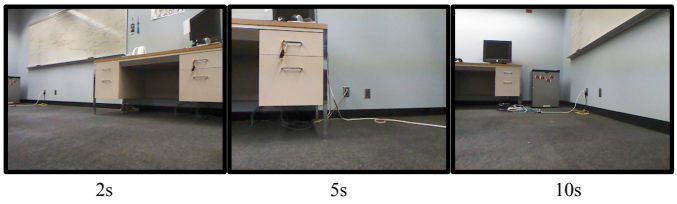


Fig. 6: **Point-To-Point Optimization**. The dashed line shows the collision-free trajectory, while the solid line shows our C-OPT trajectory. The red rectangles are the obstacles (desks in the lab). Planning with our approach provides good views of the entire target wall.

(a) Collision-free Planning (iLQG)



(b) Coverage Optimized Trajectory (C-OPT)

Fig. 7: **Implementation on a robot**. First-person view from the onboard camera of the robot as it executed both the collision-free plan and the C-OPT trajectories. In the 5s image, the coverage-optimized trajectory is able to see the wall on the right side of the desk.

Scenario	# Obstacles	Belief state size	# iLQG iterations	Time per iteration (ms)	Total time (s)
2D Acceleration – Fig. 2	0	45	23	616.6	14.181
2D Acceleration – Fig. 4	1	45	21	617.4	12.966
2D Window – Fig. 5	3	45	21	761.7	15.996
2D Lab Demo – Fig. 6	4	34	18	1013	18.2328
3D Coverage – Fig. 3	0	79	36	1803	64.907

TABLE I: Runtime results for each scenario. Runtimes are averaged over 5 runs.

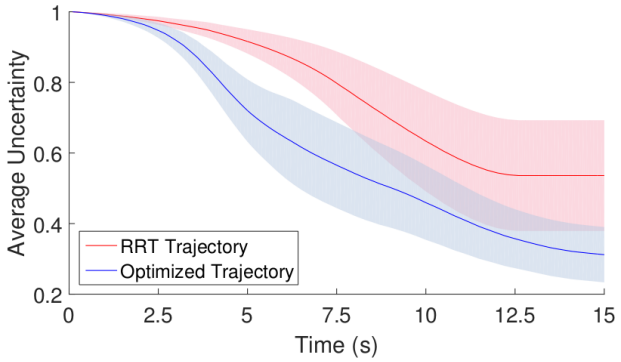


Fig. 8: **Optimized path quality comparison**. The red line shows the average region uncertainty over the collision-free planning path. The blue shows the results using C-OPT. Shaded regions denote the respective 95% confidence intervals. The C-OPT trajectory enables the robot to see much more of the lab's wall, resulting in significantly less uncertainty.

approaches to guide the robot navigation: one using standard belief-space iLQG (as proposed in [5], hereafter referred to as collision-free planner), and one based on our C-OPT proposed approach. Both approaches shared the same start and goal states, and were initialized with a trajectory from an uncertainty-aware RRT (following an approach inspired by [22]). In the C-OPT run, the robot additionally attempted to generate good views of the lab's wall while navigating to the next state.

One potential issue with C-OPT is that because it is a local optimization method, the final path is sensitive to the path it was initialized with. One measure we took to alleviate this issue was to plan a path as if all obstacles were translucent. This allows the robot to consider a wider variety of paths during trajectory optimization and "punch through" some local minima. A final optimization can be applied where obstacles are properly treated as opaque to further refine the final path.

In our experiments, both the collision-free planner and C-OPT were able to successfully reach their goal states without any collisions (despite the presence of noise in execution). However, when using our C-OPT approach, the robot was able to see much more of the target wall while still reaching its goal. Figure 7 shows a comparison between the two executions from the robot's point of view. When using the collision-free planner, the robot goes slowly, straight to the goal, diverting only to avoid collision with the upcoming desk. With our approach, the robot initially moves much more quickly as not much of the wall can be seen (Fig. 7 - 2s), then turns to the right to see the right corner of the room (Fig 7 - 5s), and finally approaches the goal position. As it reaches the goal, the robot stays slightly closer to the wall in order to reduce sensor uncertainty (Fig. 7 - 10s).

The effect of this trajectory optimization on reducing the uncertainty in the knowledge of the wall can be seen in Fig. 8. Not only is the average uncertainty of all bins along the wall reduced in half, but so is the spread in uncertainties (as measured by the 95% confidence interval).

VII. Discussion

A. Performance and Runtime Complexity

The performance of our approach was evaluated on a 3.4GHz Intel® Core™ i5 PC using 4 cores. Runtime results for each scenario are shown in Table I. All of the 2D scenarios share the same state space, and therefore have the same 45 dimensional belief state. The 3D scenario has a larger, 79 dimensional belief state space to accommodate the extra degrees of freedom in translation and rotation. We note that planning took less than 20s for most scenarios, which is fast enough to be used in a real-time planning framework (assuming the next goal was further than 20s away, and the planner was coupled with a reactive method for dynamic obstacle avoidance).

With an n dimensional robot state, the asymptotic runtime of iLQG is $O(n^6)$ [5]. This runtime comes from the most expensive operation, which involves multiplying two matrices of dimension belief state size $O(n^2)$ by belief state size. In our case, the ground state (of discretization k) is not part of the robot state. As such, our belief state is of size $O(k+n^2)$, leading to a runtime of $O((k+n^2)^3)$. Typically values for n are small and are fixed for a given robot description. While the runtime increases cubically with k, a fine discretization is not always needed to produce good coverage paths. In many of our experiments, paths generated with 30 bins were within 1% of the coverage quality as paths with 200 bins (see Appendix C).

If we include the number of obstacle vertices, v, as a free variable, we also need to include the time taken to compute the Hessian of the cost function. As we do this operation numerically, we need to evaluate the cost function $(k+n^2)^2$ times. Computing the obstacle cost (Eq. 10) takes $O(v)$ time in 2D and 3D, as we assume a disk model for the robot. The control cost (Eq. 5) can be computed in time linear in the control dimension, and the uncertainty cost (Eq. 7) can be computed in $O(k)$ time. In 2D, computing the visibility of a bin (Eq. 9) takes $O(v)$ time. However, the computation is more expensive in 3D. Assuming convex obstacles, this operation can be done in $O(mv^2 \log v)$ time, where m is the number of obstacles, by first projecting the obstacles to the surface, and then computing the union of those polygons [23].

B. Convergence Analysis

C-OPT is a local optimization approach, and hence the final trajectory is influenced by the initial trajectory. However, C-

OPT is invariant to initial trajectories that belong to the same homotopy class, and all trajectories in each homotopy will lead to the same solution (see accompanying video).

We also analyzed the sensitivity of the total cost to the number of iLQG iterations. In all of our experiments, the cost quickly decreases over the first few iterations, and then slowly converges to a locally optimal value (See Appendix D for more details). This could allow for an easy integration of C-OPT into an anytime algorithm, as a valid trajectory quickly becomes available.

VIII. Conclusions and Future Work

In this paper, we proposed a new approach to coverage planning under uncertainty using trajectory optimization. We adopted methods from belief-space planning to create an algorithm, C-OPT, capable of finding a locally optimal solution for this new problem. Our approach develops a principled way of accounting for the sensing uncertainty of a given region as the robot travels along a path while accounting for positional uncertainty. We experimentally validated our approach via simulations in different scenarios with different robot dynamics models, and showed results on a physical robot. In all our scenarios, our algorithm provided increased coverage of the target region, while maintaining efficient, collision-free paths.

Limitations: The assumptions outlined in Section 3 lead to some important limitations in how our approach can be generally applied in practical settings. Firstly, because we assume the robot knows the location of obstacles, our method cannot be directly applied to exploration tasks in unseen environments. We note though, that this limitation can be partially overcome by exploiting the concept of frontiers [18] and treating these exploration frontiers as part of the coverage target. Additionally, our implementation can be too slow for realtime planning when considering large environments with many obstacles or fine discretization of the coverage region. Lastly, the path is only locally optimal, and must be integrated with a global path planning approach.

These above limitations suggest many promising avenues for future work. Some aspects of the computation may be well suited for parallelized acceleration using GPUs to enable real-time planning. Planning in complex scenes with many obstacles can also be sped up with recent techniques for fast geometric processing and distance queries [24], [25]. Finally, we would like to address the potential to get stuck in poor local minima. While approaches such as the translucent obstacle heuristic (we used in Section VI) or planning repeatedly with many different initial paths can reduce the effect of this issue, we hope to develop methods which can produce trajectories which are true global minima.

References

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[2] A. Zelinsky, R. A. Jarvis, J. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *International Conference on Advanced Robotics*, vol. 13, 1993, pp. 533–538.

[3] R. Platt Jr, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," *Robotics: Science and Systems*, 2010.

[4] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[5] J. van den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[6] S. Patil, Y. Duan, J. Schulman, K. Goldberg, and P. Abbeel, "Gaussian belief space planning with discontinuities in sensing domains," in *IEEE International Conference on Robotics and Automation*, 2014.

[7] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[8] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara, "BSA: a complete coverage algorithm," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 2040–2044.

[9] Y. Guo and M. Balakrishnan, "Complete coverage control for nonholonomic mobile robots in dynamic environments," in *IEEE International Conference on Robotics and Automation*, 2006, pp. 1704–1709.

[10] J. Jin and L. Tang, "Coverage path planning on three-dimensional terrain for arable farming," *Journal of Field Robotics*, vol. 28, no. 3, pp. 424–440, 2011.

[11] P. Cheng, J. Keller, and V. Kumar, "Time-optimal uav trajectory planning for 3d urban structure coverage," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 2750–2757.

[12] B. Kartal, J. Godoy, I. Karamouzas, and S. J. Guy, "Stochastic tree search with useful cycles for patrolling problems," in *IEEE International Conference on Robotics and Automation*, 2015.

[13] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 723–730.

[14] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *International Journal of Robotics Research*, vol. 28, pp. 1448–1465, 2009.

[15] G. A. Hollinger, B. Englot, F. S. Hover, U. Mitra, and G. S. Sukhatme, "Active planning for underwater inspection and the benefit of adaptivity," *International Journal of Robotics Research*, vol. 32, no. 1, pp. 3–18, 2013.

[16] A. Kim and R. M. Eustice, "Active visual SLAM for robotic area coverage: Theory and experiment," *International Journal of Robotics Research*, vol. 34, pp. 457–475, 2014.

[17] T. Bretl and S. Hutchinson, "Robust coverage by a mobile robot of a planar workspace," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 4582–4587.

[18] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar, "Information-theoretic planning with trajectory optimization for dense 3D mapping," in *Robotics: Science and System*, 2015.

[19] W. Li and E. Todorov, "Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system," *International Journal of Control*, vol. 80, no. 9, pp. 1439–1453, 2007.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[21] Y. Li, Z. Littlefield, and K. E. Bekris, "Sparse methods for efficient asymptotically optimal kinodynamic planning," in *Algorithmic Foundations of Robotics XI*, 2015, pp. 263–282.

[22] B. Luders, M. Kothari, and J. P. How, "Chance constrained RRT for probabilistic robustness to environmental uncertainty," in *AIAA guidance, navigation, and control conference, Toronto, Canada*, 2010.

[23] F. Martínez, A. J. Rueda, and F. R. Feito, "A new algorithm for computing boolean operations on polygons," *Computers & Geosciences*, vol. 35, no. 6, pp. 1177–1185, 2009.

[24] A. Lee, Y. Duan, S. Patil, J. Schulman, Z. McCarthy, J. van den Berg, K. Goldberg, and P. Abbeel, "Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 5660–5667.

[25] C. Lauterbach, Q. Mo, and D. Manocha, "gProximity: Hierarchical GPU-based Operations for Collision and Distance Queries," in *Computer Graphics Forum*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 419–428.

APPENDIX

A. Robot Dynamics Models

We list below the different robot dynamics models we considered in our various experiments.

1) 2D Acceleration
This model was used in Sections IV-D, V-B, and Appendix B.

$$\begin{aligned}\mathbf{x}_t &= \begin{bmatrix} x & y & \theta & v_x & v_y \end{bmatrix}^T \\ \mathbf{u}_t &= \begin{bmatrix} a_x & a_y & \omega \end{bmatrix}^T \\ \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) &= \begin{bmatrix} x + v_x\Delta t + \frac{1}{2}a_x\Delta t^2 \\ y + v_y\Delta t + \frac{1}{2}a_y\Delta t^2 \\ \theta + \omega\Delta t \\ v_x + a_x\Delta t \\ v_y + a_y\Delta t \end{bmatrix}\end{aligned} \tag{12}$$

2) 3D Acceleration
This model was used in Section V-A. This is an extension of the above 2D Acceleration model to 3D.

3) Oriented Acceleration
This model was used in Section VI.

$$\begin{aligned}\mathbf{x}_t &= \begin{bmatrix} x & y & \theta & v_x & v_y \end{bmatrix}^T \\ \mathbf{u}_t &= \begin{bmatrix} \tau & \omega \end{bmatrix}^T \\ \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) &= \begin{bmatrix} x + v_x\Delta t - \frac{1}{2}\cos(\theta)\tau\Delta t^2 \\ y + v_y\Delta t + \frac{1}{2}(\sin(\theta)\tau)\Delta t^2 \\ \theta + \omega\Delta t \\ v_x - \cos(\theta)\tau\Delta t \\ v_y + (\sin(\theta)\tau)\Delta t \end{bmatrix}\end{aligned} \tag{13}$$

4) Oriented Acceleration with Gravity
This model was used in Section V-C.

$$\begin{aligned}\mathbf{x}_t &= \begin{bmatrix} x & y & \theta & v_x & v_y \end{bmatrix}^T \\ \mathbf{u}_t &= \begin{bmatrix} \tau & \omega \end{bmatrix}^T \\ \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) &= \begin{bmatrix} x + v_x\Delta t - \frac{1}{2}\cos(\theta)\tau\Delta t^2 \\ y + v_y\Delta t + \frac{1}{2}(\sin(\theta)\tau + g_z)\Delta t^2 \\ \theta + \omega\Delta t \\ v_x - \cos(\theta)\tau\Delta t \\ v_y + (\sin(\theta)\tau + g_z)\Delta t \end{bmatrix}\end{aligned} \tag{14}$$

B. Localization Models

In some scenarios, if GPS is not available, localization can be provided via beacons. However, it may be infeasible to distribute these beacons around the environment. We consider the case where we only have one localization beacon at the initial position of the robot, and the accuracy of the beacon decays as the robot moves further away. We use the same 2D acceleration dynamics and the obstacle environment shown in Fig. 4 to highlight the difference between the original optimized path, and the path optimized with spatially-varying uncertainty. We assume a quadratic decay of the signal strength, so our localization is worse the farther away we are. Results are shown in Fig. 9. Comparing to Fig. 4(b), we can see that the distance from the obstacle is significantly greater on the right side of the obstacle, and even greater on the return trip past the right side. This reflects the greater uncertainty the robot has on the right side of the obstacle.

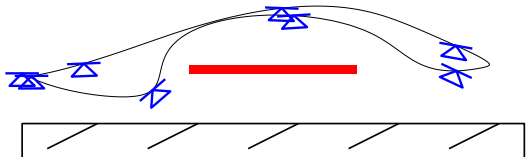


Fig. 9: **Planning with spatially varying uncertainty**. A localization beacon is placed at the starting position on the left. Positional sensing quality decays the further the robot is from the beacon. The robot plans a path that stays further from the obstacle when it is less certain about its position.

C. Effect of Ground Discretization

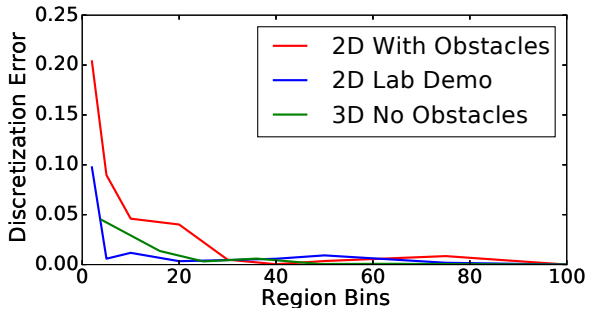


Fig. 10: **Coverage quality for various discretizations**. The lines show the effect of increasing the region discretization on the average uncertainty of the optimized trajectory. The region uncertainty for each discretization is compared to the uncertainty obtained planning a path where the region is discretized into 200 bins. As k increases, the coverage quality of the path converges.

D. Cost over optimization iterations

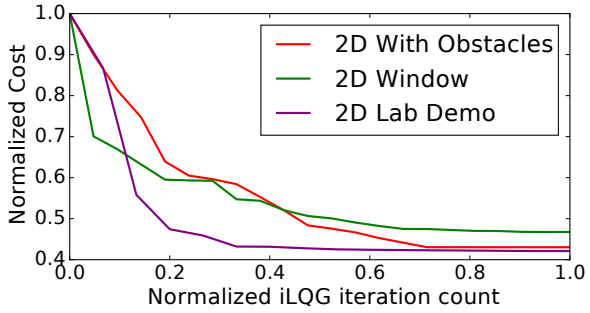


Fig. 11: **Cost over iLQG iterations**. The cost over iterations for each of the scenarios is shown. Costs quickly decrease at first, and then converge to a locally optimal value. The costs for each scenario are normalized based on the cost of the initial trajectory. The iLQG iterations for each scenario are normalized based on the total number of iterations it took to run each scenario.