Scaling up Gaussian Belief Space Planning through Covariance-Free Trajectory Optimization and Automatic Differentiation

Sachin Patil, Gregory Kahn, Michael Laskey, John Schulman,
Ken Goldberg, Pieter Abbeel

University of California, Berkeley

Abstract. Belief space planning provides a principled framework to compute motion plans that explicitly gather information from sensing, as necessary, to reduce uncertainty about the robot and the environment. We consider the problem of planning in Gaussian belief spaces, which are parameterized in terms of mean states and covariances describing the uncertainty. In this work, we show that it is possible to compute locally optimal plans without including the covariance in direct trajectory optimization formulations of the problem. As a result, the dimensionality of the problem scales linearly in the state dimension instead of quadratically, as would be the case if we were to include the covariance in the optimization. We accomplish this by taking advantage of recent advances in numerical optimal control that include automatic differentiation and state of the art convex solvers. We show that the running time of each optimization step of the covariance-free trajectory optimization is $O(n^3T)$, where n is the dimension of the state space and T is the number of time steps in the trajectory. We present experiments in simulation on a variety of planning problems under uncertainty including manipulator planning, estimating unknown model parameters for dynamical systems, and active simultaneous localization and mapping (active SLAM). Our experiments suggest that our method can solve planning problems in 100 dimensional state spaces and obtain computational speedups of 400$\times$ over related trajectory optimization methods.

1 Introduction

A key challenge in robotics is to robustly complete tasks such as navigation and manipulation in the presence of uncertainty. One way to deal with uncertainty is to explicitly gather information from sensing to reduce uncertainty about aspects of the robot and the environment that are critical for completion of a task. The problem of computing motion plans that optimally perform information gathering actions as necessary can be formalized as a Partially Observable Markov Decision Process (POMDP) [20], which is defined over the space of probability distributions of the state space, also referred to as the *belief* space.

Computing globally optimal solutions for the POMDP problem is known to be computationally intractable [7]. As a result, recent work including, but not limited to, [24] [29] [15] [27] [35] [19] [11] [26], has focused on solving this problem in Gaussian belief spaces, where beliefs are concisely parameterized as Gaussian distributions and are propagated using a Bayesian filter such as an extended Kalman filter (EKF). This body

of work has lead to the development of novel methods for solving the problem. Unfortunately, the computational effort involved in computing plans using these methods is a bottleneck when there is considerable uncertainty during execution and it is potentially necessary to re-plan at every time step in a receding horizon control context [27].

In this work, we formulate the planning problem in Gaussian belief spaces as a trajectory optimization problem. We show that it is possible to compute locally optimal plans by optimizing over just the control inputs and mean states as opposed to optimizing over control inputs, mean states, and covariances [27] [26]. We refer to this formulation as covariance-free trajectory optimization in Gaussian belief spaces. Excluding the covariance from the optimization has two major implications – (i) the dimension of the optimization problem is now linear in the state dimension instead of quadratic, thereby leading to considerable computational speedups, and (ii) this eliminates the constraint that ensures that the covariances at all time steps are consistent with the Bayesian belief update, which is nonlinear by virtue of matrix operations such as the matrix inverse involved in the update.

We accomplish covariance-free trajectory optimization by taking advantage of two recent advances in numerical optimal control. The first is the use of *reverse mode* automatic differentiation, which is a technique for efficiently evaluating derivatives of scalar-valued computer represented functions up to machine precision [16] [2]. The second is the development of efficient receding horizon convex solvers [13] [14] that exploit a priori knowledge about the temporal structure of the problem to generate efficient solver code specific to a problem instance. We show that a combination of both techniques is important for achieving a computational complexity of $O(n^3T)$ per optimization step, where n is the state space dimension and T is the number of time steps.

We evaluate our approach in simulation vis-à-vis other trajectory optimization methods, including dynamic programming [35] and trajectory optimization with covariances [27] [26]. We consider planning problems in a variety of domains including planning manipulator motions under uncertainty [34], estimating parameters of inaccurate dynamical systems [38], and active simultaneous localization and mapping (active SLAM) [19]. Our experiments suggest that by not including the covariance in the optimization, it is possible to compute plans in Gaussian belief spaces for 100 dimensional state spaces and we have obtained computational speedups of 400$\times$ over related methods.

2 Related Work

Planning in the belief space for many robotic tasks is naturally defined over continuous state, action, and observation spaces. In this setting, prior work has tackled the planning problem from a number of different angles:

(1) Point-based value iteration methods [28] [23] select a limited set of representative belief points and iteratively apply value updates to those points to compute a control policy over belief space.
(2) Simulation-based methods [32] [33] generate a few potential plans and select a plan that optimizes a given metric such as information gain.
(3) Regression-based planning [21] uses logical representations of the belief state to compute plans that achieve a desired goal.
(4) Sampling-based methods [29] [17] [6] [18] [1] use randomized exploration strategies to explore the belief space in search of an optimal plan.

(5) Policy search methods [9] [11] directly optimize parameters of a control policy using approximate inference in Gaussian belief spaces.
(6) Trajectory optimization methods [24] [15] [27] [35] [36] [19] [22] [26] compute locally optimal trajectories (and policies, if applicable) that trade off actuation and sensing actions to maximize information gain over a finite horizon.

Gaussian belief space planning focuses on Gaussian parameterizations of the belief in terms of the mean and covariance. Of the aforementioned categories, specialized methods in categories (4), (5), and (6), have been developed for computing locally optimal plans in Gaussian belief spaces. Trajectory optimization methods (category (6)) can be further classified into two categories (Fig. 1):

- Dynamic programming [4] in Gaussian belief spaces: Examples include using linear quadratic regulator (LQR) [27], differential dynamic programming (DDP) [15], and iterative LQG [35]. In addition to computing a locally optimal trajectory, these methods also compute an associated control policy.
- Direct optimization methods [5] in Gaussian belief spaces: Examples include optimizing just over controls (also known as shooting) [24] [19] or optimizing over controls, mean states, and covariances (also known as collocation) [27] [26]. These methods compute a locally optimal open-loop trajectory.

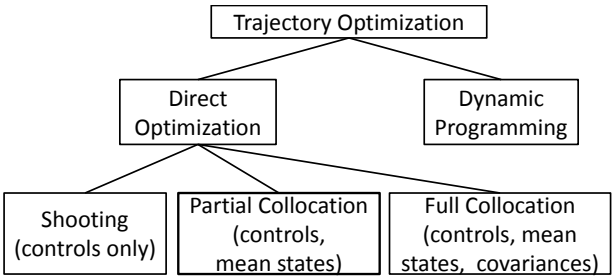


Fig. 1. Taxonomy of trajectory optimization methods for Gaussian belief space planning. Our covariance-free optimization formulation optimizes over controls only (shooting) or both controls and mean states (partial collocation).

Our covariance-free trajectory optimization method lies in the sub-category of direct trajectory optimization methods that optimize over controls and mean states only. Prior work has explored shooting methods by directly optimizing over the sequence of control inputs using control sampling [24], which is not desirable in the case of continuous action spaces common in robotic tasks. Indelman et al. [19] use gradient descent to optimize over the sequence of controls but use finite differences to compute the gradient of the objective function. As we will show in Sec. 5, use of finite differences over long trajectories leads to poorly conditioned gradients, leading to slow convergence. In this work, we compute exact gradients using automatic differentiation (Sec. 4.2).

Prior work has also explored the possibility of optimizing over mean states and control inputs for planning under uncertainty. However, the formulation of Vitus et al. [37] cannot account for state- and control-dependent noise, which is important for belief space planning. Kontitsis et al. [22] use covariance matrix adaption (CMA) based optimization to optimize the objective subject to constraints on the evolution of the mean state but this sampling-based optimization method is computationally expensive.

3 Gaussian Belief Space Planning: Preliminaries and Notation

Let $\mathbf{x} = [\mathbf{x}^R, \mathbf{x}^{\mathcal{O}}]^\intercal \in \mathbb{R}^{n_\mathbf{x}}$ be the system state consisting of the state $\mathbf{x}^R$ of the robot and the state $\mathbf{x}^{\mathcal{O}}$ of relevant objects in the environment. Let $\mathbf{u} = [\mathbf{u}^R, \mathbf{u}^{\mathcal{O}}]^\intercal \in \mathbb{R}^{n_\mathbf{u}}$ denote the combined control input applied to the system and $\mathbf{z} = [\mathbf{z}^R, \mathbf{z}^{\mathcal{O}}]^\intercal \in \mathbb{R}^{n_\mathbf{z}}$ be the vector of measurements obtained about the system state using sensors. We are given stochastic dynamics and measurement models given by nonlinear differentiable functions $\mathbf{f}$ and $\mathbf{h}$:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{q}_t), \quad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, I), \tag{1}$$

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{r}_t), \qquad \mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, I), \tag{2}$$

where $\mathbf{q}_t$ is the Gaussian dynamics noise and $\mathbf{r}_t$ is Gaussian measurement noise.

We consider a Gaussian parameterization of the belief $(\hat{\mathbf{x}}_t, \Sigma_t)$ consisting of the mean state $\hat{\mathbf{x}}_t$ and the covariance Σ_t. We assume that the initial belief $(\hat{\mathbf{x}}_0, \Sigma_0)$ is given. Given a current belief $(\hat{\mathbf{x}}_t, \Sigma_t)$, a control input $\mathbf{u}_t$, and a measurement $\mathbf{z}_{t+1}$, the belief state evolves using a Bayesian filter such as an extended Kalman filter (EKF), according to a stochastic process given by [35]:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}) - K_t(\mathbf{z}_{t+1} - \mathbf{h}(\mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}), \mathbf{0}), \tag{3a}$$

$$\Sigma_{t+1} = (I - K_t H_t)\Sigma_{t+1}^-, \tag{3b}$$

$$A_t = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}),\ Q_t = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}),\ \Sigma_{t+1}^- = A_t \Sigma_t A_t^\intercal + Q_t Q_t^\intercal, \tag{3c}$$

$$H_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{t+1}, \mathbf{0}),\ \ R_t = \frac{\partial \mathbf{h}}{\partial \mathbf{r}}(\hat{\mathbf{x}}_{t+1}, \mathbf{0}),\ \ K_t = \Sigma_{t+1}^- H_t^\intercal (H_t \Sigma_{t+1}^- H_t^\intercal + R_t R_t^\intercal)^{-1}. \tag{3d}$$

We consider discrete-time Gaussian belief space planning problems that are solved over a finite horizon T in which a robot performs information gathering actions as necessary, to minimize uncertainty during task execution. For example, in localization, a robot seeks to reduce the variance of its state, and in parameter estimation, the robot seeks to reduce the variance of its model parameters. In general, objectives that are functions of means, covariances, and control inputs can be considered.

Depending on the optimal control method used, the objective is to either compute a sequence of controls $\mathbf{u}_t$ or a control policy $\mathbf{u}_t = \pi_t(\hat{\mathbf{x}}_t, \Sigma_t)$ for all $0 \leq t < T$ that minimizes the objective:

$$\mathop{\mathrm{E}}_{\mathbf{z}_{1:T}}\left[c_T(\hat{\mathbf{x}}_T, \Sigma_T) + \sum_{t=0}^{T-1} c_t(\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t)\right], \tag{4}$$

where c_T and c_t are given immediate cost functions and the expectation is taken over the stochastic measurements. The planning problem can be illustrated as a graphical model as shown in Fig. 2. The solid lines in the graphical model indicate relationships between the means $\hat{\mathbf{x}}_{0:T}$, covariances $\Sigma_{0:T}$, and controls $\mathbf{u}_{0:T-1}$. The dashed lines indicate the dependence of the immediate cost functions $c_{0:T}$ and the different variables.

In our experiments, we encode the objective of minimizing the uncertainty while penalizing the control effort by using cost functions of the form:

$$c_t(\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t) = \mathrm{tr}(M_t \Sigma_t) + \mathbf{u}_t^\intercal N_t \mathbf{u}_t, \quad c_T(\hat{\mathbf{x}}_T, \Sigma_T) = \mathrm{tr}(M_T \Sigma_T) \tag{5}$$

where minimizing the trace of the covariance Σ_t minimizes the uncertainty and matrices M_t and N_t are positive semi-definite cost matrices. However, the cost functions are general enough to include additional problem-specific terms.

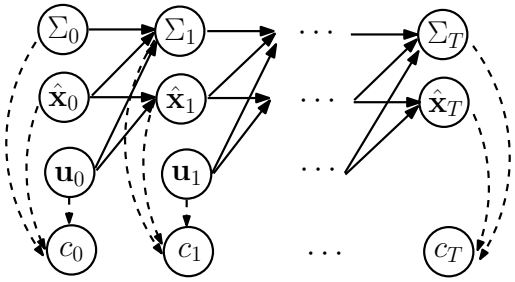


Fig. 2. Graphical model for finite horizon Gaussian belief space planning. Given the initial belief $(\hat{\mathbf{x}}_0, \Sigma_0)$, all subsequent $\hat{\mathbf{x}}_t$ and Σ_t are functions of $\hat{\mathbf{x}}_0$, Σ_0, and the sequence of controls $\mathbf{u}_{0:t-1}$. The immediate cost functions c_t and c_T are functions of the controls, mean states, and covariances.

4 Covariance-Free Trajectory Optimization

4.1 Formulation

Direct methods for trajectory optimization [5] formulate the planning problem as a nonlinear trajectory optimization problem. In this setting, the stochastic, partially-observed control problem described in Sec. 3 is replaced by a deterministic optimal control problem, which is computationally tractable. As first pointed out by Platt et al. [27], this can be accomplished by making the assumption that the maximum likelihood observation is obtained at each time step, i.e., $\mathbf{z}_t = \mathbf{h}(\hat{\mathbf{x}}_t, \mathbf{0})$. This eliminates the stochasticity in the evolution of the mean state (Eq. (3a)), converting the problem into a deterministic optimal control problem. The goal is to compute a sequence of controls $\mathbf{u}_{0:T-1}$ that minimizes the objective $c_T(\hat{\mathbf{x}}_T, \Sigma_T) + \sum_{t=0}^{T-1} c_t(\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t)$, without the expectation term appearing in the original objective (Eq. (4)). This class of methods computes an open-loop sequence of controls $\mathbf{u}_{0:T-1}$. During execution, we follow the model predictive control paradigm [8] of repeatedly re-planning to account for the current observation. However, the key is to be able to re-plan sufficiently fast.

Since all subsequent $\hat{\mathbf{x}}_t$ and Σ_t are functions of $\hat{\mathbf{x}}_0$, Σ_0, and the sequence of controls $\mathbf{u}_{0:t-1}$, as shown in Fig. 2, it is possible to formulate the optimization problem in terms of one or more of controls, mean states, and covariances. The three possible formulations are shown in Table 1. Here, $\mathcal{C}(\cdot)$ represents the objective expressed in Eqs. 4, 5 in terms of the optimization variables, $\hat{\mathbf{x}}_{\text{target}}$ represents the desired mean target state, and $\mathcal{X}_{\text{feasible}}$ and $\mathcal{U}_{\text{feasible}}$ are sets of feasible states and control inputs, respectively.

Of these, the shooting and partial collocation formulations constitute the covariance-free optimization formulations considered in this work and are described below:

(i) **Shooting**: The objective $\mathcal{C}(\hat{\mathbf{x}}_0, \Sigma_0, \mathbf{u}_{0:T-1})$ is a nonlinear function of the initial belief $(\hat{\mathbf{x}}_0, \Sigma_0)$ and the sequence of controls $\mathbf{u}_{0:T-1}$ since the immediate cost functions $c_t(\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t)$ are dependent on the previously applied controls (Fig. 2). For planning problems, it is often desired that the mean state at the final time step is at a desired target, i.e., $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}}$. However, since $\hat{\mathbf{x}}_T$ is dependent on the sequence of controls, the optimization contains a possibly nonlinear constraint $\tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:T-1}, \mathbf{0}) = \hat{\mathbf{x}}_{\text{target}}$, where $\tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:T-1}, \mathbf{0}) = \mathbf{f}(\mathbf{f}(\ldots(\mathbf{f}(\hat{\mathbf{x}}_0, \mathbf{u}_0), \mathbf{u}_1), \ldots), \mathbf{u}_{T-1})$ computes the state at time step T. A similar iterative nonlinear constraint arises for restricting the states $\hat{\mathbf{x}}_{0:T}$ to the set of feasible states $\mathcal{X}_{\text{feasible}}$. In practice, these nonlinear constraints are typically added as costs to the optimization objective. The optimization problem has dimension $n_{\mathbf{u}}T$.

(ii) **Partial Collocation**: A second formulation considers optimizing over the controls $\mathbf{u}_{0:T-1}$ and mean states $\hat{\mathbf{x}}_{0:T}$. Similar to shooting, the objective $\mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_0, \mathbf{u}_{0:T-1})$

Covariance-free Trajectory Optimization		
Shooting	Partial Collocation	Full Collocation
$\min_{\mathbf{u}_{0:T-1}} \; \mathcal{C}(\hat{\mathbf{x}}_0, \Sigma_0, \mathbf{u}_{0:T-1})$ $\text{s.t} \quad \tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:T-1}, \mathbf{0}) = \hat{\mathbf{x}}_{\text{target}}$ $\tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:t-1}, \mathbf{0}) \in \mathcal{X}_{\text{feasible}}$ $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$	$\min_{\substack{\mathbf{u}_{0:T-1} \\ \hat{\mathbf{x}}_{0:T}}} \; \mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_0, \mathbf{u}_{0:T-1})$ $\text{s.t} \quad \hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0})$ $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}},$ $\hat{\mathbf{x}}_t \in \mathcal{X}_{\text{feasible}},$ $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$	$\min_{\substack{\mathbf{u}_{0:T-1} \\ \hat{\mathbf{x}}_{0:T} \\ \Sigma_{0:T}}} \; \mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_{0:T}, \mathbf{u}_{0:T-1})$ $\text{s.t} \quad \hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}),$ $\Sigma_{t+1} = (I - K_t H_t)\Sigma_{t+1}^-,$ $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}},$ $\hat{\mathbf{x}}_t \in \mathcal{X}_{\text{feasible}},$ $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$

Table 1. Three possible formulations for direct trajectory optimization in Gaussian belief spaces. Our covariance-free optimization formulation either optimizes only over controls (shooting) or over controls and mean states (partial collocation).

is nonlinear and dependent on the initial covariance Σ_0 and the mean states and controls (Fig. 2). In this formulation, the final target constraint $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}}$ is directly included in the optimization as $\hat{\mathbf{x}}_T$ is included in the optimization. Similarly, bounds on the controls and mean states are directly included in the optimization. The optimization problem has dimension $(n_{\mathbf{x}} + n_{\mathbf{u}})T$.

Comparison with Full Collocation: Prior work has typically considered full collocation that optimizes over means, controls, and covariances [27] [26]. The main advantage of full collocation is that the objective $\mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_{0:T}, \mathbf{u}_{0:T-1})$ is only dependent on local variables at each time step. The objective considered in this work turns out to be quadratic, which is suitable for numerical optimization methods. Also, constraints on the mean states and controls are directly included in the optimization formulation.

However, since the covariance matrices Σ_t are included in the optimization, it is important to ensure that the covariance matrices are consistent with the evolution of the belief state, as given in Eq. (3b). This equality constraint is nonlinear because of the presence of matrix operations, particularly the matrix inverse used to compute the Kalman gain (Eq. (3d)), and is difficult to satisfy in a numerical optimization procedure. In addition, the optimization dimension increases because of the inclusion of the covariance matrices. The optimization problem has dimension $(n_{\mathbf{x}}(n_{\mathbf{x}}+1)/2 + n_{\mathbf{u}})T$, where we exploit the symmetry in Σ_t to only store the lower diagonal half [27] [26]. Additional care, such as using the square root of the covariance matrix instead of the covariance itself [26], also needs to be taken to ensure that the covariances remain positive semi-definite during the course of the optimization.

4.2 Tools for Covariance-Free Trajectory Optimization

We rely on two key advances in numerical optimal control – (i) automatic differentiation to accurately compute gradients of the nonlinear objective, and (ii) state of the art convex solvers that are used in a sequential quadratic programming framework for optimizing the nonlinear objective subject to constraints.

Automatic Differentiation: For covariance-free trajectory optimization, the objective $\mathcal{C}(\hat{\mathbf{x}}_0, \Sigma_0, \mathbf{u}_{0:T-1})$ is a nonlinear function of the initial belief $(\hat{\mathbf{x}}_0, \Sigma_0)$ and the sequence of controls $\mathbf{u}_{0:T-1}$. If numerical finite differences are used to compute the gradient of the nonlinear objective, then the gradients are poorly conditioned. For instance,

a small change in the control input $\mathbf{u}_0$ will often have a dramatically large effect on the objective as compared to a small change in $\mathbf{u}_{T-1}$. In Sec. 5, we show that using gradients computed using finite differences indeed lead to slower convergence. One alternative would be to hand-compute the analytical expressions of first and preferably second-order derivatives. However, this is difficult for complex functions such as the matrix inverse involved in the belief dynamics (Eq. (3)). Special cases also need to be taken into account to correctly handle singularities in computation.

Automatic differentiation (AD) [16] is a technique for evaluating derivatives of computer represented functions and can deliver directional derivatives, up to machine precision, of arbitrary computer-represented functions. Automatic differentiation has resulted in the development of efficient numerical optimal control methods [12] and recent work on optimization on manifolds for robotics applications [31]. We note that automatic differentiation is different from symbolic differentiation, which directly operates on functions represented in a special purpose symbolic language. We refer the reader to Griewank et al. for a comparative study [16].

In our case, since the objective is scalar valued, we use the *reverse* mode for differentiation that offers considerable savings to be made by exploiting the structure, sparsity, and symmetry of the Jacobian. Several computational tools have been developed to facilitate reverse mode automatic differentiation. Examples include Theano [3], ADOL-C [16], and CasADi [2]. We use CasADi since it also supports matrix-valued atomic operations. We only compute the gradients using automatic differentiation and not the complete Hessian of the objective. Even though it is possible to compute the entire Hessian, it is computationally very expensive and does not scale well to larger problems. We use the symmetric rank 1 (SR1) update method to update the Hessian using the gradients computed using automatic differentiation [25].

An important consequence of using reverse-mode automatic differentiation is what is known as the *cheap gradient* principle which states that the complexity of computing the gradient of a scalar-valued function is bounded above by a small constant factor times the complexity of evaluating the function itself [16]. This fact will be used in the analysis of the running time of covariance-free trajectory optimization methods.

State of the art Convex Solvers: We use sequential quadratic programming (SQP) to locally optimize the non-convex, constrained optimization problem that results from the covariance-free formulation. SQP [25] optimizes problems in parameter θ of the form $\min_\theta \mathcal{C}(\theta)$ subject to constraints. One repeatedly constructs a quadratic program (quadratic objective and linear constraints) that locally approximates the original problem around the current solution θ. Then one solves the quadratic program to compute a step $\Delta\theta$ that make progress on the original problem. Two necessary ingredients in a SQP implementation are trust regions and merit functions. A trust region constrains θ in each subproblem to the region where the approximation is valid. The trust region is adaptively changed based on the merit function, which has the form $f_\mu(\theta) = f(\theta) + \mu \cdot \text{ConstraintViolation}(\theta)$. Here, μ is a given penalty parameter that penalizes violations of nonlinear constraints, and it ensures that the steps taken by the algorithm make progress on both the cost function and the constraints. The optimization algorithm solves a series of problems $\min_\theta f_{\mu_0}(\theta), \min_\theta f_{\mu_1}(\theta), \ldots, \min_\theta f_{\mu_n}(\theta)$ for $\mu_0 < \mu_1 < \cdots < \mu_n$ where the penalty parameter μ is sequentially increased in an

outer loop. We used sequential quadratic programming (SQP) with ℓ_1 penalties [25], also used by Schulman et al. [30] for robot motion planning in state space.

At the core of the SQP method is a QP solver. We efficiently solve the underlying QPs using a numerical optimization code generation framework called FORCES [14]. FORCES generates code for solving QPs that is based on the interior-point method and is specialized for convex multistage problems such as trajectory optimization. Automatic code generation for convex solvers has gained popularity since it is able to exploit the fact that all problem dimensions and the structure of the problem is known a priori. This permits generation of highly customized and fast solver code that solves instances of a particular problem. We use this solver for all our experiments, including for the full collocation formulation.

An important consequence of using the FORCES code generation framework is that the complexity of solving a QP in m optimization variables and T time steps is $O(m^3T)$, instead of worst case complexity of $O(m^3T^3)$ that is associated with a condensing procedure used to reduce the number of optimization variables in QP solvers [13]. This speedup is obtained from exploitation of the known temporal structure of the problem. This fact will be used in the analysis of the running time of covariance-free trajectory optimization methods.

4.3 Running Time Analysis

For the sake of analysis, we assume that the dimension $n_{\mathbf{x}}$ of the state space, $n_{\mathbf{u}}$ of the control input space, and $n_{\mathbf{z}}$ of the measurements are $O(n)$. As a result, the covariance matrix has dimension $O(n^2)$. Let T be the number of time steps in the trajectory being optimized, also referred to as the trajectory horizon. We analyze the complexity of each step of the optimization procedure since the number of optimization steps required for convergence cannot be expressed in terms of n or T.

Computing the objective (Eq. (5)) requires the propagation of the beliefs along a trajectory according to Eq. (3). Since the Bayesian update requires matrix operations such as multiplication operations and matrix inversion of matrices of order $O(n^2)$, the complexity of each update step is $O(n^3)$. As a result, the overall complexity of computing the objective for all time steps is $O(n^3T)$.

We analyze the complexity of each step of the optimization for covariance-free optimization. The complexity of computing the gradients using reverse-mode automatic differentiation is $O(n^3T)$ using the cheap gradient principle. The shooting and partial collocation formulations have $n_{\mathbf{u}}T$ and $(n_{\mathbf{x}}+n_{\mathbf{u}})T$ optimization variables, respectively. The complexity of solving each QP using the FORCES code generation framework is $O(n^3T)$. This is the same order of complexity as computing the objective, and hence is a lower bound on the planning complexity for trajectory optimization methods.

We note that the locally optimizing the full collocation formulation is of the order of $O(n^6T)$, since it contains $(n_{\mathbf{x}}(n_{\mathbf{x}}+1)/2+n_{\mathbf{u}})T$ or $O(n^2)T$ optimization variables. In practice, however, the nonlinear equality constraint that ensures that the beliefs are consistent with the belief update (Eq. (3b)) requires introduction of additional slack variables in the SQP method with ℓ_1 penalties, which results in a large constant factor. We also note that the complexity of dynamic programming methods is $O(n^6T)$ for iLQG [35], which can be reduced to $O(n^4T)$ if the immediate cost functions are assumed to be quadratic in the mean and linear in the variance, as is the case in our work.

5 Experiments

We present experimental results in simulation for Gaussian belief space planning in a variety of domains involving uncertainty, including planning manipulator motions, estimating model parameters for uncertain dynamical models, and active simultaneous localization and mapping (active SLAM). All execution times are based on a C++ implementation of our method running on a single 3.5 GHz Intel i7 processor core.

5.1 6-DOF Manipulator Planning with Kinematics

In this experiment, we consider a 6-DOF A465 CRS robot arm moving in a 3D environment [34], as shown in Fig. 3. The state $\mathbf{x}_t = [\theta_t^1, \ldots, \theta_t^6]^\intercal$ of the robot is a 6D vector consisting of the joint angles. The control input $\mathbf{u}_t = [\omega_t^1, \ldots, \omega_t^6]^\intercal$ is a 6D vector consisting of the angular speeds at each of the joints, corrupted by dynamics noise $\mathbf{q}_t = [q_t^1, \ldots, q_t^6]^\intercal \sim \mathcal{N}(\mathbf{0}, I)$. The robot receives feedback from an overhead stereo camera setup that measures the position of the end effector $\mathbf{p}_t = [x_t, y_t, z_t]^\intercal$. Each camera c_i has a known location $[x^i, y^i, z^i]^\intercal$ and unit focal distance. The measurement $\mathbf{z}_t$ is a 4D vector consisting of the pixel coordinates of the end effector on the imaging planes of both cameras, corrupted by Gaussian measurement noise $\mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, I)$. This results in the following stochastic dynamics and measurement models:

$$\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{q}_t) = \mathbf{x}_t + \tau(\mathbf{u}_t + \alpha \mathbf{q}_t), \tag{6a}$$

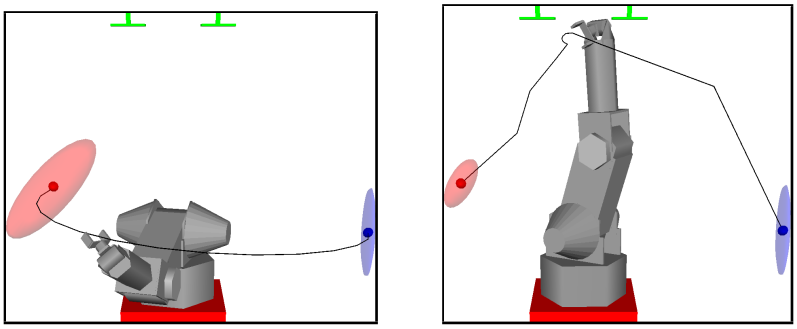
$$\mathbf{h}(\mathbf{x}_t, \mathbf{r}_t) = \left[\frac{(x_t - x^1)}{(y_t - y^1)}, \frac{(z_t - z^1)}{(y_t - y^1)}, \frac{(x_t - x^2)}{(y_t - y^2)}, \frac{(z_t - z^2)}{(y_t - y^2)}\right]^\intercal + \beta \mathbf{r}_t, \tag{6b}$$

where τ is the duration of each time step, and α and β are scaling constants for the dynamics and measurement noise terms. It is important to note that the signal to noise ratio, and hence the reliability of measurements, increases with a decrease in the distance to the stereo camera setup.

The objective is to move the robot end effector from an initial position to a target position while minimizing uncertainty in the end effector position. Let $\mathcal{N}(\hat{\mathbf{p}}_T, \Sigma(\hat{\mathbf{p}}_T))$ be the uncertainty in the end effector position with mean $\hat{\mathbf{p}}_T$ and covariance $\Sigma(\hat{\mathbf{p}}_T) \in \mathbb{R}^{3\times 3}$. We approximate $\Sigma(\hat{\mathbf{p}}_T)$ as $\Sigma(\hat{\mathbf{p}}_T) = J\Sigma_T J^\intercal$, where Σ_T is the covariance in the state at the final time step and $J = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_T)$ is the Jacobian of the forward kinematics function $\mathbf{g}(\mathbf{x}) = \mathbf{p}$ evaluated at the final mean state $\hat{\mathbf{x}}_T$. The objective is to minimize $\mathrm{tr}(M_T \Sigma(\hat{\mathbf{p}}_T)) + \sum_{t=0}^{T-1} \mathbf{u}_t^\intercal N_t \mathbf{u}_t$, which optimizes the trade-off between minimizing uncertainty and the cumulative control effort, for given matrices M_T and N_t, $0 \leq t < T$.

Experiments: Fig. 3(a) shows a baseline interpolated trajectory between the start position and target position. Fig. 3(b) shows an instance of a trajectory computed using belief space planning using covariance-free partial collocation. The plan is able to infer that it is advantageous to get closer to the camera to get more reliable measurements before heading to the target to reduce uncertainty at the final time step.

We compared the performance of the three direct optimization formulations on this problem. Dynamic programming methods such as iLQG [34] are unable to compute a solution for this problem due to the inability to enforce joint angle constraints. To evaluate the performance of these methods, we considered 100 random initial start positions while keeping the mean target position constant for the end effector. All trajectories have a horizon of $T = 15$ time steps. We compared the different methods on the basis



(a) (b)

Fig. 3. 6-DOF Manipulator: The objective is to move the end effector from the initial position (blue sphere) to the final mean target position (red sphere) while minimizing the uncertainty in its position. The robot receives feedback from a stereo camera setup mounted overhead (shown in green). The trajectory of the end effector is shown in black. The initial and final variance is shown as a blue and red ellipse, respectively. (a) A naïve trajectory obtained by interpolating between the initial and target states (joint angles) results in considerable uncertainty at the final time step. (b) A belief space plan infers that it is advantageous to get more reliable measurements by getting closer to the stereo camera setup, hence resulting in reduced uncertainty at the target.

of computation time and by how much the planning was able to improve the considered objective when compared to a baseline interpolated trajectory between the start and target states (Fig. 3(a)). The results are summarized in Table 2. The experiments show a significant speed up of up to 200× is obtained by excluding the covariance from the optimization. The partial collocation formulation performs best, in part because the dynamics constraint $\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0})$ is a linear constraint. In the shooting method, we include an additional cost term to penalize violation of the constraint that the end effector should be at a desired target position. This cost term conflicts with the original objective, leading to lower quality trajectories.

	Opt. vars	Time (s)	Improvement over baseline (%)
Shooting	84	0.046 ± 0.014	29.7 ± 5.1
Partial Coll. (auto diff.)	174	0.024 ± 0.004	71.6 ± 1.7
Partial Coll. (finite diff.)	174	0.902 ± 0.184	69.5 ± 2.4
Full Coll.	405	6.518 ± 0.510	56.8 ± 2.3

Table 2. Comparison of trajectory optimization methods on the 6-DOF manipulator scenario in terms of computation time and improvement over a baseline interpolated trajectory.

We also compared the effect of using automatic differentiation (Sec. 4.2) versus using numerical finite differences for the partial collocation formulation. Using finite differences leads to poorly conditioned gradients, which leads to slower convergence. In our experiments, we observed slowdowns of up to 40× as compared to implementations that used automatic differentiation to compute gradients.

5.2 Parameter Estimation for Two-Link Pendulum with Dynamics

In this experiment, we consider a two-link pendulum [10] actuated at both joints. However, the lengths $[l^1, l^2]^\intercal$ and masses $[m^1, m^2]^\intercal$ of the pendulum are not exactly known.

We follow the method of Webb et al. [38] to estimate these uncertain parameters using Gaussian belief space planning by considering an augmented state consisting of the pendulum state and the parameters. The objective is to infer the model parameters $\{m^1 = m^2 = 0.5\text{kg},\ l^1 = l^2 = 0.5\text{m}\}$, given noisy measurements of the end-effector position of the pendulum and the joint velocities. While random exploration can be used to solve this problem, we demonstrate that Gaussian belief space planning can be used to intelligently explore by acquiring information from sensing and hence converge faster to the actual model parameters.

The system state $\mathbf{x}_t = [\theta_t^1, \theta_t^2, \dot{\theta}_t^1, \dot{\theta}_t^2, l_t^1, l_t^2, m_t^1, m_t^2]^\intercal$ of the robot is a 8D vector consisting of the joint angles, the angular velocities of the joints, and the four parameters that need to be estimated. The control input $\mathbf{u}_t = [\mu_t^1, \mu_t^2]^\intercal$ is a 2D vector consisting of the motor torques. The robot state is also corrupted by Gaussian noise $\mathbf{q}_t = [q_t^1, \ldots, q_t^4, \mathbf{0}_{4\times 1}]^\intercal \sim \mathcal{N}(\mathbf{0}, I)$. This yields the following nonlinear dynamics model:

$$\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{q}_t) = \mathbf{x}_t + \tau[\dot{\theta}_t^1, \dot{\theta}_t^2, \ddot{\theta}_t^1, \ddot{\theta}_t^2, \mathbf{0}_{4\times 1}]^\intercal + \alpha \mathbf{q}_t, \quad \text{where} \tag{7a}$$

$$\begin{bmatrix} \ddot{\theta}_t^1 \\ \ddot{\theta}_t^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} m_t^1 l_t^1 l_t^1 + m_t^2 l_t^1 l_t^1 & \frac{1}{2} m_t^2 l_t^2 l_t^1 \cos(\theta_t^1 - \theta_t^2) \\ \frac{1}{2} l_t^1 l_t^2 m_t^2 \cos(\theta_t^1 - \theta_t^2) & \frac{1}{3} m_t^2 l_t^2 l_t^2 \end{bmatrix}^{-1} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \tag{7b}$$

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -l_t^1 (\frac{1}{2} m_t^2 l_t^2 \dot{\theta}_t^2 \dot{\theta}_t^2 \sin(\theta_t^1 - \theta_t^2) - g \sin \theta_t^1 (\frac{1}{2} m_t^1 + m_t^2)) + \mu_t^1 \\ \frac{1}{2} m_t^2 l_t^2 (l_t^1 \dot{\theta}_t^1 \dot{\theta}_t^1 \sin(\theta_t^1 - \theta_t^2) + g \sin \theta_t^2) + \mu_t^2 \end{bmatrix}, \tag{7c}$$

where τ is the duration of the time step, α is the noise scaling factor, and $g = 9.82\text{m/s}^2$ is the gravitational constant. In our implementation, we use Runge-Kutta (RK4) integration of the dynamics for numerical stability.

The robot obtains noisy measurements of the position of the end effector and the angular velocities at each joint. The measurement $\mathbf{z}_t$ is a 4D vector, corrupted by measurment noise $\mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, I)$ that is related to the state $\mathbf{x}_t$ according to

$$\mathbf{h}(\mathbf{x}_t, \mathbf{r}_t) = [l_t^1 \cos \theta_t^1 + l_t^2 \cos \theta_t^2, l_t^1 \sin \theta_t^1 + l_t^2 \sin \theta_t^2, \dot{\theta}_t^1, \dot{\theta}_t^2]^\intercal + \beta \mathbf{r}_t, \tag{8}$$

where β is the measurement noise scaling factor.

Experiments: Fig. 4 shows the performance of different optimal control methods on convergence of the parameter m^1 as compared to execution of a random sequence of controls. We note that the parameter values are not updated as part of the planning process. We execute the first control for each computed plan in a model predictive control fashion and then update the parameters using the full Bayesian update of the belief that incorporates the current simulated observation (Eq. (3)). Belief space planning is able to explore by intelligently gathering information as required for faster convergence in terms of the number of execution steps required. We consider a trajectory with $T = 15$ time steps. Due to space limitations, we only show convergence results for one parameter m^1. Similar results were obtained for the second mass parameter. However, the length parameters l^1 and l^2 converge faster to the correct values since they explicitly occur in the measurement model.

The performance of optimal control methods is also compared in Fig. 4, as averaged over 100 runs and 300 execution time steps. In terms of convergence, all the methods require roughly the same number of execution steps for converging to a reasonable estimate. However, there is a considerable difference in execution times. The shooting

	Opt. vars	Time(s)
Shooting	28	0.004 ± 0.002
Partial Coll.	148	0.152 ± 0.109
Full Coll.	570	1.595 ± 0.068
iLQG	–	0.153 ± 0.017

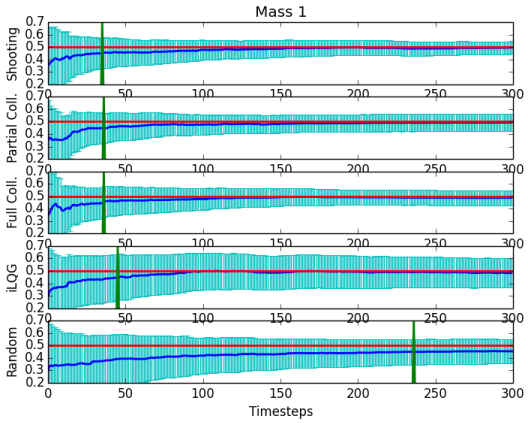


Fig. 4. Parameter Estimation: (left) The performance of different optimal control methods on the parameter estimation scenario. (right) The convergence of the parameter m^1 to the true value of 0.5 kg (red) for different optimal control methods. The value of the parameter m_t^1 is plotted over time (blue) and three standard deviations of the uncertainty in the parameter value is shown in cyan. The timestep where the value is within 10% of the true value is marked in green.

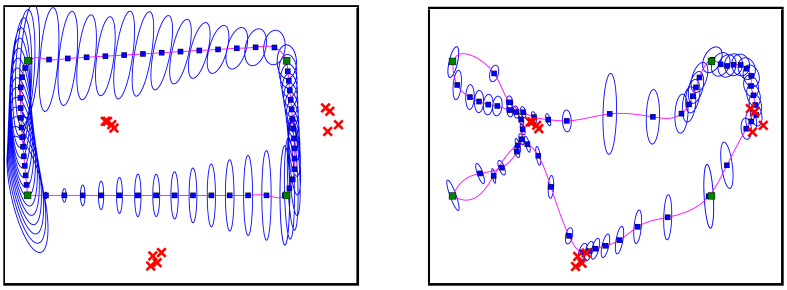
formulation is up to $400\times$ faster than the full collocation formulation with the covariance in the optimization. We also compare with a state of the art iLQG implementation [36] and found that iLQG was $10\times$ faster than full collocation but convergence of iLQG to the true parameter values was slower than the covariance-free formulations.

5.3 Active SLAM

Active simultaneous localization and mapping (active SLAM) [19] [32] [33] aims to compute plans for a mobile robot to explore the environment (represented in terms of landmarks) such that the uncertainty about the environment and the robot state are simultaneously minimized in a SLAM framework.

We consider a car-like robot with state $\mathbf{x}_t^R = [x_t, y_t, \theta_t]^\intercal$ consisting of the car position $[x_t, y_t]^\intercal$ and heading angle θ. The control inputs $\mathbf{u}_t = [v_t, \phi_t]^\intercal$ consist of the velocity v_t and steering angle ϕ_t, corrupted by Gaussian noise $\mathbf{q}_t = [q_t^1, q_t^2]^\intercal$. Let W be the length of the wheelbase of the robot. The state of the environment $\mathbf{x}^{\mathcal{O}} \in \mathbb{R}^{2\cdot L}$ is a vector consisting of L landmark positions $[x^i, y^i]^\intercal,\ i \in \{1, ..., L\}$. The combined system state is then given by $\mathbf{x}_t = [\mathbf{x}_t^R, \mathbf{x}^{\mathcal{O}}]^\intercal$. The observation function $z \in \mathbb{R}^{2\cdot L}$ consists of the distance and heading measurements from the current state of the robot relative to each landmark. Following the framework of standard EKF-SLAM, this gives us the following stochastic dynamics and measurements models:

$$\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{q}_t) = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ \mathbf{x}_{\mathcal{O}} \end{bmatrix} + \begin{bmatrix} (v_t + q_t^1)\cos(\theta_t + \phi_t + q_t^2) \\ (v_t + q_t^1)\sin(\theta_t + \phi_t + q_t^2) \\ (v_t + q_t^1)\tan(\phi_t + q_t^2)/W \\ \mathbf{0}_{2\cdot L \times 1} \end{bmatrix}, \quad \mathbf{h}(\mathbf{x}_t, \mathbf{r}_t) = \begin{bmatrix} \sqrt{(x_t - x_t^1)^2 + (y_t - y_t^1)^2} \\ \tan^{-1}(\frac{y_t - y_t^1}{x_t - x_t^1}) - \theta_t \\ \vdots \\ \sqrt{(x_t - x_t^L)^2 + (y_t - y_t^L)^2} \\ \tan^{-1}(\frac{y_t - y_t^L}{x_t - x_t^L}) - \theta_t \end{bmatrix} + \mathbf{r}_t.$$



(a) (b)

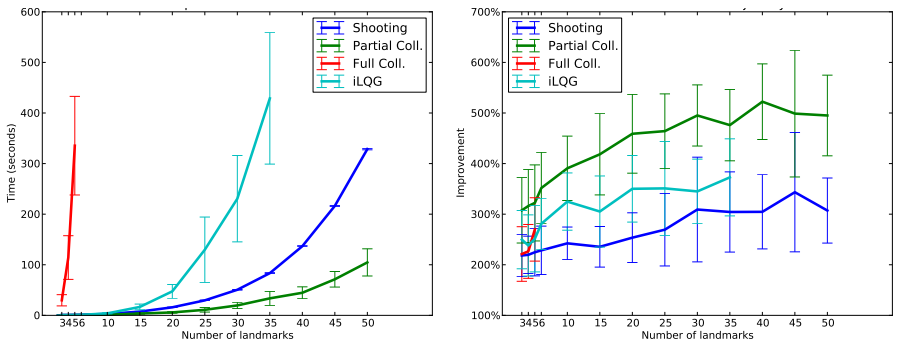
Fig. 5. Active SLAM: A car-like robot navigating in an environment with $L = 12$ landmarks. The robot uses EKF-SLAM for simultaneous localization and mapping. The objective is to plan motions for the robot to visit four landmarks (in counter-clockwise order) in the environment, starting from the bottom left, to minimize uncertainty in its state and the landmark positions. (a) A state space trajectory is unable to detect any landmarks due to the limited range of sensing of the robot, resulting in considerable accumulation of uncertainty (shown as blue ellipses). (b) Belief space planning using covariance-free partial collocation computes a plan that leads the robot to visit landmarks en route to waypoints to considerably reduce uncertainty.

Experiments: In this scenario, the robot has a limited sensing range and can only sense landmarks within a distance of $d_{\max}$ from its current position. In order to get a smooth measurement function, we smooth the boundary of the circular sensing region of radius $t_{\max}$ using a sigmoid function [27]. Fig. 5(a) shows a state space trajectory that visits four waypoints in the environment (in counter-clockwise order), each segment consisting of $T = 15$ time steps. However, due to the limited sensing range, the robot is unable to detect any of the landmarks, hence resulting in a considerable increase in uncertainty along the entire trajectory. In contrast, belief space planning using the partial collocation formulation is able to compute plans that visit landmarks en route to waypoints to reduce uncertainty in robot state and landmark positions (Fig. 5(b)).

We compared the performance and improvement over a baseline state space trajectory (Fig. 5(a)) for varying number of landmarks. The landmark positions were sampled within three clusters in the environment to preserve the complexity of the planning problem. Fig. 6(a) shows the performance of covariance-free optimization formulations (both shooting and partial collocation) as compared to full collocation and iLQG [36]. The covariance-free formulations are faster than full collocation and iLQG, which is consistent with our preliminary analysis of the running times of these formulations. The shooting formulation is slower than partial collocation because of the target constraints imposed by the waypoints, which leads to slower convergence.

Fig. 6(b) shows the improvement in the objective using belief space planning relative to the objective evaluated for a baseline state space trajectory. Partial collocation leads to greatest reduction in the objective as compared to other methods. The improvement increases with an increase in the number of landmarks, indicating that belief space planning computes different plans that optimize the objective further, for different arrangements of landmarks.

We could not scale the covariance-free formulations beyond 50 landmarks because of the inability of CasADi [2] to compute gradients using automatic differentiation since



(a) (b)

Fig. 6. Active SLAM: (a) Performance of trajectory optimization methods as the number of landmarks in the environment increase shown in terms of the mean and 1 standard deviation computed across 25 runs. (b) Improvement in the objective relative to the objective evaluated for a baseline state space trajectory shown in Fig. 5(a). Partial collocation performs best in terms of both criteria. Overall, covariance-free optimization scales to 50 landmarks (103 dimensional state space) with an average computation time of 103 seconds for the entire trajectory.

CasADi usage exceeded available memory. However, this tool is under active development and we envision that future releases will allow us to scale beyond this number.

6 Discussion and Conclusion

In this work, we focused on trajectory optimization formulations for computing locally optimal plans in Gaussian belief spaces. We showed that by excluding the covariance from the optimization, we can solve planning problems in 100 dimensional state spaces and obtain computational speedups of 400× compared to related approaches that incorporate the covariance in the state. The running time complexity per step of the optimization is $O(n^3T)$, which is an improvement over related approaches. We summarize the pros and cons of the different trajectory optimization formulations in Table. 3.

	Covariance-free Opt.			
	Shooting	Partial Collocation	Full Collocation	Dynamic Programming
Quadratic objective (Eq. (5))	X	X	✓	✓
Control policy	X	X	X	✓
Max likelihood observation assumption	✓	✓	✓	X
Infeasible initialization	X	✓	✓	X
Ensures $\Sigma_t \succeq 0$ (PSD)	✓	✓	X	X
Target constraint	X	✓	✓	X
State bounds	X	✓	✓	X
Control bounds	✓	✓	✓	X
Covariance bounds	X	X	✓	X
Optimization problem size	$n_{\mathbf{u}}T$	$(n_{\mathbf{x}}+n_{\mathbf{u}})T$	$(\frac{n_{\mathbf{x}}(n_{\mathbf{x}}+1)}{2}+n_{\mathbf{u}})T$	T problems of size $(\frac{n_{\mathbf{x}}(n_{\mathbf{x}}+1)}{2}+n_{\mathbf{u}})$
Complexity per optimization step	$O(n^3T)$	$O(n^3T)$	$O(n^6T)$	$O(n^6T)$ [35] $O(n^4T)$ [36]

Table 3. Comparison of trajectory optimization methods for Gaussian belief space planning.

Our experiments suggest that covariance-free partial collocation offers considerable promise moving forward. In future work, we plan to improve on the scalability of covariance-free trajectory optimization. We also plan to address the issue of collision avoidance by adding a cost term to the objective as in Van den Berg et al. [35]. An expanded version of the paper and code for reproducing the results reported in this paper, is available at: http://rll.berkeley.edu/beliefopt/.

Acknowledgments

This research has been funded in part by AFOSR-YIP Award #FA9550-12-1-0345, by NSF under award IIS-1227536, by a DARPA Young Faculty Award #D13AP00046, CITRIS Seed Grant, and by a Sloan Fellowship. Michael Laskey has been funded by an NSF Graduate Research Fellowship.

References

1. Agha-mohammadi, A., Chakravorty, S., Amato, N.M.: FIRM: Sampling-based Feedback Motion Planning Under Motion Uncertainty and Imperfect Measurements. Int. Journal of Robotics Research 33(2), 268–304 (2014)
2. Andersson, J., Åkesson, J., Diehl, M.: CasADi: A Symbolic Package for Automatic Differentiation and Optimal Control. In: Recent Advances in Algorithmic Differentiation, pp. 297–307. Springer (2012)
3. Bergstra, J.: Theano: a CPU and GPU Math Expression Compiler (2011), http://deeplearning.net/software/theano/
4. Bertsekas, D.: Dynamic Programming and Optimal Control. Athena Scientic (2001)
5. Betts, J.T.: Practical Methods for Optimal Control and Estimation using Nonlinear Programming, vol. 19. SIAM (2010)
6. Bry, A., Roy, N.: Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). pp. 723–730 (2011)
7. C. Papadimitriou, J.T.: The Complexity of Markov Decision Processes. Mathematics of Operations Research 12(3), 441–450 (1987)
8. Camacho, E.F., Bordons, C.: Model Predictive Control. Springer Verlag, London, UK (2004)
9. Dallaire, P., Besse, C., Ross, S., Chaib-draa, B.: Bayesian Reinforcement Learning in Continuous POMDPs with Gaussian Processes. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS). pp. 2604–2609 (2009)
10. Deisenroth, M., Mchutchon, A., Hall, J., Rasmussen, C.E.: PILCO policy search framework (2013), http://mloss.org/software/view/508/
11. Deisenroth, M.P., Peters, J.: Solving Nonlinear Continuous State-Action-Observation POMDPs for Mechanical Systems with Gaussian Noise. In: European Workshop on Reinforcement Learning (EWRL 2012) (2013)
12. Diehl, M.: Numerical Optimal Control (2011), http://homes.esat.kuleuven.be/~mdiehl/TRENTO/numopticon.pdf
13. Domahidi, A., Zgraggen, A., Zeilinger, M., Morari, M., Jones, C.: Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control. In: IEEE Conf. on Decision and Control (CDC). pp. 668 – 674 (2012)
14. Domahidi, A.: FORCES: Fast Optimization for Real-time Control on Embedded Systems (2012), http://forces.ethz.ch
15. Erez, T., Smart, W.D.: A Scalable Method for Solving High-Dimensional Continuous POMDPs Using Local Approximation. In: Conf. on Uncertainty in Artificial Intelligence. pp. 160–167 (2010)

16. Griewank, A., Walther, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM (2008)
17. Hauser, K.: Randomized Belief-Space Replanning in Partially-Observable Continuous Spaces. In: Algorithmic Foundations of Robotics IX, pp. 193–209. Springer (2011)
18. Hollinger, G., Sukhatme, G.: Stochastic Motion Planning for Robotic Information Gathering. In: Robotics: Science and Systems (RSS) (2013)
19. Indelman, V., Carlone, L., Dellaert, F.: Towards Planning in Generalized Belief Space. In: Int. Symp. on Robotics Research (ISRR) (2013)
20. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and Acting in Partially Observable Stochastic Domains. Artificial Intelligence 101(1-2), 99–134 (1998)
21. Kaelbling, L.P., Lozano-Pérez, T.: Integrated Task and Motion Planning in Belief Space. Int. Journal of Robotics Research 32(9-10), 1194–1227 (2013)
22. Kontitsis, M., Theodorou, E.A., Todorov, E.: Multi-Robot Active SLAM with Relative Entropy Optimization. In: Proc. American Control Conference (ACC). pp. 2757–2764 (2013)
23. Kurniawati, H., Bandyopadhyay, T., Patrikalakis, N.M.: Global Motion Planning under Uncertain Motion, Sensing, and Environment Map. Autonomous Robots 33(3), 255–272 (2012)
24. Leung, C., Huang, S., Kwok, N., Dissanayake, G.: Planning under Uncertainty using Model Predictive Control for Information Gathering. Robotics and Autonomous Systems 54(11), 898–910 (2006)
25. Nocedal, J., Wright, S.: Numerical Optimization. Springer Verlag (1999)
26. Patil, S., Duan, Y., Schulman, J., Goldberg, K., Abbeel, P.: Gaussian Belief Space Planning with Discontinuities in Sensing Domains. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA) (2014)
27. Platt, R., Tedrake, R., Kaelbling, L., Lozano-Perez, T.: Belief Space Planning assuming Maximum Likelihood Observations. In: Robotics: Science and Systems (RSS) (2010)
28. Porta, J., Vlassis, N., Spaan, M., Poupart, P.: Point-based Value Iteration for Continuous POMDPs. Journal of Machine Learning Research 7, 2329–2367 (2006)
29. Prentice, S., Roy, N.: The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. Int. Journal of Robotics Research 28(11–12), 1448–1465 (2009)
30. Schulman, J., Ho, J., Lee, A., Bradlow, H., Awwal, I., Abbeel, P.: Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization. In: Robotics: Science and Systems (RSS) (2013)
31. Sommer, H., Pradalier, C., Furgale, P.: Automatic Differentiation on Differentiable Manifolds as a Tool for Robotics. In: Int. Symp. on Robotics Research (ISRR) (2013)
32. Stachniss, C., Grisetti, G., Burgard, W.: Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. In: Robotics: Science and Systems (RSS). vol. 2 (2005)
33. Valencia, R., Morta, M., Andrade-Cetto, J., Porta, J.M.: Planning reliable paths with Pose SLAM. IEEE Trans. on Robotics 29(4), 1050–1059 (2013)
34. van den Berg, J., Abbeel, P., Goldberg, K.: LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information. Int. Journal of Robotics Research 30(7), 895–913 (2011)
35. van den Berg, J., Patil, S., Alterovitz, R.: Motion Planning under Uncertainty using Iterative Local Optimization in Belief Space. Int. Journal of Robotics Research 31(11), 1263–1278 (2012)
36. van den Berg, J., Patil, S., Alterovitz, R.: Efficient Approximate Value Iteration for Continuous Gaussian POMDPs. In: Proc. AAAI Conference on Artificial Intelligence (2012)
37. Vitus, M.P., Tomlin, C.J.: Closed-Loop Belief Space Planning for Linear, Gaussian Systems. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). pp. 2152–2159 (2011)
38. Webb, D.J., Crandall, K.L., van den Berg, J.: Online Parameter Estimation via Real-Time Replanning of Continuous Gaussian POMDPs (2013)