Model Predictive Path Integral Control: From Theory to Parallel Computation

Grady Williams,* Andrew Aldrich,† and Evangelos A. Theodorou‡
Georgia Institute of Technology, Atlanta, Georgia 30332

DOI: 10.2514/1.G001921

In this paper, a model predictive path integral control algorithm based on a generalized importance sampling scheme is developed and parallel optimization via sampling is performed using a graphics processing unit. The proposed generalized importance sampling scheme allows for changes in the drift and diffusion terms of stochastic diffusion processes and plays a significant role in the performance of the model predictive control algorithm. The proposed algorithm is compared in simulation with a model predictive control version of differential dynamic programming on nonlinear systems. Finally, the proposed algorithm is applied on multiple vehicles for the task of navigating through a cluttered environment. The current simulations illustrate the efficiency and robustness of the proposed approach and demonstrate the advantages of computational frameworks that incorporate concepts from statistical physics, control theory, and parallelization against more traditional approaches of optimal control theory.

Nomenclature

B	=	diffusion term in the stochastic dynamics
B_c	=	submatrix of the diffusion term in the stochastic dynamics.
$\tilde{B}$	=	diffusion term in the augmented multivehicle stochastic dynamics
$\mathrm{d}t$	=	time step in continuous time
f	=	drift term in the stochastic dynamics
f_a	=	indirectly actuated drift
f_c	=	directly actuated drift
$\tilde{f}$	=	drift term in the augmented multivehicle stochastic dynamics
G	=	state-dependent control transition matrix
$\mathcal{G}$	=	substitution matrix
G_c	=	submatrix of the state-dependent control transition matrix
$\tilde{G}$	=	state-dependent control transition matrix in the augmented multivehicle stochastic dynamics
p, q	=	probabilities of trajectories
q	=	state-dependent running cost
R	=	control weight matrix
$\tilde{R}$	=	augmented control weight matrix for multivehicle stochastic dynamics
S	=	state-dependent part of the cost for a trajectory
$\tilde{S}$	=	trajectory cost with the generalized importance sampling terms
$\boldsymbol{u}$	=	control for single-vehicle stochastic dynamics
$\tilde{\boldsymbol{u}}$	=	augmented control for multivehicle stochastic dynamics
V	=	value function
$\boldsymbol{x}$	=	state for single-vehicle stochastic dynamics
$\boldsymbol{x}^{(a)}$	=	indirectly actuated state
$\boldsymbol{x}^{(c)}$	=	directly actuated state
$\tilde{\boldsymbol{x}}$	=	augmented state for multivehicle stochastic dynamics
z, μ	=	auxiliary variables
α	=	matrix for process-sampling noise modification
Γ, Q	=	auxiliary variables
ζ	=	auxiliary variable
Δt	=	time step in discrete time
$\delta\boldsymbol{u}$	=	control updates
λ	=	temperature
Λ	=	auxiliary variable
Σ	=	variance of process noise
ϕ	=	terminal cost
Ψ	=	desirability function

Subscripts

i	=	subscript for time instances
j	=	subscript for vehicle id

I. Introduction

THE path integral optimal control framework [1–3] provides a mathematically sound methodology for developing optimal control algorithms based on stochastic sampling of trajectories. The key idea in this framework is that the value function for the optimal control problem is transformed using the Feynman–Kac lemma [4,5] into an expectation over all possible trajectories, which is known as a path integral. This transformation allows stochastic optimal control problems to be solved with a Monte Carlo approximation using forward sampling of stochastic diffusion processes.

There has been a variety of algorithms developed in the path integral control setting. The most straightforward application of path integral control is when the iterative feedback control law suggested in [6] is implemented in its open-loop formulation. This requires that sampling takes place only from the initial state of the optimal control problem. A more effective approach is to use the path integral control framework to find the parameters of a feedback control policy. This can be done by sampling in policy parameter space; these methods are known as policy improvement with path integrals [1]. Another approach to finding the parameters of a policy is to attempt to directly sample from the optimal distribution defined by the value function [7]. Other methods along similar threads of research were included in [8,9].

Another way that the path integral control framework can be applied is in a model predictive control setting. In this setting, an open-loop control sequence is constantly optimized in the background while the machine is simultaneously executing the "best guess" that the controller has. An issue with this approach is that many trajectories must be sampled in real time, which is difficult when the system has complex dynamics. One way around this problem is to drastically simplify the system under consideration by using a hierarchical scheme [10]; then, use path integral control to generate trajectories for a point mass that is then followed by a low-level controller. Even though this approach may be successful for

Received 3 December 2015; revision received 12 April 2016; accepted for publication 12 June 2016; published online Open Access 23 January 2017.

*Ph.D. Graduate Student, School of Interactive Computing, Autonomous Control and Decision Systems Laboratory.

†M.S. Graduate Student, Daniel Guggenheim School of Aerospace Engineering, Autonomous Control and Decision Systems Laboratory.

‡Assistant Professor, Daniel Guggenheim School of Aerospace Engineering, Autonomous Control and Decision Systems Laboratory.

certain applications, it is limited in the kinds of behaviors that it can generate because it does not consider the full nonlinearity of dynamics. A more efficient approach is to take advantage of the parallel nature of sampling and use a graphics processing unit (GPU) [11] to sample thousands of trajectories from the nonlinear dynamics.

A major issue in the path integral control framework is that the expectation is taken with respect to the uncontrolled dynamics of the system. This is problematic because the probability of sampling a low-cost trajectory using the uncontrolled dynamics is typically very low. This problem becomes more drastic when the underlying dynamics are nonlinear and sampled trajectories can become trapped in undesirable parts of the state space. It has previously been demonstrated how to change the mean of the sampling distribution using Girsanov's theorem [3,6]; this can then be used to develop an iterative algorithm. However, the variance of the sampling distribution has always remained unchanged. Although, in some simple simulated scenarios, changing the variance is not necessary; in many cases, the natural variance of a system will be too low to produce useful deviations from the current trajectory. Previous methods have either dealt with this problem by artificially adding noise into the system and then optimizing the noisy system [1,9], or they have simply ignored the problem entirely and sampled from whatever distribution worked best [11,12]. Although these approaches can be successful, both are problematic in that the optimization either takes place with respect to the wrong system or the resulting algorithm ignores the theoretical basis of path integral control.

Given the limitations of prior work on path integral stochastic control, the contributions of this paper are given as follows:

1) We derive a novel generalized importance sampling scheme for model predictive stochastic optimal control. The approach we take here generalizes prior results in that it enables both the mean and the variance of the sampling distribution to be changed by the control designer, without violating the underlying assumptions made in the path integral derivation. This enables the algorithm to converge fast enough that it can be applied in a model predictive control setting.

2) After deriving the model predictive path integral control (MPPI) algorithm, we compare it with an existing model predictive control formulation based on differential dynamic programming (DDP) [13–15]. DDP is one of the most powerful techniques for trajectory optimization. It relies on a first- or second-order approximation of the dynamics and a quadratic approximation of the cost and value function along a nominal trajectory. In this work, we show that MPPI outperforms DDP in tasks such as navigation of a quadrotor through a cluttered environment. The superior performance of the MPPI is due to the fact that it can optimize cost functions that are hard to approximate as quadratic functions along nominal trajectories.

3) We provide simulation results that demonstrate the scalability of our approach to high-dimensional stochastic optimal control problems. In particular, we apply our proposed algorithm to teams of three and nine quadrotors for the task of navigating through a cluttered environment. The team of three quadrotors consists of 48 state dimensions (16 dimensions per vehicle), whereas the team of nine quadrotors consists of 144 state dimensions.

4) We perform an analysis in simulation to demonstrate the connection between computational complexity, scalability, and performance of the resulting MPPI control policy. This analysis demonstrates key tradeoffs among the aforementioned factors and indicates the importance of parallelization as a computational paradigm for real-time stochastic optimal control.

5) We discuss how the generic hardware structure of a GPU memory can be used in an efficient way to support the implementation of the proposed MPPI algorithm. In particular, we discuss how the computation of the MPPI algorithm is split between global, shared, and register memory within the GPU memory architecture.

The work in this paper is organized as follows: In Sec. II, an overview of the path integral control framework is provided. Section III includes the main theorem on the generalized importance sampling that changes the mean and variance of the diffusion under consideration. The derivation of the generalized importance sampling scheme is provided in the form of a proof in the appendices A, B. In Sec. IV, we provide the path integral control formulation for the case of multiagent systems; in Sec. V, we discuss the model predictive control implementation, as well as parallelization methods used to speedup optimization. Finally, in Sec. VI, we provide a set of simulations; whereas in Sec. VII, we conclude with observations of the proposed framework.

II. Path Integral Control

In this section, we review the path integral optimal control framework [2]. There are multiple perspectives from which this framework can be derived [3]. In the approach we take here, we start from the stochastic Hamilton–Jacobi–Bellman partial differential equation (PDE) for systems affine in controls and noise. Then, by making an exponential transformation of the value function and assuming a relationship between the noise and controls, we linearize the PDE, which enables the Feynman–Kac theorem to be applied. This theorem relates the linearized PDE to a path integral representation of the value function, which we differentiate to obtain the optimal control.

A. Exponential Transformation of the Value Function

Let $\boldsymbol{x}_t = \boldsymbol{x}(t) \in \mathbb{R}^n$ denote the state of a dynamical system at time t; $\boldsymbol{u}(\boldsymbol{x}_t, t) \in \mathbb{R}^m$ denotes a control input for the system, $\tau{:}[t_0, T] \to \mathbb{R}^n$ represents a trajectory of the system, and $\mathrm{d}\boldsymbol{w} \in \mathbb{R}^p$ is a Brownian disturbance. In the path integral control framework, we suppose that the dynamics take the following form:

$$\mathrm{d}\boldsymbol{x} = (f(\boldsymbol{x}_t, t) + G(\boldsymbol{x}_t, t)\boldsymbol{u}(\boldsymbol{x}_t, t))\mathrm{d}t + B(\boldsymbol{x}_t, t)\,\mathrm{d}\boldsymbol{w} \tag{1}$$

In other words, the dynamics are affine in control and subject to an affine Brownian disturbance. We also assume that G, B, f, and $\boldsymbol{x}$ are partitioned as follows:

$$G(\boldsymbol{x}_t, t) = \begin{pmatrix} 0_{l\times m} \\ G_c(\boldsymbol{x}_t, t) \end{pmatrix}, \qquad B(\boldsymbol{x}_t, t) = \begin{pmatrix} 0_{l\times p} \\ B_c(\boldsymbol{x}_t, t) \end{pmatrix},$$
$$f(\boldsymbol{x}_t, t) = \begin{pmatrix} f_a(\boldsymbol{x}_t, t) \\ f_c(\boldsymbol{x}_t, t) \end{pmatrix} \qquad \boldsymbol{x}_t = \begin{pmatrix} \boldsymbol{x}_t^{(a)} \\ \boldsymbol{x}_t^{(c)} \end{pmatrix} \tag{2}$$

where

$$G_c(\boldsymbol{x}_t, t){:}\ \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^{(n-l)\times m}$$

$$B_c(\boldsymbol{x}_t, t){:}\ \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^{(n-l)\times p}$$

$$f_a(\boldsymbol{x}_t, t){:}\ \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^{l\times 1}$$

$$f_n(\boldsymbol{x}_t, t){:}\ \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^{(n-l)\times 1}$$

and

$$\boldsymbol{x}_t^{(a)} \in \mathbb{R}^l \text{ and } \boldsymbol{x}_t^{(c)} \in \mathbb{R}^{n-l}$$

The state–space formulation given by Eqs. (1) and (2) can represent a large class of dynamical systems. Examples of such systems are aerospace, mechanical, and robotic systems with rigid-body and multibody dynamics operating under the presence of stochastic forces [16–18]. In addition, biomechanical and neuromuscular systems [19] with stochasticity due to neural noise can also be represented by Eq. (1) with the partitioning in Eq. (2). Next, we suppose that the cost function for the optimal control problem has a quadratic control cost and an arbitrary state-dependent cost. Let $\phi(\boldsymbol{x}_T)$ denote a final terminal cost and $q(\boldsymbol{x}_t, t)$ a state-dependent running cost, and define $R(\boldsymbol{x}_t, t)$ as a positive definite matrix. We allow $R(\boldsymbol{x}_t, t)$ to be time and state dependent as long as it remains positive definite for all values of $\boldsymbol{x}$ and t. The value function $V(\boldsymbol{x}_t, t)$ for this optimal control problem is defined as follows:

$$V(\boldsymbol{x}_t, t) = \min_{u} \mathbb{E}_{\mathbb{Q}}\left[\phi(\boldsymbol{x}_T, T) + \int_t^T \left(q(\boldsymbol{x}_t, t) + \frac{1}{2}\boldsymbol{u}(\boldsymbol{x}_t, t)^T R(\boldsymbol{x}_t, t)\boldsymbol{u}(\boldsymbol{x}_t, t)\right) \mathrm{d}t\right] \tag{3}$$

where the expectations over trajectories taken with respect to Eq. (1) are denoted as $\mathbb{E}_{\mathbb{Q}}[\cdot]$; we will also be interested in taking expectations with respect to the uncontrolled dynamics of the system [i.e., Eq. (1) with $\boldsymbol{u} \equiv 0$]. These will be denoted $\mathbb{E}_{\mathbb{P}}[\cdot]$. The stochastic Hamilton–Jacobi–Bellman equation [20,21] for the type of system in Eq. (1) and for the cost function in Eq. (3) is given as follows:

$$-\partial_t V = q(\boldsymbol{x}_t, t) + f(\boldsymbol{x}_t, t)^T V_x - \frac{1}{2} V_x^T G(\boldsymbol{x}_t, t) R(\boldsymbol{x}_t, t)^{-1} G(\boldsymbol{x}_t, t)^T V_x + \frac{1}{2}\mathrm{tr}(B(\boldsymbol{x}_t, t) B(\boldsymbol{x}_t, t)^T V_{xx}) \tag{4}$$

with the boundary condition $V(\boldsymbol{x}_T, T) = \phi(\boldsymbol{x}_T, T)$. And, the optimal control is expressed in terms of the solution to this PDE as follows:

$$\boldsymbol{u}^*(\boldsymbol{x}_t, t) = -R(\boldsymbol{x}_t, t)^{-1} G(\boldsymbol{x}_t, t)^T V_x \tag{5}$$

Therefore, in order to compute the optimal controls, we need only solve this backward-in-time PDE. Unfortunately, classical methods for solving partial differential equations of this nature suffer from the curse of dimensionality and are intractable for systems with more than a few state variables. This motivates the approach we take in the path integral control framework, which is to transform the PDE into a path integral, which is an expectation over all possible trajectories of the system. This expectation can then be approximated via a Monte Carlo approximation using forward sampling of the stochastic dynamics.

The Feynman–Kac formula, which relates PDEs to path integrals, can only be applied to linear PDEs. Thus, we need to transform Eq. (3) into a linear PDE. If we assume a relationship between the noise and control costs of the system, then this can be done exactly. First, we apply the exponential transform to the value function. We define the desirability function $\Psi(\boldsymbol{x}, t)$ such that

$$V(\boldsymbol{x}, t) = -\lambda \log(\Psi(\boldsymbol{x}, t)) \tag{6}$$

Note that trajectories with a low cost will have a high Ψ value, and vice versa. This explains calling Ψ desirability. To continue, we need to compute the derivatives of $V(\boldsymbol{x}, t)$ in terms of $\Psi(\boldsymbol{x}, t)$. In the following presentation, for notational compactness, we drop the functional dependencies on $\boldsymbol{x}$ and t. The derivatives with respect to time and the gradient with respect to the state are easily computed as follows:

$$\partial_t V = -\frac{\lambda}{\Psi}\partial_t \Psi \qquad V_x = -\frac{\lambda}{\Psi}\Psi_x \tag{7}$$

Computing a nice expression for the Hessian matrix V_{xx} in terms of Ψ is a little more difficult. The ijth entry of V_{xx} in terms of Ψ is

$$\nabla_{xx} V_{ij} = -\lambda\left(\frac{\Psi(\partial\Psi/\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j) - (\partial\Psi/\partial \boldsymbol{x}_i)(\partial\Psi/\partial \boldsymbol{x}_j)}{\Psi^2}\right) \tag{8}$$

With this equation in hand, we can compactly write the Hessian matrix as follows:

$$V_{xx} = -\frac{\lambda}{\Psi}\Psi_{xx} + \frac{\lambda}{\Psi^2}\Psi_x \Psi_x^T \tag{9}$$

Now, we substitute these expressions into Eq. (4):

$$\lambda\frac{\partial_t \Psi}{\Psi} = q - \lambda\frac{f^T \Psi_x}{\Psi} + \frac{\lambda}{2}\frac{\Psi_x^T}{\Psi} G R^{-1} G^T \left(\lambda\frac{\Psi_x}{\Psi}\right) - \frac{1}{2}\mathrm{tr}\left(BB^T\left(\frac{\lambda}{\Psi}\Psi_{xx}\right)\right) + \frac{\lambda}{2\Psi^2}\mathrm{tr}(BB^T \Psi_x \Psi_x^T) \tag{10}$$

But, using basic properties of the trace, we see that

$$\mathrm{tr}(BB^T \Psi_x \Psi_x^T) = \mathrm{tr}(\Psi_x^T BB^T \Psi_x) \tag{11}$$

But, $\Psi_x^T BB^T \Psi_x$ is a scalar, so

$$\mathrm{tr}(BB^T \Psi_x \Psi_x^T) = \Psi_x^T BB^T \Psi_x \tag{12}$$

Substituting this expression for the trace, multiplying by Ψ/λ, and simplifying yields the following:

$$\partial_t \Psi = \frac{\Psi}{\lambda} q - f^T \Psi_x - \underline{\frac{\lambda}{2\Psi}\Psi_x^T G R^{-1} G^T \Psi_x} - \frac{1}{2}\mathrm{tr}(BB^T \Psi_{xx}) + \underline{\frac{1}{2\Psi}\Psi_x^T BB^T \Psi_x} \tag{13}$$

It is easy to see that, if we make the assumption that $BB^T = \lambda G R^{-1} G^T$, then the two underlined terms will cancel, and what remains is a linear PDE. This assumption implies that the noise in any given state is proportional to the control authority that can be exercised over that state. In other words, if a state suffers from high variance, the following are necessary:

1) We can directly actuate that state.

2) It is cheap to control.

We now assume that the assumption holds; then, we have the following:

$$\partial_t \Psi(\boldsymbol{x}_t, t) = \frac{\Psi(\boldsymbol{x}_t, t)}{\lambda} q(\boldsymbol{x}_t, t) - f(\boldsymbol{x}_t, t)^T \Psi_x - \frac{1}{2}\mathrm{tr}(B(\boldsymbol{x}_t, t) B(\boldsymbol{x}_t, t)^T \Psi_{xx}) \tag{14}$$

which is a linear PDE in terms of Ψ. This particular partial differential equation is known as the backward Chapman–Kolmogorov PDE. Finally, in order for the connection between control authority and noise to hold, there are two requirements. These requirements are that the stochasticity has to enter the dynamics via the control channel and that the control cost has to be tuned properly to match the variance of the stochastic forces. The first requirement is restrictive in cases of dynamical systems where the source of stochasticity is not only due to additive stochastic forces but also due to model uncertainty in indirectly actuated states. The second requirement is less restrictive because the cost function under minimization has additional state-dependent terms. These terms can be tuned to capture a large envelope of cost functions and behaviors, despite the fact that the control cost weight is determined based on the variance of the process noise.

B. Application of the Feynman–Kac Lemma

With the stochastic Hamilton–Jacobi–Bellman (HJB) equation transformed into the linear backward Chapman–Kolmogorov equation, we can now apply the Feynman–Kac formula, which gives the solution to the PDE as follows:

$$\Psi(\boldsymbol{x}_{t_0}, t_0) = \mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}\int_{t_0}^T q(\boldsymbol{x}, t)\,\mathrm{d}t\right)\Psi(\boldsymbol{x}_T, T)\right] \tag{15}$$

By recognizing that the term $\Psi(\boldsymbol{x}_T)$ is the transformed terminal cost $e^{-(1/\lambda)\phi(\boldsymbol{x}_T)}$, we can rewrite this expression as follows:

$$\Psi(\boldsymbol{x}_{t_0}, t_0) \approx \mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(\tau)\right)\right] \tag{16}$$

where

$$S(\tau) = \phi(\boldsymbol{x}_T) + \int_{t_0}^T q(\boldsymbol{x}_t, t)\,\mathrm{d}t$$

is the cost to go of the state-dependent cost of a trajectory. There are several interesting properties of these equations. The most obvious

property is that all of the terms are forward-in-time processes, whereas the HJB-PDE is purely backward in time. Another property to take note of is that the expectation is with respect to the uncontrolled dynamics of the system, and the control costs have completely disappeared. At first inspection, this appears very strange, because the optimal controls are being computed without ever explicitly referring to the controlled dynamics of the system or the control costs. The reason this is possible is that the controls are linked to the passive dynamics through the assumption between the noise and controls. The optimization is taking into account the control matrix G and the control costs, but only implicitly.

Equation (16) provides a path integral form for the value function; however, we are interested in the gradient of the value function with respect to $\boldsymbol{x}$ because that is the function that provides the optimal control input. Fortunately, we can compute the gradient of Ψ with respect to the initial state $\boldsymbol{x}_{t_0}$ analytically. This is a straightforward, albeit lengthy, computation, so we omit it and refer the interested reader to [1,3]. After taking the gradient, we obtain the following:

$$\boldsymbol{u}^* \mathrm{d}t = \boldsymbol{\mathcal{G}}(\boldsymbol{x}_{t_0}, t_0) \frac{\mathbb{E}_{\mathbb{P}}[\exp(-(1/\lambda)S(\tau))B_c(\boldsymbol{x}_{t_0}, t_0)\,\mathrm{d}\boldsymbol{w}]}{\mathbb{E}_{\mathbb{P}}[\exp(-(1/\lambda)S(\tau))]} \tag{17}$$

where the matrix $\boldsymbol{\mathcal{G}}(\boldsymbol{x}_t, t)$ is defined as follows:

$$\boldsymbol{\mathcal{G}}(\boldsymbol{x}_t, t) = R(\boldsymbol{x}_t, t)^{-1} G_c(\boldsymbol{x}_t, t)^T (G_c(\boldsymbol{x}_t, t) R(\boldsymbol{x}_t, t)^{-1} G_c(\boldsymbol{x}_t, t)^T)^{-1} \tag{18}$$

Note that, if $G_c(\boldsymbol{x}_t, t)$ is square (which is the case if the system is not overactuated), this reduces to $G_c(\boldsymbol{x}_t, t)^{-1}$. Equation (17) is the path integral form of the optimal control. The fundamental difference between this form of the optimal control and classical optimal control theory is that, instead of relying on a backward-in-time process, this formula requires the evaluation of an expectation that can be approximated using forward sampling of stochastic differential equations.

Equation (17) provides an expression for the optimal control in terms of a path integral. However, these equations are for continuous time and, in order to sample trajectories on a digital computer, we need discrete time approximations. We first discretize the dynamics of the system. We have that $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \mathrm{d}\boldsymbol{x}_t$ where $\mathrm{d}\boldsymbol{x}_t$ is defined as follows:

$$\mathrm{d}\boldsymbol{x}_t = (f(\boldsymbol{x}_t, t) + G(\boldsymbol{x}_t, t)\boldsymbol{u}(\boldsymbol{x}_t, t))\Delta t + B(\boldsymbol{x}_t, t)\epsilon\sqrt{\Delta t} \tag{19}$$

The term ϵ is a time-varying vector of standard normal Gaussian random variables. For the uncontrolled dynamics of the system, we have the following:

$$\mathrm{d}\boldsymbol{x}_t = f(\boldsymbol{x}_t, t)\Delta t + B(\boldsymbol{x}_t, t)\epsilon\sqrt{\Delta t} \tag{20}$$

Another way we can express $B(\boldsymbol{x}_t, t)\,\mathrm{d}\boldsymbol{w}$ that will be useful is as follows:

$$B(\boldsymbol{x}_t, t)\,\mathrm{d}\boldsymbol{w} \approx \mathrm{d}\boldsymbol{x}_t - f(\boldsymbol{x}_t, t)\Delta t \tag{21}$$

Lastly, the term $S(\tau)$ is defined as

$$S(\tau) \approx \phi(\boldsymbol{x}_T) + \sum_{i=1}^{N} q(\boldsymbol{x}_t, t)\Delta t$$

where $N = (T - t)/\Delta t$. Next, by defining p as the probability induced by the discrete time uncontrolled dynamics and taking into account the partitioning of f and B in Eq. (2), we can approximate Eq. (17) as follows:

$$\boldsymbol{u}(\boldsymbol{x}_{t_0}, t_0)^* \Delta t = \boldsymbol{\mathcal{G}}(\boldsymbol{x}_{t_0}, t_0) \frac{\mathbb{E}_p[\exp(-(1/\lambda)S(\tau))(\mathrm{d}\boldsymbol{x}_{t_0}^c - f_c(\boldsymbol{x}_{t_0}, t_0)\Delta t)]}{\mathbb{E}_p[\exp(-(1/\lambda)S(\tau))]} \tag{22}$$

By dividing with Δt from both sides, we will have the following:

$$\boldsymbol{u}(\boldsymbol{x}_{t_0}, t_0)^* = \boldsymbol{\mathcal{G}}(\boldsymbol{x}_{t_0}, t_0) \frac{\mathbb{E}_p[\exp(-(1/\lambda)S(\tau))((\mathrm{d}\boldsymbol{x}_{t_0}^c/\Delta t) - f_c(\boldsymbol{x}_{t_0}, t_0))]}{\mathbb{E}_p[\exp(-(1/\lambda)S(\tau))]} \tag{23}$$

Note that we have moved the Δt term, multiplying $\boldsymbol{u}$ over to the right-hand side of the equation, and inserted it into the expectation.

III. Importance Sampling with Diffusion Processes

Importance sampling techniques using Girsanov's theorem have been used in state estimation methods such as particle filtering [22–24] to perform inference of the state of a partially observable stochastic system given noisy observations. In this work, we use importance sampling within the context of stochastic optimal control. In particular, we derive a novel importance sampling scheme that allows sampling from general stochastic processes that are different from the initial stochastic process in the drift and diffusion terms. Our approach aims to improve the performance of path integral control when applied to nonlinear stochastic systems with many dimensions.

To motivate the use of importance sampling, we start our analysis with Eq. (23). In particular, Eq. (23) provides a method for approximating the optimal control, via Monte Carlo methods, which can be easily implemented on a digital computer. Given an approximate model of the system, samples of the uncontrolled dynamics can be generated in large numbers and the expectation can then be evaluated based on those samples. In practice, this approach is unlikely to succeed. The problem is that p is typically an inefficient distribution to sample from (i.e., the cost to go will be high for most trajectories sampled from p). Intuitively, sampling from the uncontrolled dynamics corresponds to turning a machine on and waiting for the natural noise in the system dynamics to produce interesting behavior. In theory, as the number of samples goes to infinity, interesting behavior will occur purely due to the noise in the system. However, the number of samples needed in order to get this behavior may be incomprehensibly large.

To efficiently approximate the controls, we require the ability to sample from a distribution that is likely to produce low-cost trajectories. We can do this by changing the mean (adding an initial control input) of the system and by changing the variance of the system noise. In previous applications of path integral control [3,6], the mean of the sampling distribution has been changed by applying Girsanov's theorem. Applying Girsanov's theorem only works for changing the mean of the sampling distribution. But, in systems where the natural variance of the system is small, changing the mean is insufficient because the state space is never aggressively explored by the sampling. It is therefore necessary to change the variance of the sampling distribution as well. In the following derivation, we provide a method for changing both the initial control input and the variance of the sampling distribution.

A. Likelihood Ratio

We suppose that we have a sampling distribution with nonzero control input and a changed variance, which we denote as q; and we would like to approximate Eq. (23) using samples from q as opposed to p. Now, if we write the expectation term in Eq. (23) in integral form, we get the following:

$$\frac{\int \exp(-(1/\lambda)S(\tau))((\mathrm{d}\boldsymbol{x}_{t_0}^{(c)}/\Delta t) - f_c(\boldsymbol{x}_t, t))\mathsf{p}(\tau)\,\mathrm{d}\tau}{\int \exp(-(1/\lambda)S(\tau))\mathsf{p}(\tau)\,\mathrm{d}\tau} \tag{24}$$

where we are abusing notation and using τ to represent the discrete time trajectory $(\boldsymbol{x}_{t_0}, \boldsymbol{x}_{t_1}, \ldots \boldsymbol{x}_{t_N})$. Next, we multiply both integrals by $1 = \mathsf{q}(\tau)/\mathsf{q}(\tau)$ to get the following:

$$\frac{\int \exp(-(1/\lambda)S(\tau))((\mathrm{d}\boldsymbol{x}_{t_0}^{(c)}/\Delta t) - f_c(\boldsymbol{x}_t, t))(\mathsf{q}(\tau)/\mathsf{q}(\tau))\mathsf{p}(\tau)\,\mathrm{d}\tau}{\int \exp(-(1/\lambda)S(\tau))(\mathsf{q}(\tau)/\mathsf{q}(\tau))\mathsf{p}(\tau)\,\mathrm{d}\tau} \tag{25}$$

and we can then write this as an expectation with respect to q:

$$\frac{\mathbb{E}_{\mathsf{q}}[\exp(-(1/\lambda)S(\tau))((\mathrm{d}\boldsymbol{x}_{t_0}^{(c)}/\Delta t) - f_c(\boldsymbol{x}_t, t))(\mathsf{p}(\tau)/\mathsf{q}(\tau))]}{\mathbb{E}_{\mathsf{q}}[\exp(-(1/\lambda)S(\tau))(\mathsf{p}(\tau)/\mathsf{q}(\tau))]} \tag{26}$$

We now have the expectation in terms of a sampling distribution q for which we can choose 1) the initial control sequence from which to sample around; and 2) the variance of the exploration noise that determines how aggressively the state space is explored.

However, we now have an extra term to compute: $\mathsf{p}(\tau)/\mathsf{q}(\tau)$. This is known as the likelihood ratio (or Radon–Nikodym derivative) between the distributions p and q. To derive an expression for this term, we first have to derive equations for the probability density functions of $\mathsf{p}(\tau)$ and $\mathsf{q}(\tau)$ individually. We can do this by deriving the probability density function for the general discrete time diffusion processes $P(\tau)$. The form of the probability density function is given in the following proposition: the proof of which is provided in the Appendix A.

Proposition 1: Consider the stochastic dynamics of the form

$$\mathrm{d}\boldsymbol{x}_t = (f(\boldsymbol{x}_t, t) + G(\boldsymbol{x}_t, t)\boldsymbol{u}(\boldsymbol{x}_t, t))\Delta t + B(\boldsymbol{x}_t, t)\epsilon\sqrt{\Delta t} \tag{27}$$

with the terms $G(\boldsymbol{x}_t, t)$, $B(\boldsymbol{x}_t, t)$, and $f(\boldsymbol{x}_t, t)$ defined as in Eq. (2). The probability of the trajectory

$$\mathsf{p}(\boldsymbol{x}_{t_0}, \boldsymbol{x}_{t_1}, \ldots \boldsymbol{x}_{t_N}) = \prod_{i=1}^{N} \mathsf{p}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}})$$

is given by the equation that follows:

$$\mathsf{p}(\boldsymbol{x}_{t_0}, \boldsymbol{x}_{t_1}, \ldots \boldsymbol{x}_{t_N}) = Z(\tau)^{-1} \exp\left(-\frac{\Delta t}{2}\sum_{i=1}^{N}(z_i - \mu_i)^T\Sigma_i^{-1}(z_i - \mu_i)\right) \tag{28}$$

where the terms $Z(\tau)$, Σ_i, ζ_i, and μ_i are specified as follows:

$$Z(\tau) = \prod_{i=1}^{N}(2\pi)^{n/2}|\Sigma_i|^{1/2} \quad \Sigma_i = B_c(\boldsymbol{x}_{t_i}, t_i)B_c(\boldsymbol{x}_{t_i}, t_i)^T\Delta t$$

$$z_i = \frac{\mathrm{d}\boldsymbol{x}_{t_i}^{(c)}}{\Delta t} - f^{(c)}(\boldsymbol{x}_{t_i}, t_i)$$

$$\mu_i = G^{(c)}(\boldsymbol{x}_{t_i}, t_i)\boldsymbol{u}(\boldsymbol{x}_{t_i}, t_i)$$

With Eq. (28) in hand, we are now ready to state the generalized likelihood ratio between two diffusion processes. The likelihood ratio is given in the form of a theorem: the proof of which is provided in Appendix B.

Theorem 1: Let $\mathsf{p}(\tau)$ be the probability density function for trajectories under the uncontrolled discrete time dynamics:

$$\mathrm{d}\boldsymbol{x}_t = f(\boldsymbol{x}_t, t)\Delta t + B(\boldsymbol{x}_t, t)\epsilon\sqrt{\Delta t}$$

and let $\mathsf{q}(\tau)$ be the probability density function for trajectories under the controlled dynamics with an adjusted variance:

$$\mathrm{d}\boldsymbol{x}_t = (f(\boldsymbol{x}_t, t) + G(\boldsymbol{x}_t, t)\boldsymbol{u}(\boldsymbol{x}_t, t))\Delta t + B_E(\boldsymbol{x}_t, t)\epsilon\sqrt{\Delta t}$$

where the adjusted variance has the following form:

$$B_E(\boldsymbol{x}_t, t) = \begin{pmatrix} 0 \\ A_t B_c(\boldsymbol{x}_t, t) \end{pmatrix}$$

and define z_i, μ_i, and Σ_i as in Proposition 1. Let Q_i be defined as follows:

$$Q_i = (z_i - \mu_i)^T\Gamma_i^{-1}(z_i - \mu_i) + 2(\mu_i)^T\Sigma_i^{-1}(z_i - \mu_i) + \mu_i^T\Sigma_i^{-1}\mu_i \tag{29}$$

where Γ_i is

$$\Gamma_i = (\Sigma_i^{-1} - (A_{t_i}^T\Sigma_i A_{t_i})^{-1})^{-1} \tag{30}$$

Then, under the condition that each A_{t_i} is invertible and each Γ_i is invertible, the likelihood ratio for the two distributions is as follows:

$$\left(\prod_{i=1}^{N}|A_{t_i}|\right)\exp\left(-\frac{\Delta t}{2}\sum_{i=1}^{N}Q_i\right) \tag{31}$$

The key difference between this theorem and earlier path integral works that use an application of Girsanov's theorem to sample from a nonzero control input is that this theorem allows for a change in the variance as well as a nonzero control input. In the expression for the likelihood ratio derived here, the last two terms

$$\left[2\mu_i^T\Sigma_i^{-1}(z_i - \mu_i) + \mu_i^T\Sigma_i^{-1}\mu_i\right]$$

are exactly the terms from Girsanov's theorem. The first term $[(z_i - \mu_i)^T\Gamma_i^{-1}(z_i - \mu_i)]$, which can be interpreted as penalizing overaggressive exploration, is the only new term.

B. Likelihood Ratio as Additional Running Cost

The form of the likelihood ratio just derived is easily incorporated into the path integral control framework by folding it into the cost to go as an extra running cost. Note that the likelihood ratio appears in both the numerator and denominator of Eq. (23). Therefore, any terms that do not depend on the state can be factored out of the expectation and canceled. This removes the numerically troublesome normalizing term

$$\prod_{j=1}^{N}|A_{t_j}|$$

So, only the summation of Q_i remains. Recall that

$$\Sigma = \lambda G_c(\boldsymbol{x}_t, t)R(\boldsymbol{x}_t, t)^{-1}G_c(\boldsymbol{x}_t, t)^T$$

This implies that

$$\Gamma = \lambda((G_c(\boldsymbol{x}_t, t)R(\boldsymbol{x}_t, t)^{-1}G_c(\boldsymbol{x}_t, t)^T)^{-1} - (A^T G_c(\boldsymbol{x}_t, t)R(\boldsymbol{x}_t, t)^{-1}G_c(\boldsymbol{x}_t, t)^T A)^{-1})^{-1}$$

Now, define

$$\boldsymbol{H} = G_c(\boldsymbol{x}_t, t)R(\boldsymbol{x}_t, t)^{-1}G_c(\boldsymbol{x}_t, t)^T$$

and $\tilde{\Gamma} = (1/\lambda)\Gamma$. We then have the following:

$$Q = \frac{1}{\lambda}((z - \mu)^T\tilde{\Gamma}^{-1}(z - \mu) + 2\mu^T\boldsymbol{H}^{-1}(z - \mu) + \mu^T\boldsymbol{H}^{-1}\mu)$$

Then, by redefining the running cost $q(\boldsymbol{x}_t, t)$ as

$$\tilde{q}(\boldsymbol{x}, \boldsymbol{u}, \mathrm{d}\boldsymbol{x}) = q(\boldsymbol{x}_t, t) + \frac{1}{2}(z - \mu)^T\tilde{\Gamma}^{-1}(z - \mu) + \mu^T\boldsymbol{H}^{-1}(z - \mu) + \frac{1}{2}\mu^T\boldsymbol{H}^{-1}\mu$$

and

$$\tilde{S}(\tau) = \phi(\boldsymbol{x}_T) + \sum_{j=1}^{N} \tilde{q}(\boldsymbol{x}, \boldsymbol{u}, \mathrm{d}\boldsymbol{x})$$

we have

$$\boldsymbol{u}_t^* = \boldsymbol{\mathcal{G}}(\boldsymbol{x}_t, t) \frac{\mathbb{E}_q[\exp(-(1/\lambda)\tilde{S}(\tau))((\mathrm{d}\boldsymbol{x}_t^{(c)}/\Delta t) - f_c(\boldsymbol{x}_t, t))]}{\mathbb{E}_q[\exp(-(1/\lambda)\tilde{S}(\tau))]}$$

Also note that $\mathrm{d}\boldsymbol{x}_t$ is now equal to

$$\mathrm{d}\boldsymbol{x}_t = (f(\boldsymbol{x}_t, t) + G(\boldsymbol{x}_t, t)\boldsymbol{u}(\boldsymbol{x}_t, t))\Delta t + B(\boldsymbol{x}_t, t)\epsilon\sqrt{\Delta t}$$

So, we can rewrite

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\Delta t} - f(\boldsymbol{x}_t, t)$$

as

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\Delta t} - f(\boldsymbol{x}_t, t) = G(\boldsymbol{x}_t, t)\boldsymbol{u}(\boldsymbol{x}_t, t) + B(\boldsymbol{x}_t, t)\frac{\epsilon}{\sqrt{\Delta t}}$$

Given the partitioning of the terms G, B, f, and $\boldsymbol{x}$ in Eq. (2), and because $G_c(\boldsymbol{x}_t, t)$ does not depend on the expectation, we can pull it out and get the iterative update law:

$$\boldsymbol{u}_t^* = \boldsymbol{\mathcal{G}}(\boldsymbol{x}_t, t)G_c(\boldsymbol{x}_t, t)\boldsymbol{u}(\boldsymbol{x}_t, t) + \boldsymbol{\mathcal{G}}(\boldsymbol{x}_t, t)\frac{\mathbb{E}_{\mathrm{q}}[\exp(-(1/\lambda)\tilde{S}(\tau))B_c(\boldsymbol{x}_t, t)(\epsilon/\sqrt{\Delta t})]}{\mathbb{E}_{\mathrm{q}}[\exp(-(1/\lambda)\tilde{S}(\tau))]} \quad (32)$$

C. Special Case

The update law [Eq. (32)] is applicable for a very general class of systems. In this section, we examine a special case that we use for all of our experiments. We consider dynamics of the following form:

$$\mathrm{d}\boldsymbol{x}_t = f(\boldsymbol{x}_t, t)\Delta t + G(\boldsymbol{x}_t, t)\left(\boldsymbol{u}(\boldsymbol{x}_t, t)\Delta t + \frac{1}{\sqrt{\rho}}\epsilon\sqrt{\Delta t}\right)$$

and for the sampling distribution, we set A equal to $\sqrt{\nu}I$. We also assume that $G_c(\boldsymbol{x}_t, t)$ is a square invertible matrix. This reduces $\boldsymbol{\mathcal{G}}(\boldsymbol{x}_t, t)$ to $G_c(\boldsymbol{x}_t, t)^{-1}$. Next, the dynamics can be rewritten as follows:

$$\mathrm{d}\boldsymbol{x}_t = f(\boldsymbol{x}_t, t)\Delta t + G(\boldsymbol{x}_t, t)\left(\boldsymbol{u}(\boldsymbol{x}_t, t) + \frac{1}{\sqrt{\rho}}\frac{\epsilon}{\sqrt{\Delta t}}\right)\Delta t$$

Then, we can interpret

$$\frac{1}{\sqrt{\rho}}\frac{\epsilon}{\sqrt{\Delta t}}$$

as a random change in the control input; to emphasize this, we will denote this term as

$$\delta\boldsymbol{u} = \frac{1}{\sqrt{\rho}}\frac{\epsilon}{\sqrt{\Delta t}}$$

We then have

$$B_c(\boldsymbol{x}_t, t)\frac{\epsilon}{\sqrt{\Delta t}} = G_c(\boldsymbol{x}_t, t)\delta\boldsymbol{u}$$

This yields the iterative update law as follows:

$$\boldsymbol{u}(\boldsymbol{x}_t, t)^* = \boldsymbol{u}(\boldsymbol{x}_t, t) + \frac{\mathbb{E}_{\mathrm{q}}[\exp(-(1/\lambda)\tilde{S}(\tau))\delta\boldsymbol{u}]}{\mathbb{E}_{\mathrm{q}}[\exp(-(1/\lambda)\tilde{S}(\tau))]} \quad (33)$$

which can be approximated as follows:

$$\boldsymbol{u}(\boldsymbol{x}_{t_i}, t_i)^* \approx \boldsymbol{u}(\boldsymbol{x}_{t_i}, t_i) + \frac{\sum_{k=1}^{K}\exp(-(1/\lambda)\tilde{S}(\tau_{i,k}))\delta\boldsymbol{u}_{i,k}}{\sum_{k=1}^{K}\exp(-(1/\lambda)\tilde{S}(\tau_{i,k}))} \quad (34)$$

where K is the number of random samples (termed rollouts), and $S(\tau_{i,k})$ is the cost to go of the kth rollout from time t_i onward. This expression is simply a reward-weighted average of random variations in the control input. Next, we investigate what the likelihood ratio addition to the running cost is. For these dynamics, we have the following simplifications:

Simplification 1:

$$z - \mu = G_c(\boldsymbol{x}_t, t)\delta\boldsymbol{u}$$

Simplification 2:

$$\tilde{\Gamma}^{-1} = (1 - \nu^{-1})(G_c(\boldsymbol{x}_t, t)R(\boldsymbol{x}_t, t)^{-1}G_c(\boldsymbol{x}_t, t)^T)^{-1}$$

Simplification 3:

$$\boldsymbol{H}^{-1} = (G_c(\boldsymbol{x}_t, t)R(\boldsymbol{x}_t, t)^{-1}G_c(\boldsymbol{x}_t, t)^T)^{-1}$$

Given these simplifications, $\tilde{q}$ reduces to the following:

$$\tilde{q}(\boldsymbol{x}, \boldsymbol{u}, \mathrm{d}\boldsymbol{x}) = q(\boldsymbol{x}_t, t) + \frac{(1 - \nu^{-1})}{2}\delta\boldsymbol{u}^T R\delta\boldsymbol{u} + \boldsymbol{u}^T R\delta\boldsymbol{u} + \frac{1}{2}\boldsymbol{u}^T R\boldsymbol{u}$$

This means that the introduction of the likelihood ratio simply introduces the original control cost from the optimal control formulation into the sampling cost, which originally only included state-dependent terms.

IV. Multiagent Systems

The multiagent formulation of path integral control was first investigated in [25] and further explored in [26], where a distributed algorithm was proposed. Here, we will review the multiagent formulation and show its iterative version using the importance sampling scheme that was derived in Sec. III. In particular, for the multivehicle case, we consider a team of M vehicles/agents. The optimal control problem in this case will take the following form:

$$V(\boldsymbol{x}_t, t) = \min_u J(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, t) = \min_u \mathbb{E}_{\mathbb{Q}}\left[\phi(\tilde{\boldsymbol{x}}_T, T) + \int_t^T (q(\tilde{\boldsymbol{x}}_t, t) + \frac{1}{2}\tilde{\boldsymbol{u}}^T\tilde{R}\tilde{\boldsymbol{u}})\,\mathrm{d}t\right] \quad (35)$$

subject to the vehicle dynamics:

$$\mathrm{d}\boldsymbol{x}^{(j)} = f(\boldsymbol{x}_t^{(j)}, t)\,\mathrm{d}t + G(\boldsymbol{x}_t^j, t)\boldsymbol{u}^{(j)}(\boldsymbol{x}_t, t)\,\mathrm{d}t + B(\boldsymbol{x}_t^{(j)}, t)\,\mathrm{d}\boldsymbol{w}^{(j)},$$
$$\boldsymbol{x}_{t_0}^{(j)} = \boldsymbol{x}_0^{(j)}, \qquad \forall\ j = 1, \ldots M \quad (36)$$

where

$$\tilde{\boldsymbol{x}}^T = (\boldsymbol{x}^{(1)T}, \boldsymbol{x}^{(2)T}, \ldots, \boldsymbol{x}^{(M)T})$$

and

$$\tilde{\boldsymbol{u}}^T = (\boldsymbol{u}^{(1)T}, \boldsymbol{u}^{(2)T}, \ldots, \boldsymbol{u}^{(M)T})$$

and $\tilde{R}$ is a block diagonal matrix

$$\tilde{R} = \text{Block Diag}(R_1, R_2, \ldots, R_M)$$

with the submatrices $R_j > 0, \forall\ j = 1, \ldots, M$. In this formulation of multivehicle stochastic optimal control, there is coupling of the states of the vehicles only through the cost function under minimization. The stochastic dynamics in a more compact form can be expressed as follows:

$$\mathrm{d}\tilde{\boldsymbol{x}} = \tilde{f}(\tilde{\boldsymbol{x}}_t, t)\,\mathrm{d}t + \tilde{G}(\tilde{\boldsymbol{x}}_t, t)\tilde{\boldsymbol{u}}(\tilde{\boldsymbol{x}}_t, t)\,\mathrm{d}t + \tilde{B}(\tilde{\boldsymbol{x}}_t, t)\,\mathrm{d}\tilde{\boldsymbol{w}} \tag{37}$$

where $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{u}}$ are the augmented state and controls that were already defined, whereas the augmented noise term is defined as

$$\mathrm{d}\tilde{\boldsymbol{w}}^T = (\mathrm{d}\boldsymbol{w}^{(1)T}, \mathrm{d}\boldsymbol{w}^{(2)T}, \ldots, \mathrm{d}\boldsymbol{w}^{(M)T})$$

The terms $\tilde{f}$, $\tilde{G}$, and $\tilde{B}$ are the augmented drift, control, and diffusion matrices, which are defined as follows:

$$\tilde{f} = \begin{pmatrix} f^{(1)} \\ f^{(2)} \\ \cdots \\ f^{(M)} \end{pmatrix}, \qquad \tilde{G} = \begin{pmatrix} G^{(1)} & 0 & \ldots & 0 \\ 0 & G^{(2)} & \ldots & 0 \\ 0 & 0 & \ldots & G^{(M)} \end{pmatrix}, \quad \text{and}$$

$$\tilde{B} = \begin{pmatrix} B^{(1)} & 0 & \ldots & 0 \\ 0 & B^{(2)} & \ldots & 0 \\ 0 & 0 & \ldots & B^{(M)} \end{pmatrix}$$

where we use the notations $f^{(j)} = f(\boldsymbol{x}_t^{(j)}, t)$, $G^{(j)} = G(\boldsymbol{x}_t^{(j)}, t)$, and $B^{(j)} = B(\boldsymbol{x}_t^{(j)}, t)$. The optimal control for the augmented system is expressed as follows:

$$\tilde{\boldsymbol{u}}(\tilde{\boldsymbol{x}}, t) = -\tilde{R}^{-1}\tilde{G}^T(\tilde{\boldsymbol{x}}_t, t)V_{\tilde{x}}(\tilde{\boldsymbol{x}}, t) \tag{38}$$

Given the form of the augmented drift $\tilde{f}$, control $\tilde{G}$, and diffusion $\tilde{B}$ terms, as well as the block diagonal structure of the control weight matrix $\tilde{R}$, the computation of the total optimal control $\tilde{\boldsymbol{u}}$ can be split into the computation of the individual controllers $\boldsymbol{u}^{(j)}$ as follows:

$$\boldsymbol{u}^{(j)}(\boldsymbol{x}, t) = -R_j^{-1}G^T(\boldsymbol{x}_t^{(j)}, t)V_{x^{(j)}}(\boldsymbol{x}, t), \quad \forall\ j = 1, 2, \ldots, M \tag{39}$$

When incorporating the desirability function $\Psi(\boldsymbol{x}, t)$, the optimal control takes the following form:

$$\boldsymbol{u}_*^{(j)}\,\mathrm{d}t = \boldsymbol{\mathcal{G}}(\boldsymbol{x}_t^{(j)}, t)\frac{\mathbb{E}_{\mathbb{P}}[\exp(-(1/\lambda)S(\tau))B_c(\boldsymbol{x}_t^{(j)}, t)\,\mathrm{d}\boldsymbol{w}^{(j)}]}{\mathbb{E}_{\mathbb{P}}[\exp(-(1/\lambda)S(\tau))]} \tag{40}$$

Note that, although the optimal control $\boldsymbol{u}_*^{(j)}$ can be locally evaluated for each agents, its computation requires the knowledge of the cost term $S(\tau)$, which is a function of the states of the other agents as well. For the case of iterative path integral control, the update law for each agent is given as follows:

$$\begin{aligned}\boldsymbol{u}_*^{(j)} &= \boldsymbol{\mathcal{G}}(\boldsymbol{x}_t^{(j)}, t)G_c(\boldsymbol{x}_t^{(i)}, t)\boldsymbol{u}^{(j)}(\boldsymbol{x}_t, t) \\ &\quad + \boldsymbol{\mathcal{G}}(\boldsymbol{x}_t^{(j)}, t)\frac{\mathbb{E}_{\mathrm{q}}[\exp(-(1/\lambda)\tilde{S}(\tau))B_c(\boldsymbol{x}_t^{(j)}, t)(\epsilon^{(j)}/\sqrt{\Delta t})]}{\mathbb{E}_{\mathrm{q}}[\exp(-(1/\lambda)\tilde{S}(\tau))]}\end{aligned} \tag{41}$$

Similar to the noniterative case, the computation of the update law in Eq. (41) requires the knowledge of the cost term $\tilde{S}(\tau)$, which depends on the state and controls of the other agents. Similar to the single-vehicle case, the control for each vehicle in the multiagent case can be simplified to

$$\boldsymbol{u}^{(j)}(\boldsymbol{x}_{t_i}, t_i)^* \approx \boldsymbol{u}^{(j)}(\boldsymbol{x}_{t_i}, t_i) + \frac{\sum_{k=1}^K \exp(-(1/\lambda)\tilde{S}(\tau_{i,k}))\delta\boldsymbol{u}_{i,k}^{(j)}}{\sum_{k=1}^K \exp(-(1/\lambda)\tilde{S}(\tau_{i,k}))} \tag{42}$$

where

$$\delta\boldsymbol{u}_{i,k}^{(j)} = \frac{1}{\sqrt{\rho}}\frac{\epsilon_{i,k}}{\sqrt{\Delta t}}$$

Next, we discuss the receding horizon version of path integral control.

V. Model Predictive Path Integral Control

We apply the iterative path integral control update law, with the generalized importance sampling term, in a model predictive control setting. At every time step (20 ms in our experiments), thousands of trajectories are sampled from the importance sampling distribution $\boldsymbol{q}$; and these are used to estimate the optimal control. For this scheme to work, we have to be able to sample thousands of trajectories in real time. To achieve this requirement, we implement the sampling and approximation of the path integral on a GPU for efficient parallel computation.

In the parallel implementation, trajectory samples are computed by each parallel thread by forward propagating the dynamics with random variations in the control inputs. This operation is constrained by two problem specific elements: 1) the complexity of the dynamics under consideration; and 2) the number of time steps to simulate, which is related to the time horizon of the problem and the desired control frequency.

A. Compute Unified Device Architecture Implementation

The compute unified device architecture (CUDA) provides an application program interface to Nvidia GPUs for developing heterogeneous (running both CPU and GPU) programs. Modern graphics processing units are extremely parallel processors specialized for performing graphics computations. These capabilities lend themselves well to Monte Carlo problems like the computation of the path integral here. Using GPUs to compute Monte Carlo approximations of path integrals has been done in fields outside of control theory, such as computational finance [27], biology [28], and physics [29]. However, using GPUs for model predictive control using path integrals is a new concept that has not been thoroughly explored.

To understand why the GPU implementation is efficient, it is first necessary to give a brief overview of CUDA's memory structure. For a detailed summary, and an introduction to GPU programming in general, we refer the reader to [30]. CUDA's memory structure can be coarsely subdivided into three layers: 1) global memory, 2) shared memory, and 3) register memory. Global memory is the most flexible, in the sense that every parallel thread has read/write access to global memory elements. However, this flexibility comes at significant overhead cost, and large read/write operations to global memory are inevitably bottlenecks in GPU programs. Shared memory and register memory are much faster than global memory; however, they are locally restricted. Shared memory is only accessibly to threads grouped together as blocks, and register memory is local to a single thread. Moreover, these memory spaces are limited in the capacity that they can hold. (Shared memory can hold up 48 kB of data, and register memory only has a very small capacity per thread.)

The Monte Carlo approximation of the path integral is able to exploit this architecture because the main components of the algorithm (computing dynamics and evaluating costs) can be performed with a very limited amount of memory compared to the arithmetic computations necessary. This allows the main computations to be performed only by referencing shared and register memory, with only a limited amount of global memory read/writes performed. Figure 1 provides a diagram of the most straightforward version of the implementation. In this implementation, each thread computes the dynamics and costs of an entire trajectory. At each time step, a thread makes a read to global memory to get the current control input and random variation. Given this

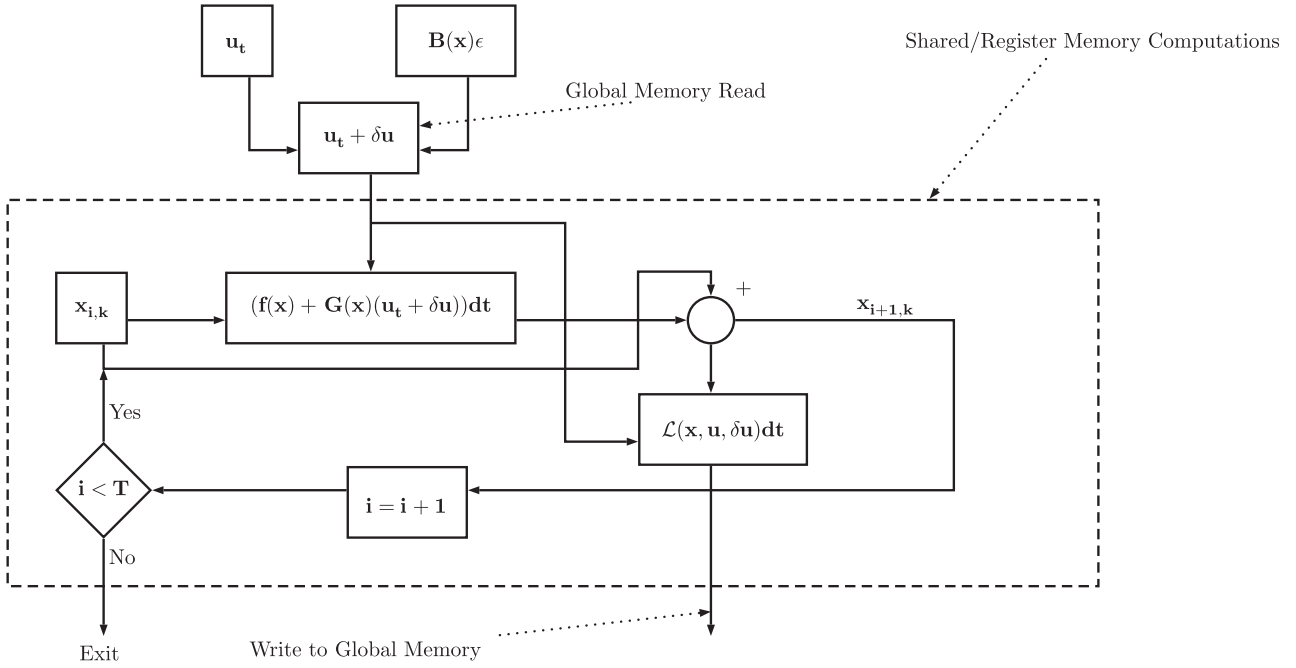

Fig. 1 Computational diagram of the dynamics/costs for the CUDA implementation.

information, the thread then computes the dynamics and costs using only data stored in shared or register memory. Lastly, the thread writes the cost to global memory and continues to the next time step.

If the dynamics are sufficiently complex, or the state space is exceptionally large, additional speedup can be achieved by parallelizing the evaluation of the dynamics. Under this scheme, each sampled trajectory is computed using a block of threads as opposed to the single-thread case. In our simulations, we use this procedure when controlling multiple quadrotors. The GPU implementation, along with the new importance sampling term that enables tuning of the exploration variance, enables us to develop a simple model predictive path integral control algorithm capable of real-time control.

B. Model Predictive Control Algorithm

We now describe the full version of MPPI. We assume that we are given the system dynamics and a term ν for adjusting the exploration variance, an initial control sequence (possibly zeros), and a cost function for the given task. First, $K \times T$ random control variations are produced (where K is the number of rollouts and T the number of time steps). This is done efficiently on a GPU using CUDA's random number generation library [31]. Next, the path integral approximation of the sequence of the optimal controls is performed using the iterative parallel sampling procedure described in Sec. V.A. Then, the current control input is executed, and the sequence of control inputs is slid down by one time step. This enables the next round of optimization to warm start with the unexecuted portion of the control sequence from the previous iteration. This is essential because, by the time a control is actually executed, it has been seen by the optimization for the entire length of the time horizon. Lastly, the very final time step is initialized using a preset constant control input.

VI. Simulations

We tested the model predictive path integral control algorithm on four simulated platforms: a cart–pole, a miniature race car, a quadrotor, and a team of three and nine quadrotors attempting to navigate in an obstacle filled environment. For the race car and quadrotor, we used a model predictive control version of the differential dynamic programming algorithm as a baseline comparison. In all of these experiments, the controller operated at 50 Hz; this meant that the open-loop control sequence was reoptimized every 20 ms.

All of the simulation experiments required the selection of parameters that determined the behavior of the system. These were parameters related to the cost function, the time horizon, and the exploration variance of the MPPI algorithm. The state-dependent portion of the cost functions we selected for the simulation experiments consisted of three or fewer components: 1) a quadratic cost for achieving a desired position or being in specified set (for instance, in the race car, we want the car to be in the center of the track), 2) a quadratic cost on the system velocities that improves the stability of the system, and 3) a cost for avoiding obstacles. Through our simulations, we determined that the MPPI controller performed best with two obstacle terms. The first was a soft cost that discouraged traveling near obstacles; and the second cost was for actually colliding with an obstacle, and should be set high enough to annihilate the colliding trajectory.

The time horizon also plays an important role in the algorithmic performance. Longer time horizons are more difficult to optimize for because there are more variables to consider; in the MPPI setting, they increase the computational demands on the algorithm because each trajectory takes more time to simulate. Due to these considerations,

Algorithm 1: Model predictive path integral control

1: **Given**: K: Number of samples
2: N: Number of time steps
3: $(\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots \boldsymbol{u}_{N-1})$: Initial control sequence
4: Δt, $\boldsymbol{x}_{t_0}$, f, G, B, v: System/sampling dynamics
5: ϕ, q, R, λ: Cost parameters
6: $\boldsymbol{u}_{\text{init}}$: Value to initialize new controls to
7: **while** task not completed, **do**
8: Generate random control variations $\delta\boldsymbol{u}$
9: **for** $k \leftarrow 0$ **to** $K-1$, **do**
10: $\boldsymbol{x} = \boldsymbol{x}_{t_0}$
11: **for** $i \leftarrow 1$ **to** $N-1$, **do**
12: $\boldsymbol{x}_{j+1} = \boldsymbol{x}_i + (f + G(\boldsymbol{u}_i + \delta\boldsymbol{u}_{i,k}))\Delta t$
13: $\tilde{S}(\tau_{i+1,k}) = \tilde{S}(\tau_{i,k}) + \tilde{q}$
14: **for** $i \leftarrow 0$ **to** $N-1$, **do**
15: $\boldsymbol{u}_i \leftarrow \boldsymbol{u}_i + \left[\sum_{k=1}^{K}(\exp(-(1/\lambda)\tilde{S}_{(\tau_{i,k})})\delta\boldsymbol{u}_{i,k} / \sum_{k=1}^{K}\exp(-(1/\lambda)\tilde{S}_{(\tau_{i,k})}))\right]$
16: send to actuators $(\boldsymbol{u}_0)$
17: **for** $i \leftarrow 0$ **to** $N-2$, **do**
18: $\boldsymbol{u}_i = \boldsymbol{u}_{i+1}$
19: $\boldsymbol{u}_{N-1} = \boldsymbol{u}_{\text{init}}$
20: Update the current state after receiving feedback
21: Check for task completion

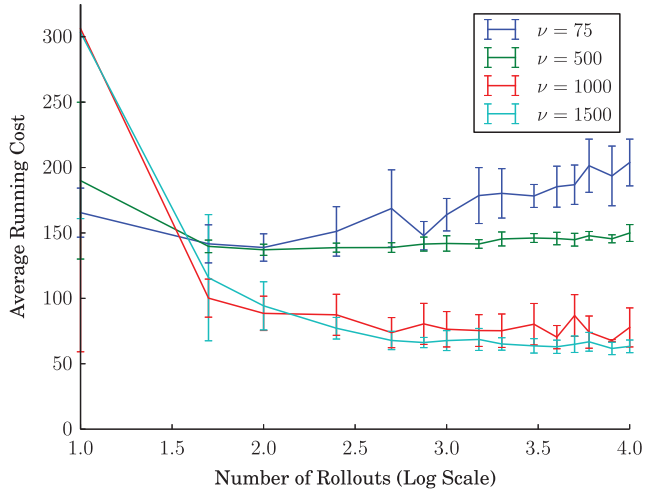

Fig. 2 Running cost for the cart–pole swingup task for different exploration variances v.

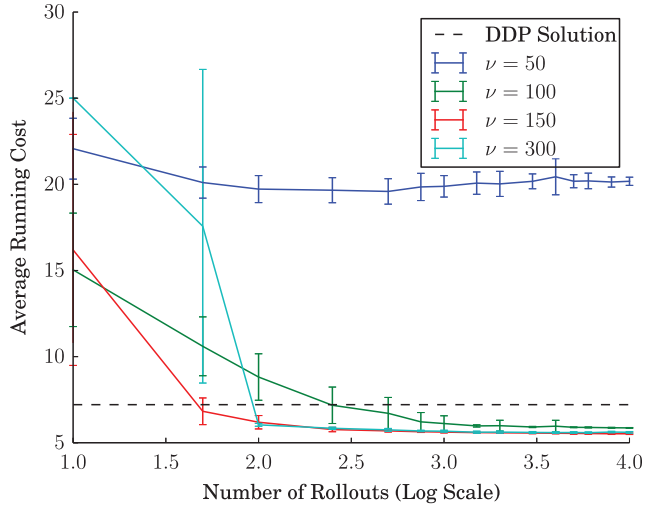

Fig. 3 Performance comparison in terms of average cost between the MPPI and MPC-DDP for the race car.

the time horizon should be set as short as possible; in the simulation experiments, we consider this is relatively easy to identify. For instance, it takes the cart–pole roughly 1 s to swing up and stabilize the pole, and it takes the race car about 2 s from entering a corner to exit. However, in general, it may not be trivial to identify a time horizon for a given task, and it may be desirable to allow for a variable time horizon.

Lastly, the exploration variance plays a fundamental role in the performance of the algorithm. If the exploration variance is too low, the algorithm is myopic and does not produce any reasonable output. If the exploration variance is too large, then the algorithm produces control inputs with significant chatter. Although, in simulation, this does not pose a problem, on a real system, this would be highly undesirable. For the systems we considered, the algorithm is able to produce useful actions with a relatively smooth control input. The ranges we select for exploration variance go from the low end of this region to the high end.

A. Cart Pole

For the cart–pole swingup task, we used the state cost $q(\boldsymbol{x}) = p^2 + 500(1 + \cos(\theta))^2 + \dot{\theta}^2 + \dot{p}^2$, where p was the position of cart, $\dot{p}$ was the velocity, and θ and $\dot{\theta}$ were the angle and angular velocity of the pole. The control input is desired velocity, which maps to velocity through the equation $\ddot{p} = 10(u - \dot{p})$. The disturbance parameter $1/\sqrt{\rho}$ was set equal 0.01, and the control cost was $R = 1$. We ran the MPPI controller for 10 s with a 1 s optimization horizon. The controller had to swing up the pole and keep it balanced for the rest of the 10 s horizon. The exploration variance parameter $\boldsymbol{v}$ was varied between 1 and 1500. The MPPI controller was able to swing up the pole faster with increasing exploration variance. Figure 2 illustrates the performance of the MPPI controller as the exploration variance and the number of rollouts were changed. Using only the natural variance of the system for exploration was insufficient in this task; in that case (not shown in the figure), the controller was never able to swing up the pole, which resulted in a cost around 2000.

B. Race Car

In the race car task, the goal was to minimize the objective function $q(\boldsymbol{x}) = 100d^2 + (v_x - 7.0)^2$, where d is defined as follows:

$$d = \left| \left(\frac{x}{13}\right)^2 + \left(\frac{y}{6}\right)^2 - 1 \right|$$

and v_x is the forward (in body frame) velocity of the car. This cost ensures that the car stays on an elliptical track while maintaining a forward speed of 7 m/s. We use a nonlinear dynamics model [32] that takes into account the (highly nonlinear) interactions between the tires and the ground. The exploration variance is set to a constant $\boldsymbol{v}$ times the natural variance of the system.

The MPPI controller is able to enter turns at close to the desired speed of 7 m/s and then slide through the turn. The DDP solution does not attempt to slide and significantly reduces its forward velocity before entering the turn; this results in a higher average cost compared to the MPPI controller. Figure 3 shows the performance comparison in terms of the average cost between MPPI and MPC-DDP as the exploration variance $\boldsymbol{v}$ changes from 50 to 300 and the number of rollouts changes from 10 to 1000. Note that the cost is capped at 25.0. As is illustrated in this figure, only with a very large increase in the exploration variance is the MPPI able to outperform MPC-DDP. This is because the highly nonlinear nature of the race car dynamics (see Appendix D) reduces the exploration of the state space when sampling is performed with small exploration variance. The reduced state-space exploration results in a suboptimal MPPI control policy.

Figure 4 illustrates the comparison of DDP (left) and the MPPI (right) performing a cornering maneuver along an ellipsoid track. The MPPI is able to make a much tighter turn while carrying more speed in and out of the corner than DDP. The direction of travel is counterclockwise. Finally, Fig. 5 illustrates the corresponding velocities.

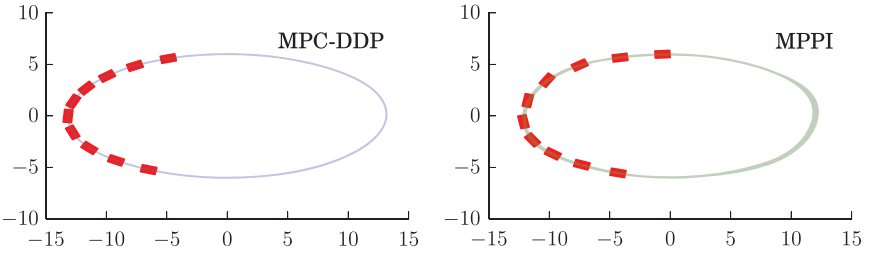

Fig. 4 Comparison of DDP (left) and MPPI (right) performing a cornering maneuver.

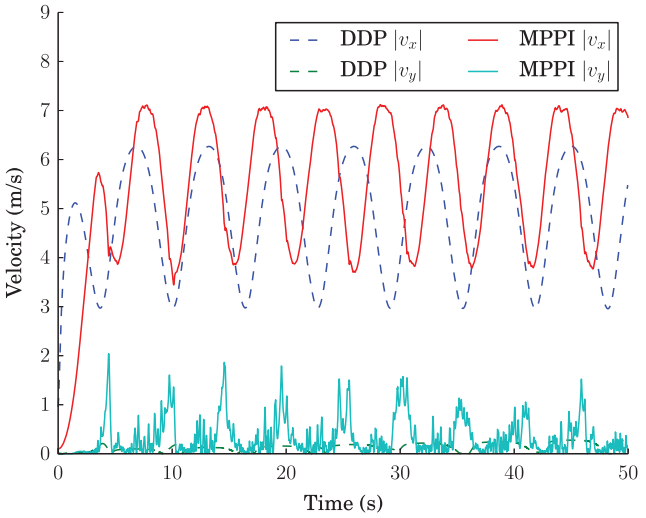

Fig. 5 Velocity comparison of DDP (left) and MPPI (right) performing a cornering maneuver.

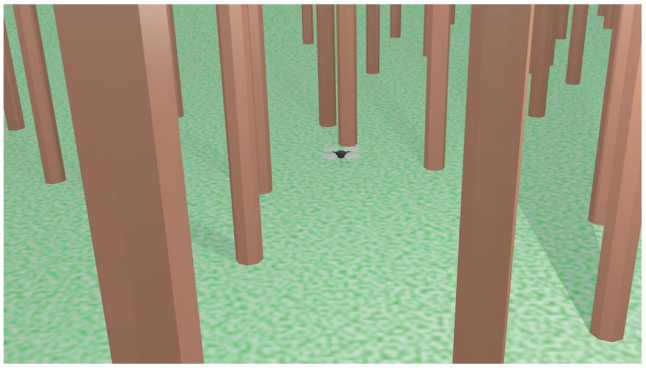

Fig. 6 Simulated forest environment used in the quadrotor navigation task.

C. Quadrotor

The quadrotor task was to fly through a field filled with cylindrical obstacles as fast as possible; see Fig. 6. We used the quadrotor dynamics model from [33]. This was a nonlinear model that included the position, velocity, Euler angles, angular acceleration, and rotor dynamics. We randomly generated three forests: the first where obstacles were (on average) 3 m apart, the second was 4 m apart, and the third was 5 m apart. We then separately created cost functions for both the MPPI and MPC-DDP controllers that guided the quadrotor through the forest as quickly as possible.

The cost function for the MPPI was of the following form:

$$q(\boldsymbol{x}) = 2.5(p_x - p_x^{\text{des}})^2 + 2.5(p_y - p_y^{\text{des}})^2 + 150(p_z - p_z^{\text{des}})^2 + 50\psi^2 + \|v\|^2 + 350\exp\left(-\frac{d}{12}\right) + 1000C$$

where (p_x, p_y, p_z) denotes the position of the vehicle. The yaw angle in radians is denoted as ψ, v is the velocity, and d is the distance to the closest obstacle. C is a variable that indicates whether the vehicle crashed into the ground or is an obstacle. Additionally, if $C = 1$ (which indicates a crash), the rollout stops simulating the dynamics and the vehicle remains where it is for the rest of the time horizon. We found that the crash indicator term is not useful for the MPC-DDP-based controller; this is not surprising because the discontinuity it creates is difficult to approximate with a quadratic function. The term in the cost for avoiding obstacles in the MPC-DDP controller consists purely of a large exponential term:

$$2000\sum_{i=1}^{N}\exp\left(-\frac{1}{2}d_i^2\right)$$

Note that this sum is over all the obstacles in the proximity of the vehicle, whereas the MPPI controller only has to consider the closest obstacle.

Because the MPPI controller can explicitly reason about crashing (as opposed to just staying away from obstacles), it is able to travel both faster and closer to obstacles than the MPC-DDP controller. Figure 7 shows the trajectories taken by MPC-DDP; one of the MPPI runs on the forest with obstacles placed (on average) 4 m away is illustrated. Because the MPPI controller can directly reason about the shape of the obstacles, it is able to safely pass through the field taking a much more direct route. Figure 8 shows the performance difference between the two algorithms in terms of the time to navigate through the forest for different densities of obstacle placement.

D. Multiple Quadrotors

We also tested the algorithm's ability to simultaneously control multiple quadrotors operating in close proximity to each other; this was done by combining several quadrotors into one large system and then attempting the same obstacle navigation task. The quadrotor was able to successfully control three quadrotors at once through the obstacle field. Most of the time, it was able to control nine quadrotors performing the same task. We also set the obstacles to move at a rate of 3 m/s in a random direction.

Figure 9 illustrates trajectories of the nine quadrotors moving through the cluttered environment with constant obstacles. Figure 10 illustrates the performance of the nine quadrotors in terms of the success rate as a function of the number of the rollouts for the case of moving and constant obstacles. As is illustrated, the performance improves as the number of rollouts increases. Finally, Fig. 11

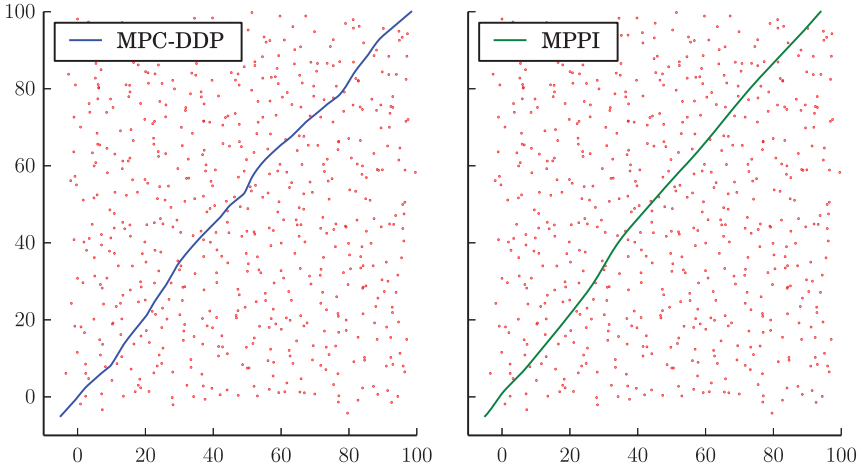

Fig. 7 Sample trajectory through a 4 m obstacle field DDP (left) and MPPI (right).

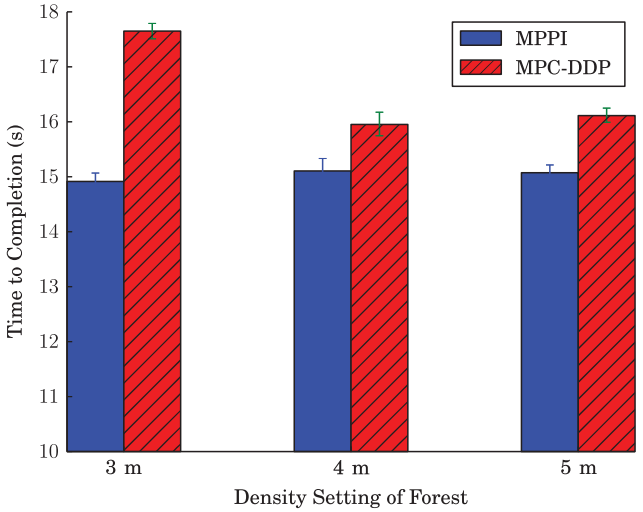

Fig. 8 Time to navigate forest. Comparison between MPPI and DDP.

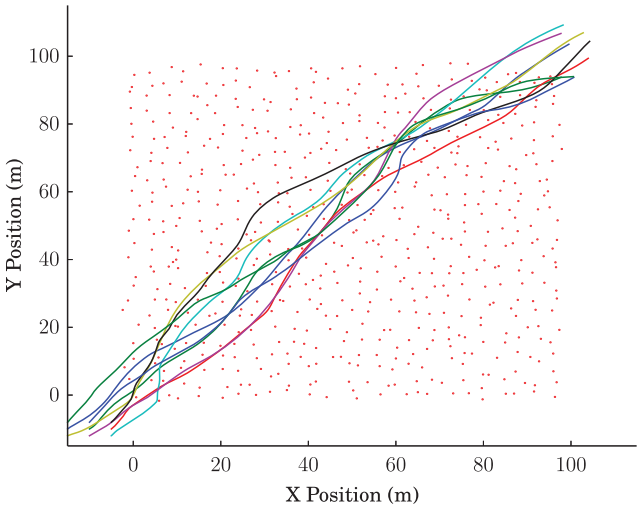

Fig. 9 Trajectories of the nine quadrotors.

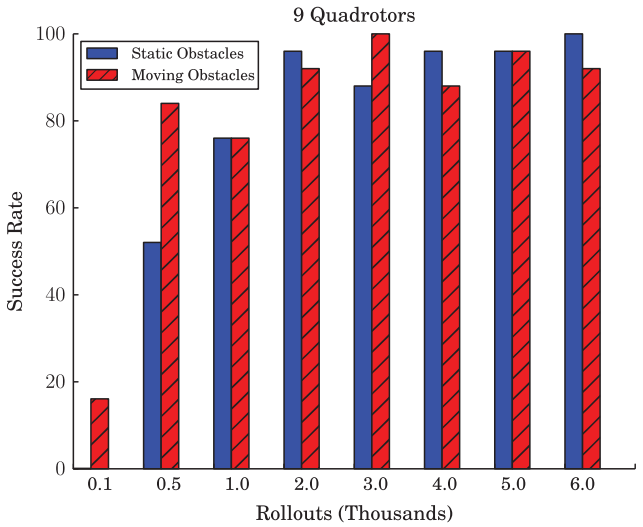

Fig. 10 Success rate for the nine quadrotors flying through a cluttered environment.

includes the time for each iteration of the MPPI controller as a function of the number of the rollout for the cases of one, three, and nine quadrotors. The dashed line in Fig. 11 represents the real time, which is in the order of 20 ms. For the case of a single quadrotor, the

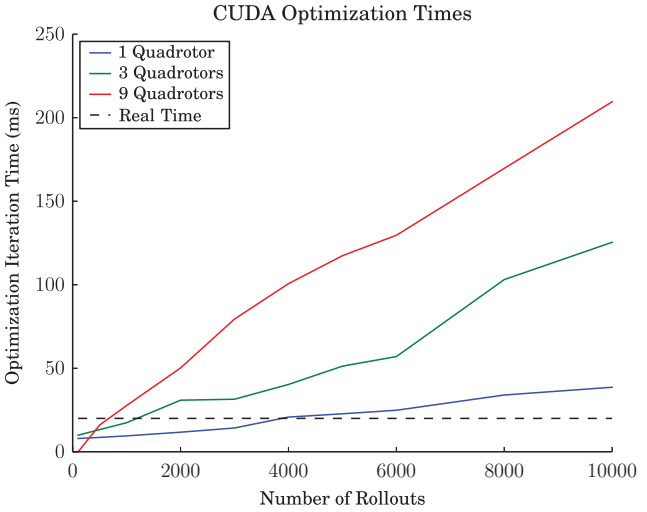

Fig. 11 Time per iteration of the MPPI controller as a function of the number of rollouts.

optimization time crosses the real time for 4000 rollouts. In the case of three and nine quadrotors, the optimization time exceeds the real time for 1200 and 500 rollouts, respectively. Twelve-hundred rollouts are sufficient to control the three-quadrotor system; however, for the nine-quadrotor system, more than 500 rollouts are needed. In the case of the nine-quadrotor system, 6000 trajectories are required, which run in about six times the real time. More important, however, the growth in computational time is approximately linear in the order of the state space. It therefore seems reasonable to believe that, with near-term hardware improvements, model predictive control of a system of the size of the nine-quadrotor system (144 state variables and 40 control inputs) will be possible in real time.

VII. Conclusions

In this paper, a model predictive path integral control algorithm is developed that is able to outperform a state-of-the-art DDP method on two difficult control tasks. The algorithm is based on stochastic sampling of system trajectories and requires no derivatives of either the dynamics or costs of the system. This enables the algorithm to naturally take into account nonlinear dynamics, and incorporate non-smooth/non- differentiable cost functions without approximations. The two keys to achieving this level of performance with a sampling based method are 1) the derivation of the generalized likelihood ratio between discrete time diffusion processes, and 2) the use of a GPU to sample thousands of trajectories in real time.

The derivation of the likelihood ratio enables the designer of the algorithm to tune the exploration variance in the path integral control framework, whereas previous methods have only allowed for the mean of the distribution to be changed. Tuning the exploration variance is critical in achieving a high level of performance because the natural variance of the system is typically too low to achieve good performance.

The experiments considered in this work only considered changing the variance by a constant multiple times the natural variance of the system. In this special case, the introduction of the likelihood ratio corresponded to adding in a control cost when evaluating the cost to go of a trajectory. A direction for future research is to investigate how to automatically adjust the variance online. Doing so could enable the algorithm to switch from aggressively exploring the state space when performing aggressive maneuvers to exploring more conservatively for performing very precise maneuvers.

Future research will also include extensions of the work presented here to more general classes of stochastic systems. These classes will include nonlinear stochastic dynamics with nonaffine controls and with stochastic disturbances that are far from being Gaussian, such as jump diffusion processes and doubly stochastic processes. The aforementioned generalizations can be achieved by using the

information theoretic formulation of the path integral control framework, which relies on the fundamental relationship between relative entropy and free energy [3,6]. Finally, a theoretical analysis on the robustness of the MPPI algorithm that is based on the number of sampled trajectories, the speed of reoptimization, and the levels of stochastic disturbances is included in ongoing work.

Appendix A: Proof of Proposition 1

We consider the stochastic dynamics

$$\mathrm{d}\boldsymbol{x}_t = (f(\boldsymbol{x}_t, t) + G(\boldsymbol{x}_t, t)\boldsymbol{u}(\boldsymbol{x}_t, t))\Delta t + B(\boldsymbol{x}_t, t)\epsilon\sqrt{\Delta t} \tag{A1}$$

The goal is to find $\mathsf{p}(\tau) = \mathsf{p}(\boldsymbol{x}_{t_0}, \boldsymbol{x}_{t_1}, \ldots \boldsymbol{x}_{t_N})$. By conditioning and using the Markov property of the state space, this probability becomes

$$\mathsf{p}(\boldsymbol{x}_{t_0}, \boldsymbol{x}_{t_1}, \ldots \boldsymbol{x}_{t_N}) = \prod_{i=1}^{N} \mathsf{p}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}) \tag{A2}$$

Now, recall that a portion of the state space has deterministic dynamics and that we partitioned the diffusion matrix as follows:

$$B(\boldsymbol{x}_t, t) = \begin{pmatrix} 0 \\ B_c(\boldsymbol{x}_t, t) \end{pmatrix} \tag{A3}$$

We can partition the state variables $\boldsymbol{x}$ into the deterministic and nondeterministic variables $\boldsymbol{x}_t^{(a)}$ and $\boldsymbol{x}_t^{(c)}$, respectively. The next step is to express the transition probability $\mathsf{p}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}})$ as the following product:

$$\mathsf{p}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}) = \mathsf{p}(\boldsymbol{x}_{t_i}^{(a)}|\boldsymbol{x}_{t_{i-1}})\mathsf{p}(\boldsymbol{x}_{t_i}^{(c)}|\boldsymbol{x}_{t_{i-1}}) \tag{A4}$$

Given the partitioning of the diffusion matrix $B(\boldsymbol{x}_t, t)$, the transition probability is $\mathsf{p}(\boldsymbol{x}_{t_i}^{(a)}|\boldsymbol{x}_{t_{i-1}}) = 1$, and thus we will have that

$$\prod_{i=1}^{N} \mathsf{p}(\boldsymbol{x}_{t_i}|\boldsymbol{x}_{t_{i-1}}) = \prod_{i=1}^{N} \mathsf{p}(\boldsymbol{x}_{t_i}^{(c)}|\boldsymbol{x}_{t_{i-1}}) \tag{A5}$$

And, from the dynamics equations, we know that each of these one-step transitions is Gaussian with mean $f_c(\boldsymbol{x}_t, t) + G_c(\boldsymbol{x}_{t_i}, t_i)\boldsymbol{u}(\boldsymbol{x}_{t_i}, t_i)$ and variance

$$\Sigma_i = B_c(\boldsymbol{x}_{t_i}, t_i)B_c(\boldsymbol{x}_{t_i}, t_i)^T\Delta t \tag{A6}$$

We then define

$$z_i = \frac{\mathrm{d}\boldsymbol{x}_{t_i}^{(c)}}{\Delta t} - f_c(\boldsymbol{x}_{t_i}, t_i)$$

and $\mu_i = G_c(\boldsymbol{x}_{t_i}, t_i)\boldsymbol{u}(\boldsymbol{x}_{t_i}, t_i)$. Applying the definition of the Gaussian distribution with these terms yields the folowing:

$$\mathsf{p}(\tau) = \prod_{i=1}^{N} \frac{\exp(-(\Delta t/2)(z_i - \mu_i)^T\Sigma_i^{-1}(z_i - \mu_i))}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} \tag{A7}$$

Then, using basic rules of exponents, this probability becomes the following:

$$Z(\tau)^{-1}\exp\left(-(\Delta t/2)\sum_{i=1}^{N}(z_i - \mu_i)^T\Sigma_i^{-1}(z_i - \mu_i)\right) \tag{A8}$$

where

$$Z(\tau) = \prod_{i=1}^{N}(2\pi)^{n/2}|\Sigma_i|^{1/2}$$

Appendix B: Proof of Generalized Importance Sampling (Theorem 1)

In discrete time, the probability of a trajectory is formulated according to Eq. (28). We thus have $\mathsf{p}(\tau)$ equal to

$$\mathsf{p}(\tau) = \frac{\exp(-(\Delta t/2)\sum_{i=1}^{N} z_i\Sigma_i z_i)}{Z_{\mathsf{p}(\tau)}}$$

and $q(\tau)$ equal to

$$\frac{\exp(-(\Delta t/2)\sum_{i=1}^{N}(z_i - \mu_i)^T(A_{t_i}^T\Sigma_i A_{t_i})^{-1}(z_i - \mu_i))}{Z_{\mathsf{q}(\tau)}}$$

Then, dividing these two equations, we have $\mathsf{p}(\tau)/\mathsf{q}(\tau)$ as follows:

$$\left(\prod_{i=1}^{N}\frac{(2\pi)^{n/2}|A_{t_i}^T\Sigma_i A_{t_i}|^{1/2})}{(2\pi)^{n/2}|\Sigma_i|^{1/2})}\right)\exp\left(-\frac{\Delta t}{2}\sum_{i=1}^{N}\zeta_i\right) \tag{B1}$$

where ζ_i is

$$\zeta_i = (z_i^T\Sigma_i^{-1}z_i - (z_i - \mu_i)^T(A_{t_i}^T\Sigma_i A_{t_i})^{-1}(z_i - \mu_i))$$

Using basic rules of determinants, it is easy to see that the term outside the exponent reduces to

$$\prod_{j=1}^{N}\frac{(2\pi)^{n/2}|A_j^T\Sigma_j A_j|^{1/2})}{(2\pi)^{n/2}|\Sigma_j|^{1/2})} = \prod_{j=1}^{N}|A_j|$$

So, we need only show that ζ_i reduces to Q_i. Observe that, at every time step, we have the difference between two quadratic functions of z_i, so we can complete the square to combine this into a single quadratic function. If we recall the definition of Γ_i from Eq. (30) and define

$$\Lambda_i = A_{t_i}^T\Sigma_i A_{t_i}$$

then completing the square yields the following

$$\begin{aligned}\zeta_i = {} & (z_i + \Gamma_i\Lambda_i^{-1}\mu_i)^T\Gamma_i^{-1}(z_i + \Gamma_i\Lambda_i^{-1}\mu_i) - \mu_i^T\Lambda_i^{-1}\mu_i \\ & - (\Gamma_i\Lambda_i^{-1}\mu_i)^T\Gamma_i^{-1}(\Gamma_\iota\Lambda_i^{-1}\mu_i)\end{aligned}$$

Now, we expand out the first quadratic term to get

$$\begin{aligned}\zeta_i = {} & z_i^T\Gamma_i^{-1}z_i + 2\mu_i^T\Lambda_i^{-1}z_i + \underline{\mu_i^T\Lambda_i^{-1}\Gamma_i\Lambda_i^{-1}\mu_i} - \mu_i^T\Lambda_i^{-1}\mu_i \\ & - \underline{(\Gamma_i\Lambda_i^{-1}\mu_i)^T\Gamma_i^{-1}(\Gamma_i\Lambda_i^{-1}\mu_i)}\end{aligned}$$

Notice that the two underlined terms are the same, except for the sign, so they cancel out and we are left with

$$\zeta_i = z_i^T\Gamma_i^{-1}z_i + 2\mu_i^T\Lambda_i^{-1}z_i - \mu_i^T\Lambda_i^{-1}\mu_i$$

Now, define $\tilde{z}_i = z_i - \mu_i$; then, rewrite this equation in terms of $\tilde{z}_i$:

$$\zeta_i = (\tilde{z}_i + \mu_i)^T\Gamma_i^{-1}(\tilde{z}_i + \mu_i) + 2\mu_i^T\Lambda_i^{-1}(\tilde{z}_i + \mu_i) - \mu_i^T\Lambda_i^{-1}\mu_i$$

which expands out to

$$\zeta_i = \tilde{z}_i^T\Gamma_i^{-1}\tilde{z}_i + 2\mu_i^T\Gamma_i^{-1}\tilde{z}_i + \mu_i^T\Gamma_i^{-1}\mu_i + 2\mu_i^T\Lambda_i^{-1}\tilde{z}_i + 2\mu_i^T\Lambda_i^{-1}\mu_i - \mu_i^T\Lambda_i^{-1}\mu_i$$

which then simplifies to

$$\zeta_i = \tilde{z}_i^T\Gamma_i^{-1}\tilde{z}_i + 2\mu_i^T\Gamma_i^{-1}\tilde{z}_i + \mu_i^T\Gamma_i^{-1}\mu_i + 2\mu_i^T\Lambda_i^{-1}\tilde{z}_i + \mu_i^T\Lambda_i^{-1}\mu_i$$

Now, recall that

$$\Gamma_i = (\Sigma_i^{-1} - \Lambda_i^{-1})^{-1}$$

so we can split the quadratic terms in Γ_i^{-1} into the Σ_i^{-1} and Λ_i^{-1} components. Doing this yields

$$\zeta_i = \tilde{z}_i^T\Gamma_i^{-1}\tilde{z}_i + 2\mu_i^T\Sigma_i^{-1}\tilde{z}_i - \underline{2\mu_i^T\Lambda_i^{-1}\tilde{z}_i} + \mu_i^T\Sigma_i^{-1}\mu_i - \underline{\mu_i^T\Lambda_i^{-1}\mu_i} + \underline{2\mu_i^T\Lambda_i^{-1}\tilde{z}_i} + \underline{\mu_i^T\Lambda_i^{-1}\mu_i}$$

and, by noting that the underlined terms cancel out, we see that we are left with

$$\zeta_i = \tilde{z}_i^T\Gamma_i^{-1}\tilde{z}_i + 2\mu_i^T\Sigma_i^{-1}\tilde{z}_i + \mu_i^T\Sigma_i^{-1}\mu_i$$

which is the same as

$$(z_i - \mu_i)^T\Gamma_i^{-1}(z_i - \mu_i) + 2\mu_i^T\Sigma_i^{-1}(z_i - \mu_i) + \mu_i^T\Sigma_i^{-1}\mu_i \qquad \text{(B2)}$$

and so $\zeta_i = Q_i$, which completes the proof.

Appendix C: Cart–Pole Dynamics

The state of the cart–pole is described by the location of the cart x, the velocity of the cart $\dot{x}$, the angle of the pole with the vertical θ, and the angular velocity of the pole $\dot{\theta}$. An additional state f is used to model the horizontal force supplied to the cart via a simple motor. The full state equations for the analytic model of the cart–pole dynamics are given as follows:

$$\ddot{x} = \frac{1}{m_c + m_p\sin^2\theta}(f + m_p\sin\theta(l\dot{\theta}^2 + g\cos\theta))$$

$$\ddot{\theta} = \frac{1}{l(m_c + m_p\sin^2\theta)}(-f\cos\theta - m_p l\dot{\theta}^2\cos\theta\sin\theta - (m_c + m_p)g\sin\theta)$$

$$\dot{f} = 20(f_{\text{des}} - f)$$

where f_{des} is the desired motor force, $g = 9.81\ \text{m/s}^2$ is the gravitational acceleration, $m_c = 1.0$ kg is the mass of the cart, $m_p = 0.01$ kg is the mass of the pole, and $l = 0.25$ m is the length of the pole. The cart–pole swingup task requires the algorithm to bring the system from the initial state, stationary with the pole pointed straight down at the origin ($x = \dot{x} = \theta = \dot{\theta} = 0$), to the final state, stationary with the pole pointed straight up at the origin ($x = \dot{x} = \dot{\theta} = 0, \theta = \pi$).

Appendix D: Race Car Dynamics

We used an analytic model of vehicle dynamics derived in [32] as the ground truth model for the simulator. This model makes the simplifying assumption that the two front tires and two back tires are lumped into one tire at the front and the back (and is therefore known as a bicycle vehicle model); this assumption makes tractable the incorporation of lateral tire forces into the model, which are an essential aspect of car dynamics when operating at high speeds. The full state equations for the model are given as follows:

$$\dot{x} = v_x\cos(\psi) - v_y\sin(\psi), \quad \dot{y} = v_x\sin(\psi) + v_y\cos(\psi),$$

$$\dot{\psi} = r \quad \dot{\beta} = \frac{F_{yF} + F_{yR}}{Mv_x} - r, \quad \dot{v}_x = \frac{F_x - F_{yF}\sin(\delta)}{M} + rv_x\beta,$$

$$\dot{v}_y = \frac{F_{yF} + F_{yR}}{M} - rv_x \quad \dot{r} = \frac{aF_{yF} - bF_{yR}}{I_z},$$

$$\dot{\delta} = 10(\delta - \delta^{\text{des}}), \quad \dot{F}_x = 10(F_x - F_x^{\text{des}})$$

where (x, y) is position, ψ is the heading, β is the sideslip angle, (v_x, v_y) are longitudinal and lateral velocities in the body frame of the vehicle, r is the heading (yaw) rate, δ is the steering angle, and F_x is the longitudinal force imparted by the rear wheels. The inputs to the model are desired commands δ^{des} and F_x^{des}. Also, (a, b) are the distances from the center of mass to the front and rear axles. The terms F_{yF} and F_{yR} are the lateral tire forces imparted by the front and rear wheels, respectively. This force is a function of the slip angle α, which we compute based on a brush tire model:

$$\alpha_F = \tan^{-1}\left(\beta + a\frac{r}{v_x}\right) - \delta, \qquad \alpha_R = \tan^{-1}\left(\beta - b\frac{r}{v_x}\right)$$

and then the lateral forces are

$$F_{yi} = \begin{cases} -C\tan(\alpha_i) + \frac{C^2}{3\xi\mu F_z}\tan(\alpha_i)^3\left|\tan(\alpha_i)\right| - \frac{C^3}{27\mu^2\xi^2F_z^2}\tan(\alpha_i)^3 \\ -\mu\xi F_z\frac{|\alpha_i|}{\alpha_i} \quad \text{if } \alpha_i \geq \gamma_i \end{cases}$$

Here, C is the cornering stiffness of the tire, and μ is the coefficient of friction between the tire and the ground. These values are set to 1200 and 0.55, respectively. The terms γ and ξ are computed as follows:

$$\xi = \left(\frac{\mu^2F_z^2 - F_x^2}{\mu F_z}\right)^{1/2}, \qquad \gamma = \left|\tan^{-1}\left(3\xi F_z\frac{|\alpha|}{\alpha}\right)\right| \qquad \text{(D1)}$$

Appendix E: Quadrotor Dynamics

The dynamic model of the quadrotor was introduced in [33]. The model includes 16 states: three for position $\boldsymbol{r}$, three for translational velocity $\dot{\boldsymbol{r}}$, three for Euler angles $\boldsymbol{\Phi}$, three for the body angular rates $\boldsymbol{\omega}$, and four for motor speeds $\boldsymbol{\Omega}$. The full state equations are given as follows:

$$\boldsymbol{\Phi} = \begin{pmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{pmatrix}^{-1}\begin{pmatrix} p \\ q \\ r \end{pmatrix},$$

$$\ddot{\boldsymbol{r}} = \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} + R\begin{pmatrix} 0 \\ 0 \\ \frac{1}{m}\sum F_i \end{pmatrix}$$

$$\boldsymbol{\omega} = I^{-1}\left[\begin{pmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{pmatrix} - \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times I\begin{pmatrix} p \\ q \\ r \end{pmatrix}\right],$$

$$\boldsymbol{\Omega} = k_m\left[\begin{pmatrix} \omega_1^{\text{des}} \\ \omega_2^{\text{des}} \\ \omega_3^{\text{des}} \\ \omega_4^{\text{des}} \end{pmatrix} - \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix}\right]$$

where ϕ, θ, and ψ are the components of $\boldsymbol{\Phi}$; p, q, and r are the components of $\boldsymbol{\omega}$; and ω_1, ω_2, ω_3 and ω_4 are the components of $\boldsymbol{\Omega}$. Additionally, $m = 0.25$ kg is the mass of the quadrotor, $g = 9.81\ \text{m/s}^2$ is gravitational acceleration, $L = 0.1$ m is the length of the moment arm from the body of the quadrotor to the motor,

$$I = \text{diag}(2.5 \times 10^{-4}, 2.5 \times 10^{-4}, 1.2 \times 10^{-3}) \text{ kg} \cdot \text{m}^2$$

is the mass moment of inertia matrix for the quadrotor in the body frame, and $k_m = 20$ is the motor constant. The forces and moments F_i and M_i used previously are given by

$$F_i = k_F \omega_i^2 \quad M_i = k_M \omega_i^2$$

where k_F is the motor force constant equal to 6.11×10^{-8} N/rpm^2, and k_M is the motor moment constant equal to 1.5×10^{-9} N $\cdot$ m/rpm^2. The rotation matrix R between the body and inertial frames is given by

$$R = \begin{pmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{pmatrix}$$

Acknowledgment

This research was partially supported by National Science Foundation grant no. NRI-1426945.

References

[1] Theodorou, E. A., Buchli, J., and Schaal, S., "A Generalized Path Integral Approach to Reinforcement Learning," *Journal of Machine Learning Research*, Vol. 11, Dec. 2010, pp. 3137–3181.

[2] Kappen, H. J., "Linear Theory for Control of Nonlinear Stochastic Systems," *Physical Review Letters*, Vol. 95, No. 20, Nov. 2005, Paper 200201.
doi:10.1103/PhysRevLett.95.200201

[3] Theodorou, E. A., "Nonlinear Stochastic Control and Information Theoretic Dualities: Connections, Interdependencies and Thermodynamic Interpretations," *Entropy*, Vol. 17, No. 5, 2015, pp. 3352–3375.
doi:10.3390/e17053352

[4] Karatzas, I., and Shreve, S. E., *Brownian Motion and Stochastic Calculus*, 2nd ed., Graduate Texts in Mathematics, Springer, New York, Aug. 1991.

[5] Friedman, A., *Stochastic Differential Equations and Applications*, Academic Press, New York, 1975.

[6] Theodorou, E. A., and Todorov, E., "Relative Entropy and Free Energy Dualities: Connections to Path Integral and KL Control," *Proceedings of IEEE Conference on Decision and Control*, IEEE Publ., Piscataway, NJ, Dec. 2012, pp. 1466–1473.

[7] Gómez, V., Kappen, H. J., Peters, J., and Neumann, G., "Policy Search for Path Integral Control," *Machine Learning and Knowledge Discovery in Databases*, Springer, New York, 2014, pp. 482–497.

[8] Thijssen, S., and Kappen, H. J., "Path Integral Control and State-Dependent Feedback," *Physical Review E*, Vol. 91, No. 3, 2015, Paper 032104,
doi:10.1103/PhysRevE.91.032104

[9] Rombokas, E., Malhotra, M., Theodorou, E. A., Todorov, E., and Matsuoka, Y., "Reinforcement Learning and Synergistic Control of the Act Hand," *IEEE/ASME Transactions on Mechatronics*, Vol. 18, No. 2, 2013, pp. 569–577.
doi:10.1109/TMECH.2012.2219880

[10] Gómez, V., Thijssen, S., Symington, A., Hailes, S., and Kappen, H. J., "Real-Time Stochastic Optimal Control for Multi-Agent Quadrotor Systems," *Proceeding of the International Conference on Automated Planning and Scheduling (ICAPS)*, June 2016, pp. 468–476.

[11] Williams, G., Rombokas, E., and Daniel, T., "GPU Based Path Integral Control with Learned Dynamics," *Neural Information Processing Systems — ALR Workshop*, 2014.

[12] Stulp, F., Buchli, J., Theodorou, E. A., and Schaal, S., "Reinforcement Learning of Full-Body Humanoid Motor Skills," *Proceedings of 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, IEEE Publ., Piscataway, NJ, Dec. 2010, pp. 405–410.

[13] Jacobson, D. H., and Mayne, D. Q., *Differential Dynamic Programming*, Elsevier, New York, 1970.

[14] Theodorou, E. A., Tassa, Y., and Todorov, E., "Stochastic Differential Dynamic Programming," *Proceedings of the American Control Conference (ACC)*, IEEE Publ., Piscataway, NJ, June 2010, pp. 1125–1132.

[15] Todorov, E., and Li, W., "A Generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems," *Proceedings of the American Control Conference (ACC)*, IEEE Publ., Piscataway, NJ, June 2005, pp. 300–306.

[16] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G., *Robotics: Modelling, Planning and Control*, Advanced Textbooks in Control and Signal Processing, Springer, London, 2010.

[17] Murray, R. M., Li, Z., Sastry, S. S., and Sastry, S. S., *A Mathematical Introduction to Robotic Manipulation*, Taylor and Francis, Washington, D.C., 1994.

[18] Markley, F. L., and Crassidis, J. L., *Fundamentals of Spacecraft Attitude Determination and Control*, Space Technology Library, Springer, New York, 2014.

[19] Valero-Cuevas, F. J., Hoffmann, H., Kurse, M. U., Kutch, J. J., and Theodorou, E. A., "Computational Models for Neuromuscular Function," *IEEE Reviews in Biomedical Engineering*, Vol. 2, Dec. 2009, pp. 110–135.
doi:10.1109/RBME.2009.2034981

[20] Stengel, R. F., *Optimal Control and Estimation*, Dover Books on Advanced Mathematics, Dover, New York, 1994.

[21] Fleming, W. H., and Soner, H. M., *Controlled Markov Processes and Viscosity Solutions*, 2nd ed., Applications of Mathematics, Springer, New York, 2006.

[22] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T., "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, Vol. 50, No. 2, Feb. 2002, pp. 174–188.

[23] Särkkä, S., and Sottinen, T., "Application of Girsanov Theorem to Particle Filtering of Discretely Observed Continuous-Time Non-Linear Systems," *Bayesian Analysis*, Vol. 3, No. 3, 2008, pp. 555–584.

[24] Särkkä, S., and Moulines, E., "On the LP-Convergence of a Girsanov Theorem Based Particle Filter," *IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE Publ., Piscataway, NJ, 2016, pp. 3989–3993.

[25] Van den Broek, B., Wiegerinck, W., and Kappen, H. J., "Graphical Model Inference in Optimal Control of Stochastic Multi-Agent Systems," *Journal of Artificial Intelligence Research*, Vol. 32, No. 1, 2008, pp. 95–122.

[26] Anderson, R. P., and Milutinovic, D., "Distributed Path Integral Feedback Control Based on Kalman Smoothing for Unicycle Formations," *Proceedings of the American Control Conference (ACC)*, June 2013, pp. 4611–4616.

[27] Cvetanoska, V., and Stojanoski, T., "High Performance Computing and Monte Carlo Simulation for Pricing American Options," *Proceedings of the 9th conference for Informatics and Information Technology (CIIT)*, 2012, pp. 170–174.

[28] Quinn, J. C., and Abarbanel, H. D. I., "Data Assimilation Using a GPU Accelerated Path Integral Monte Carlo Approach," *Journal of Computational Physics*, Vol. 230, No. 22, 2011, pp. 8168–8178.
doi:10.1016/j.jcp.2011.07.015

[29] Ahlén, O., Bohlin, G., Carlsson, K., Gren, M., Holmvall, P., and Säterskog, P., "GPU Implementation of the Feynman Path-Integral Method in Quantum Mechanics," Bachelor Thesis No. FUFX02-11-01, Department of Fundamental Physics, Division of Subatomic Physics, Chalmers Univ. of Technology, Gothenburg, Sweden, 2011.

[30] Sanders, J., and Kandrot, E., *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley Professional, Reading, MA, 2010.

[31] "Cuda Curand Library," Corporation, Software Package, Nvidia CUDA Toolkit 4.0, http://developer.nvidia.com/cuda-toolkit-40.

[32] Hindiyeh, R. Y., "Dynamics and Control of Drifting in Automobiles," Ph.D. Thesis, Stanford Univ., Stanford, CA, March 2013.

[33] Michael, N., Mellinger, D., Lindsey, Q., and Kumar, V., "The Grasp Multiple Micro-UAV Testbed," *IEEE Robotics and Automation Magazine*, Vol. 17, No. 3, 2010, pp. 56–65.