2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Aggressive Driving with Model Predictive Path Integral Control

Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou

Abstract— **In this paper we present a model predictive control algorithm designed for optimizing non-linear systems subject to complex cost criteria. The algorithm is based on a stochastic optimal control framework using a fundamental relationship between the information theoretic notions of free energy and relative entropy. The optimal controls in this setting take the form of a path integral, which we approximate using an efficient importance sampling scheme. We experimentally verify the algorithm by implementing it on a Graphics Processing Unit (GPU) and apply it to the problem of controlling a fifth-scale Auto-Rally vehicle in an aggressive driving task.**

I. INTRODUCTION

Aggressive driving at the limits of handling is a challenging robotics problem which has received significant attention in recent years. Most current algorithms approach the problem hierarchically: first plan a trajectory considering only the position and velocity of the vehicle, and then use a simple feedback controller to track the planned trajectory. This approach works well for vehicles operating in non-aggressive regions of the state space and was used by the winning teams in the DARPA Grand Challenges in 2005 and 2007 [1], [2].

The hierarchical approach has also been applied to the more challenging problem of controlling a vehicle near its limit of friction. In this case it is necessary to ensure that a planned path is feasible, which can be very difficult. If the coefficient of friction between the road and vehicle is known then optimal velocity profiles can be found [3] for pre-specified paths. And in the case where both the desired path and a feasible velocity profile are given, several methods exist for robustly tracking the given path [4], [5]. A method for high speed maneuvers which does not simply track a pre-specified trajectory is developed and deployed on small-scale vehicles in a laboratory environment in [6]. The algorithm relies on a series of pre-specified waypoints, and the planner finds a feasible interpolation between the waypoints which a model predictive controller then tracks.

These approaches highlight some of the advantages and disadvantages of the hierarchical paradigm of splitting control and planning. Excellent performance (e.g. the DARPA Grand Challenge) can be achieved if the problem specifications are safely within the handling limits of the vehicle, or if the path and velocity profile are determined by the problem structure. However, generating paths which are aggressive and feasible is a difficult constrained optimization problem, and knowing the constraint requires precise knowledge of the system dynamics in the form of the coefficient of friction.

Another approach, which we consider here, is the optimal control approach which finds a sequence of controls or a control law that is optimal with respect to a given cost and system dynamics. This approach combines planning and execution into one phase by explicitly taking the dynamics into account during optimization. The problem with the optimal control approach is that, for systems with non-linear dynamics and costs, solving the optimal control problem is computationally demanding and cannot typically be done on the fly. There have been approaches which solve for the optimal control inputs offline. For instance in [7] cornering is posed as a minimum time problem and analyzed offline. A similar method is [8], which solves for an optimal open loop control sequence offline, and then applies an LQR controller to stabilize the vehicle about the open loop trajectory. An approach for performing aggressive sliding maneuvers in order to avoid collisions is developed in [9], where optimal trajectories are generated offline for a variety of initial conditions and then a feedback controller is synthesized using Gaussian Process regression. In all of these methods the bulk of the computation is performed offline, which means that new behaviors and maneuvers cannot be produced on the fly. In contrast, the method developed here is able to generate new trajectories in real-time, which allows it to handle a wide range of scenarios.

The method we develop is a model predictive control algorithm based on the path integral control framework. This framework combines the best of the hierarchical and optimal control paradigms. We do not split the problem into a planning and execution phase, which allows for a simple problem formulation and optimal behavior with respect to the system dynamics. However, unlike in previous optimal control approaches, we perform all of the computation on the fly by leveraging recent advances in GPU computing.

II. STOCHASTIC TRAJECTORY OPTIMIZATION

In this section we derive a method for stochastic trajectory optimization based on the path integral control framework. This method serves as the basis for our model predictive control algorithm. Path integral control [10] provides a mathematically sound methodology for developing optimal control algorithms based on stochastic sampling of trajectories. One appealing aspect of algorithms developed in the path integral framework is that they require no derivatives of either the dynamics or cost. This provides flexibility and robustness when estimating the system dynamics and designing cost functions.

This research has been supported by NSF Grant No. NRI-1426945, ARO through MURI award W911NF-11-1-0046, and DURIP award W911NF-12-1-0377. The authors are with Institute for Robotics and Intelligent Machines at the Georgia Institute of Technology, Atlanta, GA, USA. Email: gradyrw@gatech.edu

The typical derivation of path integral control relies on an exponential transformation of the value function of the optimal control problem. Additionally, it requires that the noise in the system only affects directly actuated states. The exponential transform, along with the noise assumption, transforms the stochastic Hamilton-Jacobi-Bellman equation into a linear partial differential equation which can then be transformed into a path integral using the Feynman-Kac lemma. The ensuing path integral takes the form of an expectation over trajectories under the uncontrolled dynamics of the system, which is then approximated using Monte-Carlo sampling to compute the control for the current state. This is the approach used in the path integral control frameworks of: [10]–[15].

The derivation we provide here provides two benefits over the standard path integral approach: (1) It relaxes the condition between the noise and control authority to allow for noise in both indirectly and directly actuated states, and (2) It explicitly provides a formula for the optimal controls over the entire time horizon instead of only the initial time. The key insight which allows for this is the use of a fundamental relationship [16], [17] between the information theoretic notions of free energy and relative entropy (also know as KL-Divergence). This relationship can be used to derive the form of an optimal probability distribution over trajectories. We can then solve the optimal control problem by directly minimizing the relative entropy between the optimal distribution and the distribution induced by the controller.

This approach can be viewed as the cross-entropy method applied to stochastic diffusion processes, which was originally studied in [18]. In that work, the strategy was used to fit the parameters of a feedback-control policy offline. In contrast, we do not restrict the controls to a pre-specified policy and perform all computation online. Two crucial concepts in this approach are the Radon-Nikodym derivative, which relates two probability measures to each other, and Girsanov's theorem which provides the form of the Radon-Nikodym derivative for systems subject to Brownian disturbances.

A. Problem Formulation

We consider stochastic dynamical systems with the state and controls at time t denoted $\mathbf{x}_t \in \mathbb{R}^n$ and $\mathbf{u}_t \in \mathbb{R}^m$. These dynamics are disturbed by the Brownian motion $\mathrm{d}\mathbf{w} \in \mathbb{R}^p$. We also define $\mathbf{u}(\cdot) : [t_0, T] \to \mathbb{R}^m$ as the function which maps time to control inputs, and define the function $\tau : [t_0, T] \to \mathbb{R}^n$ as a trajectory of the system. In the classical stochastic optimal control setting we seek a control sequence $\mathbf{u}(\cdot)$ such that:

$$\mathbf{u}^*(\cdot) = \underset{\mathbf{u}(\cdot)}{\operatorname{argmin}} \mathbb{E}_{\mathbb{Q}}\left[\phi(\mathbf{x}_T, T) + \int_{t_0}^{T} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t, t)\mathrm{d}t\right] \quad (1)$$

where the expectation is taken with respect to the dynamics: $\mathrm{d}\mathbf{x} = \mathbf{F}(\mathbf{x}_t, \mathbf{u}_t, t)\mathrm{d}t + \mathbf{B}(\mathbf{x}_t, t)\mathrm{d}\mathbf{w}$. In this paper we consider costs composed of an arbitrary state-dependent term and a quadratic control cost:

$$\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t, t) = q(\mathbf{x}_t, t) + \frac{1}{2}\mathbf{u}_t^{\mathrm{T}}\mathbf{R}(\mathbf{x}_t, t)\mathbf{u}_t \quad (2)$$

and dynamics which are affine in controls:

$$\mathbf{F}(\mathbf{x}_t, \mathbf{u}_t, t) = \mathbf{f}(\mathbf{x}_t, t) + \mathbf{G}(\mathbf{x}_t, t)\mathbf{u}_t \quad (3)$$

In [16] an interpretation of path integral control is given in terms of the information theoretic concepts of free energy and relative entropy. The interpretation is based on the following equality:

$$-\lambda\mathcal{F}(S(\tau)) = \inf_{\mathbb{Q}}\left[\mathbb{E}_{\mathbb{Q}}[S(\tau)] + \lambda\mathbb{D}_{KL}\left(\mathbb{Q} \parallel \mathbb{P}\right)\right] \quad (4)$$

In this equation $\lambda \in \mathbb{R}^+$, $S(\tau)$ is defined as the state-dependent cost-to-go term $\phi(\mathbf{x}_T, T) + \int_{t_0}^{T} q(\mathbf{x}_t, t)\mathrm{d}t$, the free energy $\mathcal{F}(S(\tau))$ is defined as: $\log\left(\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(\tau)\right)\right]\right)$, $\mathbb{P}$ is the probability measure over the space of trajectories induced by the uncontrolled stochastic dynamics: $\mathbf{f}(\mathbf{x}_t, t)\mathrm{d}t + \mathbf{B}(\mathbf{x}_t, t)\mathrm{d}\mathbf{w}$, and $\mathbb{Q}$ is any probability measure defined over the space of trajectories such that $\mathbb{Q}$ is absolutely continuous with $\mathbb{P}$ (i.e. they agree on which sets have measure zero). The $\mathbb{D}_{KL}\left(\mathbb{Q} \parallel \mathbb{P}\right)$ denotes relative entropy between $\mathbb{Q}$ and $\mathbb{P}$ which is defined as $\mathbb{E}_{\mathbb{Q}}\left[\log\left(\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}\right)\right]$.

The controlled dynamics induce another probability measure on the space of trajectories, which we denote $\mathbb{Q}(\mathbf{u})$. The relative entropy term between the uncontrolled distribution $\mathbb{P}$ and the controlled distribution $\mathbb{Q}(\mathbf{u})$ can be computed by applying Girsanov's theorem. The result is that:

$$\mathbb{D}_{KL}\left(\mathbb{Q}(\mathbf{u}) \parallel \mathbb{P}\right) = \frac{1}{2}\int_{t_0}^{T} \mathbf{u}_t^{\mathrm{T}}\mathbf{G}(\mathbf{x}_t, t)^{\mathrm{T}}\Sigma(\mathbf{x}_t, t)^{-1}\mathbf{G}(\mathbf{x}_t, t)\mathbf{u}_t\mathrm{d}t \quad (5)$$

Where $\Sigma(\mathbf{x}_t, t) = \mathbf{B}(\mathbf{x}_t, t)\mathbf{B}(\mathbf{x}_t, t)^{\mathrm{T}}$. Applying this result, and if we assume that the control cost matrix takes the form:

$$\mathbf{R}(\mathbf{x}_t, t) = \lambda\mathbf{G}(\mathbf{x}_t, t)^{\mathrm{T}}\Sigma(\mathbf{x}_t, t)^{-1}\mathbf{G}(\mathbf{x}_t, t) \quad (6)$$

we get the following correspondence between the right hand side of (4) and (1):

$$\mathbb{E}_{\mathbb{Q}(\mathbf{u})}[S(\tau)] + \lambda\mathbb{D}_{KL}\left(\mathbb{Q} \parallel \mathbb{P}\right) = \mathbb{E}_{\mathbb{Q}(\mathbf{u})}\left[S(\tau) + \frac{1}{2}\int_{t_0}^{T} \mathbf{u}_t^{\mathrm{T}}\mathbf{R}(\mathbf{x}_t, t)\mathbf{u}_t\mathrm{d}t\right] \quad (7)$$

Moreover, [16] explicitly derives the form of the optimal probability measure $\mathbb{Q}^*$ in terms of the Radon-Nikodym derivative[1] with respect to the uncontrolled dynamics. This takes the form:

$$\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}} = \frac{\exp\left(-\frac{1}{\lambda}S(\tau)\right)}{\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(\tau)\right)\right]} \quad (8)$$

Previous works, [16], [17], make these connections between the information theoretic notions of free energy, relative entropy, and classical optimal control theory. However, they

[1]The Radon-Nikodym derivative defines the measure $\mathbb{Q}^*$ by means of the equation $\mathbb{Q}^*(A) = \int_A \frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}}\mathrm{d}\mathbb{P}$

do not provide a method for computing a control law independent of the HJB-equation, as we do here.

The main idea in our approach is that since we have the form of the optimal distribution, it is possible to pursue the following optimization scheme: instead of trying to directly solve the optimal control problem by computing the solution to the stochastic HJB equation, we can solve the minimization problem defined by (4) by moving the probability distribution induced by the controller, $\mathbb{Q}(\mathbf{u})$, as close as possible to the optimal probability measure, $\mathbb{Q}^*$, defined by the Radon-Nikodym derivative $\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}}$. Using the relative entropy between $\mathbb{Q}^*$ and $\mathbb{Q}(\mathbf{u})$ as a notion of distance, we obtain the following minimization problem:

$$\mathbf{u}^*(\cdot) = \underset{\mathbf{u}(\cdot)}{\operatorname{argmin}} \, \mathbb{D}_{KL}\left(\mathbb{Q}^* \parallel \mathbb{Q}(\mathbf{u})\right) \tag{9}$$

This is an intuitively appealing strategy since the optimal probability measure $\mathbb{Q}^*$ takes the form of a softmax function with temperature λ. If a trajectory has low cost, then $\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}}$, has a high value. Therefore, if a trajectory is drawn at random from $\mathbb{Q}^*$, it is likely to have a very low cost. If $\mathbb{Q}(\mathbf{u})$ is able to closely approximate $\mathbb{Q}^*$, then applying the controls $\mathbf{u}(\cdot)$ to the system will have a high probability of generating a low cost trajectory. This objective is closely related to the Bayesian approximate inference approach to stochastic optimal control [19]. In that approach, the relative entropy between the controlled distribution and a target distribution is minimized ($\mathbb{D}_{KL}\left(\mathbb{Q}(\mathbf{u}) \parallel \mathbb{Q}^*\right)$ in our notation). However, since relative entropy is not symmetric, this leads to a very different optimization problem than what is derived here.

B. Relative Entropy Minimization

We seek to minimize the relative entropy between the optimal distribution $\mathbb{Q}^*$ and the distribution induced by the controller $\mathbb{Q}(\mathbf{u})$. Applying the definition of relative entropy we have:

$$\mathbb{D}_{KL}\left(\mathbb{Q}^* \parallel \mathbb{Q}(\mathbf{u})\right) = \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{Q}(\mathbf{u})}\right)\right] \tag{10}$$

In order to optimize this function we need to find an expression for the Radon-Nikodym derivative $\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{Q}(\mathbf{u})}$. To do this, we apply the chain rule property of Radon-Nikodym derivatives:

$$\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{Q}(\mathbf{u})} = \frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}}\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}(\mathbf{u})} \tag{11}$$

We know what $\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}}$ is from the definition of $\mathbb{Q}^*$, and we can apply Girsanov's theorem again to compute $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}(\mathbf{u})}$ as:

$$\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}(\mathbf{u})} = \exp(\mathcal{D}(\tau, \mathbf{u}(\cdot))) \tag{12}$$

with $\mathcal{D}(\tau, \mathbf{u}(\cdot))$ defined as:

$$\begin{aligned}\mathcal{D}(\tau, \mathbf{u}(\cdot)) = &-\int_0^T \mathbf{u}_t^{\mathrm{T}}\mathbf{G}(\mathbf{x}_t, t)^{\mathrm{T}}\Sigma(\mathbf{x}_t, t)^{-1}\mathbf{B}(\mathbf{x}_t, t)\mathrm{d}\mathbf{w}^{(0)} \\ &+ \frac{1}{2}\int_0^T \mathbf{u}_t^{\mathrm{T}}\mathbf{G}(\mathbf{x}_t, t)^{\mathrm{T}}\Sigma(\mathbf{x}_t, t)^{-1}\mathbf{G}(\mathbf{x}_t, t)\mathbf{u}_t\mathrm{d}t\end{aligned} \tag{13}$$

Where $\mathrm{d}\mathbf{w}^{(0)}$ is a Brownian motion with respect to $\mathbb{P}$ (i.e. $\mathbb{E}_{\mathbb{P}}\left[\int_0^t \mathrm{d}\mathbf{w}^{(0)}\right] = 0, \ \forall t$). Combining this with (8), we get the following expression for $\mathbb{D}_{KL}\left(\mathbb{Q}^* \parallel \mathbb{Q}(\mathbf{u})\right)$:

$$\mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{\exp(-\frac{1}{\lambda}S(\tau))\exp(\mathcal{D}(\tau, \mathbf{u}(\cdot)))}{\mathbb{E}_{\mathbb{P}}\left[\exp(-\frac{1}{\lambda}S(\tau))\right]}\right)\right] \tag{14}$$

Then using basic rules of logarithms and exponents we can re-write this as:

$$\mathbb{E}_{\mathbb{Q}^*}\left[-\tfrac{1}{\lambda}S(\tau) + \mathcal{D}(\tau, \mathbf{u}(\cdot)) - \log\left(\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\tfrac{1}{\lambda}S(\tau)\right)\right]\right)\right] \tag{15}$$

Since $S(\tau)$, does not depend on the controls $\mathbf{u}(\cdot)$, we can remove the first and last terms from the minimization to get:

$$\underset{\mathbf{u}(\cdot)}{\operatorname{argmin}} \, \mathbb{D}_{KL}\left(\mathbb{Q}^* \parallel \mathbb{Q}(\mathbf{u})\right) = \underset{\mathbf{u}(\cdot)}{\operatorname{argmin}} \, \mathbb{E}_{\mathbb{Q}^*}[\mathcal{D}(\tau, \mathbf{u}(\cdot))] \tag{16}$$

The goal is to find the function $\mathbf{u}^*(\cdot)$ which minimizes (16). However, since we inevitably apply the control in discrete time it suffices to consider the class of step functions:

$$\mathbf{u}_t = \begin{cases} \vdots \\ \mathbf{u}_j \quad \text{if } j\Delta t \le t < (j+1)\Delta t \\ \vdots \end{cases} \tag{17}$$

With $j = \{0, 1, \ldots N\}$. Applying this parameterization to $\mathcal{D}(\tau, \mathbf{u}(\cdot))$ yields:

$$-\sum_{j=0}^{N} \mathbf{u}_j^{\mathrm{T}} \int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{G}}(\mathbf{x}_t, t)\mathrm{d}\mathbf{w}^{(0)} + \frac{1}{2}\mathbf{u}_j^{\mathrm{T}} \int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{H}}(\mathbf{x}_t, t)\mathrm{d}t \ \mathbf{u}_j \tag{18}$$

Where we have denoted:

i) $\boldsymbol{\mathcal{G}}(\mathbf{x}, t) = \mathbf{G}(\mathbf{x}, t)^{\mathrm{T}}\Sigma(\mathbf{x}, t)^{-1}\mathbf{B}(\mathbf{x}, t)$
ii) $\boldsymbol{\mathcal{H}}(\mathbf{x}, t) = \mathbf{G}(\mathbf{x}, t)^{\mathrm{T}}\Sigma(\mathbf{x}, t)^{-1}\mathbf{G}(\mathbf{x}, t)$
iii) $N = T/\Delta t$.

By noting that each $\mathbf{u}_j$ does not depend on the trajectory taken, when we apply the expectation operator we have $\mathbb{E}_{\mathbb{Q}^*}[\mathcal{D}(\tau, \mathbf{u}(\cdot))]$ equal to the following:

$$\begin{aligned}\mathbb{E}_{\mathbb{Q}^*}[\mathcal{D}(\tau, \mathbf{u}(\cdot))] = &-\sum_{j=0}^{N} \mathbf{u}_j^{\mathrm{T}}\mathbb{E}_{\mathbb{Q}^*}\left[\int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{G}}(\mathbf{x}_t, t)\mathrm{d}\mathbf{w}^{(0)}\right] \\ &+ \sum_{j=0}^{N} \frac{1}{2}\mathbf{u}_j^{\mathrm{T}}\mathbb{E}_{\mathbb{Q}^*}\left[\int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{H}}(\mathbf{x}_t, t)\mathrm{d}t\right]\mathbf{u}_j\end{aligned} \tag{19}$$

Now it's easy to see that this is convex with respect to each $\mathbf{u}_j$, so to find $\mathbf{u}_j^*$ we can take the gradient of (19) with respect to $\mathbf{u}_j$, set it equal to zero and solve for $\mathbf{u}_j$. Doing this yields:

$$\mathbf{u}_j^* = \mathbb{E}_{\mathbb{Q}^*}\left[\int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{H}}(\mathbf{x}_t, t)\mathrm{d}t\right]^{-1}\mathbb{E}_{\mathbb{Q}^*}\left[\int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{G}}(\mathbf{x}_t, t)\mathrm{d}\mathbf{w}^{(0)}\right] \tag{20}$$

For small Δt we can make the approximations that:

$$\begin{aligned}\int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{H}}(\mathbf{x}_t, t)\mathrm{d}t &\approx \boldsymbol{\mathcal{H}}(\mathbf{x}_{t_j}, t_j)\Delta t \\ \int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{G}}(\mathbf{x}_t, t)\mathrm{d}\mathbf{w}^{(0)} &\approx \boldsymbol{\mathcal{G}}(\mathbf{x}_{t_j}, t_j)\int_{t_j}^{t_{j+1}} \mathrm{d}\mathbf{w}^{(0)}\end{aligned}$$

which allows us to obtain:

$$\mathbf{u}_j^* = \frac{1}{\Delta t}\mathbb{E}_{\mathbb{Q}^*}\left[\boldsymbol{\mathcal{H}}(\mathbf{x}_{t_j}, t_j)\right]^{-1}\mathbb{E}_{\mathbb{Q}^*}\left[\boldsymbol{\mathcal{G}}(\mathbf{x}_{t_j}, t_j)\int_{t_j}^{t_{j+1}} \mathrm{d}\mathbf{w}^{(0)}\right] \tag{21}$$

Since we cannot sample from the $\mathbb{Q}^*$ distribution, we need to change the expectation to be an expectation with respect to the uncontrolled dynamics, $\mathbb{P}$. We can then directly sample trajectories from $\mathbb{P}$ to approximate the controls. The change in expectation is achieved by applying the Radon-Nikodym derivative:

$$\mathbf{u}_j^* = \frac{1}{\Delta t}\mathbb{E}_{\mathbb{P}}\left[\frac{\exp\left(-\frac{1}{\lambda}S(\tau)\right)\boldsymbol{\mathcal{H}}(\mathbf{x}_{t_j}, t_j)}{\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(\tau)\right)\right]}\right]^{-1} \mathbb{E}_{\mathbb{P}}\left[\frac{\exp\left(-\frac{1}{\lambda}S(\tau)\right)\boldsymbol{\mathcal{G}}(\mathbf{x}_{t_j}, t_j)\int_{t_j}^{t_{j+1}} \mathrm{d}\mathbf{w}^{(0)}}{\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(\tau)\right)\right]}\right] \tag{22}$$

C. Special Case: State Independent Control Matrix

In this section we consider a special case of (22) which we use in all of our experiments. We suppose that the control matrix and diffusion matrix have the form:

$$\mathbf{G} = \begin{pmatrix} 0 \\ \mathbf{G}_c \end{pmatrix}, \quad \mathbf{B}(\mathbf{x}_t) = \begin{pmatrix} \mathbf{B}_a(\mathbf{x}_t) & 0 \\ 0 & \mathbf{B}_c \end{pmatrix} \tag{23}$$

In other words, there are no correlations between noise in the directly actuated and non-directly actuated states, and the diffusion for the directly actuated states is state-independent. In this case the covariance matrix becomes:

$$\Sigma(\mathbf{x}) = \begin{pmatrix} \mathbf{B}_a(\mathbf{x})\mathbf{B}_a(\mathbf{x})^{\mathrm{T}} & 0 \\ 0 & \mathbf{B}_c\mathbf{B}_c^{\mathrm{T}} \end{pmatrix} \tag{24}$$

And then the terms $\boldsymbol{\mathcal{H}}(\mathbf{x}_{t_j})$ and $\boldsymbol{\mathcal{G}}(\mathbf{x}_t)$ are no longer state dependent and reduce to:

$$\boldsymbol{\mathcal{H}} = \mathbf{G}_c^{\mathrm{T}}(\mathbf{B}_c\mathbf{B}_c^{\mathrm{T}})^{-1}\mathbf{G}_c \quad \boldsymbol{\mathcal{G}} = \mathbf{G}_c^{\mathrm{T}}(\mathbf{B}_c\mathbf{B}_c^{\mathrm{T}})^{-1}\mathbf{B}_c \tag{25}$$

And since all of these matrices are state-independent we can pull them out of the expectation to get:

$$\mathbf{u}_j^* = \frac{1}{\Delta t}\boldsymbol{\mathcal{H}}^{-1}\boldsymbol{\mathcal{G}}\left(\mathbb{E}_{\mathbb{P}}\left[\int_{t_j}^{t_{j+1}} \frac{\exp\left(-\frac{1}{\lambda}S(\tau)\right)\mathrm{d}\mathbf{w}^{(0)}}{\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(\tau)\right)\right]}\right]\right) \tag{26}$$

D. Numerical Approximation

In order to numerically approximate (26) there are two problems that need to be addressed. First, we need to re-write the equation for sampling in discrete time. Secondly, the expectation is with respect to the uncontrolled dynamics which in many cases is a very inefficient distribution to sample from (sampling from $\mathbb{P}$ corresponds to turning the machine on and waiting for the natural noise in the system to produce useful trajectories). So we need a way to perform importance sampling with (26).

In discrete time the dynamics of the system transform into:

$$\mathrm{d}\mathbf{x}_{t_j} = \left(\mathbf{f}(\mathbf{x}_{t_j}, t_j) + \mathbf{G}(\mathbf{x}_{t_j}, t_j)\mathbf{u}_j\right)\Delta t + \mathbf{B}(\mathbf{x}_{t_j}, t_j)\epsilon_j\sqrt{\Delta t} \tag{27}$$

Where ϵ_j is a vector where each entry is a standard normal random variable. Substituting $\epsilon_j\sqrt{\Delta t}$ into (26) yields:

$$\mathbf{u}_j^* = \frac{1}{\Delta t}\boldsymbol{\mathcal{H}}^{-1}\boldsymbol{\mathcal{G}}\left(\mathbb{E}_p\left[\frac{\exp\left(-\frac{1}{\lambda}S(\tau)\right)\epsilon_j\sqrt{\Delta t}}{\mathbb{E}_p\left[\exp\left(-\frac{1}{\lambda}S(\tau)\right)\right]}\right]\right) \tag{28}$$

where p is the probability distribution corresponding to the discrete time uncontrolled dynamics (i.e. Equation (27) with $\mathbf{u}_{t_j}$ set to zero). Instead of sampling from p we can choose to sample from a different probability distribution $q_{\mathbf{u}}^{\nu}$ which corresponds to sampling from the dynamics:

$$\mathrm{d}\mathbf{x}_{t_j} = \mathbf{f}(\mathbf{x}_{t_j}, t_j)\Delta t + \mathbf{G}(\mathbf{x}_{t_j}, t_j)\mathbf{u}_{t_j}\Delta t + \mathbf{B}_E(\mathbf{x}_{t_j}, t_j)\epsilon_j\sqrt{\Delta t} \tag{29}$$

The new diffusion matrix $\mathbf{B}_E$ is defined as:

$$\mathbf{B}_E(\mathbf{x}_t) = \begin{pmatrix} \mathbf{B}_a(\mathbf{x}_t) & 0 \\ 0 & \nu\mathbf{B}_c \end{pmatrix} \tag{30}$$

with $\nu \geq 1$. When sampling from this distribution the designer gets to choose:

i) The initial controls from which sampling is centered about.
ii) The magnitude of the exploration variance defined by ν.

In order to sample from $q_{\mathbf{u}}^{\nu}$ instead of p it's necessary to compute the likelihood ratio between the two distributions. We omit this derivation for brevity and refer the reader to [20]. Inserting the likelihood ratio corresponds to changing the running cost from $q(\mathbf{x}_t, t)$ to:

$$\tilde{q}(\mathbf{x}_t, \mathbf{u}_t, \epsilon_t, t) = q(\mathbf{x}_t, t) + \frac{1}{2}\mathbf{u}_t^{\mathrm{T}}\mathbf{R}\mathbf{u}_t + \lambda\mathbf{u}^{\mathrm{T}}\boldsymbol{\mathcal{G}}\frac{\epsilon}{\sqrt{\Delta t}} + \frac{1}{2}\lambda(1-\nu^{-1})\frac{\epsilon^{\mathrm{T}}}{\sqrt{\Delta t}}\mathbf{B}_c^{\mathrm{T}}\left(\mathbf{B}_c\mathbf{B}_c^{\mathrm{T}}\right)^{-1}\mathbf{B}_c\frac{\epsilon}{\sqrt{\Delta t}} \tag{31}$$

The first two additional terms can be interpreted as penalties for shifting the mean of the exploration away from zero, and the last term is a penalty for sampling from an over-aggressive variance. Note that $\mathbf{B}_c\frac{\epsilon}{\sqrt{\Delta t}}$ can be interpreted as the effective change in the control input due to noise.

The update rule (26) also changes when sampling from a distribution with non-zero control input. This is due to the random variable under consideration shifting from the zero mean term $\mathbf{B}_c\epsilon_j\sqrt{\Delta t}$ to the non-zero mean term: $\mathbf{G}_c\mathbf{u}\Delta t + \mathbf{B}_E\epsilon_j\sqrt{\Delta t}$. By substituting this term into (28) and simplifying, and if we denote $\tilde{S}(\tau)$ as:

$$\tilde{S}(\tau) = \phi(\mathbf{x}_T, T) + \sum_{j=0}^{N}\tilde{q}(\mathbf{x}_t, \mathbf{u}_t, \epsilon_t, t)\Delta t \tag{32}$$

We get that (26) becomes the iterative update rule:

$$\mathbf{u}_j^* = \mathbf{u}_j + \boldsymbol{\mathcal{H}}^{-1}\boldsymbol{\mathcal{G}}\left(\mathbb{E}_{q_{\mathbf{u}}^{\nu}}\left[\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau)\right)\frac{\epsilon_j}{\sqrt{\Delta t}}}{\mathbb{E}_{q_{\mathbf{u}}^{\nu}}\left[\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau)\right)\right]}\right]\right) \tag{33}$$

and then the term inside the parenthesis is approximated as:

$$\sum_{k=1}^{K}\left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_k)\right)\frac{\epsilon_{j,k}}{\sqrt{\Delta t}}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_k)\right)}\right) \tag{34}$$

where each trajectory τ_k is drawn from the sampling dynamics (29). The iterative update rule can be seen as computing the new control based on a reward weighted average over trajectories. Note that if G_c and B_c are multiples of each other with $\gamma \mathbf{B}_c = \mathbf{G}_c$, then $\boldsymbol{\mathcal{H}}^{-1}\boldsymbol{\mathcal{G}} = \gamma$.

III. MODEL PREDICTIVE CONTROL ALGORITHM

Equations (33) and (34) provide an iterative update law which we can apply in a model predictive control setting. In this setting optimization takes place on the fly: the trajectory is optimized, then a single control input is executed before re-optimization occurs. Since the derivation of path integral control provided here gives a formula for optimizing the entire sequence of controls, as opposed to just the current time instance, we can re-use the un-executed portion of the control sequence to warm start the optimization. This is crucial for the performance of the algorithm since, for a complex system operating at a reasonable control frequency, it is typically only possible to perform a small number of iterations every timestep.

The other key aspect of real-time implementation is that the bulk of the computation involved in the path integral control update law can be done in parallel. We take advantage of this by implementing the sampling step on a Graphics Processing Unit (GPU). Parallel trajectory sampling using a GPU is an efficient operation, which takes less than 15 millisecond in our implementation, even when sampling thousands of trajectories from complex non-linear dynamics. The description of the model predictive path integral control (MPPI) algorithm is given in Algorithm 1. The simplest approach to running Algorithm 1 on a GPU is to simply parallelize the sampling for-loop using one thread per sample. However, since the vehicle dynamics are fairly complex, we obtained additional speed up by using multiple threads per sample and parallelizing the dynamics predictions as well.

Algorithm 1: Model Predictive Path Integral Control

Given: K: Number of samples;
N: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, ...\mathbf{u}_{N-1})$: Initial control sequence;
$\Delta t, \mathbf{x}_{t_0}, \mathbf{f}, \mathbf{G}, \mathbf{B}, \mathbf{B}_E$: System/sampling dynamics;
$\phi, q, \mathbf{R}, \lambda$: Cost parameters;
$\mathbf{u}_{\text{init}}$: Value to initialize new controls to;

while *task not completed* **do**
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} = \mathbf{x}_{t_0}$;
    for $i \leftarrow 1$ **to** $N-1$ **do**
      $\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{f} + \mathbf{G}\mathbf{u}_i)\Delta t + \mathbf{B}_E \epsilon_{i,k}\sqrt{\Delta t}$;
      $\tilde{S}(\tau_k) = \tilde{S}(\tau_k) + \tilde{q}(\mathbf{x}_i, \mathbf{u}_i, \epsilon_{i,k}, t_i)$;
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{u}_i = \mathbf{u}_i + \boldsymbol{\mathcal{H}}^{-1}\boldsymbol{\mathcal{G}}\left[\sum_{k=1}^{K}\left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_k)\right)\frac{\epsilon_{j,k}}{\sqrt{\Delta t}}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_k)\right)}\right)\right]$;
  send to actuators$(\mathbf{u}_0)$;
  for $i \leftarrow 0$ **to** $N-2$ **do**
    $\mathbf{u}_i = \mathbf{u}_{i+1}$;
  $\mathbf{u}_{N-1} = \mathbf{u}_{\text{init}}$
  Update the current state after receiving feedback;
  check for task completion;

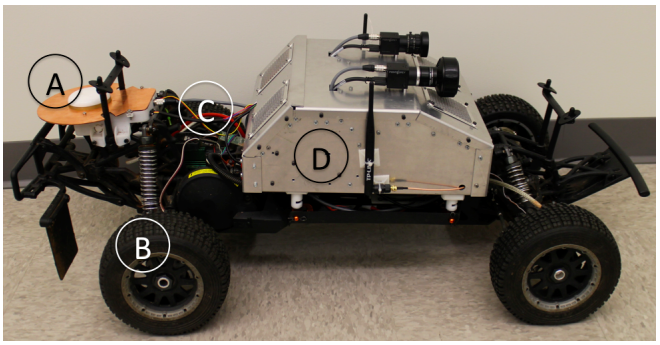

Fig. 1. AutoRally vehicle with protective body removed. A: IMU and GPS enclosure B: Hall effect wheel speed sensors C: High performance electric motor D: Rugged computer and battery enclosure

IV. AUTONOMOUS DRIVING SYSTEM

The ability of MPPI to handle complex non-linear dynamics and costs on the fly make it an appealing candidate for autonomously controlling a vehicle near its friction limits. In order to test the controller in a realistic, off-road environment we applied it to the Auto-Rally vehicle developed at the Georgia Institute of Technology in an aggressive driving scenario.

A. Fifth-Scale Auto-Rally Platform

The Auto-Rally vehicle is based on a 1/5 scale remote control vehicle equipped with an electric motor. It has been modified to host onboard an ASUS Mini-ITX motherboard with an Intel quad-core i7 processor, 16GB RAM, 2 SSDs, an Nvidia GTX-750ti graphics card, and a 222Wh battery. Additionally it has been equipped with 2 forward facing cameras, a Lord Microstrain 3DM-GX4-25 IMU, an RTK corrected GPS receiver, and hall effect wheel speed sensors. All computational components have been housed in a rugged aluminum enclosure able to withstand violent vehicle rollovers with no damage to internal components. The GPS and IMU are housed in a separate protective enclosure at the rear of the vehicle (Fig. 1). The total platform is approximately 22 kg and measures 0.9 meters from front to back. This platform allows fully self-contained testing of all algorithms, with no reliance on external position systems or computation beyond a GPS receiver. The onboard GPU allows the MPPI algorithm to control the vehicle in real time. Our implementation can sample over 2500, 2.5 second long trajectories in under 1/60 of a second, allowing state feedback from the vehicle to enter the optimization at 60Hz.

B. State Estimation

In order to accurately determine the vehicle state to feed back into the MPPI controller, the IMU and GPS measure-

ments are combined and pose, velocity, and bias estimates are obtained at each time step. To perform this optimization and estimation, a factor graph is constructed with successive measurements and iteratively optimized using the software package GTSAM and iSAM2 [21]. In order to keep computational loads low while maintaining high accuracy, this graph only optimizes for state nodes at 10Hz, corresponding to GPS measurements [22]. To obtain a state estimate at 200 Hz, IMU measurements are integrated to interpolate the 10Hz smoothed position to 200 Hz.

C. Data Driven System ID

In order to deploy the MPPI algorithm, an approximate model of the system dynamics is required. We opted for an approach which utilized information from physics based analysis and tools from machine learning in order to perform system identification. Analytic vehicle models were used to identify the important non-linearities in the dynamics, these were then used to construct 24 different basis functions, which we denote $\phi(\mathbf{x})$. We then fit a (bayesian) linear model using these basis functions. The result is that each element of the passive dynamics takes the form $f_i(\mathbf{x}) \sim \mathcal{N}(\mu_i(\mathbf{x}), \sigma_i^2(\mathbf{x}))$ where:

$$\mu_i(\mathbf{x}) = \frac{1}{\sigma_n^2}\phi(\tilde{\mathbf{x}})^{\mathrm{T}} A_i^{-1} \Phi(\tilde{\mathbf{X}}) Y_i, \quad \sigma_i^2(\tilde{\mathbf{x}}) = \phi(\tilde{\mathbf{x}})^{\mathrm{T}} A_i^{-1} \phi(\tilde{\mathbf{x}})$$

Where the matrix $A_i = \frac{1}{\sigma_n^2}\Phi(\tilde{\mathbf{x}})\Phi(\tilde{\mathbf{x}})^{\mathrm{T}} + \Sigma_p^{-1}$. In these equations $\tilde{\mathbf{x}}$ is $\mathbf{x} - \bar{\mathbf{x}}$ where $\bar{\mathbf{x}}$ is the mean of the training data. $\Phi(\tilde{\mathbf{X}})$ is an $n \times d$ matrix representing the basis functions for all elements of the training set which has size d. Σ_p defines the prior over the training data and σ_n is the estimated noise due to observation error. The state space $\mathbf{x}$ is composed

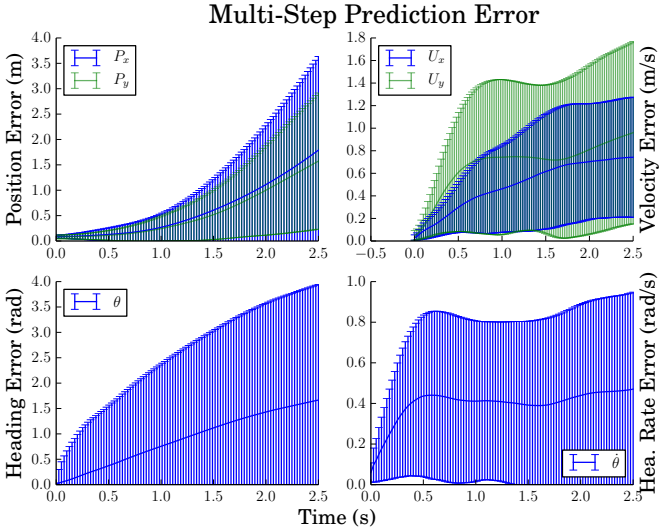

Fig. 2. Mean absolute error and standard deviation of the multi-step prediction as a function of time.

of global position in the world frame: p_x, p_y, the heading and roll angle of the vehicle: θ, ψ, and the longitudinal and lateral velocity of the vehicle in body frame U_x, U_y. We also include $\dot{\theta}$ which is the derivative of the heading angle. The stochastic nature of the algorithm can result in jittery control inputs, so we found it helpful to smooth the controls via the equation: $\dot{\mathbf{u}} = k(\mathbf{u}_{des} - \mathbf{u})$. Therefore the MPPI algorithm actually considers $\mathbf{u}_{des}$ as the control input of the system, and considers $\mathbf{u}$ as a state variable. It is only necessary to learn the dynamics for ψ, U_x, U_y, and $\dot{\theta}$ since the others are kinematically obvious.

We set $k = 10$ and $\mathbf{B}_c = \frac{1}{100} I_{2\times 2}$. The sampling noise was set to 25 times $\mathbf{B}_c$. We also experimented with including noise from the passive dynamics in the sampling procedure by using the variance estimate of the model. This corresponds to setting $\mathbf{B}_a(\mathbf{x}_t)$ equal to a diagonal matrix with $\mathbf{B}_a(\mathbf{x}_t)_{i,i} = \sigma_i(\mathbf{x}_t)$. This results in slightly less aggressive control inputs compared with setting $\mathbf{B}_a$ to zero, however it was unclear what the overall effect on the system performance was. During our experiments we include this extra noise in the sampling.

This model provides a relatively fast and accurate method for sampling trajectories in order to estimate (34). However, the modeling error is significant, especially after 1.5 seconds of simulation, so the MPPI algorithm has to be robust to this error in order to achieve good performance. Fig. 2 provides the mean absolute error of the prediction after 0.5, 1.0, 1.5, 2.0, and 2.5 seconds of simulation.

V. EXPERIMENTS

We gave the MPPI algorithm the task of navigating the AutoRally platform around a roughly ellipsoid track (see Fig. (3) for the track dimensions and Fig. (5) for a image of the track) with a maximum outer diameter of 30 meters and a uniform 3 meter wide driving surface. We tested the controller first with a moderate target speed of (6 m/s) and then at a higher target speed of (8 m/s). We consider the aggressive regime of the state space to be a target velocity which is above the friction limit of the vehicle while cornering on the test track. This was empirically determined to be between 5-6 m/s with the actual limit varying due to changes in the track conditions.

A. Cost Formulation

We formulated the the task of driving around the track as a finite-horizon optimal control problem in order to deploy the MPPI algorithm. The cost function consists of three main parts: (1) A cost for staying on the track, (2) A cost for achieving a desired velocity, and (3) A control cost. In addition to the track cost, if a given trajectory leaves the track then the dynamics are set to zero and the car remains in its current location for the rest of the simulation. A cost on the sideslip angle was also added which improved the stability of the vehicle. The final cost is formulated as:

$$2.5\left(V_{des} - V\right)^2 + 50.0h(p_x, p_y)^2 + 40.0C + 10.0\|\mathbf{u}\|^2 \quad (35)$$

Here V_{des} is the desired speed of the vehicle, and $V = \sqrt{U_x^2 + U_y^2}$. The function $h(p_x, p_y)$ is a function which returns -1 if the vehicle is on the inside boundary of the track (or off the track inside), 1 if the vehicle is on the outside boundary of the track (or off the track outside) and a value in $[-1, 1]$ if the vehicle is on the track with 0 representing

the middle, C is an indicator variable which turns on if the side-slip angle exceeds .25 radians (14.33 degrees). The coefficients which determine the relative weights of the cost components were set manually through experimentation.

The negative exponentiation of the cost-to-go can be numerically problematic if all of the cost-to-go terms are large, to combat this we normalize the cost according to the current cost-to-go of the nominal trajectory. This corresponds to multiplying with $\frac{\alpha}{\tilde{S}_n}$, where $\tilde{S}_n$ is the current cost-to-go, instead of $\frac{1}{\lambda}$. We set $\alpha = 10$.

B. 6 m/s Target Velocity

For navigating the track at a moderate velocity we set the speed cost in (35) to 6 m/s. Although we chose this as the moderate speed target, it is important to note that this is still a challenging control task since cornering at 6 m/s is beyond the friction limits of the vehicle. For instance, a constant speed PID controller following a pre-specified path of waypoints will start to slip and lose control around corners at 5 m/s. The MPPI controller enters turns at a little under

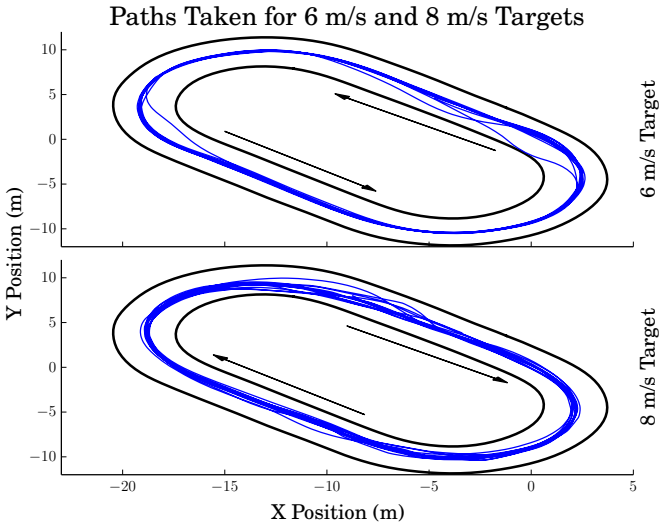

Fig. 3. Top: Path of the vehicles movement at the 6 m/s setting, Bottom: Paths at 8 m/s.

5 m/s and then accelerates out of them while sliding a small amount (indicated by the non-zero lateral velocity). Fig. 4 shows the velocity profile of the vehicle during the trial, and Fig. 3 shows the path of the vehicle during a 2 minute long run. There are a couple of occasions where the vehicle's path diverges from the rest of the trajectories. In these scenarios the car applies throttle while the back wheels have broken lose from the ground. This causes the tires to spin instead of providing forward acceleration. The result is that the vehicle over-steers around the corner. This behaviour is caused by a mismatch between the learned and actual dynamics of the vehicle. Despite these disturbances, the controller is quickly able to re-converge on a trajectory and move the car back towards the center of the track.

C. 8 m/s Target Velocity

For a high target speed we chose 8 m/s. For this size of track it is only possible for the vehicle to be at this

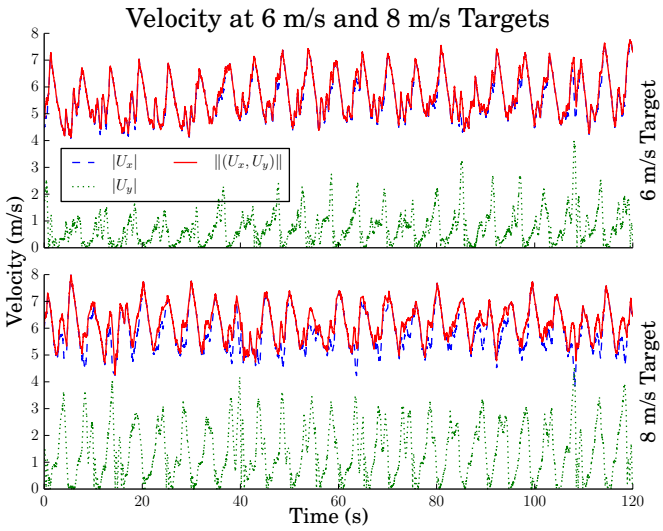

Fig. 4. Velocity profile for Top: 6 m/s target trial and Bottom: 8 m/s target velocity.

speed for a small amount of time, if at all, before it has to start slowing down to maneuver the corners. There is

TABLE I

PERFORMANCE METRICS FOR 6 M/S AND 8 M/S TEST RUNS.

Performance Metric	6 m/s target	8 m/s target
Average Speed	5.67 m/s	6.15 m/s
Top Speed	7.75 m/s	7.98 m/s
Average Lap Time	11.26s	10.04s
Maximum Side-Slip Angle	43.99°	64.83°

a significant difference, both qualitative and quantitative, between this setting and the 6 m/s setting. Table I highlights the quantitative difference in performance metrics between the two trials. The vehicle takes more velocity into the turns at slightly above 5 m/s, and then makes a sharp turn while sliding a significant amount. The amount of sliding is indicated by the body frame lateral velocity (i.e. the velocity perpendicular to the forward facing direction of the car), during the 8 m/s trial run this velocity regularly exceeds 2.5 m/s during cornering and reaches as high as 4 m/s (approx. 9 mph) during some turns.

VI. CONCLUSIONS

In this work we have derived a sampling based MPC algorithm and experimentally verified it on a challenging real system. The MPPI algorithm relies on a new derivation of path integral control which is based on relative entropy minimization. The benefits of this approach are that it explicitly provides a formula for the controls over the entire time horizon, as opposed to the first timestep only, and it relaxes the usual condition between control authority and noise required in path integral control. In addition to this theoretical contribution, we've given a demonstration of the algorithm in an aggressive autonomous driving scenario using a fifth-scale Auto-Rally vehicle. The MPPI controller is able to maneuver the vehicle around the track, even when the desired speed is set beyond the friction limits of the vehicle,

and it is even able to perform controlled power slides around the corners (Fig. 5).

These behaviors highlight the advantages of this approach over the hierarchical and traditional optimal control approaches. In the hierarchical case the path planner has very limited knowledge of vehicle/track dynamics and has to either be conservative or use expert knowledge imparted by the control designer to select feasible trajectories. And, unlike previous optimal control approaches which perform most of the computation offline, the MPPI algorithm is able to generate completely new behaviors on the fly. This enables the controller to push the vehicle to its operational limits using only an approximate model of the system and a simple cost function.

Finally we would like to conclude by discussing the advantages of the proposed framework when compared to standard optimal control methodologies using Dynamic Programming. In particular, the derivation of the optimal control is performed without the requirement to make connections with the Hamilton-Jacobi-Bellman PDE, as is typically required in stochastic optimal control. Therefore the minimization of the relative entropy in (9) corresponds to an alternative stochastic optimal control formulation that goes beyond the standard HJB-PDE approach and Dynamic Programming.

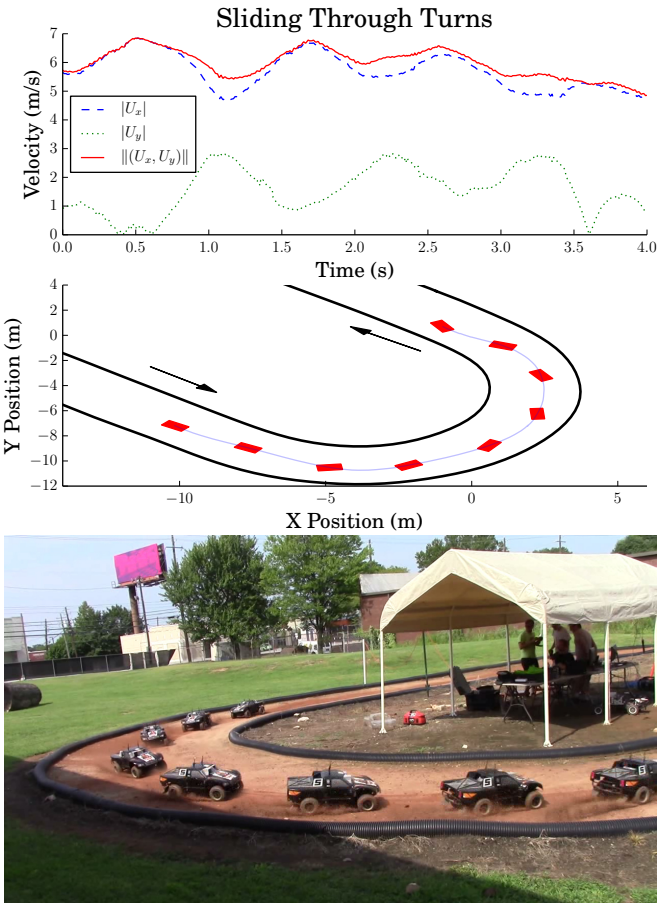

Fig. 5. Top: Velocity Profile of the Auto-Rally vehicle cornering. Middle: Path taken and heading progression. Bottom: Time lapse video of the maneuver.

REFERENCES

[1] Urmson *et al.*, "Tartan racing: A multi-modal approach to the darpa urban challenge," 2007.

[2] Thrun *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[3] E. Velenis and P. Tsiotras, "Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation," *Journal of Optimization Theory and Applications*, vol. 138, no. 2, pp. 275–296, 2008.

[4] K. L. Talvala, K. Kritayakirana, and J. C. Gerdes, "Pushing the limits: From lanekeeping to autonomous racing," *Annual Reviews in Control*, pp. 137 – 148, 2011.

[5] K. Kritayakirana and J. C. Gerdes, "Using the centre of percussion to design a steering controller for an autonomous race car," *Vehicle System Dynamics*, vol. 50, no. sup1, pp. 33–51, 2012.

[6] N. Keivan and G. Sibley, "Realtime simulation-in-the-loop control for agile ground vehicles," *Towards Autonomous Robotic Systems: 14th Annual Conference*, 2013.

[7] E. Velenis, P. Tsiotras, and J. Lu, "Modeling aggressive maneuvers on loose surfaces: The cases of trail-braking and pendulum-turn," in *Control Conference (ECC), 2007 European*. IEEE, 2007, pp. 1233–1240.

[8] M. Gerdts, S. Karrenberg, B. Müller-Beßler, and G. Stock, "Generating locally optimal trajectories for an automatically driven car," *Optimization and Engineering*, vol. 10, no. 4, pp. 439–463, 2009.

[9] P. Tsiotras and R. S. Diaz, "Real-time near-optimal feedback control of aggressive vehicle maneuvers," in *Optimization and optimal control in automotive systems*. Springer, 2014, pp. 109–129.

[10] H. J. Kappen, "Linear theory for control of nonlinear stochastic systems," *Phys Rev Lett*, vol. 95, p. 200201, 2005, journal Article United States.

[11] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *The Journal of Machine Learning Research*, vol. 9999, pp. 3137–3181, 2010.

[12] E. Rombokas, M. Malhotra, E. Theodorou, E. Todorov, and Y. Matsuoka, "Reinforcement learning and synergistic control of the act hand," *Mechatronics, IEEE/ASME Transactions on*, vol. 18, no. 99, pp. 569–577, 2013.

[13] V. Gómez, H. J. Kappen, J. Peters, and G. Neumann, "Policy search for path integral control," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I*, 2014, pp. 482–497.

[14] G. Williams, E. Rombokas, and T. Daniel, "Gpu based path integral control with learned dynamics," in *Neural Information Processing Systems, Autonomously Learning Robots Workshop*, 2014.

[15] V. Gómez, S. Thijssen, H. J. Kappen, S. Hailes, and A. Symington, "Real-time stochastic optimal control for multi-agent quadrotor swarms," *RSS Workshop, 2015. arXiv:1502.04548*, 2015.

[16] E. A. Theodorou and E. Todorov, "Relative entropy and free energy dualities: Connections to path integral and kl control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 1466–1473.

[17] E. A. Theodorou, "Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations," *Entropy*, vol. 17, no. 5, p. 3352, 2015.

[18] W. Zhang, H. Wang, C. Hartmann, M. Weber, and C. Schutte, "Applications of the cross-entropy method to importance sampling and optimal control of diffusions," *SIAM Journal on Scientific Computing*, vol. 36, no. 6, pp. A2654–A2672, 2014.

[19] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," in *Int. Conf. on Robotics Science and Systems (R:SS 2012)*.

[20] G. Williams, A. Aldrich, and E. Theodorou, "Model predictive path integral control using covariance variable importance sampling," preprint, http://arxiv.org/abs/1509.01149v2.

[21] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *International Journal of Robotics Research*, vol. 31, no. 2, SI, pp. 216–235, FEB 2012.

[22] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.