Robust Sampling Based Model Predictive Control with Sparse Objective Information

Grady Williams, Brian Goldfain, Paul Drews, Kamil Saigol, James M. Rehg, and Evangelos A. Theodorou
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology

***Abstract*—We present an algorithmic framework for stochastic model predictive control that is able to optimize non-linear systems with cost functions that have sparse, discontinuous gradient information. The proposed framework combines the benefits of sampling-based model predictive control with linearization-based trajectory optimization methods. The resulting algorithm consists of a novel utilization of Tube-based model predictive control. We demonstrate robust algorithmic performance on a variety of simulated tasks, and on a real-world fast autonomous driving task.**

I. Introduction

Autonomous robots are increasingly being asked to solve safety critical tasks in highly dynamic and non-linear environments. In order to competently operate in such environments, a robot needs to be able to plan and execute trajectories utilizing the full range of its dynamic capabilities, while ensuring that it achieves any relevant task objectives. Theoretically, the generation of flexible, safe, and high performance behaviors for non-linear systems can be achieved through an optimal control or stochastic optimal control framework. In optimal control, a high level cost function is specified, and then the generation of a trajectory and control plan is achieved by minimizing the cost with respect to the system dynamics. When a dynamics model is unknown a-priori, this approach is referred to as model-based reinforcement learning. This optimization based approach can be especially effective in a model predictive control (MPC) framework, where planning and execution are continuously interleaved.

Although optimal control theory provides an elegant mathematical framework for controlling robotic systems, and has many practical successes, specification of a cost or reward function is a non-trivial and time consuming problem [20]. This is because the solutions to optimal control problems are heavily constrained by the system dynamics, and are therefore very difficult to obtain. The result is that cost and constraint function specifications often become more about creating smooth cost functions with few local minima, as opposed to creating an easy to interpret encoding of a high-level behavior. For simple tasks, this is merely frustrating. However, in actual deployments of autonomous robots in complex environments, it could be crippling. In autonomous driving, for instance, it is impossible to test how a certain cost function would work in all of the scenarios an autonomous vehicle could encounter. Therefore, engineers designing robotic systems must have confidence that the representation of the robot's objective will lead to the desired behavior in all circumstances, without extensive tuning.

One possible solution to this challenging problem is to use gradient-free, sampling based optimal control methods, such as cross-entropy or path integral control [8, 21, 10]. Recently, these frameworks have been applied in MPC settings [25, 6, 4], where they have demonstrated the ability to control high-dimensional, non-linear systems. Since these methods do not require a gradient, they can theoretically utilize very simple encodings of tasks descriptions with sparse gradient information. For example, in this paper we consider cost functions encoded with weighted indicator functions:

$$\sum_{i=1}^{N} w_i \mathbb{1}_{C_i}(\mathbf{x}). \tag{1}$$

These types of functions have the advantage of clearly encoding whatever task is specified, and it is possible to compose many of them together since there are no gradients that can interfere with each other. However, the fact that the gradient of these functions is zero wherever it is defined, makes it difficult to use these cost functions with any type of gradient-based optimization.

In contrast to gradient-based optimization methods, sampling-based MPC can, in theory, handle cost functions of the form of Equation 1. In practice, unfortunately, when using cost functions with such sparse objective information, sampling based MPC methods are brittle and prone to failure in the face of unexpected disturbances and non-linear dynamics. The fundamental problem is that sampling based methods, while gradient free, are still iterative local search methods. This is simply because it is intractable to fully sample high

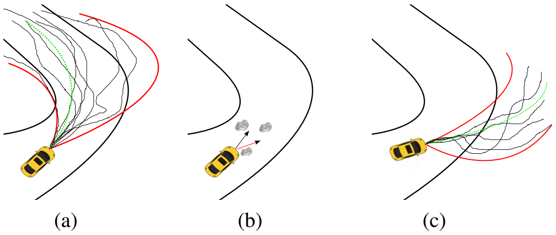

Fig. 1: Effect of disturbances on sampling based MPC. In (a), an autonomous vehicle has a good sampling distribution. In (b) the vehicle executes the control, but hits a disturbance, resulting in (c) the sampling distribution leads to high cost.

dimensional state spaces, and as a result such methods require a good initialization in order produce reliably good results. In an MPC setting, the initialization takes the form of a warm start, where if $(\mathbf{u}_0, \mathbf{u}_1, \dots \mathbf{u}_{T-1})$ is the current control solution, then $(\mathbf{u}_1, \mathbf{u}_2, \dots)$ will be used to initialize the next iteration. Implicit in this procedure is the assumption that the actual next state is close to the predicted next state. In the presence of disturbances, that assumption may fail. Figure 1 gives an example. Without a consistent gradient signal to lead the sampling distribution back to the low cost region, the sampling procedure may get stuck in a bad local minimum or diverge entirely.

In this paper, we develop a novel solution to the robustness problem that is inherent in sampling based MPC. Our key contribution is to augment a sampling based MPC method with an ancillary controller for disturbance rejection by utilizing Tube-MPC [16, 15]. Tube-MPC was originally developed as a way to guarantee robustness for constrained linear systems in the presence of disturbances, and was later extended to non-linear systems. The original version of non-linear Tube-MPC, which we utilize in this work, consists of two model predictive controllers. The first controller, termed the nominal controller, attempts to solve the primary optimal control problem for an idealized nominal state, and the second controller, called the ancillary controller, has the goal of rejecting disturbances in order to keep the actual system state close to the nominal state.

II. Related Work

Sampling-based approaches to motion planning and control have a long history in robotics. RRTs and variants thereof [12, 9] have emerged as the dominant frameworks for motion planning, and the winning robot in the 2005 DARPA Grand Challenge [22] used an on-line path planning method based on trajectory sampling. Motion planning methods sample in state or path space, and typically only produce kinematically feasible plans, which have to be executed with a low level controller. This can be a problematic paradigm when operating noisy systems near their dynamic limits, since failure may occur if a planned path is not feasible or if disturbances push the trajectory off the initially planned path. Within the motion planning literature, our work is most closely related to [14], which also endeavors to control a system state by keeping it within a tube. However, this method restricts behavior to a finite library of pre-generated maneuvers, and requires both the initial library generation and a stabilizing feedback controller to be pre-specified. In our method, we only need to be given a dynamics model and cost function.

Outside of motion planning, sampling based control algorithms can be directly derived from stochastic optimal control theory. For example, a number of sampling based methods have been derived using a bayesian approximate inference approach to stochastic optimal control [19, 13], path integral control theory [21, 8, 5, 25], and the cross-entropy method [4, 24, 10, 11]. Despite all of the success in these areas, on-line sampling of trajectories with un-stable, non-linear dynamics in the presence of disturbances remains a key problem, and is usually addressed via ad-hoc cost function tuning.

In this work, we make use of non-linear Tube-MPC to prevent the divergence of the importance sampling distribution in sampling-based MPC. In addition to [15], which used a second MPC as an ancillary controller, there have been several non-linear variants of Tube-MPC which utilize alternative methods to devise a non-linear ancillary controller [18, 7, 3]. In [17], the advantages and disadvantages of Tube-MPC are discussed, and several modifications to the original scheme are suggested, which we also utilize in our approach.

The purpose of these earlier works is fundamentally different than our goal in this paper. Previous Tube-MPC approaches aimed to improve performance or provide guarantees for traditional MPC methods (e.g methods that utilize gradient based optimization in order to stabilize a system or track a trajectory). This work is the first usage of Tube-MPC for solving general optimal control problems, with a sampling based nominal controller. Our goal with combining these methods is not to simply improve performance, but to enable the solution of entirely new classes of stochastic optimal control problems.

III. Preliminaries

We consider general discrete time non-linear systems of the form:

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t, \mathbf{u}_t + \epsilon_t) + \mathbf{w}_t, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^N$ is the state, and $\mathbf{u} \in \mathbb{R}^M$ is the control input. The term $\epsilon \in \mathcal{N}(0, \Sigma)$ is a disturbance directly on the control input which is a reasonable assumption for robotic systems where an outputted control signal is used as a set-point for a lower level controller. The term $\mathbf{w}$ is an external disturbance, which exists due to a combination of modeling error and purely stochastic or unobserved environmental effects. In this paper our goal is to optimize systems with running costs of the following form:

$$\mathcal{L}(\mathbf{x}, \mathbf{u}) = \mathcal{C}(\mathbf{x}) + \lambda \mathbf{u}^{\mathrm{T}} \Sigma^{-1} \mathbf{u}, \tag{3}$$

$$\mathcal{C}(\mathbf{x}) = k(\mathbf{x})^{\mathrm{T}} Q k(\mathbf{x}) + \sum_{i=1}^{N} w_i \mathbb{1}_{C_i}(\mathbf{x}). \tag{4}$$

Our assumption is that the control cost is inversely proportional to that control channels variance, so that very noisy inputs are cheap to control and vice versa. For the state dependent portion of the cost, the term $k(\mathbf{x})$ is a (preferably simple) state feature, Q is a positive definite weight matrix, w_i is a cost weighting, and $\mathbb{1}_C$ is the indicator function for the set C, which is 1 if $\mathbf{x} \in C$ and 0 otherwise.

The goal of the first portion of the cost is to encode some overarching directive to the robot (e.g. go a certain speed), and the second portion of the cost acts to encode constraint like objectives into the system, and has a zero (or undefined) gradient. Using this type of cost function has a variety of benefits: these costs are readily interpretable and easy to encode, even though they are technically soft constraints they act like hard constraints in the sense that all of the penalty is obtained immediately upon crossing the constraint boundary. However, unlike hard constraints, they have the additional benefit that the importance of different constraints can be delineated by setting different weights.

IV. Robust Sampling Based MPC

In this section we describe our robust sampling based MPC method, which is based on Tube-MPC. The linear version of Tube-MPC utilizes a nominal controller, a nominal state, an ancillary controller, and the actual system state. The nominal controller is able to select the initial nominal state (subject to it being nearby the actual system state) and the nominal solution, both of which are readily computable via the solution of a quadratic program. The ancillary controller then takes the form of a simple linear feedback gain, which maintains the actual state of the system in a tube around the nominal state solution.

In non-linear Tube-MPC, which is the basis for our approach, much of the convenience of the solution for the linear case is lost. However, the algorithmic structure and the end result remain the same. We consider the two systems:

$$\bar{\mathbf{x}}_{t+1} = \mathbf{F}(\bar{\mathbf{x}}, \bar{\mathbf{u}}), \tag{5}$$

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}, \mathbf{u} + \epsilon) + \mathbf{w}. \tag{6}$$

These systems are identical, except that one is disturbed via noise, and the other is disturbance free. The nominal controller then takes the form of a non-linear model predictive controller, which can consider general costs and constraints, and it computes a solution $\{(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \ldots \bar{\mathbf{x}}_{T-1}, \bar{\mathbf{x}}_T), (\bar{\mathbf{u}}_0, \bar{\mathbf{u}}_1, \ldots \bar{\mathbf{u}}_{T-1})\}$. The nominal system is allowed to ignore system disturbances, so we can have $\bar{\mathbf{x}}_0 \neq \mathbf{x}_0$, where $\mathbf{x}_0$ is the actual state of the system. The role of the ancillary controller is to then track the nominal system state. We implement the ancillary controller as a gradient based MPC method which solves a standard tracking problem.

Unlike the linear Tube-MPC case, the nominal controller does not consider the initial nominal state as an input variable, however in certain instances, the nominal state can be reset back to the actual state. As in the linear case, it can be shown that the actual system state stays within a tube centered about the idealized nominal state. However, the bound on the size of the tube is difficult to compute in practice, and in this study we are more concerned with demonstrating the practical ability of Tube-MPC to prevent the divergence of sampling based controllers. Therefore we do not concern ourselves with computing this bound. There are then 3 components of the Tube-MPC algorithm that we need: (1) a nominal controller, (2) a method for setting the nominal state, and (3) an ancillary controller. We use an information theoretic interpretation of model predictive path integral control (MPPI) [26], so we will hereon refer to our method as Tube-MPPI.

A. Nominal Controller - Model Predictive Path Integral

The nominal controller is required to be a sampling based method so that it can handle the types of sparse cost functions that we are interested in. We use an information theoretic interpretation of path integral control implemented in an MPC setting (MPPI). In MPPI, we consider stochastic trajectory optimization problems of the form:

$$U^* = \underset{U}{\operatorname{argmin}}\, \mathbb{E}_{\mathbb{Q}} \left[\phi(\mathbf{x}_T) + \sum_{t=0}^{T-1} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) \right], \tag{7}$$

Where ϕ is a terminal cost, and $\mathcal{L}$ a running cost in the form of (3). The term $\mathbb{Q}$ is the distribution corresponding to the dynamics: $\mathbf{F}(\mathbf{x}, \mathbf{u} + \epsilon)$. These dynamics consider sources of noise directly acting on the control input, but not sources due to modeling error or environmental disturbances. Note that we are actually optimizing with the assumption that there is some noise in the system, even though the nominal system is noise free. We do this because the assumption of noisy inputs is necessary in order to utilize sampling based methods derived from stochastic optimal control theory, and we do not consider it to be detrimental: since noise is present in the actual system it does not hurt to plan for it even if the theory does not explicitly require it.

In the information theoretic approach to MPPI, the trajectory optimization problem is transformed into a probability matching problem. Suppose that $U = \{\mathbf{u}_0, \mathbf{u}_1, \ldots \mathbf{u}_{T-1}\}$ is a sequence of commanded inputs, and then let V be the resulting sequence of perturbed inputs such that:

$$V = U + \mathcal{E}, \quad \mathcal{E} = \{\epsilon_0, \epsilon_1, \ldots \epsilon_T\} \tag{8}$$

With each $\epsilon_t \sim \mathcal{N}(0, \Sigma)$. Then, by using an information theoretic lower bound, it is possible to show that there exists an "optimal" distribution over controls, in the sense that trajectories sampled from that distribution have a lower expected cost than any other distribution. It can be shown [26] that this takes the form:

$$q^*(V) \propto \exp\left(-\frac{1}{\lambda} S(V)\right) p(V),$$

$$p(V) = \frac{1}{\left(\sqrt{2\pi \|\Sigma\|}\right)^{T-1}} \exp\left(\sum_{t=0}^{T-1} \mathbf{v}_t^{\mathrm{T}} \Sigma^{-1} \mathbf{v}\right),$$

$$S(V) = \phi(\mathbf{x}_T) + \sum_{t=0}^{T-1} \left(k(\mathbf{x})^{\mathrm{T}} Q k(\mathbf{x}) + \sum_{i=1}^{N} w_i \mathbb{1}_{C_i}(\mathbf{x}) \right).$$

The goal is to then minimize the KL-Divergence between the controlled and optimal distribution, which results in the formula:

$$U^* = \int q^*(V) V \mathrm{d}V, \tag{9}$$

where the optimal controls take the form of an expectation with respect to the optimal distribution $q^*(V)$. This equation is impossible to compute directly, but it can be approximated using an iterative importance sampling method where the $k+1$

iterate is related to the k_{th} iterate via:

$$U_{k+1} = U_k + \sum_{i=1}^{N} w(\mathcal{E}_i)\mathcal{E}_i, \tag{10}$$

$$w(\mathcal{E}) = \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}\left(S(U_k + \mathcal{E}_i) + \lambda\sum_{t=0}^{T-1}\mathbf{u}^{\mathrm{T}}\Sigma^{-1}\epsilon_t\right)\right). \tag{11}$$

where $\mathcal{E}_i$ is the disturbance sequence that generates the i_{th} trajectory sample out of a total of N samples. Notice how the negative exponentiation in the importance sampling weight enables the algorithm to remove trajectories with significantly higher cost than other samples from the solution. This is important for the kinds of cost functions that we are considering, since trajectories that do not trigger the indicator cost terms will be weighted much less than trajectories that do. Using a GPU, it is possible to parallelize the sampling, which makes it possible to run MPPI with expensive non-linear dynamics. Algorithm 1 describes in psuedo-code the MPPI algorithm. Control constraints are handled by augmenting the dynamics with an element-wise clamping function $g(u) = \max(\min(u_{max}, u), u_{min})$. Note that this procedure only changes the system dynamics, and therefore does not affect the convergence of the MPPI algorithm.

```
Algorithm 1: Nominal Controller (MPPI)
Parameters: $\mathbf{F}$: Transition Model;
$K, T$: Number of samples, timesteps;
$\Sigma, \phi, \mathcal{C}$: Cost functions/parameters;
while not done do
    $\bar{\mathbf{x}}_0 \leftarrow$ SetNominalState();
    for $k \leftarrow 0$ to $K - 1$ do
        $\mathbf{x} \leftarrow \mathbf{x}_0$;
        Sample $\mathcal{E}^k = \left(\epsilon_0^k \dots \epsilon_{T-1}^k\right)$, $\epsilon_t^k \in \mathcal{N}(0, \Sigma)$;
        for $t \leftarrow 1$ to $T$ do
            $\bar{\mathbf{x}} \leftarrow \mathbf{F}(\bar{\mathbf{x}}, g(\bar{\mathbf{u}}_{t-1} + \epsilon_{t-1}^k))$;
            $S_k \mathrel{+}= \mathcal{C}(\mathbf{x}_t) + \lambda\bar{\mathbf{u}}_{t-1}\Sigma^{-1}\epsilon_{t-1}^k$;
        $S_k \mathrel{+}= \phi(\mathbf{x})$;
    $\rho \leftarrow \min_k [S_k]$;
    for $k \leftarrow 1$ to $K$ do
        $\tilde{w}_k \leftarrow \exp\left(-\frac{1}{\lambda}(S_k - \rho)\right)$;
        $\eta \mathrel{+}= \tilde{w}_k$;
    for $t \leftarrow 0$ to $T - 1$ do
        $\bar{U} \leftarrow \bar{U} + \frac{1}{\eta}\sum_{k=1}^{K}\tilde{w}_k\mathcal{E}^k$;
    $\bar{X} \leftarrow$ Simulate$(g(\bar{U}))$;
    PublishSolution$(g(\bar{U}), \bar{X})$;
    for $t \leftarrow 1$ to $T - 1$ do
        $\bar{\mathbf{u}}_{t-1} \leftarrow \bar{\mathbf{u}}_t$;
```

One of the keys to MPPI, as well as other sampling based methods, is re-using the left-over portion of the previously optimized control sequence to warm-start the optimization at the next time-step. This enables the method to run on-line, but also makes it vulnerable to catastrophic failures if there are large disturbances coupled with sparse cost information. This is because the planned control sequence, $(\mathbf{u}_1, \mathbf{u}_2, \dots \mathbf{u}_T)$, is assumed to be near optimal for initial conditions drawn from $\hat{\mathbf{x}} \sim \mathbf{F}(\mathbf{x}_0, \mathbf{u}_0 + \epsilon)$, but the actual next state is drawn from $\mathbf{x} \sim \mathbf{F}(\mathbf{x}_0, \mathbf{u}_0 + \epsilon) + \mathbf{w}$. Even in linear systems, a small change in the initial condition can lead to a large change in resulting behavior. This means that if $\mathbf{w}$ pushes $\mathbf{x}$ into a region where the distribution induced by $\mathbf{F}(\mathbf{x}, \mathbf{u}{+}\epsilon)$ has small or zero probability mass, then the planned control sequence may result in a much different state sequence than anticipated. If there is a consistent gradient signal to follow back to low cost regions, then the algorithm can recover. But in our case, this gradient signal does not exist, and the algorithm can easily become stuck in local minima or diverge. Secton V-A demonstrates how this can happen even for a simple linear system.

B. Setting the Nominal State

The nominal controller controls the state of the idealized noise free nominal system, termed the *nominal state*. In the original description of non-linear Tube-MPC, the nominal state is initially set equal to the actual state, and then it is simulated forward without ever receiving feedback from the actual system. This scheme has two primary drawbacks, the first being that the algorithm is completely reliant on the tracking ability of the ancillary controller. In cases where the ancillary controller fails, the nominal state and actual state will diverge resulting in a failure of the overall control scheme. The second issue is that most disturbances are not catastrophic to the nominal controller, and in those cases it is preferable to let feedback enter the nominal controller in order to re-plan from the actual system state. In some cases, disturbances can even be beneficial. If the fortunate situation occurs where a disturbance improves the state, it should be taken advantage of, not rejected.

In [17] a modification to Tube-MPC is suggested whereby two copies of the nominal controller are run, one from the nominal state and one from the actual system state. If the nominal controller finds a better solution using the actual state of the system, then the superior solution is used and the nominal state is reset back to the actual system state before moving onto the next time-step. We propose a similar, albeit more relaxed version of this mechanism, where we accept the solution from the nominal controller using the actual system state as long as the cost is less than the nominal state solution plus some threshold. The threshold is set as follows: let $\{C_{i_0}, C_{i_1}, \dots C_{i_M}\}$ denote the sets of constraints that are considered safety critical, then the minimum of $\{w_{i_0}, w_{i_1} \dots w_{i_M}\}$ is set as the threshold. This mechanism ensures that a disturbance can never push the solution of the nominal controller into a constraint region. The procedure of accepting or rejecting the solution and setting the nominal state is shown in Alg. 2.

```
Algorithm 2: Nominal State Selection
Input: $\bar{\mathbf{x}}$: Current nominal state;
$\mathbf{x}$: Current (actual) state;
$K$: Threshold for accepting solution from actual state;
$\bar{U}, \bar{X} \leftarrow$ MPPI($\bar{\mathbf{x}}$);
$U, X \leftarrow$ MPPI($\mathbf{x}$);
if $S(U) \leq S(\bar{U}) + K$ then
    $\bar{\mathbf{x}} \leftarrow \mathbf{x}$ ;
    $\bar{U}, \bar{X} \leftarrow U, X$;
return $\bar{\mathbf{x}}, \bar{U}, \bar{X}$;
```

C. Ancillary Controller - iLQG

The last component of the Tube-MPPI controller is an ancillary controller which solves a tracking problem in order to keep the actual system state within a tube centered about the nominal state. This is a standard tracking problem, where there is a small initial error and a quadratic cost, and there are numerous effective solutions. We elected to used iterative linear quadratic gaussian control iLQG (as in [23]) as the ancillary controller, and found that it provided good performance at a relatively small computational cost.

D. Implementation Details and CPU/GPU Utilization

In our real-time implementation, which we used for the autonomous vehicle system, the two MPPI iterations run together in a single loop where the bulk of the computation is off-loaded to a GPU. Each instantiation of MPPI samples 1200 2 second long trajectories with a control frequency of 50 Hz, which requires over 100,000 queries of the non-linear system dynamics per control cycle. The nominal controller publishes solutions at a rate of 50 Hz. The ancillary controller runs asynchronously on a separate CPU thread, and performs optimization for the latest solution published by the nominal controller. The ancillary controller optimizes for a shorter time horizon (1 second), but runs at a faster frequency (100Hz).

V. Experimental Results

We tested the Tube-MPPI algorithm on: a simulated linear point mass system, a simulated helicopter landing task, and both a simulated and real-world autonomous racing task. Through-out these experiments we refer to 3 different experimental conditions for MPPI:

i) **Baseline-MPPI** refers to MPPI operating on a system where there is no additional disturbance beyond the control dependent noise assumed in the MPPI framework.
ii) **Disturbance-MPPI** refers to the normal MPPI algorithm operating on a system with additional disturbances besides the what has been assumed by the MPPI algorithm. Depending on the system, this additional noise takes the form of extra noisy control inputs or non-control dependent noise.
iii) **Tube-MPPI** refers to the algorithm described in Sec. IV operating on the same extra-noisy system as disturbance MPPI.

Note that the Baseline-MPPI method is impossible to implement in reality, since it requires a perfect description of the systems dynamics and noise distribution. We include it in the experiments in order to highlight the fundamental role that un-anticipated disturbances have on sampling based MPC.

A. Illustrative Example: Point Mass System

This illustrative example visually demonstrates the advantage of Tube-MPPI in terms of stabilizing the optimization for the nominal controller. Consider the simple 2-D double integrator system:

$$\mathbf{x}_{t+1} = \begin{pmatrix} I_2 & I_2\Delta t \\ 0 & I_2 \end{pmatrix} \mathbf{x}_t + \begin{pmatrix} 0 \\ I_2\Delta t \end{pmatrix} (\mathbf{u}_t + \epsilon_t) \,. \tag{12}$$

The goal is to move this system at a constant velocity while staying within a ring centered about the origin, this can be interpreted mathematically as:

$$\mathcal{C}(\mathbf{x}_t) = \left(\sqrt{v_x^2 + v_y^2} - v_{des}\right)^2 + 1000\left(\mathbb{1}_C(\mathbf{x})\right), \tag{13}$$

$$C = \{\mathbf{x} \mid 1.875 < \sqrt{x^2 + y^2} < 2.125\}. \tag{14}$$

The level of noise that MPPI assumes present is $\epsilon \in \mathcal{N}(0, \Sigma)$ with $\Sigma = I$. For Disturbance-MPPI and Tube-MPPI the actual noise present in the system is set ten times higher at $\tilde{\Sigma} = 10I$. Fig. 2 shows the accumulation of the warm-start trajectories for each condition. These trajectories are obtained by simulating the control sequence used to warm-start MPPI at each iteration from the new initial state of the system. This trajectory defines the mean of the sampling distribution (in the linear case), so it is essential that it lies in a good region of the state-space.

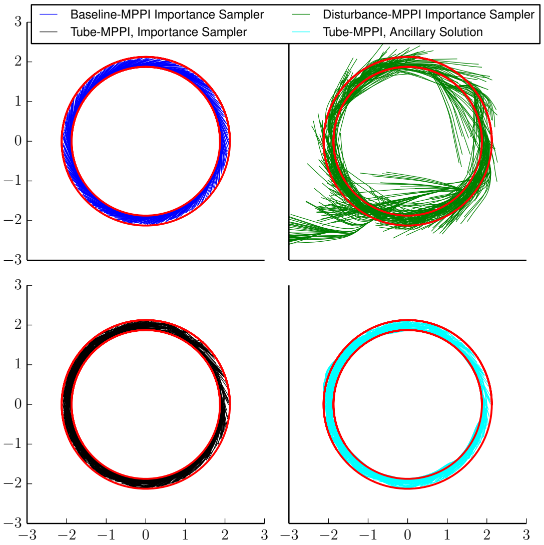

Fig. 2: Point mass system results of for Baseline-MPPI (top-left), Disturbance-MPPI (top-right), Tube-MPPI's nominal controller's importance sampling (bottom left) (c), and ancillary control plan (bottom right).

Baseline-MPPI performs perfectly, and the system state is always kept within C. With Disturbance-MPPI the increased noise in the system results in the state leaving C, and eventually diverging. The reason for this failure is that the system disturbances push the warm-start trajectories into poor regions of the state space, since sampling takes place locally around the warm-start trajectory, it becomes likely that no trajectory that stays within C is sampled. With Tube-MPPI, the nominal control plan is prevented from leaving the constraint set due to the condition for selecting the nominal state, this results in the importance sampling behaving similarly to the Baseline-MPPI condition, even with the increased system noise.

B. Simulated Helicopter Landing

In this simulated example we demonstrate the advantage of using weighted indicator costs as atomic elements for building objectives with a complex cost structure. We consider the task of landing a helicopter on a circular pad subject to Gaussian disturbances. For helicopter dynamics we use the non-linear model described in [2]. The state space for this helicopter is position (x, y, z), orientation (ϕ, θ, ψ), body frame velocity (v_x, v_y, v_z), and body frame angular velocity (p, q, r). The control inputs are collective thrust u_τ, roll rate u_p, pitch rate u_q, and yaw rate u_r. The cost function for the landing task then takes the form:

$$\mathcal{C}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}} Q \mathbf{x} + \sum_{i=1}^{8} w_i \mathbb{1}_{C_i},$$

$$C_1 = \{\mathbf{x} \mid (|\phi| > .15 \vee |\theta| > .1) \wedge z < -9.5\},$$

$$C_2 = \{\mathbf{x} \mid (\|(v_x, v_y, v_z)\| > 5 \vee v_z > 2.5) \wedge z > -8\},$$

$$C_3 = \{\mathbf{x} \mid \|(x, y)\| > 1.0 \wedge z > -8\},$$

$$C_4 = \{\mathbf{x} \mid x < -1.0\}, \quad C_5 = \{\|(v_x, v_y, v_z\| > 12\},$$

$$C_6 = \{\mathbf{x} \mid z > \max(-.5\|(x, y)\| - 7.5, -50) \wedge \|(x, y\| > 1\},$$

$$C_7 = \{\mathbf{x} \mid |\phi| + |\theta| > .33\},$$

$$C_8 = \{\mathbf{x} \mid z > -7.5 \wedge x \notin C_1 \wedge x \notin C_2 \wedge x \notin C_3\},$$

$$w_1 = w_2 = w_3 = w_4 = 10000, \quad w_5 = w_6 = 1000$$

$$w_7 = 100, \quad w_8 = -10000.$$

The first three terms direct the helicopter to land in the proper area with limits on the orientation and speed. The fourth term disallows the helicopter from over-shooting the landing area, the fifth and sixth terms prevent the helicopter from going too fast or using too aggressive of a combined roll and pitch angle, the seventh term directs the vehicle to stay above a certain glide-path, and the last term is a reward for successfully meeting all the landing criteria. The constraints are tightened to allow for some error in the final landing criteria, since we expect Tube-MPPI to keep the actual system close to the nominal state, but with a small amount of error.

Note that creating a cost function with a smooth gradient for this task, with either soft or hard constraints, would be *extremely challenging*! Many of the conditions have non-differentiable components (e.g. the max and norm operators) and composing a cost with eight different non-linear terms could easily result in local minima being created. In this case, specifying the cost function is easy and intuitive, and results in predictable behavior.

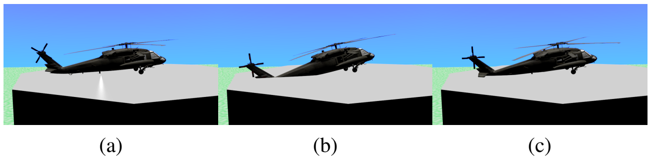

Fig. 3: Results of helicopter landing experiment for ((a)) Baseline-MPPI, ((b)) Disturbance-MPPI (worst trial by pitch magnitude), ((c)) Tube-MPPI (worst trial by pitch magnitude).

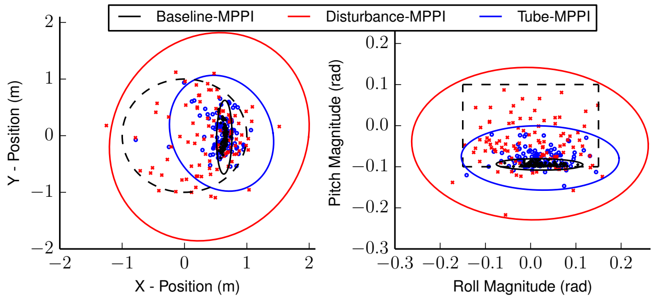

Fig. 4: Helicopter landing positions (left) and orientations (right) for 100 random trials of Baseline-MPPI, Disturbance-MPPI, and Tube-MPPI with large noise. Dashed lines are specified landing area, colored line indicate 3-sigma bounds for a Gaussian distribution fitted to the 100 trials.

MPPI assumes that there is noise in the control inputs with $\Sigma_{\mathbf{u}} = \text{Diag}(0.75, 0.125, 0.125, 0.125)$, and for Disturbance-MPPI and Tube-MPPI we inject additional noise into the system by increasing $\Sigma_{\mathbf{u}}$ and adding the additional disturbances for the velocities and orientation:

$$\Sigma_{\mathbf{u}} = (1.25)I_4, \quad \Sigma_{v_x, v_y, v_z} = (1.25)I_3, \quad \Sigma_{\phi,\theta,\psi} = (0.0125)I_3$$

Figure 4 shows the results over 100 randomized trials for Baseline, Disturbance, and Tube-MPPI. Baseline-MPPI performs perfectly, and never violates any constraints while landing the helicopter. Disturbance-MPPI ends most trials with a satisfying landing. However, there are several large outliers that significantly miss the target region, this would be catastrophic on an actual helicopter system. The distribution for Tube-MPPI closely mirrors that for Baseline-MPPI, but with a higher covariance. With Tube-MPPI there are not any outliers which miss the landing conditions by a significant amount.

Figure 3 shows the resulting orientations for the trials with the highest pitch magnitude, in the case of Tube-MPPI the worst case pitch is still within an acceptable landing envelope, whereas with Disturbance-MPPI the result would be the tail contacting the platform before the wheels touched down. Table I shows the mean, standard deviation, and worst case over the 100 trials for total distance from origin, roll angle, and pitch angle at touch-down.

C. Simulated Autonomous Racing

In this simulation experiment, we used a Gazebo simulation of 1/5 scale autonomous vehicles operating on a roughly

TABLE I: Helicopter Landing Statistics

	Distance	Roll	Pitch
MPPI - Small Noise	0.66 +/- 0.025	0.02 +/- 0.03	-0.09 +/- 0.00
MPPI - Large Noise	0.77 +/- 0.25	0.0 +/- 0.08	-0.04 +/- 0.05
Tube - MPPI	0.69 +/- 0.15	0.02 +/- 0.05	-0.07 +/- 0.02

TABLE II: Racing Simulation Statistics

	Avg. Lap Time	Max Speed	Max Slip
Disturbance -MPPI	11.87 +/- .47	5.22 +/- 0.06	0.04 +/- 0.04
Tuned - MPPI	8.33 +/- 1.05	7.53 +/- 0.04	0.09 +/- 0.15
Tube - MPPI	9.39 +/- 0.76	7.51 +/- 0.18	0.12 +/- 0.10

elliptical track [1], Fig. 5. In this simulation environment, we do not have access to the underlying model, so we fit one using a hybrid physics-neural networks approach. The state space of the vehicle is $\mathbf{x} = (x, y, \theta, r, v_x, v_y, \dot{\theta})$, and the model has the form: $\mathbf{F}(\mathbf{x}, \mathbf{u}) = \mathbf{x}_t + \left(W^{\mathrm{T}}\phi(\mathbf{x}, \mathbf{u}) + N(\mathbf{x}, \mathbf{u}; \theta)\right)\Delta t$ where W is a linear weight matrix, and $N(\mathbf{x}, \mathbf{u}; \theta)$ represents a neural network. This model is fit via a combination of linear regression and stochastic gradient descent, and there is a significant error between the learned model and the actual system dynamics. This error is the source of disturbances in this experiment.

Learning a model means that we cannot apply the Baseline-MPPI condition (since we cannot remove the extra disturbances), so instead we compare our novel controller to a version of MPPI that has been extensively tuned with a cost function for this track. We refer to this as **Tuned-MPPI**. This is an important comparison, as it quantifies our ability to use an intuitive indicator function-based cost structure to approach a level of performance only achievable previously through extensive hand-tuning. The cost function for Tube-MPPI was:

$$\mathcal{C}(\mathbf{x}) = \|v_x - v_x^{des}\|^2 + w_1 \mathbb{1}_{C_{\text{track}}}(\mathbf{x}) + w_2 \mathbb{1}_{C_{\text{slip}}}(\mathbf{x}) \tag{15}$$

$$C_{\text{slip}} = \left\{\mathbf{x} \mid \left\|\arctan^{-1}\left(\frac{v_y}{|v_x|}\right)\right\| < 1.25\right\} \tag{16}$$

$$w_1 = w_2 = 10000 \tag{17}$$

C_{track} is the set of points that lie inside the track boundaries. The first component of the cost tells the vehicle to try and achieve a desired velocity, the second component tells the vehicle to stay on the track, and the last term tells the vehicle to keep the slip angle below 1.25 radians (70 degrees). In the case of Tuned-MPPI the cost function takes the form:

$$\mathcal{C} = w_1 M(x,y) + w_2\|v_x - v_x^{des}\|^2 + w_3 \tan^{-1}\left(\frac{v_y}{v_x}\right)^2 \tag{18}$$

$$+ \beta^t \left(w_4 \mathbb{1}_{C_{\text{track}}}(\mathbf{x}) + w_5 \mathbb{1}_{C_{\text{slip}}}(\mathbf{x})\right) \tag{19}$$

$$w_1 = 100,\ w_2 = 4.25,\ w_3 = 250,\ w_4 = 10000 \tag{20}$$

$$w_5 = 10000,\ \beta = 0.9 \tag{21}$$

The first term $M(x, y)$ is a signed distance function for the set C_{track}. This term helps push the sampling distribution back towards the track if large disturbances are found. Note that even with this term it is still necessary to add a time decay on the hard-cost weighted indicator terms, which prioritizes avoiding trajectories that immediately violate constraints.

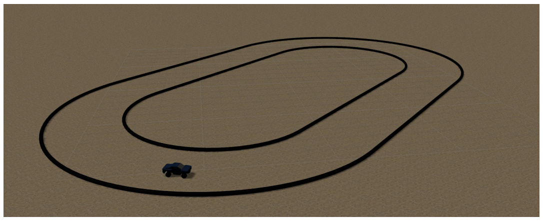

Fig. 5: Gazebo simulation environment

For each experimental condition, the target speed was gradually increased from 5 m/s until the algorithm could no longer consistently complete 100 laps while staying on the track. For Tuned-MPPI the maximum target speed was 8 m/s, and for Tube-MPPI this was 9 m/s. For Disturbance-MPPI there was a massive performance drop-off, with the maximum target speed only reaching 5 m/s. The performance statistics for each of the three trial conditions is shown in Table III. Both Tube-MPPI and Tuned-MPPI achieve top velocities slightly over 7.5m/s, and sub 10 second lap times. However, since Tube-MPPI optimizes with slightly tightened boundaries it takes a longer overall path around the track, which results in longer lap times.

D. 1/5 Scale Autonomous Racing Experiment

In order to validate the performance of the Tube-MPPI controller in the real-world, we tested the algorithm on the task of aggressive driving using the AutoRally 1/5 scale autonomous vehicle platform. This platform is approximately 1 meter long, weighs over 20 kilograms, and has a top speed over 20 m/s. Previous works have demonstrated that the MPPI controller (with tuned soft cost terms) is capable of navigating this type of vehicle around a simple elliptical track [25, 26], which we did our best to match in our simulation experiments. Our real-world experiments use the same type of vehicle as these prior works, but in a more challenging environment (Fig. 8). This track features a variety of different radius turns, and a long straight-away. An important detail of this track is that there are several areas where the boundaries for different segments of the track either touch or are very close to each other. This makes designing a smooth cost or constraint function based on a signed distance function difficult, since such a function would have local minima that would encourage the vehicle to drive over the track boundaries. However, using only weighted sums of indicator functions we obtain a very simple cost design based on a grid of binary values that represent the set of points on the track. The cost function for this task is the same as for the Gazebo simulation environment, where there's a term for speed, a term for staying on the track, and a term for avoiding excessive slip angle. The desired speed was set to 9 m/s, and we collected 12 laps around the test track, which is approximately 2 kilometers worth of driving data.

Figure 6 depicts the trajectory traces of the 12 trial laps around the track. This figure identifies one of the main benefits of using sparse indicator cost functions: since the vehicle is only penalized for leaving the track, it is free to use the entire track surface in order to achieve its primary goal of going fast. As a result, the position of the vehicle on the

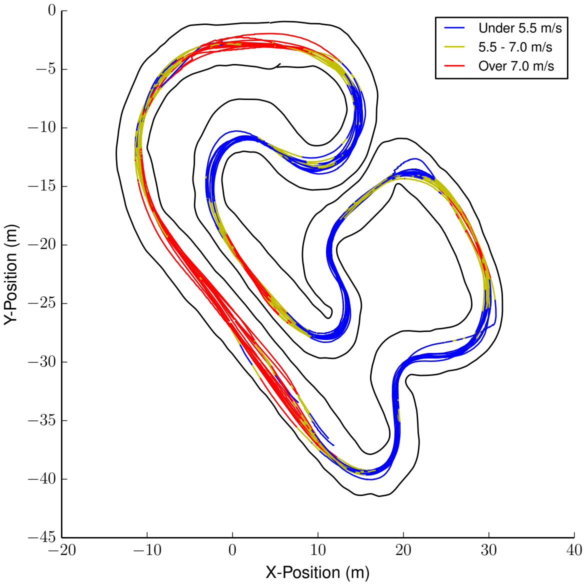

Fig. 6: Trajectory traces of test run with Tube-MPPI

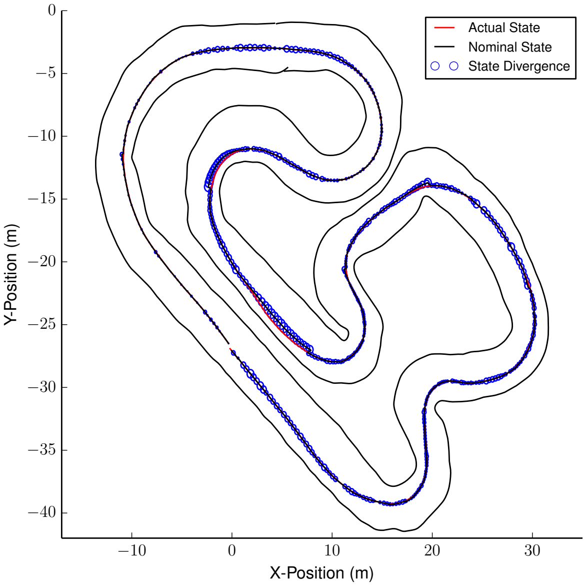

Fig. 7: Magnitude of the state divergence over a test lap.

track does not follow the center line, but significantly varies depending on the upcoming track geometry (note that the overall direction of travel is clockwise). A key component of the Tube-MPPI algorithm is the tracking performance of the ancillary controller. Figure 7 shows the magnitude of the state divergence as the vehicle navigates the track over the course of one lap. Overall, the magnitude of the positional state divergence stays relatively small compared to the overall track width. The mean state divergence for the lap shown is 14 centimeters, and the maximum is 47 centimeters over the trial. Footage from the trials is available in the video supplement.

TABLE III: Racing Experiment Statistics

	Avg. Lap Time	Max Speed	Max Slip
Tube - MPPI	32.02 +/- 7.27	8.52 +/- -0.26	0.88 +/- 0.48

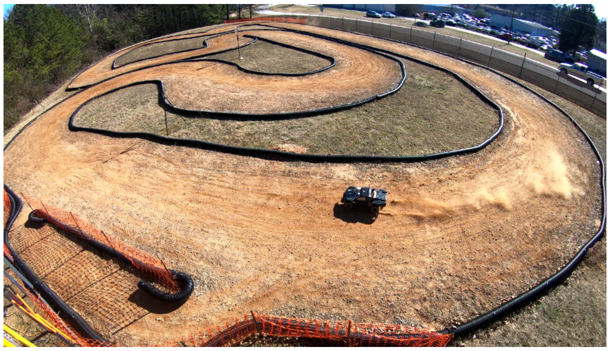

Fig. 8: Test track for 1/5 scale vehicle.

VI. Conclusion

We have proposed a novel robust sampling-based MPC framework based on a combination of model predictive path integral control and nonlinear Tube-MPC. The benefit of combining these methods is that the Tube-MPC procedure stabilizes the importance sampling distribution, which means that a gradient signal is not necessary to help the sampling distribution recover from large disturbances. This enables the use of very simple cost terms, such as weighted sums of indicator functions, in formulating the optimal control problem. Our method also takes advantage of the different hardware (CPU/GPU) requirements of the nominal and ancillary controllers. We carried out a variety of simulation experiments which demonstrate the advantage of our method in terms of solving the underlying problem of stabilizing the importance sampling distribution, thereby enabling the use of simple cost functions that are easy to compose into complex performance criteria.

The focus of this paper was on the practical ability of Tube-MPPI to optimize with simple, easily composed cost functions, and we therefore did not focus on the theoretical aspects of Tube-MPC. However, the theoretical guarantees of Tube-MPC are an important component of making the proposed approach a complete system. For instance, in the helicopter example we would like to be able to estimate the size of the tube in order to tighten the landing area constraints enough to ensure that the helicopter never misses a landing. Future work will focus on how to guarantee that a bound for the tube exists, and on how to incorporate the bound into the optimal control problem solved by the nominal controller.

Stochastic optimal control provides the most general and elegant mathematical framework for generating behaviors for autonomous systems, but generating cost functions that both describe the task at a high level and are easy to optimize with remains a key challenge. This paper is a step forward in easily specifying and solving general classes of stochastic optimal control problems.

Acknowledgements

This work was made possible by the ARO award W911NF-12-1-0377, the Georgia Tech Vertical Lift Center of Excellence (VLCROE), and the Qualcomm Innovation Fellowship.

References

[1] Autorally software package. https://github.com/AutoRally/autorally. Accessed: 2018-05-25.

[2] Pieter Abbeel, Varun Ganapathi, and Andrew Y Ng. Learning vehicular dynamics, with application to modeling helicopters. In *Advances in Neural Information Processing Systems*, pages 1–8, 2006.

[3] Vishnu Desaraju, Alexander Spitzer, and Nathan Michael. Experience-driven predictive control with robust constraint satisfaction under time-varying state uncertainty. In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017. doi: 10.15607/RSS.2017.XIII.067.

[4] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2786–2793. IEEE, 2017.

[5] Vicenç Gómez, Hilbert J Kappen, Jan Peters, and Gerhard Neumann. Policy search for path integral control. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 482–497. Springer, 2014.

[6] Vicenç Gómez, Sep Thijssen, Andrew Symington, Stephen Hailes, and Hilbert J Kappen. Real-time stochastic optimal control for multi-agent quadrotor systems. 2016.

[7] Sumeet Singh Marco Pavone Jean-Jacques and E Slotine. Tube-based mpc: a contraction theory approach.

[8] Hilbert J Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of statistical mechanics: theory and experiment*, 2005(11):P11011, 2005.

[9] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[10] Marin Kobilarov. Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871, 2012.

[11] Marin Kobilarov and Sergio Pellegrino. Trajectory planning for cubesat short-time-scale proximity operations. *Journal of Guidance, Control, and Dynamics*, 37(2):566–579, 2014.

[12] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[13] Rudolf Lioutikov, Alexandros Paraschos, Jan Peters, and Gerhard Neumann. Sample-based informationl-theoretic stochastic optimal control. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3896–3902. IEEE, 2014.

[14] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[15] David Q Mayne and Eric C Kerrigan. Tube-based robust nonlinear model predictive control1. *IFAC Proceedings Volumes*, 40(12):36–41, 2007.

[16] David Q Mayne, María M Seron, and SV Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[17] David Q Mayne, Eric C Kerrigan, and Paola Falugi. Robust model predictive control: advantages and disadvantages of tube-based methods. *IFAC Proceedings Volumes*, 44(1):191–196, 2011.

[18] Saša V Rakovic, Basil Kouvaritakis, Mark Cannon, Christos Panos, and Rolf Findeisen. Parameterized tube model predictive control. *IEEE Transactions on Automatic Control*, 57(11):2746–2761, 2012.

[19] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference.

[20] Stefan Schaal and Christopher G Atkeson. Learning control in robotics. *IEEE Robotics & Automation Magazine*, 17(2):20–29, 2010.

[21] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11(Nov):3137–3181, 2010.

[22] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[23] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference, 2005. Proceedings of the 2005*, pages 300–306. IEEE, 2005.

[24] Ari Weinstein and Michael L Littman. Open-loop planning in large-scale stochastic domains. In *AAAI*, 2013.

[25] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1433–1440. IEEE, 2016.

[26] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1714–1721. IEEE, 2017.